****

**PROJECT REPORT**

**ON**

**“REVIEW PAPER ON APP PERMISSION EVOLUTION & USER PRIVACY ”**

**Submitted To**

**School of Cyber Security & Digital Forensics,**

**National Forensic Sciences University**

**For partial fulfilment for the award of degree**

**MASTER OF SCIENCE**

**In**

**CYBER SECURITY**

**Submitted By**

**RABARI PRADIPBHAI RAMESHBHAI**

**012200300002024**

**Under the Supervision of**

**Dr. PARAG SHUKLA**

**Assistant Professor**

**National Forensic Sciences University,**

**Gandhinagar Campus, Gandhinagar – 382009, Gujarat, India.**

**July, 2023**

**DECLARATION**

I , Rabari Pradipbhai Rameshbhai certify that

a. The work contained in the dissertation is original and has been done by myself under the supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

d. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the dissertation and giving their details in the references.

e. Whenever I have quoted written materials from other sources and due credit is given to the sources by citing them.

f. From the plagiarism test, it is found that the similarity index of whole dissertation within 25% and single paper is less than 10 % as per the university guidelines.

**Date:**

**Place: NFSU , GANDHINAGAR**

**Rabari Pradipbhai Rameshbhai**

**Enrollment No. :- 012200300002024**

**CERTIFICATE**

This is to certify that the work contained in the dissertation entitled **“ APP PERMISSION EVOLUTION & USER PRIVACY ”**, submitted by **RABARI PRADIPBHAI RAMESHBHAI (Enrollment No. :- 012200300002024)** for the award of the degree of **Master of Technology in Cyber Security and Incident Response** to the **National Forensic Sciences University**, **Gandhinagar Campus**, is a record of bonafide research works carried out by him under my direct supervision and guidance.

**Date:**

**Place: NFSU , GANDHINAGAR**

**Dr. Parag Shukla**

**Assistant Professor,**

**School of Cyber Security & Digital Forensics,**

**National Forensic Sciences University,**

**Gandhinagar Campus, Gujarat, India.**

**ACKNOWLEDGEMENTS**

I extend my sincere gratitude to all those who have contributed to the completion of my dissertation work. Your support, guidance, and encouragement have been invaluable throughout this research journey.

First and foremost, I express my heartfelt thanks to my project mentor, Dr. Parag Shukla, for their unwavering support, expertise, and valuable insights. Their mentorship and guidance have shaped the direction and focus of my research. I am truly grateful for their patience, dedication, and feedback, which have enriched the quality of my work.   
 I am also grateful to the head of the department, Dean Dr. Naveenkumar Chaudhary , for their encouragement and support. Their vision and leadership have created an environment conducive to research and learning, enabling my growth as a researcher & critical thinker .

Furthermore, I acknowledge my lab mates and colleagues for their collaboration, stimulating discussions, and support. Their insights and willingness to share expertise have enriched my research experience.

I also thank the individuals and organizations , individual researchers whose contributions were essential for the successful completion of this research. Your assistance and cooperation in providing necessary resources, research materials, and technical support are deeply appreciated.

Finally, I am grateful to my friends and family for their belief in me and continuous support. Your encouragement, understanding, and love have been my source of strength.

This acknowledgement is a reflection of my gratitude and appreciation towards all individuals who have played a role in my research work. I acknowledge their contributions with utmost respect and professionalism. Any controversial statements or non-academic sentiments have been excluded, adhering to the provided guidelines.

Once again, I extend my heartfelt thanks to everyone involved, as your contributions have been invaluable to the successful completion of my dissertation.

With Sincere Regards,

**RABARI PRADIPBHAI RAMESHBHAI**

**M.Sc.(Cyber Security)**

**NFSU , GANDHIINAGAR**

**ABSTRACT**

The evolution of Android permissions in android 10 & below which has 47.19% market share as of H2 of 2022 android has been a significant area of study in mobile operating systems due to less permission security oriented approach . Several research papers have addressed various aspects of this topic, highlighting the importance of user knowledge, security issues, privacy risks, and permission recommendation systems. In this Review Summary, I am trying to provide an overview of the key findings and recommendations from multiple abstracts to create a comprehensive abstract for a review paper. The analysis of user’s knowledge of Android permissions reveals a lack of understanding among users about permission granting , leading to uninformed decision-making during app installations. To address this issue, a conceptual model/APP is recommended to protect users from harmful applications and their developers, without disrupting their daily activities. Additionally, a comprehensive study of the Android permissions system demonstrates continuous growth in the number of permissions over time, accompanied by security concerns. The study also examines the evolution of permissions usage in top Android apps, identifying an increasing in permission requests, including potentially overprivileged ones. Furthermore, the potential impact of individual permissions on user privacy is quantified, emphasizing the complexity that arises when multiple permissions are granted. The prevalence of unnecessary or redundant permissions requested by "free" apps is highlighted, along with strategies to discourage such behaviour. Another study explores how apps evolve in their permission requests across different releases, revealing an increasing trend in permission requests and shedding light on potential biases in popular tools' reporting on overprivileged apps. Overall, these review paper will highlight the importance of user awareness, security enhancements which has done over the years, privacy considerations, and permission recommendation systems in the evolution of Android permissions.We also have done analysis of App Permission Analysis app available on google play store to evaluate user privacy seveeirty. The findings contribute to a better understanding of the permissions landscape, offering valuable insights for researchers, developers, and Android users alike who are still .

**Keywords: Android permissions, Mobile operating systems, Permission security, User knowledge, Security concerns, Privacy risks, Permission recommendation systems, Overprivileged permissions ,Permission evolution, User awareness .**

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| AAA | Authentication, Authorization, and Accounting |
| CIA | Confidentiality, Integrity & Availability |
|  |  |
| CIA |  |
|  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Description** | **Page No** |
| Table 1.1 | HTTP Return Code (Sample Text) | 27 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig No** | **Figure Description** | **Page No** |
| Figure 1 | Use Case | 18 |
| Figure 2 | Entity Relationship Diagram | 19 |
|  |  |  |

**LIST OF SCREENSHOTS**

|  |  |  |
| --- | --- | --- |
| **Screenshot No** | **Screenshot Description** | **Page No** |
| Screenshot 1 | Welcome Screen | 35 |
| Screenshot 2 | Home Page | 35 |
|  |  |  |

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acknowledgement | | | | III |
| Abstract | | | | IV |
| Abbreviations | | | | V |
| List of Tables | | | | VI |
| List of Figures | | | | VII |
| List of Screenshots | | | | VIII |
| List of Symbols | | | | IX |
| **Chapter 1.** | **Introduction** | | | **1-X** |
|  | 1.1 | Purpose , Introduction of Android Architecture | |  |
|  | 1.2 | User Privacy | |  |
|  | 1.3 | Scope of the Project | |  |
| **Chapter 2.** | **Literature Survey** | | | **2-X** |
|  | 2.1 | Current/Existing System | |  |
|  |  | 2.1.1 | Study of Current System |  |
|  |  | 2.1.2 | Problem & Weakness of Current System |  |
|  | 2.2 | Requirements of New System | |  |
|  | 2.3 | Feasibility Study | |  |
|  |  | 2.3.1 | Technical Feasibility |  |
|  |  | 2.3.2 | Operational Feasibility |  |
|  | 2.4 | Tools/Technology Required | |  |
| **Chapter 3.** | **Design: Analysis, Design Methodology and Implementation Strategy** | | | **3-X** |
|  | 3.1 | Function of System | |  |
|  |  | 3.1.1 | Use Case Diagram |  |
|  |  | 3.1.2 | Activity Diagram |  |
|  |  | 3.1.3 | Sequence Diagram |  |
|  | 3.2 | Data Modelling | |  |
|  |  | 3.2.1 | Entity-Relationship Diagram |  |
|  |  | 3.2.2 | Class Diagram |  |
|  | 3.3 | Functional & Behavioural Modelling | |  |
|  |  | 3.3.1 | Data Flow Diagram |  |
|  |  | 3.3.2 | Data Dictionary |  |
| **Chapter 4.** | **Implementation** | | | **4-X** |
|  | 4.1 | Implementation Environment | |  |
|  |  | 4.1.1 | Model Used in Developing |  |
|  |  | 4.2.2 | Software Prototyping Types |  |
|  | 4.2 | Coding Standard | |  |
|  | 4.3 | Laboratory Setup | |  |
|  | 4.4 | Tools and Technology Used | |  |
|  | 4.5 | Screenshots/Snapshots | |  |
| **Chapter 5.** | **Summary of Results and Future Scope** | | | **5-X** |
|  | 5.1 | Advantages/Unique Features | |  |
|  | 5.2 | Results and Discussions | |  |
|  | 5.3 | Future Scope of Work | |  |
| **Chapter 6.** | **Conclusion** | | | **6-X** |
| **Bibliography- List of references** | | | |  |
| **Appendices (if Any)** | | | | I |
| **List of Publications/Online Reference/etc..** | | | | II |

