Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019)
IEEE Xplore Part Number: CFP19J32-ART; ISBN: 978-1-5386-9439-8

Android Malware Detection using Deep Learning

Sagar Sabhadiya
Department Computer Engineering
Sarvajanik College of Engineering & Technology
Surat, Gujarat, India
sagar.sabhadiya100@gmail.com

Prof. Jaydeep Barad
Department Computer Engineering
Sarvajanik College of Engineering & Technology
Surat, Gujarat, India
jaydeep.barad@scet.ac.in

Prof. Jaydeep Gheewala
Department Computer Engineering
Sarvajanik College of Engineering & Technology
Surat, Gujarat, India
jaydeep.gheewala@scet.ac.in

***Abstract*— Android has become the most standard smartphone operating system. The rapidly growing acceptance of android has resulted in significant increase in the number of malwares when compared with earlier years. There exists plenty of antimalware programs which are designed to efficiently protect the user's sensitive data in mobile systems from such attacks. Here, I have examined the different android malwares and their methods based on deep learning that are used for attacking the devices and antivirus programs that act against malwares to care for Android systems. Then, we have discuss on different deep learning based android malware detection techniques such as, Maldozer, DroidDetector, DroidDeepLearner, DeepFlow, Droid Delver and Droid Deep. We aim to implement a model based on deep learning that can automatically identify whether an android application is malware infected or not without installation.**

Keywords—Android Security, Malware Detection Technique, Deep Learning based Malware Detection

I. INTRODUCTION

In our daily life Mobile Applications have become an essential part since countless facilities are providing to us by using Mobile Apps. It will change the way of communication, as the apps are installed on most of the smart devices. Mobile devices have refined sensors like cameras, gyroscopes, microphones and GPS. These several sensors open up entire innovative world of applications for the users and create massive quantities of data containing highly complex data. Security solutions are therefore needed to defend operators from malicious applications that exploit the complexity of smart devices and their complex data. Android OS physically grows through the power of a wide range of smart devices. In mobile computing industry, it has largest part with 85% in 2017 due to its vulnerable source distribution [1].

Currently on Android platforms to defend against malware is a risky communication system that notifies users for the required permissions earlier each application is installed. This system is slightly ineffective because it offers permissions on its personal. To distinguish malware from benign applications, the user want excessively much methodical knowledge. The same permissions are required for the both benign and malicious application, consequently we cannot be distinguished by this permission based system. Generally, the permission based methodologies are largely not developed for the detection of malware, but it is used for the risk assessment [2].

The Android Operating System make malware more difficult for the installation and execution, because of the Android itself provide a several security solution for example Android permission and Google's Bouncer to address the progressively widespread security threats. Every Android application need to ask the user for the permission to execute certain task on Android devices, such as transfer SMS message, during the installation process. Most of the users are allow the permission without even considering what kinds of permissions they demand thus the Android permission system is knowingly weaken. Accordingly, the Android permission system spread the malicious apps itself and it is very challenging in training [3]. On the other hand, the Google's "Bouncer" is provide a service by the Android's official market Google Play in 2012, with the aim to scanning of applications (new and uploaded) automatically. Even if another Android defense line added by the Bouncer, it still has a huge number of limitations. Firstly, the Bouncer can scan the Android application for a limited period of time, allowing a malicious app too effortlessly bypass because of during the scan phase it doing nothing malicious. At the second step, when scanned by the Bouncer, no malicious code must be included in the initial installer. In this case, the malicious app may have a higher chance of avoiding the detection of Bouncer. At time when the application has passed the Bouncer security scan and on the real user's Android device it is installed, the additional malicious code is downloaded by the malicious app to run or associate to its control server and remote control to upload taken data or obtain supplementary commands [3][4].

