Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 3 (2023)

Enhancing Android Framework Used to Detect Unexpected Permission Authorization of Mobile Application

Manisha Patil[1] and Dhanya Pramod[2]

[1]Faculty of Computer Studies, Symbiosis International (Deemed University) (SIU), Lavale, Pune, 412115, Maharashtra, India. Indira College of Commerce and Science.

[2]Symbiosis Centre for Information Technology (SCIT), Symbiosis International (Deemed University), Hinjewadi, Pune, 411057, Maharashtra, India

Abstract:

The use of mobile devices is expanding daily in today's technological age. Mobile marketplaces are constantly providing an expanding selection of mobile applications to satisfy the demands of smartphone users. Many Android applications fall short in their attempts to adequately address security-related issues. This is usually brought on by a lack of automated methods for permission-based vulnerability discovery, testing, and resolution during early design and development phases. As a result, it is generally agreed that addressing such issues quickly is preferable to sending updates and fixes for already-released apps.A proactive set of permissions declared by mobile app developers can protect users' data privacy is the research concern here. This paper reviews the AndRev-Android framework functionality and its envisioned purpose. The researcher tried to justify the research questions raised by the experts, which will add value to future researchers in this domain.

Keywords: *Privacy, Mobile apps, Static Analysis, Permission, Android APK.*

1. Introduction:

Mobile devices contain privacy-related data since they are kept close to the user, are more likely to be lost or stolen, and can connect to multiple networks at once [1]. As a result, confining mobile devices to existing security measures will not be sufficient. A few of the dangers [3] that can happen in an Android mobile device are malicious behaviors including physically accessing the device, establishing connections with untrusted networks, installing, and running untrusted programs, and executing untested code blocks and contents. Therefore, security precautions should be tightened while reducing vulnerabilities to protect the data on Android devices.

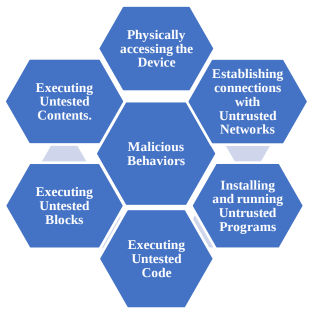
Physically accessing the Device
Establishing connections with Untrusted Networks
Installing and running Untrusted Programs
Executing Untested Code
Executing Untested Blocks
Executing Untested Contents.
Malicious Behaviors

Figure 1.1 Types of Malicious Behaviour for Mobile Apps

It is necessary to install adequate security measures to increase the security of Android devices and their applications. Therefore, identifying Android applications' weaknesses and applying the security best practices, it is advisable to perform permission-based vulnerability discovery.

The research questions addressed are as follows:

Research Questions:

RQ 1**:**What would be the significance of this research considering the upcoming modifications as

Google Play has consistently revised its policies[11]?

RQ 2: What is the rationale behind the sampling?

RQ 3: What is the perspective for permission Analysis?

RQ 4: How does PCA improve model accuracies via accurate feature selection?

RQ 5: What are the added functionalities in the AndRev tool over APKTool?

RQ 1: What would be the significance of this research considering the upcoming modifications as

Google Play has consistently revised its policies[11]?

Even though Android mobile OS and its permission model both are secured, it is observed that there is an increase in malware for Android mobile apps and user’s data privacy is always at risk. Numerous apps in the Apple App Store for iOS and the Google Play Store for Android expose consumers to privacy dangers, according to Symantec research[4]. Some mobile applications for iOS and Android users ask for excessive permissions to access a user's personal data. In the study, Symantec detailed how users' personal information is collected and how dubious apps on the Google Play Store are overrun with intrusive adverts, endangering their privacy.

RQ 2: What is the rationale behind the sampling?

Is there a Sampling Bias when only looking at the Top50 Free Apps in 5 Categories? By limiting your search to this sample, will you be missing anything?

A mobile user visits the Google Play Store with the aim of getting TOP FREE APPS. TOP FREE APPS is a focused dataset for this research. Here on this page user gets a few recommended apps as per the requested category or functionality by the app user. As per the Chi-square test for two variable independence and Z-test of proportion with a p-value less than 0.05, we have considered the TOP 50 FREE Apps across the 5 Categories, in all 250 apps. Apps are listed for users to download or install from the Google Play Store website. Secondly, while collecting features of apps across the categories with similar functionality, it was found that features are almost getting repeated.