1. **INTRODUCTION**

**1.1 Purpose**

Android Operating System (OS) has gained significant popularity, dominating 74.5% of the mobile device market. This has led to a surge in the development of third-party applications by both individual developers and companies. Android's open-source nature and unrestricted application distribution have provided developers with opportunities to create and distribute their own apps. The platform offers a comprehensive Application Programming Interface (API), enabling access to system resources and device hardware. To safeguard user privacy, the Android platform enforces a permission-based model that requires applications to request access to sensitive information. During the installation process, users are prompted to grant explicit permissions to the app. However, users often accept these permissions without fully understanding their implications, which can lead to privacy invasions and malicious activities. Thus, the effectiveness of Android's permission system is crucial for maintaining security. **Previous studies on Android permissions have mainly focused on static analysis techniques and have not covered the latest Android versions**. **This paper aims to fill this gap by providing a comprehensive analysis of the Android permissions system, considering all its components and versions.** The study include a discussion of the user-permissions model in android operating system .   
  
**1.1.2 Users Privacy & Smartphones**

In today's modern world, smartphones have become an integral part of our daily lives, serving not only as mobile communication devices but also as powerful tools for managing various aspects of our personal and professional lives. With the ever-increasing number of smartphone users, these devices are poised to reach a significant majority of the global population. As smartphones become smarter and more capable, they play an increasingly significant role in our daily routines, and we entrust them with a substantial amount of our personal information. However, this concentration of personal data also makes smartphones attractive targets for businesses and service providers seeking to extract valuable insights into consumer behaviour. Therefore, it is crucial for individuals to understand how their personal data is accessed and utilized by these devices, as well as the potential impact on their privacy. Among the various operating systems designed for smartphones, Android stands out as a popular choice. Initially developed by Android Inc. in the early 2000s, Google acquired it in 2005 to further enhance its capabilities as an operating system and platform for mobile devices.

**2. Literature Survey**

**2.1 Operating System of Android**

Android operates on the foundation of the Linux kernel, offering robust security measures at the operating system level. It facilitates secure communication between applications running in separate processes through a reliable inter-process communication facility. Each application operates within a distinct identity, benefiting from the security features provided by the Linux kernel. Android employs a multiuser operating system approach, isolating user resources from one another. By adopting the underlying Linux security model and philosophy, Android aims to safeguard user resources from unauthorized access, ensuring a secure mobile computing environment. To establish a secure environment, Android assigns a unique user ID to each application and runs it as a separate process, preventing interactions between applications and restricting their access to the operating system. This separation is achieved through an application sandbox at the kernel level, reminiscent of the classic UNIX approach, where processes and file permissions are strictly segregated.

A diagram of a system

Description automatically generated with medium confidenceBelow Figure represent the android framework architecture.

**2.2 Privacy & Android Devices**

The availability of applications in the Google Play Store, formerly known as Android Market, has witnessed exponential growth. In just a few years, the number of available apps has surged from one million in July 2013 to over two million by February 2016. In response to this rapid expansion, Google implemented a policy requiring app developers to provide a detailed privacy policy when submitting their apps to the Google Play Store. Furthermore, Google intends to present privacy policies explicitly to customers, allowing them to review the relevant information before downloading and installing an app.

**2.3 Security Model of Android O.S.**

Android employs a permission-based security model to regulate app access to user's private data. At the API level, each app must request specific permissions to access various resources on the user's device. These permissions are declared in the app's AndroidManifest.xml file, clearly indicating the functionality and sensitive data required by the app. During the installation process, Android prompts the user when an application necessitates access to certain resources, such as location, media, or device identity. If the user denies these permissions, the application is not installed. However, the static declaration of permissions fails to inform users about the potential privacy implications of granting those permissions. Users may not be aware of how their sensitive data will be utilized once an app is installed. For instance, a mapping application might request access to the user's location to provide real-time directions. However, without proper insight into how the application handles this location data, the user's privacy could be compromised. Simply presenting users with a list of required permissions does not adequately inform them about the consequences of granting those privileges, leaving their privacy vulnerable to exploitation. Below is the figure showing the official releases of android till 11 & the permissions severity.  
A picture containing text, number, font, receipt

Description automatically generated

A diagram of a security enforcement

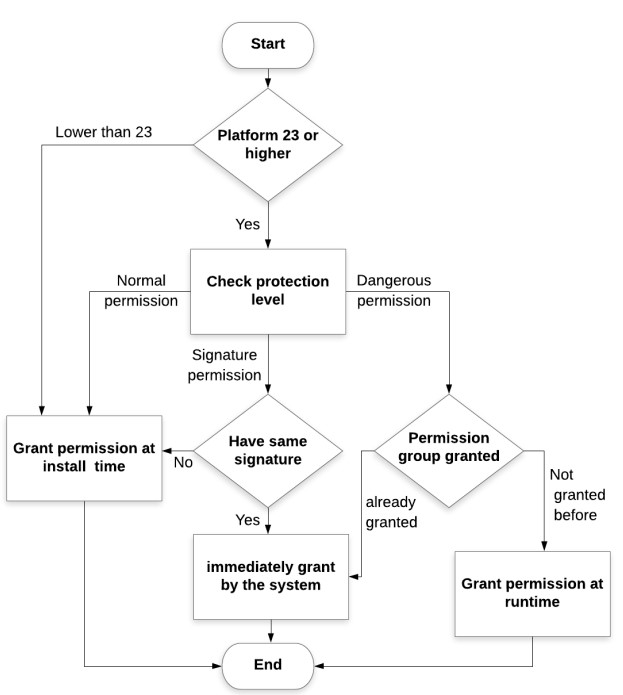
Description automatically generated with medium confidenceBelow Image figure present the android Security Model:

**2.4 Permissions in Android Architecture & User’s Privacy**

Privacy and security are two distinct yet interconnected concepts that are of paramount importance in the digital age. Privacy refers to an individual's ability to control their personal information and make decisions regarding its sharing and usage, while security focuses on safeguarding personal data from unauthorized access or intrusion. In the context of Android devices, app permissions play a critical role in both privacy and security. However, a significant number of Android users remain unaware of how app permissions can compromise their privacy and security. Many users neglect to review the list of permissions requested by an app before downloading it, often unaware of the sensitive information being accessed without their knowledge or consent. Compounding this issue is the presentation of permissions by app developers, which typically lacks transparency regarding the scope, type of data accessed, and potential privacy implications associated with each permission. The Android operating system addresses app permissions through its security architecture, which categorizes permissions into different levels. The most notable permission levels include   
A:- Normal Permission level :- Normal permissions pose no significant threat to user privacy and are automatically granted upon app installation.