There are different types of existing malwares such as Trojans, Backdoors, Worms, Spyware, Ransomware and Riskware. They are as follows [7][8]:

- **Trojans [11][12]**

Trojan is a software that looks to deliver several valued functions but as a substitute it provide accommodations a malicious program. Likewise a Trojan horse constantly has a resolution of user communication to be activated.

- **Backdoors**

It provides a basis rights to the malware and create simple way for them to hide from antiviruses. There are different root activities that can control it, such as Exploid, Rageagainstthecage (RATC) and Zimperlich.

- **Worms**

Worm is a one type of software program that generates copies and allocate them to the network.

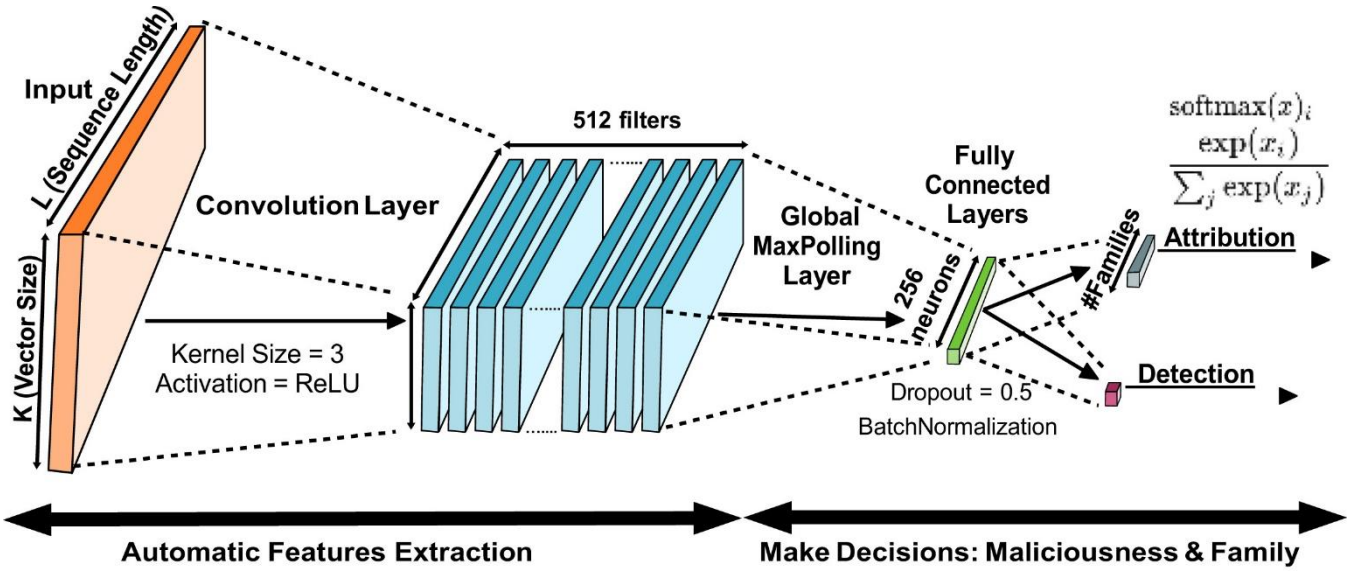

Fig. 1 Neural Network Architecture of MalDozer [1]

- **Spyware [11] [12]**

Spyware observers the user's data such as messages, contacts, bank mTANs, location, etc.

- **Ransomware**

Ransomware stop the user from retrieving their data from the device. It will locked the device and demand for the ransom amount to stop it.

- **Riskware**

Riskware is a one type of software that can delete, copy or modify user's data from the device.

The rest of the report is organized with their respective contents as: In section 2, we discuss the Literature Survey of existing deep learning based Android Malware Detection Techniques. In Section 3, we describe the Comparative Analysis of existing deep learning based Android Malware Detection techniques. In Section 4, we have a conclusion of this survey. At last, the references in Section 5.

