https://doi.org/10.51483/IJAIML.2.2.2022.21-34 ISSN: 2789-2557

SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Malicious Applications Detection in Android Using Machine Learning

Muhammad Mugees Asif[1]*, Sana Asif[2], Iqra Mubarik[3] and Rabia Hussain[4]

[1]Department of Computer Science, Garrison University, Lahore 94777, Pakistan. E-mail: mugee-sasifm@gmail.com
[2]Management Department, Air University, Islamabad 44230, Pakistan. E-mail: mesanaasif@gmail.com
[3]Department of Computer Science, Garrison University, Lahore 94777, Pakistan. E-mail: iqramubarik914@gmail.com
[4]Department of Computer Science, Garrison University, Lahore 94777, Pakistan. E-mail: rabiahussain993@gmail.com

Article Info

Volume 2, Issue 2, July 2022
Received : 12 March 2022
Accepted : 22 June 2022
Published : 05 July 2022
doi: 10.51483/IJAIML.2.2.2022.21-34

Abstract

A huge number of applications available for Android-based smartphone devices have emerged over the past years. Due to which a huge number of malicious applications has been growing explosively. Many approaches have been proposed to ensure the security and quality of application in the markets. Usually, Machine Learning approaches are utilized in the classification process of malicious application detection. Calculating accurate results of characterizing applications behaviors, or other features, has a direct effect on the results with Machine Learning calculations. Android applications emerge so quickly. The behavior of current applications has gotten progressively malicious. The extraction of malware-infected features from applications is thus become a difficult task. According to our knowledge, a ton of features have been extricated in existing work however no survey has overviewed the features built for identifying malicious applications efficiently. In this paper, we will in general give an extensive review of such sort of work that identifies feature applications by describing various practices of uses with various kinds of features. In this survey we have discussed the following dimensions: extraction and selection of feature methods if any, methods of detection and evaluation performed. In light of our review, we notice the issues of investigating malware-affected features from applications, give the scientific categorization and demonstrate the future headings.

Keywords: *Android security, Machine learning, Malware analysis, Malicious application detection, Survey*

1. Introduction

Android has become one of the most famous operating systems for smartphones in the past few years. According to the report created by International Data Corporation (IDC) in the year 2017, Android occupies 85% of the worldwide market share. Due to this popularity, many information-stealing cases are also increasing as this popularity attracts hackers to steal information from regular users. Till now, no application can forestall

* *Corresponding author: Muhammad Mugees Asif, Department of Computer Science, Garrison University, Lahore 94777, Pakistan. E-mail: mugeesasifm@gmail.com*

malicious applications effectively. Some applications developed to detect Malware but these types of applications are not having the option to distinguish updated Malware application or other apps which are infected by different types of viruses like Trojans, Spyware (Wei *et al.*, 2019). So, preventing the Android platform from malicious applications is still a challenging task. While platforms other than Android like IOS allow users to install the application from their official store like iTunes, but Android framework permit clients to download applications into their mobile phones from outsider sources like Torrents or direct downloads which is a major tread for Android security. These reasons provide easy access for attackers to distribute their malicious applications. Whenever anti-hackers came up with new analyses to protect the platforms, the hackers start developing new encrypted techniques to bypass that. That is the reason due to which we need a new technique here and we are trying to implement Machine Learning Techniques in our system to protect the Android platform from malicious software. These days, we can say that Machine Learning is the future because if you look around, you can see that there is a lot of data everywhere from text, calls to emails, and so on. It is very important to manage all of this data efficiently. If you consider humans to perform this task, there is a limitation for the amount of data that humans can manage otherwise this is nearly impossible for a human being. So the only way left is to approach the Machine Learning techniques (Aafer and Yin, 2013).

Machine Learning is the ability of a machine to learn without too much programming. This is something like if you tell a machine to perform a task two times, the machine will perform that task a third time automatically and it will increase its efficiency according to the number of times this process repeats. That is our aim and here we will be developing a mobile application based on Machine Learning techniques to detect applications that are encoded by Malware and other types of viruses to prevent the Android operating system from being infected (Wermke *et al.*, 2018).

Figure 1 will showcase our taxonomy of features developed for malicious application detection in android. First, we depict the investigation techniques for recognizing vindictive applications in android, at that point, we portray the most utilized features. From that point forward, we will play out an examination among the connected work.

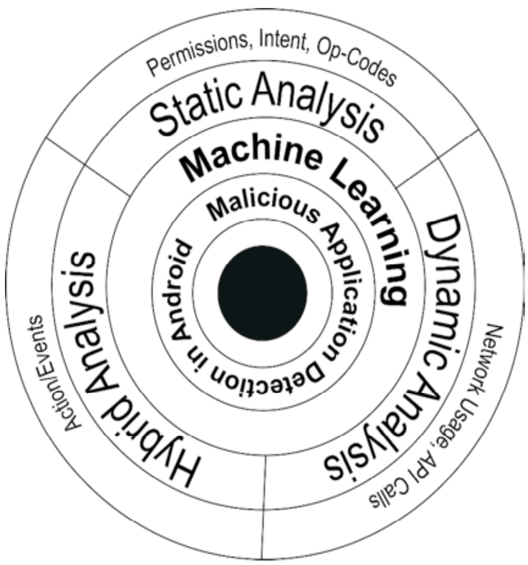

Figure 1 : Taxonomy of Android Malicious Application Detection

We elaborate on the basic issues of extracting the categories of features. Due to complicating behaviors and expanding measure of an Android Bundle (APK), extracting of features become time-consuming, coming about within the non-effective location. For illustration with a static examination, it takes fifteen minutes to eliminate work call outlines for an apk with 15 MB. Regularly it isn't commendable for ongoing discovery for end clients. The measure of features that can be separated from an application can be up to 1,000,000. Nevertheless, various features are zero (Faruki *et al.,* 2013). Instructions to beneficially deal with the insufficient vectors are basic issue.

