EasyChair Preprint

№ 7694

Android Malware Detection using Deep Learning

Gudla Surya Bharti, Syed Shabeeb Nibras,
Mukkamala Shyam Srujan,
Kesanapalli V K Raghavendra Chowdary and Siriki Bala Murali

April 2, 2022

ANDROID MALWARE DETECTION USING DEEP LEARNING

Ms. Gudla Surya Bharti, Syed Shabeeb Nibras, Mukkamala Shyam Srujan, Kesanapalli V K Raghavendra Chowdary and Siriki Bala Murali

Gitam[Deemed to be University], Visakhapatnam, Andhra Pradesh
bgudla@gitam.edu, syedshabeeb0@gmail.com, shyamsrujan29@gmail.com, vkrkesanapalli@gmail.com, sirikibalu24@gmail.com.

Abstract

A large part of Android's popularity is due to the ease with which it can be operated and the wide range of capabilities it offers its users, both of which have made it a prime target for cyber thieves. Recent malware may get through the cracks of Android's traditional anti-malware solutions, such as those that rely on signatures or monitor power use. This new method uses NEURAL Networks and the KMEANS algorithm to develop a predictive analytics model to identify virus in Android applications. By extracting the permissions list from the Android APK file, we can use these two techniques to detect malware in Android apps. We use the Android Permission Features dataset to develop and evaluate our strategy. Our deep learning method exceeds other methods with a 99 percent accuracy rate, according to the results of our tests. Static analysis, API-calls, and permissions are all examples of Android malware.

1 Introduction

Smartphones are now used more than ever before, and this trend will only continue to grow. By the year 2021, there will be 3.8 billion smart mobile devices in use worldwide [1]. As a result, more than two-thirds of these smartphones are powered by Android. In addition, antivirus software is rarely installed on Android devices by consumers. Even if it is installed, it may not be able to identify malware very well [3]. As a result of the vast number of people who use Android smartphones and the wealth of data they may have access to, the Android operating system may become a more attractive target for cyber criminals. Furthermore, as the number of people using a service grows, so does the amount of sensitive data that can be obtained by a hacker. Using an application that has already been uploaded to Google Play, the attacker can acquire access. Upon doing so, the victim unwittingly provides the attacker with access [5, 6]. Likewise 2.86 million Android apps was accessible for download in the third quarter of 2020 [7]. Furthermore, an average of 16,000 new malware programmes are identified each

day, a monthly average of 482,579 [8]. More advanced malware detection techniques are needed because of the large number of Malware programmes. As far as Android virus identification goes it very well may be isolated into static, dynamic, and Hybrid methodologies. Analyzing an Android app statically extracts its features rather than running it on a real device or an emulator. Because of this, it is important to keep an eye out for suspicious or out-of-the-ordinary behaviour. Code obfuscation and dynamic code loading are two of the most significant drawbacks of this approach, which can nonetheless achieve excellent feature coverage. Nonetheless, unique investigation recovers highlights by executing them on an Android Emulator or a real-life Android device. A benefit of this approach over static analysis may be the discovery of additional characteristics or hazards not picked up by static analysis. In terms of efficiency, static analysis beats dynamic analysis every time [10]. Hybrid analysis, on the other hand, is a combination of static and dynamic analysis. [11] Detection is more successful and efficient with this method. Static analysis is used in our technique.

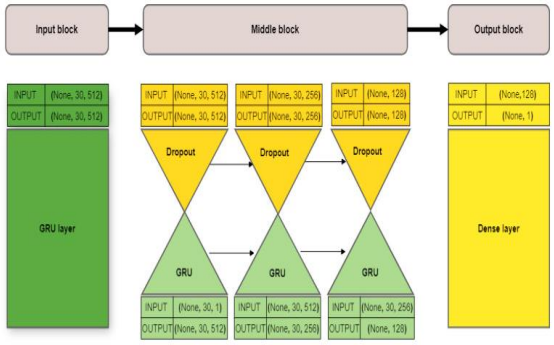

Fig.1: Deep learning model

pseudodynamic analysis and Programming interface call charts for every execution course founded on a profound learning model developed and prepared on an informational collection of around 30 thousand malevolent applications and 25 thousand harmless applications are used to detect malicious Android apps. A graph of API calls was constructed to address all conceivable execution pathways that malware could take during its runtime. A low-layered numeric vector highlight set can be put into a profound neural network as it has low dimensions. As part of this effort, they evaluated several embedding techniques and network setup parameters to ensure that the optimal selection of hyper-parameters was achieved to achieve the maximum accuracy outcomes.