II. Literature Survey

There are several literature that discuss techniques for Android Malware Detection. Here we use the different deep learning techniques for Malware Detection [9]. Some of the techniques from the literatures are briefly described as follows:

A. MalDozer [1]

MalDozer is a Convolutional Neural Network based Android Malware Detection System. MalDozer have an simple design in which minimal preprocessing is used to obtain the assembly processes. These are based on the concrete neural network in terms of extraction and detection / attribution of the features. With quick preprocessing and execute neural network, MalDozer is very efficient. Note that the preprocessing procedure is same between training and deployment to check the accuracy of the detection.

The MalDozer workflow consider the DEX file for extracting API call sequences from Android app packages. Android apps are treated as a series of API method calls. The MalDozer takes app API calls into consideration without purifying in which the information they used, they have a malware identification order. After that, MalDozer discretize the API method call sequences which are extracted from DEX file. They used the Discretization algorithm to discretize the API method call sequences. The sequence identifier must be designed to fit into our neural network. By representing a vector for each identifier they can solve this. They used one-hot vectors, where the value of the vector is one in the edge value row and rest is the zero. From each app, they got the vector sequences that contains the instruction of the unique API calls and every vector has static size K.

The multilayer neural network is the main module in the MalDozer structure. There are multiple hyper-parameters like amount of layers and the model's complication. During the deployment time, they try to have the neural network model as humble as likely. To routinely determine the design in the raw method calls, MalDozer depend on the convolution layers. The vector sequence is used as input to the neural network, i.e. an L×K shaped matrix. In the training phase, depend on the app vector classification and its tags, MalDozer trains neural network parameters for: (i) malicious or novel for the recognition task, and (ii) malicious relations for the attribution task. In deployment phase, the embedding model is used to produce the vector sequence and mine the sequence of techniques. At last, they use the vector sequence for detect the android app is malware infected or not.

The MalDozer architecture has high results and exceeds several of the latest standards with a comparatively simple neural network scheme. For the detection task they want only one neuron at the output layer for the reason that the neural network selects the application is malicious or not. There are several neurons for the attribution task, one for each Malware family. The MalDozer have the identical architecture for the detection and attribution. As shown in Fig. 1, at first the Convolution layer with the rectified linear unit (ReLU) and also has the activation function. After that,

MalDozer usage Global Max-Polling layer and that associate it to a Fully Connected layer. They used the Batch normalization to increase the results. At last, they take an output layer in which the amount of neurons depends on the detection or attribution tasks.

B. *Droid Detector [2]*

Droid Detector characterize the Android apps that are malware and benign. For feature extraction from every app they used a static and dynamic analysis. There are main three types of features like required permission, sensitive APIs and dynamic behaviors. These include the extraction of required permissions and sensitive APIs through static analysis, while dynamic behavior is extracted through dynamic analysis. In particular, the installation file (i.e. apk file) from every Android app is all they need. Droid Detector used the different tools like 7-zip, AXMLPrinter2, TinyXml, Baksmali and DroidBox for the features extraction.

There are many different types of deep learning models are available. Here they use the Deep Belief Network (DBN) to construct the deep learning model for characterize Android apps. They implemented the Droid Detector an Android Malware Detection device in Deep Learning based on Deep Belief Network.

The user can identify the Android app is malware infected or not by using Droid Detector and it is available online as an open source as shown in Fig. 2. At first user have to submit the .apk file in the system, Droid Detector will check its reliability and defines whether an Android application is truthful, complete and appropriate. To get the features like permissions and sensitive APIs the Droid Detector executes the static analysis and the dynamic behavior identifies using the dynamic analysis.

