UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
COMPUTACIÓN

Trabajo de titulación previo a la obtención del título de:
Ingeniero en Ciencias de la Computación

TEMA:
ESTADO DEL ARTE UTILIZANDO MAPEO SISTEMÁTICO PARA LAS TÉCNICAS DE ANÁLISIS DE MALWARE EN ANDROID

AUTORES:
ESPINOSA MIRANDA IVAN EDUARDO
SANABRIA ALTAMIRANO MARCO STIVEN

TUTOR:
AGUAYO MORALES JOSÉ LUIS

Quito, Noviembre de 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Ivan Eduardo Espinosa Miranda, con documento de identificación N° 1722887062 y Marco Stiven Sanabria Altamirano con documento de identificación N° 1724227390, manifiestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: ESTADO DEL ARTE UTILIZANDO MAPEO SISTEMÁTICO PARA LAS TÉCNICAS DE ANÁLISIS DE MALWARE EN ANDROID, mismo que ha sido desarrollado para optar por el título de INGENIERO EN CIENCIAS DE LA COMPUTACIÓN en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento de hacer entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

.....

Iván Eduardo Espinosa Miranda
1722887062

.....

Marco Stiven Sanabria Altamirano
1724227390

Quito, noviembre del 2021

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Artículo Académico, con el tema: ESTADO DEL ARTE UTILIZANDO MAPEO SISTEMÁTICO PARA LAS TÉCNICAS DE ANÁLISIS DE MALWARE EN ANDROID, realizado por Ivan Eduardo Espinosa Miranda y Marco Stiven Sanabria Altamirano, obteniendo un producto que cumple con todos los requisitos por la Universidad Politécnica Salesiana, para ser considerado como trabajo final de titulación.

.....
Jose Luis Aguayo Morales
CI: 1709562597
Quito, Noviembre de 2021

Estado del arte utilizando mapeo sistemático para las técnicas de análisis de Malware en Android

1st Ivan Espinosa Miranda
iespinosam@est.ups.edu.ec

2nd Marco Sanabria Altamirano
msanabria@est.ups.edu.ec

3rd José Aguayo Morales
jaguayo@ups.edu.ec

***Resumen*—Los ataques de Malware enfocados en Android son un problema global, para prevenir los ataques a dispositivos móviles se han propuesto técnicas para estudiar los diversos tipos de programas maliciosos. En la presente investigación se formuló un estado del arte mediante mapeo sistemático, para extraer las revistas y conferencias de mayor relevancia, de 2017 al 2021. La revisión de literatura sistemática analizó las técnicas de Malware en Android, para construir un esquema de clasificación y ponerlos en la estructura de análisis estáticos, dinámicos e híbridos. La taxonomía reveló los métodos, herramientas y sistemas de aprendizaje más utilizados, por lo que se encontraron las mejoras y limitaciones de cada una de las técnicas de análisis. Los datos cuantitativos mostrados por la métrica de *acurracy* con la que se identificó un promedio con respecto a los resultados: análisis estático = 95.36%, análisis dinámico = 92.44% y análisis hibrido = 96.81%. Por tanto, estos resultados muestran que el análisis híbrido cuenta con ventajas sobre las técnicas de análisis estático o dinámico, reduciendo sus limitaciones y mejorando la detección de Malware en Android. Finalmente, la mejor opción para analizar el malware es el aprendizaje supervisado para clasificar las nuevas generaciones de Malware a partir de dos características: sus familias y la frecuencia de uso.**

***Palabras Clave*—Android, Malware, Mapeo Sistemático,Técnica Análisis**

***Abstract*—Malware attacks go to Android are a global problem, so that to prevent attacks on mobile devices, some techniques have been proposed to study the differents types of malicious programs. In this research, a state of the art was formulated using a systematic mapping, to extract the most relevant journals and conferences, from 2017 to 2021. The systematic literature review analyzed the Malware techniques in Android, in order to build a scheme of classification and put them into the structure of static, dynamic, or hybrid analyzes. The taxonomy revealed the methods, tools and learning systems more used, so that found the improvements and limitations of each of the analysis techniques. Quantitative data showed by the accuracy metric, were: static analysis = 95.36%, dynamic analysis = 92.44% and hybrid analysis = 96.81%. Therefore, these results show that the hybrid analysis has some advantages over the static or dynamic analysis technique, reducing its limitations and improving the detection of Malware in Android. Finally, the better option to analize malware is the supervised learning to classify the new generations of Malware from two features: their families and frequency used.**

***Keywords*—Android, Malware, Systematic Mapping, Analysis Technique**

I. Introducción

En los últimos años, es posible observar el incremento del uso de dispositivos Android volviéndose necesario para organizaciones e individuos, por ende, la cantidad de ataques por Malware ha ido en aumento, para proteger nuestros dispositivos Android debemos tener en cuenta que la superficie de ataque incluye mucho más que Malware, considerando que el 60% de los usuarios no posee una solución de seguridad móvil [1].

Existen personas que se dedican a desarrollar diferentes tipos de Malware para poder obtener información sin el consentimiento del usuario, estos programas que tienen como fin afectar el uso de dispositivos, vulnerando su seguridad, obtener el control del dispositivo del usuario y manejarlo de forma remota, lo que genera pérdidas para los usuarios [2].

Debido a la velocidad con la que el Malware ha ido en aumento en dispositivos Android, los investigadores han proporcionado varias soluciones de seguridad ante este crecimiento de infecciones, cubriendo el análisis de aplicaciones Android, ya sea con técnicas estáticas dinámicas e híbridas [3]. El análisis estático se encarga de analizar la información sobre archivos binarios sin ejecutar directamente el Malware, mientras que el análisis dinámico ejecuta Malware en un entorno controlado, como una máquina virtual (VM) o Sandbox para analizar el funcionamiento del Malware, el análisis híbrido utiliza los puntos fuertes del análisis estático y dinámico [4].

Para tener la visión clara del análisis de Malware de Android usando las técnicas estáticas, dinámicas e híbridas en los últimos cinco años; Realizamos un Mapeo Sistemático (del inglés SM, Systematic Mapping) con tres fases y aplicamos una Revisión Sistemática de la Lectura (del inglés SLR, Systematic Literature Revision) una vez identificado a fondo los estudios relacionados. A continuación, enumeramos las principales contribuciones de la investigación.

i) Realizamos el SM y SLR basándonos en la taxonomía de análisis de Malware en Android. ii) Este estudio divide el análisis de Malware en las tres técnicas estáticas, dinámicas e híbridas; basadas en métodos, herramientas y técnicas de aprendizaje. iii) A partir de entonces, al analizar la evidencia empírica, el estudio extrajo las características de mayor relevancia, así mismo las ventajas y limitaciones de cada técnica de análisis de Malware en Android. iv) Determinar las técnicas de análisis que se obtienen mejores resultados.

II. METODOLOGÍA

Para la investigación se utilizaron las metodologías de mapeo sistemático y revisión sistemática de la literatura. El

SM se apoya en la identificación, reconocimiento y clasificación de las técnicas de análisis de Malware en Android [5], SLR permite planificar, ejecutar y reportar la revisión sistemática [6], para obtener una mejor referencia acerca de investigaciones entre los años 2017 al 2021. El Malware se ha definido como: "cualquier código agregado, modificado o eliminado de un sistema de software con el fin de causar daño intencionalmente o subvertir la función deseada del sistema" [7].

Para la orientación de la metodología en el estado del arte se implementó SM y SLR, las cuales se organizan en las siguientes fases: En la fase 1 se definen los objetivos y el alcance de los criterios de selección. En la fase 2 se ejecuta la revisión, se definen los principales contenidos de la investigación y se dividen en dos pasos. En la fase 3 se reporta la revisión, los estudios seleccionados se clasifican en diferentes categorías, "así como informar la revisión mediante las pautas como objetivo de cumplir con el SM y SLR" [6].

A. Fase 1

Para la adquisición de artículos relacionados con las diferentes técnicas de análisis de Malware en Android, se aplicó el método PICOC utilizado para la descripción de los elementos de búsqueda en la tabla I.

TABLE I
DESARROLLO MÉTODO PICOC

Population (P): ¿Quién?	Malware en Android
Intervention (I): ¿Qué?, ¿Cómo?	Técnicas de análisis de malware
Comparison (C): ¿Con qué comparar?	Estudios que presenten las técnicas de análisis de malware en Android
Outcomes (O): ¿Qué se busca conseguir/mejorar?	Ventajas y limitaciones de las diferentes técnicas de análisis
Context (C): ¿En qué tipo de organización y bajo qué circunstancias?	Revisar estudios existentes sobre las técnicas de análisis de malware

Posteriormente se definieron los términos que se utilizarán en la creación de la cadena de búsqueda ver Tabla II. Se utilizaron expresiones booleanas como "AND" u "OR", que definieron nuestra cadena de búsqueda de la siguiente manera: ("Analysis Techniques" OR "dynamic analysis" OR "static analysis" OR "hybrid analysis") AND ("Malware" OR "malicious software" OR "malicious code") AND ("Android" OR "android system" OR "android devices" OR "android").

TABLE II
TÉRMINOS PARA REALIZACIÓN DE LA CADENA DE BÚSQUEDA

Términos	Términos Semejantes
Analysis Techniques	Dynamic analysis, static analysis, hybrid analysis
Malware	Malicious software, malicious code
Android	Android system, android devices

1) Criterios de selección de estudios: Para seleccionar los estudios relevantes, se usaron los criterios de inclusión y exclusión para SM y SLR que detallamos a continuación:

- **Criterios de exclusión:** Se excluyeron todos los artículos no escritos en inglés, menores a cinco hojas, duplicados en conferencias y revistas, lo artículos que este relacionados con cualquier otra aplicación que no sea Android.
- **Criterios de inclusión:** Se establecieron algunos criterios en el proceso de búsqueda en cuatro repositorios electrónicos: la selección de términos de búsqueda en los títulos y resúmenes se incluyó en los artículos (ver Tabla 2), se buscó investigaciones de evidencias empíricas usando herramientas acerca de técnicas de análisis de malware en Android.

