Available online at sjuoz.uoz.edu.krd

Science Journal of University of Zakho
Vol. 11, No. 4, pp. 469– 480, October-December, 2023

journals.uoz.edu.krd

p-ISSN: 2663-628X
e-ISSN: 2663-6298

GUARDING ANDROID: A COMPREHENSIVE REVIEW OF INTRUSION DETECTION TECHNIQUES FOR SMARTPHONES

Ibrahim Mahmood Ibrahim [a*] , Amira Bibo Sallow [b]

[a] Department of Information Technology, Technical College of Informatics-Akre, Duhok Polytechnic University, Duhok, Iraq
ibrahim.mahmood@dpu.edu.krd

b Department of Information Technology, Technical College of Duhok, Duhok Polytechnic University, Duhok, Iraq
amira.bibo@dpu.edu.krd

Received: 26 May, 2023 / Accepted: 2 Aug., 2023 / Published: 15 Oct., 2023. https://doi.org/10.25271/sjuoz.2023.11.4.1156

ABSTRACT:
The popularity of using the Android operating system has increased the number of developers and intruders in this field. Many applications are developed in this area and perform malicious activities like ransomware attacks, installing backdoors, phishing, sending premium short message service, and stealing private data. These activities pose many threats to smartphone users. This study provides a review of the main strategies used in intrusion detection systems to detect malicious activities at the application and system levels. The study illustrates the advantages and disadvantages of each method and the significant features used to discriminate against malicious activities and highlights several open issues that warrant further investigation and improvement. It is a comprehensive review that may be useful for academic researchers interested in cybersecurity.

KEYWORDS: Intrusion Detection System, Malware detection, Static features, Dynamic features, Hybrid features

1. INTRODUCTION

Smartphones play an essential role in our daily lives. They offer a variety of enticing features that allow mobile users to make use of a variety of high-quality customized services (Ribeiro, Saghezchi, Mantas, Rodriguez, & Abd-Alhameed, 2020). Some of the services provided by smartphones are voice and video communication, e-mail addresses, internet browsing, online shopping, banking, and many other functions (Radoglou-Grammatikis & Sarigiannidis, 2017). Many operating systems are available for smartphone devices. Among them, Android is the most widely used (Agrawal & Trivedi, 2019). The popularity of the Android system has opened the door for attackers to increase the number of threats to Android devices (da Costa et al., 2020). According to the McAfee Labs Threats' report in the first quarter of 2020, 98% of attackers target Android devices (Bayazit, Sahingoz, & Dogan, 2020).

Android is a Linux-based operating system that comes with a number of useful applications and middleware. Google allows third-party developers to build applications and release them to the Android Market in order to fully utilize and explore the capabilities of Android (Hein & Myo, 2018). The hacker takes advantage of Android application capabilities to compromise device security and privacy, posing a major risk of personal data leakage such as the user's location, contact information, accounts, images, and so on (R. Kumar et al., 2019).

Many levels of security tools exist to protect against cyberattacks, like firewalls, anti-virus, and Intrusion Detection system (IDS) (Uğurlu & Doğru, 2019). Firewalls primarily control access between networks and do not produce any signals if an attack occurs internally (Alqahtani et al., 2020). Antivirus is effective for detecting known malware and ineffective for detecting unknown malware. It needs to constantly update its databases to be effective and detect new samples of malware. IDSs come in two main categories: signature-based and anomaly-based detection. Signature detection identifies suspicious behaviour by matching it to an attack signature that has already been saved in a database. This type is less effective for detecting malicious activities, especially for new attacks, because they need to constantly update their signature databases. Anomaly detection considers normal behaviour to be a model and attempts to detect any deviation from the model before deciding whether or not to create an alarm. An anomaly-based IDS is better suited for detecting new malware attacks (Elkhadir, Chougdali, & Benattou, 2016). The major aim of an IDS is to detect various types of malwares like botnets, Trojans, spyware, backdoors, worms, ransomware, and riskware as quickly as possible, which is unachievable with a regular firewall (Zachariah, Akash, Yousef, & Chacko, 2017; Khraisat, Gondal, Vamplew, & Kamruzzaman, 2019).

Anomaly-based IDS employs three techniques to analyse and understand the behaviour of the malware, which are static, dynamic, and hybrid analysis. The static analysis technique decompiles an Android application package to obtain two main files (i.e., manifes.xml and classes. dex) and then starts to extract static features from these two files without executing them. Many features can be extracted from static analysis, such as permissions, intents, opcodes, used features, and API calls (Malik). The dynamic analysis technique is used to identify malicious behaviour when the application is running by extracting many features like system calls, network traffic, or hardware features utilization like CPU, memory, and battery. The hybrid technique combines the benefits of dynamic and static analysis (Bayazit et al., 2020; H. Zhou, Yang, Pan, & Guo, 2020).

Machine learning techniques are commonly used to detect Android malware, whether using static, dynamic, or hybrid analysis methods. Machine learning-based malware detection has the capability to detect previously unseen kinds of malware and can provide better detection and efficiency than traditional methods, such as signature-based malware detection, which is

* Corresponding author

based on identifying the specific patterns of known malware (K. Liu et al., 2020).

This paper addresses the crucial issue of Android device security in the face of increasing threats and attacks. With smartphones playing an essential role in our daily lives and Android being the most widely used operating system, it has become a prime target for attackers. Anomaly-based IDSs offer a promising approach by considering normal behaviour as a model and detecting any deviations from it. This paper focuses on the analysis of malware behaviour using static, dynamic, and hybrid techniques and leverages machine learning to enhance malware detection. The findings of this study have the potential to enhance Android device security and mitigate the risks associated with personal data leakage and privacy breaches.

The remaining sections of our paper are organized as follows. Section 2 provides background information about Intrusion Detection Systems (IDS). Section 3 presents literature review in the field. Section 4 contains a discussion and comparison of our findings, followed by Section 5 which explores the challenges and recommendations associated with the topic. Finally, in Section VI, the points of conclusion arrived at throughout the study are presented.

2. INTRUSION DETECTION SYSTEM

Intrusion detection systems (IDS) monitor and analyse network logs, file system activity, and real-time events in the local system to detect cyber-attacks (P. Liu, 2019). Another definition of IDS is a new type of security technology that monitors a system for harmful activity (Borkar, Donode, & Kumari, 2017). An IDS can usually perform a variety of tasks, such as information monitoring and recording, security officer notification, and report generation. In general, smartphones consist of hardware, an operating system, and applications. An IDS system can monitor either an event at the application level, the kernel level, the hardware level, or a hybrid of them. The IDS system can work to detect malicious activities generated by malware apps like damaging file systems, information leakage, or any other types of threats. There are many classifications for IDS based on topology, source, approach, and focus. The details of the classification are illustrated in Figure 1 (Georgios Kambourakis, 2018).

