IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

PRIVACY PROTECTION FRAMEWORK FOR ANDROID

[1]Jannath Parvin, [2]Aiswarya K Shaji, [3]Hani Haridas P

[1]Engineer, [2]Engineer, [3]Engineer [1]Computer Science and Engineering,
[1]Name of organization of 1st Author, City, Country

Abstract : Numerous creative and clever Android applications have emerged as a result of the Android platform's recent growth in popularity and users (apps). Numerous of these applications are extremely interactive, programmable, and dependent on user data to function. While this is practical, user privacy is the main issue. It is not assured that these applications don't break algorithms or keep user data for their own use. Android secures and safeguards user data through a permissions mechanism. During installation or runtime, the user can grant permissions to the necessary resources**.**

IndexTerms **- Instrumentation, Authorization model, Security and privacy in relation to Android.**

INTRODUCTION

Android is the most popular operating system (OS). reaching mobile platforms According to figures from throughout the world, iOS had a 25% market share in the mobile OS industry in June 2020, while Android OS held close to 75% of the market. Users favor Android because it is free and open-source and supports a wide variety of applications. Because Android is open-source, developers favor it over the competing iOS. Applications are what Android apps, which are mostly built in Java, are known as. A crucial component of applications is security. Because of the nature of Android applications, it is challenging to rely on traditional, static, and dynamic systems for malware research. Google developed a program to improve the security of Google Play applications in order to provide security services to Google Play application developers for the security of their applications. Applications are screened for potential viruses before they are released to the Play Store. In order to increase device security and Play Store usage, Google tried to detect malware and potentially hazardous applications in 2017. Play Protect by Google. To prevent background apps from having access to the camera, microphone, and sensors, Google limited the access of background sensors in Android 9. Google has included a feature for hardware-backed up keys to protect user information and passwords.

Additionally, a Safe Browsing app programming interface (API) is available for defense against misleading websites. Some apps that collect user data may still be hazardous, despite considerable advancements in platform security, application development security, and a safe Android operating system. It might be difficult to distinguish whether apps in the Google Play Store are dangerous or may gather user data for the purpose of analyzing their behavior or selling it to a third party because there are already over 2.7 million of them available. Android relies on application permission to prevent data security issues and malicious use, which means that applications must request the rights required for their applications from the user and that Android only provides access to the APIs for that permission if the user approves it. Normal, hazardous, and signature permissions are the security level categories for Android permissions. Just the necessary permissions must be mentioned by the developer in the manifest file. Users are not asked for these permissions by Android. Unsafe permissions may cause significant concerns with data breaches. Therefore, the user should be prompted to grant the program permission to use these permissions within the application. These permissions should be listed in the manifest file. allowed users to utilize the program without actually giving any permissions access. The data leakage issue was still unresolved, though, as several programs began to break when the user refused to provide permission. As was already mentioned, the mission methods used by the Android OS platform. Some applications demand a lot of permissions. These permissions were created with the intention of examining Android's current security procedures, user privacy issues, and data trading

NEED OF THE STUDY.

The security and privacy of user data on Android devices are guaranteed by the privacy protection framework for Android. Millions of devices utilize the open-source operating system called Android, which was created by Google. It is critical to build a strong framework to secure user privacy given the broad use of Android.

The history of Android's privacy protection mechanism may be traced back to three important factors including growing worries about data privacy: With the prevalence of mobile devices and the enormous amounts of personal data they hold, there are growing worries about data privacy. Increased privacy measures are now more necessary as a result of high-profile data breaches and unauthorized access to customer information.

Global governments and regulatory organizations have put in place policies to protect user information after realizing the value of data privacy. For instance, the General Data Protection Regulation (GDPR), which specifies standards for the gathering, storing, and processing of personal data, was introduced by the European Union. Companies, especially those involved in the Android ecosystem, are under pressure from these restrictions to give privacy protection top priority.

User wants and expectations: Users are more concerned about their privacy than ever before, and they expect the software and hardware they use to protect their private data. As a result, there is a rising need for Android smartphones to have privacy-focused features and controls.