B. Fase 2

1) Preguntas de Investigación: El objetivo principal de este estudio es actualizar el estado del arte de técnicas de análisis de Malware en Android, para lo cual se defineron preguntas de investigación para el Mapeo Sistemático (SMP#) y Revisión de la Literatura Sistemática (SLRP#) las cuales se describen a continuación:

- *SMP1: ¿Existe una taxonomía para el análisis de Malware en android?*
- *SMP2: ¿Cuál es la distribución de estudios con respecto a revistas y conferencias en los últimos cinco años?*
- *SLRP1: ¿Qué técnicas de aprendizaje se destacan en el análisis el Malware en las técnicas estáticas, dinámicas e híbridas?*
- *SLRP2: ¿Cuáles son los conjuntos de datos que se destacan en el análisis de Malware en Android?*
- *SLRP3: ¿Cuáles son las ventajas y limitaciones de las técnicas de análisis de Malware de Android?*
- *SLRP4: ¿Qué herramientas se destacan en el análisis de Malware en las técnicas estáticas, dinámicas e híbridas?*
- *SLRP5: ¿Qué medidas de rendimiento se utilizan para el análisis de Malware de Android?*
- *SLRP6: ¿Qué Métodos se destacan en el análisis de Malware en las técnicas estáticas, dinámicas e híbridas?*

2) Estrategias de Búsqueda: Se aplicó la cadena de búsqueda en cuatro repositorios véase Tabla III, previamente seleccionados y garantizando que existan suficientes artículos para realizar un estudio de SM y SLR. En el primer filtro se recopilaron 3826 estudios primarios. Para los repositorios Scopus, ACM y ScienceDirect se consideró el rango de búsqueda entre 2017 al 2021 a excepción del repositorio de la IEEE que se realizó búsquedas entre el 2018 al 2021 optimizando el patrón de búsqueda.

En el segundo filtro mediante el software EndNote x9, se filtró la información aplicando la cadena de búsqueda en título y resumen, por consecuencia la cantidad de estudios disminuyo notablemente a 182 estudios. En el tercer filtro, se descargaron los textos completos y se procedió a leer todos los artículos uno por uno, aplicando los criterios de exclusión e inclusión para finalmente seleccionar 66 estudios.

TABLE III
CADENA DE BÚSQUEDA EN CADA REPOSITORIO

Repositorio	Cadena de Búsqueda	Tipo Artículo	Filtro 1	Filtro 2	Filtro 3
IEEE	Título: “Analysis Techniques” OR “Malware” OR “Android” Resumen: “dynamic analysis” OR “static analysis” OR “hybrid analysis” AND “malicious software” OR “malicious code” AND “android system” OR “android devices” OR “android”	Revistas y Conferencia	1.529	52	17
SCOPUS	(“Analysis Techniques” OR “dynamic analysis” OR “static analysis” OR “hybrid analysis”) AND (“Malware” OR “malicious software” OR “malicious code”) AND (“Android” OR “android system” OR “android devices” OR “android”)	Revistas	887	88	36
ACM	Título: “Analysis Techniques” ,“Malware”, “Android” Resumen: “dynamic analysis” OR “static analysis” “hybrid analysis” ,“android”	Revistas	854	19	6
Science Direct	Analisys Techniques OR static analysis OR dynamic analysis OR hybrid analysis AND Malware OR malicious code AND Android OR android	Revistas	555	23	7
		TOTAL	3.826	182	66

C. Fase 3

La extracción de datos se logra mediante un diseño de formularios como en la Figura 1 , para extraer con precisión la información de los artículos encontrados en pasos anteriores. De acuerdo con los datos obtenidos en los formularios, se lograría responder a las preguntas de investigación para la SM y SLR. Una vez extraída la información de los estudios, el siguiente paso es la síntesis de datos, que se podrá resumir a la taxonomía (ver Figura 1) y comparar en la extracción de datos, que nos concederá la convicción para lograr contestar las preguntas del estado del arte.

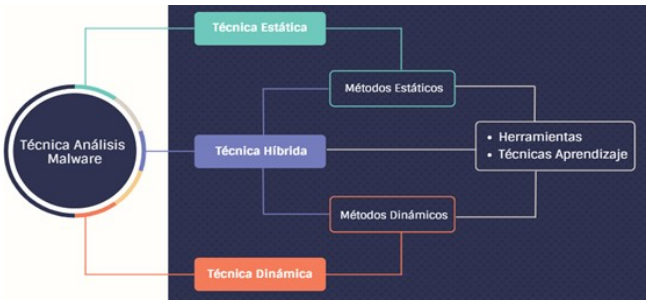


Fig. 1. Taxonomía de las técnicas de análisis de Malware en Android.

1) Conjuntos de Datos: En los estudios revisados por la literatura, los investigadores recurren a los conjuntos de datos en donde existe gran variedad de Malware clasificado según su familia a la que pertenece. La Figura 2 muestra la frecuencia de uso de los DataSet, que son utilizados para llevar a cabo el análisis ya sea estático, dinámico o híbrido. Los DataSet más recurrentes para las investigaciones son los siguientes:

- **Google Play (GP)** contiene más de 9.660 aplicaciones, el 71% de datos es Malware nuevo y 29% Malware conocido y dividido en 23 familias. La ventaja de GP es que por su intervención de la tienda Google Play Store su información es actualizada.
- **Drebin (DR)** es parte de un proyecto de recolección de Malware entre los años 2010 y 2012, el cual contienen 5.560 aplicaciones de 179 familias de Malware.
- **Android Malware Genome Project (MGP)** es un proyecto de los años 2010 a 2011 que recopiló alrededor de 1.200 muestras de Malware clasificadas.

2) Herramientas: En los estudios revisados en la literatura, encontramos varias herramientas que permiten extraer características y comportamientos, para después ingresar esos datos en los algoritmos de aprendizaje. Las herramientas identificadas en la literatura se organizaron en Figura 6, Figura 10 y Figura 14, donde se categorizaron en cinco grupos (ver Figura 3). i) Decodificador, permite el ingreso codificado en binario de las aplicaciones dividiendo en malignas o benignas para sacar una mejor precisión de los algoritmos de aprendizaje. ii) Emulador, permite ejecutar la aplicación en un entorno virtual, para conocer el comportamiento del Malware dentro de un dispositivo. iii) Lenguaje de Programación, permite ejecutar los algoritmos de aprendizaje para analizar la información y obtener los resultados. iv) Software, permite agilizar el proceso de clasificación de características y comportamiento del Malware en tiempos de ejecución. v) Las herramientas propias son creadas por los investigadores, usando un método de análisis para ser utilizadas como una medida de protección. Algunas herramientas propias han sido implementadas en los antivirus por su grado de efectividad al momento de analizar las aplicaciones.

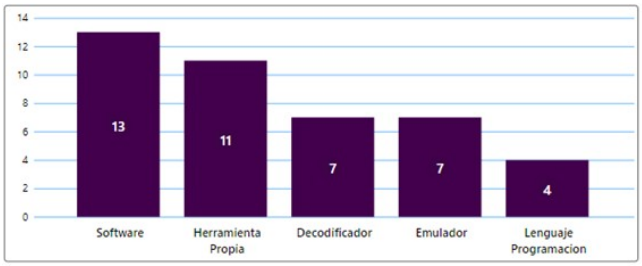
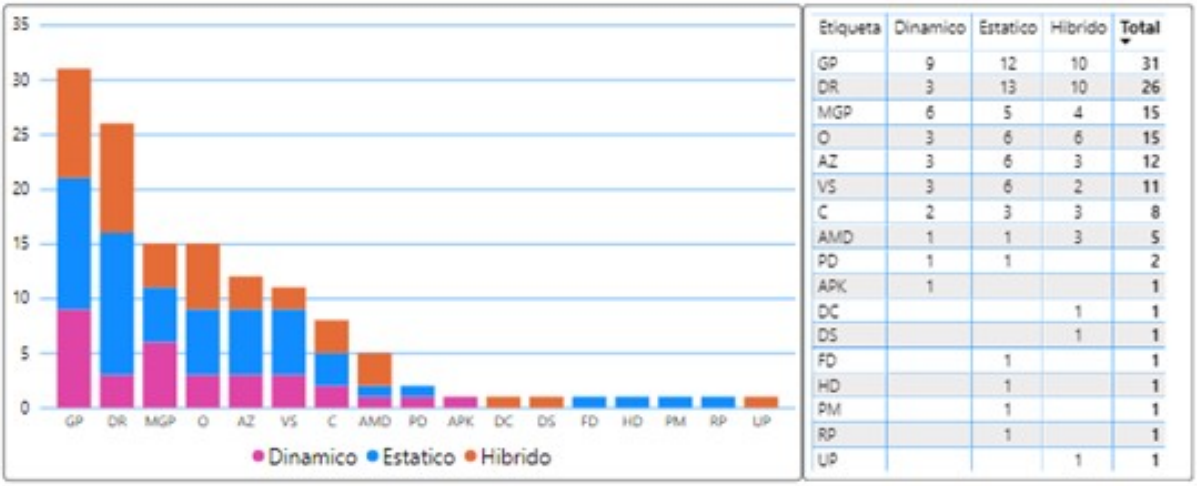


Fig. 3. Distribución de las Herramientas.



Etiqueta	Dinamico	Estatico	Hibrido	Total
GP	9	12	10	31
DR	3	13	10	26
MGP	6	5	4	15
O	3	6	6	15
AZ	3	6	3	12
VS	3	6	2	11
C	2	3	3	8
AMD	1	1	3	5
PD	1	1		2
APK	1			1
DC			1	1
DS			1	1
FD		1		1
HD		1		1
PM		1		1
RP		1		1
UP			1	1

Fig. 2. Recurrencia de los conjuntos de datos.

3) Técnicas de Aprendizaje: Las técnicas de aprendizaje son un grupo de algoritmos que permiten resolver una tarea específica, para después entrenarlo usando gran cantidad de datos siendo capaz de realizar predicciones. Por parte de la investigación se categorizaron las diferentes técnicas detalladas en la Figura 4.

i) Aprendizaje supervisado, son algoritmos que entrenan datos históricos etiquetados, buscando una función con variable de entrada para así obtener una salida adecuada a un nuevo valor y etiqueta. ii) Aprendizaje semi supervisado, son algoritmos para entrenar datos etiquetados como no etiquetados, y modela la distribución de probabilidad con las características y las etiquetas. iii) Aprendizaje no supervisado, son algoritmos que entrenan datos no etiquetados, únicamente describe la estructura de datos permitiendo un análisis exploratorio. iv) Aprendizaje profundo, son algoritmos basados en redes neuronales que permiten acceder a una mayor cantidad de información para hacer que el modelo de estudio tenga una mejor experiencia. v) Otros aprendizajes que no son de aprendizaje automático, también se han encontrado en la literatura que han permitido obtener mejores resultados en los análisis.

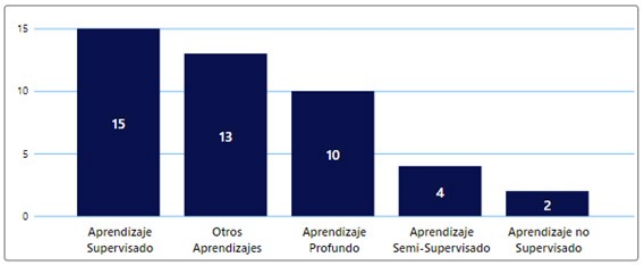


Fig. 4. Distribución de las técnicas de aprendizaje.

