ANDROGRAPHIC: A WEB-BASED TOOL FOR VISUALIZING ANDROID APPS SETTING

A REPORT
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF SAN FRANCISCO STATE UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Tejal Rajendra Ghadge
December 2020

CERTIFICATION OF APPROVAL

I certify that I have read “AndroGraphic: A Web-Based Tool for Visualizing Android Apps Setting" by Tejal Rajendra Ghadge, and that in my opinion this work meets the criteria for approving a project submitted in partial fulfillment of the requirements for the degree: Master of Science of Computer Science at San Francisco State University.

Abeer AlJarrah 12/11/2020

Abeer AlJarrah,
Assistant Professor of Computer Science

Shahrukh Humayoun,
Assistant Professor of Computer Science

Abstract

Analyzing Android apps for different purposes such as studying privacy and security concerns of third-party libraries or analyzing software engineering patterns among Android developers, is a common and often repetitive task in research. Such a task typically entails collecting and analyzing a large dataset of apps. Yet, these datasets are not being effectively shared and distributed to the research community, which puts a limit on the opportunity to compare research results and advancing the field. In this work we present a platform for sharing, merging, exploring, downloading charts, and downloading/uploading large datasets of Android applications. The tool is currently supporting a dataset of 2 million contextual records and 1 million technical records collected from the official Google Play store. The purpose of this tool to help fellow researchers to analyze different security aspects of android applications by building own charts, exploring in-built summary charts, and finding patterns by gaining knowledge from the correlation between different measures to identify the malicious nature of the applications. This tool indeed helps other researchers to distribute their datasets and conduct security analytic work pertaining Android apps.

Acknowledgments

I would like to express my sincere gratitude towards my project advisor Dr. Abeer Aljarrah for this opportunity and her invaluable guidance throughout this year. I am indebted to Dr. Abeer for her immense support and for being flexible during such hard times. I wish to thank Dr. Fadi Mohsen for sharing the research data and guiding me during the initial stages.

I would like to extend my gratitude to my second committee member, Dr. Shah Rukh Humayoun, for showing interest in my project, providing invaluable feedback, and attending my project defense. I am grateful to our department chair, Dr. Arno Puder, and all the instructors and staff of the SF state Computer Science department for giving me an opportunity and resources for making this research and defense possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The mobile app industry is quite challenging due to the changing nature of the underlying platforms. The Android operating system, for example, has gone through numerous changes ever since it was first introduced to the market. Some developers cannot keep up with these constant modifications, which would result in having an online store full of vulnerable and/or buggy applications. Examples of these modifications, permissions prompt changed from upon installation to at run-time, programming languages from Java to Kotlin, and many more. Cybersecurity research efforts are closely conjugated with emerging and/or heavily used technologies. The world today heavily relies on mobile hardware and software innovation more than any time [3]. Android maintained its position as the leading mobile operating system worldwide in December 2019, controlling the mobile OS market with a 74.13 percent share [9]. One of the main issues in cybersecurity research is the prominent asymmetric dynamics of cyber warfare [8], which in its existing form, significantly favors attackers over defenders in terms of required effort and resources, exerted damage, and acquired knowledge. This is because the existing research efforts are scattered and in essence reactive in nature. The lack of public raw data and individual diffused efforts on data collection result in research findings that are based on a sliced data. There is a dire need to have a collective, public, and interactive repository available for researcher's disposal. The objective of this work is as follow: 1)provide a data visualization platform for researchers who do not have the skills to analyze the datasets or do not wish to waste time or resources, the ability to query the dataset, generate, and download figures for them to use in their articles. 2) provide a platform for researchers to share their Android datasets with the rest of the community 3) provide an in-built machine learning model to classify the application as malicious or not. 4)for researchers who are interested in analyzing the dataset can either choose all of it or a portion of it

The current implementation provides views for displaying and comparing the apps and their contextual and technical features. The repository currently has a dataset of 2 million applications collected from the Google play store. The contextual information are simply what users see on

Google play store pages. The technical features represents the *permissions* and *system actions* retrieved from the applications' *AndroidManifest* files.

In this report, we discuss (i) the data collection and dataset building process, (ii)different characteristics and attributes of this dataset (iii) database organization, and (iv) AndroGraphic: a web portal developed to share the existing data, to upload the new data, to classify the android apk as malicious or benign, and very importantly analyze the android data and make the analytical results available to use in other researches

Chapter 2

Related Work

2.1 Related Work

Most of the work that has been done is in the form of collecting datasets, performing analysis and deducing results. We categorize related efforts by the purpose of collecting apps datasets as follows:

Research Community The need to provide a set of android apps for research purposes has been addressed in different ways. In this work [6], researchers collect slightly over a thousand applications, to be precise 1179. Their dataset contains features such as permissions, intents, and minimum SDK, obtained from running several static analysis tools. The dataset is built by mining FDroid android app repository to collect version control commits that are used for over-permissions or potential bugs as compare to the app's previous version. The researchers use Stowaway, Androrisk, Sonar tools for performing static analysis. The tool offers fixed queries 2.2 and results are shown in tabular format as shown in figure 2.1.


Show 10 entries

Search: FB

Name	# Versions	Current Version	Repo Type
FBReader TTS+ Plugin	12	3.5.4	git
FBReader TTS plugin	2	1.2	git
FBReader	37	1.8.2	git
FBReader Calibre connector	2	1.2	git

Showing 1 to 4 of 4 entries (filtered from 1,179 total entries)

Previous 1 Next

Figure 2.1: A Dataset of Open-Source Android Applications: Tabular view [6]

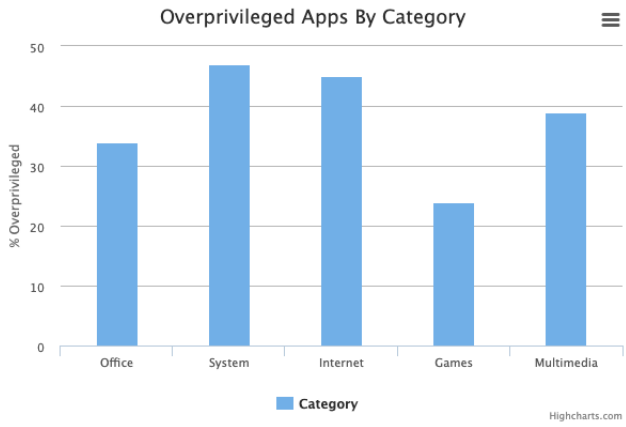

Figure 2.2: Over-privileged apps by category: static view [6]

Another work [1] presents the AndroZoo, a collection of more than 3 million Android Applications collected from several sources. The apps are scanned using several Anti-Virus tools. While the tool is online, the APK Analysis and search APIs are dead now. Though the user can get access to download an android apk by following a certain procedure. The researchers the list of applications flagged as malware and hence is vital information. The paper shows the distribution of applications from various sources that are flagged as malware are shown in the following figure 2.3. This paper allows user to access the apk however, fails to provide a filtered, collective, ready to use dataset.

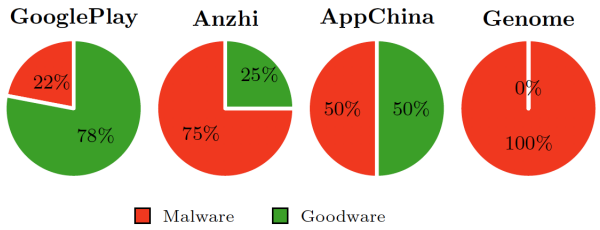

Figure 2.3: Share of malware in dataset: Androzoo [2]

Avdiienko et al.[2] developed a toolchain for mining Android Apps metadata and code for researchers. The analysis models are fixed to clustering, topics modeling and anomaly detection. In 2018, Geiger and Malavolta [4] surveyed 31 existing Android datasets for the purpose of classification and comparison. The work highlights several limitations such as outdated data, lack of documented access, data inaccessibility, incompleteness and lack of coordination among researchers.

Malware Analysis Using large malware samples collected from leading anti-virus repositories to detect malware behaviors is a common testing mechanism. Several approaches for detection are used, examples include Yerima et al. [12] who use ensemble machine learning, Wei et al. [11] who use

over 24 thousand apps and leverage a mix of manual analysis, anti-virus scan results, and automation techniques in categorizing to help understand malicious apps behavior. The paper is based on the foundation of ensemble machine learning, where Random Forest combines the Random Decision Tree with bagging for achieving high classification accuracy. The paper shows a comparative study between Random Forest and Naïve Bayes, Decision Trees, Random Trees, and Simple Logistic.

Analyze/Test Different Aspects of Apps Using datasets to either test a specific approach or discern a specific feature related to Android Apps is a common and justifiable practice. Examples include locating potential performance bottleneck where researchers analyzed a set of 500,000 apps from Google Play store [10]. Grano et al. [5] showed that information available in user reviews can be helpful in identifying the main issues users experience while using mobile applications such as GUI problems and crashes. Another example is a work that presents an approach to comprehend apps from a structural and historical perspective, leveraging three factors for the analysis: source code, usage of third-party APIs, and historical data [7]. In this paper, researchers provide a web-based software analytic platform called SAMOA that provides a snapshot view that shows revision of specific app(figure 2.4),a history view to depict the evolution and ecosystem view to see multiple apps at once. The entire view helps the user to understand information like the number of classes, Line of Code(LOC), Cyclomatic complexity, the number of internal and external calls, etc.

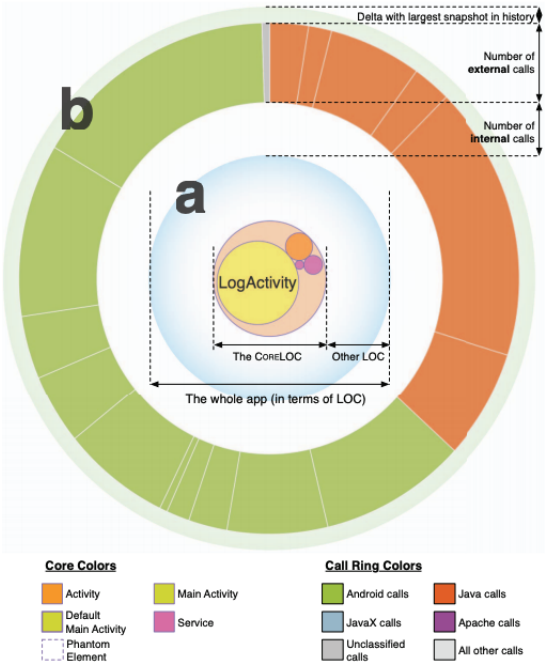

Figure 2.4: Snapshot view of Alogcat application [7]

In this work, we are trying to overcome limitations in previous work. Unlike [6], we have collected

data from over 1M android applications and built a large dataset which acts as a seed data for our analytic tool. The paper[1] indeed provides a vast repository for android apks but we have combined data obtained from parsing android apk files, scrapping Google playstore and merging virus total count from[1] to get more useful data in one dataset. By providing, AndroGraphic, the web analytic tool we want user to get a comparative, outline and, detailed views by analyzing data of 1M applications. We address the problems of the previous work, by (1) Providing comprehensive data that include contextual, technical and virus total count (2) offering interactive charting capabilities that enables users to build the query online (3) Providing greater accessibility by allowing users to download charts and/or datasets and upload datasets. (4) Providing an initial framework to identify the application class i.e. malicious or benign.