Google has been aggressively working on improving privacy measures in the Android ecosystem to allay these worries and satisfy regulatory framework needs. Google has launched a number of significant programs and features, some of which are:

Android now comes with a Privacy Dashboard that gives users a summary of the permissions given to each app and the data those apps have accessed. Users now have greater access to and command over their data because to this.

Google has updated Android's permission mechanism to provide users greater granular control over the rights granted to apps. Users now have more control over their data and privacy thanks to the ability to give or refuse access on a feature-by-feature basis.

Improved app transparency: In order to guarantee transparency in how apps handle user data, Google has placed stronger standards on developers. The disclosure of the categories of data gathered and how it is used, as well as clear and unambiguous privacy policies, are requirements for developers.

Scoped storage: Scoped storage is a feature of Android that limits an app's access to user data, improving security against unauthorized access.

Privacy-focused APIs have been developed by Google, enabling programmers to create apps with privacy in mind. These APIs make features like encrypted backups and on-device data processing possible.

The Android privacy protection framework is a continuous effort to address privacy issues and raise user confidence in the system.

RESEARCH METHODOLOGY

The methodology section outline the plan and method that how the study is conducted. This includes Universe of the study, sample of the study,Data and Sources of Data, study's variables and analytical framework. The detailsare as follows;

3.1 Analysis and instrumentation of the apk

The resources and permissions that the application needs are used to analyze it. It is done to analyze permissions in relation to API calls and application utility. In order to determine whether the program needs further rights in order to steal user data, the permit suggestions are employed. Permission set mining and cooperative filtering algorithms are frequently employed for this purpose. The data used to create the training set was gathered for several applications across various categories. Instrumentation tools that support APK tools to support the function to call the When the findings from the two algorithms are combined, Data Protection Service sends the result, which includes dangerous and special authorization. Both algorithms rely on the classification of an app's permissions depending on the Play Store category it belongs to and the data collection that has been obtained..

3.2 Data protection service

With the help of a broadcast receiver and the Android framework, users can register for events analytically and dynamically. The lifespan of a dynamic application component depends on whether it has enabled Context.registerReceiver () and Context.unregisterReceiver (). Static receivers have the same lifetime as the application and are defined in AndroidManifest.xml. To override the SDK call, the receiver employs a callback method, namely BroadcastReceiver.onReceive ().

3.3 Data collection

There are two parts to the data assortment strategy. Through the event and growth one information assortment application, the initial information assortment is completed. About 300 users have downloaded this app, and through that, information on 1,000 different programs has been gathered. The information was then checked for permissions, permissions granted by the user were retrieved, and the 0.5 probability rule was used to determine if a privilege is beneficial or detrimental. As a result, unique application information is added to the information whenever the algorithms that execute on the server provide value to it. This might help the dataset get better and the proposed framework develop over time.

3.3.1 Download APK files

When downloading APK files for a privacy protection framework for Android, it is important to consider certain factors to ensure data integrity, privacy, and security. Here is an elaboration of the steps involved in downloading APK files for your privacy protection framework:

- Source Selection: Identify trusted and reliable sources for downloading APK files. Consider reputable app stores, official developer websites, or other well-known platforms that provide legitimate and verified APK files. Avoid downloading from unofficial or unknown sources to minimize the risk of malware or compromised applications.
- App Selection: Determine the criteria for selecting the apps to include in your dataset. You may choose to focus on popular apps, apps from specific categories, or apps with a large user base. The selection criteria should align with the goals and objectives of your privacy protection framework.
- Legal Considerations: Ensure that your APK file downloads comply with copyright and intellectual property laws. Respect the terms and conditions set by the app developers and avoid any unauthorized distribution or use of their applications. If necessary, seek permission from the app developers or adhere to open-source licensing requirements.
- Automate the Download Process: If you are dealing with a large number of APK files, consider automating the download process to streamline efficiency. You can develop scripts or utilize existing tools to automate the retrieval of APK files from the selected sources. Be mindful of the terms of service and usage limits of the sources you are downloading from to avoid any violations.
- Metadata Collection: Along with downloading the APK files, collect additional metadata associated with each app. This metadata can include the app name, package name, version, developer information, and app category. This information will be useful for categorizing and analysing the apps in your privacy protection framework.
- Verification and Integrity Checks: Implement measures to verify the integrity and authenticity of the downloaded APK files. Perform checks such as verifying the file's digital signature, comparing the checksums, or using tools like Virus Total to scan for any potential malware or security risks. Ensuring the legitimacy of the APK files is crucial to protect users and maintain data quality.
- Storage and Security: Store the downloaded APK files securely to protect them from unauthorized access or tampering. Implement appropriate security measures, such as encryption and access controls, to safeguard the files and prevent any potential data breaches. Adhere to data protection and privacy regulations to ensure compliance with legal requirements.
- Documentation and Record-keeping: Maintain proper documentation of the downloaded APK files, including the source, download date, and any relevant metadata. This documentation will serve as a reference and provide transparency regarding the sources and origin of the data used in your privacy protection framework.
- By following these steps, you can ensure the secure and responsible downloading of APK files for your privacy protection framework. These measures help maintain data integrity, protect user privacy, and ensure compliance with legal and ethical considerations.

