Hindawi
Security and Communication Networks
Volume 2018, Article ID 2329891, 12 pages
https://doi.org/10.1155/2018/2329891

Research Article

An Improved Permission Management Scheme of Android Application Based on Machine Learning

Shaozhang Niu,[1] **Ruqiang Huang**,[1] **Wenbo Chen**,[1] **and Yiming Xue**[2]

[1]*Beijing Key Lab of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China*

Correspondence should be addressed to Shaozhang Niu; szniu@bupt.edu.cn

Received 26 June 2018; Revised 25 September 2018; Accepted 2 October 2018; Published 18 October 2018

Guest Editor: Zhaoqing Pan

The Android permission mechanism prevents malicious application from accessing the mobile multimedia data and invoking the sensitive API. However, there are still lots of deficiencies in the current permission management, which results in the permission mechanism being unable to protect users' private data properly. In this paper, a dynamic management scheme of Android permission based on machine learning is proposed to solve the problem of the existing permission mechanism. In order to accomplish the dynamic management, the proposed scheme maintains a dynamic permission management database which records the state of permissions for each application. Only the permission which is granted state in the database can be used in this application. In the whole process, the scheme first classifies the application by means of machine learning, then retrieves the corresponding permission information from databases, and issues the dangerous permission warning to users. Finally, the scheme updates the dynamic management database according to the users' decisions. Through this scheme, users can prevent malicious behaviour of accessing private data and invoking sensitive API in time. The solution increases the flexibility of permission management and improves the security and reliability of multimedia data in Android devices.

1. Introduction

While smart devices bring us a lot of convenience, they also become the attractive targets of cyberattacks [1]. Multimedia data in mobile device features as large storage and speedy transmit of high-definition data, which increases its popularity among big data environment [2]. However, its security and privacy becomes a growing serious problem. Android ensures that all applications get access to the privacy data (such as contact list, photo album, and other private data) in a reasonable situation by means of requiring that applications should declare permissions in the configuration file. Thus, permission management is the straightest and most fundamental method to protect user privacy data. Permission mechanism plays a critical role in controlling access to the resources in the device.

However, there are still a number of problems in the permission mechanism: Android does not support dynamic configuration and customized management of the permissions. In addition, most developers of Apps do not follow "Least Privilege Principle" when they create function for their products. Felt AP et al. analysed nearly 1, 000 applications and found that more than 1/3 of all applications have declared unnecessary permissions [3]. In other word, more than 300 applications may use these unnecessary permissions to extract private data. Thus, it is so difficult to implement the security of Android system with current permission mechanism.

Furthermore, with the constant appearances of Android system vulnerabilities, there arise a large amount of attack behaviours and malwares. Many researchers pay attention to the malware detection, such as Jaewoo S who proposed an approach to distinguish the malware in Android Unity [4] and Huanran W who introduced object reference graph in malware detection [5]. Many attackers are used to exploiting the deficiencies of permission security mechanism to implement their illegal purpose. This illegal behaviour produces serious threat to user's privacy data such as obtaining user's privacy data and running Trojan program.

Because of the openness of Android system, the research on its security has never been interrupted. These researches help us to implement the effective protection to the private data. Furthermore, there are various researches on the mechanism of Android permission. Zhang Y et al. declared one dynamic analysis platform, named VetDroid [6]. This tool constructs the correct permission usage specification to prevent the malicious behaviour in malware. Pearce P et al. introduced a privilege separation framework upon the mobile advertisements [7]. This framework creates a new advertisement API, which separates the application and the advertisement. However, all these researches do not give user the ability to decide which permission can be granted. It is so limited in implementing the permission management.

Aiming at the drawback of Android permission mechanism, this paper puts forward a kind of permission management scheme based on machine learning. This research makes use of the application category and permission-related data to implement the application classification training and testing. The application will adopt specific permission white list corresponding to the classification result. This scheme helps users to manage permissions more actively, and furthermore, improves the flexibility of original permission mechanism and the security of Android OS. User is allowed to manage the permission properly to prevent risk behaviors such as accessing private data.

2. Android System Permission Mechanism

This section analyses the existing permission mechanism of Android and summarizes its defects. In order to refine the design and implementation of the permission management scheme, this section analyses the usage situation of the Android application permission.

2.1. Drawback of Existing Permission Mechanism. Android system adopts permission checking mechanism which protects Android devices and user's private data effectively. Meanwhile, there are also many defects shown in the following aspects.

Firstly, when user installs the application, they either grant the required permissions or reject them all. If you reject it, the installation process will be failed, which named "ALL-or-NONE" pattern [8]. It is obvious that this pattern makes an impact on the security of private data. If user sticks to install the application, he has no choice except to grant all permissions, although it is risky operation.

The second drawback is that as long as the application installation is complete, the granted permissions can no longer be modified. Sometimes, it is too difficult to be aware of the malicious behavior in the background for users. When they find out the problem, they have no idea to prevent this illegal behavior by modifying the permission list. In other word, if the multimedia data is accessed illegally, users cannot prevent this event happened unless uninstalling the application.

Thirdly, the permission mechanism lacks the sufficient protection, especially in the native layer, For example, when malware use JNI method to operate the Native layer of the system and obtain the system services, which would lead to worse situation.

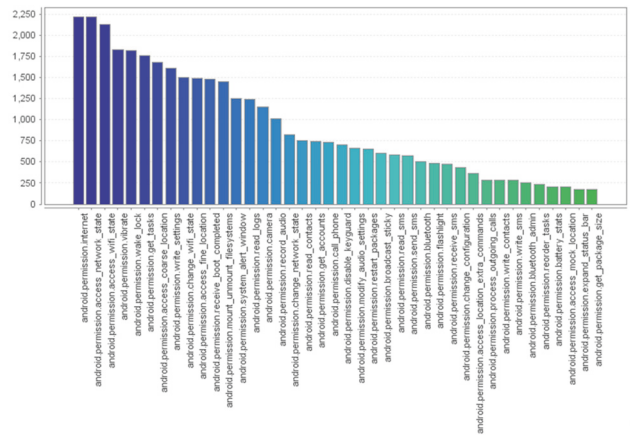


FIGURE 1: The permission statistic result. The column chart shows the first forty permissions. Nearly 2240 applications have declared the permission of accessing Internet. The fortieth (nearly 210 times) is obtaining the package size of application which belongs to private data.

