Permission Manager

Amit Patra
G.H. Raisoni College of Engineering and Management, Wagholi, Pune, Maharashtra
Tushar Joshi
G.H. Raisoni College of Engineering and Management, Wagholi, Pune, Maharashtra
Kiran Zade
G.H. Raisoni College of Engineering and Management, Wagholi, Pune, Maharashtra

Vaishali Shinde
G.H. Raisoni College of Engineering and Management, Wagholi, Pune, Maharashtra

Abstract *- The objective of this App Permission Manager android application is to provide feature of automatic permission. Day by day Privacy concern required on android. Android doing research and implements the solution on upcoming version but what about the older version? Because so many people still using Old Android version.*

So here our app comes to solve the problem for that. Android 11 has a lot of useful feature for app permission but globally android older version is running and most of the people concern about their privacy. Our app implements and solves the problem for other older versions.

I. INTRODUCTION

Now a day's mobile phone is a necessary part of the people's life. There is continuously rising in a number of mobile computing applications, cantered on the people's daily life. In such applications, security dependent systems have been detected as an important application. Such application which presents the architecture and implementation of such a security is commonly known as Permission Manager. We propose architecture of mobile permission security system for Android Mobile Phones that is able to provide permission information to the mobile users conveniently. Our system takes advantage of light-weighted "Android Secured Permission" technology that can combine more than one data sources to create value-added services, while overcomes the limitations of mobile devices.

Google Play has made it a basic requirement to make certain privacy-related disclosures to users, in accordance with applicable law. These disclosures are typically made available to users via a privacy notice that is easily accessible from within the app.

II. IDENTIFY, RESEARCH AND COLLECT IDEA

i. Privacy policy requirements for Android apps:

A lot of people ask for sample privacy policies for apps. The exact required contents of a privacy policy depend upon the law applicable to you and may even need to address requirements across geographical boundaries and legal jurisdictions.

For this reason, it's always advisable that you approach your (legally mandated) privacy policy with the strictest applicable regulations in mind.

ii. Sensitive permissions:

In addition to this, you need to make sure that you disclose your use of any of the following "dangerous" permission groups (personal or sensitive user data mentioned earlier) in your privacy policy:

CALENDAR
CAMERA
CONTACTS
LOCATION
MICROPHONE
PHONE
SENSORS
SMS
STORAGE

iii. Prominent disclosures:

If your app processes the personal data of users for reasons unrelated to the functionality of your app, you're required to make additional, easily visible disclosures about this usage and collect user consent where required.

If your app collects and transmits personal or sensitive user data unrelated to functionality described prominently in the app's listing on Google Play or in the app interface, then prior to the collection and transmission, it must prominently highlight how the user data will be used and have the user provide affirmative consent for such use.

Your in-app disclosure:

- Must be within the app itself, not only in the Play listing or a website;
- Must be displayed in the normal usage of the app and not require the user to navigate into a menu or settings;
- Must describe the type of data being collected;
- Must explain how the data will be used;
- Cannot only be placed in a privacy policy or terms of service; and
- Cannot be included with other disclosures unrelated to personal or sensitive data collection.

Your app's request for consent:

- Must present the consent dialog in a clear and unambiguous way;
- Must require affirmative user action (e.g. tap to accept, tick a check-box, a verbal command, etc.) in order to accept;

- Must not begin personal or sensitive data collection prior to obtaining affirmative consent;
- Must not consider navigation away from the disclosure (including tapping away or pressing the back or home button) as consent; and
- Must not utilize auto-dismissing or expiring messages.

It's worth noting that it seems that Google considers any data collection activity that isn't made obvious from your app page or from within your interface to be covered by this prominent disclosure policy.

Therefore, a separate user notice is required in addition to your privacy policy – which your notice should ultimately link – to for a full explanation of the data processed. Again, the data must not be processed until you have affirmative consent by your user.