Google Store is the main data source to get free Android APKs required for this research study. There is no other way for the normal user to validate the user’s privacy and app security, so the user is simply dependent on the review system provided by Google Play Store which then becomes a trusted data source for the user to download apps. Various categories of apps are available which affect the distribution of permissions but for this study categories of apps like – Games, Education, Entertainment, Shopping, and General/Tools are considered. 50 top apps from each category from the mentioned data source were selected and downloaded for this experiment. As a result, 250 free Apps, which is a good sampling of Apps. In this research, a Z-test is conducted to decide the sample size. The two variables proportion test is conducted for sample sizes of 50, 100, 150, 200, 250, 300, and 1000 Android APKs. Z-test results are significant and there is not much variation observed for sample sizes 200, 250, 300, and 1000 samples of Android APKS. The calculated P value for the Z-test is less than 0.05. In this research, permissions are divided into the categoriesof normal, medium, and high levels according to their impact on 44 making the app safe or vulnerable. A statistical chi-square test for variable independence is carried out to check application safety is dependent on the type of permissions declared by the app, this test is accepted. The next step is to reverse-engineer the APKs to extract permissions. Drebin dataset, (Arp, 2014), (Spreitzenbarth, 2013), provides text files with the list of features and permissions but, APK files are not provided. The dataset does not provide information on whether the app is safe or not safe. Therefore, there is a need to build a research dataset with APKs.

RQ 3: What is the perspective for permission Analysis?

Install-time permissions, runtime permissions, and special permissions are the three main categories under which Android divides permissions. When the system grants your app that permission, each category represents the range of restricted data that the app can access and the range of restricted actions that the app can carry out[10].

The research approach primarily focuses on static analysis of Android mobile apps, the researcher has extracted a set of permissions that are declared in the manifest.xml file of each mobile app. Permissions declared in the manifest.xml file are installed time permissions. Primarily discussion is about permissions declared by the app user and compared to the requirement of permissions to fulfill the functionality of the app to satisfy the user expectations. A proactive set of permissions declared by mobile app developers can protect users' data privacy is the research concern here.

RQ 4:How does PCA improve model accuracies via accurate feature selection?

Comparison between the performance of various classifiers before and after PCA

Table 1.1: Performance of Learning Models with all features (before applying PCA)

Performance of Learning Models with all features (before applying PCA)							
Type	Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes	Naive Bays	64	41.9	59.6	64	59.8	74.5
Functions	Logistic Regression	65	33.1	69.5	65	58.9	82.7
	Multilayer Perceptron	64	25.5	65	64	64.3	77.2
	Simple Logistic Regression	70	22.5	71.4	70	69.7	79.5
	SMO	65	61.1	55.3	65	54.9	55.6
Meta	Logit Boost	62	36.8	59.2	62	60.1	73.3
	Bagging	66	43.1	64.5	66	62.5	70.8
Tree	Random Forest	70	37.8	69	70	67.7	79
	J48	63	34.1	62.2	63	62.4	63.2
Rules	Decision Table	59	62.7	37.7	59	46	45.6

Table 1.2: Performance of Learning Models with all features (after applying PCA)

Performance of Learning Models with reduced features (after applying PCA)							
Type	Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes	Naive Bays	67.5	23.6	69.5	67.5	68.3	73
Functions	Logistic Regression	65	33.1	69.5	65	58.9	82.7
	Multilayer Perceptron	70	27.4	69.3	70	67.9	78.9
	Simple Logistic Regression	56	53.7	47.8	56	50	66.5
	SMO	65	33.1	69.5	65	58.9	69.9
Meta	LogitBoost	65	33.1	69.5	65	58.9	81.4
	Bagging	62.5	44.7	57.8	62.5	58.5	69.8
Tree	Random Forest	74	31.4	72.9	74	71.8	84
	J48	69	24.4	69.6	69	68.8	78.2
Rules	Decision Table	57	38.3	54.1	57	55.3	63.5

A few more observations during the comparison between the performance of various models before and after applying principle component analysis (PCA) are –

Bagging, Simple Logistic Regression, and Logistic Regression results are not significantly affected after applying PCA, as shown in Table 1.2 and Figure 1.2.

Precision, F-measure, and ROC values are higher for the Decision Table classifier-based model but a major change in the value of the False positive detection rate, value is drastically decreased, as shown in Table 1.1 and Figure 1.1.