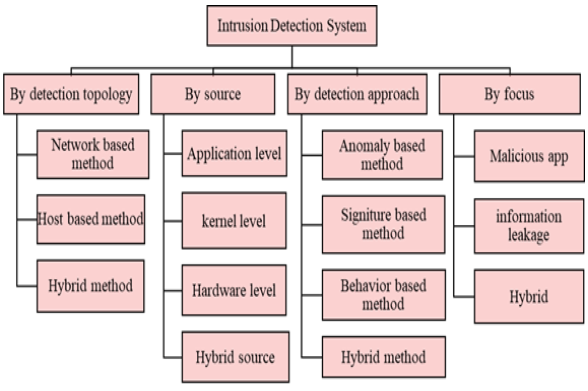

Figure 1. IDS Classification

2.1. Detection approach

The detection method applied to identify intrusion detection systems (IDS) is classified to Signature-based and anomaly-based detection approaches. The misuse (signature) detection method maintains a database of predefined patterns and uses that database to track the usage of those patterns (BalaGanesh, Chakrabarti, & Midhunchakkaravarthy, 2018). Signatures are hashes created by many algorithms like MD5 and SHA256 for previously seen malicious applications. These hash values are saved in a database and can be used for scanning applications. During the scanning process, if the hash of the scanned application is found in the database, this application is marked as malicious (Borek, 2017). The advantage of misuse-based IDS is that it is capable of effectively recognizing known attacks, but it fails to detect unknown attacks. In order to store the signature of each known attack, a database must be persistently updated (Istiaque, Khan, & Waheed, 2020). In the anomaly method, a basic profile of a regular network or system activity is generated and then every incoming packet or event which differs from the profile is treated as an intrusion. The advantage of this method is that it is capable of detecting new attacks (Shamshirband et al., 2020). The problem with this strategy is that if a normal application makes more system calls, it may be classified as malware; and compared to signature-based IDS, anomaly-based IDSs are more computationally expensive (Chawla, Lee, Fallon, & Jacob, 2019).

2.2. Detection topology

Intrusion detection systems vary in design depending on the user's perspective. In general, they can be classified into host-based IDS and network-based IDS. In Host IDS (HIDS), the software installed on the host device helps to prevent malicious cyber-attacks by systems. One strategy is to detect normal system behaviour based on system call sequences performed by system processes (Chawla et al., 2019). An IDS based on the network (NIDS) monitors network traffic for suspicious, abnormal, or unauthorized activities that may result in a cyber-attack. Intrusion detection based on the network is based on the knowledge acquired. The methods for detecting such actions are based either on signatures (misuse detection) or behaviour-based (anomaly detection). Systems based on signatures detect attacks that are programmed to notify, while systems based on behaviour detect deviations from the usual behaviour profile and are able to detect unauthenticated attacks (S. Kumar, Viinikainen, & Hamalainen, 2016).

3. LITERATURE REVIEW

In 2011, smartphone manufacturers distributed around 450 million devices. In the same period, mobile malware spread quickly, leading to the development of many IDSs with the purpose of performing malware detection (Georgios Kambourakis, 2018). In this study, focus is on the intrusion detection framework related to the smartphone-based Android operating system at the application and system level and the analysis method used by researchers.

3.1. Malware IDS-based static features

Static analysis is used to detect malicious applications without requiring them to be run or executed on smartphones (Gyamfi & Owusu, 2018; Khariwal, Singh, & Arora, 2020). Static analysis is primarily concerned with analyzing the basic files of an Android application instead of executing them, which allows them to execute relatively quickly. The "AndroidManifest.xml" and "classes.dex" files are the two most common files in the static analysis process, from which representative features like permissions, intents, opcodes, API calls, and functional call graphs can be extracted, respectively (Jannat, Hasnayeen, Shuhan, & Ferdous, 2019; Lei, Qin, Wang, Li, & Ye, 2019). The process of static analysis is illustrated in Figure 2.

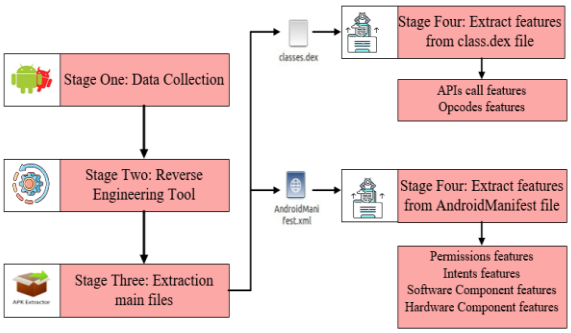

Figure 2. Process of static analysis

3.2. Manifest features

Analysing the Android manifest file is a common technique used in Android malware detection. The Android manifest file is an essential component of an Android application that contains important information about its functionalities and permissions. By examining this file, researchers can gain insights into the behaviour and potential risks associated with an app. The analysis of the Android manifest file involves extracting and inspecting various elements, such as permissions requested by the app, declared components (activities, services, and receivers), intent filters, and other metadata. This information helps in understanding the app's intended functionality and potential interactions with the device and other apps. Some malicious apps exploit these features to engage in unauthorized or harmful activities. The researchers built many machine learning and deep learning models for the purpose of detecting malicious activities resulting from mobile applications. In their studies, they employed single-feature category and multi-feature categories. Khatter (2018) presented an intrusion detection system for classifying malicious applications based on permission analysis. Their system involves feature extraction, machine learning training, and model performance evaluation using a testing dataset. The study evaluates different machine learning algorithms, with kernel logistic regression achieving 98.2% accuracy for malware detection and LibLinear achieving 87.83% accuracy for classifying 81 malware families.

Sirisha and Anuradha (2019) introduced a sequential neural network model for predicting the presence of malware in Android APK files obtained from the web and Play Store. The model was trained on a dataset of 398 APK files with 331 features, and during testing, it was evaluated using data from the Play Store and malicious sites like Droidbench, which contained both malicious and benign APK files. The proposed method achieved an accuracy of over 85% in real-time detection of malicious applications based on permission extraction from the APK.

The researchers Kapoor, Kushwaha and Gandotra (2019), Alsoghyer and Almomani (2020), and Mathur, Podila, Kulkarni, Niyaz, and Javaid (2021) built and trained many machine-learning models based on permissions for detecting malware apps on Android devices, and all studies achieved accuracy above 96%. The researcher (Sandeep, 2019) presents a novel approach by developing a fully connected deep learning model for the detection of malicious applications, utilizing permission features. One distinguishing aspect of this study is the inclusion of additional details, such as application name and version, during the malware detection process.

Dharmalingam and Palanisamy (2021) proposed the Permission Grading System (PGS) to extract requested permissions from applications and detect the permissions that are special to malware and benign apps, with the contribution of each permission calculated. The presented model is effective only when malware and benign applications have different sets of permission.

Amer (2021) introduced a mechanism for analysing Android applications based on permission combinations indexed within the app. To improve the performance of the proposed model, multiple machine-learning algorithms were combined to create an ensemble model.

Analysing the intent filters declared by an app can help identify potential misuse. Malware authors may register intent filters to capture sensitive intents or intercept system-level events for malicious purposes. For instance, Sewak, Sahay and Rathore (2020) developed a method named Deep-Intent, which is an online Intrusion Detection System (IDS) that uses an E2E DL implementation for both supervised learning and unsupervised feature engineering, and only uses implicit intent as a feature. The experiment findings reveal that the presented intent-based IDS could detect malware application software with an AUC of 81% and an accuracy of 77.2%.

Khariwal et al. (2020), Jiang, Mao, Guan and Huang (2020), and Sangal and Verma (2020) have proposed robust models for detecting malicious applications by combining permissions and intents along with various machine learning algorithms. Khariwal et al. (2020) and Jiang et al. (2020) used an information gain algorithm to find the best subset of combined intents with permissions. Whereas Sangal and Verma (2020) utilized the PCA algorithm for feature reduction. In addition to permissions and intents, Y. Zhang, Feng, Huang, Ye and Weng (2020) added two other features, such as hardware and app component features, to their model for the prediction of the Android malware family based on the Driben dataset.

3.3. Classes or source code features

The classes.dex file contains the compiled bytecode of the app's classes and holds valuable information about the app's functionality. Usually, researchers extract two main feature categories from the class, which are API calls and opcodes. Analysing these features can help detect suspicious or malicious activities, such as accessing sensitive system resources, making unauthorized network requests. Zhao, Li, Zheng and Shi (2018) and Niu et al. (2020) extracted opcode sequences from the bytecode or classes of the Android app and utilized them as input for a deep learning model in order to detect Android malware. Both studies trained on small datasets and got high scores.