Chapter 3

Definitions

The terminologies used in this study are defined in this section. Android popularity has been increased in the last decade and many consumers with different knowledge are using android applications on daily basis. Many applications are released on app stores which may be vulnerable in many terms. There are some terminologies about android that the readers should be familiar with, which are defined as follows:

- **Permissions:** The permission allows the android application to grant access to restricted data or restricted actions.
- **System actions:** System actions are the intend activity performed when specific permission is granted. e.g. REBOOT
- **Genre:** Categories of android applications e.g. Entertainment.
- **Developer:** A person/company responsible for developing the application.
- **Normal:** A permission with a lower-risk value to other apps, system or the user, which provides access to app-level features. System by default grants this type of permission at installation even though the user always has an option to review.
- **Signature:** Permission granted to an app if another app with the same permission has the same certificate. If the certificates match the system grants the permission without asking for the user's approval.
- **Dangerous:** Permission with a higher-risk value to private user data or control over the device can have negative impact. Due to this reason, the system does not grant the permission without the user's explicit approval.

- **Malicious Index/VT Detection:** The count of the number of antivirus tools has detected an application as malware by AndroZoo
- **Malicious:** AndroGraphic labels an application as malicious if it is detected as malware by at least one antivirus tool i.e. Malicious index is one or greater.
- **Benign:** AndroGraphic labels an application as benign if the malicious index is zero.
- **Contextual Data:** Data extracted from Google Playstore.
- **Techinical Data:** Permissions and system actions data extracted from android apks.

Chapter 4

Methodologies

As the research project is based on the android dataset, we started with collecting android apps to build our own dataset. In this section, we focus on the process of collecting data, merging the dataset with AndroZoo to get the seed dataset.

4.1 Data Collection

To collect the data two crawlers have been used. The first crawler has been used to extract the information from the Google play store which results in populating 2M application records. These records are categories as contextual data. The second crawler has been used to download android apk files to extract permissions and system actions. This results in collecting around 1M android applications. The data collected with the help of these applications are categorized as technical data. The data collection process is depicted in the following figure 4.1.

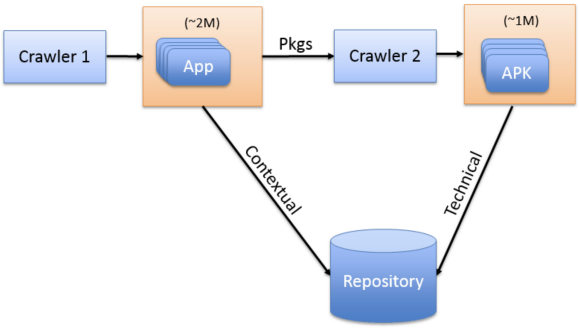

Figure 4.1: Data Collection

The applications downloaded from google store are presumed as benign but there are possibilities

that applications can be malicious as the researchers have been advised to run security tools to double-check. To our best knowledge, our dataset is the first to provide a collective of both technical and contextual android app information.

4.2 The Dataset

As we discussed earlier, the contextual and technical data are merged to build a seed dataset of 1M application records. The AndroZoo provides the list of applications detected as malware by the total number of antivirus tools called VT Detection [1]. This information is indeed important to know the malicious nature of any android app. We have combined the AndroZoo dataset and our own dataset based on application package name to get collective information of malicious index along with contextual and technical. The final dataset provides 1M application records. The minimum value of VT Detection/ malicious index present in the final dataset is 1 and the maximum is 54

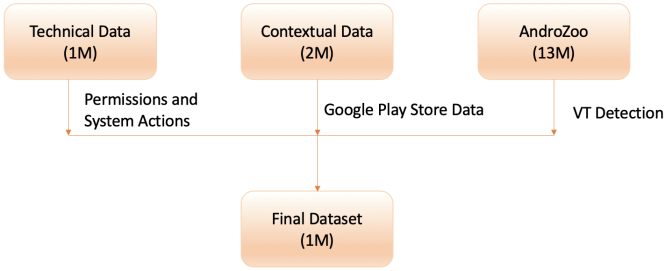

Figure 4.2: Combined Seed Dataset

The contextual information scraped from Google play store consists of total of 22 features: Ratings, Title, Four Star Ratings, Developer Address, Last Updated, Reviews Average, Price, Three Star Ratings, Privacy Policy Link, Genre, Five Star Ratings, One Star Ratings, Url, Content Rating, Current Version, Developer Email, Android Version, Developer Website, Developer Name, File Size, Two Star Ratings, and Downloads. The technical information is extracted from the manifest files, namely, the permissions and the system actions. There are a total of 174 system actions and 138 permissions. The android permissions are categorized in different protection levels namely Normal, Signature, Dangerous. These permissions need user approval at runtime before the app uses them. The dangerous permissions categorization is shown in table 4.2. The dataset consists of total of **1164115** applications. The application is categorized as malicious if it is detected as malware by at least one antivirus tool i.e., VT Detection value of AndroZoo. We call VT Detections as Malicious Index. Out of total applications, **151649** applications are malicious and remaining **1012466** are Benign. Table 4.2 shows the applications present the malicious index range.

Group	Permissions (Usage %)
CALENDAR	READ_CALENDAR (1.5) WRITE_CALENDAR (1.55)
CAMERA	CAMERA (11.8)
CONTACTS	READ_CONTACTS (3.9) WRITE_CONTACTS (1.95) GET_ACCOUNTS (13.55)
LOCATION	ACCESS_FINE_LOCATION (18.54) ACCESS_COARSE_LOCATION (18.51)
MICROPHONE	RECORD_AUDIO (6.44)
PHONE	READ_PHONE_STATE (20.72) CALL_PHONE (5.61) READ_CALL_LOG (0.36) WRITE_CALL_LOG (0.22) ADD_VOICEMAIL (0.01) USE_SIP (0.08) PROCESS_OUTGOING_CALLS (0.63)
SENSORS	BODY_SENSORS (0.01)
SMS	SEND_SMS (2.18) RECEIVE_SMS (1.63) READ_SMS (0.9) RECEIVE_WAP_PUSH (0.05) RECEIVE_MMS (0.08)
STORAGE	READ_EXTERNAL_STORAGE (14.21) WRITE_EXTERNAL_STORAGE (45.94)

Table 4.1: The dangerous permissions categories

Malicious Index range	Total Applications
0 - 5	1138261
6 - 10	18344
11 - 15	4888
16 - 20	1773
21 - 25	558
26 - 30	234
31 - 35	47
36 - 40	3
41 - 45	3
46 - 50	3
51 - 54	1

Table 4.2: Applications in malicious index range

Chapter 5

Architecture

This section describes the architecture of the AndoGraphic framework. As discussed earlier, the combined seed data consist of technical, contextual, and malicious index information. The dataset has been imported in the MySQL database.

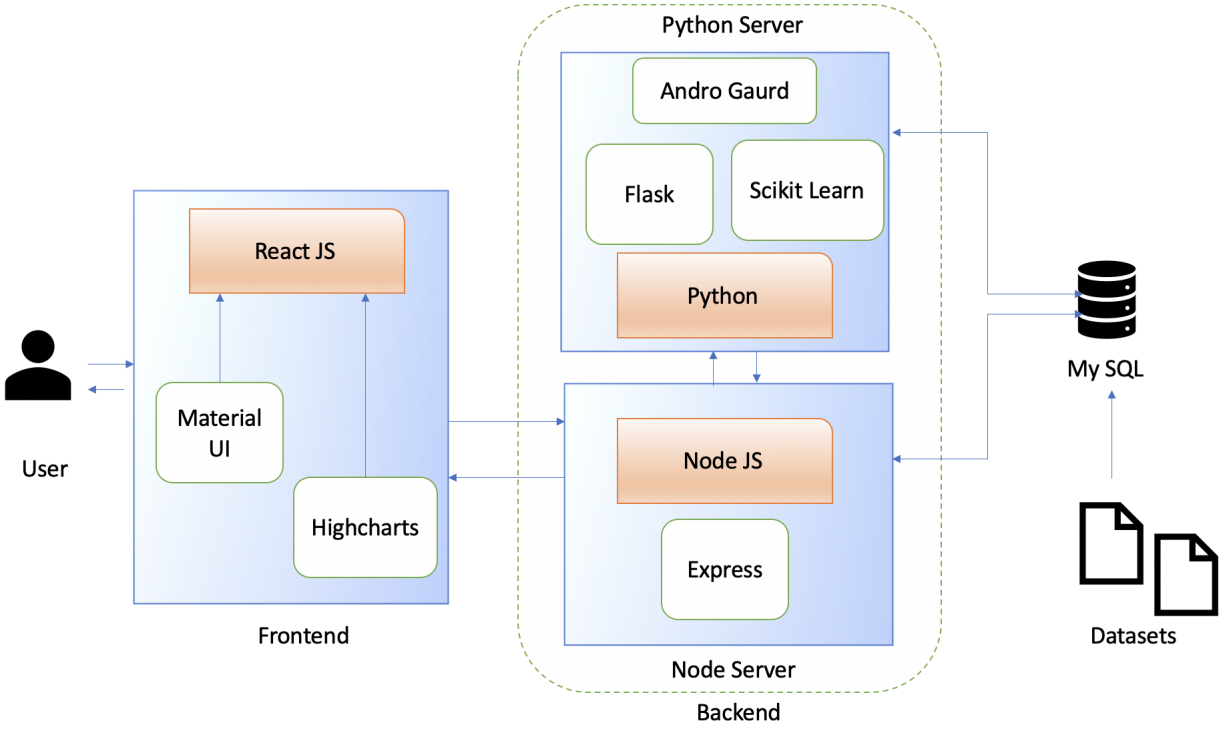

Figure 5.1: Architecture

The AndroGraphic is a web-based tool that is hosted on AWS EC2 instance and the database is

running on AWS RDS instance. User can access the tool from anywhere with the help of an access URL. The framework used for front-end development is React along with a few supporting libraries. All the react components are created with Material UI following the hybrid structure of class-based components and hook-based components. As the data visualization is a pivot functionality of AndorGraphic, we have used Highcharts JS for visualizing charts.

The backend server is responsible to handle all incoming REST services. Node JS is used with Express to route the different UI requests to appropriate REST controllers. The responsibility of the node server is to communicate with UI and provide the appropriate results by interacting with the MySQL database. The Node server is also responsible to spawn a thread for a Python server based on the URL pattern. A specific URL pattern has been used to redirect the matching request to the Python REST controller. The python script is used for doing Binary Classification of android application. All the machine learning operations are performed on a Python server where the Flask framework is used to expose REST services. We have also used the AndroGuard library to read the android apk and extract permissions and system actions. This information is later used for building data for classification. Python script also communicates with MySQL to populate data for model training.

5.1 Database Schema

For the persistent storage of the seed dataset, we have used a SQL database i.e., MySQL. The datasets which are present in CSV files have been imported in MySQL. Figure 5.2 shows the database schema used by AndroGraphic. All the contextual and technical information are imported in CONTEXTUAL and TECHNICAL table respectively. We have created two tables PERMISSION_VIEW, SYSTEM_ACTION_VIEW based on views which stores precalculated permissions and system actions data to boost the performance. The permission and system action names are mapped to different ids that are used as column names in technical table. This mapping is required to overcome the limitation of the length of the column name in MySQL. It is stored in the MAPPING table along with protection level information.MALWARE_INFO table stores a few fields of AndroZoo dataset used to merge VT Detection i.e., malicious index field. USER table stores all users who are registered to either download or upload the dataset. Along with our own dataset, AndroGraphic also allows uploading other datasets to visualize the data. The information of other datasets and their sources are maintained in the RESOURCE table.