2. Background

Before presenting our research, we to begin with a detailed presentation on the Android stage and security instruments. The essential information will assist you in overcoming the problems and hazards associated with the Android stage. It's imperative data for unused issues featured by Android application security investigation strategies and advances.

2.1. Android Operating System

Based on Linux bit Android is an open-source portable working framework that is outlined essentially for savvy gadgets. The Linux component layer, library layer, application system layer, and application layer make up the Android operating system. Linux part layer gives a few fundamental capacities such as memory administration, handle administration, and arrange conventions. This layer comprises the middle drivers for all of the hardware pieces' critical gadgets. To assist the application framework layer, the Library layer provides a middle library that fuses the underlying library and outsider library for apps in a mastermind. The application framework layer is similar to a central layer that academic people encourage the sections to use, and it refreshes the overall structure's adaptability (Wei *et al.,* 2019). The application system includes various framework administrations, such as Action Director, Window Chief, Asset Supervisor, Area Supervisor, Substance Supplier, and so on, to complete this function. The application layer, the so to speak layer that can be associated with customers, contains all applications operating on Android gadgets. There are numerous techniques for IoT security accessible with secure shows (Xu *et al.,* 2019; Sadeghi *t al.,* 2017). In this paper, we particularly discuss application security in the Android system.

2.2. Android Applications

Application for Android is created in Kotlin and Java programming language using Android Software Development Kit (SDK). Other than the Java code, an app may moreover contain a few nearby libraries that are given by the Android framework or executed by engineers. An application's assembled code enclosed by information and assets is full into a report record, which is called Android Application Bundle (APK). An Android application operates by using a runtime environment. An application contains four essential parts: Action, Broadcast Collectors, Benefit, and Substance Supplier. Action directs the Client Interface and controls the customer communication with the savvy telephone screen. Broadcast Collectors manage correspondence between the working system and applications (Aafer and Yin, 2013). Advantage administers establishment planning of an application to perform long-running tasks. Substance Supplier gives the data sharing over applications.

2.3. Security Mechanisms

Security Professionals present security instruments when they plan the security approaches for the Android stage. Android framework depends on progressive design, and each layer has its security component. We will discuss here the two most adopted mechanisms, which are the Permission inspection-based mechanism and sandbox mechanism (Faruki *et al.,* 2013).

2.3.1. Access Control

The traditional access control system is the same as the Linux piece security component in Android. The subject is restricted by access control (for example, customers or organizations) to get to the inquiry (for example, resources). This might be a fundamental way to deal with guarantee the mystery and judgment of data. Get to control incorporates two sorts of strategies, required to get to control (MAC) and optional get to control (DAC). The Linux security module executes Macintosh. Record get to control executes DAC (Zhao *et al.,* 2016).

2.3.2. Permission Inspection

Android employments permission-based security demonstrates to confine applications get to a few assets. In case apps need to utilize limited assets, they have to be applied for consent through XML records. Applications cannot utilize limited assets until the Android framework endorses Typical, Unsafe, Signature, and Signature/ Framework are the four tiers of Android assents. Low-level consents, which check for common and harmful levels, are granted quickly after an applicant submits an application. Mark level and mark / framework level approvals are known as cutting-edge consents. An application can apply for these approvals at any time, but it must first attain stage level affirmation. Be that as it may, there are numerous inadequacies in this instrument. Clients have to be chosen on the off chance that the authorizations that an applicant applies ought to be authorized, however, clients don't have sufficient information to judge it (Yang *et al.*, 2014).

2.3.3. Sandbox

In the Android structure, Sandbox is used to apportion running applications. A sandbox gives an immovably controlled arrangement of resources for applications to run in. Each application runs in Dalvik Virtual Machine (VM) and has had to handle resources and space amid the run-season of the Android applications. Consequently, multiple applications can't relate to one another and can't get to every other's resource and memory space (Yang *et al.*, 2017).

3. Methodology

We conduct our SLR after planning the study. The following sections portray the comprehensive information of our SLR procedure.

3.1. Queries of Research

Depicting significant learning with ML approaches is also an issue since we found that the latest work was done in 2019. A few assessments have driven AI strategies in the Android Platform.

3.2. Search Procedure

Our SLR primarily focuses on seeking logical databases instead of books and other reports. This study chose two databases to perform the SLR search process:

- https://scholar.google.com
- https://ieeexplore.ieee.org

The accompanying watchwords were utilized to discover related examinations to achieve this SLR research: "Malicious Application Detection using Machine Learning" OR "Malicious Application Detection in Android using Machine Learning" OR "Malicious Application Detection in Android using ML" OR "Malicious Application Detection in Android using DL"

3.3. Search Procedure

Figure 2 demonstrates the hierarchy of our research methodology.