B:- Dangerous Permissions level :- Dangerous permissions, on the other hand, can compromise user privacy, and when requested, prompt a dialog box that allows users to grant or deny permission. However, if a user grants a dangerous permission, the app can subsequently access other dangerous permissions without further consent.

C:- Signature Permission level:- Requiring Android application packages (.apk) files to be signed with a certificate owned exclusively by the app developer. This certificate helps the Android OS identify the app's author. and user ID and file access permissions.

Below is the analysis & structure of Permission protection level

Furthermore, Android assigns a unique user ID to each app package during installation, ensuring data stored by an app is isolated and inaccessible to other packages using concept of sandbox. Despite these security measures, a notable drawback of the Android security architecture is the absence of a mechanism in the Android developer's APIs that allows developers to request permissions in a hierarchical manner, starting with the least significant permissions and progressively requesting more privileged ones based on a set of criteria. Such a system could offer a check-and-balance approach to protect user privacy. In this review paper, we will delve into the evolution and analysis of Android permissions, examining their implications for user privacy and security. We will explore the challenges posed by the current permission system, potential privacy breaches resulting from uncontrolled permissions, and possible avenues for improvement. By shedding light on these issues, we aim to raise awareness and encourage the development of more privacy-centric and secure practices in the Android ecosystem..

**2.5 Permission Group according to Android System**

Android developers have organized all dangerous permissions into a specialized group known as the permission group, which may also include normal permissions. The user receives a notification only when the app requests dangerous permissions, informing them about the type of permission being requested, but not disclosing how the app intends to utilize it. Table 1 consist of most of all permissions which MobSF & Android Developers consider Normal / Dangerous / Signature based access .

1:- Normal Permissions

|  |
| --- |
| ACCESS\_LOCATION\_EXTRA\_COMMANDS  ACCESS\_NETWORK\_STATE  ACCESS\_WIFI\_STATE  BLUETOOTH  BLUETOOTH\_ADMIN  CHANGE\_NETWORK\_STATE  CHANGE\_WIFI\_STATE  INTERNET  NFC  QUERY\_ALL\_PACKAGES  READ\_SYNC\_SETTINGS  READ\_SYNC\_STATS  RECEIVE\_BOOT\_COMPLETED  REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS  USE\_FULL\_SCREEN\_INTENT  VIBRATE  WAKE\_LOCK  WRITE\_SYNC\_SETTINGS |

2:- Dangerous Permission

|  |
| --- |
| ACCEPT\_HANDOVER  ACCESS\_BACKGROUND\_LOCATION  ACCESS\_COARSE\_LOCATION  ACCESS\_FINE\_LOCATION  ACTIVITY\_RECOGNITION  ANSWER\_PHONE\_CALLS  BODY\_SENSORS  CALL\_PHONE  CAMERA  GET\_ACCOUNTS  PROCESS\_OUTGOING\_CALLS  READ\_CALENDAR  READ\_CALL\_LOG  READ\_CONTACTS  READ\_EXTERNAL\_STORAGE  READ\_PHONE\_NUMBERS  READ\_PHONE\_STATE  READ\_SMS  RECEIVE\_MMS  RECEIVE\_SMS  RECEIVE\_WAP\_PUSH  RECORD\_AUDIO  SEND\_SMS  USE\_BIOMETRIC  USE\_FINGERPRINT  WRITE\_CALENDAR  WRITE\_CALL\_LOG  WRITE\_CONTACTS  WRITE\_EXTERNAL\_STORAGE |

3:- Signature Permission

|  |
| --- |
| BIND\_ACCESSIBILITY\_SERVICE  BIND\_AUTOFILL\_SERVICE  BIND\_CARRIER\_MESSAGING\_SERVICE  BIND\_CARRIER\_SERVICES  BIND\_CHOOSER\_TARGET\_SERVICE  BIND\_CONDITION\_PROVIDER\_SERVICE  BIND\_DEVICE\_ADMIN  BIND\_DREAM\_SERVICE  BIND\_INCALL\_SERVICE  BIND\_INPUT\_METHOD  BIND\_MIDI\_DEVICE\_SERVICE  BIND\_NFC\_SERVICE  BIND\_NOTIFICATION\_LISTENER\_SERVICE  BIND\_PRINT\_SERVICE  BIND\_QUICK\_SETTINGS\_TILE  BIND\_REMOTEVIEWS  BIND\_SCREENING\_SERVICE  BIND\_TELECOM\_CONNECTION\_SERVICE  BIND\_TEXT\_SERVICE  BIND\_TV\_INPUT  BIND\_VISUAL\_VOICEMAIL\_SERVICE  BIND\_VOICE\_INTERACTION  BIND\_VPN\_SERVICE  BIND\_VR\_LISTENER\_SERVICE  BIND\_WALLPAPER  MANAGE\_DOCUMENTS  SYSTEM\_ALERT\_WINDOW  USE\_SIP  WRITE\_SETTINGS |

4:- System Permission

|  |
| --- |
| ACCESS\_DRM\_CERTIFICATES  ACCESS\_MEDIA\_LOCATION  ACCESS\_NOTIFICATION\_POLICY  ACCESS\_PRINT\_JOB\_STATS  ACCESS\_UWB\_STATE  ACCESS\_WIFI\_PASSWORD  ACTIVATE\_VPN  ASSIGN\_TIME\_ZONE  BIND\_DIRECTORY\_SEARCH  BROADCAST\_NETWORK\_PRIVILEGED  CALL\_PRIVILEGED  CAMERA\_DISABLE\_TRANSMIT\_LED  CONFIGURE\_DISPLAY\_COLOR\_MODE  CONFIGURE\_DISPLAY\_SURROUND\_MODE  CONFIGURE\_WIFI\_DISPLAY  CONTROL\_DISPLAY\_BRIGHTNESS  CONTROL\_DISPLAY\_SATURATION  CONTROL\_DISPLAY\_VIBRATION  CONTROL\_KEYGUARD  CONTROL\_WIFI\_DISPLAY  DELETE\_CACHE\_FILES  DELETE\_PACKAGES  DIAGNOSTIC  DISABLE\_KEYGUARD  DUMP  FACTORY\_TEST  FILTER\_EVENTS  FORCE\_BACK  GET\_APP\_OPS\_STATS  GET\_DETAILED\_TASKS  GET\_TOP\_ACTIVITY\_INFO  GLOBAL\_SEARCH  GRANT\_RUNTIME\_PERMISSIONS  HARDWARE\_TEST  INJECT\_EVENTS  INSTALL\_GRANT\_RUNTIME\_PERMISSIONS  INSTALL\_LOCATION\_PROVIDER  INSTALL\_PACKAGES  INSTANT\_APP\_FOREGROUND\_SERVICE  KILL\_BACKGROUND\_PROCESSES  LOCATION\_HARDWARE  LOOPER\_STATS  MANAGE\_ACTIVITY\_STACKS  MANAGE\_APP\_TOKENS  MANAGE\_CA\_CERTIFICATES  MANAGE\_DEVICE\_ADMINS  MANAGE\_MEDIA\_PROJECTION  MANAGE\_OWN\_CALLS  MANAGE\_PROFILE\_AND\_DEVICE\_OWNERS  MANAGE\_USB  MANAGE\_VOICE\_KEYPHRASES  MODIFY\_AUDIO\_SETTINGS  MODIFY\_PHONE\_STATE  MTP\_WRITE\_PERMISSION  NFC\_HANDOVER\_STATUS  OVERRIDE\_WIFI\_CONFIG  PACKAGE\_USAGE\_STATS  PERSISTENT\_ACTIVITY  PROCESS\_CALLS  READ\_DEVICE\_CONFIG  READ\_DREAM\_STATE  READ\_INPUT\_STATE  READ\_LOGS  READ\_PRECISE\_PHONE\_STATE  READ\_PRIVILEGED\_PHONE\_STATE  READ\_RUNTIME\_PROFILES  READ\_WALLPAPER\_INTERNAL  REAL\_GET\_TASKS  REBOOT\_DEVICE  RECOVERY  REGISTER\_CALL\_PROVIDER  REMOTE\_AUDIO\_PLAYBACK  REMOVE\_TASKS  REQUEST\_COMPANION\_RUN\_IN\_BACKGROUND  REQUEST\_COMPANION\_USE\_DATA\_IN\_BACKGROUND  REQUEST\_DELETE\_PACKAGES  REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS  REQUEST\_INSTALL\_PACKAGES  RESET\_SHORTCUT\_MANAGER  RESTART\_PACKAGES  RETRIEVE\_WINDOW\_CONTENT  ROTATE\_DISPLAY  RUN\_ANY\_IN\_BACKGROUND  SEND\_RESPOND\_VIA\_MESSAGE  SEND\_SMS\_NO\_CONFIRMATION  SET\_ACTIVITY\_WATCHER  SET\_ALARM  SET\_ALWAYS\_FINISH  SET\_ANIMATION\_SCALE  SET\_DEBUG\_APP  SET\_INPUT\_CALIBRATION  SET\_ORIENTATION  SET\_POINTER\_SPEED  SET\_PREFERRED\_APPLICATIONS  SET\_PROCESS\_LIMIT  SET\_SCREEN\_COMPATIBILITY  SET\_TIME  SET\_TIME\_ZONE  SET\_WALLPAPER  SET\_WALLPAPER\_HINTS  SIGNAL\_PERSISTENT\_PROCESSES  START\_ANY\_ACTIVITY  START\_TASKS\_FROM\_RECENTS  STATUS\_BAR  SYSTEM\_ALERT\_WINDOW  TRANSMIT\_IR  UNINSTALL\_SHORTCUT  UPDATE\_APP\_OPS\_STATS  UPDATE\_DEVICE\_STATS  USE\_BIOMETRIC\_INTERNAL  USE\_SIP  USE\_SIP\_INTERNAL  VERIFY\_STORAGE  VIBRATE  WAKE\_LOCK  WRITE\_APN\_SETTINGS  WRITE\_DEVICE\_CONFIG  WRITE\_DREAM\_STATE  WRITE\_EXTERNAL\_STORAGE  WRITE\_GSERVICES  WRITE\_MEDIA\_STORAGE  WRITE\_SECURE\_SETTINGS  WRITE\_SETTINGS  WRITE\_SYNC\_SETTINGS  WRITE\_VOICEMAIL |