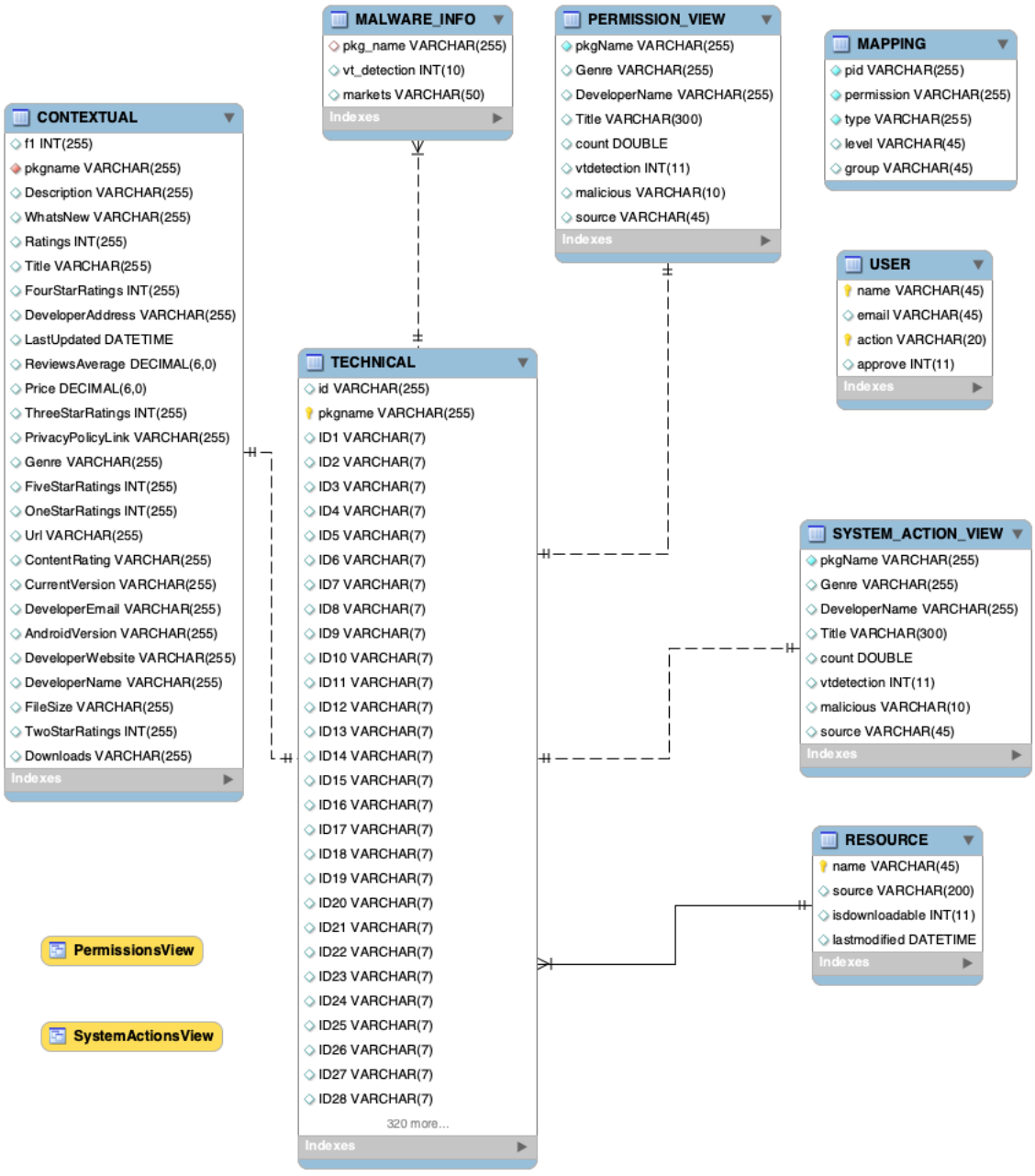

Figure 5.2: Database Schema

5.2 Components

The AndroGraphic web-based tool consists of many components that are responsible to provide different data visualization functionalities. Figure 5.3 shows the overview of the components.

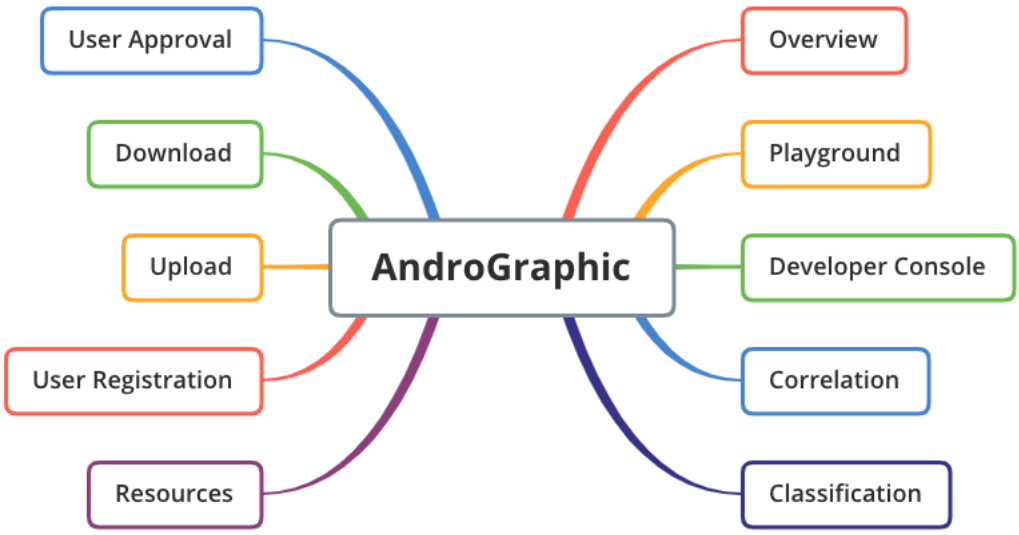

Figure 5.3: Components

Components named Overview, Playground, Developer Console and Correlations components can be used for visual analytics. The classification module helps to perform binary classification of android application. User Registration, User approval, Upload, Download, and Resources components are mainly responsible for data distribution to other researchers.

Chapter 6

The Tool: AndroGraphic

The AndroGraphic is build to provide various data visualization functionalities along with classification and data distribution. This section mainly focuses on the actual implementation of the tool components. The landing page of the application with all the components is shown in figure 6.1.

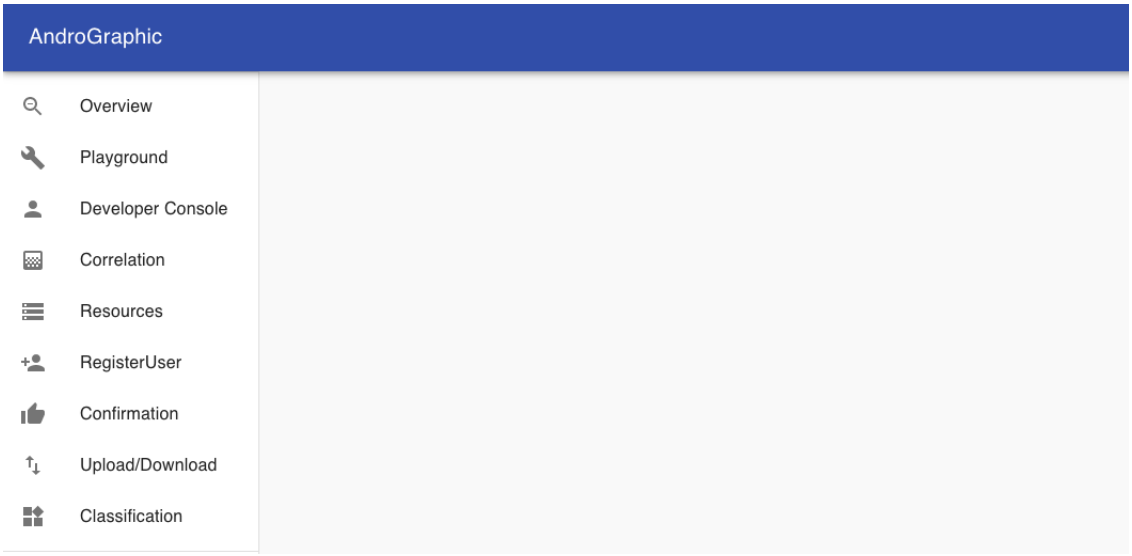

Figure 6.1: Home Page

The user can select one of the options to explore the charting functionalities. On most of the AndroGraphic components, we support a global filter for data.

- **All:** Populates the entire dataset
- **Malicious:** All applications with malicious index greater than or equal to one.

- **Benign:** All applications with malicious index equal to zero.

Along with this, users can also export the in-built or newly built chart for the other research work by using the export option that is provided on almost every chart. The user can click on the export button shown in figure 6.2 to export the chart in various formats.

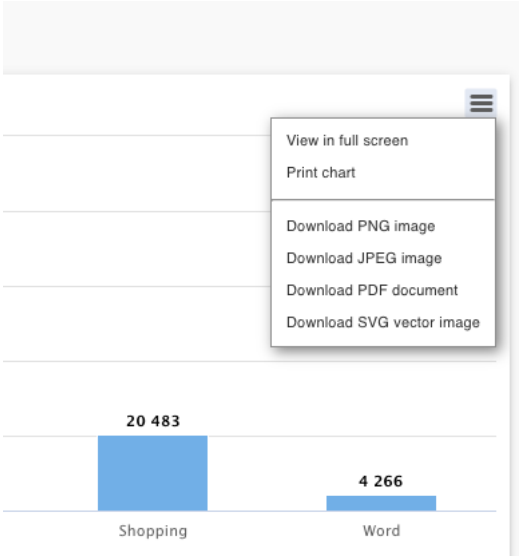

Figure 6.2: Export chart

6.1 Overview

The overview section provides an in-build dashboard of summary charts with filters to get a detailed overview of the dataset. The top tiles of the dashboard show the total, malicious and benign applications as well as the count of distinct developers who developed these applications. When the data is filtered on the malicious type, malicious index selection is made available. There are four different types of charts provided on overview view that shows the data in different views Bar charts, Pie chart, Word Cloud, and Bubble chart. The bubble chart is the default view and with the default view, user can see charts having a number of applications by genre, permission usage, system action usage, number of applications developed by the developer. In short, the overview dashboard provides the following insights:

- Total, malicious, benign apps count in a dataset
- Genre in which maximum applications are developed
- Top five widely used permissions

- Top five widely used system actions
- Developer whose has developed maximum malicious applications w.r.t malicious index
- Specific set of permissions repetitively used by malicious applications
- Specific set of system actions repetitively used by malicious applications
- Overview of genre, permission, system action and developer views w.r.t entire, benign and malicious index

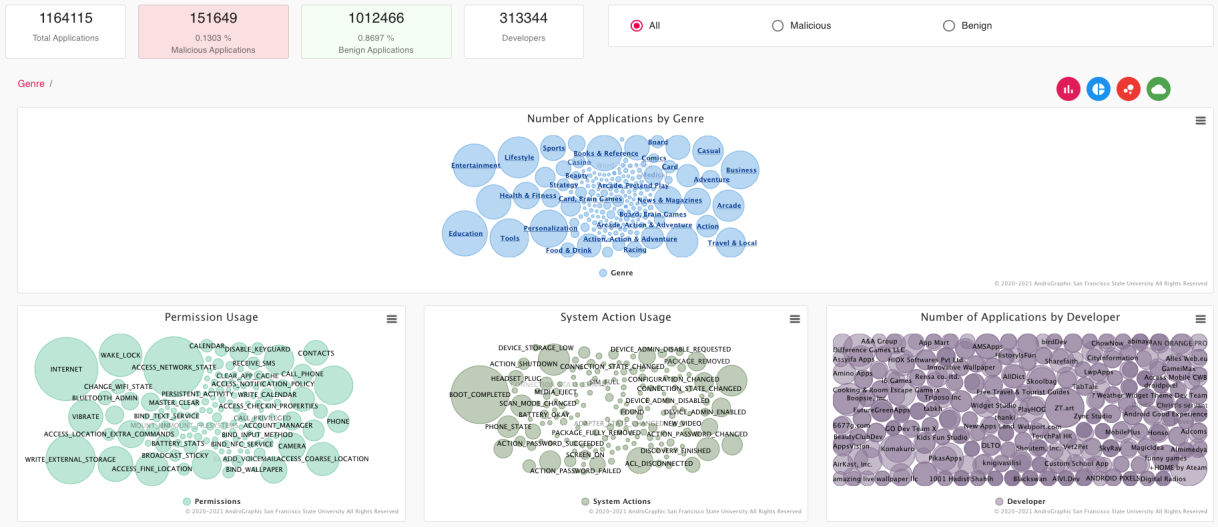

Figure 6.3: Overview of entire dataset

As the main highlight of the overview view is the Genre, we have provided a cross-link of the genre chart with all other charts. For example, by looking at the number of an application by genre chart one can infer that there are many applications present in Lifestyle genre so to get the detailed view user can click on Lifestyle genre and all other charts which are cross-linked with the genre charts will automatically be filtered. To remove the filter, the user can click on the breadcrumb provided above the genre chart (figure 6.4). Figure 6.5, representation shows the data in word cloud charts.