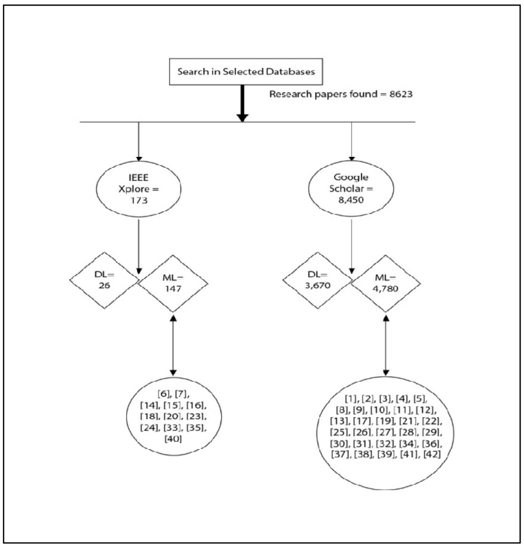

Figure 2: Hierarchy of Research Methodology

3.4. Papers Citations

Table 1 shows the number of citations of chosen research papers, which are taken from IEEE Explore and Google Scholar. The table shows that the majority of the papers are very much referred to, that true papers are investigated for this similar writing survey.

Table 1: Reference Papers with Citations and Database

Sr#	Citations	Database
1.	34	Google Scholar
2.	644	Google Scholar
3.	46	Google Scholar
4.	128	Google Scholar
5.	45	Google Scholar
6.	45	IEEE Xplore
7.	16	IEEE Xplore
8.	214	Google Scholar
9.	4	Google Scholar
10.	130	Google Scholar
11.	644	Google Scholar
12.	290	Google Scholar
13.	10	Google Scholar
14.	69	IEEE Xplore
15.	1	IEEE Xplore
16.	61	IEEE Xplore
17.	62	Google Scholar
18.	13	IEEE Xplore
19.	2	Google Scholar
20.	84	IEEE Xplore
21.	48	Google Scholar
22.	214	Google Scholar
23.	55	IEEE Xplore
24.	2	IEEE Xplore
25.	94	Google Scholar
26.	3	Google Scholar
27.	101	Google Scholar
28.	9	Google Scholar
29.	4	Google Scholar
30.	784	Google Scholar
31.	31	Google Scholar
32.	15	Google Scholar
33.	21	IEEE Xplore
34.	6	Google Scholar
35.	12	IEEE Xplore
36.	3	Google Scholar
37.	138	Google Scholar

Table 1 (Cont.)

Sr#	Citations	Database
38.	40	Google Scholar
39.	29	Google Scholar
40.	63	IEEE Xplore
41.	26	Google Scholar
42.	230	Google Scholar
43.	153	Google Scholar
44.	0	Google Scholar
45.	44	IEEE Xplore
46.	55	Google Scholar
47.	5	Google Scholar
48.	136	Google Scholar
49.	64	Google Scholar
50.	26	Google Scholar
51.	15	Google Scholar
52.	5	Google Scholar

Figure 3 shows the quantities of assessments by year of dispersion. It shows that the year 2014 has more inclination than different years. According to Figure 3, the amount of production was reduced in the year 2020 of studies utilizing Machine Learning approaches.

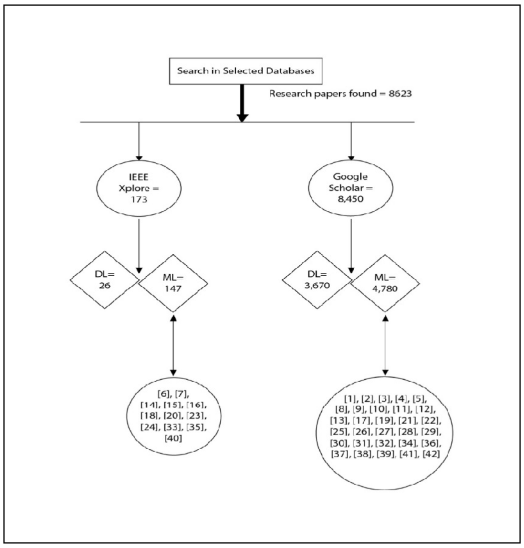

Figure 3: Research Papers According to Years

4. Related Work

The process of detecting Malicious Applications in Android is shown in Figure 4 below. The quality of the features chosen affects the discovery's performance. Selected features can further be classified into the following categories:

- Static Features

- Dynamic Features
- Features based on Meta-data

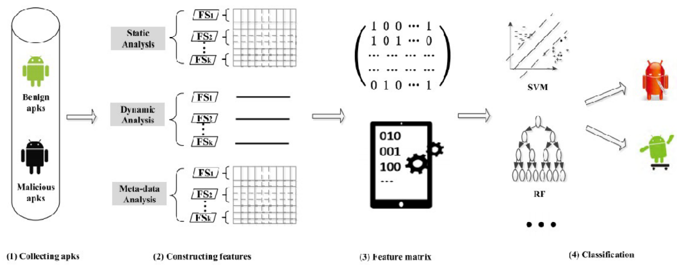

Figure 4: Architecture of Malicious Application Detection

4.1. Static Features

These features can be fetched without executing the applications by investigating the code. These features incorporate such sort of features, which are accessible in apk records, for example, AndroidManifest.xml and Java code document. The accuracy achieved for permission-based detection is around 90% (Faruki *et al.*, 2013; Aafer *etal.*, 2013; Ahao *et al.*, 2016; Yang *et al.*, 2014), which is additionally improved with extra features (Wermke *et al.*, 2019; Yang *et al.*, 2017; Suarez-Tangil *et al.*, 2017). In the next section, we will discuss the most used static features in detail.