Apart from the permissions classified as dangerous by Android developers, there are other permissions that pose a significant threat to user privacy. For instance, granting an app permission to access the Wi-Fi connection enables it to transmit sensitive data to the app developer. Similarly, allowing access to user accounts or authentication information of email services or social media accounts puts personal information at risk. Many free apps heavily rely on advertisements as a means to generate revenue and sustain their business operations. However, this scenario often leads app developers to request additional permissions to gather user data. It is important to note that not all of these permissions are essential for the proper functioning of the respective app.

**2.6 Permission System Architecture & It’s Privacy Framework for Android Applications**

Android's core operating system is built on the robust Linux kernel, which provides a range of security features to safeguard smartphones. Security within the Android ecosystem follows a two-step process. Firstly, each application operates within its own user identity, thereby leveraging the Android system's inherent user isolation capabilities based on the security measures of the Linux system. Additionally, the Android middleware includes a reference monitor, as depicted in, which facilitates secure inter-component communication.  
ICC (Inter-Component Communication) Reference Monitor:- is a security mechanism implemented by the Android operating system to enforce secure communication between different app components. It ensures that only authorized components can interact with each other and prevents unauthorized access to sensitive data or functionality. Here are the unique points describing ICC Reference Monitor and its implementation in Android which has some key points like

Purpose: The ICC Reference Monitor serves as a guardian that regulates the communication between app components such as activities, services, broadcast receivers, and content providers. It acts as a mediator, enforcing security policies and preventing unauthorized interactions. [1]

Access Control: The ICC Reference Monitor enforces access control policies to determine which components can communicate with each other. It verifies the permissions required by the calling component and ensures that the target component has the necessary permissions to respond to the request.

Intent-based Communication: Android uses the Intent mechanism for inter-component communication. Intents are messages used to request actions or send information between components. The ICC Reference Monitor intercepts and examines intents to ensure that the intended communication is permitted and complies with the defined security policies.

Permission Checks: The ICC Reference Monitor checks the permissions associated with the requested operation before allowing communication between components. It verifies that the calling component holds the required permissions to access sensitive resources or perform specific actions. This prevents unauthorized components from accessing protected functionalities.

Security Labels: Android assigns security labels to app components and enforces security policies based on these labels. Components are categorized as either system or app components, with different privileges and access rights. The ICC Reference Monitor ensures that only authorized components can communicate with system-level components or access restricted resources. Runtime Enforcement: The ICC Reference Monitor enforces security policies at runtime, dynamically analyzing the communication flow between components. It allows secure communication channels to be established on-demand, adapting to the changing context of the app. This runtime enforcement helps protect against potential security vulnerabilities that may arise during the app's execution. Prevention of Privilege Escalation: The ICC Reference Monitor prevents privilege escalation by ensuring that components cannot impersonate or gain unauthorized access to other components. It validates the integrity of communication requests, preventing malicious attempts to bypass the security measures and exploit system vulnerabilities. Secure Inter-App Communication: The ICC Reference Monitor not only enforces security within an app but also extends its control to inter-app communication. It ensures that components from different apps can only communicate when authorized and with appropriate permissions. This prevents unauthorized data leakage or interference between apps. Platform Integrity: The ICC Reference Monitor is an integral part of the Android platform's security architecture. It is designed to be tamper-proof and resistant to malicious manipulation. By providing a centralized security enforcement mechanism, it helps maintain the overall integrity and trustworthiness of the Android ecosystem. By implementing the ICC Reference Monitor, Android establishes a robust security framework that governs inter-component communication, ensures user privacy, and protects against unauthorized access to sensitive data and system resources. It acts as a crucial safeguard, enhancing the overall security posture of the Android operating system. While Android offers security mechanisms, privacy remains an area that requires attention. To address this concern, the development of a privacy framework is imperative, empowering users to safeguard their personal information across diverse applications.

A picture containing text, screenshot, line, font

Description automatically generated

Introducing a counter-based notification system for Android users serves as a proactive approach towards enhancing privacy protection. This notification system focuses on Android permissions and encompasses various functionalities, including the recording of sensitive permissions, permission access counters, user notifications, and privacy settings. Each of these aspects is elaborated upon below: Recording of sensitive permissions: This feature keeps track of the sensitive permissions accessed by individual apps, ensuring transparency regarding the specific permissions utilized. Permission access counter: The permission access counter tallies the number of permissions accessed by different apps, providing insights into their respective permission requirements. User notifications: By employing notifications, this system promptly informs users whenever a particular app accesses sensitive permissions, such as personal information or data, enabling them to stay informed and take necessary actions. Privacy settings: Privacy settings grant users the flexibility to enable or disable specific permissions according to their preferences, thereby regulating app access to various permissions as deemed necessary. By implementing this counter-based notification system within the Android framework, users gain better control over their privacy, ensuring a more secure and personalized app experience.

