Malware Analysis through the Visualization of Convolutional Neural Network

Project Team

Sarah Muhammad	P13-6042
Shabnam Rani	P14-6113
Ibtihaj Tahir	P13-6064

Session 2014-2018

Supervised by

Muhammad Amin

Co-Supervised by

Dr. Mohammad Nauman

Department of Computer Science

National University of Computer and Emerging Sciences
Peshawar, Pakistan

May, 2018

Student’s Declaration

We declare that this project titled “*Malware Analysis through the Visualization of Convolutional Neural Network*”, submitted as requirement for the award of degree of Bachelors in Computer Science, does not contain any material previously submitted for a degree in any university; and that to the best of our knowledge, it does not contain any materials previously published or written by another person except where due reference is made in the text.

We understand that the management of Department of Computer Science, National University of Computer and Emerging Sciences, has a zero tolerance policy towards plagiarism. Therefore, We, as authors of the above-mentioned thesis, solemnly declare that no portion of our thesis has been plagiarized and any material used in the thesis from other sources is properly referenced.

We further understand that if we are found guilty of any form of plagiarism in the thesis work even after graduation, the University reserves the right to revoke our BS degree.

Sarah Muhammad	Signature: ________________
Shabnam Rani	Signature: ________________
Ibtihaj Tahir	Signature: ________________

____

Verified by Plagiarism Cell Officer
Dated:

Certificate of Approval

The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *Malware Analysis through the Visualization of Convolutional Neural Network*, submitted by Sarah Muhammad (P13-6042), Shabnam Rani (P14-6113), and Ibtihaj Tahir (P13-6064), in its current form, and it is satisfying the dissertation requirements for the award of Bachelors Degree in Computer Science.

Supervisor

Muhammad Amin — Signature: ____________________

Co-Supervisor

Dr. Mohammad Nauman — Signature: ____________________

____

Shakir Ullah

FYP Coordinator
National University of Computer and Emerging Sciences, Peshawar

____

Dr. Omar Usman Khan

HoD of Department of Computer Science
National University of Computer and Emerging Sciences

Acknowledgements

....

Sarah Muhammad

Shabnam Rani

Ibtihaj Tahir

Abstract

Malwares have different categories. They are hard to detect. The market of the android is growing day by day. Both the users and developers are moving to the android community. As it is growing so rapidly, there is a risk of being online. Malwares can damage the smart phones or the data contained in that smart phones is vulnerable. Our work targets to extract the features of malware so they can easily be classified. We will analyze the activations of hidden layers of convolutional neural network (Deep Learning) to achieve this goal.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we are going to talk about Android users which are at risk because of a huge number of malwares. Smart phones are essential part of life on days. People are using android, ios and other devices. The usage of android is dominant to other operating system. The risk factor is also dominant due to ack of security. the ratio of malware development increasing enormously.

1.1 What is Android

Android is developed by google which is a mobile operating system based on Linux kernel and other open source software. It is designed for smart-phones and tablets, further more development in Android Tv, Android auto for cars and Android wear for wrist watches.

First commercial Android device lunched in September 2008. Current version 8.1 Oreo was released in December 2017.

1.2 Architecture of Android

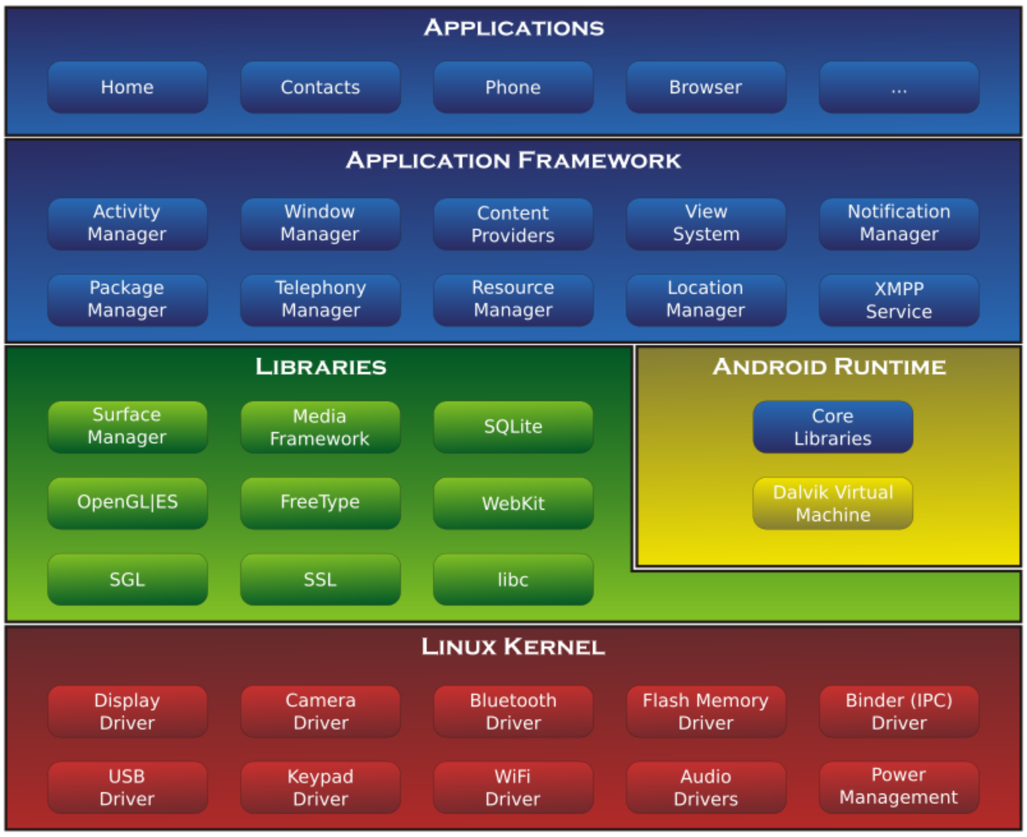


Figure 1.1: Android Architecture

Software Stack : A set of programs that work together to produce a result, typically operating system and application e.g: web browser.

Components: Android system consists of different software component arranged in stack which include the following components:

1. Linux kernel
2. Libraries
3. Android Run Time

4. Application Framework

5. Application

Linux kernel: Linux kernel is a bottom layer of android which is built on Linux 2.6 kernel. It provides basic functionality such as process management, memory management, device management and device drivers which make our work easier by interacting the android with peripheral devices.

