
# PARUDroid: Validation of Android Malware Detection Dataset

[1]Arvind Mahindru, [2]A.L. Sangal
[1],[2]Department of Computer Science & Engineering, Dr. B.R. Ambedkar National Institute of Technology Jalandhar 144001, India [1]Department of Computer Science & Applications, D.A.V. University, Sarmastpur, Jalandhar 144001, India
er.arvindmahindru@gmail.com[1]

## Abstract

Android has gained its popularity due to its open nature and number of free apps in its play store. Till date, Android has captured 87% of the total market share. 2.8 million apps are present in the official market of Android. Android apps depend upon permissions for its proper functioning. This dataset contains distinct 5,60,142 Android apps that belong to thirty different categories. These Android application packages (.apk) is collected from Google-play store and other promised repositories. In this study, we performed a dynamic analysis of these collected .apk packages and extracted features, i.e., PARU (Permissions, API calls, Rating of an app, and Users download the app). As per the knowledge, this is the first dataset that extracted features by using the Android 6.0 (API 23) version as an Android operating system. The paper discusses the potential usefulness of the dataset for future research in the field of cybersecurity. Further, to check the potential of our dataset, in this research paper malware detection model is developed by using five different classification machine-learning algorithms. Experiment result reveals that model developed using Deep Neural Network (DNN) can able to detect 98.8% malware-infected apps. Dataset URL: http://dx.doi.org/10.17632/mg5c8jxbhm.2
**Keywords:** Android apps, Permissions model, API calls, Intrusion detection, Cyber security, Smartphone

## 1 Introduction

Android has been released in the market in the year 2008 and gains popularity in the year 2010. Till now, Android has introduced eighteen versions [1] in the market. According to the study [2], Android version 6.0 has the highest market share, i.e., 16.9%. till the end of 2019. To make the Android operating system more secure, Google has launched an app permissions facility in this version. In the earlier versions of the Android operating system, the system itself grants the permissions to the apps. In Android 6.0 (i.e., API 23), Google has introduced the facility for users to grant or revoke the permissions during the installation of an app.

Android apps depend upon the permission model for its proper functioning. Permissions are divided in to three different classes i.e., "Normal", "Signature" and "Dangerous". Normal permissions when an app access resources or data which lie outside the sandbox and pay minimal risk to the user's privacy. Signature permissions are those which are granted by the system itself when an app is installed. Dangerous permissions are those which access resources or data that involve the user's private information. Cybercriminals take advantage of these permissions and develop malware-infected apps daily. According to the report published by Kaspersky [3], there are 68,362 new mobile ransomware Trojans, 69,777 new mobile banking Trojans and 3,503,952 malicious installation packages are still present in the Android devices.

There are traditional datasets available for desktop malware detection [1], but there is an inadequate dataset related to Android permissions. Collecting and analyzing these Android apps at a large scale is still a challenging job because Android app repositories implement several restrictions for this. In this regard, researchers and academicians are stick to use small datasets or collect a limited number of apps, and their experiment results are not often reproducible.

[1]https://en.wikipedia.org/wiki/Android_version_history
[2]https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/
[3]https://securelist.com/mobile-malware-evolution-2019/96280/

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

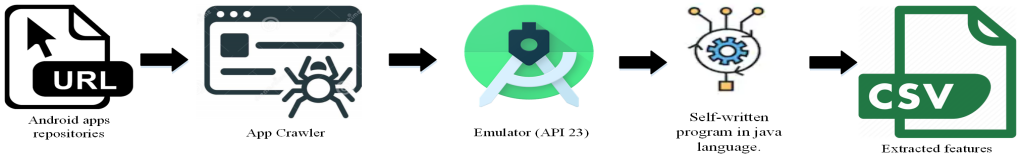


**Figure 1** Flow chart of the proposed work

To address the problem of Android dataset collection and feature extraction (i.e., permissions, API calls, number of users download the app, and rating of an app), we have invested a long-time effort to crawl apps from different promise repositories and extract permissions and API calls from them. After crawling Android apps from various repositories, we collect 5,60,142 unique Android apps. With PARUDroid, we aim to provide academicians and researchers with a scalable, up-to-date, and unrestricted access to the features dataset extracted from Android. In this study, we developed specialized crawlers for distinct Android market that automatically download Android apps that are free and store into our repository. To the best of our knowledge, this is the first dataset which extracted a large number of features from the collected Android apps and published for Android security purpose. The unique and novel contributions of this paper are as follows:-

- To the best of our knowledge, this is the first research dataset that has 5,60,142 unique Android apps that belongs to thirty distinct categories. This dataset is collected from June 2014 to April 2020.
- Dynamic analysis is performed on Android 6.0, and 1419 unique permissions and 310 API calls are extracted.
- This is the first dataset, that considers the rating of an app and the number of users download an app.
- Validation of dataset is done by using five distinct machine learning algorithms.