3.3.2 Install Andro guard

Installing Andro guard is a crucial step in the data collection process for a privacy protection framework for Android. Andro guard is an open-source tool that provides functionalities for analysing Android applications and extracting various information, including permissions. Here is an elaboration of the installation step:

- Obtain Andro guard: Visit the official Andro guard repository or website to obtain the necessary installation files. Andro guard is typically available as a Python package, so ensure that you have Python installed on your system.
- Set up the Environment: Create a suitable environment for installing Andro guard. It is recommended to use a virtual environment to isolate the dependencies and avoid conflicts with other Python packages on your system. You can use tools like virtual env or Anaconda to create and manage the virtual environment.
- Install Dependencies: Before installing Andro guard, ensure that you have all the required dependencies installed. Andro guard relies on various libraries and tools, such as libxml2, libxslt, pygraphviz, and others. Refer to the Andro guard documentation or installation instructions for a complete list of dependencies and their installation instructions.
- Install Andro guard: Once the environment and dependencies are set up, you can proceed with installing Andro guard. Use the package manager or pip, the Python package installer, to install Andro guard within your virtual environment. Run the appropriate command, such as pip install Andro guard, to install the latest version of Andro guard.
- Verify Installation: After the installation is complete, verify that Andro guard is properly installed by running a simple test script or using the Andro guard command-line interface. This ensures that Andro guard is functioning correctly and ready for use in the subsequent steps of your data collection process.
- Keep Andro guard Updated: It is important to keep Andro guard updated with the latest version to benefit from bug fixes, performance improvements, and new features. Periodically check for updates and follow the recommended upgrade procedures to ensure you are working with the most recent version of Andro guard.

By following these steps, you can successfully install Andro guard, which provides you with the necessary tools and functionalities to analyse APK files and extract permissions for your privacy protection framework. Andro guard serves as a powerful resource in understanding the behaviour of Android applications and gathering important data for privacy-related analysis**.**

3.3.3 Extract permissions

Using the Andro guard tool, you can extract the permissions declared by each APK file. By analysing the APK files, Andro guard identifies the permissions requested by the Android applications. These permissions represent access rights to various device resources or sensitive data. When extracting permissions, Andro guard parses the APK file's manifest, which contains essential information about the application, including its permissions. The tool retrieves these permissions and provides them as output.

3.3.4 Save data to Excel

As you extract permissions from each APK file using Andro guard, save the data to an Excel file. Each row in the Excel file represents an APK file, and the columns contain the relevant information such as the app name, app category, and the extracted permissions. To save the data, you can utilize libraries or modules available for working with Excel files in your programming language of choice. These libraries provide functionality to create and write data to Excel files programmatically

3.3.5 Repeat for all APK files

Iterate through all the downloaded APK files and repeat the process of extracting permissions using Androguard for each file. This ensures that you extract permissions from every APK file in your dataset.By going through this iterative process, you will extract permissions, retrieve app names and app categories, and save the data to the Excel file for each APK file.Once you complete these steps, you will have an Excel file containing the extracted permissions, along with the corresponding app names and app categories, for the downloaded APK files. This dataset serves as the foundation for subsequent steps in your privacy protection framework, such as collaborative filtering or frequent permission set mining, where you can analyse the extracted permissions to derive insights and build privacy protection mechanisms.