Libraries: On the top of Linux kernel another layer is present called libraries. It provides different libraries useful for well-functioning of android os .These libraries are Java libraries build for Android os. Some of them are: SSL, openGL, SQlite, media framework, webskit.

Android Run Time: It is the third layer of android architecture and placed in second layer from bottom. It provides most important part of android called Dalvik Virtual Machine. It is similar to JVM but only difference is that it is designed and optimize for Android. DVM uses core function of Linux such as memory management and multithreading and enables each android app to run in its own process.

Application Framework: It is the second topmost component on android os stack. Android application directly interacts with application framework. It manages the basic functions of device such as resource management and voice call management. Some of them are content provider, activity manager, location manager,package manager.

Application: Application layer is generated by third party users or developer will install it on application layer.

1.3 .dex File

One of the common remarkable features of the Dalvik Virtual Machine is that it doesn't use Java bytecode. Instead DEX, a homegrown format was introduced .Not even the bytecode instructions are the same as Java bytecode instructions.

- Android application code file is compiled.
- Android programs are follow into .dex files, which are turn into single .apk file on devices. .dex files can be created by automatically translating compiled applications written in the Java programming language.
- We can also do some Reverse Engineering Techniques to make a jar file or java class file from a .dex file.

1.4 API

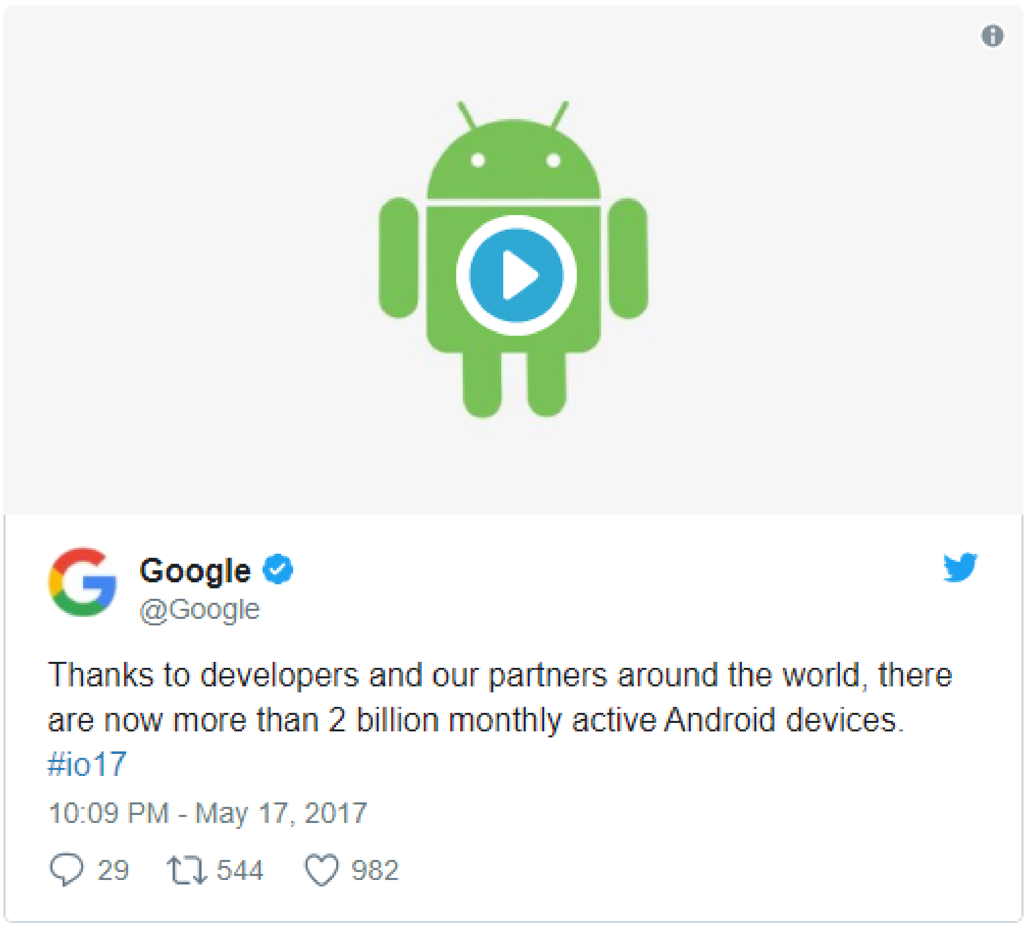
The Android API refers to the collection of various software modules which make up the complete Android SDK. In simpler words the Android API or Android SDK or just plain simple Android basically refers to the same thing. Since the software you write yourself interacts with the Android software to do various things, so the Android part is like an API. EXP:Maps Api to get location. Generally API is used to combine with the basic Android system.

API level:

- There are 5 levels of API.
- API level is basically the Android version.
- API is ready-made source code library.
- The levels of API make sure that all users get best experience either they are developers or simple users.
- These levels provide maximum framework levels and API revision that they require.

1.5 Impact

Number Of Active Android Devices : On May 17th 2017, Google's CEO Sundar Pichai announced that google has crossed a major milestone. It now has more than 2 billion monthly active devices. 7 services of google i.e. Google Maps, YouTube, Chrome, Gmail, Search, and Google Play has around 1 billion users. People downloaded around 82 billion apps in 2016.



Google
@Google

Thanks to developers and our partners around the world, there are now more than 2 billion monthly active Android devices. #io17

10:09 PM - May 17, 2017

29 544 982

Figure 1.2: Active android devices

Explanation: Android has proved to be an amazing smartphone operating system. The growth of android is very vast and fast because of multiple reasons. Development in android is easy. A lot of packages and tool are available for android developers. The community support is a lot better for example Forum XDA. So people, seeing a lot of facilities move towards android every day. If we talk about a layman's usage, Androids are very interactive hence providing easiness for user's who has a simple use of android.

Number Of Apps In Play Store : There are more than 3.5 million apps on play store till date. The development in android has grown a lot in past few years. The following graph shows us the new applications (uploaded to google play store) every year.