Figure 1. demonstrates the phases which are followed in extracting features from Android apps. In the very first phase, we identify the URLs from which Android apps are to be collected. In the second phase, we take the help of an app crawler to download the apps from identified URLs. Our developed app crawler can download as many apps as possible and do not pay any impact on the android app repository. To perform dynamic analysis of the collected Android apps, we use Android studio as an emulator [2]. To extract permissions and API calls, we use Android 6.0, i.e., Marshmallow. There are two reasons to select Android 6.0; first is that Android 6.0 has captured the highest market share, and the second main reason is that in this version, Android asks users to grant or revoke the permissions which Android apps demand at the time of its installation. This facility was not present in the earlier version of Android. Further, we write a program in java language and extract permissions and API calls from them and save into the .csv file for researchers and academicians.

The rest of the paper is summarized as follows. In section 2, we discuss the related work. Discussion of various repositories from where Android apps are collected and working on the app crawler is presented in Section 3. Section 4 discusses the categories of the Android app to which they belong. In section 5, we discuss the extraction process from Android apps. Machine learning techniques used in this study is presented in section 6. Parameters used for evaluation are discussed in section 7. In section 8, we present the experimental setup for validation of dataset. Result of performed experiment presented in section 9. Leveraging of the dataset is presented in section 10. Section 11 discusses the limitations, and section 12 presents the conclusion.

## 2 Related Work

Zhou and Jiang [7] present Android Malware Genome was the first research project that has provided the community an Android malware dataset. This dataset has been the only well-labeled one and has been widely studied and used by the research community. Android Malware Genome consist more than 1200 malware-infected apps from 2010 to 2011. Malware-infected apps belong to 49 different malware families. Result reveals that 93.0% of Android apps exhibit the bot-like capability, 86.0% of them having malicious payloads and 36.7% apps exploit platform-level escalate privilege. Wei et al. [8] proposed dataset that contain 24,650 malware-infected apps that is further divided into 135 varieties based on malware behavioral semantics which belong to 71 malware families.

Allix et al. [9] proposed AndroZoo dataset which contains more than 3 million Android apps from Google Play, other smaller markets, and app repositories. Goal of the Androzoo dataset to collect a comprehensive

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

app for software engineering studies. Contagio Minidump [4] and VirusShare [5] are two other repositories for dataset but both were having limitation they do not provide a comprehensive label or comprehensive malware collection and behavior information on the malware. ANDRUBIS [10] combines static and dynamic analysis to automatically extract feature and behaviors from Android apps, and studies the changes in the malware threat landscape and trends among "goodware," or benign apps, developers.

## 2.1 Limitations in existing datasets

Our dataset consist the dynamic behavior of Android apps that are developed during 2013 to 2020, behavior of Android apps is not the present in existing dataset. Existing dataset is not properly labelled, but our dataset is properly labelled with benign and malicious apps. Further our dataset contains all the permissions, that an app demand from installation to run-time. Existing dataset contains limited number of benign and malware-infected Android apps.

# 3 App source and Crawler

In this section of the paper, we discuss the details of the repositories from where Android apps are collected and also discuss the working of the app crawler.

## 3.1 App source

### 3.1.1 Google play

Google play [6] is an official market of Android from where users download paid and free apps for their personal use. 2.8 million apps are present in the official market of Android, which belongs to distinct categories of Android. No app can be download from Google play store without a valid Google account. Google also applies restrictions in downloading apps up to a specific limit in a given time frame.

### 3.1.2 AppChina

AppChina [7] is another app repository from which the user can download and install the app. Downloading an app from this repository is also a big challenge. It put several hours ban if the user downloads simultaneously Android apps from this.

### 3.1.3 Hiapk

Hiapk [8] is a Chinese app repository from which users can download and install the app on their smartphone.

### 3.1.4 Android

Android [9] is a Chinese app repository from which users can download and install the app on their smartphone.

### 3.1.5 Mumayi

Mumayi [10] is a Chinese app repository from which users can download and install the app on their smartphone.

### 3.1.6 Slideme

Slideme [11] is an Android app repository from which users can download and install the app on their smartphone.

[4] http://contagiominidump.blogspot.com/
[5] https://virusshare.com/
[6] https://play.google.com/store?hl=en
[7] http://m.appchina.com/
[8] http://apk.hiapk.com/
[9] https://android.d.cn/
[10] http://www.mumayi.com/
[11] http://slideme.org/

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

#### 3.1.7 Pandaapp

Pandaapp [12] is an Android app repository from which users can download and install the app on their smartphone.

#### 3.1.8 Botnet samples

Botnet samples are collected from [3]; it consists of a 1929 botnets sample that belongs to 14 different botnet families.

#### 3.1.9 Malgenomeproject

Malware samples are also collected from the Android Malware Genome project [4]; it consists of 1200 malware samples that cover the majority of existing Android malware families.