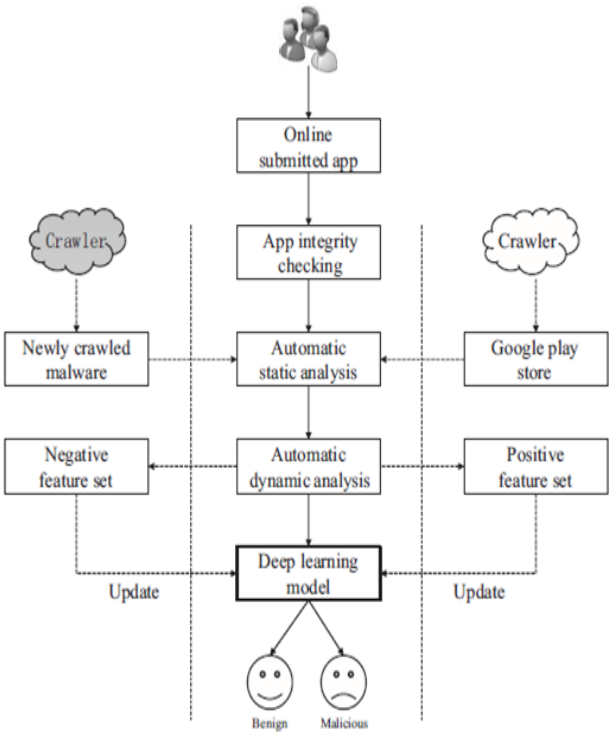

Fig. 2 Framework of DroidDetector [2]

The static and dynamic analysis of Droid Detector has been fully automated. They include the deep learning classification model after 192 binary features described. The reported user will get the result, containing complete information from the integrity check and both analyses. Since new types of applications are constantly emerging, two crawler modules have been designed. For crawling the benign apps from the Google Play Store they used one crawler and the other crawler is used to scroll malware from known sources of malware.

C. *Droid Deep Learner [3]*

Droid Deep Learner is an Android Malware categorization and identification method. Droid Deep Learner uses deep learning method to report the present requirement for malware detection. In this method, they required a set of features for the detection. The features like permissions, APIs, Actions, Intents, IP addresses and URLs are encrypted in the apk file. Based on source recompilation tool, they construct a decoder to decode the apps into readable format. After decoding the app, they get the manifest (*.xml) file which contain the permissions that require for the installation of Android app. There are certain samples of risky permissions which is harmful for the user's data such as *android.permission.CALL_PHONE, android.permission.SEND_SMS, android.permission.WRITE_EXTERNAL_STORAGE, android.permission.ACCESS_LOCATION, android.permission.READ_CONTACTS.*

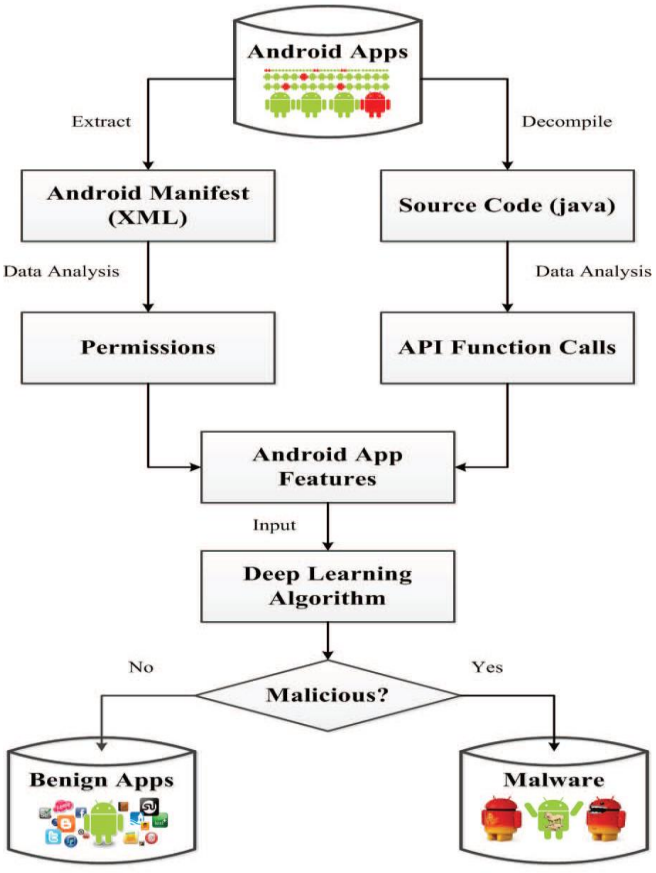

Fig. 3 Overview of DroidDeepLearner Approach [3]