**A picture containing text, screenshot, font, number

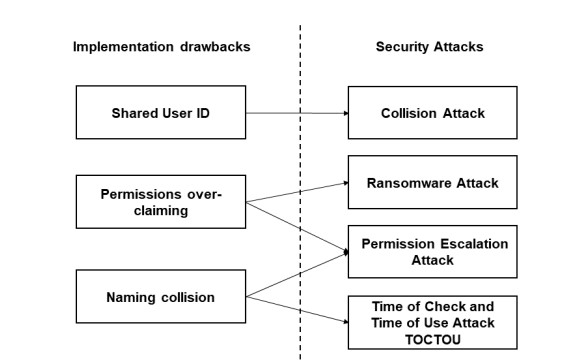
Description automatically generated**

To enhance privacy in the Android ecosystem, we propose a privacy-based notification system. This system empowers users to closely monitor the behaviour of each app. Whenever an app accesses a sensitive permission, the Android system generates a notification, promptly informing the user about the app's action. By implementing this system, users gain better control over the permissions requested by apps, ensuring that critical data remains protected.  
Here is the analysis of popular apps on internet & their Normal Permissions & Dangerous Permissions . [1]

**A picture containing text, menu, number, receipt

Description automatically generated**

**2.7 Android’s Permission related Security Problems**

The proliferation of Android OS functionality has brought about an increase in vulnerabilities. In response to these security concerns, Google has imposed restrictions on third-party applications, preventing them from accessing system-level APIs. This below section focuses on the advancements, issues, and implications of permission-based security. Topic:   
  
A:-Security\_Enhancment\_with\_Permission\_Based:  
  
To bolster protection, Android OS has implemented measures to limit third-party app access to system resources. As a result, newer versions of Android require explicit permission checks throughout applications. Additionally, an analysis of the Android permissions system reveals a continuous deprecation of permissions. Consequently, outdated applications relying on system permissions are incompatible with recent Android versions. Android 4 introduced a permission view that categorizes permissions into groups, providing users with an organized display. Users can still view brief descriptions of each permission within a group. A significant security enhancement to the permission system was the introduction of runtime permissions in Android 6.0. This feature empowers users to control access for both Marshmallow and pre-Marshmallow applications. Moreover, Android 8.0 implemented new access control measures, requiring users to explicitly grant permission to install applications from non-first-party market stores. Android 10 also introduced the Permission Controller module as a separate entity responsible for handling permission updates.

B: Security Issue’s of Permission based.

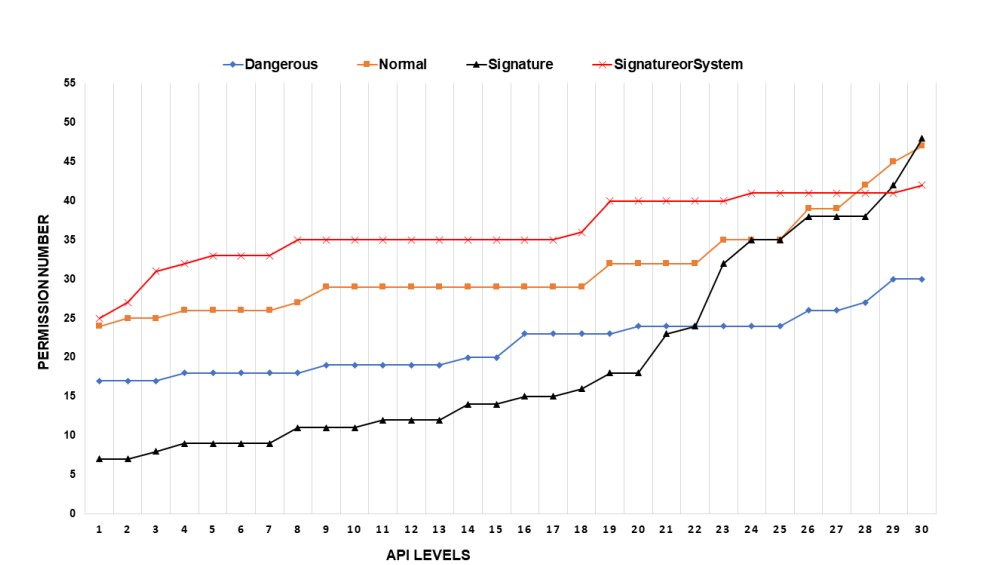
Despite the perceived security of the Android permission system, continuous enhancements have introduced various security issues and drawbacks .Access enforcement in the Android platform is subject to the targeted API level, making the system susceptible to abuse by applications aiming to boost their privileges. For instance, if an application targets an API level prior to Android 6.0, it will be granted all permissions, including dangerous ones, at install-time. Another concern is the presence of over-privileged apps. Such applications may request permissions they do not actually utilize. These security weaknesses expose the Android OS to various attacks, including collision attacks, permission escalation attacks, and ransomware attacks. Despites the relationship between implementation drawbacks and security attacks.   
  
**1:-** **Collision Attack** :- A collision attack occurs when two applications share the same User ID. The User ID, which corresponds to the package name, is a unique attribute found in the AndroidManifest.xml file. When two or more applications possess the same User ID and are signed by the same certificate, they can access each other's granted permissions as it called signature permission as they do belong from same family . For example, if application A and B have the shared User ID "Shared\_User\_ID," and application A is granted permissions to READ\_SMS

&

CALL\_PHONE while application B has permission to use CAMERA, both apps can utilize all three mentioned permissions due to their shared User ID, "Shared\_User\_ID".   
  
**2:-** **Escalation Attack :-** An escalation attack occurs when two applications collaborate to access sensitive data or system resources without explicitly requesting permissions. For instance, component (comB) of application B may be granted permission to access the resources of application A. Subsequently, application C can collaborate with application B to transgressively access the resources of application A. Once application B grants application C access to one of its components, application C gains access to resources in application A through component (comB).

**3:- TOCTOU Attack :-** The absence of naming restrictions in the Android permission system allows for the equivalence of two permissions with the same name, even if they belong to different apps. Suppose application A is granted permission p1, which provides access to critical system resources. A malicious app B could declare a permission p2 with the same name as p1. Consequently, the malicious app B can exploit permission p2 to gain unauthorized access to the critical system resources. Topic: Ransomware Attack Misuse of Android permissions and API packages renders Android applications and users vulnerable to ransomware attacks. Ransomware attacks can lock or encrypt Android users' devices or data, demanding ransom for their release.The permissions required to execute such attacks typically include   
SYSTEM\_ALERT\_WINDOW,  
 WAKE\_LOCK,\_&   
KILL\_BACKGROUND\_PROCESS.

**2.7 Problems with android permissions**

**2.7.1** Permission Evolution in permission protection level

**2.8 App permission analysis of Popular Apps**

In this section, we will conduct a comprehensive analysis of various Google apps that are highly rated and popular among Android users. The main objective of this analysis is to identify potential privacy threats associated with these apps when they are granted different permissions by users. A. Methodology To analyse the permission system, we have categorized it into ,

9: Broad categories: Location, Storage, Network Access, Messages, Phone Calls, Camera, User Accounts & Contacts, Hardware Controls, and System-related permissions. By evaluating eight apps from the Google app store, we aim to determine which categories require the most permissions and assess the privacy threats posed by each category .

Here is the selected popular Apps chosen :

App1: WhatsApp (Communication)

App2: Facebook (Social Media)

App3: Google Docs

App4: Spotify (Music and Audio)

App5: Duolingo (Education)

App6: YouTube (Video)

App7: Just Eat (Lifestyle)

App8: Amazon Shopping (Shopping)

Analysis:

App1: WhatsApp - It is a renowned messaging and calling service with an average rating of 4.4 and 1 billion downloads. Privacy concerns arise due to its permissions related to tracking location, sending data without notification, using SMS and MMS, accessing call logs, utilizing the camera and microphone, and accessing system settings.

App2: Facebook - As a popular social media service with 1 billion downloads and an average rating of 4, Facebook jeopardizes user privacy by requesting permissions such as location (network and GPS-based), messaging-related sensitive permissions, access to call logs, ability to make calls, access to the camera, reading and editing user contacts, microphone usage at any time, and sending files without user permission.

App3: Google Docs - This well-known business app has an average rating of 4.2 and 100 million downloads. During installation, it requests permissions that may raise privacy concerns for users, including accessing user accounts and reading users' contact information.