4) Técnica de análisis estático: El método de análisis estático clasifica aplicaciones conocidas como maliciosas o benignas basadas en la extracción de características o coincidencias sin necesidad de ejecutarlas. Analiza código fuente, coincidencia de reglas y otras características como permisos, flujo de datos o de control, archivos de recursos o de manifiesto [8]–[10]. La característica principal de este método es aplicar ingeniería inversa a las aplicaciones para inspeccionar a fondo su estructura [11]. Al no ejecutarse las aplicaciones maliciosas su comportamiento no es modificado [12], permitiendo recolectar información para inferir en una base de datos y permitir identificar ataques o comportamientos hostiles [13]. Una deficiencia común de este método es que no detecta de manera efectiva el malware que utiliza técnicas de ofuscación como: empaquetado, cifrado, compresión y deformación [14].

El análisis estático se organiza en una serie de métodos para realizar diferentes tipos de análisis e identificar el Malware, como podemos observar en la Tabla IV. Hemos identificado cada uno con una etiqueta para facilitar la búsqueda de los métodos de estudio. Como se visualiza en la Tabla IV, la recurrencia de algunos métodos en los estudios es muy notable. Para tener una mejor perspectiva de cómo se distribuyen en la Figura 5 se destacan los métodos Basado en Permisos (M1), Llamadas API (M4) y Análisis de código fuente (M9).

TABLE IV
MÉTODOS DE ANÁLISIS ESTÁTICO

Etq	Método	Referencias
M1	Permission Based	[8], [10], [12], [15]–[27]
M2	SimGru	[28]
M3	Based on Signature	[10]
M4	API Calls	[9], [10], [12], [18], [20], [21], [23]–[26], [29]–[33]
M5	Based on Intent and Activity	[10], [20], [26], [27]
M6	MuViDa	[34]
M7	URL score	[20], [35]
M8	Fusion of Functions and Rules	[14], [18], [36]
M9	Based on code analysis	[10], [11], [22], [36], [37]

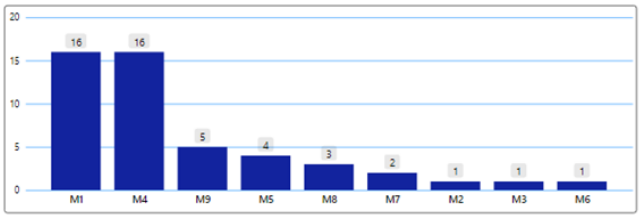


Fig. 5. Recurrencia de método estáticos.

5) Herramientas estáticas: Para la implementación de los métodos de análisis, se utilizaron herramientas que realizan tareas determinadas en un lapso definido permitiendo analizar las características de las aplicaciones Android. Las herramientas que han sido utilizadas en el análisis estático se las describen en la Tabla V.

TABLE V
MÉTODOS DE ANÁLISIS ESTÁTICO

Etq	Herramienta	Categoría	Referencias
D-H1	ApkTool	Decodificador	[8], [15], [16], [27], [29]
D-H12	MalScan	Decodificador	[31]
D-H16	MaMaDroid	Decodificador	[9]
D-H6	API-RDS	Decodificador	[29]
E-H5	Androguard	Emulador	[10]
E-H7	PSO	Emulador	[17]
E-H15	MV Dalvik	Emulador	[14], [36]
L-H2	Python	L. Programación	[8], [10], [12], [21], [23], [26], [27], [35], [37]
L-H11	Assembler	L. Programación	[20]
L-H3	JVM	L. Programación	[22], [28], [29], [33]
P-H17	ONAMD	H. Propia	[24]
P-H4	BadDroid	H. Propia	[25]
P-H8	AdDroid	H. Propia	[18]
P-H9	EveDroid	H. Propia	[30]
S-H10	Bio Analyzer	Software	[19]
S-H13	Scoot	Software	[32]
S-H14	FlowDroid	Software	[14], [32]
S-H18	Dex2jar	Software	[15]
S-H28	Weka	Software	[34]
S-H42	GA	Software	[11]

Como se observa en la tabla V, la recurrencia de las herramientas en los estudios es notable. Para entender el panorama de la distribución en la Figura 7 donde las herramientas python (D-H1), JVM (L-H2) y ApkTool (L-H3) son las que se destacan.

6) Técnicas de aprendizaje estáticas: En la Figura 6 los dos algoritmos que con más frecuencia que se ha utilizado en el aprendizaje supervisado son: Ramdon Forest (S-RF) es usado para la clasificación y regresión, para reducir la dimensionalidad y agrupar en un modelo más robusto las familias de Malware que existen en los conjuntos de datos. Support Vector Machines (S-SVM) es usado para clasificar los conjuntos de datos en código malicioso o benigno, e identificar la familia de Malware para tener una mejor precisión en el análisis.

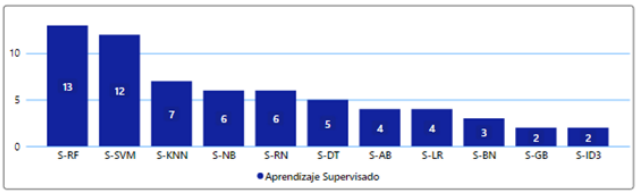


Fig. 6. Recurrencia de técnicas de aprendizaje supervisado para análisis estático.

Continuando, la Figura 8 muestra la recurrencia del uso de los algoritmos de cuatro tipos de aprendizaje. En el aprendizaje semi supervisado el uso del algoritmo J48 (SS-J48) es el más frecuente. Es usado para analizar la información de los conjuntos de datos para crear un árbol de clasificación que permitirá clasificar al Malware en sus diferentes familias. En el aprendizaje profundo la frecuencia no es tan alta ya que solo existen seis algoritmos y su uso es único en los estudios. En [28] usan funciones de similitud con los algoritmos InputSimGru (P-ISG) que muestra celdas básicas de GRU, HiddenSimGru (P-HSG) que busca los estados ocultos del Malware en las celdas de GRU y InputHiddenSimGru (P-IHSG) que es la combinación de los dos algoritmos anteriores. En aprendizaje no supervisados no están siendo utilizados para la clasificación de las características que se extraen en el código como se ve en la Figura 8 solo existe uno el NS-KN. Otros aprendizajes también utilizados son: el linear learning (O-AL) que se usó para recopilar la información extraída de las herramientas y clasificar según la familia del Malware y nonlinear learning (O-ANL se utilizó para no extraer características repetidas de las mismas familias de Malware.

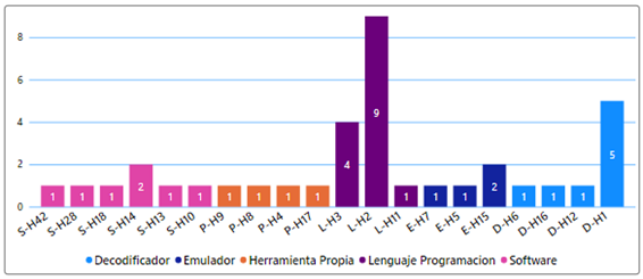


Fig. 7. Frecuencia del uso de las herramientas en análisis estático.

7) Técnica de análisis dinámico: El método de análisis dinámico se emplea para mejorar las limitaciones que existen en el análisis estático [38], para descompilar el APK ya que algunas aplicaciones están ofuscadas y cifradas [26]. Los comportamientos que se busca son las llamadas al sistema, llamadas API, tráfico de red o incluso acceso a archivos [38] aprovechando varios modelos de aprendizaje automático [26]. Se busca detectar software monitoreando el comportamiento

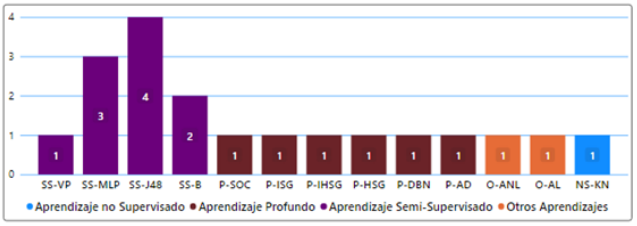


Fig. 8. Recurrencia de técnicas de aprendizaje en el análisis estático.

en tiempos específicos al momento de ejecutar la aplicación, clasificando el Malware en genéricos o específicos extrayendo sus características [9]. El enfoque ideal es monitorear las acciones y su comportamiento que ocurren durante la ejecución de la aplicación en un entorno virtual controlado [13] como un emulador, o un dispositivo real [39].

La técnica dinámica es más efectiva frente al código dinámico, pero a su vez compleja, teniendo un alto costo computacional. Una solución es analizar el Malware utilizando los algoritmos eficientes mientras se reduce el análisis de sobrecarga [40]. Para entender el comportamiento de la aplicación se obtiene la secuencia y frecuencias de las llamadas API de los códigos en el tiempo de ejecución [13]. En ocasiones permite detectar la variante de Malware, que sufre una transformación en las características estáticas, pero conserva las dinámicas [40]. La gran parte de métodos previos requieren la transformación del Kernel para analizar Malware. El problema de realizar este paso es la reconstrucción del Kernel a su estado original y el modelo exacto, por lo que se vuelve tedioso para el investigador [40].

El análisis dinámico se desglosa en una serie de métodos para realizar diferentes tipos de inspecciones e identificar el Malware, como podemos observar en la Tabla VI. Al igual que en los métodos estáticos hemos identificado cada uno por una etiqueta para que en la lectura facilite la búsqueda de los métodos de estudio.

TABLE VI
MÉTODOS DE ANÁLISIS DINÁMICO

Etq	Método	Referencias
M10	Based Behavior	[13], [26], [38], [40]–[43]
M11	DFA Analysis	[44]
M12	API-Images Calls	[13], [45]
M13	Trees Suffixes	[40]
M14	Based on Code Frequency	[39], [46]
M15	Prediction Conformal	[47]
M16	Based Behavior API Calls	[9], [26], [48]
M17	Based Self-Concealment Behavior	[49]
M18	SpyDroid	[50]

Como se visualiza en la Tabla VI, la recurrencia de algunos métodos en los estudios es muy notable. Para tener una mejor perspectiva de cómo se distribuyen en la Figura 9 se destacan los métodos Basado en Comportamiento (M10), Basado en Comportamiento de Llamadas API (M16), Llamadas API-Imágenes (M12) y Basado en Frecuencia del Código (M14) a continuación, se explicará cada uno de ellos.

8) Herramientas dinámicas: En la implementación de los métodos de análisis dinámicos, se utilizaron herramientas que se encargan de recolectar la información del comportamiento del Malware, estas herramientas se referencian en la Tabla VII.