Furthermore, under regulations like the GDPR, you are legally required to obtain informed, explicit consent before processing any personal data of users specifically where it falls outside the what's required for the functioning of your service.

iv. Report a policy violation:

Some issues require specific information for an investigation to be completed. Those issues include:

- Copyrighted content: Distribution of my copyrighted content without my permission.
- Trademark infringement: Concern about use of registered trademark without my permission.
- Inappropriate reviews: Report comments about applications published on Google Play that violate Google's policies for posting reviews.

If the violation you've noticed is related to Google Play, but doesn't fall into one of these categories, then please use this contact form to report it:

- Report Inappropriate Apps.

v. Existing Systems & Limitation:

In existing system user don't have control on their data means if a user installed one app and given it all the permission. The permission remains to app until it's not get Uninstall.

Example of this is if user gives location access to some app and used that app only sometime but the location permission remains to app and app know user location all time and it is the big threat to user security.

There are some apps which wants the permission it's not needed ex. Calculator wants camera and location permission that is also the big threat to user security.

III. Experimental Verification and Analysis

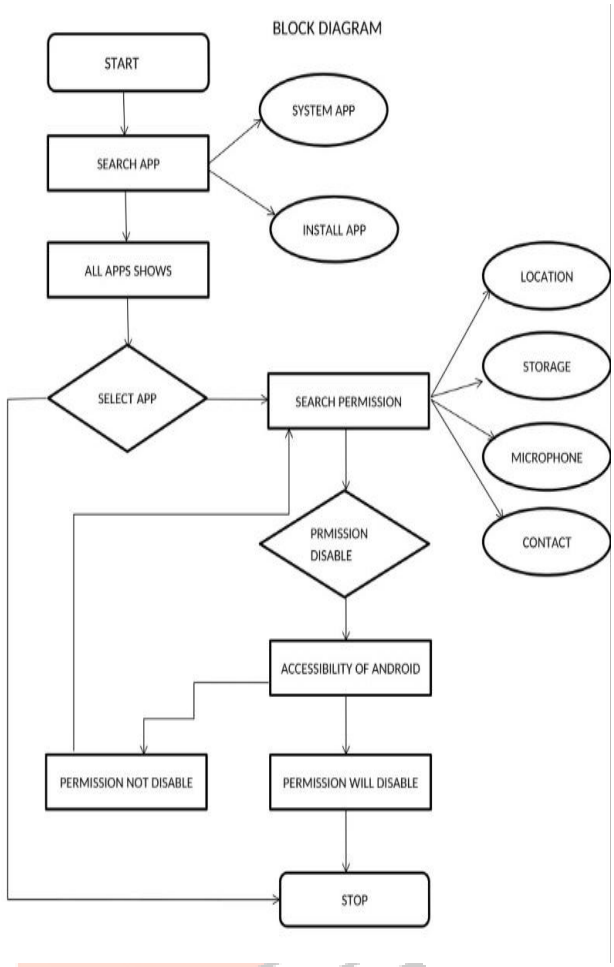
This section introduces the verification and analysis of the scheme's feasibility.

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.

Device Version	Redmi Note 3
Android Version	7.0.1
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