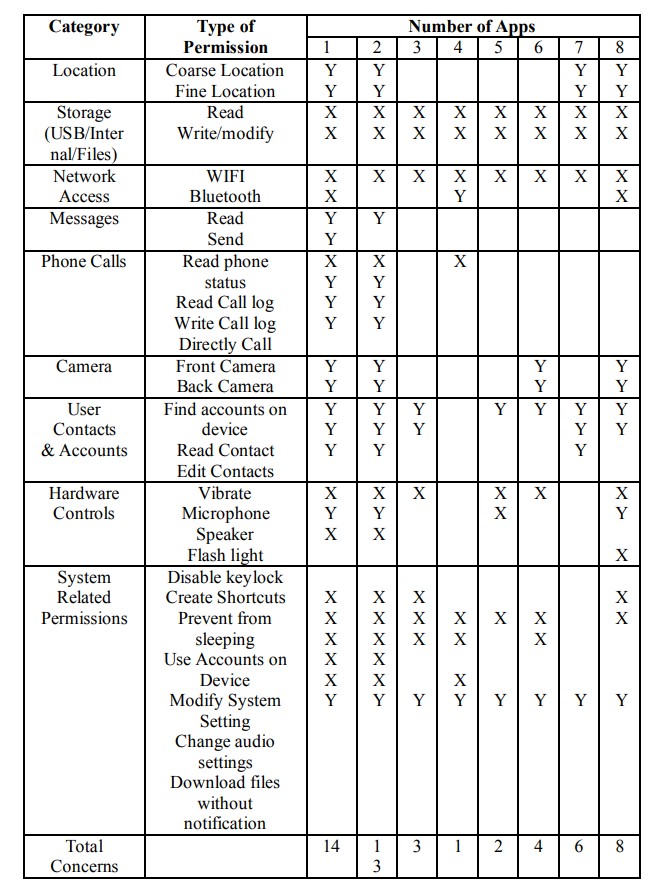
App4: Spotify - As a popular music app with a user rating of 4.5 and 100 million downloads, Spotify raises privacy concerns by requesting permission to access Bluetooth settings and exchange files over the internet without user knowledge.

App5: Duolingo - This famous language learning app has 50 million downloads and an average rating of 4.7. Privacy concerns arise from its requests for account information and downloading files without user consent.

App6: YouTube - As a widely-used video sharing app with a current user rating of 4.1 and 1 billion downloads, YouTube presents privacy concerns related to permissions for the camera, user account information, and downloading files.

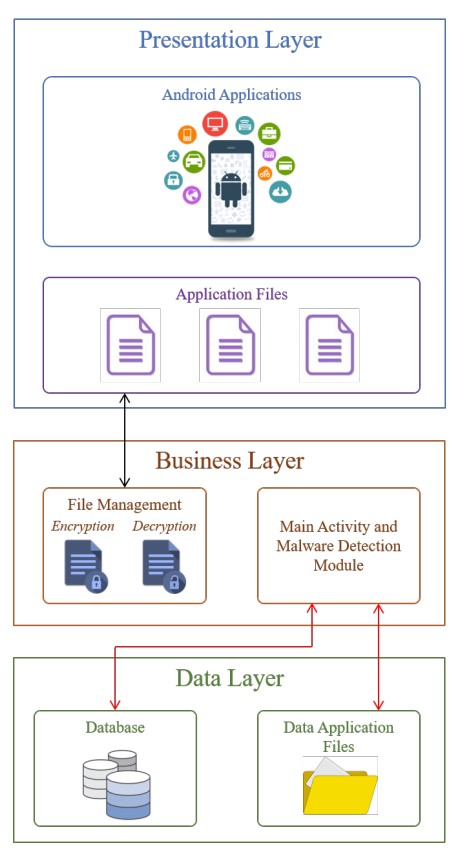
App7: Just Eat - Belonging to the lifestyle category, this app allows users to order food online. It enjoys a user rating of 4.6 with 100 thousand downloads. Privacy concerns arise from its access to sensitive permissions such as location, user account information, user contacts, and downloading files without user knowledge.

App8: Amazon Shopping - This online shopping app has a rating of 4.3 and 50 million downloads. Privacy threats for users emerge from permissions related to location, camera, user contacts and account information, microphone usage, and downloading files from the internet. In the table below, we present all the dangerous permissions categorized by Google as "bad behavior," denoted by Y. These permissions are considered threats that may invade user privacy. All other permission types are denoted by X in the table..

****

**2.8 Proposed Ideal Conceptual Model by**

Below Figure illustrates a novel security model aimed at enhancing the existing Android security framework. The ultimate objective is to ensure comprehensive protection against user-installed applications. This proposed model builds upon the established tiered application development model .



A. Encryption of Files:- The initial phase of the proposed system involves the enforcement of file encryption through the utilization of the Advanced Encryption Algorithm (AES). This algorithm employs a 128-bit key size to facilitate fast processing and is widely recognized as a highly secure encryption technique for both hardware and software applications. The File Management module within the Business Layer is responsible for executing this encryption process. All critical files are encrypted using this technique, rendering them highly resistant to unauthorized access. In the event that an application attempts to access an encrypted file, a security application programming interface (API) is triggered. This API conducts vulnerability scans, denying access to the application if any vulnerabilities are detected. Conversely, if no vulnerabilities are found, the API grants access by decrypting the file.

B. Advantages of the Proposed Security Model:- The proposed conceptual security model offers several significant benefits:

• Automated Encryption: The system automatically encrypts and decrypts user files stored on the device. Users are relieved from the task of creating passwords to protect specific files, as the system provides seamless encryption functionality. • File Access Restrictions: The model effectively restricts applications from accessing files belonging to other applications on the device. The objective is to ensure that, even if application A and application B are installed on the same device, each remains unaware of the other's existence.

• Advan

ced Malware Detection: The model utilizes state-of-the-art encryption techniques that include signature matching to detect and defend against malware threats.

• Non-intrusive Background Operation: The model operates in the background, ensuring that users can perform their activities without any disruption, all while benefiting from robust security features.

• Heuristic Learning: The model incorporates heuristic-based methods to proactively learn about potential new types of malware, further enhancing its ability to safeguard users' devices.

• Protection from Unknowingly Installed Malware: Users are safeguarded even if they unwittingly install a malicious application.

C. Malware Detection Techniques The system provides comprehensive security features by employing three generic methods for malware detection. These methods can be categorized as follows: signature-based, behavioural-based, and heuristic-based.

Signature-based Method: This technique relies on identifying patterns extracted from known malicious applications. Each signature serves as a unique identifier or fingerprint used to recognize applications. This method boasts a low error rate during utilization. However, it has limitations, such as requiring substantial time for signature extraction and its inability to detect unknown malware. To address these drawbacks, behavioural-based methods are employed.

Behavioural-based Method: This approach detects malware by analysing an application's behaviour, determining whether it exhibits malicious characteristics. Instead of focusing solely on the application's description, this method evaluates its actions. One of the primary advantages of this technique is its ability to identify malware that remains undetectable by signature-based methods. However, it also demands a considerable amount of time for detection, similar to the signature-based approach.

Heuristic-based Method: To overcome the shortcomings of signature- and behavioural-based methods, heuristic-based techniques are employed. These methods employ machine learning and data mining to gain insights into an application's behavior. Heuristic-based techniques necessitate some form of input for malware classification. In the proposed model, an API call sequence, utilizing Hamming distance, is adopted to detect malware. This approach enables the model to learn and identify new types of malware.

D. Innovations in Android Security :-

While Android's security model is inherently robust and secure, it is not impervious to malicious attacks, especially when users are required to make informed decisions regarding application permissions. Social engineering attacks, for instance, can manipulate users into installing and granting permissions to dangerous applications. Therefore, the focus of this research was to enhance Android's security model by designing a more resilient framework that prioritizes the end user's security. By doing so, potential attacks, including those driven by social engineering, can be mitigated. Even if a user were to install a malicious application, the proposed model's malware detection technique would identify the threat and render the data unreadable to the malicious application.