4.2. Permission

A permission-based security model is used by Android to restrict an application from accessing user's sensitive information to ensure client data security. Applications request permission from the users before their installation. After clients acknowledge these authorizations, the application installs itself on the mobile phone.

So many ways exist for Android malware application detection by extracting permission. Wang *et al.* (2019) performs an investigation on the dangers of individual permissions and cooperative authorizations. They positioned the authorizations regarding their dangers. Sarma (Aafer *et al.*, 2013) utilized both the permissions that an application mentioned and authorizations mentioned by different applications in a similar classification. The reason for this technique was to check whether the applications exceeded their normal dangers. Some studies (Wermke *et al.*, 2019; Faruki *et al.*, 2013; Zhao *et al.*, 2016; Yang *et al.*, 2014; 2017; Suarez-Tangil *et al.*, 2017; Aafer *et al.*, 2013; Chen *et al.*, 2014; Nath and Mehtre, 2014; Avdiienko *et al.*, 2015; da Costa *et al.*, 2016; Saracino *et al.*, 2018; Vidas *et al.*, 2014; Feldman *et al.*, 2015; Wu *et al.*, 2016; Lindorfer *et al.*, 2015; Chen *et al.*, 2017; Yang *et al.*, 2014; Cen *et al.*, 2015; Zhu *et al.*, 2016; Idrees *et al.*, 2017; Dam and Touili, 2017; Chen *et al.*, 2016; Leeds *et al.*, 2017; Lin *et al.*, 2017) brought permissions just as some different features and used ML ways to deal with recognize malicious applications. This methodology-accomplished precision as over 94% (Wei *et al.*, 2019), Because applying permission security is critical for attackers to achieve their hacking goals, permission is the most commonly used static feature in Android. For example, if an application has a calling feature, the Android framework will check if the application has granted the necessary permission to access the calling feature. Based on this circumstance, permissions are given more weight in existing malicious program detection research than other static attributes.

4.3. App Component

These are essential structure squares of an Android application. These components are the section that focuses on the framework to access the application as they are linked with the AndroidManifest.xml, which describes how these components interact. The four major components of an Android application are Activity, Service, Broadcast Receiver, and Content Provider (Wei *et al.,* 2019).

Some work (Chen *et al.,* 2014; Nath and Mehtre, 2014; Avdiienko *et al.,* 2015; da Costa *et al.,* 2016; Saracino *et al.,* 2018; Vidas *et al.,* 2014) considered action as features in Malicious application recognition. (Chen *et al.,* 2014) removed the number of exercises and different features to identify malignant applications. Studies (Nath and Mehtre, 2014; da Costa *et al.,* 2016; Vidas *et al.,* 2014) further applied assistance and broadcast beneficiaries as features in malicious application discovery. Feldman *et al.* (2015) picked the recurrence of highly need recipients and manhandled administrations to recognize malicious applications. The FPR and FNR were both around 10%, but the outcome of perfection was up to 90% (Wei *et al.,* 2019). While Mohsen (Wu *et al.,* 2016) examined the code and researched examples of Broadcast collector parts of malicious applications. The tests demonstrated that utilizing the Broadcast recipients with permissions expanded malicious applications' forecast precision to 97%.

4.4 Filtered Intent

The intent message is used to handle the correspondence between parts of the equivalent or assorted applications by sending point objects. In arrange to prompt the Android system, every one of them has at least one expectation channel. Expectation channels offer assistance app components that dismiss the undesirable entomb and take off the specified intents. They are portrayed within the show records and used in malicious application locations. Lindorfer *et al.* (2015) extracted the intent which gets a response by application via the transmission gatherer. (Chuang and Wang, 2015) gathered dormant features including filtered intents by Android foundation record. They utilized SVM for location reasons and the exploratory comes about appeared that DREBIN recognized 94% of malicious applications with less false caution. (Chen *etal.,* 2017) used a blend of intents and permissions for perceiving Android malicious applications. They advanced the process to fruition with gathering techniques additionally, coming to fruition in 99.8% precision.

4.5. API

Application Programming Interface calls represent how an application collaborates with the Android structure. Each application requires APIs to connect with the device. Consequently, some work utilizes APIs as features for malicious application detection. It is crucial to catch the API calls and the conditions among these calls. This data can be obtained via static examination and dynamic investigation.

Various methodologies are available for Android malicious application detection by investigating applications API. A few examinations (Wermke *et al.,* 2018; Chen *et al.,* 2014; Avdiienko *et al.,* 2015; Dam and Touili, 2017; Grace *et al.,* 2012; Song*et al.,* 2016; Martín *et al.,* 2016; Chuang and Wang, 2015; Liu *et al.,* 2015; Li and Li, 2015; Dam and Touili, 2017; Wang *et al.,* 2018; Ham *et al.,* 2014; Bhandari *et al.,* 2015; Yuan *et al.,* 2016; Ozdemir and Sogukpinar, 2014; Petsas *et al.,* 2014 Suarez-Tangil *et al.,* 2017; Feldman, *et al.,* 2015) fetched APIs just as some different features and used AI to distinguish malicious applications. Studies (Wermke *et al.,* 2018; Zhu *et al.,* 2016; Chuang and Wang, 2015; Ham *et al.,* 2014) are all thought to be confined API's and dubious APIs as features to recognize malicious applications. Rather than using APIs straightforwardly, (Lindorfer *et al.,* 2020) further recognized the API calls having a place with a similar technique in the small code into a square, to be specific API call block. Their exploratory outcomes demonstrated that the API call block outer framed utilizing API calls straightforwardly in Android malicious application recognition.