TABLE VII
HERRAMIENTAS DE USO EN EL ANÁLISIS DINÁMICO

Etq	Herramienta	Categoría	Referencias
D-H22	Autoencoders	Decodificador	[13]
D-H27	Euphony	Decodificador	[50]
E-H19	DroidBox	Emulador	[26], [41]
E-H25	AUNTIEDroid	Emulador	[9]
E-H32	Sandbox	Emulador	[46], [49]
P-H21	MalDroid	H. Propia	[48]
P-H24	DL-Droid	H. Propia	[39]
P-H41	MLDroid	H. Propia	[42]
S-H20	Monkey	Software	[38], [40], [47], [50]
S-H23	Dynalog	Software	[39]
S-H26	KerTSDroid	Software	[43]
S-H28	Weka	Software	[44], [45]

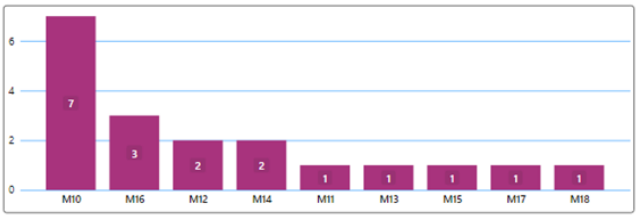


Fig. 9. Recurrencia de métodos dinámicos.

Como se observa en la Tabla VII, la recurrencia de las herramientas en los estudios es notable. Para entender el panorama de la distribución en la Figura 10 donde las herramientas: Monkey (S-H20), Sandbox (E-H32) y DroidBox (E-H19) son las que se destacan.

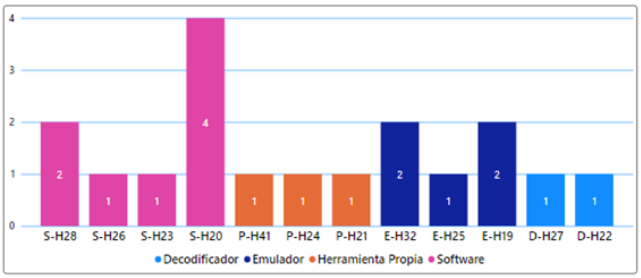


Fig. 10. Recurrencia de las herramientas dinámicas.

9) Técnicas de aprendizaje dinámicas: En la Figura 11, los algoritmos con mayor frecuencia se han utilizado en el

aprendizaje supervisado son los mismos que se encontraron en el análisis estático. Por lo cual hablaremos del algoritmo decisión tree (S-DT) es usado para obtener decisiones de los procesos del comportamiento de Malware al momento de ser ejecutado en el entorno virtual. Esto facilita la comparación de comportamiento del Malware para identificar a que familia pertenece y conocer si la aplicación es maliciosa o benigna.

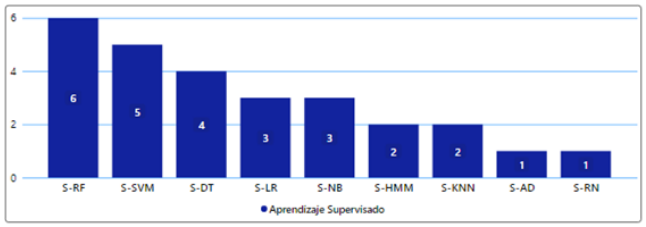


Fig. 11. Recurrencia de técnicas de aprendizaje supervisado para análisis dinámico.

Continuando en la Figura 12 muestra la recurrencia del uso de los algoritmos de cuatro tipos de aprendizaje. En el aprendizaje semi supervisado el uso del algoritmo multilayer perceptron (SS-MLP) es una red neuronal artificial que toma los conjuntos de características del comportamiento extraído de P-H41 y P-H32 como método de entrada. SS.MLP al tener una capa oculta mejora la precisión de análisis de Malware de otras redes neuronales. El aprendizaje profundo encontramos dos algoritmos donde su frecuencia es única en los estudios de análisis dinámico. Estos algoritmos son Deep Learning (P-DL) y Self concealment behavior (P-SOC) este último ayuda a encontrar al Malware que se oculta en las aplicaciones buscando su comportamiento. En otros aprendizajes la frecuencia es igualmente única como: EnDroid (O-ED) entrenan un modelo adoptando un meta clasificador, el cual permite distinguir entre aplicaciones benignas y maliciosas generando un vector de características para cada aplicación [41]. DATDroid (O-DD) procede a clasificar en forma de árbol los comportamientos extraídos durante la ejecución esto incluye características y permisos de la aplicación [38]. Los otros dos algoritmos Sub-Detectores (O-SD) en [50] y Mondrian's inductive conformal prediction (O-LCMICP) en [47].

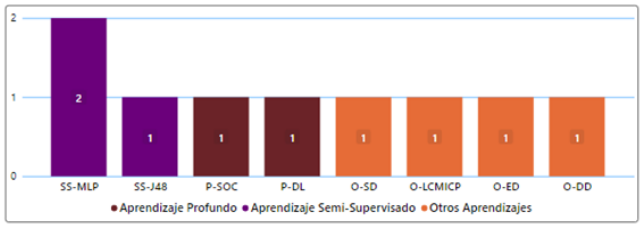


Fig. 12. Recurrencia de técnicas de aprendizaje en el análisis dinámico.

10) Técnica de análisis híbrido: El análisis híbrido es propuesto por la necesidad de detectar Malware de una manera eficiente que mejora su rendimiento utilizando características como la cobertura de código y extracción de características durante la ejecución de las aplicaciones. Con esta técnica es posible detectar Malware ofuscado, reduciendo la tasa de falsos positivos [51]–[53].

El análisis híbrido se desglosa en una serie de métodos para realizar diferentes tipos de inspecciones e identificar el Malware, como podemos observar en la Tabla VIII. Al igual que en los métodos estáticos y dinámico hemos identificado a cada uno con una etiqueta para que facilite la lectura facilite la búsqueda de los métodos de estudio.

TABLE VIII
MÉTODOS USADOS POR EL ANÁLISIS HIBRIDO

Etq	Método	Referencias
M1	Permission Based	[53]–[63]
M3	Based on Signature	[64]–[66]
M4	API Calls	[53], [56]–[59], [67]
M5	Based on Intent and Activity	[63]
M8	Fusion of Functions and Rules	[65], [68]
M10	Based Behavior	[55]–[57], [59], [68], [69]
M12	API-Images Calls	[69]
M14	Based on Code Frequency	[63]
M19	Byte2vec	[51]
M20	ORGB	[52]
M21	System Calls	[53], [58], [59], [67]
M22	Based on Anomalies	[58], [60], [66]
M23	LSTM Stacked	[70]
M24	TANMAD	[58]
M25	Based on Deep Learning	[71]
M26	Activity Biagrams	[72]
M27	Based Network Traffic	[61]–[63]
M28	SAMADroid	[73]

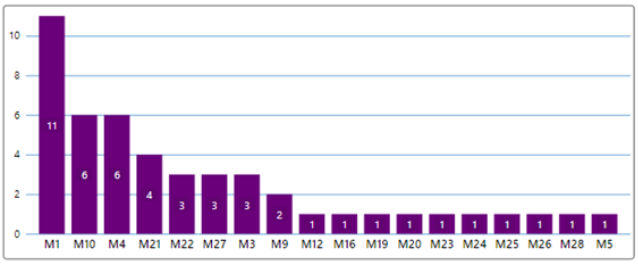


Fig. 13. Recurrencia de métodos híbridos.

Como se visualiza en la Tabla VIII, la recurrencia de algunos métodos en los estudios es muy notable. Para tener una mejor perspectiva de cómo se distribuyen en la Figura 13 se destacan los métodos como: Basado en Permisos (M1) y Llamadas API (M4) que son los más frecuentes en el análisis estático (ver Figura 5), también Basado en Comportamiento (M10) que es el más frecuente en el análisis dinámico (ver Figura 9). La unificación de estáticos y dinámicos cuenta con limitaciones que son superados por métodos híbridos pero su frecuencia de uso está en desarrollado. En la Figura 13, Llamadas al Sistema (M21) y Basado en

Anomalías (M22) son los más destacados en métodos híbridos.

11) Herramientas híbridas: En la implementación de los métodos de análisis híbrido, se utilizaron herramientas que se encargan de recolectar la información del comportamiento del Malware, estas herramientas se referencian (ver Tabla IX).

TABLE IX
HERRAMIENTAS USADAS EN EL ANÁLISIS HÍBRIDO

Etq	Herramienta	Categoría	Referencias
D-H1	ApkTool	Decodificador	[53], [64], [71], [72]
D-H30	DAMBA	Decodificador	[52]
E-H5	Androguard	Emulador	[55]
E-H5	MV Dalvik	Emulador	[57]
E-H19	DroidBox	Emulador	[55], [63], [72], [73]
E-H29	BXDBoost	Emulador	[51]
E-H32	Sandbox	Emulador	[53], [59], [60], [67]
L-H2	Python	L. Programación	[55], [58], [59], [62], [63], [67], [69]
L-H11	Assembler	L. Programación	[54]
L-H3	JVM	L. Programación	[64]
L-H33	R-Studio	L. Programación	[70]
P-H31	DroidNative	H. Propia	[65]
P-H35	MalPat	H. Propia	[56]
P-H36	Rapid	H. Propia	[57]
P-H39	NTPDroid	H.Propia	[61]
S-H18	Dex2jar	Software	[53], [71]
S-H20	Monkey	Software	[60]
S-H23	Dynalog	Software	[57]
S-H34	AspectDroid	Software	[68]
S-H37	Androtomist	Software	[66]
S-H38	Tcpdump	Software	[60]
S-H40	APIMonitor	Software	[73]

Como se observa en la Tabla IX, la recurrencia de las herramientas en los estudios es notable. Para entender el panorama de la distribución en la Figura 14 donde las herramientas: Python (L-H2), Apktool(D-H1) mencionados en el método estático ver (Figura 7), Sandbox (E-H32), Droidbox (E-H19) en el método dinámico ver (Figura 10) y Dex2jar (S-H18) son las que destacan.

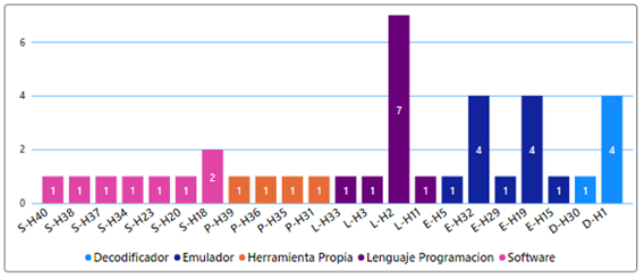


Fig. 14. Recurrencia de las herramientas híbridas.