3.4 Analysis and instrumentation

Decompiling the app exploitation Apktool, which is used to reverse-engineer automaton apps, is the initial stage. The decompiled programme is run in assembly code called Smali by the Dalvik Virtual Machine, which is a component of Android's Java Virtual Machine. The parser and instrumentation engine processes the decompiled code in the following steps. The application is afterwards repackaged using Apktool. Information Protection Service loads it onto the user's automated phone.

3.4.1 Permission analysis

Cooperative filtering and regular permission set mining are the two methods that were most frequently used to determine the permissions that an app requires. The coaching package includes apps from every category available in the Google Play store**.**

3.4.2 Collaborative filtering

(i) Finding the feature vector within the projected engine:

The collaborative filtering engine uses feature vectors based on the app permissions An application's permissions are retrieved as an enormously long vector, V = P1, P2,..., Pn >, where Pi might take values between [0, 1] depending on whether the programme accepts the given permission. We extract feature vectors from the coaching information set's apps. The engine first pulls the applications from the same class as the examination app. Then, for filtering and recommendation, all of the permissions are retrieved from the feature vectors. Ai = AP1, AP2, AP3,..., and APn where APi gets the value from the set [0, 1].

(ii) Evaluation of similarity:

A similarity score can indicate how closely related two things are. The Jaccard similarity score is used to determine how similar an app is to all other apps in its class: S (Ai, At) = F11 / (F01 + F10 + F11).

Ai is the application from the same-class information set. The test app can be found at. F11 is the frequency of matches between Ai and At's permissions. In the case of Ai, F01 is the frequency of permissions one, while in the case of At, F01. The frequency of permissions at frequency 10 is one in the case of At and zero in the case of Ai.

(iii) Recommendation of Permissions

The app At generates a recommendation score for each permission request. Victimization could be determined as RScoreCF (Pi) = S (Ai, At). In this case, the vote of the majority is taken into consideration, with the weight of the vote proportionate to the similarity score obtained higher than. The RScore that is produced is normalised. if the app is recommended to use it, the permission is listed as safe; if not, it is considered unsafe.

3.4.3 Frequent permission set mining:

Support is used to identify connections between entities.Let's say that an event (event B) chosen from a dataset of N events occurs f times (frequency of event B).Sup (B) = Frequency (B) / N is the formula for event B's support.Look at the project's permissions in the app. Depending on whether the app requests the permission, Pi will use values from the set [0, 1] at the square measured extracted in V = P1, P2,..., Pn >. The permissions of coaching knowledge apps are included in vectors with the formula Ai =, where APi selects values from the range [0, 1].

IV. RESULTS AND DISCUSSION

Based on the provided information, it appears that a permissions analysis was conducted on the Brightest Flashlight and Peacock Flashlight applications. The analysis revealed that these apps had vulnerabilities related to the LOC (location) and READ PHONE STATE permissions. The investigation involved instrumenting the app and observing its runtime interactions with the background service.During the analysis, it was discovered that the apps received false location information. However, after instrumenting the app, it was found to function as intended, indicating that the false location information was likely a result of the testing procedure rather than an actual vulnerability.

Following the completion of the entire procedure, the suggested framework produced the following outcomes:A flashlight app typically requires access to the camera to function properly. This suggests that the camera permission is necessary for the flashlight functionality.The remaining permission requests of the two applications (Brightest Flashlight and Peacock Flashlight) were deemed dangerous. It is implied that these dangerous permissions were identified during the analysis but not explicitly mentioned in the provided information. The results of these applications were added to a dataset for later use, possibly for further analysis or reference.The application (presumably one of the flashlight apps) was repackaged and instrumented to restore its original functionality while ensuring the protection of user information that could potentially be exploited for illicit purposes. This suggests that measures were taken to secure the app and prevent unauthorized use or misuse of user data.