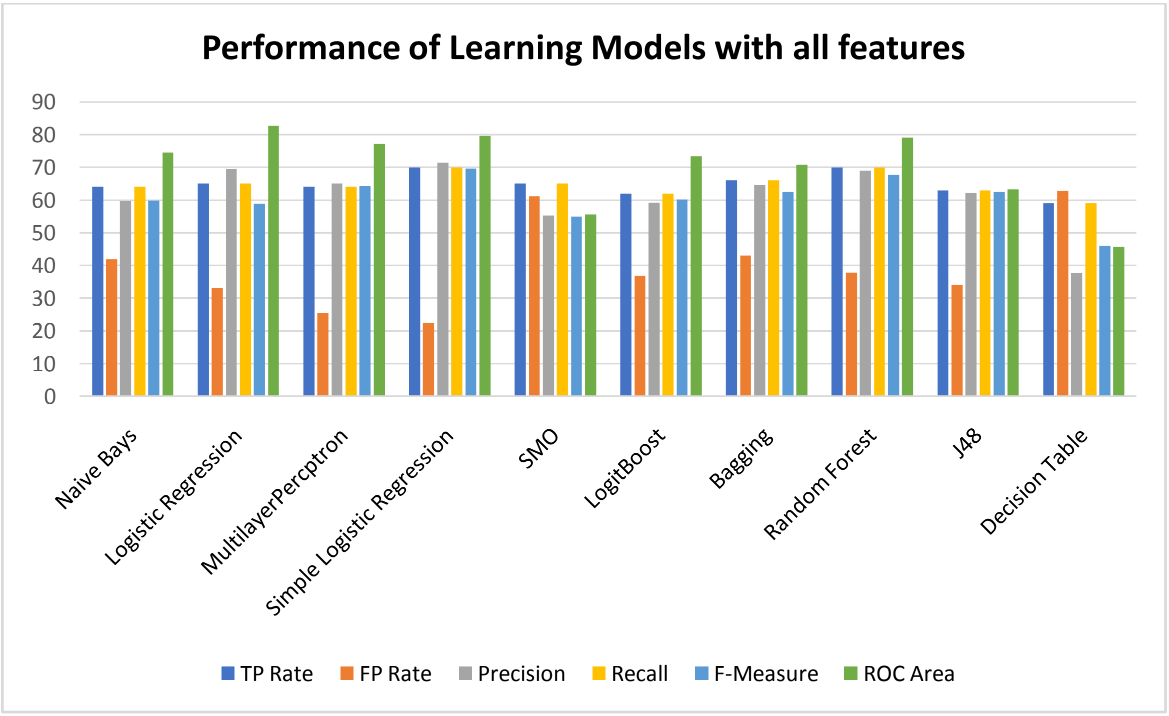

Figure 1.1: Performance of Learning Models with all features (before applying PCA)