12) Técnicas de aprendizaje híbridas: En la Figura 15 los algoritmos que con más frecuencia que se ha utilizado en el aprendizaje supervisado son: Random Forest (S-RF) y Support Machine Vector (S-SVM) tanto en el análisis estático como dinámico (ver Figura 6 y Figura 11). En la Figura 15 existen otros algoritmos que son usados con frecuencia como: k-Nearest Neighbors (S-KNN) se usó para clasificar si una aplicación es benigna o maliciosa, para después poder ordenar las aplicaciones maliciosas por su familia. Naive Bayes (S-NB) se usó para que aprenda de los datos de entrenamiento para facilitar la predicción en tiempo real y mejor exactitud en clasificación de las familias de Malware.

Fig. 15. Recurrencia de técnicas de aprendizaje supervisado para análisis híbrido.

Continuando, la Figura 16 muestra la recurrencia del uso de las técnicas de aprendizaje. En el aprendizaje profundo encontramos seis algoritmos donde su frecuencia es única excepto el algoritmo Generating Patters (P-GP). En [69] implementan P-GP para generar patrones de coincidencia en la frecuencia de las llamadas al sistema que usan las aplicaciones benignas. En [56] usan P-GP para identificar patrones ocultos en las llamadas API utilizando aplicaciones maliciosas o benignas. En otros aprendizajes la frecuencia es igualmente única como: XGBoost (O-XGB) en [51] se usó como clasificador de ataques seriales, demostrando que tiene un mejor rendimiento cuando las características extraídas son el número de bytes del Malware. En [61] usan FP-Growth (O-FPG) que genera patrones de frecuencia basándose en las características extraídas por los métodos M27 y M1. En el aprendizaje no supervisado existen un solo algoritmo que es stochastic gradient descent (NS-SGD) que se usó para optimizar y evitar repetir características extraídas por los métodos M1 y M4.

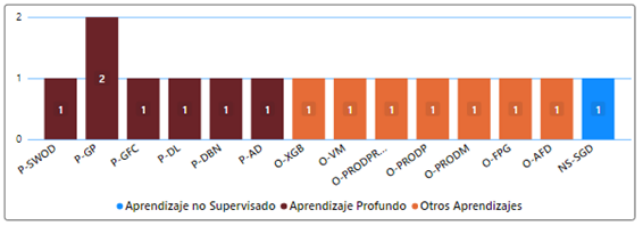


Fig. 16. Recurrencia de técnicas de aprendizaje en el análisis híbrido.

13) Métricas: En la revisión de la literatura se encontraron varías métricas con las que miden la efectividad y exactitud

del análisis de Malware en Android, para estas medidas se utilizan modelos matemáticos para su cálculo en la Figura 17 se observa la recurrencia de uso de estas métricas.

- *Accuracy (AC):* Mide el porcentaje que el modelo ha acertado, es la relación entre el número de predicciones correctas y el número total de muestras de entrada.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

- *Precision (PR):* Mide la calidad del modelo en tareas de clasificación, es el número de resultados positivos correctos dividido por el número de resultados positivos predichos por el clasificador.
- *F-Measure (F1):* Es la media armónica entre precisión y recall en un solo valor. El rango de F1 es [0, 1] y dice que tan preciso es el clasificador.
- *Recall (RC):* La recuperación es el porcentaje de archivos maliciosos conocidos que nuestras firmas detectan..
- *False Positive (FPR):* Proporciona los casos negativos de Malware que la prueba detecta como positivos.
- *True Positive (TPR):* Proporciona los casos verdaderamente positivos de Malware entre los casos positivos detectados.
- *False Negative (FNR):* Proporciona los casos positivos de Malware que son detectados como negativos.
- *True Negative (TNR):* Proporciona los casos verdaderamente negativos de Malware entre los casos negativos detectados.

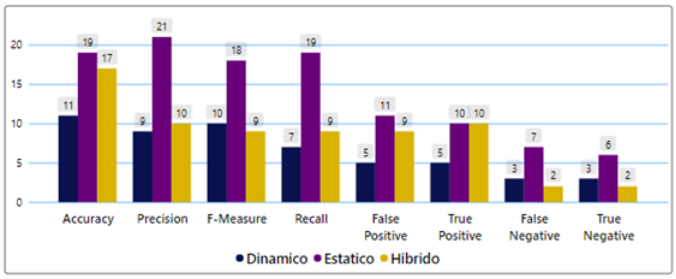


Fig. 17. Recurrencia de las métricas usadas por las técnicas de análisis de Malware.

III. Resultados y Discusión

SMP1: ¿Existe una taxonomía para el análisis de Malware en Android?

La taxonomía permite clasificar de una manera jerárquica y organizada las técnicas de análisis de Malware en Android. En la Figura 1 se observan las tres técnicas de análisis que son: estático, dinámico e híbrido. Las técnicas cuentan con métodos que sirven para extraer características de código y comportamiento de Malware. También se cuenta con herramientas categorizadas según la técnica que se vaya a emplear para analizar el Malware. Por último, tenemos las técnicas de aprendizajes que proporcionan la clasificación de las aplicaciones en benignas y maliciosas. La taxonomía es la parte de estructuración de fases que se debe seguir para tener un análisis de Malware en Android adecuado.

SMP2: ¿Cuál es la distribución de estudios con respecto a revistas y conferencias en los últimos cinco años?

Los estudios con respecto a revistas y conferencias se distribuyen de la siguiente manera: en los años 2017 y 2021 se identificaron estudios sobresalientes solo en revistas, entre los años 2018 a 2020 se identificaron estudios tanto en revistas como conferencias, sin embargo, en revistas existe una mayor cantidad de estudios (ver Figura 18).

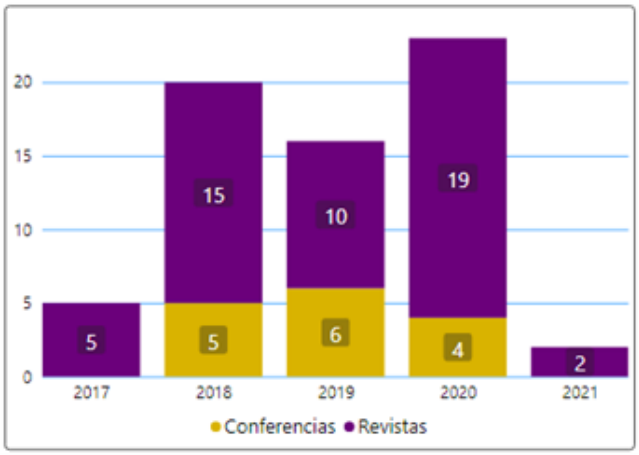


Fig. 18. Distribución de estudios en los últimos cinco años.

SLRP1: ¿Que conjuntos de datos son utilizados en el análisis de Malware en Android?

En la revisión de literatura, detallan que utilizan diecisiete conjuntos de datos distribuidos en los diferentes estudios (ver Figura 2). El DataSet de mayor relevancia es Google Play, el uso es de doce veces en el análisis estático, nueve veces en el análisis dinámico y diez veces en el análisis híbrido. Los siguientes DataSet que tiene una relevancia significante es Drebin y Android Malware Genome Project, observar sección 2.3.1.

SLRP2: ¿Qué Métodos se destacan en el análisis de Malware en las técnicas estáticas, dinámicas e híbridas?

En el análisis estático existen varios métodos como se pueden observar en la Tabla IV, así mismo la frecuencia de uso se detalla en la Figura 5 para destacar los siguientes métodos:

- *Basado en Permisos(M1):*En Android un mecanismo de seguridad es la declaración de permisos para que una aplicación acceda a los recursos del sistema, los permisos se declaran en el archivo AndriodManifest.xml. El M1 es

propenso a falsos positivos ya que aplicaciones benignas pueden solicitar permisos peligrosos. Así mismo otorga a los usuarios permisos en tiempo de ejecución, pero en algunas ocasiones no es posible que sean otorgados.
- *Llamadas API(M4):* En [15] extraen las características de invocación considerando la sobrecarga y analizando el bytecode de forma semántica secuencial. En [10] analizan las funciones API, extrayendo características que el Malware utiliza comúnmente y se categoriza por recursos. En [29] eliminan idénticas instancias de los paquetes API de las aplicaciones para encontrar la diferencia de ransomware y aplicaciones limpias. En [30] relacionan varios grupos de llamadas API para poder diferenciar y desencadenar eventos distintos para la clasificación de familias Malware. En los demás estudios extraen el archivo classes.dex que contiene las funciones de las invocaciones API por parte de las aplicaciones, además el archivo almacena los códigos bytes.
- *Análisis de Código Fuente (M9):* En [10] describen a M9 como una estructura específica para enumerar las clases y métodos que usa una aplicación. En [11] el M9 se usa para mejorar y busca parámetros para minimizar las funciones de múltiples categorías. En los estudios [22], [36], [37] usan aprendizaje automático para proporcionar mejores precisiones y revelar comportamiento de aplicaciones más granulares.

En el análisis dinámico existen varios métodos como se puede observar en la Tabla VI, así mismo la frecuencia de uso se detalla en la Figura 9 para destacar los siguientes métodos:

- *Basado comportamiento (M10):* El método está basado en la recopilación de datos del sistema en un determinado tiempo de ejecución del código, se recopila datos como: llamadas al sistema, datos de red, archivos y memoria modificación.
- *Basado comportamiento llamadas API (M16):* En los estudios [9], [26], [48] , el comportamiento detecta las secuencias de llamadas API abstractas o criptográficas ejecutadas en un entorno virtual, esto permite mayor resistencia en los cambios API y reducir el entrenamiento de modelos. Se extrae las características de API clasificando en paquetes de familia, se realiza esto para evitar equivocaciones en aplicaciones que tengan prefijos en los paquetes.
- *Llamadas API-Imágenes (M13):* En los estudios [13], [45] el método extraer características de comportamiento al momento de la ejecución de las llamadas API o permisos de manera secuencial formando matrices dispersas en parecen imágenes. Para este método se transforma el Malware usando la técnica de visualización y partiendo el código en varias secciones dejando en forma de imagen, esto se usará como "huella digital" del comportamiento de las aplicaciones.

En el análisis híbrido existen varios métodos como se puede observar en la Tabla VIII, así mismo la frecuencia de uso se detalla en la Figura 13 para destacar los siguientes métodos:

- *Llamadas al sistema (M21):* En [53] extraer las características de M1 y M4 en un entorno de caja de arena con el comando strace para encontrar cadenas de similitudes. En [58], [59] resuelven la limitación de cobertura de código, generando una secuencia finita de acuerdo con la ejecución de llamadas API en el código. En [67] muestran M21 como un artefacto que extrae llamadas de archivos ReadFromWeb que leen una página web en un determinado URL.
- *Basado en Anomalías (M22):* En [60], [66] busca distintos parámetros del sistema y componentes, para lograr detectar Malware en el dispositivo, para esto se supervisa el comportamiento del dispositivo. En [58] busca la secuencia de M1, M4 Y M21 en tiempos de ejecución para acceder a las funciones protegidas y observa las anomalías al momento que el Malware actúa.