#### 3.1.10 AndroMalShare

Malware samples are also collected from AndroMalShare [13], which belong to distinct Android malware families.

### 3.2 App Crawler

To collect apps from sources mentioned above, we developed a web crawler by using scrapy[14] framework. Our developed framework works on four distinct steps, i.e., a) downloads the app b)it ensures that the app is not downloaded earlier c)compute its SHA256 checksum d) archives the app. To identify that app having the same name has not been download from two different repositories, we identified a unique ID, i.e., APP_Source_APP_Identifier and store in a Cough DB [15]. To overcome the limitation, for downloading the app from distinct app sources, we write a code which deals with download agent and a central dispatcher. We have used agents on up to five machines located in INDIA, so that risk of being blacklisted is minimized.

## 4 Android app categories

After collecting Android apps from different repositories, next, we scan each of the collected apps by using virus-total [16] as an anti-virus scanner to identify that app belongs to benign or malware class. Table 1 shows the distinct categories of Android apps and the class to which they belong.

Table 2 shows the top 82 malware families identified in this collected dataset w.r.t to number of samples. Malware families are identified by using virus-total as an anti-virus scanner.

## 5 Extraction of Android permissions and API calls

In this study, we used Android studio [17] as an emulator and run each one of the collected apps on it and extract permissions and API calls from them. To extract features from collected Android apps, we write a self-written algorithm (mentioned in algorithm 1) and used java language to integrate with an emulator. By this, we extract features and save into .csv file format, which is available publicly for the research community. Further, we also identify the top twenty permissions and API calls, which are demanded by the number of Android apps. Table 3 shows the top twenty permissions and API calls, which demanded by several apps during installation and run-time. List of extracted permissions and API calls are available at-promise repository [18].

[12] www.pandaapp.com
[13] http://sanddroid.xjtu.edu.cn:8080/
[14] https://scrapy.org/
[15] https://couchdb.apache.org/
[16] https://www.virustotal.com/gui/home
[17] https://developer.android.com/studio
[18] http://dx.doi.org/10.17632/mg5c8jxbhm.2

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

**Table 1** Collected Android apps.

| ID | Category | Normal | Trojan | Backdoor | Worms | Botnet | Spyware |
|---|---|---|---|---|---|---|---|
| DS1 | Arcade and Action (AA) | 16291 | 1440 | 104 | 200 | 230 | 500 |
| DS2 | Books and Reference (BR) | 15235 | 2000 | 256 | 50 | 140 | 160 |
| DS3 | Brain and Puzzle(BP) | 14928 | 1820 | 55 | 38 | 40 | 49 |
| DS4 | Business (BU) | 18308 | 1520 | 160 | 140 | 20 | 24 |
| DS5 | Cards and Casino(CC) | 12886 | 760 | 75 | 71 | 104 | 40 |
| DS6 | Casual(CA) | 12010 | 3210 | 79 | 36 | 160 | 130 |
| DS7 | Comics (CO) | 17667 | 650 | 90 | 31 | 7 | 5 |
| DS8 | Communication (COM) | 18414 | 2500 | 50 | 400 | 53 | 53 |
| DS9 | Education (ED) | 18744 | 5600 | 58 | 60 | 60 | 58 |
| DS10 | Entertainment(EN) | 14222 | 5000 | 500 | 500 | 102 | 40 |
| DS11 | Finance (FI) | 13999 | 500 | 200 | 90 | 65 | 101 |
| DS12 | Health and Fitness(HF) | 18551 | 98 | 85 | 25 | 120 | 160 |
| DS13 | Libraries and Demo (LD) | 15655 | 70 | 90 | 110 | 106 | 400 |
| DS14 | Lifestyle (LS) | 17650 | 155 | 150 | 150 | 195 | 190 |
| DS15 | Media and Video (MV) | 18019 | 90 | 120 | 160 | 460 | 76 |
| DS16 | Medical(ME) | 16000 | 65 | 100 | 65 | 165 | 165 |
| DS17 | Music and Audio (MA) | 27057 | 12 | 12 | 13 | 27 | 22 |
| DS18 | News and Magazines (NM) | 18164 | 500 | 42 | 500 | 200 | 22 |
| DS19 | Personalization (PE) | 14334 | 100 | 100 | 100 | 100 | 32 |
| DS20 | Photography (PH) | 19133 | 100 | 516 | 250 | 250 | 62 |
| DS21 | Productivity (PR) | 19850 | 100 | 120 | 50 | 96 | 500 |
| DS22 | Racing (RA) | 17766 | 100 | 100 | 120 | 150 | 50 |
| DS23 | Shopping (SH) | 12673 | 50 | 100 | 210 | 100 | 180 |
| DS24 | Social (SO) | 26159 | 100 | 240 | 100 | 450 | 112 |
| DS25 | Sports (SP) | 22669 | 100 | 50 | 210 | 150 | 150 |
| DS26 | Sports Games (SG) | 13889 | 120 | 500 | 550 | 475 | 563 |
| DS27 | Tools (TO) | 13346 | 100 | 145 | 145 | 650 | 198 |
| DS28 | Transportation (TR) | 13796 | 120 | 23 | 700 | 50 | 25 |
| DS29 | Travel and Local (TL) | 23180 | 400 | 320 | 140 | 68 | 90 |
| DS30 | Weather (WR) | 12841 | 2 | 500 | 100 | 100 | 20 |