Ma, Ge, Liu, Zhao and Ma (2019) developed a novel approach based on the API call to identify malicious code in the Android system. They proposed a selection approach for API features relevant to malware, explored the structure interactions among these APIs, and developed a CNN classifier model for the classification of the API features. A real-world dataset with 3,697 malicious apps and 3,312 benign apps showed that the API features were efficient in classifying Android malware. H. Zhang, Luo, Zhang and Pan (2019) proposed a new malware detection method for Android applications based on correlation relationships among abstracted API calls. The source code is divided into function methods and association rules mining is used to define an application in computational semantics. Machine learning algorithms are used to distinguish benign and malicious.

There are many studies that combine the features from manifests and classes for the purpose of increasing the accuracy of the model. Lê, Nguyen, Truong, Nguyen and Ngô (2020) proposed a method of machine learning to identify Android malware apps. The features that are used to train machine learning are built based on behaviour, requisite permissions, and other features of malicious applications. The findings achieved an accuracy of 98.66% with a set of 28879 samples containing malicious and benign apps. Tiwari and Shukla

(2018) proposed a system for detecting malware in mobile devices based on two common types of features: permission and API calls using machine learning algorithms. Due to the limited resources of the mobile device, the proposed system reduced the number of features in the dataset to 30 features using the PCA algorithm.

Singh, Wadhwa, Ahuja, Soni and Sharma (2020) proposed model uses Latent Semantic Indexing (LSI) to reduce the representation of opcodes in a lower-dimensional space, allowing the system to work with a smaller set of opcodes. Additionally, permissions and intents are added to the feature set to improve the performance of the model classifiers. The authors combine permissions with API call detecting ransomware application. Almomani et al (2021) developed an approach for detecting ransomware by combining permissions with API calls. The method employed the SVM approach for the classification while addressing the imbalance between benign and ransomware apps in the dataset through the use of the Synthetic Minority Oversampling Technique (SMOTE). This method offers a promising solution for accurately identifying ransomware based on its distinctive permissions and API patterns, providing enhanced security measures against this prevalent threat.

R. Kumar et al. (2019), Yerima and Alzaylaee (2020), and Esmaeili and Shahriari (2019) focused on combining manifest features with source code features to train deep learning models, including multimodal deep neural networks and CNNs. However, it is worth noting that these studies employed unbalanced datasets during the training process. Yerima and Alzaylaee (2020), Esmaeili and Shahriari (2019) focused on only one type of malware, which is botnets. Further, a framework based on the stacking approach is proposed by Xie, Qin and Di (2023). The proposed approach comprises three parts: dataset creation, feature reduction, and optimization method GA-StackingMD. The implementation of a stacking model comprising five base classifiers has resulted in a significant enhancement in detection accuracy when compared to the use of individual classifiers.

All the aforementioned studies on static analysis have a common limitation in that they solely relied on computer-based examination processes and did not involve the actual use of mobile devices. This limitation raises concerns about the real-world applicability and performance of the developed models for detecting malware on mobile devices. Feng, Liu and Lin (2019) introduced the MobiDroid system, which comprises two components for detecting malicious applications on mobile devices. The server component generates feature dictionaries and trains deep neural networks, while the mobile device component utilizes the dictionaries and trained model for on-device malware detection. The approach takes into account the performance limitations of Android devices, allowing users to balance classification accuracy and overall cost. Yuan, Jiang, Li and Cai (2019) developed an on-device lightweight Android malware detector based on the broad learning method. The presented system detector uses primarily one-shot model training calculation. It can therefore be trained directly or progressively on mobile devices. The model can be increased further via on-device model retraining.

The analysis of the Android manifest file and source code features has proven to be a common and effective technique in Android malware detection. These files contain crucial information about an application's functionalities and provide valuable insights into an app's behaviour and potential risks. Researchers have built numerous machine learning and deep learning models based on single-feature and multi-feature categories and got promising results. Despite these promising results, it is essential to address certain limitations. Most of the studies rely solely on computer-based testing, without considering the actual use of mobile devices. Ensuring real-world applicability and performance of the developed models on mobile devices remains a critical consideration for practical deployment.

Based on specific parameters, Table 1 represents the summary of the previous works that are based on static analysis.

Table 1. Summary of the papers based on static analysis

Reference	Dataset Sources	No. of Sample	Method	Analysis Tool	Feature Type	Acc. %	Pre. %	Recall %	F1 score%
(Khatter, 2018)	Drebin, Androtracker	B 533 M 527	NB,KLR,SLR,RBF, SMO, PART, CNN, SVM	AndroData Weak tool	Permission	98.2	---	---	---
(Zhao et al., 2018)	Drebin, Chinese app	B 1200 M 1200	CNN	APK Tool	Opcode	99.07	99.12	99.11	99.12
(Tiwari & Shukla, 2018)	GP, PRAGaurd	B 652 M 669	PCA and chi Square test for feature reduction and SVM for detection	AXMLPrinter2.jar	permission, API call	94.69	---	---	---
(R. Kumar et al., 2019)	GP,VirusShare, Malgenome	B 21047 M 14284	multimodal deep neural network	APK Tool	permission, API call, Opcode, others	98	---	98.2	---
(Sirisha & Anuradha, 2019)	---	398 for B and M	Sequential neural network	Androguard	Permission	85	---	---	---
(Kapoor et al., 2019)	GP , VirusShare and other trusted sites	B 1500 M 2500	LR, LDA, KNN, SVM, GNB, DT	python script	Permission	99.34	99.4	---	99.5
(Sandeep, 2019)	GP, Virsushre	---	RF for feature Selection DL for classification	Androguard	Permission	94.65	---	---	---

(Ma et al., 2019)	androzoo, AMD	B 10010 M 10683	DNN with LSTM	FLOWDROID	API call	---	99.15	98.82	98.98
(H. Zhang et al., 2019)	Drebin. AMD and androzoo	B 26464 M 26403	SVM, KNN,RF for classification and ARM for feature reduction	Andguard	API call	96	97	95	96
(Esmaeili & Shahriari, 2019)	Drebin	---	NB,DT,KNN	Androbug and MobSF tool	permission, API call, Intent, Network, Hardware, others	96	9.6	81.4	88.3
(Feng et al., 2019)	GP, Drebin, Genome,virusshare	B 21499 M 21499	CNN	ApkTool, AndroGuard, Soot, FlowDroid	permission, API call, Opcode	97	96.87	96.87	---
(Yuan et al., 2019)	AndroZoo, VirusShare and Contagio	B 33655 M 22221	BL	AXMLPrinter baksmali	permission, API call, Intent	94.71	92.4	89.61	93.42
(Khariwal et al., 2020)	Genome Drebin and Koodous	B 1414 M 1714	SVM, NB, RF for detection and IG for feature selection	Apktool	permission, Intent	94.37	---	---	---
(Alsoghyer & Almomani, 2020)	HelDroid, RansomProber ,Virus Total and Koodous	B 500 M 500	RF, J48, SMO, NB	Apk tool Weka tool	Permission	96.9	---	---	---
(Dharmalingam & Palanisamy, 2021)	GP, AAGM, AMD	B 1005 M 1592	Proposed PGS, SVM, DT, DNN, TF-IDF for feature Reduction	APKParser tool, XML-DOM	Permission	94.22	---	93.4	---
(Sewak et al., 2020)	GP and Drebin	B 5721 M 5560	SAE and MLP DNN	APKTOOL	Intent	77.2	---	---	---
(Jiang et al., 2020)	Xiaomi App Store, MalGenome and Andro-MalShare	B 1700 M 1600	SVM, KNN, NB, J48) for detection and IG for feature selection	Apktool Weka	permission, Intent	---	---	94.5	94.5
(Sangal & Verma, 2020)	CICInvesAndMal2019	B 1126 M 396	RF, KNN, NB, DT, SVM for classification and PCA For feature Selection	Weka	permission, Intent	96.05	96	96.1	96
(Y. Zhang et al., 2020)	Drebin	M 5554	LR , GBDT, RF	Dex.-jar	permission, Intent, Hardware, others	97.98	98.51	98.37	98.47
(Niu et al., 2020)	Virusshare , AndroZoo and Peapod.	B 1000 M 1796	LSTM	FlowDroid, drizzleDumper and FUPK3	Opcode	97	97	97	97
(Lê et al., 2020)	GP, Virusshare	B 12290 M 16589	NB,RF,DT, Gradient Boosting and AdaBoost	Dexdunp	permission, API call	98.66	---	---	---
(Singh et al., 2020)	CICInvesAndMal2019 Android Botnet	B 1147 M 905	LSI, SVM, LR, RF, KNN	Androguard	Permissions, Intents, opcodes	93.92	96.89	88.64	92.58
(Yerima & Alzaylaee, 2020)	ISCX botnet dataset	B 6803 M 1929	CNN	Developed by the author	permission, API call, Intent, others	98.9	98.3	97.8	98.1
(Mathur et al., 2021)	AndroZoo, Virus total, AMD	B 14730 M 14700	KNN,RF, LR, SVM,ET, XG,AB,BG	Androguard	Permission	96.95	---	---	---
(Amer, 2021)	Drebin MalGenome	B 9476 M 5560	Ensemble Model(RF, MLP, AdaBoost, SVM, DT)	---	Permission	99.3	98.8	99.3	99
(Almomani et al., 2021)	GP,VirusTotal, Koodous, RansomProper Project	B 9653 M 500	PSO, SVM and SMOTE	APK Tool	permission, API call	---	---	96.4	---