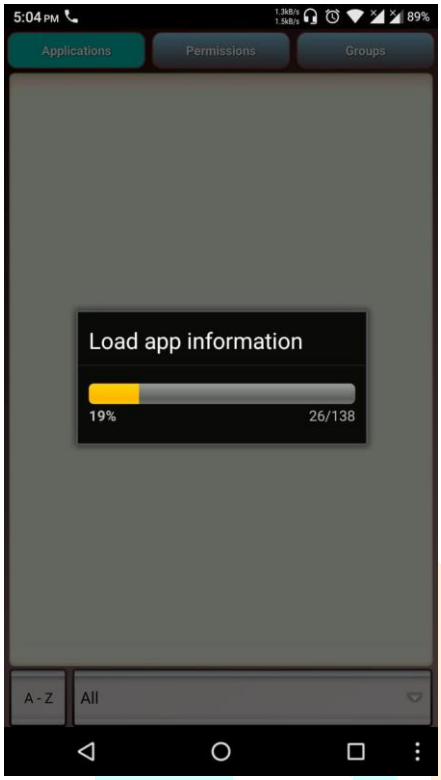
Table Developing Model

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

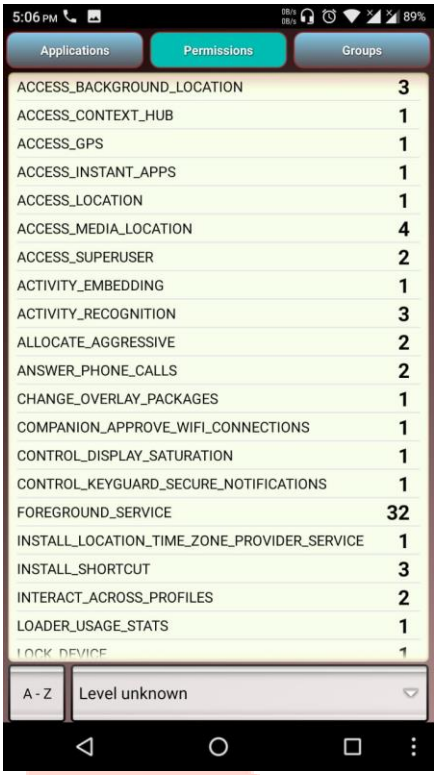


Flow of Application in block Diagram

Operation Flow. Loading of all installed and system application.

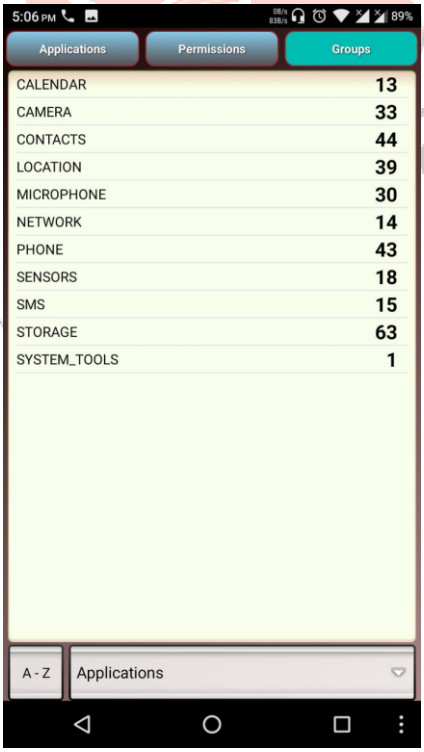


All system and installed application list.