These defects bring a number of problems to the Android system and have an impact on the security and practicability of the multimedia data on the device. Excessive permissions are requested in the process of application development, which may be exploited by malicious software. User cannot predominate over the management work of the permission mechanism, when they wonder to modify the configuration. Thus, the existing permission mechanism still cannot satisfy the requirement from users.

2.2. Statistical Analysis of Permission Usage in Android system. As for Android applications, different applications have different functions. If the application needs to make use of the system resources, the application has to declare the corresponding permissions in the *AndroidManifest* file. Thus, different types of application have different permission declarations. This paper summarizes the permission sets according to the different application categories. And next, this section analyzes the differences of permission sets in different categories and the relevance of different permissions in the same category.

More than 2200 applications were downloaded in this research, and the samples applications were divided into 21 categories according to the method provided by one popular app market software [9]. The permission statistic result is shown in Figure 1. It is easy to conclude that 99% of the applications in the statistical sample declare the permission named *android.permission.internet*. While *android.permission.access_network_state* and *android.permission.access_wifi_state* these two permissions are the second and third most permission declared in application.

The average amount of permissions declared in each group is found to be different, it is concluded that the

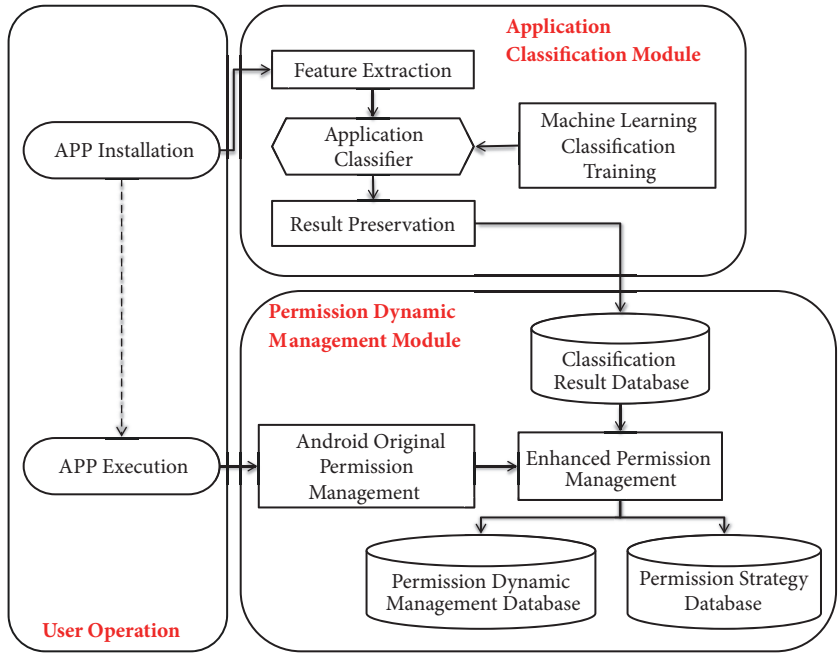


Figure 2: The architecture of the permission management scheme.

applications of communication's category have more granted permissions than other categories generally. The applications of children's category have the least declared permissions, since they are less functional and simpler to implement.

Based on the result of the above analysis, it is easy to know that the permissions between different application categories is very different. Therefore, the category that the application belongs to should be taken into consideration in the permissions management scheme.

3. Design of Permission Management Scheme

This section mainly describes the overall design of the permission management scheme. It accomplishes dynamic permission management according to the analysis result of the permission usage situation of different application categories. The scheme is divided into application classification module and dynamic permission management module.

3.1. The Architecture of Scheme. This paper aims to design an improved permission dynamic management scheme. After the researching on Android permission management technology, the permission management scheme of Android application software is designed and implemented and the architecture is shown in Figure 2. The whole scheme consists of application classification module and permission dynamic management module.

The application classification module will firstly extract the permissions and the sensitive API information by decompiled the APK file. These two records will be considered as the feature value of application classification.

The permission dynamic management module employs the classifier mentioned above to determine which one category the application belongs to and then puts forward the permission warning to users according to the permission whitelist of the corresponding category. Finally, the module asks users to decide whether to grant the permissions.

3.2. Application Classification Module. The accurate and efficient work of classifier provides the whole scheme guarantee of the effective execution. Actually, classification technology is employed to solve problem in a large scale of industries, even in medical business [10]. Most machine learning algorithms provide a convenient way to represent sequential observations and tend to cluster between different components [11]. Therefore, this section mainly introduces the design of the application classifier model generation, and then explains each step in the module.

3.2.1. Training of Application Classifier. The application classifier is obtained by training and testing a large number of APK applications on PC or server, which is the most critical and complex facet of the permission management scheme. The process of machine learning classification training includes the collection of APK data set, data preprocessing, data modelling, feature extraction and processing, training, and testing (see Figure 3). These steps are described below.