(Xie et al., 2023)	CICMalDroid2020	B 13204 M 4039	GA-StackingMD (SVM, KNN, LGBM, CatBoost, and RF)	Androguard	permission, API call, Opcode, Intent, others	98.66	99.15	99.06	99.1

3.4. Malware IDS-based dynamic features

In the dynamic approach, the application is executed in a controlled environment, monitors the running code, and inspects its interaction with the system. The dynamic analysis traces many features of an application and system, such as system calls, system components, network traffic, CPU consumption, memory usage, battery usage, and user interaction with the system (Painter & Kadhiwala, 2018). The system can detect malware in runtime with this approach (Riasat, Sakeena, Sadiq, & Wang, 2018). The process of dynamic analysis is illustrated in Figure 3.

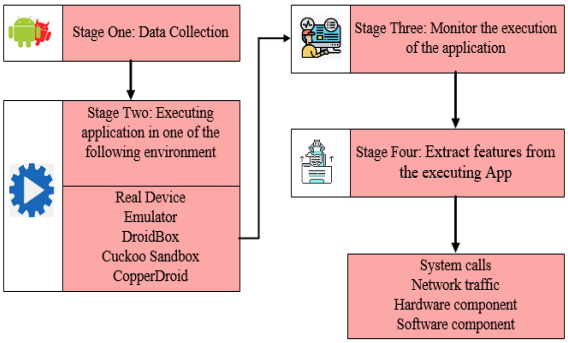

Figure 3. Dynamic analysis process

3.5. System calls features

Monitoring system calls made by an app during execution can reveal suspicious activities, such as file manipulation, network communication, or attempts to escalate privileges. Unusual or unauthorized system calls may indicate the presence of malware. Xiao, Zhang, Mercaldo, Hu and Sangaiah (2019) developed a novel classifier using Long Short Term Memory (LSTM) networks and system call sequences to detect malware on Android operating systems. The classifier consists of two LSTM networks trained on malware and legitimate applications, respectively. When a new sequence is encountered, similarity scores from the two networks are computed to classify the associated application. Q. Zhou et al. (2019) developed a novel malicious software detection system, introducing an innovative machine-learning classification algorithm. Unlike previous versions, this system utilizes the Monte Carlo technique to randomly adjust weights, enhancing optimization based on structural risk minimization and design risk reduction objectives. Comparative analysis with nine well-known machine learning algorithms revealed that the proposed system achieved an accuracy rate of 97.85% and higher accuracy than all compared algorithms.

According to Abderrahmane, Adnane, Yacine and Khireddine (2019) and X. Zhang et al. (2022), a client-server architecture was employed to analyze and classify the execution state of Android applications as either malware or benign. The mobile apps are sent to a remote server via a user-friendly interface. It would be installed and run in a simulation of a human user's interaction with it. After the execution of an application, the system calls produced by the Linux kernel are collected, analyzed, and fed into a neural network model that will be used to determine if the analyzed applications are malware or benign.

John, Thomas and Emmanuel (2020) proposed a new malware detection mechanism for Android with Graph Convolutional Network (GCN). With GCN, a four-dimensional representation of Android applications with centralizing graph measures as features are produced. The proposed method obtained a four-dimensional feature representation for Android applications, as well as a detection accuracy of 92.3% on datasets including malware that had been obfuscated by the attacker. Manzil (2022) presented a dynamic behaviour analysis for detecting malicious Android applications. The CICMalDroid2020 dataset was employed in the proposed study, whereby the system calls of applications were extracted via dynamic analysis. Several machine learning algorithms were trained on the dynamic features (system calls). Mahdavifar, Kadir, Fatemi, Alhadidi and Ghorbani (2020), Bansal, Baliyan and Ghosh (2022), and AlOmari, Yaseen and Al-Betar (2023) have incorporated system calls along with other features, including binders and composite behaviours, to identify the behavior of Android applications. These studies utilized the CICMalDroid2020 dataset to analyse APK files within a controlled environment.

3.6. Hardware and network features

By monitoring the behaviour of the device's hardware components and examining the data exchanged over the network, suspicious activities and potential malware can be identified. In hardware analysis, abnormal behaviour such as excessive battery usage or unauthorized access to device features like the camera can indicate the presence of malware. Network traffic analysis involves inspecting the data transferred between the device and external servers, looking for patterns associated with malware, such as communication with command-and-control servers or unauthorized data transfers.

Ariyapala, Do Hoang, Huynh, Wee and Conti (2016), Ribeiro et al. (2020), and Barbhuiya, Kilpatrick and Nikolopoulos (2020) have developed intrusion detection systems (IDS) aimed at detecting malware on mobile devices, utilizing a combination of hardware and network features. The IDS proposed by Ariyapala et al. (2016) collects host-based data, including CPU utilization, battery consumption, process execution, user activity, and network traffic. This data is then transmitted to a remote server for analysis, enabling the identification of malicious activity on the smartphone. On the other hand, Ribeiro et al. (2020) and Barbhuiya et al. (2020) focus on one-class classification models trained on standard usage patterns of CPU, battery, memory, and network traffic. These models raise intrusion alarms when significant deviations from the regular patterns are detected.

Radoglou-Grammatikis and Sarigiannidis (2017) developed intrusion detection system (IDS) aims to identify abnormal actions on mobile Android devices by continuously monitoring network activity and collecting

NetFlows features. An artificial neural network (ANN) processes the collected data streams to determine the presence of an attack. While Wang et al. (2019) proposed a lightweight malware identification framework for Android systems, where network traffic from the mobile app is redirected to a server for analysis, reducing resource utilization on the device.

Android malware detection based on dynamic features in most previous studies got promising results. However, several limitations need to be addressed. Some of the proposed approaches involve transmitting data to remote servers, which raises privacy issues about private user data. Additionally, these methods might not work in situations when the internet connection is slow or unavailable. Some approaches may face challenges in real-time detection, as the analysis of system calls may require time-consuming computations, leading to delayed responses to emerging threats. Studies based on monitoring network traffic and hardware resources can introduce additional overhead, affecting the device's performance and battery life. Table 2 represents the summary of the previous works that are based on dynamic analysis.