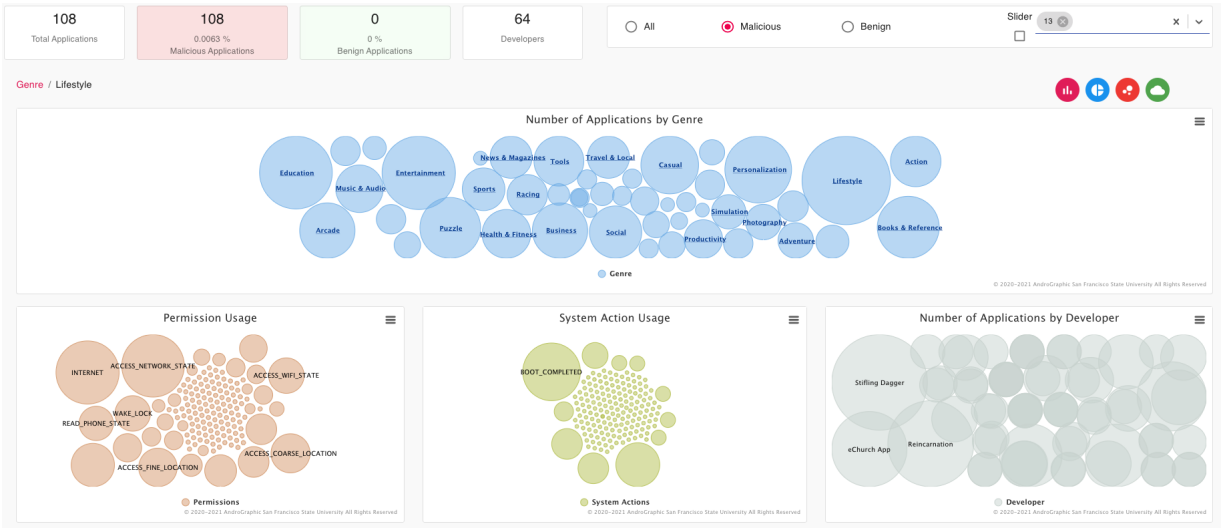

Figure 6.4: Overview of malicious data with index 13

.

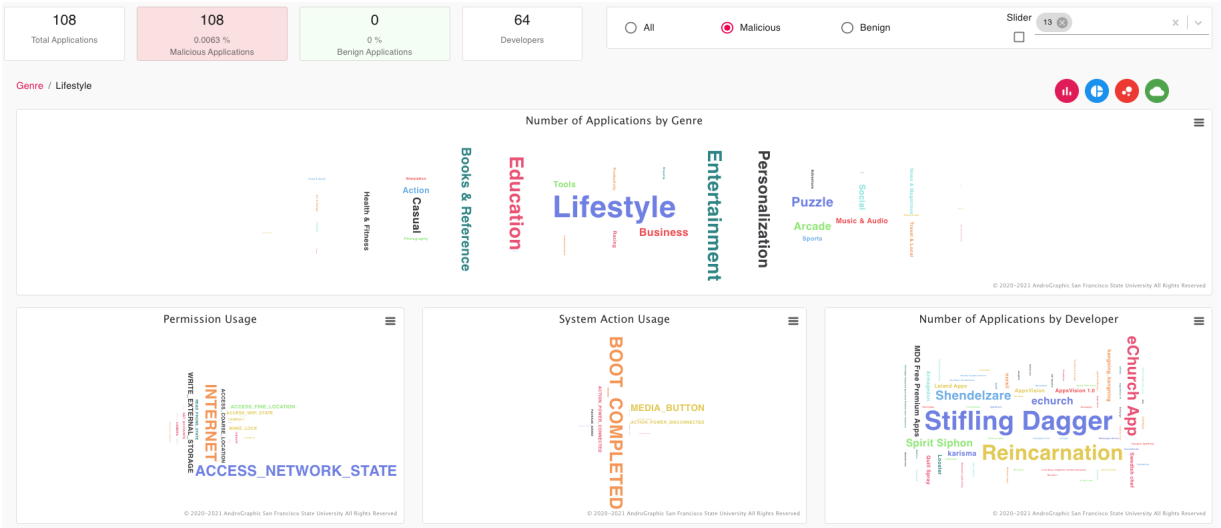

Figure 6.5: Word cloud view for Lifestyle Genre

6.2 Playground

The playground sections allow the user to filter the available data with different measures and visualize different views. This is a build and play section where the user is not bound to any specific view. Users can explore many different combinations by selecting the filters to draw an analysis

between entities. This component allows the user to answer many questions like:

- How many numbers of benign/malicious/total applications are present in X genre let's say the 'Entertainment' genre as compared to other genres?
- How many numbers of benign/malicious/total applications are developed by X developer as compared to other developers?
- What is the permission usage of X, Y, Z permissions for the entire dataset or for the specific genre? Or the dataset filtered by benign/malicious/total applications.
- What is the permission usage by dangerous permission category for the entire dataset or for the specific genre?
- What is the permission usage of permissions of specific protection levels for the entire dataset or for the specific genre?
- What is the permission usage of applications developed by any specific developers?
- What is the system action usage for the entire dataset or for the specific genre? Or the dataset filtered by benign/malicious/total applications.
- What is the system action usage of applications developed by any specific developers?

The landing view of playground is shown in figure 6.6.

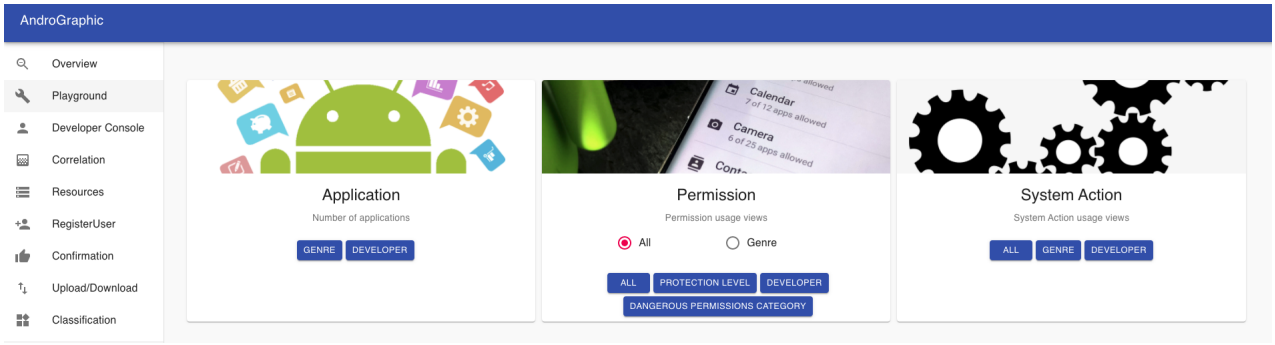

Figure 6.6: Playground

The layout of the playground charts are show in figure 6.7.

1. Dataset filter to populate All, Benign or Malicious data
2. Chart filters e.g. Genre, Permissions, Developer, System Actions, etc.
3. Different chart viewing options (column, pie, bubble, heatmap)

4. Export chart option

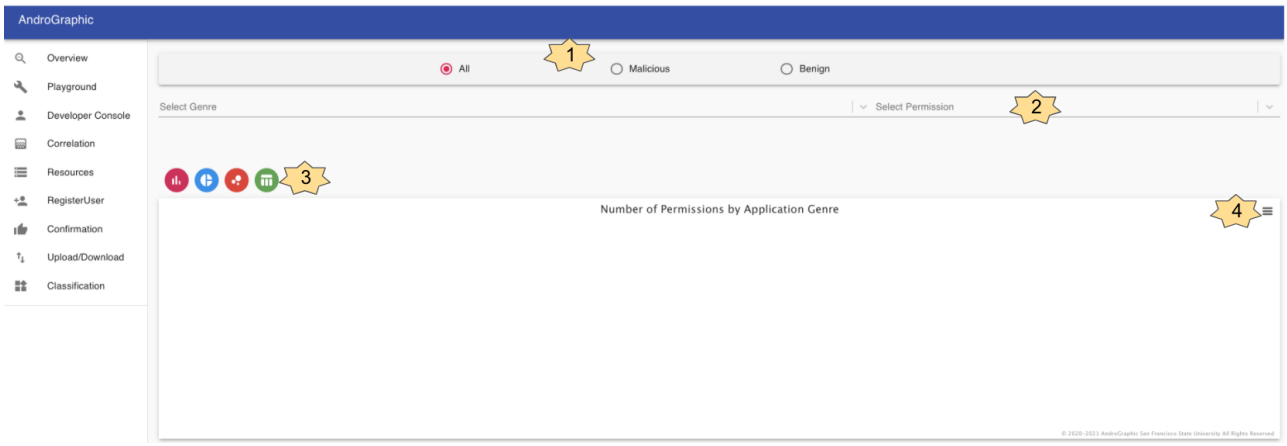

Figure 6.7: Layout of Playground

6.2.1 Application

The first tile (figure 6.6) shows the options concerning applications.

Applications by Genre

The user can get a comparative view by selecting different application genres as shown in figure 6.8. Out of the selected genres, Sports genre have maximum applications and the Comic genre has minimum. The same data has been represented in the pie chart figure 6.9 and bubble chart figure 6.10 views. The user can switch between views by clicking chart icons.