SLRP3: ¿Qué herramientas se destacan en el análisis de Malware en las técnicas estáticas, dinámicas e híbridas?

En el análisis estático existen varias herramientas como se puede observar en la Tabla V, así mismo la frecuencia de uso se detalla en la Figura 7 para destacar las siguientes herramientas:

- *Python (L-H2):* En los estudios usaban la herramienta para generar la clasificación y procesos de análisis de Malware. Otra ventaja por la cual los estudios usaban la herramienta era por las bibliotecas de análisis de datos y de aprendizaje automático.
- *JVM (L-H3):* Los estudios usan JVM para extraer el ByteCode ya sea .NET o MSIL de las aplicaciones y compilar en un código nativo de cualquier plataforma.
- *ApkTool (D-H1):* Es un programa utilizado para ingeniaría inversa, que decodifica y reconstruye recursos muy similares al original. En [16], [29] utilizaron D-H1 para descompilar y extraer los archivos .apk obteniendo permisos, intenciones explícitas.

En el análisis estático existen varias herramientas como se puede observar en la Tabla VII, así mismo la frecuencia de uso se detalla en la Figura 10 para destacar las siguientes herramientas:

- *Monkey (S-H20):* Se utiliza S-H20 para captura las llamadas al sistema en un tiempo de ejecución lo que permite la automatización de una manera aleatoria del comportamiento del Malware en los dispositivos.
- *Sandbox (E-H32):* Emula la funcionalidad de un dispositivo físico en un ambiente controlado, aislando el programa infectado para analizar su comportamiento.

- *DroidBox (E-H19):* Es una herramienta de código abierto, ejecutada en un entorno de caja de arena que monitorea y analiza en un determinado tiempo de ejecución el comportamiento del Malware. El E-H19 es similar al entramiento y prueba de los DataSet. En [41] también mencionan que E-H19 extrae características de aplicación como: archivos y operaciones de acceso a la red, fugas de información, operaciones criptográficas, entre otras.

En el análisis híbrido existen varias herramientas como se puede observar en la Tabla IX, así mismo la frecuencia de uso se detalla en la Figura 14 para destacar las siguientes herramientas:

- *Dex2jar (S-H18):* Es una herramienta basada en ingeniería inversa, descompila el archivo apk y lo convierte en un archivo JAR (Java ARchive), siendo un formato de archivo de paquete que se utiliza para agregar clases de Java, metadatos y recursos asociados como imágenes o texto [53], [71].

SLRP4: ¿Qué técnicas de aprendizaje se destacan en el análisis el Malware?

En los estudios recopilados se identificaron técnicas de aprendizaje como parte de una fase de clasificación que permite separar aplicaciones entre benignas y maliciosas. La investigación se dividió en los siguientes aprendizajes:

- Aprendizaje supervisado es la técnica de mayor uso como se puede observar en la Figura 6, Figura 11 y Figura 16, entre los algoritmos con mayor frecuencia son: Random Forest, Vector Support Machines, Decision Tree, K-Nearest-Neighbor y Naives Bayes.
- Aprendizaje semi supervisado con mayor frecuencia en el análisis estático (ver Figura 8), también son utilizados por el análisis dinámico (ver Figura 12), entre los algoritmos más relevantes son: J48 y Multilayer Perceptron.
- Aprendizaje profundo es una técnica utilizada recientemente para analizar Malware, por esa razón su mayor frecuencia es el algoritmo de Pattern Generation.
- Aprendizaje no supervisado no es una técnica óptima para el análisis de Malware, los algoritmos utilizados son: K-Neighbor en el análisis estático (ver Figura 8) y el Stochastic Gradient Descent en el análisis híbrido (ver Figura 16).
- Otros aprendizajes han sido técnicas usadas por los estudios para mejorar la efectividad del análisis de Malware, tenemos los siguientes aprendizajes: linear learning y nonlinear learning en el análisis estático, EnDroid y DATDroid en el análisis dinámico, XGBoost y FP-Growth en el análisis híbrido (ver Figura 8, Figura 12 y Figura 16).

SLRP5: ¿Cuáles son las ventajas y limitaciones de las técnicas de análisis de Malware de Android?

TABLE X
VENTAJAS Y LIMITACIONES DE LAS TÉCNICAS DE ANÁLISIS DE MALWARE.

Estático	Dinámico	Híbrido
	VENTAJAS	
Consumo computacional bajo.	Detecta Malware nuevo o descocido.	Mejora limitaciones del análisis estático y dinámico.
Utiliza ingeniería inversa para agilizar los procesos.	Analiza Malware poliforme.	Analiza el comportamiento oculto del Malware en tiempo de ejecución.
Extrae características del código.	Analiza el Malware en tiempo real.	Verifica las anomalías del código y comportamiento del Malware.
No ejecuta las aplicaciones.	Analiza Malware cifrado y detecta cargas dinámicas en el código.	Descifra el bytecode de código nativo.
Escanea y verifica el Malware rápidamente.	Analiza rutas del Malware en entorno virtual.	Detecta Malware poliforme y ofuscado.
Revela las rutas de ejecución del Malware.		Mejor tasa de detección con una alta precisión.
		Detecta familias de Malware desconocidas.
	LIMITACIONES	
No es capaz de detectar Malware nuevo.	Alto consumo computacional.	Mayor complejidad por el uso de técnicas estáticas y dinámicas.
No detecta Malware poliforme.	No es capaz de analizar comportamiento oculto del Malware.	Uso de recursos y tiempo alto.
Incapacaz de analizar Malware ofuscado.	Incapaz de analizar técnicas de encriptación en Malware.	
Procesos costosos en actualizar las firmas de Malware.	Carece de cobertura de código.	
No analiza Malware en tiempo de ejecución.		

SLRP6: ¿Qué medidas de rendimiento se utilizan para el análisis de Malware de Android?

En la Tabla XI en el primer bloque se observa los estudios que han utilizado análisis estático para detectar el Malware, con el uso del DataSet correspondiente, la técnica de aprendizaje con la que evaluaron los datos extraídos por los métodos y el resultado más relevante en el caso de los estudios que usaron más de una técnica de aprendizaje. En la Tabla XI en el segundo bloque se observa los estudios que han utilizado análisis dinámico para detectar el Malware, con el uso del DataSet, técnica de aprendizaje con la que evaluaron los datos extraídos y el resultado de mayor relevancia. En la Tabla XI en el tercer bloque se observa los estudios que han utilizado análisis híbrido para detectar el Malware, con el uso del DataSet, técnica de aprendizaje y los resultados relevantes.