Data Collection and Preprocessing. The APK data sets are selected from YingYongBao which is the most popular app store software in Chinese market. The data sets will be preprocessed and saved in different categories. Eventually, the collection obtained a total of 2240 APK data sets and all these data sets were stored in different categories. There are 21 categories in the application classification, which includes Security, Office, Navigation, Children, Tools, Shopping,

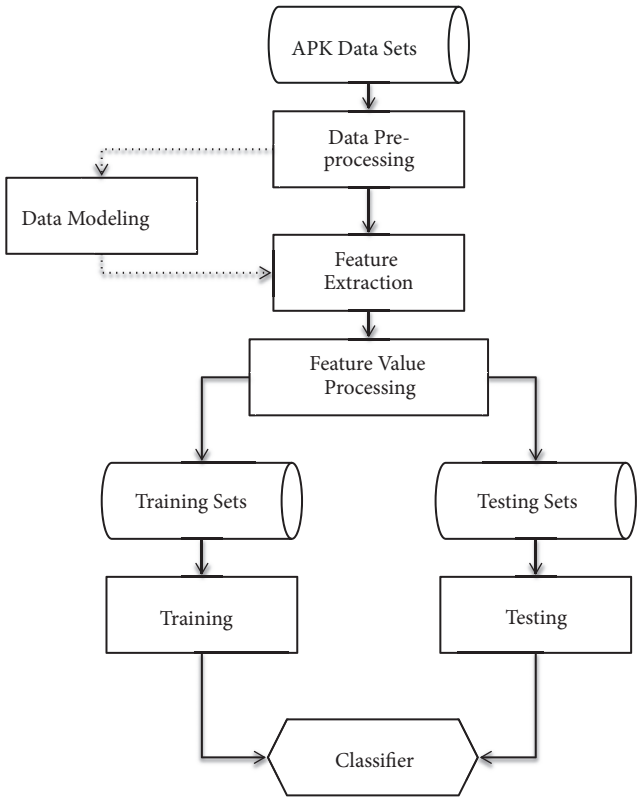


Figure 3: Flowchart of application classification training.

Health, Education, Financial, Traveling, Prettification, Social, Photography, Life, Video, Communication, System, News, Music, Entertainment, and Reading.

Data Modeling. Because the same category of Android applications implements similar functionalities, it is easy to find that they apply similar system APIs and permissions. Thus, this paper adapts the combination of permissions and sensitive APIs as the feature value in the process of data modelling.

This paper adopts the vector space model [12] and transfers the APK file into a multidimensional vector according to the importance of the feature items, and each feature matches one dimensional in the vector. The weights of feature items are calculated with Binary method. It is assumed here that the collected data sets are expressed as

$$(P_1, A_1, C_1), \ldots (P_i, A_i, C_i), \ldots, (P_n, A_n, C_n) \quad (1)$$

where P_i represents the permission usage situation of No.i APP, for example, $P_1 = (0, 1, 0, 1, 1)$ means APP1 employs the second, fourth, and fifth permission, A_i represents the usage situation of sensitive API in the No.i APP, and $A_1 = (1, 0, 1, 0, 0)$ means that APP1 uses both the first and third sensitive APIs. In addition, $C_i \epsilon \{21$ *application categories*$\}$ for example, C_1 = Education which represents APP1 belonging to education category. According to the method above, the APK data can be represented in vectors completely.

Extraction and Processing of Feature Values. Feature value extraction and processing is the most time-consuming and complex process in the application classification. The permission information in the APK file can be extracted from the *AndroidManifest.xml* file by decompiling the APK file. As for the sensitive API information, firstly use *Apktool* [13] to decompile the APK file, analyze the Smali file to obtain the API used in the APK, and after that take the API data to make the String comparison with the sensitive API library [14]. When the API is coincided with the record in database, it will be included in the feature value of this APK file.

Training and Testing. In the classification algorithm of machine learning, the whole data set will be divided into training set and test set. The training set is used to train the model according to a certain classification algorithm in order to produce the classifier [15]. The test set is used to test the classification ability of the classifier we obtained. This paper uses open source machine learning algorithm to train and test the data set, like Logistic regression algorithm [16].

3.2.2. Implementation of Application Classification Module. The application classification module employs the generated classifier to classify the application when it is installed and then saves the result in the classification result database. The detailed processes of the application classification module are shown in Figure 4.

Feature Information Extraction. The feature information extracted from APK applications includes Permission and API, which will be used in classification. Therefore, it is necessary to transfer the above feature information to the feature value that the classifier can recognize.

Classify Prediction. The first step of classify prediction process is to put the permission and sensitive API information into the classifier. The classifier judges and predicts which category this application belongs to and then outputs the classification result.

Saving Classification Results. This step is mainly to store the classification results in the classification results database. There is one classification result table in the database, which is used to store the classification results predicted by the application classifier for all the applications installed in the system. It mainly contains the UID of the application, the name of the application, and the category to which the applications belong.

3.3. Permission Dynamic Management Module. The permission dynamic management module enhances the original permission mechanism, which makes it become more fine-grained. The module can dynamically grant the permission and withdraw the authorization in the execution of the program and even notice users about the details of current permissions. This section includes the design and implementation of database, the design, and implementation of dynamic permission management.

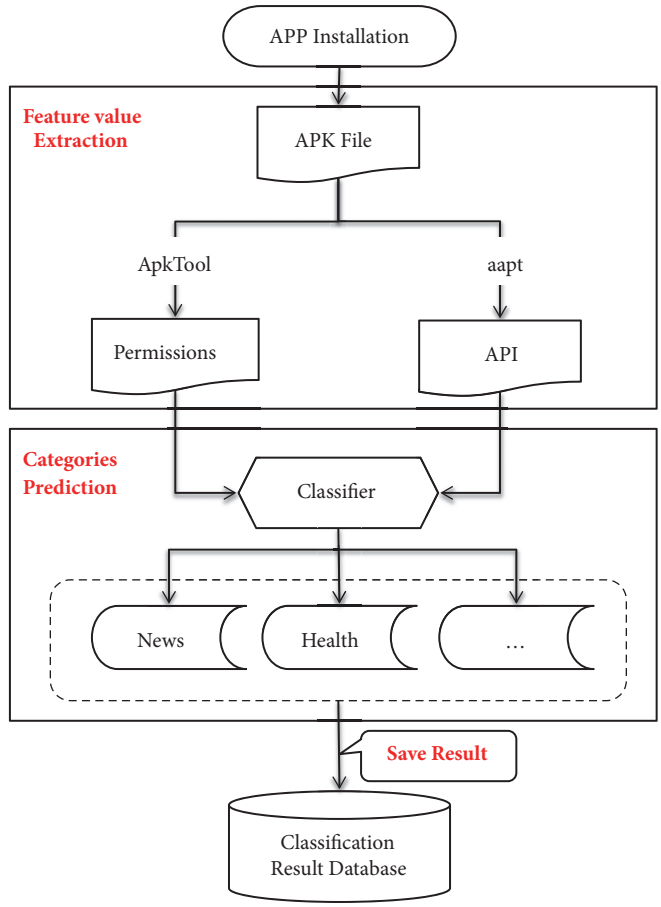


Figure 4: The flowchart of application classification module.