Additionally, it is mentioned that three similar apps in the same category and utility were studied to analyze their activity patterns. However, these three apps behaved differently due to variations in the permissions they were granted, which did not align with their stated functionality. The entitlement recommender recommended new entitlements for two of these apps, but the measurements showed no change in their behavior, indicating that their functionality had not been affected. Furthermore, the user's location was protected from potentially harmful Android applications during this process.

It is important to note that the information provided is somewhat fragmented and lacks specific details, making it difficult to provide a comprehensive analysis or draw definitive conclusions. A more detailed and complete description would be necessary to thoroughly assess the impact of the permissions vulnerabilities and the effectiveness of the suggested framework.Based on the provided information, it seems that the focus of the discussion shifted from the flashlight applications to studying the activity patterns of three apps in the same category and utility. Therefore, there is no direct mention of a Flappy Bird application or the removal of permissions specifically from that app.However, if we consider the general scenario of removing permissions from a Flappy Bird application, we can provide a hypothetical analysis.

Flappy Bird is a simple game that involves guiding a bird through obstacles. Typically, a game like Flappy Bird would not require sensitive permissions such as location or phone state access. These permissions would be considered unnecessary for the core functionality of the game.

If the Flappy Bird application had unnecessary permissions that were identified as potential vulnerabilities, the removal of those permissions would likely have the following outcomes:

- Enhanced Privacy: By removing unnecessary permissions, the application would no longer have access to sensitive user information, such as location or phone state. This would help protect user privacy by minimizing the data that the app can collect or interact with.

- Reduced Attack Surface: Removing unnecessary permissions reduces the attack surface of the application. By limiting the permissions to only what is essential for the game's functionality, the potential vectors for exploitation or misuse of user data are minimized.

- User Trust: Removing unnecessary permissions can increase user trust in the application. When users see that an app requests only the permissions it truly needs, they may feel more confident in using the app, knowing that their privacy is being respected.

The second rule of suggestion is based on anticipating permission combine values that happen concurrently. The relationships and patterns of the simultaneous requests for authorization are investigated. Regarding the connection that two permissions have, permissions are advised for joint applications. Frequency is the foundation for the preliminary support estimate in event

It's important to note that the specific impact of removing permissions from an application would depend on the nature of the permissions being removed and the design of the app itself.If the permissions were genuinely unnecessary and not utilized by the Flappy Bird game, their removal would likely have a positive effect on privacy and security.

However, without more specific details about the permissions requested by the Flappy Bird application or the context in which they were identified as vulnerabilities, it is challenging to provide a more precise analysis.

A. STATIC ANALYSIS

Static analysis is the first stage in this procedure. The permissions and classes declared in the AndroidManifest.xml and their corresponding methods from the Smali code are delivered via the engine's Smali parser. A map displaying method tracing and data flow is derived from the Python parser, as explained in section V of Analysis and Instrumentation. AP.WRITE EXTERNAL STORAGE and READ PHONE STATE. Similar code analyses of the two applications reveal that Peacock Flashlight has risky permissions while AP does not.

B. PERMISSION ANALYSIS

The permission recommendation algorithm receives permissions parsed as described in the preceding subsection as input. Each permission is assessed by the algorithm, which then generates result vectors. Each result vector has 9 entries, each of which can either be 0 or 1. If not, the value is zero. Run the Brightest Flashlight recommender with a 0.1 threshold for collaborative filtering. The generated vector looks like this:

rp denotes the resulting permissions and is equal to [0, 0, 0, 0, 0, 0, 0, 0].

Each privilege's RScoreCF is below the threshold. As a result, none of his permissions are needed for this app, and all three of his permissions—AP.ACC FINE LOC, AP.ACC COARSE LOC, and AP.READ PHONE STATE—are considered unsafe. The support for each permission was estimated and graded against the average value using frequently used permission set mining as detailed in Section V. The result vector can be obtained as follows:

rp = [0, 0, 0, 1, 1, 0, 0, 0, 0].This is the Access Grant AP ACC FINE LOC; safe are AP.ACC COARSE LOC, but 'AP. The three extra rights that the brightest flashlight has to have are as follows: Permission Execution of AP.ACC FINE LOC and AP is advised for Peacock Flashlight. I found out you need the dangerous.ACC COARSE LOC permission. Since this app didn't need any additional permissions, Splendid Torch didn't perform a privilege analysis.