**2.9 Advancement & new features introduced in latest android versions!**

Here we will talk about the latest android version 13.

App sandboxing :- is a crucial aspect of Android's security architecture as it ensures that each app operates within its own isolated environment, safeguarding both its data and code from unauthorized access or interference by other apps. By confining apps to their respective sandboxes, Android mitigates the risk of data breaches and ensures that the apps cannot tamper with each other's functionalities.

Permissions:- play a pivotal role in maintaining user privacy and data security on Android devices. Apps are required to obtain explicit permission from the user before accessing sensitive resources, such as contacts, location, or microphone. This permission-based approach empowers users to control the data accessed by apps and prevents unnecessary or unauthorized access to personal information.

In order to enhance user control, Android introduced the photo picker feature, which allows users to selectively share specific photos or videos with an app rather than granting it unrestricted access to their entire media library. To bolster device security, Google regularly releases security updates for the Android platform. These updates address potential security vulnerabilities that may have been identified either in the platform itself or in installed apps. Promptly installing these updates is vital to ensure the ongoing protection of Android devices against potential threats.

Android devices offer encryption capabilities to safeguard data stored on the device. Encryption acts as a protective shield, rendering the data inaccessible in case the device is lost or stolen, thus mitigating the risk of unauthorized access and maintaining the confidentiality of sensitive information. Biometric authentication provides an additional layer of security by utilizing fingerprint scanning or facial recognition to verify the user's identity before granting access to the device or specific apps. This robust authentication mechanism prevents unauthorized users from accessing the device or sensitive app functionalities, enhancing overall security. Notification permission grants users the ability to manage which apps can display notifications, preventing apps from inundating users with intrusive or bothersome notifications. This empowers users to tailor their notification preferences and ensures a more streamlined and personalized experience. Android's granular media permissions allow users to exercise control over which apps can access specific media files, including photos, videos, and music. By granting explicit permission, users can prevent apps from accessing media files that are unnecessary for their intended functionalities, thus safeguarding their personal media content and maintaining privacy. The nearby Wi-Fi devices permission feature empowers users to regulate which apps can scan for nearby Wi-Fi devices. This control prevents apps from collecting information about nearby Wi-Fi networks without the user's knowledge or consent, reinforcing user privacy and data security. In summary, Android's evolving permission framework and security features, such as app sandboxing, permission-based access, security updates, encryption, biometric authentication, notification permission, granular media permissions, and nearby Wi-Fi devices permission, collectively contribute to a robust security ecosystem. These measures ensure user privacy, protect against unauthorized access, and maintain data integrity on Android devices, promoting a secure and trustworthy user experience..

**3.0 Runtime permissions Management & APIs**

In order to obtain a permission on the Android platform, an app needs to declare it in the manifest file using the <uses-permission> XML element. However, this declaration alone does not guarantee that the permission will always be granted after the app is installed. Legacy Android platforms offer APIs that allow apps to check their permission statuses at runtime. With the introduction of the runtime permission model, users now have the ability to dynamically grant and revoke dangerous permissions. Consequently, newer platforms (API level 23 and above) provide additional APIs to handle users' runtime behaviours. An example from the official documentation illustrates this process, involving permission check and request. The permission management APIs generally fall into four categories:

(a) Checking permission status: Before accessing a permission-protected API, an app will verify whether it has the necessary permissions by using the CHECK APIs (e.g., ContextCompat.checkSelfPermission()). These APIs typically require a permission string as input and return whether the permission has been granted or not.

(b) Requesting dangerous permissions: At runtime, an app can request dangerous permissions by calling the REQUEST APIs, such as ActivityCompat.requestPermissions(). This action triggers a pop-up dialog that prompts the user to grant or deny the permission request, or even block subsequent permission requests. Unlike the CHECK APIs, the REQUEST APIs accept an array of permission strings to handle multiple permission request dialogs.

(c)Handling user response: The HANDLE API, onRequestPermissionsResult(int, String[], int[]), is an empty callback defined in the base GUI classes. The system invokes this callback after the user responds to the permission request, and the parameters store the user's decisions. Developers can override this callback to examine the user's choices and take appropriate actions accordingly.

(d) Explaining permission usage: The EXPLAIN API, shouldShowRequestPermissionRationale(), returns a boolean value indicating whether the user has denied the permission request and selected the "Never ask again" option. Developers can use this API to determine if the permission requests are blocked and provide explanations to the user about why the requested permission is essential. By implementing these runtime permission management techniques, Android apps can handle permission requests more effectively, ensuring a better user experience while maintaining security and privacy.

**3.1 Analysis of APP:- “App Permission Manager”**

**MobSF  
  
App Hases:-**

MD5 :- 4b989aa424eaadaa730b320f7564301e

SHA1 :- 600afdb3d4ee422e32af24ec39bc2dfe446bae17

SHA256:- e04b283977066e1f9171353e79170180e322da471a695859ce4f18223aa28439

Package Name:- com.shexa.permissionmanager

PLAY STORE RATING:- 4.1 STAR**🌟** & 1 Million+ Downloads

**Activities:-**

com.shexa.permissionmanager.screens.splash.SplashActivity  
com.shexa.permissionmanager.screens.apppermission.AppPermissionActivity  
com.shexa.permissionmanager.screens.settings.SettingsActivity  
com.shexa.permissionmanager.screens.home.HomeActivity  
com.shexa.permissionmanager.screens.Base.DemoActivity  
com.shexa.permissionmanager.screens.recentused.RecentUsedAppsNotificationActivity  
com.shexa.permissionmanager.screens.grouphome.GroupHomeActivity  
com.shexa.permissionmanager.activities.TransparentActivity  
com.shexa.permissionmanager.screens.groupapplisting.GroupAppListingActivity  
com.shexa.permissionmanager.screens.privacypolicy.PrivacyPolicyActivity  
com.shexa.permissionmanager.screens.detail.DetailActivity  
com.shexa.permissionmanager.screens.keep.KeepAppsActivity  
com.shexa.permissionmanager.screens.history.HistoryActivity  
com.shexa.permissionmanager.screens.license.LicenseActivity  
com.shexa.permissionmanager.screens.sysApp.SystemAppActivity  
com.shexa.permissionmanager.screens.specialpermission.SpecialPermissionActivity  
com.shexa.permissionmanager.screens.revert.RevertActivity  
com.shexa.permissionmanager.activities.GroupAnimationActivity  
com.shexa.permissionmanager.screens.bgapp.BgAppServiceActivity  
com.shexa.permissionmanager.screens.appbgdetail.AppBgDetailActivity  
com.shexa.permissionmanager.screens.chartpermission.ChartPermissionActivity  
com.google.android.gms.ads.AdActivity  
com.android.billingclient.api.ProxyBillingActivity  
com.google.android.play.core.missingsplits.PlayCoreMissingSplitsActivity  
com.google.android.play.core.common.PlayCoreDialogWrapperActivity  
com.google.android.gms.common.api.GoogleApiActivity

**Services:-**

com.shexa.permissionmanager.screens.splash.core.SplashScreenExtra  
com.shexa.permissionmanager.services.OverlayServiceForPermission  
com.shexa.permissionmanager.services.OverlayServiceForForceStop  
com.shexa.permissionmanager.services.AlarmJobService  
com.shexa.permissionmanager.services.GetAllAppService  
com.shexa.permissionmanager.services.FCMService  
com.google.firebase.messaging.FirebaseMessagingService  
com.google.firebase.components.ComponentDiscoveryService  
com.google.android.gms.ads.AdService  
androidx.work.impl.background.systemalarm.SystemAlarmService  
androidx.work.impl.background.systemjob.SystemJobService  
androidx.work.impl.foreground.SystemForegroundService  
androidx.room.MultiInstanceInvalidationService  
com.google.android.play.core.assetpacks.AssetPackExtractionService  
com.google.android.play.core.assetpacks.ExtractionForegroundService  
com.google.android.gms.measurement.AppMeasurementService  
com.google.android.gms.measurement.AppMeasurementJobService  
com.google.android.datatransport.runtime.backends.TransportBackendDiscovery  
com.google.android.datatransport.runtime.scheduling.jobscheduling.JobInfoSchedulerService.