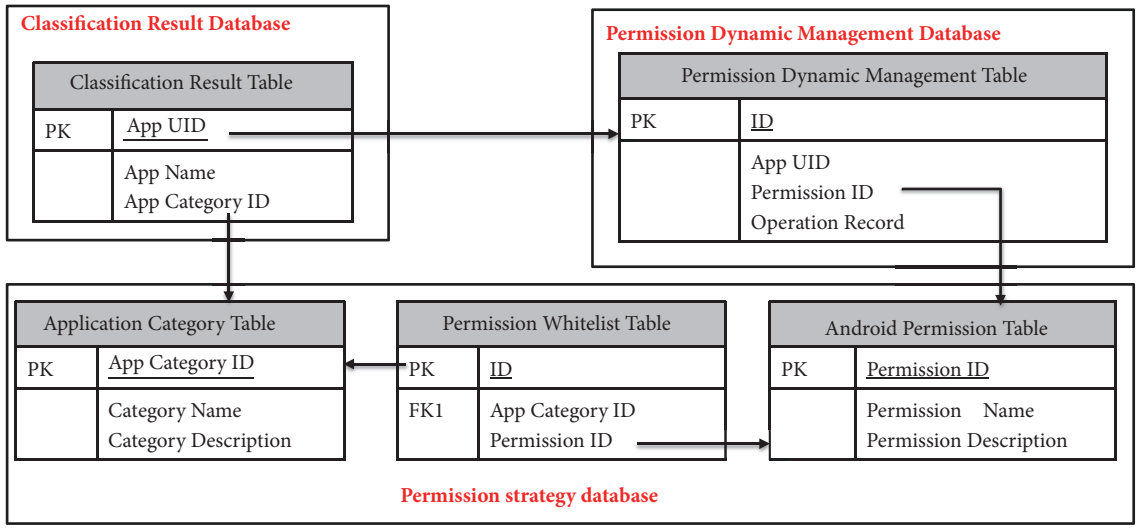


Figure 5: Database module.

3.3.1. Database. Enhanced permission management is based on three databases: they are the classification result database, application permission dynamic management database, and permission policy database. Figure 5 shows the logical model of all three databases.

Classification Result Database. The classification result database contains only one table, that is, the classification result table. The attribute named App UID is used to query the category of the application in the dynamic permissions management.

Permission Dynamic Management Database. This database contains one table named application permission dynamic management table. The table is empty at the very first. It will be updated with the installation and usage of the application. When an application first uses the permission, the dynamic management table will add a record indicating that this application had used this permission before and the operation is recorded as grant.

Permission Strategy Database. The permissions strategy database contains three tables. The application category table stores the information of application categories. Each category contains similar application functions and the same purposes. This table contains 21 categories mentioned in the Section 3.2.1. The Android permission table records all the permission information. The Permission whitelist table stores 10 permissions at most for each application category, which regarded as the whitelist permissions of the application category. If the application requests permissions which are not included in the whitelist table, the permission dynamic management module will inform user and then lets the user decide whether to grant the unknown permission.

3.3.2. Permission Dynamic Management. The enhanced permission management model makes an improvement on the original permission management model. In other word, when application make use of system resources, assuming that it have obtained authorization from the Android original permissions checking, the enhanced permission management still needs to further check. The enhanced one asks users to decide whether to authorize and update the database records at the same time. The detailed process of the enhanced permission management is shown in Figure 6.

Step 1 (query the permissions dynamic management database). The permissions dynamic management database is used to save the grant state of all permissions for this application. The state is either granted or rejected. When the database is initialized, the application does not have any granted or blocked permissions. Thus, this table will be empty when the application is installed. While the application is being executed, the application will be authorized dynamically. Additionally, users can also dynamically manage the database through a specific graphical interface.

After passing through the system permission check, the permission dynamic management module will query the permissions dynamic management database.

(a) If this application has been granted this permission, it will be passed and allowed to operate;

(b) If the permission has been rejected, the operation will be forbidden and terminated;

(c) If you do not have operation record for this permission, then go to *Step 2*.

Step 2 (identify application categories). The classification result database is used to obtain the category of the application. The classification result is predicted by using the classifier and saved in the database when the application is installed. Next go to Step 3.

Step 3 (query the permissions strategy database). Query the permission strategy database to confirm whether the permission is in the whitelist according to the category to which the application belongs.

(a) If it is in the whitelist, which means that the permission is common used and reliable in this application category, then go to the *Step 5*.

(b) If not, go to *Step 4*.

Step 4 (inform user about the permission request). Permission is not on the whitelist, which represents that the permission may be a risky permission for this application. The scheme needs to remind the function of the permission and the risk of granting permission to the user. User can determine whether or not to grant this permission.

(a) If the user chooses to grant the permission, go to *Step 5*;

(b) If the user chooses to block the permission request, go to *Step 6*.

Step 5 (pass the permission checking and update the dynamic management database). At this step, the request of new permission has already passed the checking of dynamic management module. User can access the resource in a safe environment. On the other hand, the record of this granted permission in the permissions dynamic management database needs to be updated. The state of this permission should be marked as granted. Finally, the process ends here.