Extracting API calls requires a four steps to access source files of Java. To get class.dex file from all the apps, all apk files had to be decompressed. Then they used the dex2jar for converting dex files into the jar files. After that, they used the jd-cli for converting jar files into the java source files. The jd-cli was taken from Github. At last, according to DroidAPIMiner the top 20 features extracted.

Now they present the general architecture of the proposed Droid Deep Learner method that shown in Fig. 3. We target to get two kinds of features, namely permission and API function calls from Android apps. The Android apps would first be examined to obtain their equivalent manifest files (*.xml) and source files (*.java) to achieve this objective. Then, the API function calls are extracted from java source files and from the manifest files the permissions can be extracted. After that, they will add both of them collected into the feature set of an Android app, which works as an input for both training and testing purpose for the DBN based deep learning model. Based on the deep learning model the classification result can separate malicious apps from the benign apps.

D. *Deep Flow [4]*

Deep Flow is a different deep learning model for classifying malware straight as of the data flows in Android application. Deep Flow architecture use the Deep Belief Network based model in Deep Learning for identifying the Android app is malware or not. First, the features are extracted from apk file. Deep Flow uses the FlowDroid a static exploration device to categorize complex information flows in application. To categorize delicate files pours in an Android app, at first, Flow Droid examines for life span and call-back approaches along with call to sources and sinks, to categorize complex data flows in an application. At next they have to construct the control flow graph and for simulate the communication between all these registered event managers and modules the FlowDroid generates a dummy main function. Flow Droid examines every functions that can be gotten, and creates the control pour between these functions. Here in situation, we treat the fake leading task as the private entry of a Mobile app, which significantly make things easier the analysis procedure. FlowDroid treats every sensitive sources as the beginning point of the stain exploration procedure based on control flow graph. Flow Droid associate a forward-taint exploration and an on-demand backward-alias exploration. At last, FlowDroid can give entirely categorized movements from delicate causes to sinks in the app.

Deep Flow Android Malware Detection method is constructed based on the Deep Belief Network architecture as shown in Fig. 4. At first, user have to submit the apk file from the application, then DeepFlow starts FlowDroid to perform the every complex data pours from all the complex sources to the every complex sinks. By using, SUSI technique the Deep Flow classifies the extracted flows to catch features and create a feature vector. In DBN deep learning model input the feature vector for classification of malware. The DBN model has dual crawler modules. The first one is used for crawling malware from identified malwares and from Google Play Store we crawling the benign application by using other crawler. Deep Flow can declare its precision in detecting the frequently developing different malware. In addition, Deep Flow gives the complete complex data flow information and authentic classification of the application as the detection result to the user.

E. *Droid Deep [5]*

Droid Deep is a deep learning based method for Android Malware Detection. In this approach they used the DBN based deep learning model for malware detection. Droid Deep requires a single demonstration of application that supports to define the classic indications of malware action. Droid Deep has extensive range of fixed analysis that mines feature set from different sources like API calls and Manifest.xml files. Droid Deep aim to distinguish apps into various types such as requested permission, used permission, sensitive API calls, action and app components.

In this method for feature extraction based on the static analysis that require the .apk file of the Android app. After unzipping the .apk file using apktool and Droid Deep essentially gives the attention on analyzing two files such as AndroidManifest.xml and classes.dex correspondingly. By using this two file, they can extract the above given features. Droid Deep used the different tools and the parser for the feature extraction. They can parse the AndroidManifest.xml file, AXMLPrinter2 and the TinyXml parser, then get the required permissions of the Android app. Furthermore, they parse the classes.dex file to the baksmali, to see the different types of sensitive API which are used in Android app.