TABLE XI
RESULTADO DE LAS MÉTRICAS EN EL ANÁLISIS ESTÁTICO, DINÁMICO E HÍBRIDO

Ref	DataSet	Técnica Aprendizaje	Resultado
	Bloque 1: Análisis estático		
[8]	VS; DR	S-DT; S-ID3; S-SVM; S-RF; S-KNN; SS-J48; S-BN; S-NB	SS-J48; TPR=98.8; FPR=0.12; PR=98.8; RC=98.8; F1=98.8; AC=98.7
[37]	DR	S-KNN; S-SVM; S-ID3	S-KNN; PR=93.87; RC=91.47; F1=92.65; AC=92.93
[28]	DR; AMD	P-ISG; P-HSG; P-IHSG	P-ISG; PR=98.47; RC=99.57; AC=98.95
[15]	GP; AZ	S-RF	TPR=95.10; FPR=4.70
[34]	GP; O; MGP; C	S-RF	TPR=96; FPR=2.8
[10]	O	S-RN	F1=100; PR=100; RC=100; AC=96.42
[12]	DR; PM	S-SVM	TPR=99.5; FPR=0.5; PR=99.5; RC=99.5
[29]	GP; RP; HD	S-NB; S-SVM; S-RF	TPR=96.4; FPR=97.3; AC=96.5
[16]	GP; DR;MGP;FD	S-BN; S-NB; SS-J48; S-RF; S-SVM; S-LR; S-KNN; S-DT; SS-B; S-AB	S-RF; F1=98.7; PR=98.7; RC=98.6
[35]	DR	S-SVM	F1=85; AC=99
[17]	DR; GP; AZ	SS-J48; S-KNN; S-RF; SS-MLP; S-AB	S-KNN; TPR=93.7; FPR=15; F1=91.5; PR=89.4; RC=93.7
[18]	GP; C; VS	S-AB	PR=99.33; RC=99.36; AC=99.11
[30]	VS; PD; GP	S-RN	VS; GP; PD; 2013; PR=99.8; RC=99.8; AC=99.8
[11]	DR; GP	S-NB; SS-J48; S-RF; SS-MLP	S-NB; TPR=96; FPR=24.50; F1=96.80; PR=97.50; RC=96; AC=94.17
[14]	VS; AZ	S-RN	PR=97; RC=97; F1=97;
[19]	DR; MGP	S-RN; SS-MLP; SS-VP	SS-MLP; TPR=87; FPR=7.2; F1=89.6; PR=92.3; RC=87; AC=90
[20]	DR	P-AD; S-SVM; S-KNN; S-RF; S-LR; S-NB; S-GB; S-DT; SS-B; NS-KN	S-RF; F1=94; PR=94; RC=95; AC=94.33
[31]	AZ	S-RF; S-KNN	F1=98.8; AC=98.8
[21]	AZ; VS; MGP	P-DBN	FNR=7; FPR=1.88; PR=98.15; RC=99.30; F1=98.72; AC=98.71
[32]	O	P-SOC	O; PR=84.02; RC=90.62; F1=87.19
[22]	O	S-RF; S-BN; S-SVM; S-LR; S-DT; S-AB	S-SVM; S-LR; S-AB; PR=95.7; RC=95.7; AC=95.6
[33]	DR; O	S-RF	PR=99.80; RC=84.72; F1=91.73; AC=93.77
[36]	DR	S-RN	TPR=94.6; FPR=0.96; AC=96.5
[9]	VS; GP	N/A	PR=91; RC=93; F1=92
[23]	O	S-RN; S-SVM	S-RN; PR=99.6; RC=99.2; F1=99.4; AC=99.6
[24]	C	S-SVM; S-RF	FPR=11.9; PR=89.8; RC=87.6; F1=88.7; AC=87.8
[25]	GP; AZ	O-AL; O-ANL; S-SVM	O-AL; O-ANL; AC=98.9
[26]	MGP; GP	S-KNN; S-SVM; S-LR	S-LR; AC=81.03
[27]	DR; GP	S-RF; S-NB; S-DT; S-GB	S-DT; PR=97; RC=97; F1=97
	Bloque 2: Análisis dinámico		
[41]	AZ; DR; GP	O-ED	TPR=98.2; FPR=1.6; PR=95.87; RC=98.30; F1=97.02; AC=98.18
[44]	AMD; DR	S-SVM	DR; AC=82; AMD; AC=78
[45]	DR	S-NB; S-RF; S-KNN; S-SVM	S-SVM; AC=92.7
[38]	MGP; APK	O-DD	FPR=8.3; PR=93.1; RC=90; AC=91.7
[48]	VS; GP; C	S-RF	PR=99.63; F1=98.24
[13]	GP;VS;PD;MGP;C	S-AD; S-RN; S-HMM	TPR=96; PR=98; F1=97; AC=94
[42]	GP; MGP	SS-MLP; S-DT	SS-MLP; AC=98.8
[40]	MGP; GP	S-HMM	TPR=96; TNR=97; FNR=4; FPR=3; AC=96
[46]	MGP; O	S-NB; SS-J48; S-RF; S-SVM; S-LR; SS-MLP; S-DT	S-RF; TPR=90.9; FPR=0.068; TNR=93.2; FNR=0.091; F1=91.9
[47]	AZ	S-RF; O-LCMICP	F1=74.5; AC=74.46
[39]	O	O-DL	PR=98.9; RC=99.56; F1=98.82; AC=98.5
[9]	GP; VS		PR=91; RC=93; F1=92
[49]	GP	P-SOC	PR=97.47; RC=100; F1=98.72
[43]	O	S-SVM; S-NB; S-LR; S-DT	S-KNN; TPR=98; FP=0.4; AC=98.04
[50]	GP; AZ	S-RF;O-SD	P-SD; PR=86; RC=84; F1=84
[26]	MGP; GP	S-RF; S-KNN; S-SVM; S-DT; S-LR	S-RF;PR=93;RC=93;F1=93;TPR=96.09;FNR=3.91;TNR=90.8;FPR=9.2
	Bloque 3: Análisis híbrido		
[64]	O	P-AD	AC=99.02
[51]	DR; DS; AMD	O-XGB	AMD; FNR=0.92; FPR=0.92; F1=99.08
[52]	MGP; C	S-TAN	PR= 95.68; RC=98.55; F1=97.09; AC=96.90
[65]	GP; O	P-GFC; P-SWOD	TPR=93.57; FPR=2.7; PR=97
[53]	DR; GP	S-KNN; S-RF; S-NB; S-SVM; S-DT	S-RF; TPR=99.4; FPR=12.4; PR=98.2; RC=99.4; AC=97.9
[54]	O; GP	S-IRF	TPR=95; FPR=4.6; AC=98.1
[55]	C; GP	S-SVM	TPR=96.19; TNR=98.79; AC=97.78
[68]	GP; DR	O-AFD	PR=96.4; RC=93.02; F1=94.68
[69]	MGP	P-GP	AC=90.19
[56]	GP; VS; C	P-GP	PR=91.72; RC=89.18; F1=98.04
[57]	O	P-DBN	TPR=100; FPR=3.2; PR=98.7; RC=100; F1=99.3; AC=99.1
[66]	DR; VS; AZ	S-NB; S-RF; S-KNN; S-SVM; NS-SGD; S-LR; S-AB	DR; PR=99.22; RC=99.22; F1=99.22; AC=99.22
[70]	DR	S-RN	AC=93.29
[58]	DR	S-TAN	TPR=100; AC=99
[71]	O	S-RN	AC=92.67
[59]	GP; AMD; AZ	O-PRODPRODD; O-PRODP; O-PRODM; O-VM; S-RN; S-DL	O-PRODM; TPR=98.79; TNR=97.75; FNR=1.21; FPR=2.25; AC=98.27
[60]	GP; DR	S-RF	AC=96
[72]	UP	S-RF; S-KNN; S-SVM; S-DT	S-KNN; TPR=98; FPR=0.4; AC=98.04
[61]	MGP	O-FPG	AC=94.25
[73]	DR; O; MGP; GP	S-SVM	TPR=98.5; FPR=0.5; AC=98.97
[62]	DC; DR	S-RF; S-NB; S-KNN; S-SVM; S-LR; S-DT	S-KNN; PR=100; RC=97; F1=99; AC=97
[67]	AZ; AMD	S-RN	PR=99.69; RC=97.60; F1=98.63
[63]	GP; DR	S-RF; S-GB; S-DT; S-NB	S-DT; PR=88; RC=88; F1=88; TPR=81; FPR=95

IV. Conclusiones

Los procesos para aplicar un mapeo sistemático proporcionan al lector las fases como: F1 Realización de la cadena de búsqueda, definir términos de exclusión e inclusión; F2 Definir preguntas de investigación, estrategias de búsqueda y F3 extracción, síntesis de datos. El proceso de SM facilita el reconocimiento de los intereses de investigación organizando la selección de literatura para una revisión más profunda. En nuestra investigación posibilitó tener una visión general y accesible sobre las técnicas de análisis de Malware en Android publicados entre los años 2017 a 2021 de los repositorios más destacados.

Las técnicas de análisis de Malware utilizados con mayor frecuencia son: el análisis estático basado en permisos y llamadas API; el análisis dinámico basado en comportamiento, basado en comportamiento de llamadas API y llamadas API-Imágenes; el análisis híbrido con los métodos unificados de la técnica estática y dinámica basados en anomalías y llamadas al sistema. Evalúan la información recolectada para realizar una detección óptima del Malware.

Los factores que inciden en las técnicas de análisis de Malware en Android son: el poco conocimiento de métodos y herramientas utilizadas para su detección. La evolución del Malware es capaz de usar técnicas de encriptación y ofuscación haciendo que su detección sea más compleja. Para un escaneo rápido y verificación de Malware se usa análisis estático, por su bajo consumo computacional, ya que no ejecuta el código. Para conocer el comportamiento del Malware en tiempo real se usa análisis dinámico, ya que es capaz de detectar Malware poliforme y de auto ocultación. Esta evolución ha ocasionado que análisis estático se vuelva obsoleto para las nuevas generaciones de Malware y que ciertos comportamientos ya evadan el análisis dinámico. Se concluyo que la unificación del análisis estático y análisis dinámico da como origen al análisis híbrido que corrige sus limitaciones, pero demandan un alto costo computacional y tiempo de ejecución.

En los 66 estudios relacionados detallan herramientas que facilitan la extracción de características del Malware, el uso de técnicas de aprendizaje como: supervisadas, semi supervisadas, no supervisadas y profundas mejoran las técnicas estáticas, dinámicas e híbridas. En nuestra investigación se evidenció que la métrica de mayor uso es Accuracy (AC), seleccionamos estudios en base a la clasificación especifícando el AC máximo, mínimo y realizamos un promedio con respecto a los datos identificados. Evidenciando que en el Análisis estático el AC: máximo = 99.80%, mínimo = 81.03%, promedio = 95.36%; Análisis dinámico AC: máximo = 98.80%, mínimo = 74.46%, promedio = 92.44%; Análisis híbrido AC: máximo = 99.22%, mínimo = 90.19%, promedio = 96.81%. Con lo que podemos concluir que el análisis híbrido cuenta con un promedio mayor en comparación de las otras técnicas.

References

[1] B. Denise, "Cómo funciona el malware dirigido a dispositivos móviles," ESET, Tech. Rep., Marzo 2020. [Online]. Available: https://www.eset.com/py/acerca-de-eset/sala-de-prensa/comunicados-de-prensa/articulos-de-prensa/como-funciona-el-malware-dirigido-a-dispositivosmoviles/

[2] M. Choudhary and B. Kishore, "Haamd: Hybrid analysis for android malware detection," in *2018 International Conference on Computer Communication and Informatics (ICCCI)*, 2018, pp. 1–4.

[3] A. Reina, A. Fattori, and L. Cavallaro, "A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors," *EuroSec*, pp. 1–6, 2013.

[4] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for iot malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96 899–96 911, 2020.

[5] L. G. De Almeida, A. D. D. Souza, B. T. Kuehne, and O. S. M. Gomes, "Data analysis techniques in vehicle communication networks: Systematic mapping of literature," *IEEE Access*, vol. 8, pp. 199 503–199 512, 2020.

[6] Y. Pan, X. Ge, C. Fang, and Y. Fan, "A systematic literature review of android malware detection using static analysis," *IEEE Access*, vol. 8, pp. 116 363–116 379, 2020.

[7] G. McGraw and G. Morrisett, "Attacking malicious code: A report to the infosec research council," *IEEE Software*, vol. 17, no. 5, pp. 33–41, 2000.

[8] H. Yuan, Y. Tang, W. Sun, and L. Liu, "A detection method for android application security based on tf-idf and machine learning," *PLoS ONE*, vol. 15, no. 9 September, 2020.

[9] L. Onwuzurike, M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, and E. De Cristofaro, "A family of droids-android malware detection via behavioral modeling: Static vs dynamic analysis," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, 2018, pp. 1–10.

[10] L. Duong, V. Tisenko, D. Long, N. Dam, and N. Hoang, "Detecting malicious applications on android is based on static analysis using deep learning algorithm," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 3, pp. 3494–3499, 2020.

[11] A. Firdaus, N. Anuar, A. Karim, and M. Razak, "Discovering optimal features using static analysis and a genetic search based method for android malware detection," *Frontiers of Information Technology and Electronic Engineering*, vol. 19, no. 6, pp. 712–736, 2018.

[12] A. Singh, C. Jaidhar, and M. Kumara, "Experimental analysis of android malware detection based on combinations of permissions and api-calls," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 3, pp. 209–218, 2019.

[13] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on autoencoders and api-images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26–33, 2020.

[14] W. Niu, R. Cao, X. Zhang, K. Ding, K. Zhang, and T. Li, "Opcode-level function call graph based android malware classification using deep learning," *Sensors (Switzerland)*, vol. 20, no. 13, pp. 1–23, 2020.

[15] A. Aminordin, M. Faizal, and R. Yusof, "Android malware classification base on application category using static code analysis," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 20, pp. 6853–6863, 2018.

[16] A. Kabakus, "What static analysis can utmost offer for android malware detection," *Information Technology and Control*, vol. 48, no. 2, pp. 235–249, 2019.

[17] M. Razak, N. Anuar, F. Othman, A. Firdaus, F. Afifi, and R. Salleh, "Bio-inspired for features optimization and malware detection," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 6963–6979, 2018.

[18] A. Mehtab, W. Shahid, T. Yaqoob, M. Amjad, H. Abbas, H. Afzal, and M. Saqib, "Addroid: Rule-based machine learning framework for android malware analysis," *Mobile Networks and Applications*, vol. 25, no. 1, pp. 180–192, 2020.