4.6. File Property

Document properties refer to features available in applications imperative documents, for example, the '.so' and '.zip' records, the little documents, the suspicious records, etc. Android applications are passed on as .apk compacted records. Archived documents can lessen the number of download records when introducing an app. Be that as it may, since the compressed records don't contain confinements of information sort, in some cases, are utilized to bring pernicious payloads as .so files and .zip records. Due to this, a few works use nearness or nonappearance of .zip records and .so records as features. (Faruki *et al.,* 2013) has chosen the nearness of '.so' or '.zip' records as features for Android malicious application discovery for illustration.

Table 2: Models Used in Different Papers

Work	Year	Analysis	Features	ML Techniques
Idrees *et al.*	2017	Static	Permissions, Intent	Feature Importance (FI), Ensemble Learning (EL)
Kouliaridis	2018	Dynamic	Network Usage	Base Models
Tao *et al.*	2018	Static	API Calls	Base Models
Wang *et al.*	2019	Dynamic	Permissions, Intent, API Calls	Base Models
Potha *et al.*	2020	Static	Permissions, Intent	Base Models, EL
Alzaylaee *et al.*	2020	Hybrid	Permissions, Intent, API Calls, Action/Events	Base Models, F1
Taheri *et al.*	2020	Static	Permissions, Intent, API Calls	Base Models, FI
Millar *et al.*	2020	Static	Permissions, OpCodes, API Calls	Base Models
Cai *et al.*	2021	Static	Components, API Calls, Intents, Shell commands	Base Models with weighted mapping, FI
Kouliaridis *et al.*	2021	Static	Permissions, Intents	Base Models, EL, DR

Whereas (Faruki *et al.*, 2013) recognized malicious applications based on the nearness and nonattendance of zip records interior the most application chronicle. They prepared up to 15,000 applications from Google Play out of which 732 known malicious applications (Wei *et al.*, 2019). The tests confirmed that their strategy can discover 95% of malicious applications and taken a toll on 13% of the non-malicious applications on normal over different platforms.

Table 2 shows the list of machine learning models used in different research papers according to analysis and feature based techniques.

5. Detecting Android Application Methods

Current investigation techniques of distinguishing Android applications principally comprise of static, dynamic, half and half, and meta-information examination. We present these examination strategies momentarily and order the studied papers as per the scientific categorizations of utilized features.

5.1. Static Analysis

The android stage turns into the objective of malware engineers and endures genuine kinds of malicious application threats due to its popularity. In return, Security professionals aim to detect malicious applications via static analysis. Static analysis is well known for this purpose and its mechanisms for market protection. In static analysis, applications are decompiled into types of les that define necessary information about those applications. These types of les with required information are then put into computation to confirm that if there are malicious codes available or not. Static Analysis takes fewer assets and time and that is why it is exceptionally well known.

5.2. Dynamic Analysis

Dynamic Analysis distinguishes malicious practices after sending the applications on emulators or genuine gadgets. It creates depictions of network action, processor execution, framework calls, SMS sent, calls, and so on to separate malignant applications from typical ones (Wei *et al.*, 2019). Information fetched through Dynamic Analysis represents the Application's actual behavior. Dynamic Analysis process consumes an excessive amount of time and it may not be able to detect such kind of malicious applications which stops themselves from running on testing environments.

5.3. Hybrid Analysis

It is the combination of static, dynamic, and also meta-data analysis in the detection system. Therefore, Hybrid Analysis contains advantages and disadvantages of static analysis and dynamic analysis. This is the most complete investigation since it dissects both establishment records and application practices at runtime. Therefore it consumes an excessive amount of time and Android operating system resources.

5.4. Meta-Data Analysis

Meta-Data Analysis can not be delegated static examination nor dynamic investigation since it has nothing to do with the application itself. This is a sort of indirect application analysis to identify malicious behavior in a malware-infected application.

6. Results and Discussion

We discovered 5,683,694 mobile malicious installation packages in 2020, which was 2,100,000 more than in 2019 as shown in Figure 5.

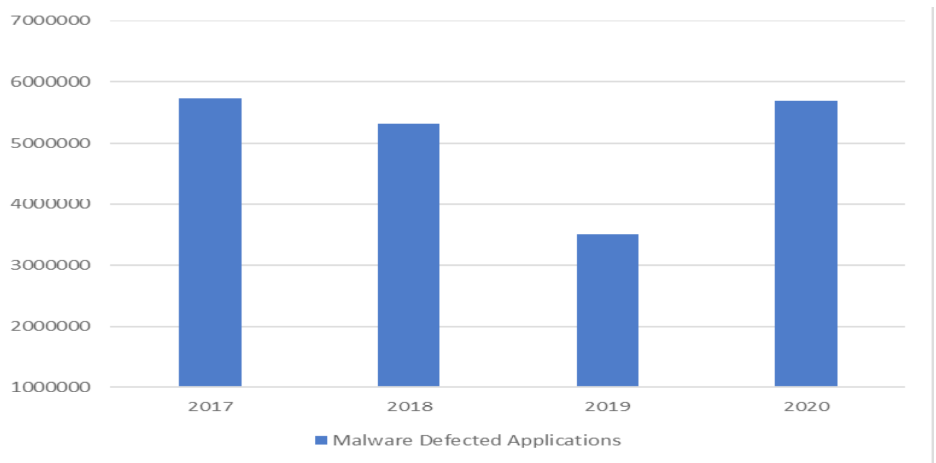

Figure 5: Number of Installed Malicious Application Packages

Table 3 shows that Iran (67.78%) was the country with the most consumers who had been attacked, owing to AdWare's relentless spread. Android is a mobile operating system. The Notifiers are a family of applications. RiskTool is an alternative Telegram client that we have detected. Another common threat was