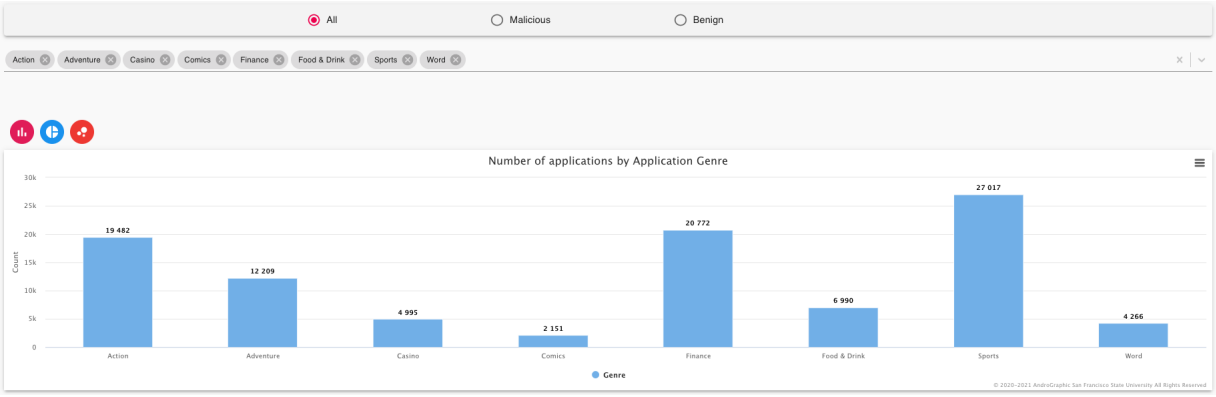

Figure 6.8: Comparative view of number of applications by genre for entire data

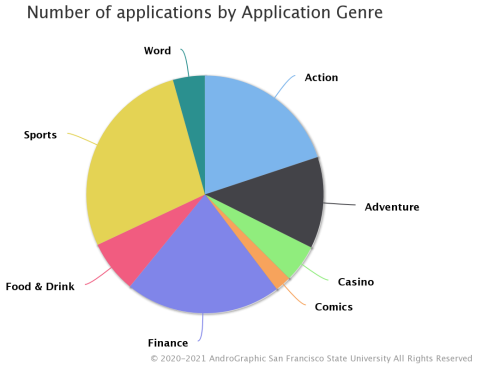

Figure 6.9: Pie chart view

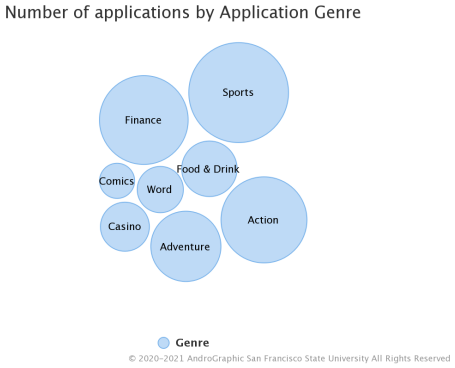

Figure 6.10: Bubble chart view

Applications by Developer

The user can also compare different developers to see the number of applications developed by them. Figure 6.11 shows the comparative view of the developer's app count. The chart shows that Yamaha Corporation has developed a greater number of applications i.e. 7 as compared to other selected developers with respect to the entire dataset.

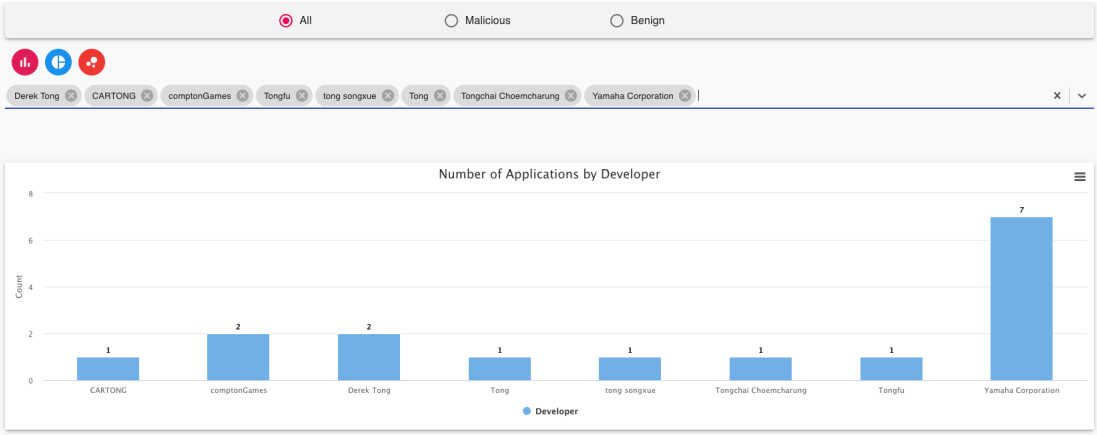

Figure 6.11: Comparative view of number of applications developed by developers

6.2.2 Permission Usage

This option provides the user to explore the permission usage for the entire dataset, by protection level permissions, for dangerous permission categories, or for developers. The user has the flexibility to filter the dataset based on genre as well. We support the following views for permission usage

- Permission usage for an entire dataset where user can select permission name from dropdown

to get the comparative view. (Figure 6.12)

- Permission usage for selected genres where user can apply genre and permission name filter (Figure 6.17, 6.18)
- Permission usage by dangerous permission categories for the entire dataset (Figure 6.13)
- Permission usage by dangerous permission categories for selected genres(Figure 6.15)
- Permission usage by protection level i.e. signature, dangerous and normal for entire dataset. Pie chart view is shown in (Figure 6.16)
- Permission usage by protection level i.e. signature, dangerous and normal for selected genres. (Figure 6.14)
- Permission usage by the developer

A few examples of permission usage charts are shown below.

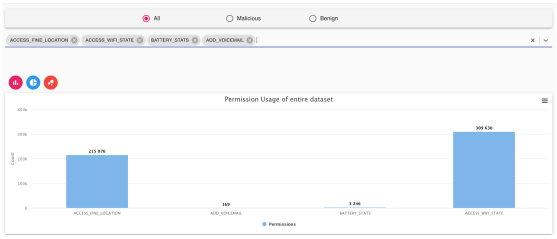

Figure 6.12: Permission usage of entire dataset
This chart shows the usage comparison of four permissions: ACCESS_WIFI_STATE, ADD_VOICEMAIL, BATTERy_STATS, ACCESS_WIFI_STATE

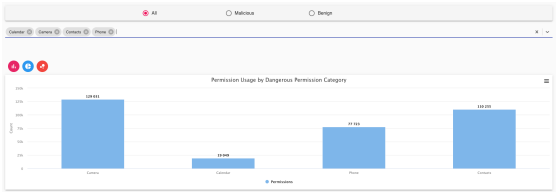

Figure 6.13: Permission usage by dangerous permission categories
It shows that the permission usage of CAMERA category is higher than other dangerous categories

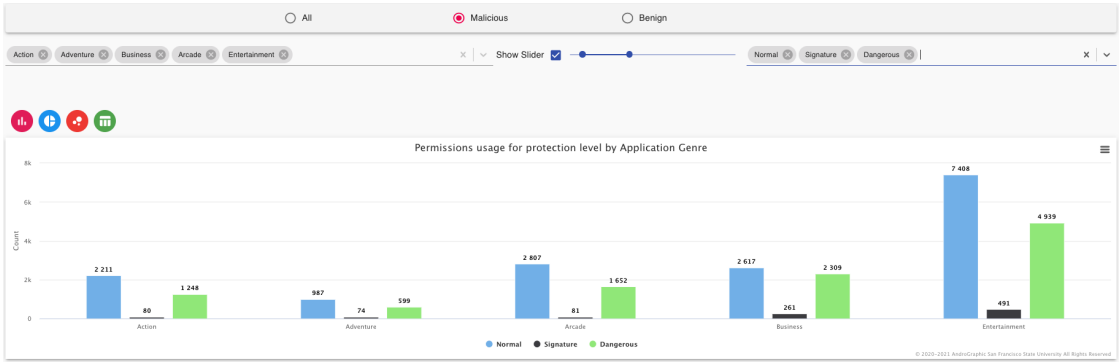

Figure 6.14: Permission usage by protection level having malicious index between 5 to 20

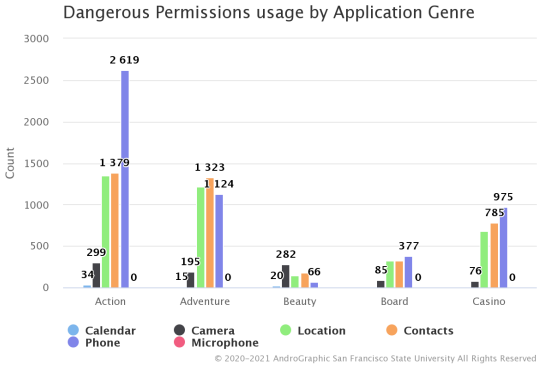

Figure 6.15: Dangerous permission categories for selected genre

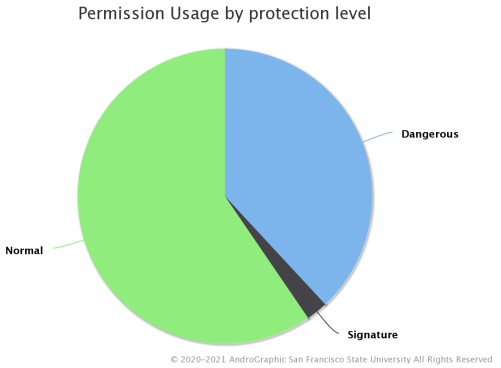

Figure 6.16: Permission usage by protection level

The 6.15 representation shows the group chart where each group represents the column for dangerous permission categories i.e. Calendar, Camera, Location, Contacts, Phone, Microphone for Action, Adventure, Beauty, Board, and Casino genres. The chart will help the user to identify out of the selected genre which one is using maximum dangerous permissions. The pie chart shows the distribution of permission usage across the protection level for the entire dataset 6.16.

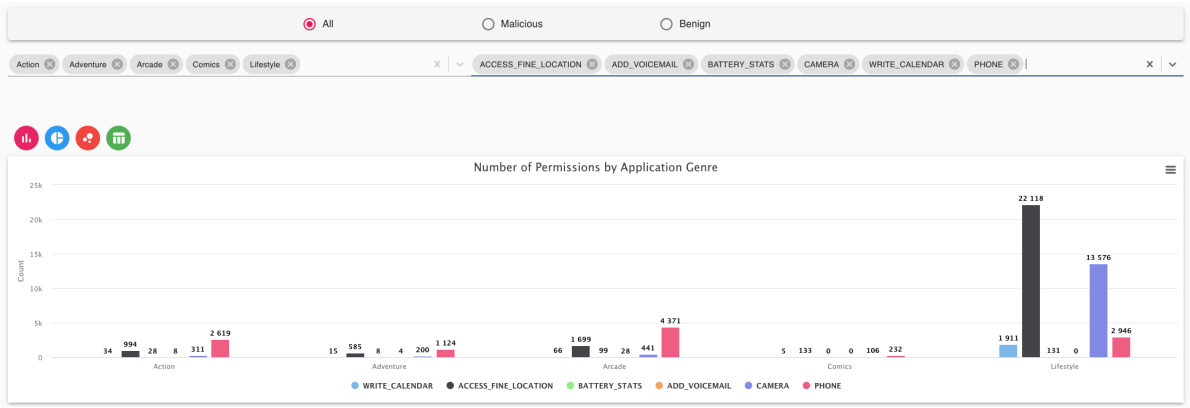

Figure 6.17: Permission usage by genre - Bar chart

This group chart shows the comparatives view of all the selected permissions (BATTERY_STATS, ACCESS_FINE_LOCATION, ADD_VOICEMAIL, CAMERA, WRITE_CALENDAR, PHONE) with respect to selected genres i.e. Action, Adventure, Arcade, Comic, Lifestyle. The heatmap view is shown in a figure 6.18 where user can infer that ACCESS_FINE_LOCATION and CAMERA are widely used in Lifestyle genre.