[19] A. Firdaus, N. Anuar, M. Razak, and A. Sangaiah, "Bio-inspired computational paradigm for feature investigation and malware detection: interactive analytics," *Multimedia Tools and Applications*, vol. 77, no. 14, pp. 17 519–17 555, 2018.

[20] M. Shohel Rana, C. Gudla, and A. Sung, "Evaluating machine learning models for android malware detection - a comparison study," 2018, pp. 17–21.

[21] X. Qin, F. Zeng, and Y. Zhang, "Msndroid: The android malware detector based on multi-class features and deep belief network," 2019.

[22] N. Milosevic, A. Dehghantanha, and K.-K. Choo, "Machine learning aided android malware classification," *Computers and Electrical Engineering*, vol. 61, pp. 266–274, 2017.

[23] J. Pye, B. Issac, N. Aslam, and H. Rafiq, "Android malware classification using machine learning and bio-inspired optimisation algorithms," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 1777–1782.

[24] R. Riasat, M. Sakeena, A. H. Sadiq, and Y.-J. Wang, "Onamd: An online android malware detection approach," in *2018 International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 1, 2018, pp. 190–196.

[25] S. Aonzo, A. Merlo, M. Migliardi, L. Oneto, and F. Palmieri, "Low-resource footprint, data-driven malware detection on android," *IEEE Transactions on Sustainable Computing*, vol. 5, no. 2, pp. 213–222, 2020.

[26] U. S. Jannat, S. M. Hasnayeen, M. K. Bashar Shuhan, and M. S. Ferdous, "Analysis and detection of malware in android applications using machine learning," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–7.

[27] N. J. Ratyal, M. Khadam, and M. Aleem, "On the evaluation of the machine learning based hybrid approach for android malware detection," in *2019 22nd International Multitopic Conference (INMIC)*, 2019, pp. 1–8.

[28] H. Zhou, X. Yang, H. Pan, and W. Guo, "An android malware detection approach based on simgru," *IEEE Access*, vol. 8, pp. 148 404–148 410, 2020.

[29] S. Alsoghyer and I. Almomani, "Ransomware detection system for android applications," *Electronics (Switzerland)*, vol. 8, no. 8, 2019.

[30] T. Lei, Z. Qin, Z. Wang, Q. Li, and D. Ye, "Evedroid: Event-aware android malware detection against model degrading for iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6668–6680, 2019.

[31] Y. Wu, X. Li, D. Zou, W. Yang, X. Zhang, and H. Jin, "Malscan: Fast market-wide mobile malware scanning by social-network centrality analysis," in *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '19. IEEE Press, 2019, p. 139–150.

[32] Z. Shan, I. Neamtiu, and R. Samuel, "Self-hiding behavior in android apps: Detection and characterization," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 728–739.

[33] A. Roy, D. Jas, G. Jaggi, and K. Sharma, "Android malware detection based on vulnerable feature aggregation," vol. 173, 2020, pp. 345–353.

[34] A. Appice, G. Andresini, and D. Malerba, "Clustering-aided multi-view classification: A case study on android malware detection," *Journal of Intelligent Information Systems*, vol. 55, no. 1, 2020.

[35] A. Salah, E. Shalabi, and W. Khedr, "A lightweight android malware classifier using novel feature selection methods," *Symmetry*, vol. 12, no. 5, 2020.

[36] Y. Ding, J. Hu, W. Xu, and X. Zhang, "A deep feature fusion method for android malware detection," in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, 2019, pp. 1–6.

[37] D. Dehkordy and A. Rasoolzadegan, "A new machine learning-based method for android malware detection on imbalanced dataset," *Multimedia Tools and Applications*, 2021.

[38] R. Thangaveloo, W. Jing, C. Leng, and J. Abdullah, "Datdroid: Dynamic analysis technique in android malware detection," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 10, no. 2, pp. 536–541, 2020.

[39] M. Alzaylaee, S. Yerima, and S. Sezer, "Dl-droid: Deep learning based android malware detection using real devices," *Computers and Security*, vol. 89, 2020.

[40] T. Kim, B. Kang, and E. Im, "Runtime detection framework for android malware," *Mobile Information Systems*, vol. 2018, 2018.

[41] P. Feng, J. Ma, C. Sun, X. Xu, and Y. Ma, "A novel dynamic android malware detection system with ensemble learning," *IEEE Access*, vol. 6, pp. 30 996–31 011, 2018.

[42] A. Mahindru and A. Sangal, "Mldroid—framework for android malware detection using machine learning techniques," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5183–5240, 2021.

[43] X. Wang and C. Li, "Kertsdroid: Detecting android malware at scale through kernel task structures," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 2019, pp. 870–879.

[44] L. Massarelli, L. Aniello, C. Ciccotelli, L. Querzoni, D. Ucci, and R. Baldoni, "Androdfa: Android malware classification based on resource consumption," *Information (Switzerland)*, vol. 11, no. 6, 2020.

[45] D. Thakur, J. Singh, P. Faruki, and T. Gera, "Classification of android malware using its image sections," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 6151–6155, 2020.

[46] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Emulator vs real phone: Android malware detection using machine learning," in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, ser. IWSPA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 65–72.

[47] H. Papadopoulos, N. Georgiou, C. Eliades, and A. Konstantinidis, "Android malware detection with unbiased confidence guarantees," *Neurocomputing*, vol. 280, pp. 3–12, 2018.

[48] G. Sethupathi, S. Siddharth, V. Kumar, P. Kumar, and A. Yadav, "Maldroid: Dynamic malware detection using random forest algorithm," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 6, pp. 311–315, 2019.

[49] L. Baird, Z. Shan, and V. Namboodiri, "Automated dynamic detection of self-hiding behavior," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, 2019, pp. 87–91.

[50] S. Iqbal and M. Zulkernine, "Spydroid: A framework for employing multiple real-time malware detectors on android," in *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)*, 2018, pp. 1–8.

[51] M. Yousefi-Azar, L. Hamey, V. Varadharajan, and S. Chen, "Byte2vec: Malware representation and feature selection for android," *Computer Journal*, vol. 63, no. 8, pp. 1125–1138, 2020.

[52] W. Zhang, H. Wang, H. He, and P. Liu, "Damba: Detecting android malware by orgb analysis," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 55–69, 2020.

[53] M. Yusof, M. Saudi, and F. Ridzuan, "Mobile botnet classification by using hybrid analysis," *International Journal of Engineering and Technology(UAE)*, vol. 7, no. 4, pp. 103–108, 2018.

[54] R. Kumar, X. Zhang, R. Khan, and A. Sharif, "Research on data mining of permission-induced risk for android iot devices," *Applied Sciences (Switzerland)*, vol. 9, no. 2, 2019.

[55] C.-H. Yung and W.-S. Juang, "Static and dynamic integrated analysis scheme for android malware," *Journal of Electronic Science and Technology*, vol. 15, no. 3, pp. 246–250, 2017.

[56] G. Tao, Z. Zheng, Z. Guo, and M. Lyu, "Malpat: Mining patterns of malicious and benign android apps via permission-related apis," *IEEE Transactions on Reliability*, vol. 67, no. 1, pp. 355–369, 2018.

[57] D. Saif, S. El-Gokhy, and E. Sallam, "Deep belief networks-based framework for malware detection in android systems," *Alexandria Engineering Journal*, vol. 57, no. 4, pp. 4049–4057, 2018.

[58] R. Surendran, T. Thomas, and S. Emmanuel, "A tan based hybrid model for android malware detection," *Journal of Information Security and Applications*, vol. 54, 2020.

[59] S. Garg and N. Baliyan, "A novel parallel classifier scheme for vulnerability detection in android," *Computers and Electrical Engineering*, vol. 77, pp. 12–26, 2019.

[60] Y.-C. Shyong, T.-H. Jeng, and Y.-M. Chen, "Combining static permissions and dynamic packet analysis to improve android malware detection," in *2020 2nd International Conference on Computer Communication and the Internet (ICCCI)*, 2020, pp. 75–81.

[61] A. Arora and S. K. Peddoju, "Ntpdroid: A hybrid android malware detector using network traffic and system permissions," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 808–813.

[62] C. C. U. López, J. S. D. Villarreal, A. F. P. Belalcazar, A. N. Cadavid, and J. G. D. Cely, "Features to detect android malware," in *2018 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2018, pp. 1–6.

[63] S. J. Hussain, U. Ahmed, H. Liaquat, S. Mir, N. Jhanjhi, and M. Humayun, "Imiad: Intelligent malware identification for android platform," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, 2019, pp. 1–6.

[64] B. Yu, P. Song, and X. Xu, "An android malware static detection scheme based on cloud security structure," *International Journal of Security and Networks*, vol. 13, no. 1, pp. 51–57, 2018.
[65] S. Alam, Z. Qu, R. Riley, Y. Chen, and V. Rastogi, "Droidnative: Automating and optimizing detection of android native code malware variants," *Computers and Security*, vol. 65, pp. 230–246, 2017.
[66] V. Kouliaridis, G. Kambourakis, D. Geneiatakis, and N. Potha, "Two anatomists are better than one-dual-level android malware detection," *Symmetry*, vol. 12, no. 7, 2020.
[67] D. Chaulagain, P. Poudel, P. Pathak, S. Roy, D. Caragea, G. Liu, and X. Ou, "Hybrid analysis of android apps for security vetting using deep learning," in *2020 IEEE Conference on Communications and Network Security (CNS)*, 2020, pp. 1–9.
[68] A. Ali-Gombe, B. Saltaformaggio, J. Ramanujam "Ram", D. Xu, and I. Richard, G.G., "Toward a more dependable hybrid analysis of android malware using aspect-oriented programming," *Computers and Security*, vol. 73, pp. 235–248, 2018.
[69] F. Tong and Z. Yan, "A hybrid approach of mobile malware detection in android," *Journal of Parallel and Distributed Computing*, vol. 103, pp. 22–31, 2017.
[70] A. Jahromi, S. Hashemi, A. Dehghantanha, R. Parizi, and K.-K. Choo, "An enhanced stacked lstm method with no random initialization for malware threat hunting in safety and time-critical systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 630–640, 2020.
[71] Y.-S. Yen and H.-M. Sun, "An android mutation malware detection based on deep learning using visualization of importance from codes," *Microelectronics Reliability*, vol. 93, pp. 109–114, 2019.
[72] O. Faruk Turan Cavli and S. Sen, "Familial classification of android malware using hybrid analysis," in *2020 International Conference on Information Security and Cryptology (ISCTURKEY)*, 2020, pp. 62–67.
[73] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "Samadroid: A novel 3-level hybrid malware detection model for android operating system," *IEEE Access*, vol. 6, pp. 4321–4339, 2018.