Table 3: Number of Users Infected According to Country

Country	Percentage of Users Attacked
Iran	67.78
Algeria	31.29
Bangladesh	26.18
Morocco	22.67
Nigeria	22.00
Saudi Arabia	21.75
India	20.69
Malaysia	19.68
Kenya	18.52
Indonesia	17.88

AndroidOS.FakGram.d. Although this isn't malware, communications transmitted through the program may end up in the hands of undesired individuals. Trojan.AndroidOS.Hiddapp.bn was an often-found malicious malware whose goal was to deliver adware to an infected device.

Algeria came in second with 31.29% of the vote. In that country, the AdWare.AndroidOS.FakeAdBlocker and AdWare.AndroidOS.HiddenAd families were the most common. Trojan-Dropper was one of the most often used harmful malware. AndroidOS.Agent.ok and Trojan.AndroidOS.Agent.sr are two AndroidOS. Agent.ok variants. Bangladesh rounded out the "top three" with 26.18%, with the FakeAdBlocker and HiddenAd adware families being the most common.

Finally, the Figure 6 represents the most widely used base categorization models among the publications surveyed. The random forest appears to be the most common classifier, followed by SVM and Naive Bayes.

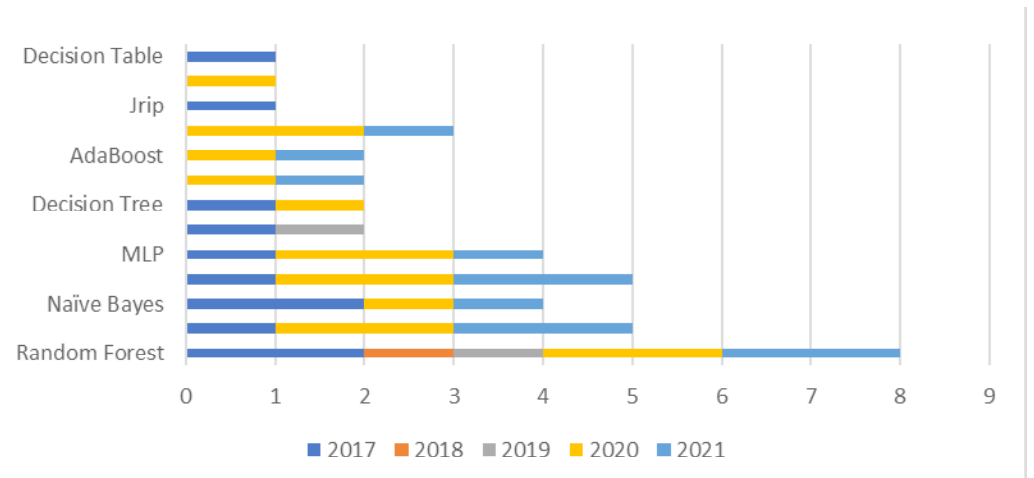

Figure 6: Machine Learning Classifiers Used According to Year

7. Conclusion and Future Work

In this paper, we have performed our analysis to capture and analyze the behavior of system call traces made by each application during their run time. We conclude that using dynamic analysis for malware detection using the system call analysis can be efficiently employed to classify the applications as malicious. Currently, static analysis is being used to detect and monitor the behavior of malicious applications that employ complex obfuscation techniques.

Funding

This research received no external funding.

Data Availability Statement

Not applicable.

Conflicts of Interest

The authors declare no conflict of interest.

References

Aafer, Y., Du, W. and Yin, H. (2013). DroidAPIMiner: Mining API-level Features for Robust Malware Detection in Android, in Proc. Int. Conf. Security. Privacy Commun. Syst., 86–103, Sydney, NSW, Springer, Australia.

Aafer, Y., Du, W. and Yin, H. (2013). DroidAPIMiner: Mining API-Level Features For Robust Malware Detection in Android, in Proc. Int. Conf. Security. Privacy Commun. Syst., 86–103, Sydney, NSW, Springer, Australia.

Alzaylaee, M.K., Yerima, S.Y. and Sezer, S. (2020). DL-Droid: Deep Learning Based Android Malware Detection Using Real Devices. *Comput. Secur.*, 89, 101663.

Avdiienko, V., Kuznetsov, K., Gorla, A., Zeller, A., Arzt, S., Rasthofer, S. and Bodden, E. (2015). Mining Apps For Abnormal Usage Of Sensitive Data. in Proc. IEEE/ACM IEEE Int. Conf. Softw. Eng., May, 426–436.

Bhandari, S., Gupta, R., Laxmi, V., Gaur, M. S., Zemmari, A. and Anikeev, M. (2015). DRACO: DRoid Analyst Combo an Android Malware Analysis Framework,'' in Proc. Int. Conf. Security. Inf. Netw., 283–289.

Cai, L., Li, Y. and Xiong, Z. (2021). JOWMDroid: Android Malware Detection Based On Feature Weighting With Joint Optimization Of Weight-mapping And Classifier Parameters. *Comput. Secur.*, 100, 102086.