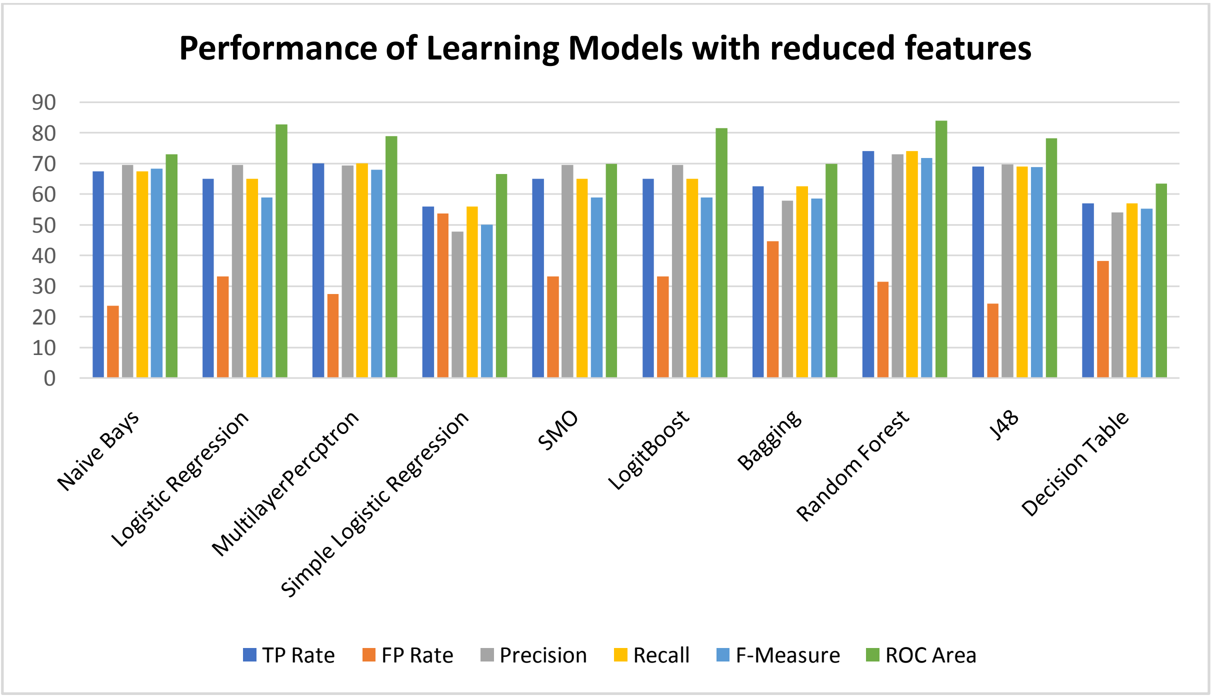

Figure 1.2: Performance of Learning Models with all features (after applying PCA)

RQ 5: What are the added functionalities in the AndRev tool over APKTool?

The existing ApkTool is invoked by the AndRev tool, which was developed as part of this research. ApkTool has the restriction of only being able to decompile one APK file at a time.

ApkTool had a limitation of decompiling only one APK file at one run. So,the researcher decided to design a batch-scripted tool AndRev, which helped to decompile multiple files at one run. In addition to the decompilation process, the designed tool also marks 1 for present permission and 0 for the case if permission is not used by a specific app. Finally, the AndRev tool generates a .CSV file with multiple unique permission sets for various apps decompiled at one run. Hence drawback of APKTool was overcome.

2. Literature Review:

Java or Kotlin can be used to create native Android applications, and Java is the most popular language for this. Android mobile applications can also be created using frameworks like Xamarin and React Native [7]. However, in addition to application files, these mobile apps also include Extensible Markup Language (XML) files for the Android Manifest, UI layouts, and resources. Consequently, it is necessary to find the problems in both XML and source code files. Both files can be examined using static analysis without being run. Manifest analysis and code analysis are the two static analysis techniques. The extraction of features is where these two approaches diverge. Some studies employ manifest analysis, some employ code analysis, and a select few employ both [13].A prominent static analysis technique is the manifest analysis. From the AndroidManifest.xml file, it may extract package names, permissions, activities, services, intents, and providers. All the permissions utilized in a certain program are listed in the Android Manifest file and are divided into three categories: risky, signature, and normal. By creating a three-level data cleaning approach, 22 permissions that were designated as significant permissions in SigPID in Reference [17] were found. These three tiers were permission ranking with a negative rate, permission mining with association rules, and support-based permission ranking. Code analysis, which considers source code files, is the second form of static analysis. Code analysis can be used to extract features including API calls, information flow, taint tracking, native code, clear-text analysis, and opcodes. The MaMaDroid [6] technique serves as an illustration for the examination of API calls. It used static code analysis techniques to abstract API call executions from programs to produce ordinary classes or packages, and the Markov chain was used to identify the call graph.

Five areas of static analysis were suggested by the authors in Reference [5]: analysis techniques, sensitivity analysis, code representation, data structures, and inspection level. Symbolic execution, taint analysis, program slicing, abstract interpretation, type checking, and code instrumentation are the analysis methodologies. For sensitivity analysis, objects, contexts, fields, pathways, and flows are taken into account.For code representation, Smali [8], Dex-Assembler [15], Jimple [14], Wala-IR [16], and Java Byte code/Java class are employed, whereas data structures Call Graph, Control Flow Graph, and Inter-Procedural Control Flow Graph. The inspection levels take into consideration kernels, programs, and emulators.

3. Methodology:

The use of machine learning (ML) approaches for finding vulnerabilities has increased recently [9]. Therefore, to properly appreciate ML-based source code vulnerability detection investigations, understanding ML processes is helpful.

The machine learning (ML) lifecycle involves the following steps: data extraction, pre-processing, feature selection, model training, assessment, and deployment [12]. ML includes supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and deep learning. To solve problems with classification and regression using a labelled dataset, the model is trained using supervised learning. Naive Bayes (NB), Logistic Regression (LR), Linear Regression, Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and k-nearest Neighbours (kNN) are a few of the algorithms that can be employed for supervised learning. Unsupervised learning discovers latent patterns in data by using grouping, association, and dimensionality reduction. Without a dataset with tags, the model can be trained. K-means clustering, PCA, and autoencoders are some of the unsupervised learning methods that can be applied. When there are few labels in the dataset being used, semi-supervised learning is used, which mixes supervised and unsupervised learning techniques. Without training data, reinforcement learning involves modifying the model's parameters based on input from the outside world. This machine learning method works in cycles of assessment and prediction. DL is defined as learning and improving by independently analyzing algorithms and comprises more or deeper processing layers. Convolutional Neural Network (CNN), Long Short Term Memory Network (LSTM), Recurrent Neural Network (RNN), Generative Adversarial Network (GAN), and Multilayer Perceptron (MLP) are a few of the well-known DL algorithms [2] .