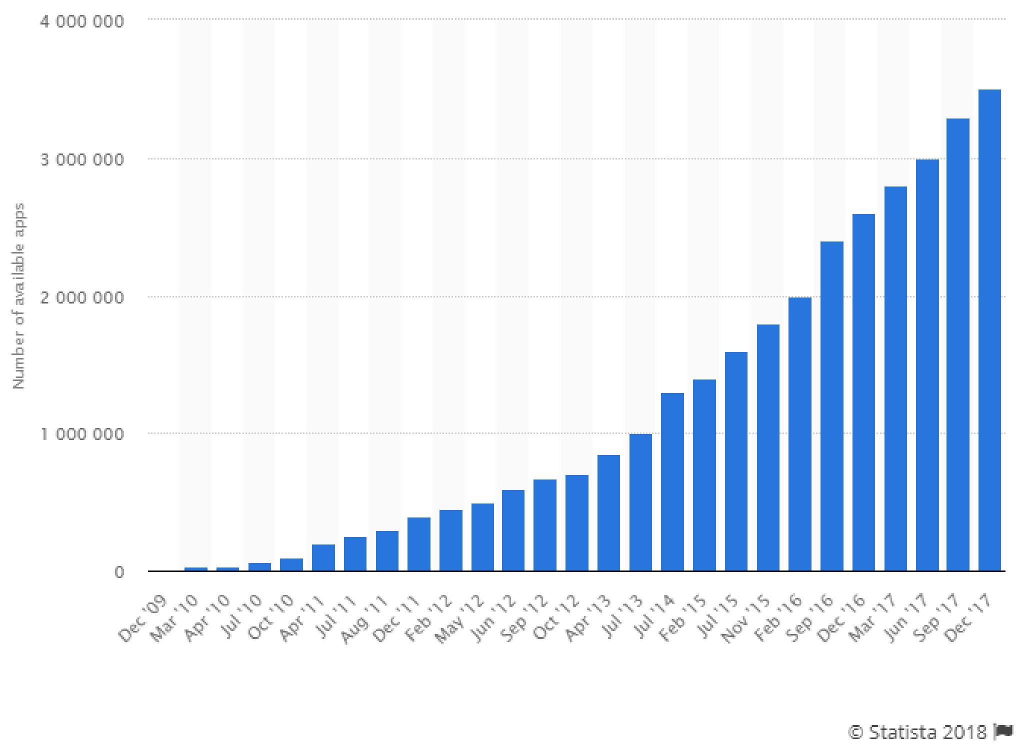


Figure 1.3: Apps in play store

Explanation: As the community is increasing day by day, development of apps is increasing with the same rate. People make new apps every day. App development has become so easy that a person without previous knowledge can easily make apps these days. We can find all sorts of apps these days. They are easily available and serve our purpose in no time. Hence the huge count of applications on play store till date.

Market Share Of Android : Android clearly sweeps away IOS in this war of technology. Android dominates the market by 87% whereas IOS is only at 12%.

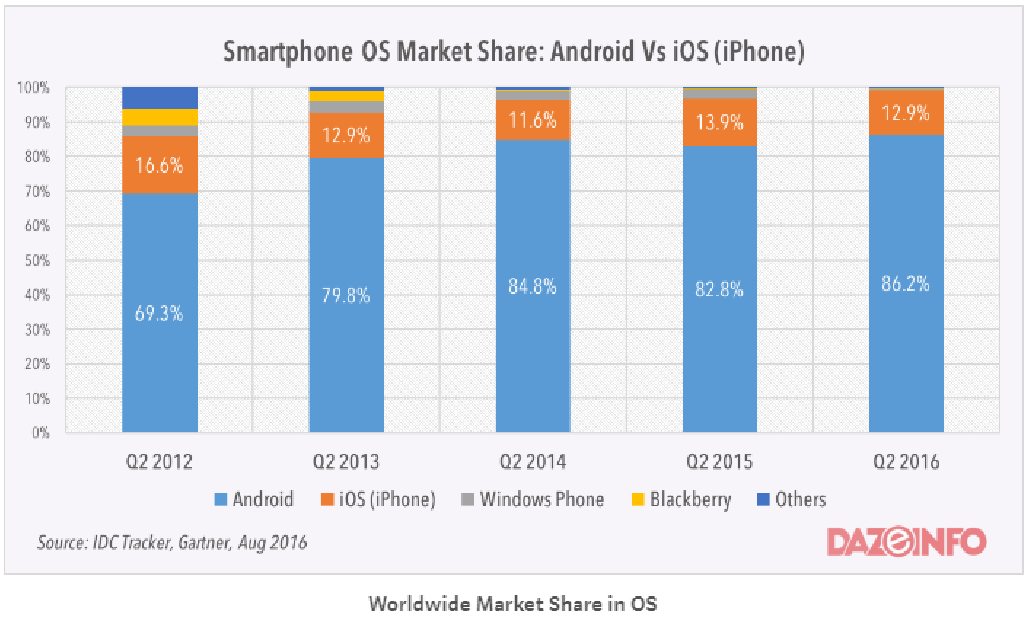


Figure 1.4: Growth of android market shares (2012 - 2016)

Geographic View:

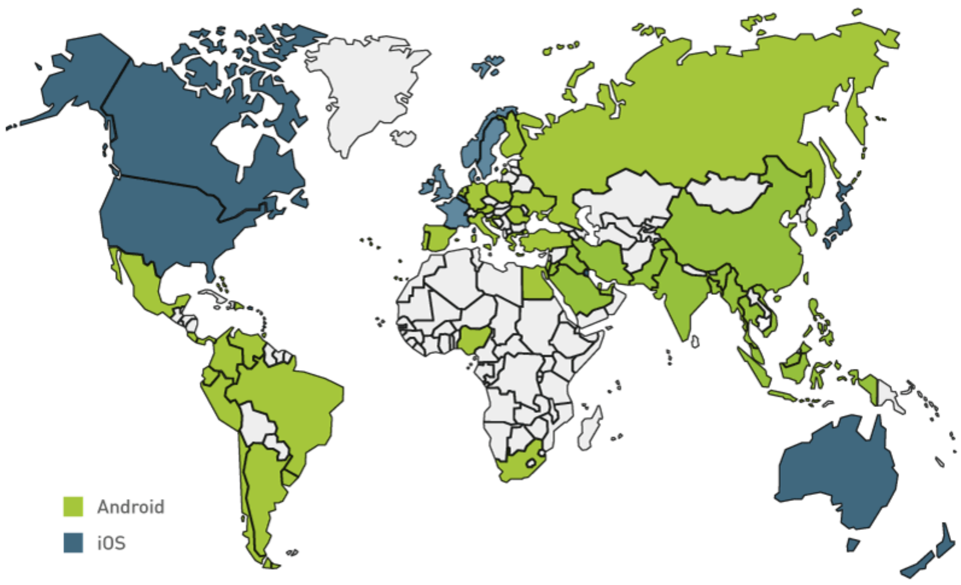


Figure 1.5: Geographic view

We can see that in continents like Asia and Africa, the number of android shares are at peak whereas continents like US, Australia, Europe have more IOS users. It maybe because of socio economic factors.