Step 6 (reject the request and update the permissions dynamic management library). At this step, the request of new permission did not pass the checking of dynamic management module. It will be prohibited to access the requested resources. Meanwhile, the record of this rejected permission in the permissions dynamic management database needs to update. The state of this permission should be marked as rejected. Finally, the process ends here.

4. Experimental Verification and Analysis

This section introduces the verification and analysis of the scheme's feasibility. The first step is the training process of classifier, and then the classifier is applied to test environment to verify the security and reliability of the dynamic permission management system.

4.1. Testing Environment. This paper implements the scheme in the development environment shown in Table 1.

4.2. Machine Learning Algorithm. In order to obtain the most accurate application classifier, we need to select the appropriate machine learning algorithm to train and test the extracted data sets. This paper makes use of open source machine learning algorithms to train the classifier.

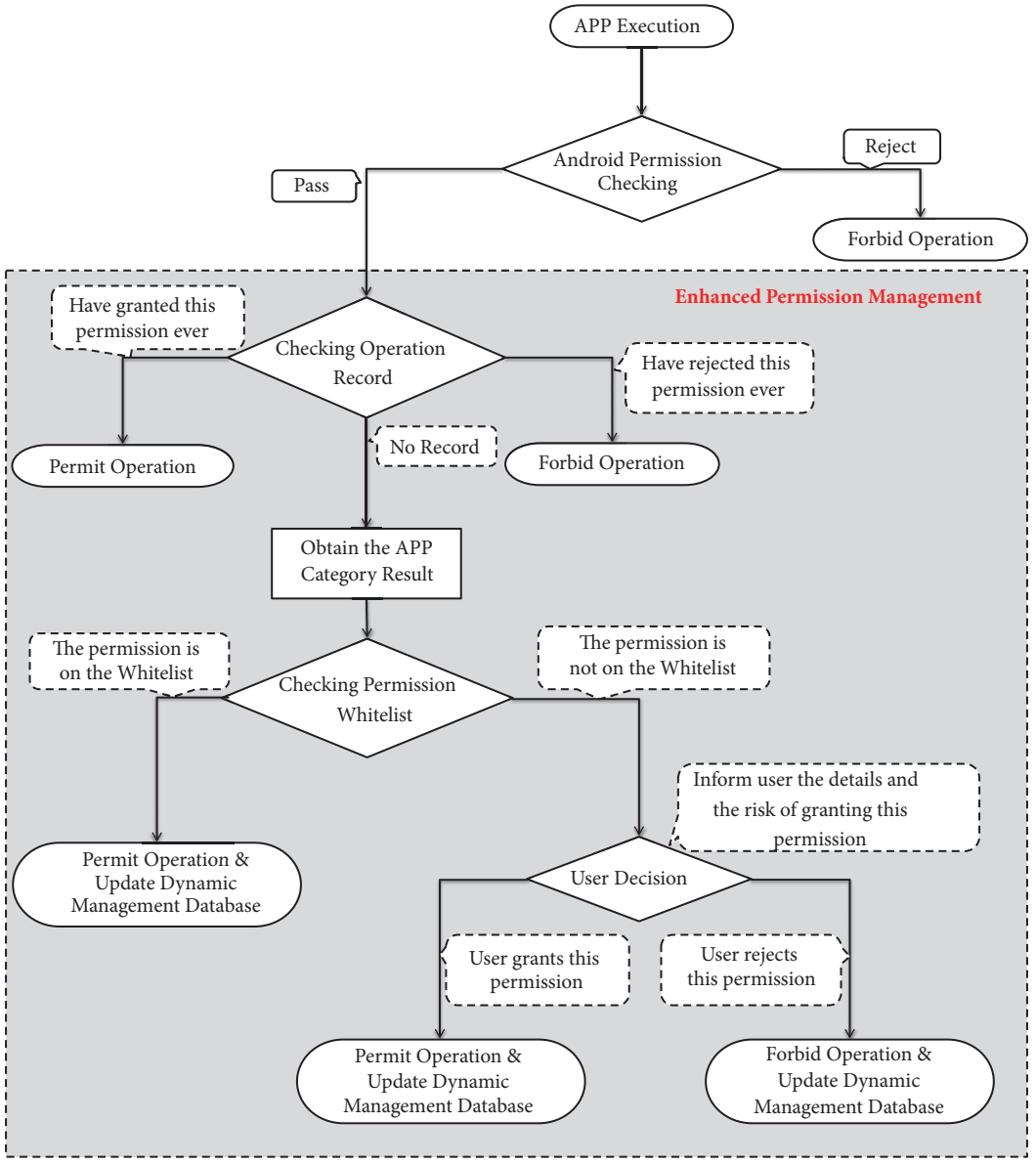


Figure 6: Enhanced permission management mechanism.

Table 1: Developing environment.

Device Version	Nexus 5
Android Version	4.4.2
Baseband Version	M8974A-1.0.25.0.23
Kernel Version	3.4.0-g0315133android-build@wpiy2.hot.corp.google.com
OS Version	aosp_hammerhead-userdebug 4.4.2 LMY48M

The widely used machine learning algorithms include Naive Bayes, Logical Regression, Decision, Tree and SVM.

In the scheme, the classification features are related to the app functions and attributes. However, the functions and attributes will be iterated and changed constantly with the development of application, so it is necessary to incorporate new training data to update the application classifier in time. In addition, the correlation between the application features has little impact to the accuracy of the classifier in our scheme.

In the research to the machine learning algorithms, we found that Logical regression algorithm has many methods of regularization model, so it does not need to consider the correlation of features. Comparing with the decision tree and SVM, logical regression algorithm is easier to update

Table 2: Results of different machine learning algorithm comparison.