2 Literature Review

2.1 An antivirus api for android malware recognition[3]

Antivirus software is woefully underdeveloped on Android. It is unable to access or screen the Android gadget's record framework or the powerful way of behaving of introduced applications because of stage constraints. After installation, malicious files can be downloaded and other file system changes can be made. As a result, malicious files can be downloaded and executed by any programme, regardless of whether or not it contains overtly dangerous code in its package file. This has serious implications for device security. Adding an antiviral interface to Android's API is the topic of this study. On-request document framework filtering and crossing, on-change record framework observing, and a bunch of

basic strategies that take into account checking of erratic document framework objects without unveiling their substance are three of the most important features of this software. Android antivirus software can use approaches for virus detection comparable to desktop antimalware structure with a help of this interface. As per Android's security plan, client protection is protected by the proposed safeguards. It is possible to provide a level of security on the Android platform equivalent to that of PC antivirus software with our technique.

2.2 Digital investigation that respects privacy [4]:

In order to conduct a thorough forensic investigation, one must disregard the privacy rights of individuals being probed. It's hard to "erase" the impact of publicising private information even if a suspect turns out to be innocent. Forensics investigators are not seldom confronted with the private information of innocent persons when they investigate suspects. Developing platforms to enforce privacy boundaries even for authorised forensics investigators is becoming increasingly important in light of these recent revelations. While there are theoretical frameworks for privacy-respecting digital investigations that can take into account different jurisdictions' needs and preserve subjects' privacy in accordance with warrant authorization, there is no actual model that can do so. Emerging cross-disciplinary research in digital forensics that respects privacy is heading in the direction of dealing with the concerns listed above. In this paper, we lay the groundwork for a multidisciplinary topic of study called "privacy-respecting digital investigation". After that, we'll take a look at the most important research being done in several fields relevant to the subject and develop the present research problems. According to EU, US and APEC rules, we look at possible privacy issues throughout digital research.

In what ways are older mobile device users more tech-savvy? [5] :

As Asia's most mobile city-state, Singapore also boasts one of the area's quickest maturing populaces. Cell phone clients beyond 45 a years old brought up in a period where data and correspondence innovation were less modern and English language training was more uncommon. They are more averse to be worried about versatile security, given these circumstances. The majority of the 55 people who took part in the study were not aware of the security and privacy dangers that come along with using a mobile device. As a result, we offer advice to this particular user group in an effort to reduce the number of people who are victims of cybercrime. Our research shows that elder digital immigrants need to be educated about cybercrime on a regular basis, as well as to cultivate a sense of security.According to a recent measurement of google play [6,7],

In spite of the way that huge number of Android clients download and use outsider applications from the Google Play store, very little is referred to about these applications overall. We made PlayDrone, the main adaptable crawler for the Google Play store, and have been utilizing it to record and examine in excess of 1,100,000 Android applications consistently, making it the biggest list of its sort. PlayDrone utilizes an assortment of hacking strategies to get past Google's limitations on ordering material from the Google Play store and makes the source code for more than 880,000 free applications accessible on the web. We exhibit the handiness of PlayDrone in decompiling and breaking down application content by investigating four already ignored issues: the portrayal of Google Play application content at large scale and its development after some time, library use in applications and its effect on application movability, duplicative application content in Google Play, and the insufficiency of OAuth and related help validation instruments bringing about vindictive clients having the option to effectively acquire unapproved admittance to client information and assets on Amazon Web Services and Facebook.