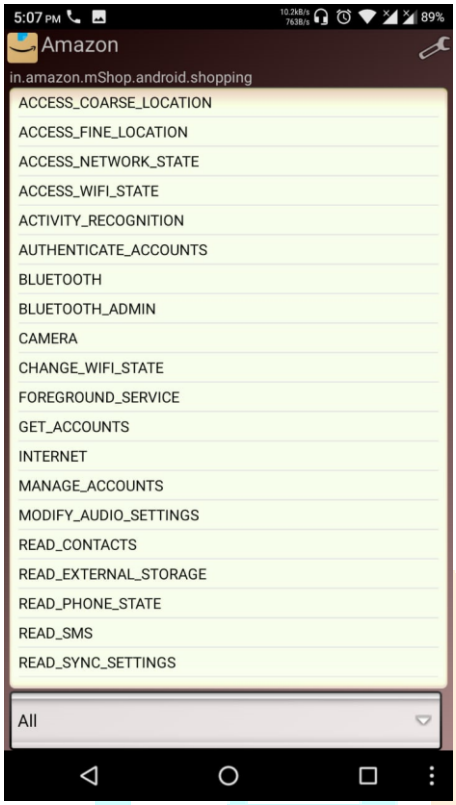


Grouped Permission

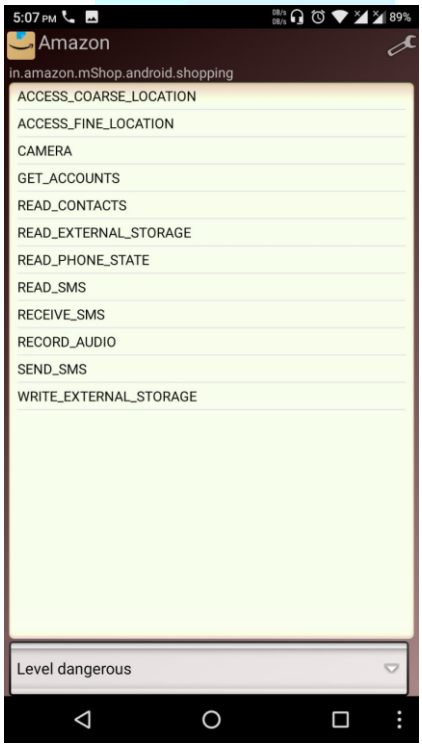
All Listed Permission

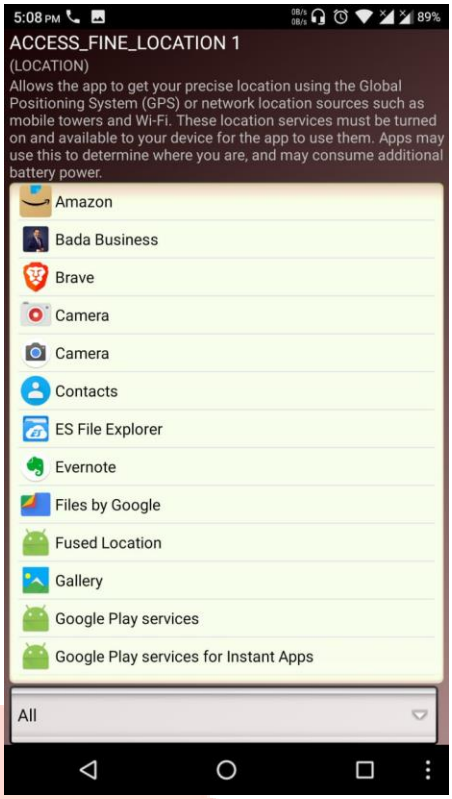


Used Application Permission

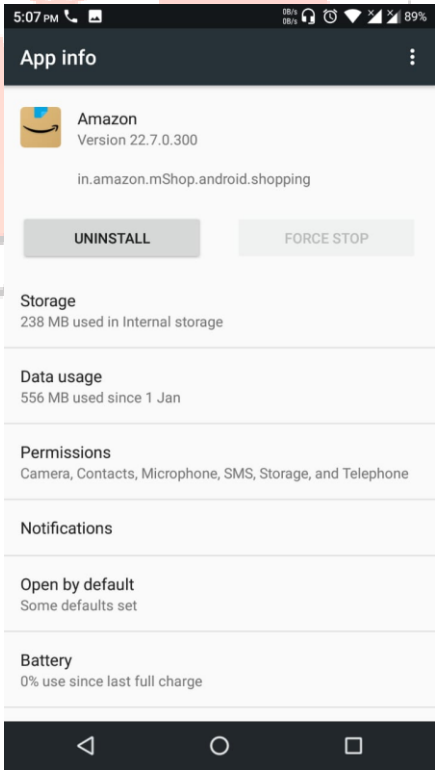


All dangers Permission list





Action Panel of Permission



Information of permission

IV. 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.

4.1. 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 "Allor-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. 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.

4.2. 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. 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 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.

V. 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.

5.1. 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. 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.