Machine learning algorithm	Auc score	Accuracy	FPR	TPR
SMO	0.90	0.94	0.06	0.83
Logical Regression	0.91	0.94	0.07	0.80
Native Bayes	0.80	0.87	0.21	0.90
Decision Tree	0.84	0.89	0.15	0.95

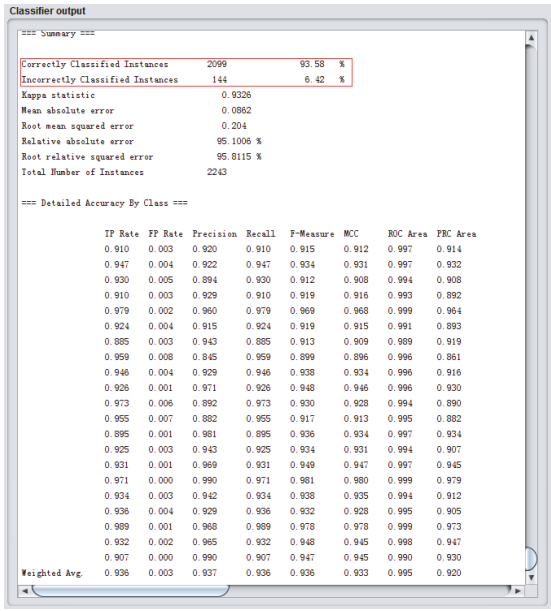


Figure 7: The classifier output module. There are 21 rows in the detailed accuracy by class, each row representing one category. The last row represents the weighted average of all application.

the model and incorporate new features data. As for the application classifier, it will be always updated with the generation of new application categories. Thus, Logical regression algorithm is the most suitable classification algorithm for the application classifier.

4.3. Classifier Training. This paper uses Weka 3.8.0 tools to implement the application classifier. Weka integrates a large number of machine learning algorithms that are able to undertake the task of data mining. The feature datasets of application permissions and sensitive API extracted from the previous steps are used in Weka as training and test datasets. Weka integrates machine learning algorithms such as SMO, logical regression, Bayesian, and decision tree.

The Classifier output module of Weka gives out the result of training and testing in the form of text, which is shown in Figure 7. The "Correctly Classified Instances" filed means that the classification accuracy is up to 93.58% by means of selecting permission and API to be the feature value for machine learning.

We also compare above four machine learning algorithms and select four metrics to evaluate their efficiency. The result is shown in Table 2. It is obvious concluding that the logic regression algorithm shows the best results both in auc score

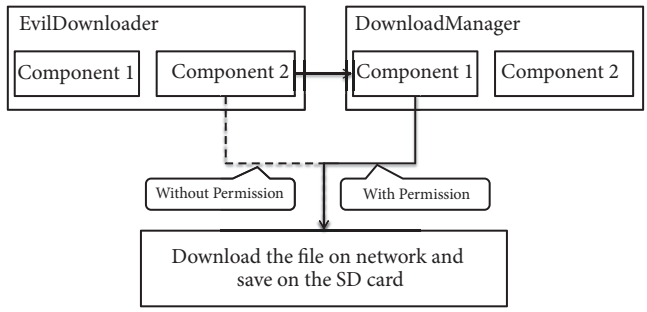


Figure 8: The Principle of this privilege escalation attack. EvilDownloader invokes the exposed component (broadcast receiver) in DownloadManager to achieve the goal of downloading file from internet.

and accuracy. All the accuracy scores are more than 0.8, which means it is feasible to use permissions and sensitive APIs in application classification.

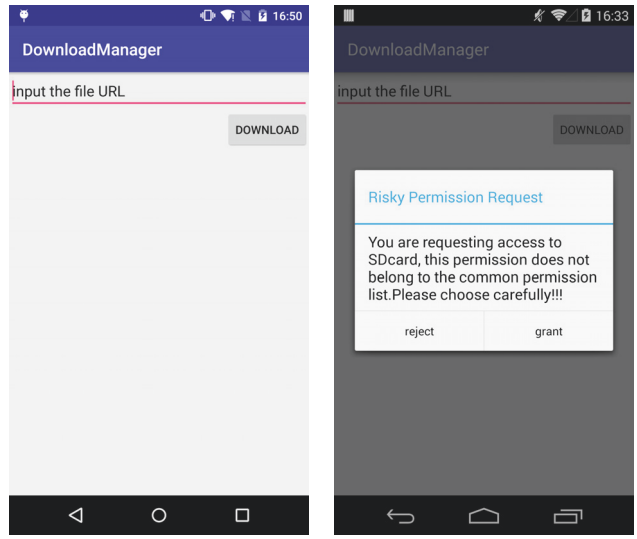
4.4. Function Testing. This paper designed a privilege escalation attack [17] to test the function of the system. The attack model program contains two applications: *EvilDownloader* (attack program, without privilege to access the network and SDcard) and *DownloadManager* (download management program, user had already granted this program with the permission to access the network and SDcard). The principle of this attack is that *EvilDownloader* can access the restricted resources through *DownloadManager*, which achieves the purpose of escalating *EvilDownloader*'s privilege. The principle is shown in Figure 8.

DownloadManager actually simulates a download program, enters the URL of the file, clicks the download button, and starts downloading. The downloaded file is saved to the SD card, and the program is shown in Figure 9(a).

DownloadManager has the permission to download files from internet and save them to the SD card. It downloads files by receiving a request broadcast and then accessing the URL address to download the file. The permission declaration and the broadcast receiver information is defined in the AndroidManifest.xml, which shown in Figure 10.

EvilDownloader behaves as an attack program without any permission. It is worth mentioning that it can also send a broadcast request to download the URL file through Intent, as shown in Figure 11.