OS Variant Distribution : Further we can also see the android os variant users. People using android but version vise.

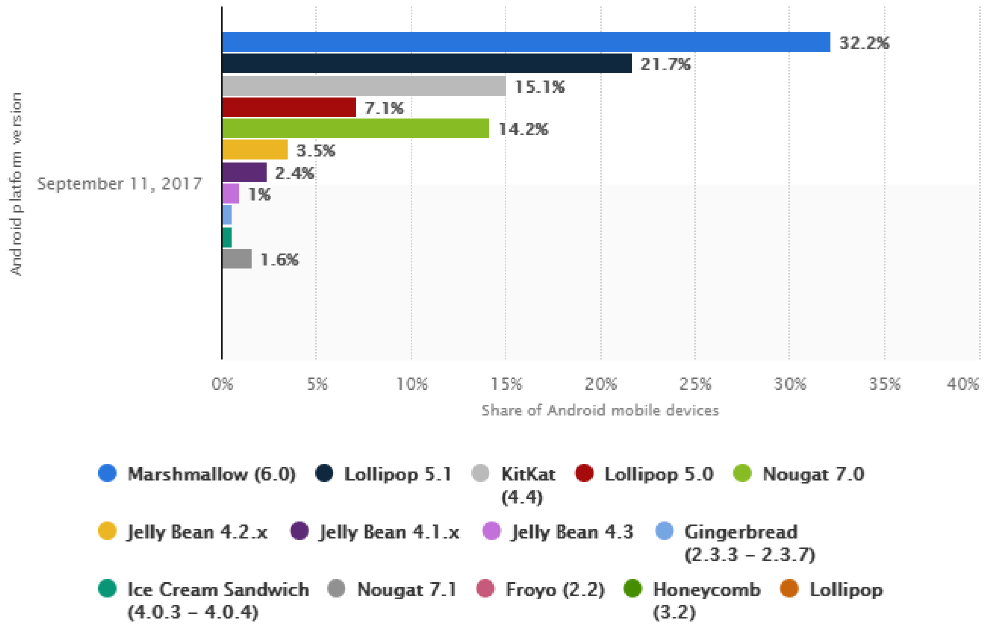


Figure 1.6: Variant distribution

Explanation: As we can see in the above graph that number of older android users still exist. Security patches for android 7 (Nougat) are not available for the older versions of android. So we can see that a lot of users are at risk.

Security And Privacy : Android is more open than that of iOS. Apple takes away the medal in this case when it comes to security and privacy. iOS is more secure than android. The third-party app installation and device fragmentation makes android more vulnerable.

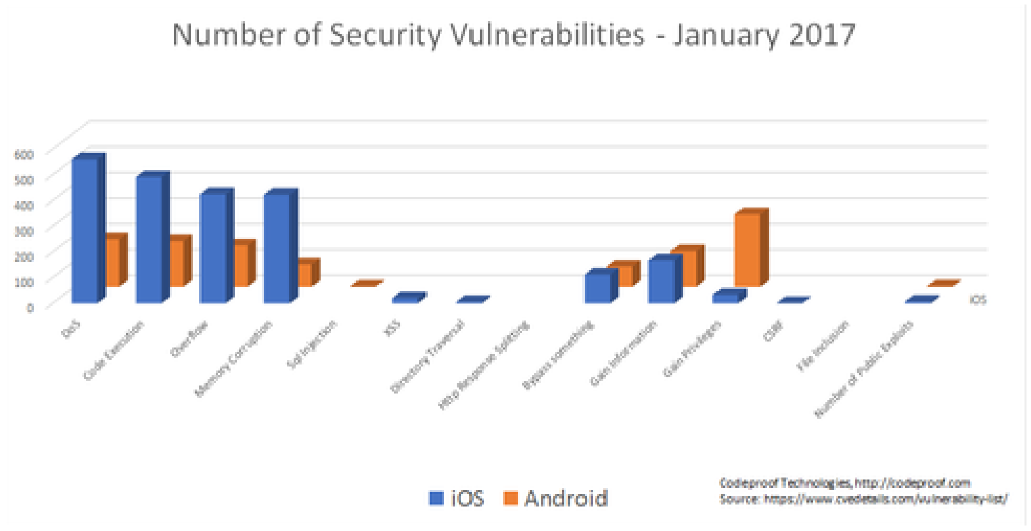


Figure 1.7: Security vulnerabilities

1.6 Malware Analysis

Malware is a study of malicious APK's that give impact on the system. The objective of malware analysis is to understand the working of malware, how it is detected and after that solve the problem. Some of them are virus, spyware, worm, Trojan horse, rootkit, adware and backdoor etc.

• Malware analysis is a multiple step process which grasp the malware formation and its purpose. Now a day's malwares are utilized to send spam messages, to perform web fakes, to take individual data like credit card data and many other things to steal the information.

1.6.1 Types of analysis

Malware could be detected on two different ways that is:

- Static Analysis
- Dynamic Analysis

Static Analysis: Static analysis is when malware is detected it before the execution of an application. The advantage of static analysis is that it can guarantee whether a file is malignant, provide information about its formation and the virus would never get a chance to be executed and would save your system from any harm.

Dynamic Analysis: Dynamic analysis is when malware is detected after the execution of an application, which may be harmful depending upon the nature of malware.

1.7 Machine Learning vs Deep Learning

Machine Learning: Machine learning is highly dependent upon features of data set the steps are to extract features from the dataset, train a model on those features and then try to classify.
Deep Learning: Deep Learning is not dependent upon features. They just take the data and try to extract features for their selves. They try to extract best features and predict on those features. Deep Learning requires high processing.
Bytecode: Bytecode is the object file of the corresponding apk file. DVM is developed by Google and optimized for the characteristics of android platforms. The bytecode in Dalvik is transferred from JVM bytecode into the dex by converting java files with a dx tool. We are using static analysis for Android (Dalvik) bytecode. Programs of android are written in java and then compiled to bytecode JVM, after that converted into Dalvik bytecode and stored in .dex which is also called Dalvik executable.