2.5 A overview of machine learning-based malware detection methods for Android [9]:

In the mobile environment, Android apps are growing at a quick pace, but Android malware is also sprouting at an unending rate. The subject of Android malware detection has been studied by a large number of researchers, who have put out hypotheses and solutions from a variety of viewpoints. Machine learning is a viable method for detecting Android malware, according to current research. Nevertheless, there are studies that have examined several challenges linked to AI based malware recognition for Android. We feel our work supplements the earlier assessments by covering a bigger scope of features of the point. There is a wide scope of Android malware recognition techniques that utilization AI in this paper. Introduction to Android applications, security measures and Android malware classification are all covered briefly in this introduction. Our next step is to examine and evaluate the current state of research in important areas for example, test gathering, information preprocessing, highlight choice, AI models and calculations, as well as discovery adequacy appraisal, all with an emphasis on AI. As an end, we take a gander at the eventual fate of Android malware identification research utilizing AI. This audit will help scholastics get an intensive image of Android malware recognition in view of AI. It could then be utilized as a beginning stage for new study and as a general reference for researchers in the field.

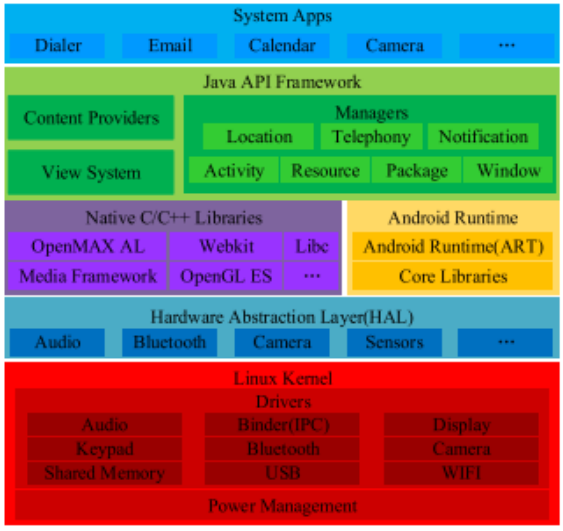

Fig.2: Android platform architecture

2.6 Static and dynamic analysis for android malware detection [10]:

Static plus dynamic features are used to compare the behaviour of virus and genuine apps in this study. Stable analysis takes into account the permissions needed for an application. Droidbox is a tool that we employ in dynamic, nevertheless. This Android sandbox can monitor some programme behaviours like network activity, file system activity, cryptography activity, information leakage, and so forth. Here, we take into account these actions and the dynamic API calls of programmes. We'd like to see an Android malware detector implemented, so that apps can be identified as potentially harmful or not before they are installed.

3 Implementation

To If an APK requests any permissions, such as the ability to send SMS messages or make programmed phone calls, the author builds a features vector and uses that to detect malware in Android apps. Extract all of the permissions in APK and put a value of one if any APK requests permission and a value of zero otherwise. The KMEANS algorithm and NEURAL Networks will use this feature vector to build a machine learning model. Predicting whether or not an android APK is infected with malware will be done using this model, which will be used to new test features.

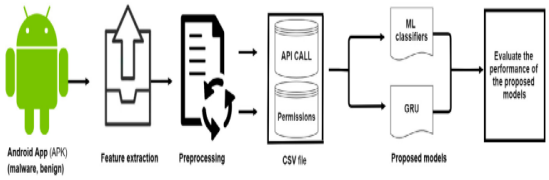

Fig.3: Malware detection phases

To steal sensitive data from a smartphone and distribute it to malicious users, criminal programmers may ask Android for permission to access sensitive data. We'll be able to determine whether or not a user requesting a particular permission is acting in good faith by analyzing it with our app.

The following is a list of the android permission features retrieved from the APK.