From privilege escalation attack code shown in Figure 11, it represents that the *EvilDownloader* will download URL file "http://i4.piimg.com/11340/7f638e192b9079e6.jpg" and save the file, which the file name is "jb.png". In other word,



(a) (b)

Figure 9: The interface of DownloadManager. In figure (a), there are an EditText and button on the UI. User need to input proper URL and click the button to download multimedia data. In figure (b), the improved permission management system pops out the alert dialog to inform user the risk operation and user can choose "reject" to protect multimedia data from being accessed illegally.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="DownloadManager"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".MainActivity"
        android:label="DownloadManager" >
        <intent-filter...>
    </activity>
    <receiver android:name=".DownloadReceiver">
        <intent-filter>
            <action android:name="com.droider.download"></action>
        </intent-filter>
    </receiver>
</application>
```

Permission

BroadcastReceiver

Figure 10: The AndroidManifest.xml code segment of DownloadManager.

```
setTitle("EvilDownloader");
button = findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.setAction("com.droider.download");
        intent.putExtra( name: "url", value: "http://i4.piimg.com/11340/7f638e192b9079e6.jpg");
        String filename = "jb.png";
        intent.putExtra( name: "filename",filename);
        sendBroadcast(intent);
    }
});
```

Figure 11: Code segment of EvilDownloader.

EvilDownloader downloads the file and stores it in SD card successfully, which implements the privilege escalation attack on the device. This type of attack poses a giant threat to multimedia data and can easily lead to the disclosure and damage of privacy data.