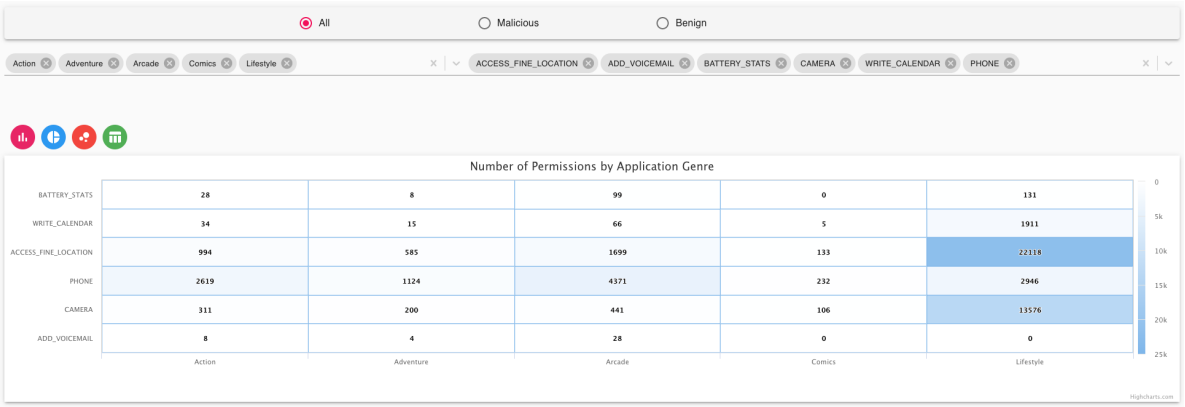

Figure 6.18: Permission usage by genre - Heatmap

6.2.3 System Action Usage

This section discusses the system actions usage tile that provides the following views:

- System action usage for an entire dataset where user can select system action name from the dropdown to get the comparative view. (Figure 6.19)
- System action usage for selected genres where user can apply genre and system action name filter (Figure 6.20, 6.21)
- System action usage by the developer

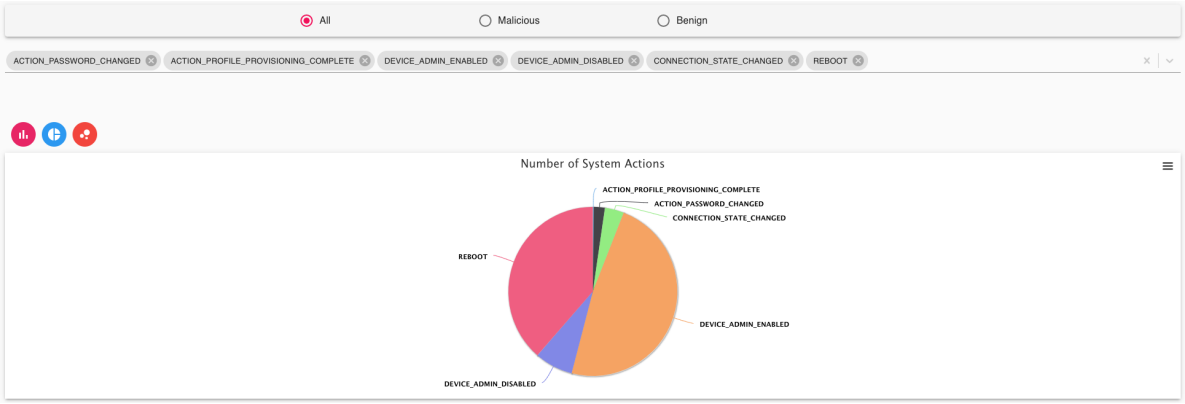

Figure 6.19: System action usage

The pie chart shows the distribution of selected system actions ACTION_PASSWORD_CHANGED, ACTION_PROFILE_PROVISIONING_COMPLETE, DEVICE_ADMIN_ENABLED,

DEVICE_ADMIN_DISABLED, CONNECTION_STATE_CHANGED, REBOOT with respect to the entire dataset as All filter is selected.

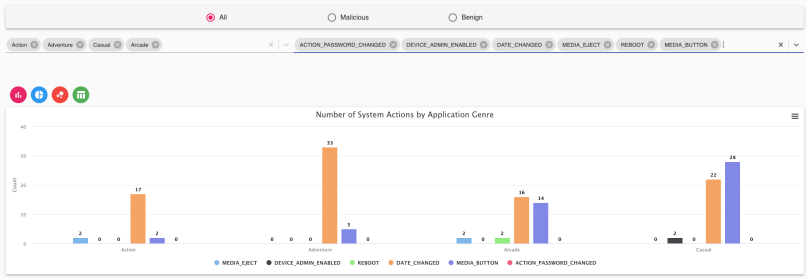

Figure 6.20: System action usage by genre

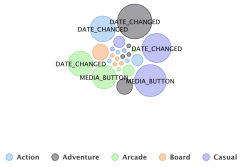

Figure 6.21: System action usage by genre- Bubble chart

The grouped bar chart shows the system action usage for selected genre i.e. Action, Adventure, Casual, Arcade. The user can also get the bubble chart view by clicking on the bubble chart icon provided at the top of the chart. As there are two measures selected, genre, and system actions, the bubble chart is generated to show the system action usage for each genre. Each genre is represented by a different color and the bigger the bubble, the higher the system action usage.

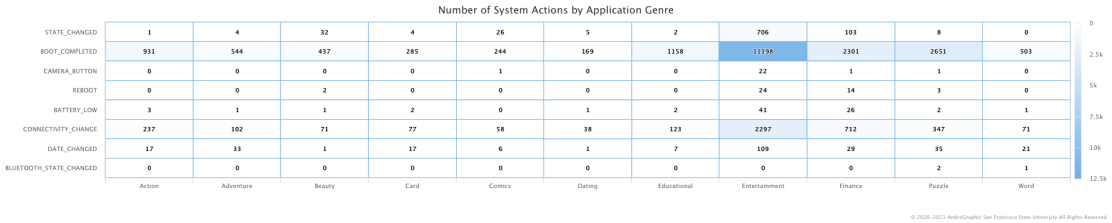

Figure 6.22: System action usage by genre heat map

6.3 Developer Console

The developer is one of the important entities of the dataset. By analyzing the data of any particular developer based on the developed applications can help the user to identify the pattern or similarities. These patterns can help the end user to derive that the developer is using certain combinations, e.g. types of permissions, that may be hazardous and can increase the security risk. With this developer console, the user will be able to get knowledge of the following queries:

- How many numbers of applications are developed by a particular developer?
- What are one, two, three, four, and five start rating developer has been received?
- What type of application developer is creating the most?

- What are the android version supported by the applications?
- Are the applications are related to one or different content types?
- Are the developer making use of dangerous permissions?
- What are the total permission and system action usage per application?

Figure 6.23 shows the detailed view of developer console for Yamaha Motor Co., Ltd.

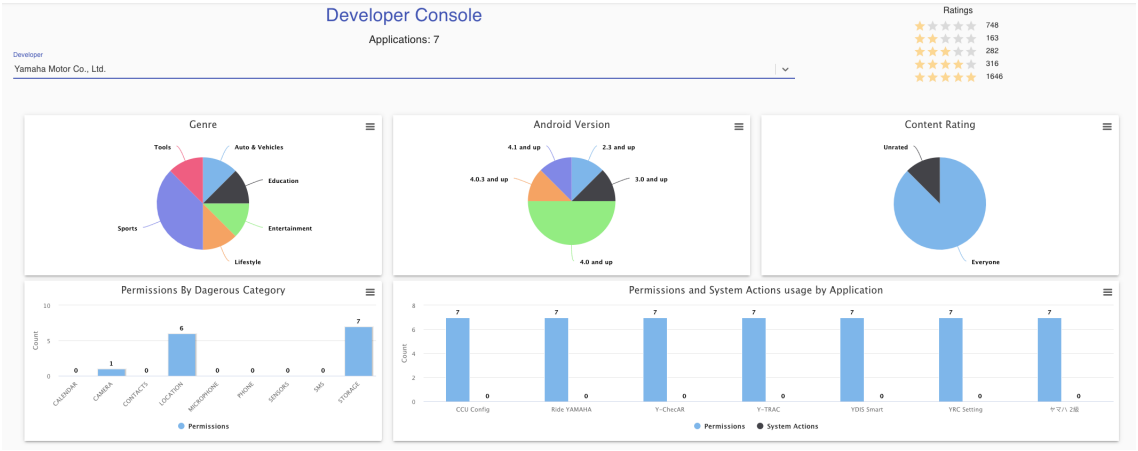

Figure 6.23: Developer Console

6.4 Correlation

This section provides a handle for the user to analyze the correlation between vital measures of the dataset. The module allows user to select permission or system actions to visualize the correlation between permission usage/ system action usage and malicious index. The correlation focuses to show how applications are scattered between these axis. Correlation charts provide the following insights

- Distribution of applications by permission/system action usage and malicious index
- Outlier applications
- Are the application using x permissions/system actions more malicious?
- Are the application using x permissions/system actions are benign?
- Density of applications using a specific count of permissions/actions with malicious index x.
- Distribution of applications w.r.t to genre

As our dataset are huge, there are many applications that overlapped on another. The scatter plot for the permission usage is show in figure 6.24.

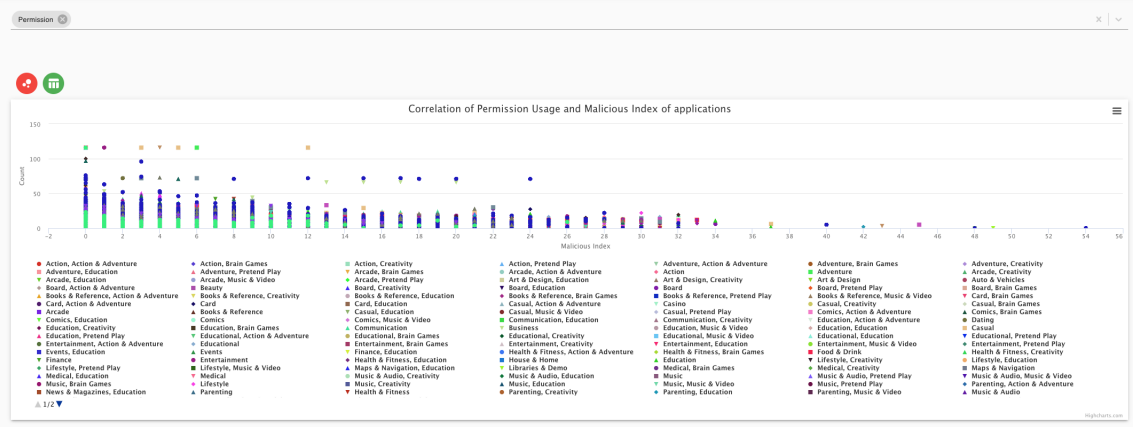

Figure 6.24: Permission usage count and Malicious index scatter plot

As mentioned earlier, data points(applications) overlaps on another so that user cannot view all 1M applications in a glance. the user can visualize the same data in a heat map to check the intensity of the applications on a single point. The heat map for permission usage and malicious index is shown in figure 6.25.