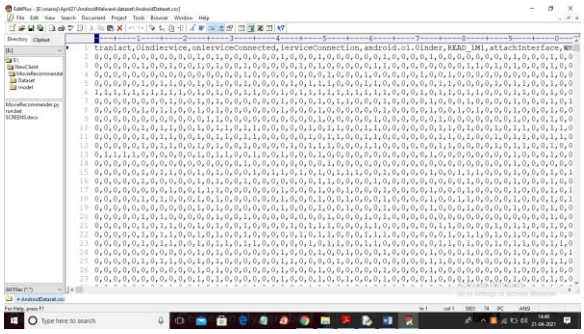

Fig.4: Android permissions feature dataset.

First, the dataset column names are android permission, and the remaining rows are either 0 or 1 for the values. The value 1 will be used if Android requests it, else the value 0 will be used. We have a class label of 0 or 1 in the last column of each row, which indicates if the record is malware or not. An algorithm that can anticipate malware from fresh Android test signatures will be trained using the aforesaid dataset.

The first phase of this endeavour is depicted in Figure 3 as a blend of malware and non-malicious components. APK files for Android are taken from a dataset that is freely available to the public at large. Then, we compose our own Python script in the Jupyter scratch pad climate to remove the static highlights. Programming interface calls and authorizations are the primary attributes. As an information outline for the preparation cycle, these highlights are arranged and put away in a Comma Separated

Values (CSV) record. Finally, we conduct a 10-fold cross-validation test on all classifiers, as well as calculate the appropriate metrics for evaluation.

3.1 Algorithms

NEURAL NETWORKS:

Computerized reasoning, AI, and profound advancing all advantage from brain organizations' capacity to mirror the human mind's working. Profound learning strategies depend on brain organizations, regularly known as counterfeit brain organizations (ANNs) or mimicked brain organizations (SNNs). Since they imitate the manner in which natural neurons speak with each other, their name and construction are gotten from the human cerebrum too.

At least one information layers, at least one secret layers, and a result layer make up a fake brain organization (ANN), which is made out of a hub layers structure. Each counterfeit neuron, or hub, has a specific weight and limit related with it. Whenever a hub's result surpasses a specific limit, it is initiated and starts shifting data to the network's following layer. Alternatively, no information is sent to a next network layer.

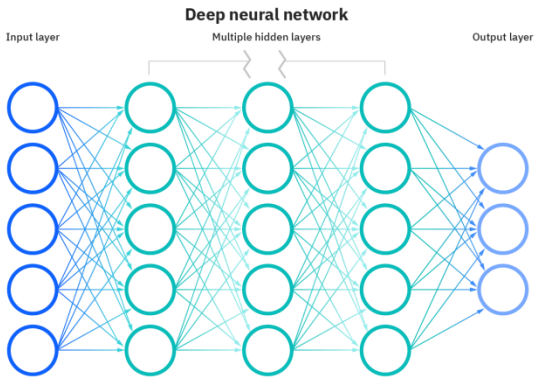

Fig.5: Deep neural network architecture

Preparing information is fundamental for brain organizations to learn and work on after some time. In software engineering and man-made brainpower, nonetheless, these learning calculations are amazing assets that permit us to order and group information rapidly and precisely. Speech acknowledgment and picture acknowledgment assignments can be finished in seconds as opposed to hours when done by a human subject matter expert. Google's pursuit calculation is one of the most outstanding known brain organizations.

I'm curious how neural networks function.

This model is a linear regression model for each node, which includes a bias and a result. This is how the equation might appear:

To calculate wixi, add bias to each term.

F(x) is equal to 1 for all values of W and B that are greater than or equal to zero; for all values of W and B that are less than or equal to zero.

After determining an input layer, weights are assigned to each layer. For each variable, these weights help to establish its importance, with bigger weights contributing more heavily to the output than less weighted variables. Finally, the sum of all inputs is multiplied by their individual weights. There follows an activation function which decides the output. This output "fires" the node, sending information to the next tier of the network if it reaches a predetermined threshold. Because of this, one node's result becomes the next node's input. This brain network is a feedforward network because of the way data is passed from one layer to the next.