Table 2. Summary of the papers that based on dynamic analysis

Reference	Dataset Sources	No. of Sample	Method	Analysis Tool	Feature Type	Acc. %	Pre. %	Recall %	F1 score%
(Ariyapala et al., 2016)	Created by Author	---	Markov Chain	WireShark	Network traffic, CPU, Battery, running process and services	---	---	---	---
(Xiao et al., 2019)	Drebin, Google Play	B 3567 M 3536	LSTM	Monkey and Strace	System call	93.7	91.3	96.6	---
(Radoglou-Grammatikis & Sarigiannidis, 2017)	CTU-13 dataset	---	MLP	---	Network traffic	85	---	---	---
(Q. Zhou et al., 2019)	Baidu Mobile Application market and VirusShare	B 920 M 379	Monte Carlo algorithm.	---	System call	97.85	---	98.7	---
(Wang et al., 2019)	Drebin	B 8321 M 5560	C4.5	TcpDump tool	Network traffic	97.89	---	---	---
(Abderrahmane et al., 2019)	ArgusLab, Darwin project	B 2450 M 10300	CNN	Android emulator, Monkey Tool, Strace Tool	System call	93.3	94.1	97.8	96
(Ribeiro et al., 2020)	Run Time Dataset Generation	B 1200 sample	One class K-mean and univariate Gaussian	---	Network traffic, CPU, Battery, Memory, running process and services	91	---	85.7	---
(John et al., 2020)	GP,Deribn, AMD and Malgenome	B 1410 M 720	Graph Convolutional Nets (GCN)	Emulator, Monkey Tool, Strace Tool, Weka	System call	92.3	91.5	93.3	92.3
(Mahdavifar et al., 2020)	CICMalDroid2020	B 1795 M 9803	DNN	CopperDroid	System call, Binder, Composite behaviour	96.7	99.16	96.54	97.84
(Barbhuiya et al., 2020)	Run Time Dataset Generation	---	One class classification and probability distribution	Emulator	Network traffic, CPU	93.3	---	---	---
(X. Zhang et al., 2022)	AMD, Google play	B 300 M 125	MLP	Monkey runner, trace tool, ADB tool	System call	99.34	99.12	99.07	99.44
(Manzil, 2022)	CICMalDroid2020	B 1795 M 1253	RF, DT, LR, SVM, and AdaBoost	Monkey runner, trace tool, ADB tool	System call	99.5	99.5	99.5	99.5
(Bansal et al., 2022)	CICMalDroid2020	B 1,795 M 9803	Light Gradient Boosting	Copper-Droid	System call, Binder, Composite behavior	98.01	99.19	98.39	98.79
(AlOmari et al., 2023)	CICMalDroid2020	B,M 11598	Light Gradient Boosting	CopperDroid	System call, Binder, Composite behavior	95.49	95.48	94.7	95.47

* Corresponding author

3.7. Malware IDS-based hybrid features

Malware IDS-based hybrid features are characterized by the combination of diverse feature types within an Intrusion Detection System (IDS) framework, aimed at the identification and detection of malware. These hybrid features integrate various sources of information, including manifest files, source codes, system calls, network traffic patterns, and hardware features. The purpose of this integration is to augment the accuracy and effectiveness of malware detection within the IDS system.

As conducted by Arshad et al. (2018), a three-level hybrid malware detection model has been created for Android operating systems that combine the strengths of the three different levels to offer excellent detection accuracy: a) dynamic and static analysis, b) remote and localhost, and c) machine learning algorithms. Kuo, Liu and Wang (2019), Fang, Yang and Ji (2019) employed a combination of static feature analysis through permissions and dynamic feature analysis through API usage to enhance the accuracy of Android malware detection.

Arora and Peddoju (2018), Shyong, Jeng and Chen (2020) focused on a hybrid method consisting of static permissions and monitoring network traffic features used for the purpose of detecting malicious activity on mobiles. The proposed system by Shyong et al. (2020) utilizes two phases. In the first phase, static permissions are collected and classified into benign and malicious applications, filtering out benign ones. The second phase involves the dynamic analysis of network traffic generated by identified malware. Features are extracted from the malware's network behaviour, and machine learning is employed to classify the malware into specific families. Y. Liu, Zhang, Li and Chen (2016) and N. Zhang et al. (2021) focused on distinguishing benign applications from malware by analysing both the manifest and system call features.

Vinayakumar, Soman, Poornachandran and Sachin Kumar (2018) presented the LSTM model for detecting malware on Android devices. Different network topologies with many network parameters are employed to obtain acceptable malware detection rates. A layered LSTM of 32 blocks of memory has shown success in detecting all individual malware behaviours, compared to other static machine learning classifications. Garg and Baliyan (2019) proposed an ensemble model combining machine learning algorithms to efficiently identify and classify malware. Features such as permission, API calls, system components, network traffic, battery, and other features were extracted from applications and devices. During the experimental results, the accuracy of the proposed system reached 98.27%. Table 3 represents the summary of the previous works that is based on hybrid analysis.

Table 3. Summary of the papers based on hybrid analysis

Reference	Dataset Sources	No. of Sample	Method	Analysis Tool	Feature Type	Acc. %	Pre. %	Recall %	F1 score%
(Y. Liu et al., 2016)	Wandoujia and Gnome project	B+M 500	SVM, KNN,NB	Apktool. ADB Tool, Strace Tool, Monkey Tool	permissions, API call, System calls	93.33~ 99.28	---	94.59 ~99.47	---
(Arshad et al., 2018)	Drebin, Genome	---	SVM RF,DT, NB	Asset Packaging tool ,Baksmali tool., Weka	Network	98.97	---	98.5	---
(Arora & Peddoju, 2018)	Genome Project	B 600 M 524	FP-Growth algorithm, IG and Chi-Square for feature selection	Apk tool	permissions, Network	94.25	---	97.9	---
(Vinayakumar et al., 2018)	MalGenome	B 408 M 1330	LSTM	Apk Tool, Adb monkey tool	permissions, Battery, Memory	93.9 ~ 97.5	---	---	---
(Kuo et al., 2019)	GP, Virusshare	B 1000 M 1000	SVM, RF	Apk tool, Emulator	permissions, API call	94	---	95	---
(Fang et al., 2019)	GP, Virusshare	B 4000 M 4000	XGBoost, Bayesian classifier for detection and classification. TF-IDF for feature filtering	Apk Tool, BackSmali Tool, MALINE Tool Monkey Tool ,Emulator, Strace Tool	permissions, API call	94.6	---	---	---
(Garg & Baliyan, 2019)	GP, Wandouji, AMD, Androzoo	B 60000 M 25450	ensemble (SVM, PART, MLP, RIDOR)	Apk tool, Emulator, ADB shell, Strace tool	permissions, API call, System calls, Network, Battery, CPU, Memory	98.27	---	98.79	---
(Shyong et al., 2020)	GP, Drebin	B 1000 M 1024	RF	aapt dump permissions, Emulator,Tcpdump ,Monkey tool	permissions, Network	96 ~98.86	---	---	---
(N. Zhang et al., 2021)	PlayDrone, Drebin	B 2978 M 2707	CNN, BiLTSM	Strace, Monkey, AndroGuard, ADB tools	Opcode, System calls	97	56.5	95.5	96

4. DISCUSSION AND COMPARISON

This section discusses the methodologies employed by researchers for analysing Android applications, along with the strengths and weaknesses of each approach. Furthermore, the study explores the types of features utilized by researchers in their analyses. Most of the researchers reviewed in this study focused on analysis at the application level instead of the

* Corresponding author

system level due to limited resources on smartphones. The behaviour of applications is typically examined through three approaches: static analysis, dynamic analysis, and hybrid analysis. Static analysis involves decompiling the application and extracting key features to differentiate between malicious and benign activities without executing the application. Dynamic analysis, on the other hand, involves executing the application on a real device or in a simulated environment to extract relevant behavioural features. Hybrid analysis combines the strengths of both static and dynamic analysis to enhance the detection capabilities of the system. In general, static analysis is known for its speed and simplicity compared to dynamic analysis, while hybrid analysis is more complex than both static and dynamic methods. Figure 4 illustrates the distribution of papers reviewed in this study, highlighting the utilization of static, dynamic, and hybrid methods for analysis.