4. Conclusion:

The study investigated the methods for detecting unexpected permission authorization of mobile applications based on attack goals, including server, network, client software, client hardware, and user. The researcher confirmed a number of viewpoints, and the findings will aid future research. In this study, free mobile apps were analyzed for unexpected permission authorization. In the past several years, authors have analyzed these issues, but in this work, a tool to extract features from the apk file was designed and stored in a .csv file to ease analysis. Feature extraction tools can be used by data scientists for conducting future research.

Data availability statement

Dataset 1 : The data that support the findings of this study are openly available at https://www.sec.cs.tu-bs.de/~danarp/drebin/download.html

Daniel Arp, Michael Spreitzenbarth, Malte Huebner, Hugo Gascon, and Konrad Rieck "Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket", 21th Annual Network and Distributed System Security Symposium (NDSS), February 2014

Michael Spreitzenbarth, Florian Echtler, Thomas Schreck, Felix C. Freling, Johannes Hoffmann, "MobileSandbox: Looking Deeper into Android Applications", 28th International ACM Symposium on Applied Computing (SAC), March 2013

References:

[1] A.V. Mbaziira, J. Diaz-Gonzales, and M. Liu, “Deep learning in detection of mobile malware,” Journal of Computing Sciences in Colleges, vol. 36, no. 3, pp. 80–88, 2020.

[2] B. Sim and D. Han, “A study on the side-channel analysis trends for application to IoT devices,” J. Internet Serv. Inf. Secur, vol. 10, pp. 2–21, 2020.

[3] C. Li et al., “Android malware detection based on factorization machine,” IEEE Access, vol. 7, pp. 184008–184019, 2019.

[4] Desk, T. Many Android, iOS apps put user privacy at risk: Symantec research. The Indian Express. https://indianexpress.com/article/technology/tech-news-technology/many-android-ios-apps-put-user-privacy-at-risk-symantec-research-5313358/, 2020.

[5] G. Kirubavathi and R. Anitha, “Structural analysis and detection of android botnets using machine learning techniques,” International Journal of Information Security, vol. 17, pp. 153–167, 2018.

[6] H.D. Trinh and Angel Fernandez Gambin, “Mobile Traffic Classification through Physical Control Channel Fingerprinting: a Deep Learning Approach,” IEEE Transactions on Network and Service Management, vol. 18, 2020.

[7] J.M. Anderson, “Why we need a new definition of information security,” Computers & Security, vol. 22, no. 4, pp. 308–313, 2003.

[8] J. Zhang, Zheng Qin, and Kehuan Zhang, “Dalvik opcodes graph based android malware variants detection using global topology features,” IEEE Access, vol. 6, pp. 51964–51974, 2018.

[9] M. Yang, Xingshu Chen, and Yonggang Luo, “An Android Malware Detection Model Based on DT-SVM,” Security and Communication Networks, vol. 2020, Article ID 8841233, 11 pages, 2020.

[10] Permissions on Android. (n.d.). Android Developers. https://developer.android.com/guide/topics/permissions/overview

[11] Provide information for Google Play's Data safety section - Play Console Help. (n.d.). https://support.google.com/googleplay/android-developer/answer/10787469?hl=en

[12] R. Langner, "Stunt: Dissecting a cyberwarfare weapon," IEEE Security & Privacy, vol. 9, no. 3, pp. 49–51, 2011.

[13] S. Garg and N. Baliyan, "A novel parallel classifier scheme for vulnerability detection in android," Computers \& Electrical Engineering, vol. 77, pp. 12–26, 2019.

[14] S. Wei, Zedong Zhang, and Shasha Li, "Calibrating Network Traffic with One-Dimensional Convolutional Neural Network with Autoencoder and Independent Recurrent Neural Network for Mobile Malware Detection," Security and Communication Networks, vol. 2021, Article ID 6695858, 10 pages, 2021.

[15] T. Kim et al., "A multimodal deep learning method for android malware detection using various features," IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773–788, 2018.

[16] X. Jiang, Security alert: new droid kungfu variant–AGAIN!– Found in Alternative Android Markets, 2011.

[17] X. Wang, Wei Wang, and Yongzhong He, "Characterizing Android apps' behavior for effective detection of malapps at large scale," Future generation computer systems, vol. 75, pp. 30–45, 2017.