KMEANS:

For an initial look at the dataset's structure, many practitioners turn to Kmeans clustering, one of the most widely-used clustering techniques. Group data points into non-overlapping groupings with kmeans. When the clusters have a spherical shape, it does a great job. Clusters' geometric shapes depart from spherical shapes, resulting in a loss of efficiency. It also cannot learn the number of collection from the information and requires it to be pre-defined, which is another problem. Good practitioners should understand the assumptions underlying algorithms and methodologies in order to assess the benefits and drawbacks of each approach. When and how to use each method will be made easier with this information.

Unsupervised learning is how K-Means clustering works. This clustering does not use labelled data, as in supervised learning. A clustering algorithm known as K-Means is used to divide things into groups based on their similarities and differences.

"K" is a numeric designation. Please specify how many clusters you want to construct.

K = 2 denotes the presence of two clusters.

To determine the best or most optimal value of K, there is an algorithm that can be applied to any given set of data.

Let's look at an example from the game of cricket to better appreciate what k-means are all about. Think of it this way: You've just gotten cricket player stats from a number of different countries, each of which includes a breakdown of the player's last ten innings in terms of runs and wickets. In light of this knowledge, it is necessary to divide the data into two groups, namely batsmen and bowlers.

An Overview of the K-Means Clustering Method:

K-Means clustering is utilized in a wide range of real-world applications, including:

1. Academic achievement. 2. methods for detecting disease 3. Search engines are the third factor. 4. Sensor networks that use wireless communication

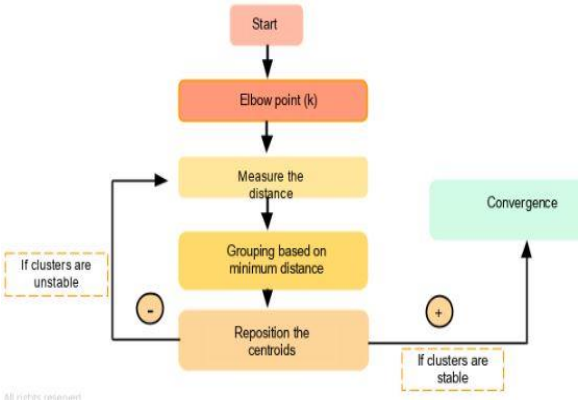

Fig.6: Clustering using k-means is depicted in the flowchart shown above.

This algorithm's primary purpose is to identify clusters in the incoming data. It is possible to accomplish this in a handful of different ways. Specifying K as a value allows us to use the trial-and-error method (e.g., 3,4, 5). We'll keep tweaking the value till we find the best clusters as we go along.

4 Experimental Result

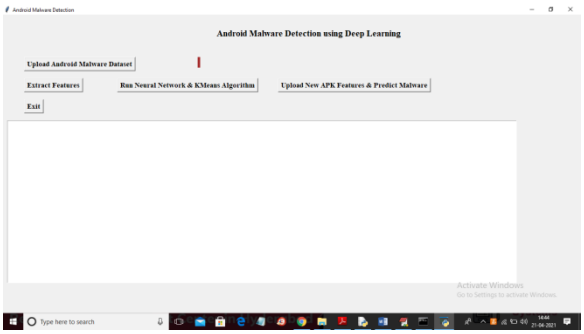

Fig.7: Home screen

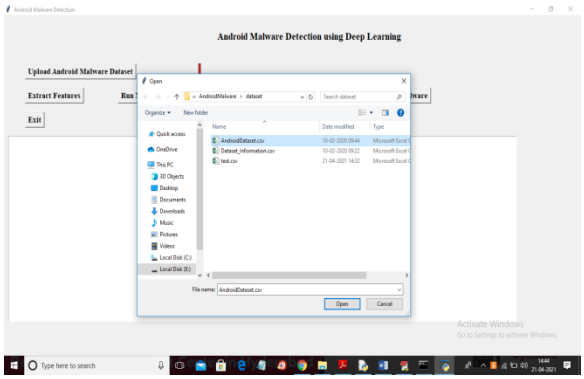

Fig.8: Upload malware dataset

In above screen dataset loaded and now click on 'Extract Features' button to read all features from dataset and to get below screen