C. INSTRUMENTATION AND FINAL RESULTS

We discovered that the flashlight application's LOC and READ PHONE STATE permissions were vulnerable during the permissions analysis stage. Target devices were outfitted with instruments and installed with Brightest Flashlight and Peacock Flashlight. The instrumented app had runtime interactions with the background service. The programme received false location information, but following instrumentation, it was discovered to function as intended. Following the completion of the entire procedure, the suggested framework produced the following outcomes:

1) To function, a flashlight app needs access to the camera.

2) The two applications' remaining permission requests are deemed dangerous. Applications' results were added to the dataset for later use.

3) The application was repackaged and instrumented to restore its original functionality while safeguarding user information that might have been utilised for illicit purposes.

To study the activity patterns of three apps in the same category and utility, we looked at them. However, three similar apps worked differently since they were given various permissions that had nothing to do with the functionality that was actually listed. Two applications need new entitlements, according to the recommendations made by the entitlement recommender. Our measurements revealed that they had not changed in behaviour, which suggests that their functionality had not been impacted. The user's location was also shielded from his possibly harmful Android applications at the same time.

I. ACKNOWLEDGMENT

As the very outset we would like to give the first honors to God, who gave the wisdom and knowledge to complete this project.Our extreme thanks to Dr. Sreepriya S., Principal for providing the necessary facilities for the completion of this project in our college.We sincerely extend our thanks to Prof. R. Rajaram, Dean Projects & Consultancy for all the help, motivation and guidance throughout the completion of this project.We also like to extend our gratitude to Prof. Manesh T. HOD CSE for all the help, motivation and guidance throughout the project.We wish to extend our sincere thanks to the project coordinators Prof. Divya K.S. and Prof. Gripsy Paul and our Project Guide Prof. Sabitha M.G, for their valuable guidance and support throughout the project.We also wish to thank all teaching and non-teaching faculty of Computer Science and Engineering department for their cooperation. We also thank our Parents, friends and all well-wishers who supported directly or indirectly during the course of this project.

REFERENCES

[1] SHUANGXI HONG, CHUANCHANG LIU, BINGFEI REN, YUZE HUANG, AND JUNLIANG CHEN State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China Received March 5, 2017, accepted April 15, 2017, date of publication April 18, 2017, date of current version May 17,2017. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7903654

[2] Dam Skillen and Mohammad Mannan Concordia Institute for Information Systems Engineering Concordia University, Montreal, Canada {a skil, mmannan}@ciise.concordia.ca https://users.encs.concordia.ca/~mmannan/publications/mobiflage-ndss2013.pdf

[3] Obiri-Yeboah, J., & Qi, M. (2016). Data security of Android applications. 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). doi:10.1109/fskd.2016.7603436

[4] AUTHOR : Alireza Sadeghi Department of Informatics University of California, Irvine, USA alirezs1@uci.edu Reyhaneh Jabbarvand Department of Informatics University of California, Irvine, USA jabbarvr@uci.edu Negar Ghorbani Department of Informatics University of California, Irvine, USA negargh@uci.edu Hamid Bagheri Department of Computer Science an Engineering University of Nebraska, Lincoln, USA bagheri@unl.edu Sam Malek Department of Informatics University of California, Irvine, USA malek@uci.edu
Link : https://dl.acm.org/doi/pdf/10.1145/3180155.3180172.

[5] Braga, Alexandre & Colito, Alfredo. (2014). Adding Secure Deletion to an Encrypted File System on Android Smartphones. 106-110.

[6] Authors : RicardoNeisse Gary Steri Dimitris Geneiatakis IgorNai FivinoLink :https://reader.elsevier.com/reader/sd/pii/S0167404816300840?token=1FA9BEE0205EAE0A7C749511195ABC1211BF337328EA09A4CFDC584CBDC089AEB996E3FA770DAA2CF1748BD349383F9F&originRegion=eu-west-1&originCreation=20221029041109