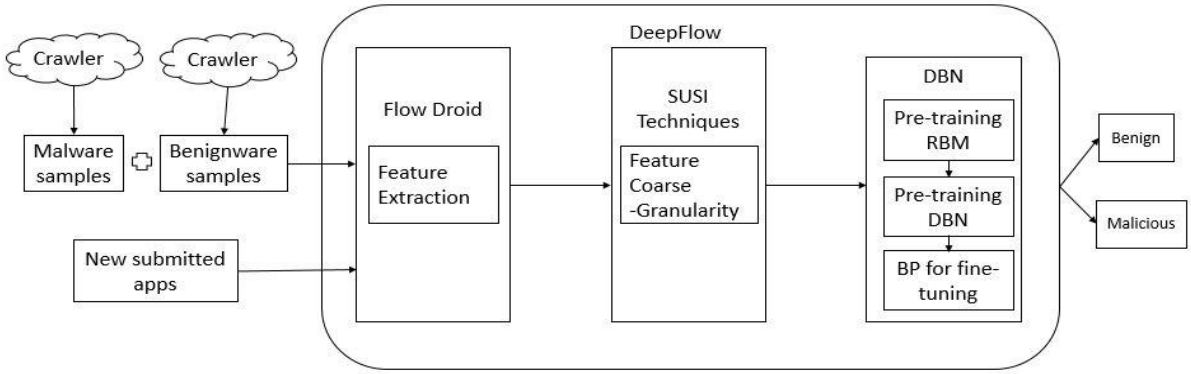

Fig. 4 Architecture of DeepFlow [4]

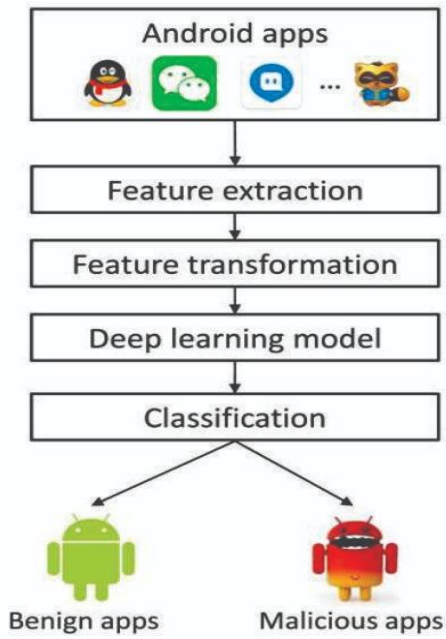

Fig. 5 Architecture of Droid Deep [5]

Droid Deep is a deep learning model based unique Android malware detection system. Droid Deep architecture is as shown in Fig. 5. The Droid Deep Architecture consists four different modules. Feature Extractor is answerable for the features which are extracted from .apk file and converting the extracted features keen on multi-dimensional vector. In deep learning model they gives the multi-dimensional vector as input and get exclusive features for the Android malware detection. At the end, the classification task based on the learning features.

F. Droid Delver [6]

Droid Delver is a deep learning based Android malware detection approach. DroidDelver uses the deep learning based DBN model for Android malware detection. Based on the extracted features the deep learning model trained for the malware detection and for the feature extraction they require the apk file from each application. By unzipping the .apk file they get the .dex file, properties, assets and manifest file. Android apps are established in Java and the development environments transform the java codes to the Dalvik executable files. The dex file format has compiled code which is written for the app. Droid Delver used the assembler or disassembler Smali that transforms the non-readable dex file into the readable smali code.