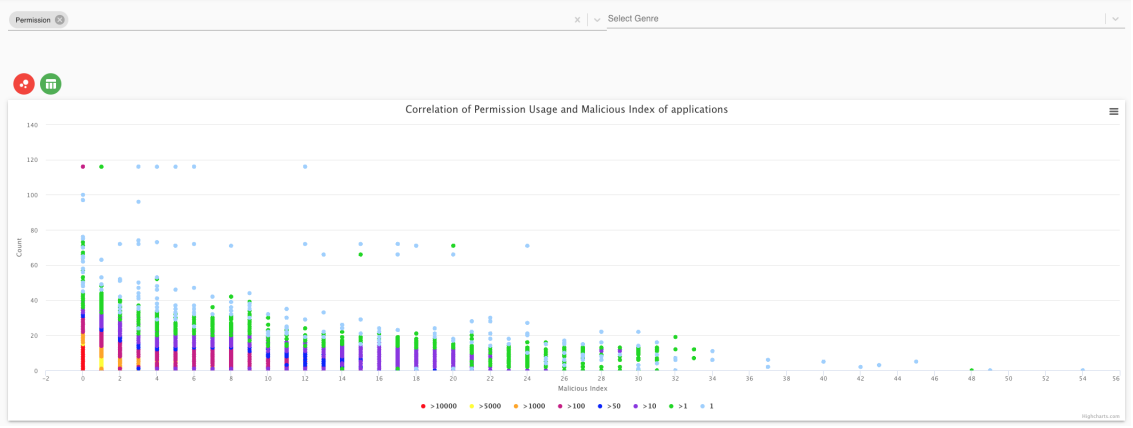

Figure 6.25: Permission usage count and Malicious index heat map

The user can explore the details by drilling down it by genre to get the filter view. For scatter plot, the user can click on the legend and for heat map, user can select the genre from the dropdown. The following charts 6.26, 6.27 show the filtered view for the Lifestyle Genre.

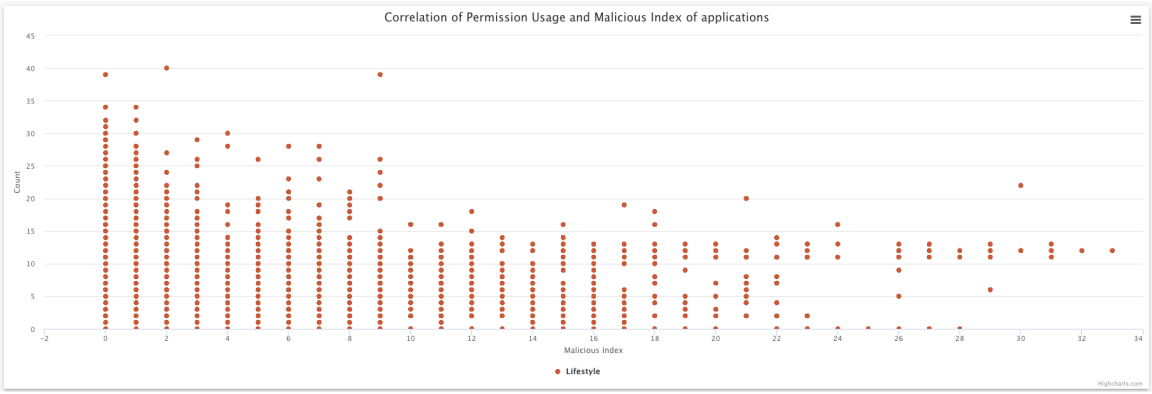

Figure 6.26: Scatter chart of permission usage count and malicious index - Lifestyle Genre

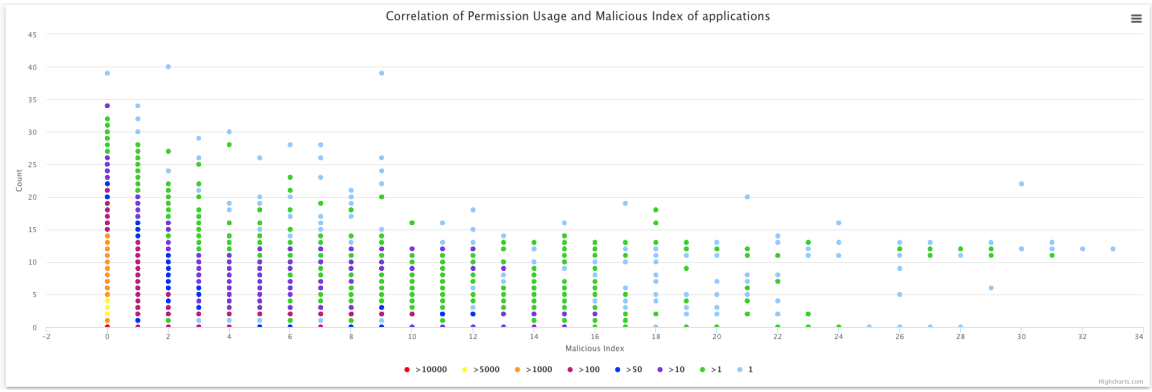

Figure 6.27: Heat map of permission usage count and malicious index - Lifestyle Genre

6.5 Register User

The major motive of the tool is to provide fellow researchers a foundation to visualize their own dataset or to distribute the existing datasets. We only want to allow users those are authentic and are approved by doing some background check. To make so, we provide a portal to register user with email address having edu domain. User need to register for upload and download action separately. When user selects upload, we provide a sample template in csv format and expect the data to be uploaded in the same format. Following figure 6.28, 6.29 shows the register user option with and without template download button.

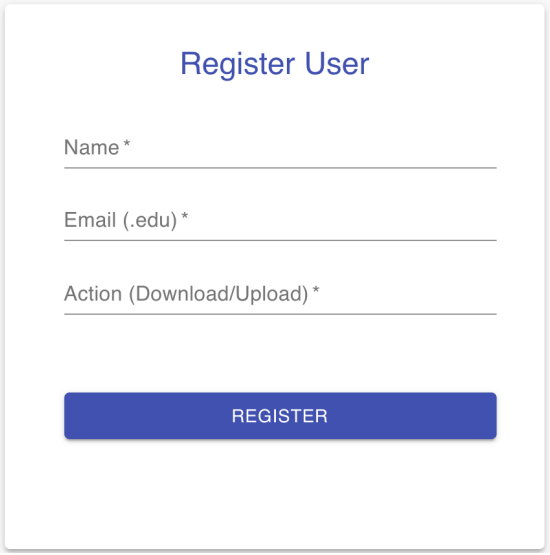

Figure 6.28: Register User

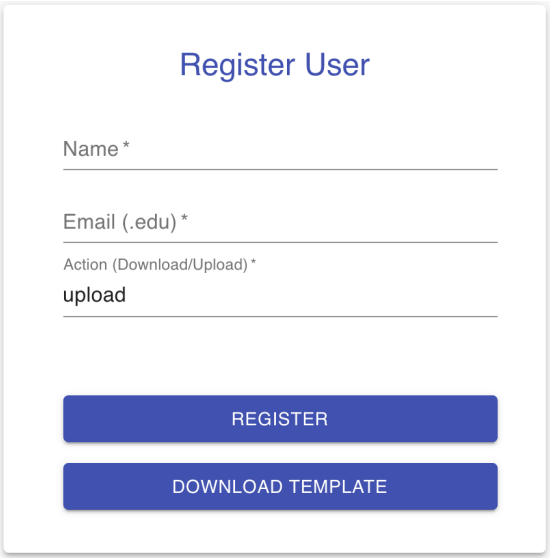

Figure 6.29: Register User with template download option

6.6 Download

The download functionality allows the user to download the existing datasets. We allow other researchers to upload their own dataset as well. We, as a mediator, stores the collection of datasets.

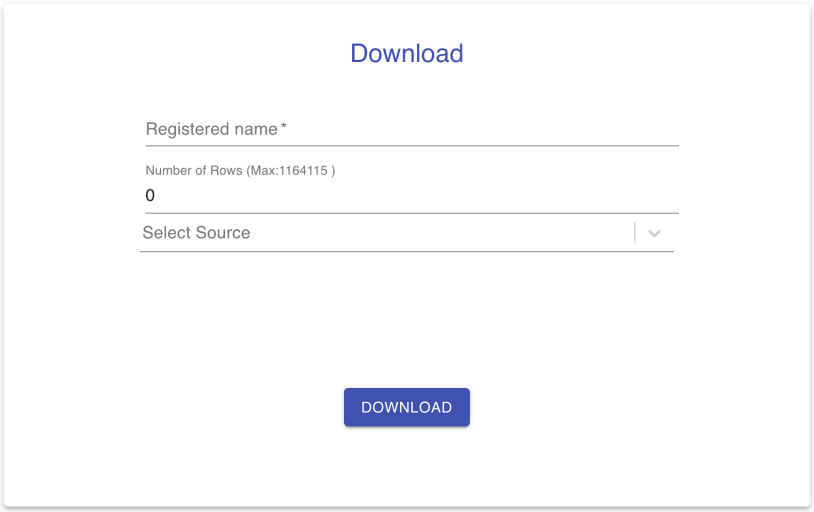

Figure 6.30: Download

The user has been provided with the option to select the source and number of records to download the data. It is mandatory for the user to register and get an approval email from AndoGraphic people to access datasets.

- **Registered name:** The name of the user used while user registration
- **Number of rows:** Number of rows to download
- **Source:** The dataset source. The user can only see resources that are available to download.

6.7 Upload

The upload feature allows user to upload their dataset provided the dataset is in the same format as the sample upload template. User registration for the upload action is mandatory to upload the data. The template provides the column names for contextual and technical data. Contextual data of any app can be obtained from the Google play store. For technical data (permissions and system actions), the user needs to enter either 0 if the given permission or system action is not used and 1 if it is used by the application. The user needs to provide following the information at the time of upload.

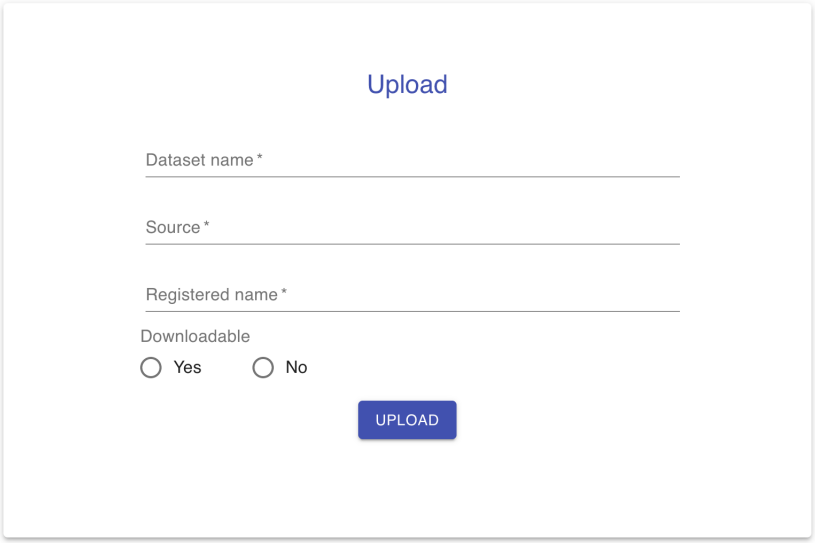

Figure 6.31: Upload

- **Dataset Name:** The name user wants to provide to the dataset
- **Source:** Description of the source if the dataset is not published otherwise URL of the published paper.