Static analysis of android bytecode with deep learning: Android programs are written in java and compiled to bytecode JVM later on translated to Dalvik bytecode and stored in .dex which is also called Dalvik executable.
Potential solution for stabilizing the malware detection effectiveness is Deep learning.

1.8 Features

Feature engineering: Feature engineering is the process of using expertise of the data to create features that make machine learning algorithms work. There are some steps that are being followed in feature engineering .The very first step test features includes understanding of features by building some test scenarios or by preparing positive and negative datasets. The decision of features we need follows the next step. Creating features is the second step in feature engineering in which features are created so that working on those features gets in line and we get the desired results by continuously improving those features if needed to be improved.

Automated feature engineering: Automated feature engineering is extraction of features automatically. Automated feature engineering helps Data scientists reduce data exploration. Automated feature engineering provides results quickly and extracts value from the data with a little effort time and cost.

Handmade features: When a subset of feature space is selected for a specific purpose, this is known as feature extraction and those chosen features are known as hand crafted features. This gives us the control of what is important for us given a certain problem.

Chapter 2

Literature Review

In this chapter background of related work is being discussed.There are several different methods and techniques for malware detection. From the research we come to know a rich information about the malware behaviors and it's analysis. In each of the research paper APK's are manipulated in the way that they are first acquired from play store or other sources or authenticated antiviruses scanned records stored and provided by the companies and are publically accessible and then those APK's are converted to binary form after extraction of the .dex files and then their further research and accuracies,future works are being discussed .

In r2d2 research paper they have executed the tasks in a way that we have followed our Approach, first extracted the dex code from APK's and then converted them into binaries and have visualized them. The dataset they have worked with is growing day by day which is resulting in the low accuracy of the corresponding research paper but the approach proved to be the most powerful of all the approaches discovered in the past. That is why we have followed this research paper and we have executed the tasks of visualizing keeping in view the steps of this research paper and we have reached to steps of visualizing the behavior of our dataset.

From literature we also came to know that we have families of different malwares. By family we mean all the malwares which are executing the same behavior making it easy for us to avoid studying behavior of each and every malware individually by applying a dominant keyword algorithm to assign a family name and thus studying ,understanding and visualizing the behavior of each and every family and building better intuitions about the behavior of malware ..[2]

2.1 Research Papers Summary

Name	**Techniques**	**Dataset**	**Features**	**Acc.**
1. A New Method to Visualize Deep Neural Networks	Conditional Sampling, Multivariate Analysis, Deep Visualization of Hidden Layers, Pre-Softmax versus Output Layer,Alexnet	Images from the ILSVRC,(a large dataset of natural images from 1000 categories)	Features Maps	-
2. A New Method to Visualize Deep Neural Networks	Conditional Sampling, Multivariate Analysis, Deep Visualization of Hidden Layers, Pre-Softmax versus Output Layer,Alexnet	Images from the ILSVRC,(a large dataset of natural images from 1000 categories)	Features Maps	-
3. Gamut: Sifting through Images to Detect Android Malware	Gamut. linear plotting,KNN	McAfee	List of API calls are used as features, 13,805 APKs labeled as malicious and 22,378 APKs labeled as benign	92%.

Table 2.1: Research Papers Summary

4. Malware Images: Visualization and Automatic Classification	k-nearest neighbors with Euclide andistance, Hypothesis Validation, Large Scale Experiments	10,072 malware samples labeled by an anti-virus software and divide the dataset into 14 malware families	GIST feature	99.2%
5. Malware classification using self organising feature maps and machine activity data	Self Organising Feature Maps, Data and architecture, Decision trees,SVM,ANN, Random Forest, BayesNet, MLP, Visualizing activity using	SOFM Kasperky, McCafee	Fuzzy neighbourhoods	-
6. Deep Learning for Network Flow Analysis and Malware Classification	Protocol Classification Using Metadata, Payload Data Collection and Data Preprocessing for Network,Application Classification, Malware Classification Using Kaggle Data, Different Convolutional Neural Networks	Microsoft's Kaggle data Which contain 9 families	Features are abstract in nature	83.8%

Table 2.2: Research Papers Summary

7. Imbalanced Malware Images Classification: a CNN based Approach	Proposed a weighted softmax loss to address the data imbalance	25 classes, 2000 images,Batch size =80	Feature maps of different classes	-
8. Understanding Neural Networks Through Deep Visualization	Regularized Optimization, Visualizing Live Convnet Activations,dataset-centric approach	Very large neural networks	Patterns of features	-
9. Empowering Convolutional Networks for Malware Classification and Analysis	Rd,convolutional, hybrid neural networks thSVM	Portable Executable (PE) files,opcode sequences	PEInfo features	-
10. Detection and Visualization of Android Malware Behavior	Visual Analysis of the Traces	Genome dataset	Goodware, Adware and Malware	-

Table 2.3: Research Papers Summary

Chapter 3

System Analysis and Design

3.1 Use Case

Following are three use case for each possible scenario when user will use the android.

Use case	Android
Primary actor	apk
Pre condition	dex
Post condition	bytecode
Success scenario	generates images