Droid Delver used the API method calls, which are used for access the system resources and the functionality in Android app. So that to extract the API calls, first they have to unzip the apk file to get access to the class.dex file, then they used the APKTool [10] for the extraction of API calls. In this method, they used the Max-Relevance algorithm to select API calls sets. After the API call extraction, they classify the API calls which appropriate to the equivalent process in the smali code into the API call block.

Droid Delver will extract the API call blocks from the Android app and use it as a features and apply a deep

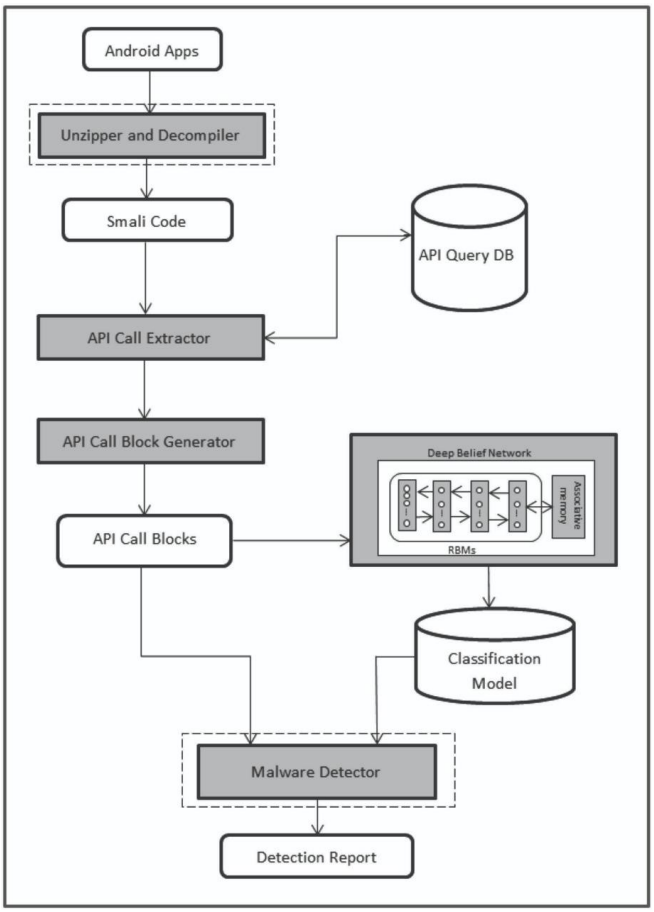

Fig. 6 System Architecture of Droid Delver [6]

Learning model for the unidentified Android malware detection. The system architecture of Droid Delver Android malware detection system is as shown in Fig. 6. Droid delver has main five components: Unzipper and Decompiler, API Call Extractor, API Call Block Generator, Deep Belief Network Classifier and Malware Detector.

The APKTool is used for extract the APKs and decompile the dex files to smali code. The API call extractor is used to extract the API calls from the smali code. The API call block generator used for the extracted APIs that will be appropriate to the several technique in smali code will be more classified into a block. Every Android app will be characterized by the API call block. The deep learning model will be used for unknown malware detection. The classification model is used as a marker whichever benign or malicious.

III. PARAMETRIC EVALUATION

In this survey paper that is shown that following parameters by which we can compare different deep learning based Android Malware Detection Techniques. The comparison table contains following parameters as Type of Deep Learning Model, Analysis, Features, Type of Applications in Datasets and Number of Applications in Datasets, etc.

- **Deep Learning Model:-** There are Different Types of Deep Learning Model that we use for the deep learning based Android Malware Detection.