**Receivers:-**

com.shexa.permissionmanager.receivers.InstallReceiver  
com.shexa.permissionmanager.receivers.BootDeviceReceiver  
com.shexa.permissionmanager.notification.service.ShowNotification  
com.google.firebase.iid.FirebaseInstanceIdReceiver  
androidx.work.impl.utils.ForceStopRunnable$BroadcastReceiver  
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryChargingProxy  
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryNotLowProxy  
androidx.work.impl.background.systemalarm.ConstraintProxy$StorageNotLowProxy  
androidx.work.impl.background.systemalarm.ConstraintProxy$NetworkStateProxy  
androidx.work.impl.background.systemalarm.RescheduleReceiver  
androidx.work.impl.background.systemalarm.ConstraintProxyUpdateReceiver  
androidx.work.impl.diagnostics.DiagnosticsReceiver  
com.google.android.gms.measurement.AppMeasurementReceiver  
com.google.android.datatransport.runtime.scheduling.jobscheduling.AlarmManagerSchedulerBroadcastReceiver.

**Providers:-**

com.google.android.gms.ads.MobileAdsInitProvider

androidx.startup.InitializationProvider

com.google.firebase.provider.FirebaseInitProvider.

**Libraries:-**

org.apache.http.legacy

**Permissions Declared in the Android Manifest File:-**

<uses-permission android:name="android.permission.INTERNET" />

<uses permissionandroid:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED" />

<uses-permission android:name="com.android.vending.BILLING" />

<uses-permission android:name="android.permission.VIBRATE" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.PACKAGE\_USAGE\_STATS" />

<uses-permission android:name="android.permission.QUERY\_ALL\_PACKAGES" />

<uses-permission android:name="android.permission.WAKE\_LOCK" />

<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />

<uses-permission android:name="com.google.android.gms.permission.AD\_ID" />

**Java Code Reference Link:**

https://mobsf.live/source\_code/?md5=4b989aa424eaadaa730b320f7564301e&type=java

**Smali Code Reference Link:**

https://mobsf.live/source\_code/?md5=4b989aa424eaadaa730b320f7564301e&type=smali

**Permission accessed by APP:-**

android.permission.ACCESS\_NETWORK\_STATE

android.permission.FOREGROUND\_SERVICE

android.permission.INTERNET

android.permission.PACKAGE\_USAGE\_STATS

android.permission.QUERY\_ALL\_PACKAGES

android.permission.RECEIVE\_BOOT\_COMPLETED

android.permission.VIBRATE

android.permission.WAKE\_LOCK

com.android.vending.BILLING

com.google.android.c2dm.permission.RECEIVE

com.google.android.finsky.permission.BIND\_GET\_INSTALL\_REFERRER\_SERVICE

com.google.android.gms.permission.AD\_ID

It Contains ANTI-VM code & ANTI-DEBUG code

**Frida-Logs**

Loaded Frida Script - api\_monitor Loaded Frida Script - debugger\_check\_bypass Loaded Frida Script - root\_bypass

Loaded Frida Script - ssl\_pinning\_bypass

[API\_Monitor\_]\_Cannotfindcom.android.okhttp.internal.http.HttpURLConnectionImpl.getInputStream

[SSL Pinning Bypass] okhttp CertificatePinner not found

[SSL Pinning Bypass] DataTheorem trustkit not found

[SSL Pinning Bypass] Appcelerator PinningTrustManager not found

[SSL Pinning Bypass] Apache Cordova SSLCertificateChecker not found

[SSL Pinning Bypass] Wultra CertStore.validateFingerprint not found

[SSL Pinning Bypass] Xutils not found

[SSL Pinning Bypass] httpclientandroidlib not found

[SSL Pinning Bypass] Cronet not found

[SSL Pinning Bypass] certificatetransparency.CTInterceptorBuilder not found

[RootDetection Bypass] test-keys check

[RootDetection Bypass] return value for binary: Superuser.apk

[RootDetection Bypass] return value for binary: su

[RootDetection Bypass] test-keys check

[RootDetection Bypass] return value for binary: Superuser.apk

[RootDetection Bypass] return value for binary: su

[Debugger Check Bypass] isDebuggerConnected() bypassed

[RootDetection Bypass] test-keys check

[RootDetection Bypass] return value for binary: Superuser.apk

[RootDetection Bypass] return value for binary: su

[RootDetection Bypass] test-keys check

[RootDetection Bypass] return value for binary: Superuser.apk [RootDetection Bypass] return value for binary: su

[Debugger Check Bypass] isDebuggerConnected() bypassed

**KeyFinding:-**

**In Android 13:-**

**1:-**According to console App (<https://support.google.com/googleplay/android-developer/answer/10158779?hl=en#zippy=>) (QUERY\_ALL\_PACKAGES )is considered as high-risk OR dangerous permission but by analyzing the app there is no permission request prompt , and it has access to ,all of my

installed app,

their package name,

their date of installation which indirectly use to find the version number & it’s related vulnerability Which can be used for reconnaissance & further exploitation .

**2:-** This App has access to know which other apps are installed in the system, with the use of android.permission.PACKAGE\_USAGE\_STATS Package ,

PackageManager & PackageInfo Classes ,

This app is able to retrieve & showing in the app itself details about:

App Name,

App Package Name,

App installed Date (Can help to indirectly retrieve the version of the app ),   
This app is able to retrieve the permission list of all individual installed apps in the system & displaying it on the screen with out any user explicit consent in system.

which indirectly use to find the version number & it’s related vulnerability Which can be used for reconnaissance & further exploitation .

**3:-** This app has access to all running state status of my installed apps

Retrieving running state of any other apps can be done by android.permission.PACKAGE\_USAGE\_STATS permission & this give accesss to about other app’s running state & also their usage pattern.

**4:-** This app is displayed on google play store , A Typical trust worthy platform for 2.5Billion users across the globe .

**5:-**

**2.10 Conclusion**

In conclusion, this review paper has provided a comprehensive analysis of the evolution of Android permissions and their associated security mechanisms. It has highlighted the disadvantages of the Android operating system, particularly in relation to privacy and security concerns. Additionally, a conceptual model aimed at addressing these issues has been proposed. The findings of this paper emphasize the critical need for users to acquire adequate knowledge about Android permissions when installing applications on their devices. Insufficient awareness can expose users to the risk of compromising their private information and files through dangerous and malicious applications. Consequently, the proposed security model offers a robust solution to protect users from such threats. It is recommended that the Android user community consider implementing the security model presented in this study, as it ensures the effective implementation of the confidentiality, integrity, and availability (CIA) triad. From a confidentiality perspective, the model employs encryption techniques to safeguard users' data and restrict access from harmful applications. To ensure integrity, the model prevents any tampering with stored files by adopting the AES encryption technique. Lastly, the model verifies the behavior of applications to detect potential malware and grants access only to trusted applications, thus ensuring the availability component of the CIA triad.

**3.0 Future Work**

While this study has identified the flaws in the current Android security architecture and proposed a conceptual model to address these issues, future work will involve implementing the proposed security model into a proof-of-concept. This implementation aims to be version-independent, enabling compatibility with all Android OS versions. The ultimate goal is to protect users from malware and application developers with malicious intent, thereby ensuring the effective implementation of the CIA triad for Android mobile users. The implementation will be thoroughly evaluated and tested to assess its effectiveness in real-world scenarios