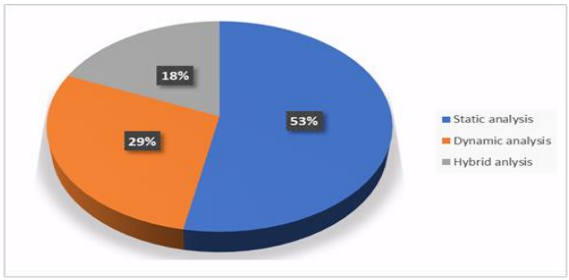

Figure 4. Statistics about the analysis technique

Comparing malware IDS based on different feature sets is crucial for assessing their strengths and weaknesses in detecting Android malware. Malware IDS relying on Android manifest analysis rapidly extracts vital information about an app's functionalities and potential risks. However, they may struggle with advanced malware that uses obfuscation techniques to conceal malicious activities within legitimate functions. Source code-based IDS, particularly API analysis, effectively detects suspicious activities like accessing sensitive resources; nevertheless, it may encounter challenges with obfuscated code, leading to false positives or negatives. System call-based IDS can identify unknown malware but requires more computational resources and time, potentially causing delays in real-time detection. Hardware-based monitoring IDS, examining resources like battery and CPU, is valuable for detecting resource-exploiting malware; however, it may impact device performance and battery life due to additional overhead. Android malware detection approaches can utilize single-category or multi-category features. Single-category features, such as permissions, offer simplicity and efficiency but result in lower detection accuracy. In contrast, multi-category features, including permissions, APIs, and network traffic, enhance detection accuracy but pose challenges in handling multiple categories and require more processing resources. Figure 5 presents the statistical analysis of the reviewed papers, showcasing the utilization of single and multi-category features in static analysis.

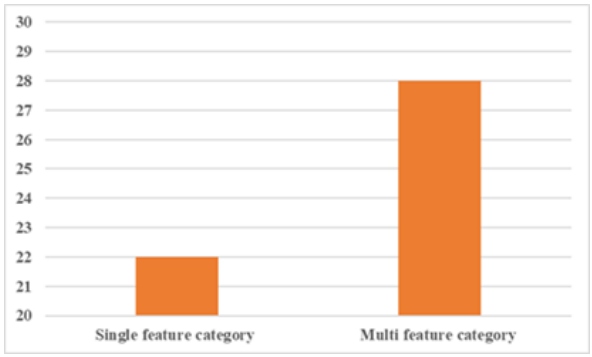

Figure 5. Single and multi-feature category statistics

Furthermore, it is crucial to highlight the significance of the features and databases employed by researchers. Figures 6 and 7 provide visual representations of the prominent features and commonly utilized databases in the reviewed papers in this work.

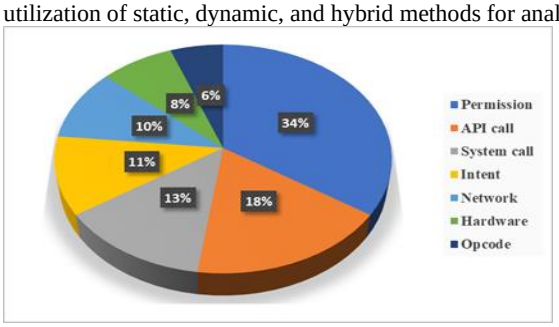

Figure 6. More frequent feature categories

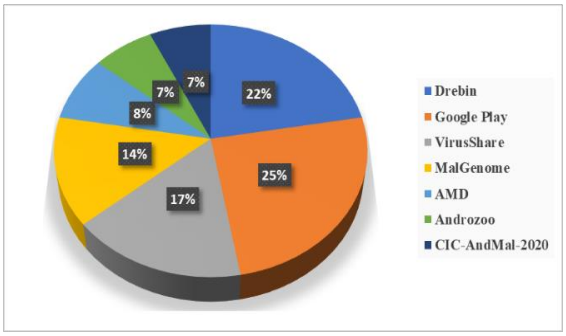

Figure 7. Frequent dataset used by researchers

Some researchers have developed IDS systems using one-class classifiers that rely exclusively on normal behaviour data obtained from running benign apps, eliminating the need for training with malicious samples. However, these models are prone to false positives due to unexpected changes in smartphone behaviour, such as CPU spikes during Android OS maintenance or sudden increases in network traffic from background downloads or installations. Therefore, when constructing a host-based IDS for detecting malicious activities on smartphones, it is advisable to strike a balance between accuracy and efficiency, considering the limited resources available on smartphones, particularly in terms of CPU, memory, and battery.

5. CHALLENGES AND RECOMMENDATIONS

The process of constructing a machine learning-based mobile malware detection model entails several key steps. Firstly, samples of both benign and malicious applications are analysed, and crucial features are extracted from them. Subsequently, a classification model is developed capable of effectively differentiating between benign and malicious samples. To assess the model's performance, it is tested on a mobile device using real-time applications. Evaluation metrics such as accuracy and performance are then used to measure the effectiveness of the model. Despite the substantial progress made by researchers in developing various models, there remain several challenges and open issues that warrant further investigation. The following points outline some of these challenges:

1- **Dataset**: Due to the lack of a comprehensive and available dataset, many researchers have relied on the Drebin dataset. While this dataset is valuable, it does not include new samples of applications. As Android continuously evolves with new versions and

updates that introduce new features and deprecate old ones, developers exploit these changes to build applications. Hence, it is recommended to construct a dataset that comprises applications based on most API levels of Android; this provides many advantages, such as improved model generalization, an accurate representation of the Android ecosystem, and enhanced model accuracy.

2- **Analysis tool**: Researchers face challenges in analysing a large number of applications, as it is a time-consuming process that often requires multiple tools and expertise. It is recommended to develop an automat and easy-to-use analysis tool that can analyse applications in static, dynamic, and hybrid modes and extract features from them. This saves time for the user to analyse as many applications as possible in different ways.

3- **Feature Selection**: The identification and selection of informative features for malware detection is a critical task. It is recommended to explore advanced feature selection techniques, such as combining existing methods or developing new ones, to enhance the model's accuracy and efficiency.

4- **Privacy Preservation**: Ensuring user privacy while conducting malware detection is essential. It is recommended to explore the possibility of conducting certain analysis tasks locally on the user's device. This approach can minimize data sharing and enhance user privacy.

5- **Real-Time Detection**: Mobile malware detection models should operate efficiently in real-time, without imposing excessive computational overhead on the device. It is recommended to create a model that relies on computationally-efficient features. Utilizing these features, the model will optimize resource utilization while maintaining its effectiveness in detecting malware promptly, enabling swift responses to emerging malicious activities.