Cen, L., Gates, C.S., Si, L. and Li, N. (2015). A Probabilistic Discriminative Model For Android Malware Detection With Decompiled Source Code. *IEEE Trans. Dependable Secure Comput.,* 12(4), 400–412.

Chen, K., Liu, P. and Zhang, Y. (2014). Achieving Accuracy And Scalability Simultaneously in Detecting Application Clones on Android Markets. in Proc. Int. Conf. Softw. Eng., 175–186.

Chen, L., Hou, S. and Ye, Y. (2017). SecureDroid: Enhancing Security Of Machine Learning-based Detection Against Adversarial Android Malware Attacks. in Proc. 33rd Annu. Comput. Security. Appl. Conf., Orlando, FL, USA, December, 362–372. [Online]. Available: http:// doi.acm.org/10.1145/ 3134600.3134636.

Chen, S., Tang, Z., Tang, Z., Xu, L. and Zhu, H. (2016). StormDroid: A streaming-glized Machine Learning-based System For Detecting Android Malware. in Proc. ACM Asia Conf. Comput. Commun. Security., 377–388.

Chuang, H.-Y. and Wang, S.-D. (2015). Machine Learning-based Hybrid Behavior Models For Android Malware Analysis. in Proc. IEEE Int. Conf. Softw. Qual., Rel. Security., August, 201–206.

da Costa, K.A.P., da Silva, L.A., Martins, G.B., Rosa, G.H., Pereira, C.R. and Papa, J.P. (2016). Malware Detection In Android-based Mobile Environments Using Optimum-path Forest. in Proc. IEEE Int. Conf. Mach. Learn. Appl., December, 754–759.

Dam, K.-H.-T. and Touili, T. (2017). Learning Android Malware. in Proc. Int. Conf. Available., Rel. Security., 59.

Faruki, P., Ganmoor, V., Laxmi, V., Gaur, M.S. and Bharmal, A. (2013). AndroSimilar: Robust statistical feature signature for Android malware detection. in Proc. Int. Conf. Security. Inf. Netw., 152–159.

Feldman, S., Stadther, D. and Wang, B. (2015). Manilyzer: Automated Android Malware Detection Through Manifest Analysis. in Proc. IEEE Int. Conf. Mobile Ad Hoc Sensor Syst., October, 767–772.

Grace, M., Zhou, Y., Zhang, Q., Zou, S. and Jiang, X. (2012). RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. in Proc. Int. Conf. Mobile Syst., Appl., Services, 281–294.

Ham, Y. J., Moon, D., Lee, H.-W., Lim, J. D. and Kim, J. N. (2014). Android Mobile Application System Call Event Pattern Analysis For Determination of Malicious Attack. *Int. J. Secure. Appl.*, 8(1), 231–246.

Idrees, F., Rajarajan, M., Conti, M., Chen, T.M. and Rahulamathavan, Y. (2017). Android: A Novel Android Malware Detection System Using Ensemble Learning Methods. *Comput. Security*., 68, 36-46.

Kouliaridis, V., Barmpatsalou, K., Kambourakis, G. and Wang, G. (2018). Mal-Warehouse: A Data Collection-as-a-Service of Mobile Malware Behavioral Patterns. In Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), October 8-12, 1503–1508, Guangzhou, China.

Kouliaridis, V., Potha, N. and Kambourakis, G. (2021). *Improving Android Malware Detection Through Dimensionality Reduction Techniques. In Machine Learning for Networking*, 57–72, Springer International Publishing, Paris, France.

Leeds, M., Keffeler, M. and Atkison, T. (2017). A Comparison of Features For Android Malware Detection. in Proc. ACM Southeast Regional Conf., 63–68.

Li, Q. and Li, X. (2015). Android Malware Detection Based On Static Analysis Of Characteristic Tree. in Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery, 84–91.

Lin, J., Zhao, X. and Li, H. (2017). Target: Category-based Android Malware Detection Revisited. in Proc. ACSW, 74:1–74:9.

Lindorfer, M., Neugschwandtner, M. and Platzer, C. (2015). Marvin: Efficient and Comprehensive Mobile App Classification Through Static And Dynamic Analysis. in Proc. Comput. Softw. Appl. Conf., July, 422–433.

Liu, Z., Lai, Y. and Chen, Y. (2015). Android Malware Detection Based On Permission Combinations. *Int. J. Simul. Process Model.*, 10(4), 315–326.

Martín, A., Menéndez, H.D. and Camacho, D. (2016). String-based Malware Detection for Android Environments. in Proc. Int. Symp. Intell. Distrib. Comput., 99–108, Paris, France: Springer.

Millar, S., McLaughlin, N., del Rincon, J.M., Miller, P. and Zhao, Z. (2020). DANdroid: A Multi-View Discriminative Adversarial Network for Obfuscated Android Malware Detection; Association for Computing Machinery: New York, NY, USA.

Nath, H. V. and Mehtre, B.M. (2014). Static Malware Analysis Using Machine Learning Methods in Proc. Int. Conf. Security. Comput. Netw. Distrib. Syst., 440–450, Springer, Thiruvananthapuram, India.

Ozdemir, M. and Sogukpinar, I. (2014). An Android Malware Detection Architecture Based on Ensemble Learning. *Trans. Mach. Learn. Artif. Intell.*, 2(3), 90–106.

Petsas, T., Voyatzis, G., Athanasopoulos, E., Polychronakis, M. and Ioannidis, S. (2014). Rage Against The Virtual Machine: Hindering Dynamic Analysis Of Android Malware. in Proc. Eur. Workshop Syst. Security., 5.