**Table 2** Top malware families used in our dataset

| ID | Family | # of samples | ID | Family | # of samples | ID | Family | # of samples |
|---|---|---|---|---|---|---|---|---|
| A1 | Airpush | 150 | A2 | AndroRAT | 140 | A3 | Andup | 300 |
| A4 | Aples | 120 | A5 | BankBot | 100 | A6 | Bankun | 133 |
| A7 | Boqx | 130 | A8 | Boxer | 122 | A9 | Cova | 100 |
| A10 | Dowgin | 100 | A11 | DroidKungFu | 100 | A12 | Erop | 120 |
| A13 | FakeAngry | 110 | A14 | FakeAV | 120 | A15 | FakeDoc | 120 |
| A16 | FakeInst | 110 | A17 | FakePlayer | 120 | A18 | FakeTimer | 120 |
| A19 | FakeUpdates | 120 | A20 | Finspy | 111 | A21 | Fjcon | 123 |
| A22 | Fobus | 102 | A23 | Fusob | 181 | A24 | GingerMaster | 192 |
| A25 | GoldDream | 20 | A26 | Gorpo | 120 | A27 | Gumen | 20 |
| A28 | Jisut | 62 | A29 | Kemoge | 720 | A30 | Koler | 200 |
| A31 | Ksapp | 290 | A32 | Kuguo | 100 | A33 | Kyview | 500 |
| A34 | Leech | 300 | A35 | Lnk | 100 | A36 | Lotoor | 20 |
| A37 | Mecor | 29 | A38 | Minimob | 330 | A39 | Mmarketpay | 200 |
| A40 | MobileTX | 500 | A41 | Mseg | 230 | A42 | Mtk | 200 |
| A43 | Nandrobox | 100 | A44 | Obad | 100 | A45 | Opfake | 120 |
| A46 | Penetho | 120 | A47 | Ramnit | 120 | A48 | Roop | 120 |
| A49 | RuMMS | 100 | A50 | SimpleLocker | 110 | A51 | SlemBunk | 120 |
| A52 | SmsKey | 120 | A53 | SMsZombie | 110 | A54 | Spambot | 115 |
| A55 | SpyBubble | 120 | A56 | Stealer | 300 | A57 | Steek | 230 |
| A58 | Svpeng | 20 | A59 | Tesbo | 21 | A60 | Triada | 200 |
| A61 | Univert | 210 | A62 | UpdtKiller | 100 | A63 | Utchi | 300 |
| A64 | Vidro | 92 | A65 | VikingHorde | 230 | A66 | Vmvol | 533 |
| A67 | Winge | 190 | A68 | Youmi | 689 | A69 | Zitmo | 230 |
| A70 | Ztorg | 1000 | A71 | Imlog | 50 | A72 | SMSreg | 50 |
| A73 | Gappusin | 50 | A74 | Adrd | 50 | A75 | Geinimi | 100 |
| A76 | Kmin | 157 | A77 | Plankton | 125 | A78 | GingerMaster | 100 |
| A79 | Iconosys | 100 | A80 | SendPay | 18 | A81 | GoldDream | 200 |

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

**Input:** Collected Android apps
**Output:** Permission feature vector A and API calls vector B
A=[]; B=[]; t = Total number of permissions
**foreach** *file in raw data folder* **do**
  **foreach** *line in file* **do**
    remove all information except the permissions name and API call;
  **end**
  store files in permission name folder and API call folder;
**end**
**foreach** *file in permission name folder* **do**
  **foreach** *line in file* **do**
    map permission name to integers i as feature name;
  **end**
  **foreach** *line in file* **do**
    map API call to integers j as feature name;
  **end**
  store files in mapped-integer data folder;
**end**
**foreach** *m=1:t* **do**
  **foreach** *n=1* **do**
    A[m]=0;
  **end**
**end**
**foreach** *i and j in mapped-integer data folder* **do**
  A[i]=1;
  B[j]=1;
**end**