6. CONCLUSION

This paper presents a comprehensive overview of Android malware intrusion detection systems and their classification. The three analytical methods discussed in previous works are static, dynamic, and hybrid analysis. The statistical analysis shows that static analysis is the approach that is most frequently used (53%), followed by dynamic analysis (29%), and hybrid analysis (18%). The results of the analysis also showed that the most used features by researchers were permissions, which reached 34%, followed by API calls, system calls, and intent features, respectively. The study emphasizes the importance of considering both the selection of databases for benign and malware applications and the appropriate tools for analysis to extract critical features from diverse sources. Notably, the Drebin database was the preferred choice for 22% of researchers for malware samples, while 25% relied on the Google Play database for benign applications. Finally, the paper addresses open challenges in the field and offers valuable recommendations for further research and development in the domain of Android malware detection.

7. ACKNOWLEDGEMENTS

Full thanks expressed to Duhok Polytechnic University (DPU).

8. COMPETING INTERESTS

Authors have declared that no competing interests exist.

9. FUNDING

We would like to clarify that there was no external funding received for this research project. The manuscript was completed without any financial support or grants. We would like to acknowledge that this study was conducted independently and self-funded.

REFERENCES

Abderrahmane, A., Adnane, G., Yacine, C., & Khireddine, G. (2019). Android malware detection based on system calls analysis and CNN classification. Paper presented at the 2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW).

Agrawal, P., & Trivedi, B. (2019). A survey on android malware and their detection techniques. Paper presented at the 2019 IEEE International conference on electrical, computer and communication technologies (ICECCT).

Almomani, I., Qaddoura, R., Habib, M., Alsoghyer, S., Al Khayer, A., Aljarah, I., & Faris, H. (2021). Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data. IEEE Access, 9, 57674-57691.

AlOmari, H., Yaseen, Q. M., & Al-Betar, M. A. (2023). A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection. Procedia Computer Science, 220, 763-768.

Alqahtani, H., Sarker, I. H., Kalim, A., Minhaz Hossain, S. M., Ikhlaq, S., & Hossain, S. (2020). Cyber intrusion detection using machine learning classification techniques. Paper presented at the Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020, Revised Selected Papers 1.

Alsoghyer, S., & Almomani, I. (2020). On the effectiveness of application permissions for Android ransomware detection. Paper presented at the 2020 6th conference on data science and machine learning applications (CDMA).

Amer, E. (2021). Permission-based approach for android malware analysis through ensemble-based voting model. Paper presented at the 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC).

Ariyapala, K., Do Hoang, G., Huynh, N. A., Wee, K. N., & Conti, M. (2016). A host and network based intrusion detection for android smartphones. Paper presented at the 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA).

Arora, A., & Peddoju, S. K. (2018). NTPDroid: a hybrid android malware detector using network traffic and system permissions. Paper presented at the 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).

Arshad, S., Shah, M. A., Wahid, A., Mehmood, A., Song, H., & Yu, H. (2018). SAMADroid: a novel 3-level hybrid malware detection model for android operating system. IEEE Access, 6, 4321-4339.

BalaGanesh, D., Chakrabarti, A., & Midhunchakkaravarthy, D. (2018). Smart devices threats, vulnerabilities and malware detection approaches: a survey. European Journal of Engineering and Technology Research, 3(2), 7-12.

Bansal, V., Baliyan, N., & Ghosh, M. (2022). Dynamic Android Malware Detection Using Light Gradient Boosting Machine. Paper presented at the 2022 4th International Conference on Artificial Intelligence and Speech Technology (AIST).

Barbhuiya, S., Kilpatrick, P., & Nikolopoulos, D. S. (2020). DroidLight: Lightweight anomaly-based intrusion detection system for smartphone devices. Paper presented at the Proceedings of the 21st International Conference on Distributed Computing and Networking.

Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2020). Malware detection in android systems with traditional machine learning models: a survey. Paper presented at the 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA).

Borek, M. (2017). Intrusion Detection System for Android: Linux Kernel System Salls Analysis.

Borkar, A., Donode, A., & Kumari, A. (2017). A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and protection system (IIDPS). Paper presented at the 2017 International conference on inventive computing and informatics (ICICI).

Chawla, A., Lee, B., Fallon, S., & Jacob, P. (2019). Host based intrusion detection system with combined CNN/RNN model. Paper presented at the ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18.

da Costa, F. H., Medeiros, I., Costa, P., Menezes, T., Vinícius, M., Bonifácio, R., & Canedo, E. D. (2020). Droidxp: A benchmark for supporting the research on mining android sandboxes. Paper presented at the 2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM).

Dharmalingam, V. P., & Palanisamy, V. (2021). A novel permission ranking system for android malware detection—the permission grader. Journal of Ambient Intelligence and Humanized Computing, 12, 5071-5081.

Elkhadir, Z., Chougdali, K., & Benattou, M. (2016). Intrusion detection system using pca and kernel pca methods. Paper presented at the Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015: MedCT 2015 Volume 2.

Esmaeili, S., & Shahriari, H. R. (2019). PodBot: a new botnet detection method by host and network-based analysis. Paper presented at the 2019 27th Iranian Conference on Electrical Engineering (ICEE).

Fang, Q., Yang, X., & Ji, C. (2019). A hybrid detection method for android malware. Paper presented at the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC).

Feng, R., Liu, Y., & Lin, S. (2019). A performance-sensitive malware detection system on mobile platform. Paper presented at the Formal Methods and Software Engineering: 21st International Conference on Formal Engineering Methods, ICFEM 2019, Shenzhen, China, November 5–9, 2019, Proceedings 21.

Garg, S., & Baliyan, N. (2019). A novel parallel classifier scheme for vulnerability detection in android. Computers & Electrical Engineering, 77, 12-26.

Georgios Kambourakis, A. S., Constantinos Kolias, and Dimitrios Damopoulos. (2018). Intrusion Detection and Prevention for Mobile Ecosystems: Taylor & Francis Group, LLC.

Gyamfi, N. K., & Owusu, E. (2018). Survey of mobile malware analysis, detection techniques and tool. Paper presented at the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).

Hein, C. L. P. M., & Myo, K. M. (2018). Permission-based feature selection for android malware detection and analysis. International Journal of Computer Applications, 181(19), 29-39.

Istiaque, S. M., Khan, A. I., & Waheed, S. (2020). Smart intrusion detection system comprised of machine learning and deep learning. European Journal of Engineering and Technology Research, 5(10), 1168-1173.

Jannat, U. S., Hasnayeen, S. M., Shuhan, M. K. B., & Ferdous, M. S. (2019). Analysis and detection of malware in Android applications using machine learning. Paper presented at the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE).

Jiang, X., Mao, B., Guan, J., & Huang, X. (2020). Android malware detection using fine-grained features. Scientific Programming, 2020, 1-13.

John, T. S., Thomas, T., & Emmanuel, S. (2020). Graph convolutional networks for android malware detection with system call graphs. Paper presented at the 2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP).

Kapoor, A., Kushwaha, H., & Gandotra, E. (2019). Permission based android malicious application detection using machine learning. Paper presented at the 2019 International Conference on Signal Processing and Communication (ICSC).

Khariwal, K., Singh, J., & Arora, A. (2020). IPDroid: Android malware detection using intents and permissions. Paper presented at the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4).

Khatter, K. (2018). Malicious Application Detection and Classification System for Android Mobiles.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(1), 1-22.

Kumar, R., Zhang, X., Wang, W., Khan, R. U., Kumar, J., & Sharif, A. (2019). A multimodal malware detection technique for Android IoT devices using various features. IEEE Access, 7, 64411-64430.

Kumar, S., Viinikainen, A., & Hamalainen, T. (2016). Machine learning classification model for network based intrusion detection system. Paper presented at the 2016 11th international conference for internet technology and secured transactions (ICITST).

Kuo, W.-C., Liu, T.-P., & Wang, C.-C. (2019). Study on android hybrid malware detection based on machine learning. Paper presented at the 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS).