TABLE I COMPARATIVE ANALYSIS OF EXISTING TECHNIQUES

Technique	Parameter				
	Deep Learning Model	**Analysis**	**Features**	**Type of Application in Dataset**	**No. of Application in Dataset**
MalDozer [1]	CNN	Static	API Method Calls	Benign & Malicious Application	38K, 33K
Droid Detector [2]	DBN	Static & Dynamic	Permission, APIs, Dynamic Behavior	Benign & Malicious Application	20000, 1760
Droid Deep Learner [3]	DBN	Static	Permission, API Calls	Benign Application	-
Deep Flow [4]	DBN	Static	API Method Calls	Benign & Malicious Application	3000, 8000
Droid Deep [5]	DBN	Static	Permission, API Calls, Action, App Component	Benign & Malicious Application	3986
Droid Delver [6]	DBN	Static	API Calls	Benign & Malicious Application	2500

- **Analysis:-** We have used two different types of analysis techniques either Static or Dynamic.
- **Features:-** There are different Features that we use in the deep learning based Android Malware Detection System such as API method calls, Permission, Dynamic Behavior, Actions, App Component Features.
- **Type of Application:-** We have uses the different types of applications in Dataset such as Benign Application, Malware Application
- **Number of Applications:-** It is shows that how many number of applications like Benign and Malicious that we used in dataset for testing and training the deep learning model.

IV. CONCLUSION

In this paper we have discussed about different types of Android Malware Detection Techniques using various Deep Learning Methods. Because of open nature on Android, countless malwares are hidden in a large number of benign apps in Android markets. These malwares are seriously threat Android security. The attacker can monitor user's information like: Messages, Contacts, Bank mTANs, Locations, etc. Here we survey on different Android Malware Detection Techniques like: MalDozer, Droid Detector, Droid Deep Learner and Deep Flow. MalDozer is used the Convolution Neural Network for Malware Detection. It works on static analysis method and API method calls as a feature to detect the application is malware infected or not. Droid Detector will use the Deep Belief Network for the detection. They used the static and dynamic analysis with features like: permissions, APIs, Dynamic behavior for malware detection. Droid Deep Learner method is also use the Deep Belief Network for malware detection. They also use a static analysis method with the features like permissions and APIs for malware detection. Deep Flow also use the Deep Belief Network with the static analysis method. In this method they use the API method calls for Android Malware Detection. But, these all methods are working after installing the application on device or upload it to their model. To overcome this problem we are trying to implement a Deep Learning model that can automatically identify the application is malicious or not before the installation.

REFERENCES

[1] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, 2018.

[2] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[3] Z. Wang, J. Cai, S. Cheng, and W. Li, "DroidDeepLearner: Identifying Android malware using deep learning," *2016 IEEE 37th Sarnoff Symposium*, 2016.

[4] D. Zhu, H. Jin, Y. Yang, D. Wu, and W. Chen, "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data," *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017.

[5] X. Su, D. Zhang, W. Li, and K. Zhao, "A Deep Learning Approach to Android Malware Feature Learning and Detection," *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016.

[6] S. Hou, A. Saas, Y. Ye, and L. Chen, "DroidDelver: An Android Malware Detection System Using Deep Belief Network Based on API Call Blocks," *Web-Age Information Management Lecture Notes in Computer Science*, pp. 54–66, 2016.

[7] R. Zachariah, K. Akash, M. S. Yousef, and A. M. Chacko, "Android malware detection a survey," *2017 IEEE International Conference on Circuits and Systems (ICCS)*, 2017.

[8] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 998–1022, 2015.

[9] A. I. A'Fifah, A. Ritahani, and A. Ahmad, "Comparative Performance of Deep Learning and Machine Learning Algorithms on Imbalanced Handwritten Data," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, 2018.

[10] V. Rao, K. Hande, "A comparative study of static, dynamic and hybrid analysis techniques for android malware detection," *International Journal of Engineering Development and Research,* pp. 1433-1436, 2017.

[11] A. Kapratwar, F. D. Troia, and M. Stamp, "Static and Dynamic Analysis of Android Malware," *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, 2017.

[12] K. Sugunan, T. G. Kumar, and K. A. Dhanya, "Static and Dynamic Analysis for Android Malware Detection," *Advances in Intelligent Systems and Computing Advances in Big Data and Cloud Computing*, pp. 147–155, 2018.