Table 3.1: Use Case

3.2 Use Case Diagram

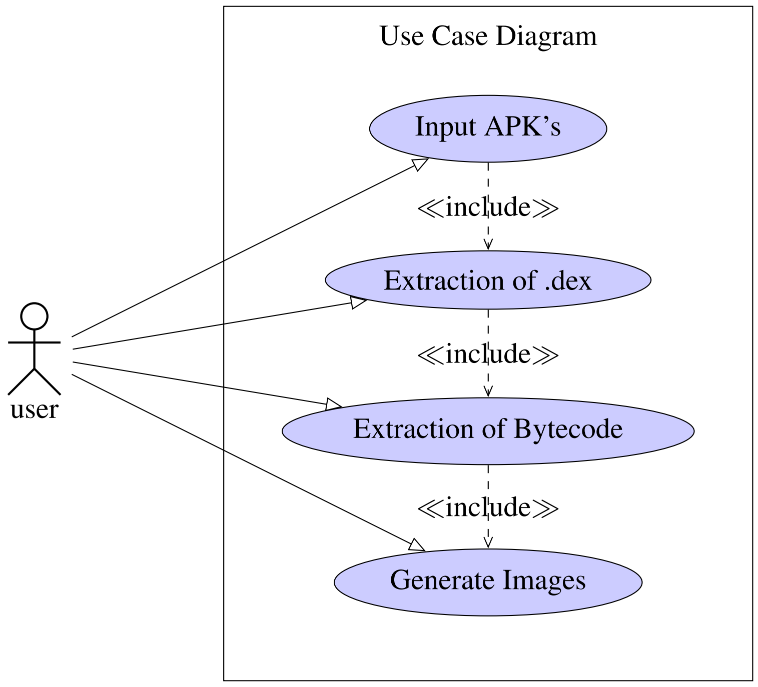


Figure 3.1: Use Case Diagram

3.3 Block Diagram

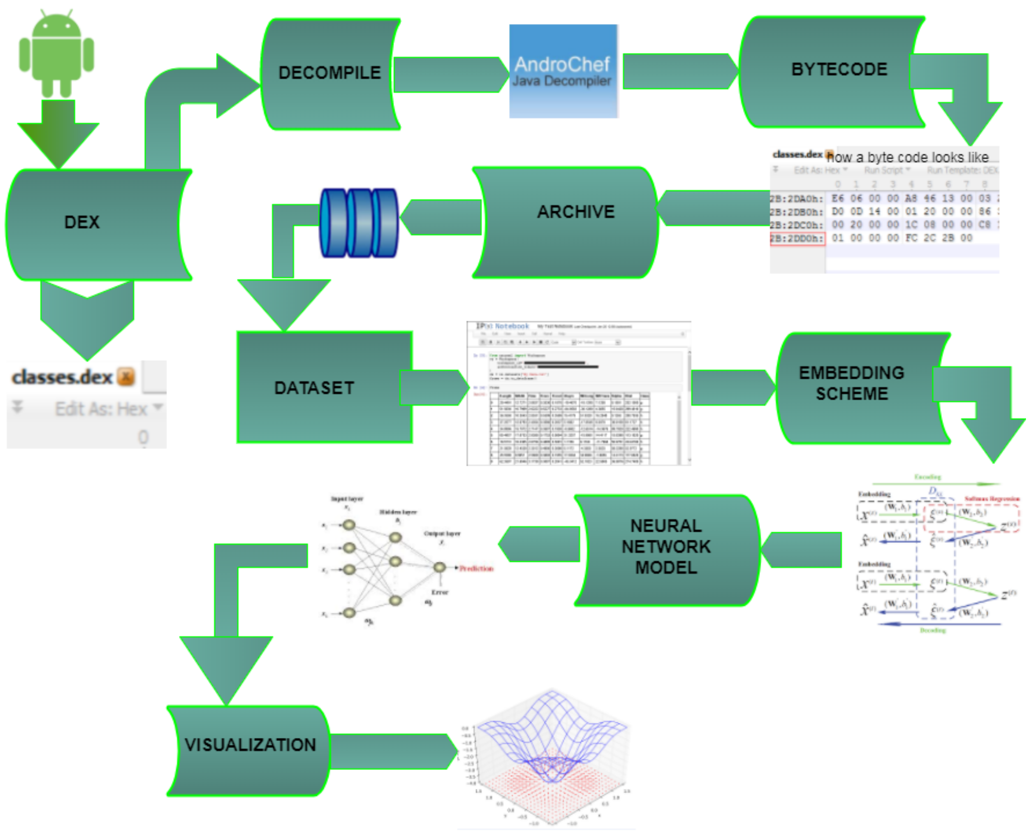


Figure 3.2: Block Diagram

3.4 Activity Diagram

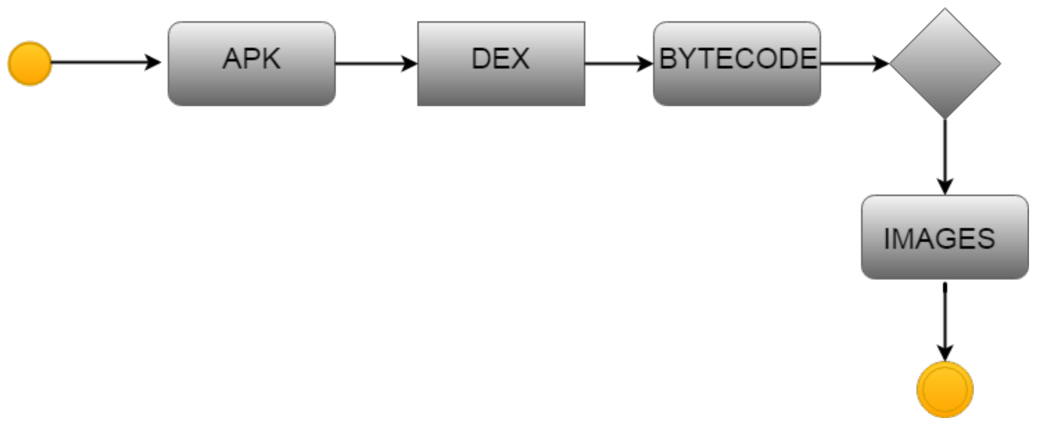


Figure 3.3: Activity Diagram

3.5 System Sequence Diagram

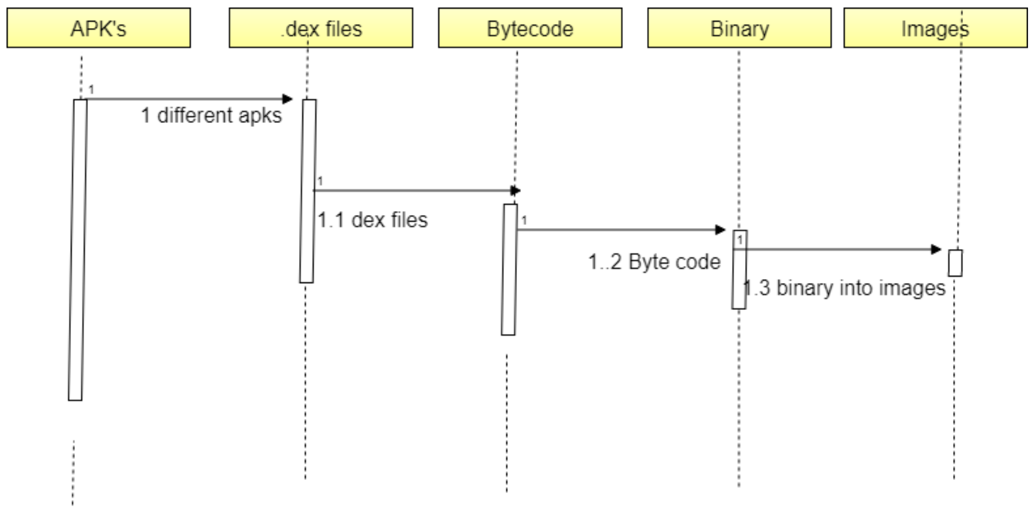


Figure 3.4: System Sequence Diagram

Chapter 4

System Implementation

4.1 Methodology

The methodology comprises of a pre-processing step, followed by deep learning and then some post-processing steps which are explained as follows:

4.1.1 Dataset Acquisition

"Android apks are included in the dataset. The dataset was downloaded from 2 sources. Benign files are acquired from play store. Malicious files are downloaded from Andro-Zoo". [1]

4.1.2 Dex files extraction from Apks

Each APk contains .dex file having bytecode .The goal is to extract the bytecode of every apk file as bytecode is a machine level instruction set and malware is nothing but a manipulation in that instruction set.

4.1.3 Bytecode conversion to decimal for Image generation.