Lê, N. C., Nguyen, T.-M., Truong, T., Nguyen, N.-D., & Ngô, T. (2020). A Machine Learning Approach for Real Time Android Malware Detection. Paper presented at the 2020 RIVF International Conference on Computing and Communication Technologies (RIVF).

Lei, T., Qin, Z., Wang, Z., Li, Q., & Ye, D. (2019). EveDroid: Event-aware Android malware detection against model degrading for IoT devices. IEEE Internet of Things Journal, 6(4), 6668-6680.

Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of android malware detection approaches based on machine learning. IEEE Access, 8, 124579-124607.

Liu, P. (2019). An intrusion detection system based on convolutional neural network. Paper presented at the Proceedings of the 2019 11th International Conference on Computer and Automation Engineering.

Liu, Y., Zhang, Y., Li, H., & Chen, X. (2016). A hybrid malware detecting scheme for mobile Android applications. Paper presented at the 2016 IEEE International Conference on Consumer Electronics (ICCE).

Ma, Z., Ge, H., Liu, Y., Zhao, M., & Ma, J. (2019). A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE Access, 7, 21235-21245.

Mahdavifar, S., Kadir, A. F. A., Fatemi, R., Alhadidi, D., & Ghorbani, A. A. (2020). Dynamic android malware category classification using semi-supervised deep learning. Paper presented at the 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech).

Malik, S. Anomaly based Intrusion Detection in Android Mobiles: A Review.

Manzil, H. H. R. (2022). DynaMalDroid: Dynamic Analysis-Based Detection Framework for Android Malware Using Machine Learning Techniques. Paper presented at the 2022 International Conference on Knowledge Engineering and Communication Systems (ICKES).

Mathur, A., Podila, L. M., Kulkarni, K., Niyaz, Q., & Javaid, A. Y. (2021). NATICUSdroid: A malware detection framework for Android using native and custom permissions. Journal of Information Security and Applications, 58, 102696.

Niu, W., Cao, R., Zhang, X., Ding, K., Zhang, K., & Li, T. (2020). OpCode-level function call graph based android malware classification using deep learning. Sensors, 20(13), 3645.

Painter, N., & Kadhiwala, B. (2018). Machine-learning-based android malware detection techniques—A comparative analysis. Paper presented at the Information and Communication Technology for Sustainable Development: Proceedings of ICT4SD 2016, Volume 1.

Radoglou-Grammatikis, P. I., & Sarigiannidis, P. G. (2017). Flow anomaly based intrusion detection system for Android mobile devices. Paper presented at the 2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST).

Riasat, R., Sakeena, M., Sadiq, A. H., & Wang, Y.-J. (2018). Onamd: an online android malware detection approach. Paper presented at the 2018 International Conference on Machine Learning and Cybernetics (ICMLC).

Ribeiro, J., Saghezchi, F. B., Mantas, G., Rodriguez, J., & Abd-Alhameed, R. A. (2020). Hidroid: prototyping a behavioral host-based intrusion detection and prevention system for android. IEEE Access, 8, 23154-23168.

Sandeep, H. (2019). Static analysis of android malware detection using deep learning. Paper presented at the 2019 International Conference on Intelligent Computing and Control Systems (ICCS).

Sangal, A., & Verma, H. K. (2020). A static feature selection-based android malware detection using machine learning techniques. Paper presented at the 2020 International conference on smart electronics and communication (ICOSEC).

Sewak, M., Sahay, S. K., & Rathore, H. (2020). Deepintent: implicitintent based android ids with e2e deep learning architecture. Paper presented at the 2020 IEEE 31st annual international symposium on personal, indoor and mobile radio communications.

Shamshirband, S., Fathi, M., Chronopoulos, A. T., Montieri, A., Palumbo, F., & Pescapè, A. (2020). Computational intelligence intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues. Journal of Information Security and Applications, 55, 102582.

Shyong, Y.-C., Jeng, T.-H., & Chen, Y.-M. (2020). Combining static permissions and dynamic packet analysis to improve Android malware detection. Paper presented at the 2020 2nd International Conference on Computer Communication and the Internet (ICCCI).

Singh, A. K., Wadhwa, G., Ahuja, M., Soni, K., & Sharma, K. (2020). Android malware detection using LSI-based reduced opcode feature vector. Procedia Computer Science, 173, 291-298.

Sirisha, P., & Anuradha, T. (2019). Detection of permission driven malware in android using deep learning techniques. Paper presented at the 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA).

Tiwari, S. R., & Shukla, R. U. (2018). An android malware detection technique using optimized permission and api with pca. Paper presented at the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS).

Uğurlu, M., & Doğru, İ. A. (2019). A survey on deep learning based intrusion detection system. Paper presented at the 2019 4th International Conference on Computer Science and Engineering (UBMK).

Vinayakumar, R., Soman, K., Poornachandran, P., & Sachin Kumar, S. (2018). Detecting Android malware using long short-term memory (LSTM). Journal of Intelligent & Fuzzy Systems, 34(3), 1277-1288.

Wang, S., Chen, Z., Yan, Q., Yang, B., Peng, L., & Jia, Z. (2019). A mobile malware detection method using behavior features in network traffic. Journal of Network and Computer Applications, 133, 15-25.

Xiao, X., Zhang, S., Mercaldo, F., Hu, G., & Sangaiah, A. K. (2019). Android malware detection based on system call sequences and LSTM. Multimedia Tools and Applications, 78, 3979-3999.

Xie, N., Qin, Z., & Di, X. (2023). GA-StackingMD: Android Malware Detection Method Based on Genetic Algorithm Optimized Stacking. Applied Sciences, 13(4), 2629.

Yerima, S. Y., & Alzaylaee, M. K. (2020). Mobile botnet detection: A deep learning approach using convolutional neural networks. Paper presented at the 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA).

Yuan, W., Jiang, Y., Li, H., & Cai, M. (2019). A lightweight on-device detection method for android malware. IEEE transactions on systems, man, and cybernetics: systems, 51(9), 5600-5611.

Zachariah, R., Akash, K., Yousef, M. S., & Chacko, A. M. (2017). Android malware detection a survey. Paper presented at the 2017 IEEE international conference on circuits and systems (ICCS).

Zhang, H., Luo, S., Zhang, Y., & Pan, L. (2019). An efficient Android malware detection system based on method-level behavioral semantic analysis. IEEE Access, 7, 69246-69256.

Zhang, N., Xue, J., Ma, Y., Zhang, R., Liang, T., & Tan, Y. a. (2021). Hybrid sequence-based Android malware detection using natural language processing. International Journal of Intelligent Systems, 36(10), 5770-5784.

Zhang, X., Mathur, A., Zhao, L., Rahmat, S., Niyaz, Q., Javaid, A., & Yang, X. (2022). An early detection of android malware using system calls based machine learning model. Paper presented at the Proceedings of the 17th International Conference on Availability, Reliability and Security.

Zhang, Y., Feng, C., Huang, L., Ye, C., & Weng, L. (2020). Detection of android malicious family based on manifest information. Paper presented at the 2020 15th International Conference on Computer Science & Education (ICCSE).

Zhao, L., Li, D., Zheng, G., & Shi, W. (2018). Deep neural network based on android mobile malware detection system using opcode sequences. Paper presented at the 2018 IEEE 18th International Conference on Communication Technology (ICCT).

Zhou, H., Yang, X., Pan, H., & Guo, W. (2020). An android malware detection approach based on SIMGRU. IEEE Access, 8, 148404-148410.

Zhou, Q., Feng, F., Shen, Z., Zhou, R., Hsieh, M.-Y., & Li, K.-C. (2019). A novel approach for mobile malware classification and detection in Android systems. Multimedia Tools and Applications, 78, 3529-3552.