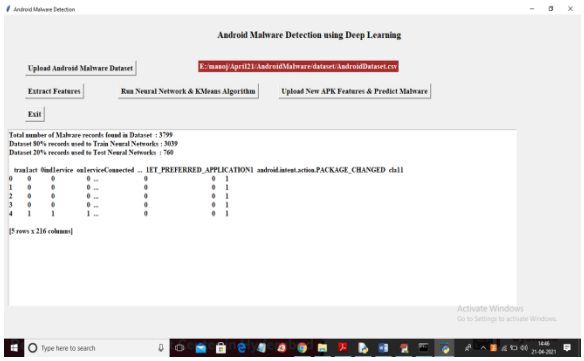

Fig.9: Extract features

The dataset shown in the first line of the screen has 3799 records, with 3039 records used to train neural networks and 760 records used to assess the accuracy of those networks, with the dataset being divided 80/20. The KMEANS algorithm is used to identify comparable data, which will subsequently be fed into a Neural Network and used to produce accuracy and confusion matrices, respectively.

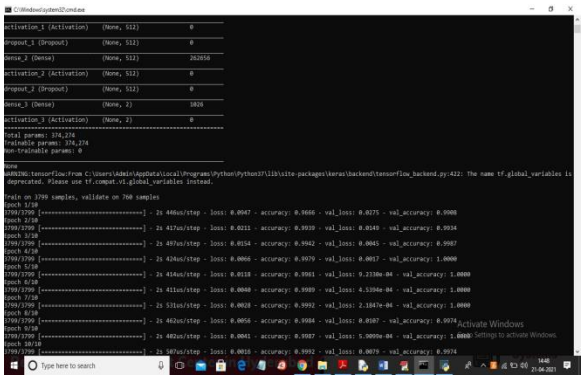

Fig.10: Neural network screen

To train neural networks, we used 10 epochs, and as the number of epochs increased, so did the correctness of the trained sample, indicating that the sample is capable of properly predicting test data once it has been built.

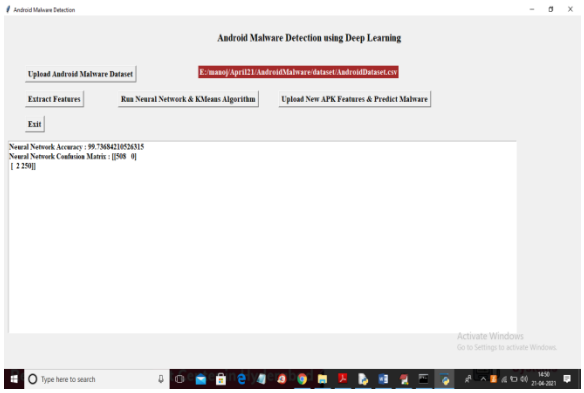

Fig.11: Neural network accuracy

Only two records were inaccurately predicted in the above confusion matrix, which resulted in a 0.23 percent decrease in the neural network's accuracy.

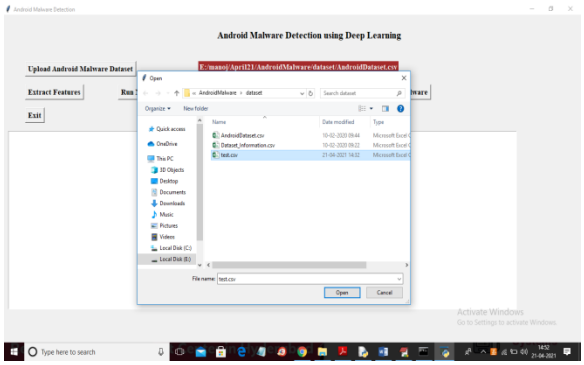

Fig.12: Upload new APK features.

In above screen selecting and uploading ‘test.csv’ file.