Potha, N., Kouliaridis, V. and Kambourakis, G. (2020). An Extrinsic Random-based Ensemble Approach For Android Malware Detection. *Connect. Sci.,* 1–17.

Sadeghi, A., Bagheri, H., Garcia, J. and Malek, S. (2017). A Taxonomy And Qualitative Comparison Of Program Analysis Techniques For Security Assessment Of Android Software. *IEEE Trans. Softw. Eng.*, 43(6), 492–530.

Saracino, A., Sgandurra, D., Dini, G. and Martinelli, F. (2018). MADAM: Effective and Efficient Behavior-based Android Malware Detection And Prevention. *IEEE Trans. Depend. Sec. Comput.*, 15(1), 83–97. DOI: 10.1109/TDSC.2016.2536605.

Song, J., Han, C., Wang, K., Zhao, J., Ranjan, R. and Wang, L. (2016). An Integrated Static Detection And Analysis Framework for Android. *Pervasive Mobile Comput.*, 32, 15–25.

Suarez-Tangil, G., Dash, S.K., Ahmadi, M., Kinder, J., Giacinto, G. and Cavallaro, L. (2017). DroidSieve: Fast and Accurate Classification of Obfuscated Android Malware. in Proc. ACM Conf. Data Appl. Security. Privacy, 309–320.

Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z. and Conti, M. (2020). Similarity-based Android Malware Detection Using Hamming Distance Of Static Binary Features. *Future Gener. Comput. Syst.,* 105, 230–247.

Tao, G., Zheng, Z., Guo, Z. and Lyu, M.R. (2018). MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs. *IEEE Trans. Reliab.*, 67, 355–369.

Vidas, T., Tan, J., Nahata, J., Tan, C.L., Tague, P. and Tague, P. (2014). A5:Automated Analysis of Adversarial Android Applications. in Proc. ACM Workshop Security. Privacy Smartphones Mobile Devices, 39–50.

Wang, S., Chen, Z., Yan, Q., Yang, B., Peng, L. and Jia, Z. (2019). A Mobile Malware Detection Method Using Behavior Features In Network Traffic. *J. Netw. Comput. Appl.*, 133, 15–25.

Wang, W., Li, Y., Wang, X., Liu, J. and Zhang, X. (2018). Detecting Android Malicious Apps And Categorizing Benign Apps With Ensemble of Classifiers. *Future Gener. Comput. Syst.*, 78, 987–994, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17300742

Wei, Wang., Meichen, Zhao., Gao, Zhenzhen., Xu, Guangquan., Xian, Hequn., Li, Yuanyuan. and Zhang, Xiangliang. (2019). Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy, and Directions. *IEEE Access*, 1-1. 10.1109 / ACCESS.2019.2918139.

Wermke, D. , Huaman, N., Acar, Y., Reaves, B., Traynor, P. and Fahl, S. (2018). A Large Scale Investigation Of Obfuscation Use In Google Play, in Proc. 34th Annu. Comput. Security. Appl. Conf., 222–235. [Online].

Wu, S., Zhang, Y. and Xiong, X. (2016). Efficient Privacy Leakage Discovery for Android Applications Based On Static Analysis. *Int. J. Hybrid Inf. Technol.*, 9(3), 199–210.

Xu, G. , Zhang, Y., Sangaiah, A. K., Li, X., Castiglione, A. and Zheng, X. (2019). CSP-E2: An Abuse-free Contract Signing Protocol With Low-storage TTP For Energy-efficient Electronic Transaction Ecosystems, *Inf. Sci.,* 476, 505–515, Feb. DOI: 10.1016 / j.ins.2018.05.022.

Yang, C., Xu, Z., Gu, G., Yegneswaran, V. and Porras, P.A. (2014). Droid- Miner: Automated Mining And Characterization Of Fine-grained Malicious Behaviors In Android Applications. in Proc. 19th Eur. Symp. Res. Com- put. Security. (ESORICS), Wroclaw, Poland, September, 163–182. DOI: 10.1007 / 978-3-319-11203-9_10.

Yang, F., Zhuang, Y. and Wang, J. (2017). Android Malware Detection Using Hybrid Analysis And Machine Learning Technique,'' in Proc. Int. Conf. Cloud Comput. Security, 565–575, Springer, Nanjing, China.

Yuan, Z., Lu, Y. and Xue, Y. (2016). Droiddetector: Android Malware Characterization And Detection Using Deep Learning. *Tsinghua Sci. Technol.,* 21(1), 114–123.

Zhao, K., Zhang, D., Su, X. and Li, W. (2016). Fest: A Feature Extraction And Selection Tool For Android Malware Detection. in Proc. Comput. Commun., 714–720.

Zhu, J., Wu, Z., Guan, Z. and Chen, Z. (2016). API Sequences Based Malware Detection for Android. in Proc. IEEE Int. Conf. Auton. Trusted Comput. IEEE Int. Conf. Scalable Comput. Commun. It's Associated Workshops Ubiquitous Intell. Comput., 673–676.

Cite this article as: Muhammad Mugees Asif, Sana Asif, Iqra Mubarik and Rabia Hussain (2022). Malicious Applications Detection in Android Using Machine Learning. *International Journal of Artificial Intelligence and Machine Learning,* 2(2), 21-34. doi: 10.51483 / IJAIML.2.2.2022.21-34.