```
27538-27538/? I/Xposed: [Application Classification]: new application is installing
27538-27538/? I/Xposed: [Application Classification]: PacakgeName = com.admin.downloadmsnager; UID = 100095
27538-27538/? I/Xposed: [Application Classification]: Features = [0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,
27538-27538/? I/Xposed: [Application Classification]: Classifier Prediction.....
27538-27538/? I/Xposed: [Application Classification]: Category Result = [tools]
27538-27538/? I/Xposed: [Application Classification]: Saving Result.....
27538-27538/? I/Xposed: [Application Classification]: classification work finished
```

Figure 12: The installing log of Downloadmanager. The system checks the application and predicts the classification result of the application and, finally, saving the result in the database.

In the experiment, the attack is respectively implemented in the normal Android environment and the environment in which our scheme is deployed.

4.4.1. Classifier Experiment. The experiment installed these two applications by the way of ADB. Figure 12 is the installing log of *DownloadManager*. The figure shows that the application classification module monitors the installing process of the application and extracts the feature value. And next, the classifier predicts the category of the installing application. The result shown in the Figure 12 is that *DownloadManger* belongs to "Tools" category.

Therefore, in the improved permission management mechanism, *DownloadManger* is executed with the permission whitelist of "Tools" category. If there is any new permission need to be granted, user will receive the alert about the suspected permission request.

4.4.2. Normal Android Environment

The Execution Result. EvilDownloader can successfully exploit the components of *DownloadManager* to download the specified file and save it in SDcard without user's awareness.

Table 3: CPU occupancy rate comparison for five applications.

CPU occupancy	APP1	APP2	APP3	APP4	APP5
without Scheme Deployed	33%	27%	41%	60%	57%
with Scheme Deployed	41%	30%	47%	67%	66%

Table 4: Memory occupancy comparison for five applications (M).

Memory occupancy	APP1	APP2	APP3	APP4	APP5
without Scheme Deployed	17	29	54	44	113
with Scheme Deployed	31	34	77	83	147

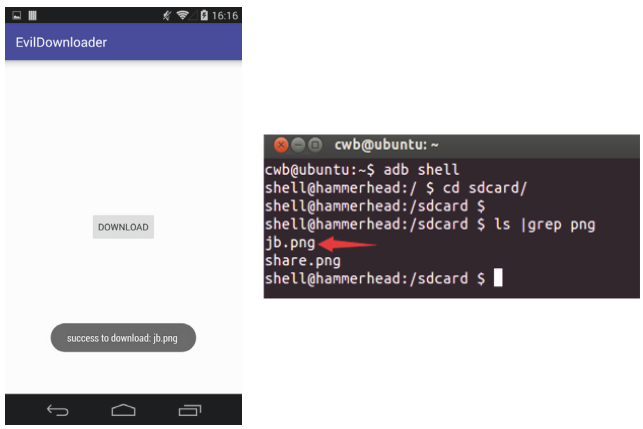


Figure 13: Experiment result in Normal Android Environment: there is a toast shown on the UI to inform the file is downloaded successfully. Checking the file folder of sdcard path, the file "jb,npg" is shown in the list.

The experimental results are shown in Figure 13. The file named "jb.png" was successfully downloaded and saved. Thus, *EvilDownloader* implemented privilege escalation attack. The attack program finished the unauthorized operation without requesting the permissions about accessing to the network and SDcard.

4.4.3. Environment with Scheme Deployed

The Execution Result. After clicking *download* button, the toast "*successful download jb.png*" did not pop up. Instead, the dialog appears on the phone, which is shown in Figure 9(b).

User has to choose whether to grant this permission or not. When clicking the "grant" button, the program will continue to work: download the file and save it on the SD card. However, if you click the "reject" button to reject the request of permission, there will appear exception to block the download operation. The SecurityException is shown in Figure 14.

The main reason is that the required permission is not in the permission whitelist. Furthermore, after the request of permission is rejected, Android throws an exception to figure the unauthorized operation out. The permission management scheme in this paper will detect the risky request of

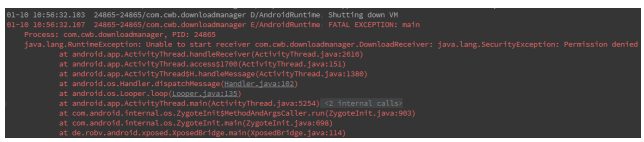


Figure 14: Experiment result in Environment with Scheme Deployed: the application throws an exception after the user choose "reject" to grant the permission. The program is interrupted by the improved permission management mechanism.

permissions and report it to the user in time. Once user rejects the authorization, system will throw a SecurityException exception to prevent malicious behavior effectively.

Therefore, the permission management scheme in this paper intercepts the malicious operation successfully, which protects the multimedia data from being accessed by attack program. This scheme accomplishes the security protection to the private data in Android devices and improves the security of current Android permission management system.

4.5. Performance Evaluation. This section conducts the performance evaluation to verify the efficiency of the scheme. It is obvious to find that the scheme will always interrupt the system operation when new request of permission appears on the phone. It is necessary to check the consumption of memory and CPU in the Android OS, and the time cost with the scheme running.

4.5.1. Device Consumption Evaluation. We executed five applications in the same testing device to compare the performance of the improved scheme. Table 3 records the CPU occupancy rate for the five applications in the two situations. The scheme obviously produce an impact to the CPU occupancy, but all the increase part are no more than 10%, which can be ignored.

Table 4 records the max value of memory occupancy when these five applications running in the devices. In the scheme, the process of querying the database is the major consumption point. According to the result, the average increase memory is no more than 25 M, which also can be ignored.

The result shows that the scheme increases the consumption of device, but it can be ignored in the running process.

Table 5: Installing Time comparison for five applications (ms).

Installing Time	APP1	APP2	APP3	APP4	APP5
without Scheme Deployed	3144	2779	4421	5843	5339
with Scheme Deployed	3797	4029	6711	6674	7936

4.5.2. Time Consuming Evaluation. The scheme resolves the APK file when the application is installed on the phone. This period is the most time-consuming part during the whole running process of the scheme. Table 5 records the time cost in the installing process. The amount of increase time depends on the size of APK file and the number of declared permissions. Generally, the larger the APK file is, the more time that the application costs in the scheme.

According to the results of performance evaluation, the improved permission management scheme produces the extra performance cost, but the amount is so small that can be ignored in the running process of the scheme.

5. Conclusions

This paper gives a research on permission management technology of Android application based on machine learning. The proposed scheme combines machine learning with permission management to enhance the original permission management mechanism. The improved scheme prohibits the illegal operation, like extracting multimedia data on the internet.

However, the implementation of the proposed scheme is based on *Xposed*. Because *Xposed* framework and relies on rooting Android devices, this requirement produces a great discount on the safety to Android system. It is necessary to migrate this scheme to the real Android system in the future work.

Data Availability

The data used to support the findings of this study are free and publicly available on Internet. The database data and classification training data used to support the findings of this study have been deposited in https://pan.baidu.com/s/1talTsZrqKzY_0paf7FrqWg. The password is "6dbv."

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China (nos. U1536121, 61370195). It is worth mentioning that the previous version of this work had been presented in the form of abstract poster on the ICCCS 2018, which was held on June 8, 2018, in Haikou.

References

[1] C. Jinhua, Z. Yuanyuan, C. Zhiping et al., "Securing Display Path for Security-Sensitive Applications on Mobile Devices Computers," *Computers, Material & Continua*, vol. 55, no. 1, pp. 017–035, 2018.

[2] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive fractional-Pixel motion estimation skipped algorithm for efficient HEVC motion estimation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 1–19, 2018.

[3] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11)*, pp. 627–638, ACM, Chicago, Ill, USA, October 2011.

[4] S. Jaewoo, L. Kyeonghwan, C. Seong-je et al., "Static and Dynamic Analysis of Android Malware and Goodware Written with Unity Framework," *Security & Communication Networks*, vol. 2018, Article ID 6280768, 12 pages, 2018.

[5] W. Huanran, H. Hui, Z. Weizhe et al., "Demadroid: Object Reference Graph-Based Malware Detection," *Security and Communication Networks*, vol. 2018, Article ID 7064131, 16 pages, 2018.

[6] Y. Zhang, M. Yang, B. Xu et al., "Vetting undesirable behaviors in Android apps with permission use analysis," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*, pp. 611–622, ACM, Berlin, Germany, November 2013.

[7] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner, "AdDroid: privilege separation for applications and advertisers in Android," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12)*, pp. 71-72, Seoul, Republic of Korea, May 2012.

[8] L. Lei and H. Yong, "The Research on Android Application of authority detection technology," *Journal of Information Security Research*, vol. 02, pp. 139–144, 2017.

[9] Classifcation of the popular APP, edited by 2017-6 http://sj.qq.com/myapp/.

[10] R. Gurusamy and V. Subramaniam, "A machine learning approach for MRI brain tumor classification," *Computers, Materials and Continua*, vol. 53, no. 2, pp. 91–109, 2017.

[11] Y. Zheng, B. Jeon, L. Sun, J. Zhang, and H. Zhang, "Student's t-hidden markov model for unsupervised learning using localized feature selection," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[12] G. Sidorov, A. Gelbukh, H. Gómez-Adorno, and D. Pinto, "Soft similarity and soft cosine measure: Similarity of features in vector space model," *Computacion y Sistemas*, vol. 18, no. 3, pp. 491–504, 2014.

[13] Ma. Z, "Android application install-time permission validation and run-time malicious pattern detection," *Personal & Ubiquitous Computing*, vol. 18, no. 8, pp. 1963–1976, 2014.

[14] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "PScout: analyzing the Android permission specification," in *Proceedings of the*

ACM Conference on Computer and Communications Security (CCS '12), pp. 217–228, ACM, October 2012.

[15] W. Huang, T. Zhao, and J. Wang, "An enhance excavation equipments classification algorithm based on acoustic spectrum dynamic feature," *Multidimensional Systems & Signal Processing*, vol. 28, no. 3, pp. 1–23, 2017.

[16] H. Hou, "Research on the correlation of learning behavior characteristics of mooc platform based on logistic regression algorithm," *Journal of Shangrao Normal University*, 2017.

[17] L. Davi, A. Dmitrienko, A. R. Sadeghi et al., "Privilege Escalation Attacks on Android Information Security," in *Proceedings of the ISC 2010 International Conference*, vol. 6531, pp. 346–360, 2011.