The extracted bytecode is translated to, first binary, which is a 16bit binary code. Then this binary code is later transferred to decimal for image generation purpose. Every instance of the bytecode is a value that lies between (0 - 255).

4.1.4 Bytecode Extraction Script

Each task said above is executed by a script in python which takes a directory of APK (separately for benign and malicious) as an input and convert it into first hexadecimal and then binary in short given an APK it will convert it into binary form in extension .txt which contains 16 bit binary in range from 0-255(justification for this range is that we plan to convert those binaries to images for visualization purpose). [4]

4.1.5 Bytecode Images of Malicious and Benign

We have a one dimensional list of binary instructions. This list is converted to decimal and then reshaped to 2 dimensions.[3]

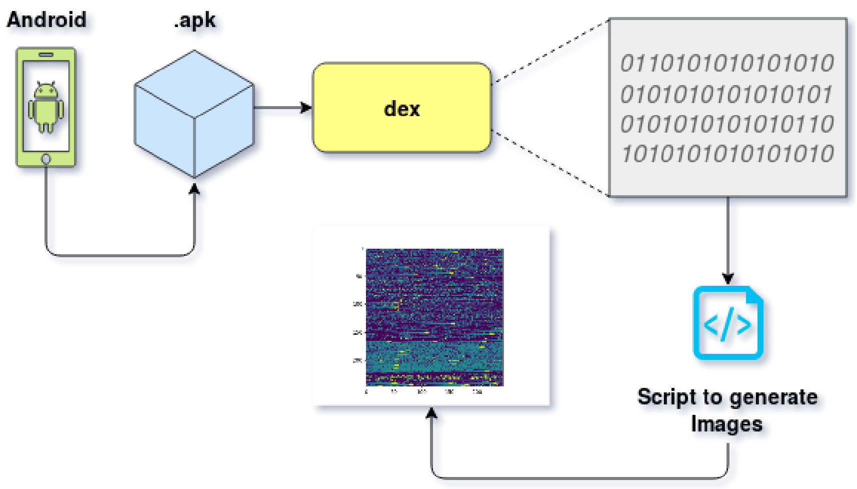


Figure 4.1: Methodology

4.1.6 Training the model

After generating images of malicious and benign bytecode, the goal is to feed it to the convolutional neural network and train it on those images in a binary classification method where '0' represents the benign bytecode and '1' represents the malicious bytecode.

4.1.7 Results

The results from our dataset, we generates images.