Fig. Code of PermissionHandler

```
@SuppressWarnings("WeakerAccess")
public abstract class PermissionHandler {

    /**
     * This method will be called if all of the requested permissions are granted.
     */
    public abstract void onGranted();

    /**
     * This method will be called if some of the requested permissions have been denied.
     *
     * @param context           The application context.
     * @param deniedPermissions The list of permissions which have been denied.
     */
    public void onDenied(Context context, ArrayList<String> deniedPermissions) {
        if (Permissions.loggingEnabled) {
            StringBuilder builder = new StringBuilder();
            builder.append("Denied:");
            for (String permission : deniedPermissions) {
                builder.append(" ");
                builder.append(permission);
            }
            Permissions.log(builder.toString());
        }
        Toast.makeText(context, "Permission Denied.", Toast.LENGTH_SHORT).show();
    }
```

5.2. 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.

Fig. Code of Permission Checker

```
public static void check(final Context context, String[] permissions, String rationale,
                         Options options, final PermissionHandler handler) {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
        handler.onGranted();
        log("Android version < 23");
    } else {
        Set<String> permissionsSet = new LinkedHashSet<>();
        Collections.addAll(permissionsSet, permissions);
        boolean allPermissionProvided = true;
        for (String aPermission : permissionsSet) {
            if (context.checkSelfPermission(aPermission) != PackageManager.PERMISSION_GRANTED) {
                allPermissionProvided = false;
                break;
            }
        }

        if (allPermissionProvided) {
            handler.onGranted();
            log("Permission(s) " + (PermissionsActivity.permissionHandler == null ?
                    "already granted." : "just granted from settings."));
            PermissionsActivity.permissionHandler = null;

        } else {
            PermissionsActivity.permissionHandler = handler;
            ArrayList<String> permissionsList = new ArrayList<>(permissionsSet);

            Intent intent = new Intent(context, PermissionsActivity.class)
                    .putExtra(PermissionsActivity.EXTRA_PERMISSIONS, permissionsList)
                    .putExtra(PermissionsActivity.EXTRA_RATIONALE, rationale)
                    .putExtra(PermissionsActivity.EXTRA_OPTIONS, options);
            if (options != null && options.createNewTask) {
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            }
            context.startActivity(intent);
        }
    }
}
```

At Present, Android is the most widely used mobile OS and it plays an important role in daily life. To protect the sensitive resources on Android device, the per- mission model and the sandboxing mechanism are enforced on Android applications (apps for short). Unlike iOS, however, Android's permission model is an "all-or-nothing" approach. At install time, it provides users a binary choice of either accepting all the requested permissions or not installing an app. Once an app is installed, it is impossible to change the app's permissions unless a special app called permission manager is in use.

Permission managers can be used by users to selectively grant or revoke app's permissions at runtime. A variety of permission managers, such as LBE [4], Advanced Per- mission Manager [5], and XPrivacy [6], are provided in Google Play and some third-party Android markets. In addition, some popular custom ROMs or manufacturers' devices also include built-in permission managers, such as CyanogenMod [7], MIUI [8], and Huawei P6 [9]. Google also includes a hidden built-in permission manager, called App Ops, in Android from version

VI. RESEARCH AND COLLECT IDEA

6.1 Figures and Tables:

The main objective of this paper is when users are on the move, it is able to provide rich and concise information timely and make them access to the service at anytime and anywhere. The proposed system is based on request and response, so there is no continuous acquisition of the bandwidth.

When users were given the option to access location only while an app in use, users selected about half of time. So in proposed system we are evolving the permission system to give users more control. We are giving users the options to temporarily grant permission valid for only a single use by an app. The permission grant remains valid while the app is in the foreground, either is visible activity or as a foreground service. The grant expires a short period after the app is moved to the background.