**Algorithm 1:** Permission and API calls extraction.

**Table 3** Top used permissions

| Highest Rank | Name of the permission | API call |
|---|---|---|
| 1 | INTERNET | android |
| 2 | ACCESS_DIRECTLY_CALL_PHONE_NUMBERS | android.util |
| 3 | ACCESS_BLUETOOTH_ADMINISTRATION | android.widget |
| 4 | ACCESS_CONTROL_VIBRATOR | android.renderscript |
| 5 | ACCESS_MOCK_LOCATION_SOURCES_FOR_TESTING | android.webkit |
| 6 | ACCESS_MAKE_RECEIVE_INTERNET_CALLS | android.media |
| 7 | ACCESS_CONTROL_FLASHLIGHT | android.media.effect |
| 8 | ACCESS_INTERCEPT_OUTGOING_CALLS | android.media.audiofx |
| 9 | ACCESS_VIEW_NETWORK_STATE | android.service.textservice |
| 10 | ACCESS_FINE_LOCATION | android.service.dreams |
| 11 | ACCESS_SURFACE_FLINGER | android.service.notification |
| 12 | ACCESS_DISABLE_KEYLOCK | android.service.wallpaper |
| 13 | ACCESS_ALLOW_WIFI_MULTICAST_RECEPTION | android.os |
| 14 | ACCESS_MANAGE_THE_ACCOUNTS_LIST | android.os.storage |
| 15 | ACCESS_MAKE_APPLICATION_ALWAYS_RUN | android.content |
| 16 | ACCESS_BLOGGER | android.content.res |
| 17 | ACCESS_KILL_BACKGROUND_PROCESSES | android.content.pm |
| 18 | ACCESS_READ_SYNC_STATISTICS | android.test |
| 19 | ACCESS_READ_EMAIL_ATTACHMENTS | android.test.suitebuilder |
| 20 | ACCESS_SEND_PACKAGE_REMOVED_BROADCAST | android.test.suitebuilder.annotation |

Doi :10.5281/zenodo.3930808
Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

**Table 4** Formulation of Sets containing (App downloaded by number of users, permissions, API calls and rating of the app) as an features.

| No. | Description related to | No. | Description related to |
|---|---|---|---|
| FS1 | PHONE_STATE and PHONE_CONNECTION | FS2 | AUDIO and VIDEO |
| FS3 | BUNDLE | FS4 | LOG_FILE |
| FS5 | SYNCHRONIZATION _DATA | FS6 | CONTACT_INFORMATION |
| FS7 | SYSTEM_SETTINGS | FS8 | BROWSER_INFORMATION |
| FS9 | CALENDAR_INFORMATION | FS10 | ACCOUNT_SETTINGS |
| FS11 | LOCATION_INFORMATION | FS12 | WIDGET |
| FS13 | SYSTEM_TOOLS | FS14 | NETWORK_INFORMATION and BLUETOOTH_INFORMATION |
| FS15 | UNIQUE_IDENTIFIER | FS16 | FILE_INFORMATION |
| FS17 | SERVICES_THAT_COST_YOU_MONEY | FS18 | PHONE_CALLS |
| FS19 | DATABASE_INFORMATION | FS20 | IMAGE |
| FS21 | Contain info. Related to API calls | FS22 | Contain info. Related to rating and downloads |
| FS23 | YOUR_ACCOUNTS | FS24 | STORAGE_FILE |
| FS25 | SMS_MMS | FS26 | READ |
| FS27 | ACCESS_ACTION | FS28 | READ_AND_WRITE |
| FS29 | HARDWARE_CONTROLS | FS30 | Default group |

### 5.1 Creation of feature dataset

After collecting Android apps from different promise repositories, we extract permissions and API calls which were demanded by an Android apps during its installation and run-time. We extract 1532 unique permissions and 310 API calls for developing malware detection model. List of extracted permissions and API calls are available for researchers and academicians [19]. A total of 1844-dimensional Boolean vector, where "1" implies that the app requires the feature and "0" implies that the feature is not required. It is very common that benign and normal apps may request a similar set of permissions and API calls for its execution. Permissions overview given by Google is used to describe the behavior of a permission i.e., "dangerous" or "normal". After extracting the permissions and API calls, we divide them into thirty different feature sets which are shown in Table 4. In this research paper, we also consider the rating of an app and number of the user download the app as a features. To normalize the data, we used the Min-max approach. This approach based on the principle of a linear transformation, which bring each data point $D_{q_i}$ of feature $Q$ to a normalized value $D_{q_i}$, that lie in between $0-1$. Following equation is considered to find the normalized value of $D_{q_i}$ :

$$Normalized(D_{q_i}) = \frac{D_{q_i} - min(Q)}{max(Q) - min(Q)},$$

where $min(Q)$ & $max(Q)$ are the minimum and maximum significance of attribute $Q$, respectively.

## 6 Machine learning technique

To validate that our collected dataset help researchers in the field of cybersecurity or not. We verify it, by implementing five distinct machine learning techniques. To develop an effective and efficient Android malware detection model, we consider the Deep Learning Model (i.e.,DNN) [11-12, 17-18], Decision Tree [2,16], Random Forest [2,15], Adaboost[13] and Naïve Bayes [2,14,15] as a machine learning technique.