- **Registered name:** The name of the user used while user registration
- **Downloadable:** Yes, if the dataset is open to distribute for other researchers, otherwise, No

6.7.1 Flowchart to Upload - Download

The flowchart of steps required to follow to perform upload and download is shown figure 6.32:

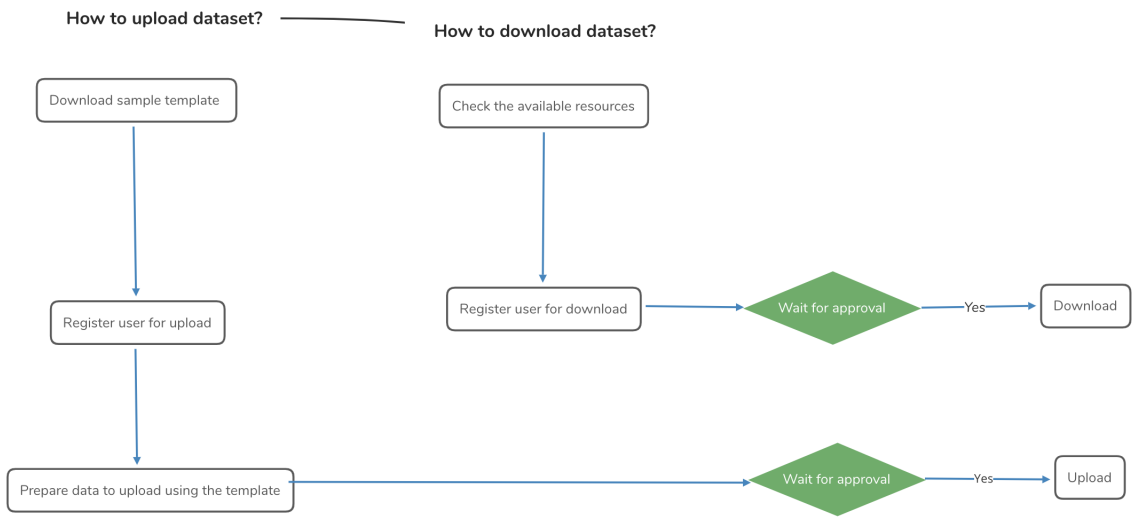

Figure 6.32: Steps to follow to upload - download dataset

6.8 Resources

The resources section shows all the available datasets. The information displayed on the resource page is the name of the dataset, the source where the URL is shown if the dataset is published otherwise description, the last modified date of the dataset, and the indicator to show if the dataset is downloadable or not.

Figure 6.33: Resources

6.9 User Approval

After user registration, the AndroGraphic admin needs to validate the identity of the user to avoid accidental tampering of data. The user can upload or download the data if the user has given access to it. This page is only accessible to AndroGraphic admin for user request approval.

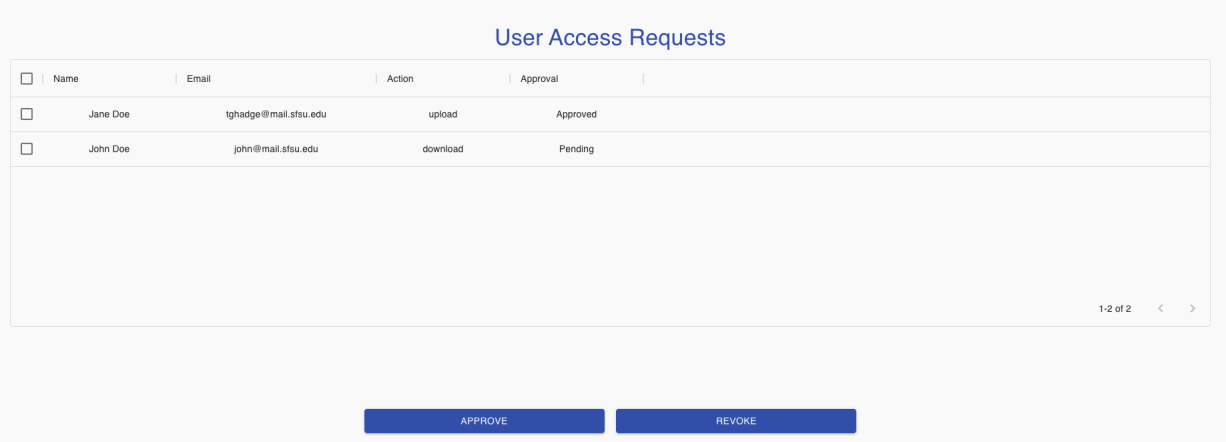

Figure 6.34: User approval

6.10 Classification

The purpose to introduce the machine learning module in AndoGraphic is to find out whether any given application is malicious or not. Later, the model can be refined and can be used to find out a pattern that can cause an application malicious. We provide an initial implementation of classification feature that allows user to check if the application is malicious or not.

6.10.1 Feature selection

Initially, the seed dataset is used to perform the classification. The dataset consists of various numeric and categorical fields. To start the classification process, the data is populated from the database on a python server. The initial step that is carried out on the dataset is prepossessing:

- Missing values are handled by removing all columns having null values
- Type conversion of all columns to numeric
- Label encoding to convert categorical data into a numeric type

After prepossessing, the following **325** features are selected for model training:

- Developer Name

- Genre
- Ratings
- Permissions
- System actions

6.10.2 Model Train

Based on the VT Detection/Malicious index field, we have categorized all applications as malicious or benign. As a result, we only have two classes to label all applications. To start with, we have used **Navie Bayes** algorithm to perform binary classification. The model has been trained using GaussianNB of the scikit learn library. The action to train model is time consuming and it is not very relevant to train model for every application. To overcome this problem, the trained model has been saved using pickle library. On every classification request, the saved model is retrieved and used to label the application.

6.10.3 Classification

As discussed earlier, to perform the classification we need some features of contextual data and some from technical. The purpose of the classification module is to capture this information (contextual and technical) and predict the class of the application. To reduce the user efforts, we have provided an option to upload apk file where we parse the manifest.xml file of the android application and populates permissions and system action using the Androgaurd library. The following shows the sample permissions 6.35 and system actions 6.36 from manifest.xml.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.CLEAR_APP_CACHE"/>
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>
<uses-permission android:name="android.permission.GET_PACKAGE_SIZE"/>
```

Figure 6.35: Permissions in manifest.xml

```
<receiver android:enabled="true" android:exported="true" android:name="aba.hit.aba_pin.receiver.PowerConnectionReceiver">
    <intent-filter>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
    </intent-filter>
</receiver>
```

Figure 6.36: Actions in manifest.xml

The user needs to manually input developer name and genre while making a classification request. The saved-trained model is used to label the class of an application. However, the machine learning module is in the initial phase, the classification result may contain bias result. The classification screen is shown in 6.37

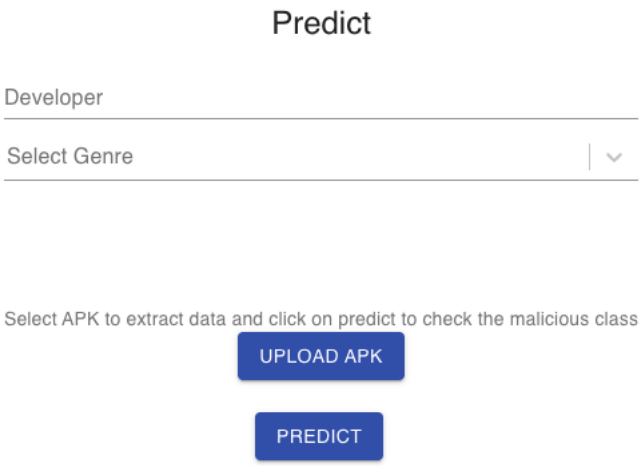

Figure 6.37: Classification

6.10.4 Results

For measuring the results of model training following performance metrics are used:

- **Precision**: This is the ratio of the correctly identified application as malicious/benign from the total malicious/benign applications.
- **Recall**: This is the ratio of the applications labeled as malicious from the actual count of malicious applications.
- **F-1 Score:** This is the harmonic mean of precision and recall.
- **Accuracy:** This is the ratio of the correctly labeled applications to the total number of applications.

It was observed that our model gave the following results as illustrated in the Table 6.1:

Precision	**Recall**	**F-1 Score**	**Accuracy**
46.45%	32.74%	61.17%	86.97%

Table 6.1: Model Performance

It shows that model accuracy is good however, the other metrics precision, recall and f1-score is comparatively very low. The reason behind this can be imbalanced dataset. The malicious data is approximately 1:10 as compared to benign data. To overcome this problem and to get appropriate results there is a need to go through the multiple iterations of data mining process to get refined and accurate feature set.

Chapter 7

Conclusion and Future Work

The researchers need to analyze security aspects of android application by undergoing the repetitive process of data collection due to the lack of publicly available raw data. There is no common platform that helps to quickly analyze the android applications to obtain insights. Thus, to solve these problems, we present a web based tool, AndroGraphic, that helps researchers by providing them ready to use, refined and combined dataset. AndroGraphic acts as a centralized repository where researchers can upload or download datasets. The analytic functionality provided by the tool by thoroughly understanding all vital measures of dataset provides various charts for analysis of malicious and benign data. AndroGraphic also provides an ease for user to export the analysis charts and use in their research.

With the context of future improvements, the classification module to detect the application as malicious or benign indeed is a step towards finding pattern between features that increases the security threat. Thus, we provide AndroGraphic, a platform for our fellow researchers and an attempt to help research community. AndroGraphic web tool URL: `http://3.101.117.225:3000`

In future, considering the possibility of increase in dataset size, we may need to scale the application at some point. We might need to add an intermediate cache to increase the performance of a few charts of the application. The current version of the application only provides the initial version of binary classification. We need to conduct a comparative study between other algorithms like SVM and Random Forest, etc. to correctly identify the algorithm with higher accuracy. The dataset is not balanced as there are more benign applications than malicious. We need to balance the dataset and need to refine the features to get better performance results.

Bibliography

[1] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16)*. ACM, New York, NY, USA, 468–471. https://doi.org/10.1145/2901739.2903508

[2] Vitalii Avdiienko, Konstantin Kuznetsov, Paolo Calciati, Juan Carlos Caiza Román, Alessandra Gorla, and Andreas Zeller. 2016. CALAPPA: a toolchain for mining android applications. In *Proceedings of the International Workshop on App Market Analytics*. 22–25.

[3] DeviceAtlas. 2020. 15 Mobile Web Predictions for 2020. "https://deviceatlas.com/blog/15-mobile-web-predictions-2020".

[4] Franz-Xaver Geiger and Ivano Malavolta. 2018. Datasets of Android Applications: a Literature Review. *CoRR* abs/1809.10069 (2018). arXiv:1809.10069 http://arxiv.org/abs/1809.10069

[5] G. Grano, A. Ciurumelea, S. Panichella, F. Palomba, and H. C. Gall. 2018. Exploring the integration of user feedback in automated testing of Android applications. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 72–83. https://doi.org/10.1109/SANER.2018.8330198

[6] Daniel E. Krutz, Mehdi Mirakhorli, Samuel A. Malachowsky, Andres Ruiz, Jacob Peterson, Andrew Filipski, and Jared Smith. 2015. A Dataset of Open-source Android Applications. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*. IEEE Press, Piscataway, NJ, USA, 522–525. http://dl.acm.org/citation.cfm?id=2820518.2820603

[7] R. Minelli and M. Lanza. 2013. Software Analytics for Mobile Applications–Insights Lessons Learned. In *2013 17th European Conference on Software Maintenance and Reengineering*. 144–153. https://doi.org/10.1109/CSMR.2013.24

[8] The Washington Post. 2019. The Cybersecurity 202: This is the biggest problem with

cybersecurity research. "https://www.washingtonpost.com/news/powerpost/paloma/the-cybersecurity-202/2019/04/18/the-cybersecurity-202-this-is-the-biggest-problem-with-cybersecurity-research/5cb7a231a7a0a46fd9222a47/".

[9] Statistica. 2020. Mobile operating systems' market share worldwide from January 2012 to December 2019. "https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/".

[10] Y. Tang, X. Zhan, H. Zhou, X. Luo, Z. Xu, Y. Zhou, and Q. Yan. 2019. Demystifying Application Performance Management Libraries for Android. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 682–685.

[11] Fengguo Wei, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. 2017. Deep Ground Truth Analysis of Current Android Malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Michalis Polychronakis and Michael Meier (Eds.). Springer International Publishing, Cham, 252–276.

[12] Suleiman Y Yerima, Sakir Sezer, and Igor Muttik. 2015. High accuracy android malware detection using ensemble learning. *IET Information Security* 9, 6 (2015), 313–320.