Fig. Package install and permission-control functionalities

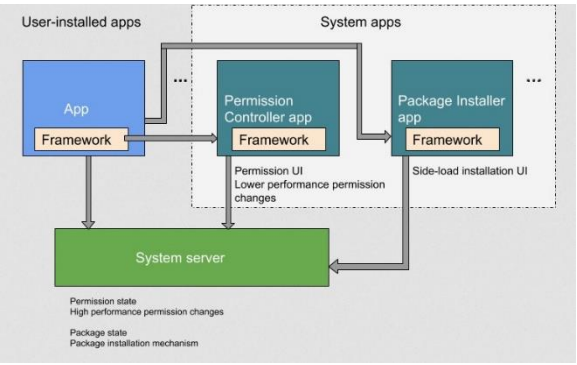
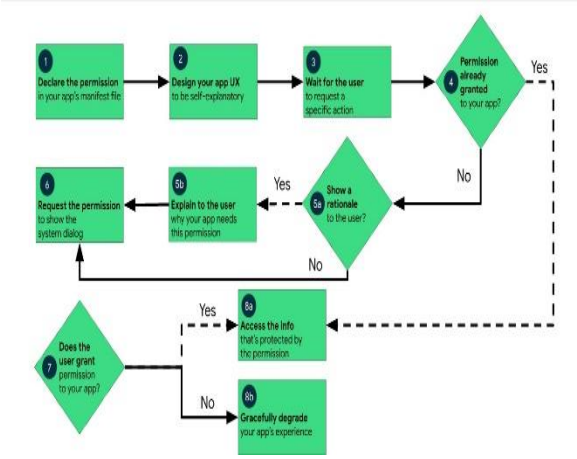


Fig. Diagram that shows the workflow for declaring and requesting runtime permissions on Android.



6.2 Tools and Technologies:

JDK8.1; The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environ- ment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development. There are different version of JDKs available and also support various platforms. The is supported by platforms like Windows, Linux or Solaris and many more.

Android Studio; Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . ... A unified environment where you can develop for all Android devices. Apply Changes to push code and resource changes to your running app without restarting your app.

JAVA; Java is an Object oriented general-purpose and very famous computer-programming language. It is also concurrent and class-based or object-oriented programming language. This is also platform independent programming language which is main advantage of using this programming language. Java is mainly designed to allow or to give privilege to application developers or software developer “write once, run anywhere" that is called as WORA which basically means that once compiled Java code can run in almost every and all plat- forms that support Java or which have JVM installed, without the need for recompilation. Java applications are basically com- piled to intermediate code that is “byte code" which can run on any virtual machine (JVM) which is regardless of the underlying computer architecture.

XML; XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. XML tags are not predefined in XML. We must define our own Tags. Xml as itself is well readable both by human and machine. Also, it is scalable and simple to develop. In Android we use xml for designing our layouts because xml is lightweight language so it doesn’t make our layout heavy.

FIREBASE; Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. ... Real-time database – the Firebase Real-time Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time.

VII. CONCLUSIONS

In this paper, we presented the design and implementation of a mobile application called Permission Manager, with which mobile users can get permission guidance information they need anytime and anywhere. So here our app come to solve the problem for that. Android 11 has a lot of useful feature for app permission but globally android older version is running and most of the people concern about their privacy. Our app implements and solve the problem for other older versions. Current mobile services are enhanced with Permission features, providing the user with better use experience. A great number of mobile phone applications appeared recently, many of which are permission-related.

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.

VII. REFERENCES

[1] M. King, B. Zhu, and S. Tang, "Optimal path planning," *Mobile Robots*, vol. 8, no. 2, pp. 520-531, March 2001.

[2] H. Simpson, *Dumb Robots*, 3rd ed., Springfield: UOS Press, 2004, pp.6-9.

[3] M. King and B. Zhu, "Gaming strategies," in Path Planning to the West, vol. II, S. Tang and M. King, Eds. Xian: Jiaoda Press, 1998, pp. 158-176.

[4] B. Simpson, et al, "Title of paper goes here if known," unpublished.

[5] J.-G. Lu, "Title of paper with only the first word capitalized," *J. Name Stand. Abbrev.*, in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Translated J. Magn. Japan*, vol. 2, pp. 740-741, August 1987 [*Digest 9th Annual Conf. Magnetics Japan*, p. 301, 1982].

[7] 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.

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

[9] 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.

[10] 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.