## 7 Evaluation of performance parameters

In this section of the paper, we discuss the fundamental definitions of the performance parameters utilized by us while evaluating our proposed model for malware detection. The confusion matrix is used to calculate all these parameters. It consists of information related to actual and detected classification built by detection models. Table 5 demonstrates the confusion matrix for the malware detection model. In this study, two performance parameters namely, F-Measure, and Accuracy are employed for measuring the performance of malware detection models. Below we yield formulas to evaluate Accuracy and F-measure:

$$Accuracy = \frac{N_{Benign \rightarrow Benign} + N_{Malware \rightarrow Malware}}{N_{classes}}.$$

[19]http://dx.doi.org/10.17632/b4mxg7ydb7.3

[19]https://developer.android.com/guide/topics/permissions/overviewr

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

**Table 5** Confusion matrix Used in this study

| | **Benign** | **Malware** |
|---|---|---|
| Malware | Benign->Malware | Malware->Malware |
| Benign | Benign->Benign | Benign->Malware |

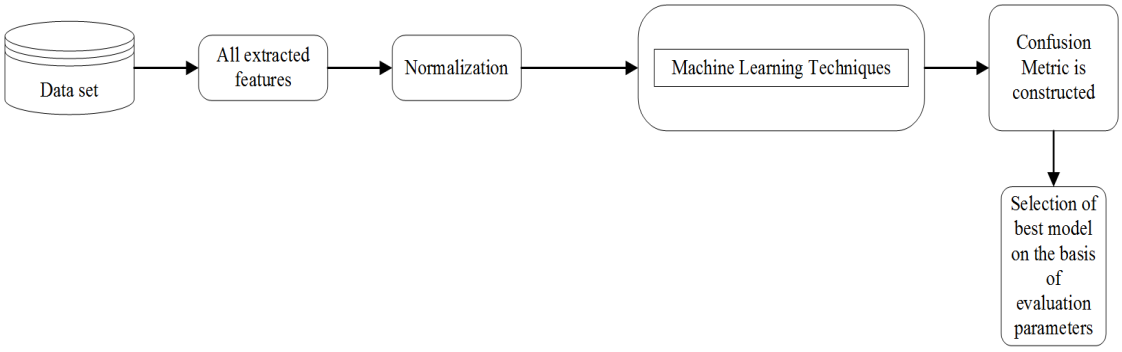


**Figure 2** Framework of proposed malware detection model.

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * a}{2 * a + b + c}, \quad (1)$$

where $a = N_{Malware \rightarrow Malware}$,
$b = N_{Benign \rightarrow Malware}$,
$c = N_{Malware \rightarrow Benign}$

# 8 Experimental setup

In the present section, we introduce the experimental setup done to find the performance of our collected dataset by using five distinct machine learning algorithms. All these datasets have a varying number of benign or malware apps that are adequate to perform our analysis. Figure 2 demonstrates the framework of proposed malware detection model.

# 9 Results of performed experiment:

In the current section of the paper, the relationship among different feature sets and malware detection at the class level is submitted. The set of features is used as an input and presents the ratio of benign and malware apps within an experiment. F-measure and Accuracy are used as performance assessment parameters to match the performance of the Android malware detection model developed by using supervised machine learning algorithms.

## 9.1 *Machine Learning Techniques :*

Hardware used to carry out this study is the Intel Core i9 processor having a secondary memory of 1TB hard disk and primary memory of 16GB. Models are developed by using the MATLAB environment. Further, the performance of each detection model is measured by using two distinct performance parameters i.e., F-measure and Accuracy. Table 6, present the outcomes obtained for distinct datasets by utilizing Deep Learning Model (i.e.,DNN) [11-12], Decision Tree (DT) [2], Random Forest (RF) [2], Adaboost (AB) and Naïve Bayes (NB) [2] as a machine learning technique.

From Table 6, it may be concluded that:

- Extracted features dataset is capable to detect malware from Android apps.
- Among implemented machine-learning techniques, DNN is more capable to detect malware-infected Android apps.