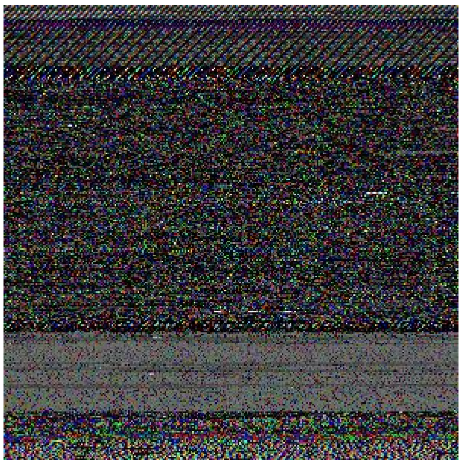


Figure 4.2: 2 dimension image

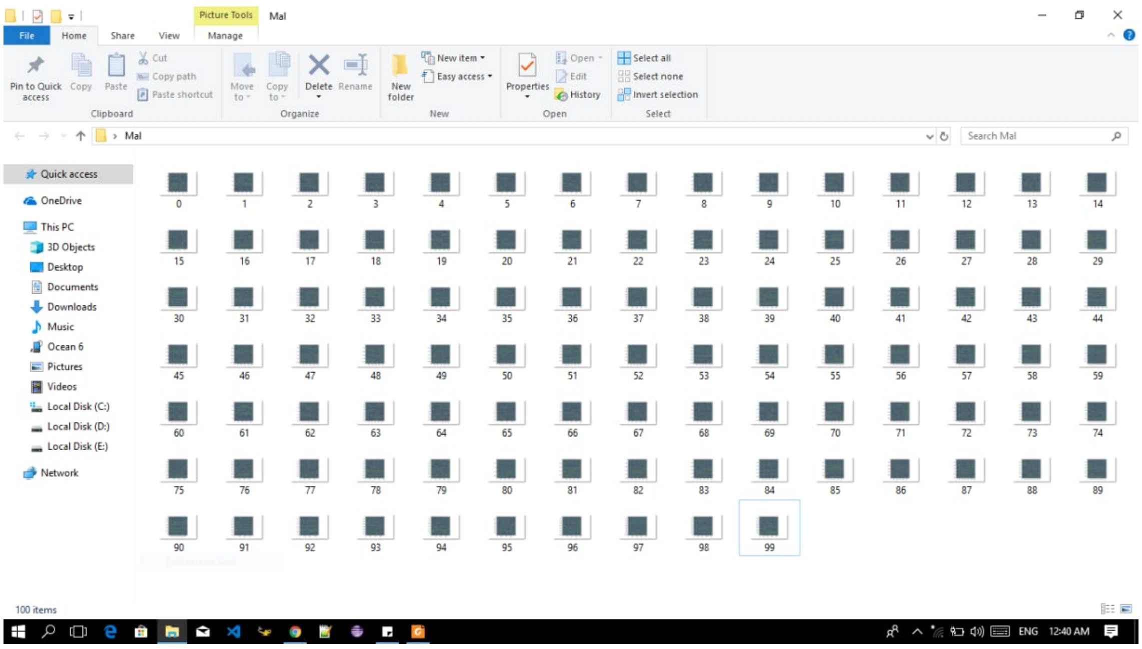


Figure 4.3: Malicious images

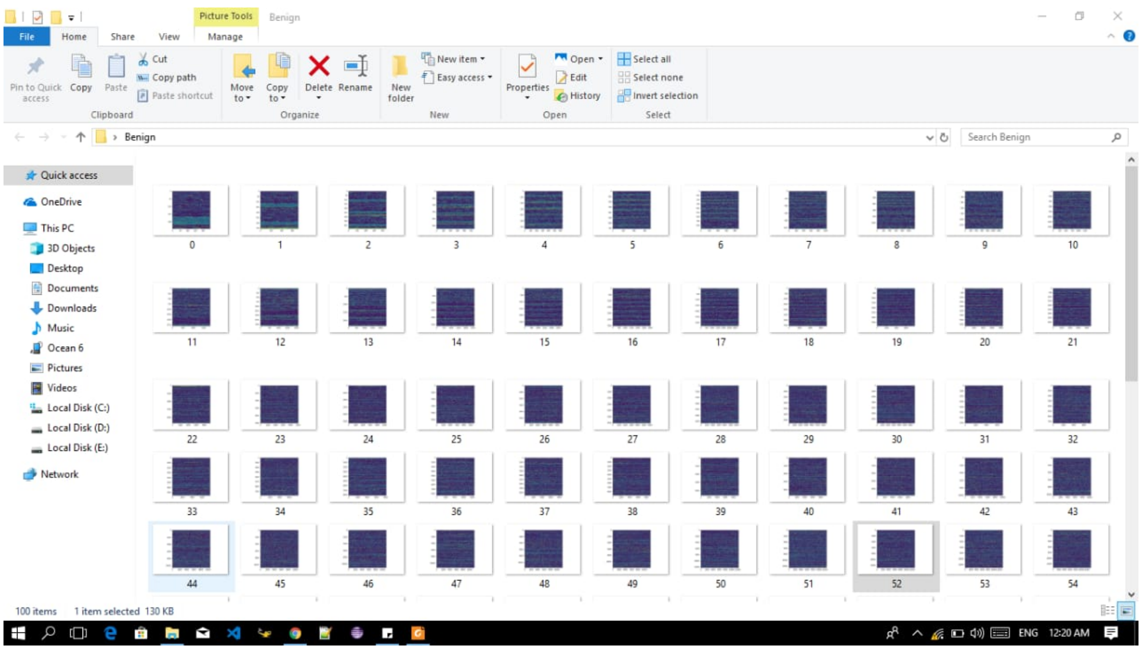


Figure 4.4: Benign images

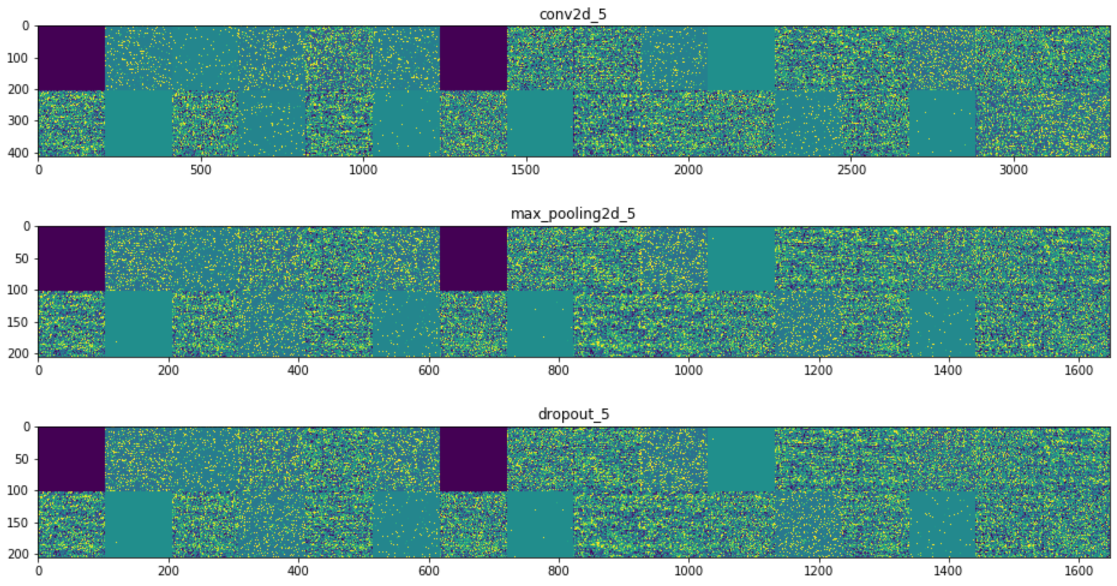


Figure 4.5: Visualization

Chapter 5

Conclusions and Future Work

5.1 Discussion

We will generate images of the byte code of android's apk and then train a neural network on it to check the accuracy. We will train CNN models only. Once we have the model giving highest accuracy, we will visualize the hidden layers of that trained neural network. We then, will analyze those activations and will try to find the behaviors and features.

5.2 Further Work

If we find any Feautres, we will use those patterns to train a neural network and then compare the accuracy of it with other work.

Bibliography

[1] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. *android*, 2016.

[2] TonTon Hsien-De Huang, Chia-Mu Yu, and Hung-Yu Kao. R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections. *android*, 2017.

[3] Fengguo Wei, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. Deep ground truth analysis of current android malware. *android*, 2017.

[4] Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution. *android*, 2012.