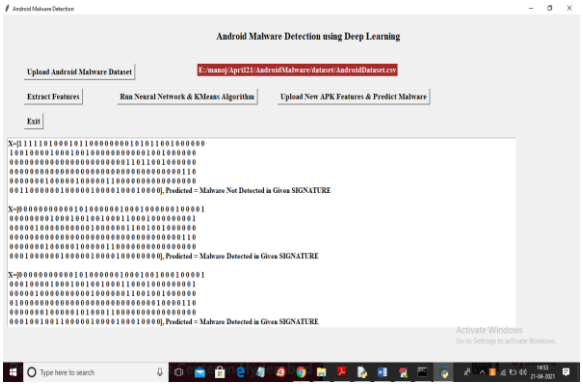

Fig.13: Prediction result

All Android permission test features are shown in square brackets on the above screen, and the next square bracket shows if MALWARE was discovered in that record.

5 Conclusion

Based on permission attributes collected from the APK, the suggested classifiers were trained using the dataset evaluated using static analysis of real-world malware and benign Android programmes. It was determined that detecting Android malware models was worth relying on more than just looking at permissions and API calls, therefore we looked at both. With consents and API calls static highlights, a profound learning classifier has outflanked all others in Android malware detection.

6 Future Scope

The more accurate detection will be possible in the future, as well as a reduction in time, power and resources

References

[1] Number of Smartphone and Mobile Phone Users Worldwide in 2020/2021: Demographics, Statistics, Predictions. https:// financesonline.com/number-of-smartphone-users-worldwide/. Accessed: 2020-Jan-11.

[2] Mobile Operating System Market Share Worldwide 2020. https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed: 2020-Jan-11.

[3] Rafael Fedler, Marcel Kulicke, and Julian Schutte. An antivirus api for android malware recognition. In ¨ 2013 8th International Conference on Malicious and Unwanted Software:" The Americas"(MALWARE), pages 77–84. IEEE, 2013.

[4] Ali Dehghantanha and Katrin Franke. Privacy-respecting digital investigation. In 2014 Twelfth Annual International Conference on Privacy, Security and Trust, pages 129–138. IEEE, 2014.

[5] C Chia, K-KR Choo, and D Fehrenbacher. How cyber-savvy are older mobile device users? In Mobile Security and Privacy, pages 67–83. Elsevier, 2017.

[6] Nicolas Viennot, Edward Garcia, and Jason Nieh. A measurement study of google play. In The 2014 ACM international conference on Measurement and modeling of computer systems, pages 221–233, 2014.

[7] Google Play: number of available apps by quarter 2020 — Statista. https://www.statista.com/statistics/289418/ number-of-available-apps-in-the-google-play-store-quarter. Accessed: 2020-Jan-12.

[8] Global Android malware volume 2020 — Statista. https://www.statista.com/statistics/680705/ global-android-malware-volume/. Accessed: 2020-Jan-12.

[9] Kaijun Liu, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, and Haifeng Liu. A review of android malware detection approaches based on machine learning. IEEE Access, 8:124579–124607, 2020.

[10] Krishna Sugunan, T Gireesh Kumar, and KA Dhanya. Static and dynamic analysis for android malware detection. In Advances in Big Data and Cloud Computing, pages 147–155. Springer, 2018.

[11] Mahima Choudhary and Brij Kishore. Haamd: hybrid analysis for android malware detection. In 2018 International Conference on Computer Communication and Informatics (ICCCI), pages 1–4. IEEE, 2018.

[12] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. Yes, machine learning can be more secure! a case study on android malware detection. IEEE Transactions on Dependable and Secure Computing, 2017.

[13] ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, and Djedjiga Mouheb. Maldozer: Automatic framework for android malware detection using deep learning. Digital Investigation, 24:S48–S59, 2018.

[14] Leong Chan, Ian Morgan, Hayden Simon, Fares Alshabanat, Devin Ober, James Gentry, David Min, and Renzhi Cao. Survey of ai in cybersecurity for information technology management. In 2019 IEEE Technology & Engineering Management Conference (TEMSCON), pages 1–8. IEEE, 2019.

[15] Arash Habibi Lashkari, Andi Fitriah A Kadir, Laya Taheri, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In 2018 International Carnahan Conference on Security Technology (ICCST), pages 1–7. IEEE, 2018.