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

**Table 6** Accuracy and F-measure

| **Accuracy** | | | | | | **F-measure** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **ID** | **NB** | **RF** | **AB** | **DT** | **DNN** | **NB** | **RF** | **AB** | **DT** | **DNN** |
| DS1 | 83 | 85 | 86 | 83 | **89.8** | 0.85 | 0.82 | 0.87 | 0.81 | **0.89** |
| DS2 | 82 | 85 | 86 | 89 | **91.8** | 0.85 | 0.83 | 0.85 | 0.81 | **0.87** |
| DS3 | 83 | 81 | 84 | 89 | **90.8** | 0.86 | 0.85 | 0.84 | 0.87 | **0.89** |
| DS4 | 89 | 85 | 87 | 89 | **90.7** | 0.86 | 0.87 | 0.81 | 0.86 | **0.88** |
| DS5 | 87 | 82 | 83 | 85 | **89.8** | 0.84 | 0.85 | 0.86 | 0.87 | **0.90** |
| DS6 | 88 | 89 | 92 | 94 | **96.7** | 0.85 | 0.88 | 0.87 | 0.88 | **0.90** |
| DS7 | 88.7 | 86 | 86.8 | 89.7 | **93.8** | 0.86 | 0.87 | 0.82 | 0.81 | **0.89** |
| DS8 | 85 | 86 | 87 | 88 | **91** | 0.83 | 0.84 | 0.84 | 0.88 | **0.89** |
| DS9 | 84 | **96** | 95 | 93 | 86 | 0.96 | **0.99** | 0.91 | 0.92 | 0.91 |
| DS10 | 89 | 89 | 89.8 | 89.7 | **97** | 0.85 | 0.88 | 0.82 | 0.88 | **0.96** |
| DS11 | 89 | 80 | 86 | 88 | **98** | 0.85 | 0.84 | 0.82 | 0.85 | **0.93** |
| DS12 | 83 | 88 | 87 | 89 | **90** | 0.82 | 0.86 | 0.86 | 0.84 | **0.89** |
| DS13 | 87 | 82 | 86 | 88 | **89.8** | 0.81 | 0.82 | 0.80 | 0.81 | **0.88** |
| DS14 | 82 | 81 | 86 | 86 | **90.9** | 0.83 | 0.85 | 0.81 | 0.82 | **0.89** |
| DS15 | 88 | 89.8 | 91 | 92 | **97** | 0.83 | 0.84 | 0.86 | 0.86 | **0.94** |
| DS16 | 83 | 81 | 86 | 87 | **88** | 0.83 | 0.80 | 0.80 | 0.81 | **0.83** |
| DS17 | 92 | 93 | 96 | 91 | **98** | 0.82 | 0.85 | 0.88 | 0.98 | **1** |
| DS18 | 91 | 92.9 | 95 | 96 | **98.9** | 0.85 | 0.86 | 0.88 | 0.90 | **0.93** |
| DS19 | 93 | 92 | 95 | 96 | **97** | 0.91 | 0.92 | 0.90 | 0.88 | **0.95** |
| DS20 | 92 | 93 | **95** | 91 | 92 | 0.87 | 0.88 | **0.89** | 0.84 | 0.85 |
| DS21 | 80.7 | 82 | 83 | 84 | **85.7** | 0.88 | 0.88 | 0.87 | 0.89 | **0.9** |
| DS22 | 90 | 92 | 96 | 95 | **98** | 0.85 | 0.85 | 0.88 | 0.89 | **0.91** |
| DS23 | 87.71 | 87.9 | **91** | 90 | 90.1 | **0.85** | 0.83 | 0.82 | 0.80 | 0.82 |
| DS24 | 89.2 | **91.3** | 90 | 89.7 | 88 | 0.82 | **0.89** | 0.82 | 0.85 | 0.85 |
| DS25 | 92 | 93 | 96 | 97 | **98.8** | 0.89 | 0.89 | 0.89 | 0.88 | **1** |
| DS26 | 96.1 | 94 | 97 | 95 | **97.9** | 0.86 | 0.85 | 0.88 | 0.86 | **0.92** |
| DS27 | 91.9 | 93.6 | 97 | 95.8 | **97.9** | 0.87 | 0.85 | 0.86 | 0.87 | **0.91** |
| DS28 | 91 | **98** | 86 | 81 | 91 | 0.86 | **0.89** | 0.88 | 0.85 | 0.88 |
| DS29 | 87 | **92** | 85 | 84 | 89 | 0.87 | 0.86 | **0.87** | 0.85 | 0.88 |
| DS30 | 91 | 92 | 95 | 91 | **98** | 0.87 | 0.85 | 0.87 | 0.87 | **1** |

Doi :10.5281/zenodo.3930808
Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

## 10 Leveraging PARUdroid

This dataset has been used to research in the field of Android malware detection using machine learning techniques in small portions. Mahindru and Singh [2] used 11,000 Android apps in their study and able to achieve an acceptable detection rate. Mahindru and Sangal [5-6] used 2,00,000 Android apps to develop malware detection framework, which is capable of detecting malware from real-world apps.

Potential usages could be found in the fields of large scale studies on permission usage and API calls, Malware analysis, application similarity, developing new frameworks for malware detection, etc.

## 11 Limitations

The main limitation while collecting Android apps is a lack of storage capacity. The collection of Android apps was interrupted for the number of days, weeks, or even the number of months for this. That's why it took several years to download the app and extract features from them. Second, due to a shortage of funds, we consider only limited paid apps. The third limitation is, sometimes app repositories took various measures to prevent their app market from being automatically mined. Then in several cases, .apk file download has size 0 MB, but at the time of extraction of features from them, they demand more features when compared to the app belonging to a similar category.

## 12 Conclusions

This work is emphasized on collecting a Android malware detection dataset, whether a particular Android app belongs to malware class or benign class. The experiment was performed by taking assistance of thirty distinct categories of Android apps.

Our submissions after performing the experiment are the following:

- On the basis of experimental finding, we observed that by taking extracted features dataset malware detection models are developed that are capable to detect malware from real-world apps.
- On the basis of the proposed detection framework, it is seen that model build by utilizing DNN is capable to detect 98.8% unknown malware from real-world apps.

In this study, we have presented the PARUDroid dataset having 5,60,142 unique Android apps collected from distinct promised repositories. We extract different features from them and is readily available for the research community to contribute more reliable, generalizable and reproducible studies based on representative, large-scale, and up-to-date samples. Further, work can be extended to develop a dataset for malware detection which predicts whether a particular feature is capable to detect malware or not.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

[1] Anderson, Hyrum S., and Phil Roth. "Ember: an open dataset for training static pe malware machine learning models." arXiv preprint arXiv:1804.04637 (2018).

[2] Mahindru, Arvind, and Paramvir Singh. "Dynamic permissions based android malware detection using machine learning techniques." In Proceedings of the 10th innovations in Software Engineering Conference, 2017. pp. 202-210.

[3] Kadir, Andi Fitriah Abdul, Natalia Stakhanova, and Ali Akbar Ghorbani. "Android botnets: What urls are telling us." In International Conference on Network and System Security, Springer, Cham, 2015. pp. 78-91.

[4] Zhou, Yajin, and Xuxian Jiang. "Dissecting android malware: Characterization and evolution." In 2012 IEEE symposium on security and privacy, IEEE, 2012. pp. 95-109.

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020

[5] Mahindru, Arvind, and A. L. Sangal. "PerbDroid: Effective Malware Detection Model Developed Using Machine Learning Classification Techniques." In A Journey Towards Bio-inspired Techniques in Software Engineering, Springer, Cham, 2020. pp. 103-139.

[6] Mahindru, Arvind, and A. L. Sangal. "Feature-Based Semi-supervised Learning to Detect Malware from Android." In Automated Software Engineering: A Deep Learning-Based Approach, Springer, Cham, 2020. pp. 93-118.

[7] Zhou, Yajin, and Xuxian Jiang. "Dissecting android malware: Characterization and evolution." In 2012 IEEE symposium on security and privacy, IEEE, 2012. pp. 95-109.

[8] Wei, Fengguo, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. "Deep ground truth analysis of current android malware." In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, Cham, 2017. pp. 252-276.

[9] Allix, Kevin, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. "Androzoo: Collecting millions of android apps for the research community." In 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), IEEE, 2016. pp. 468-471.

[10] Lindorfer, Martina, Matthias Neugschwandtner, Lukas Weichselbaum, Yanick Fratantonio, Victor Van Der Veen, and Christian Platzer. "Andrubis–1,000,000 apps later: A view on current Android malware behaviors." In 2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS), IEEE, 2014. pp. 3-17.

[11] Mahindru, Arvind, and A. L. Sangal. "DLDroid: Feature Selection based Malware Detection Framework for Android Apps developed during COVID-19". International Journal on Emerging Technologies, 11(3), 2020. pp. 516-525.

[12] Mahindru, Arvind, and A. L. Sangal. "GADroid: A framework for Malware Detection from Android by using Genetic Algorithm as Feature Selection approach". International Journal of Advanced Science and Technology, Vol. 29, No. 5, 2020. pp. 5532 - 5543

[13] Zarni Aung, Win Zaw. "Permission-based android malware detection." International Journal of Scientific & Technology Research 2, no. 3, 2013. pp. 228-234.

[14] Amos, Brandon, Hamilton Turner, and Jules White. "Applying machine learning classifiers to dynamic android malware detection at scale." In 2013 9th international wireless communications and mobile computing conference (IWCMC), IEEE, 2013. pp. 1666-1671.

[15] Yu, Wei, Hanlin Zhang, Linqiang Ge, and Rommie Hardy. "On behavior-based detection of malware on android platform." In 2013 IEEE global communications conference (GLOBECOM), IEEE, 2013. pp. 814-819.

[16] Alam, Mohammed S., and Son T. Vuong. "Random forest classification for detecting android malware." In 2013 IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing, IEEE, 2013. pp. 663-669.

[17] Zarni Aung, Win Zaw. "Permission-based android malware detection." International Journal of Scientific & Technology Research 2, no. 3, 2013. pp. 228-234.

[18] Yuan, Zhenlong, Yongqiang Lu, and Yibo Xue. "Droiddetector: android malware characterization and detection using deep learning." Tsinghua Science and Technology 21, no. 1, 2016. pp. 114-123.

[19] Mahindru, Arvind, and A. L. Sangal. "DeepDroid: Feature Selection approach to detect Android malware using Deep Learning." In 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), IEEE, 2019. pp. 16-19.

Doi :10.5281/zenodo.3930808

Received: Feb 1 2020; Revised: April 5 2020; Accepted: April 28 2020