Security and Privacy Analysis of Mobile Applications for Vulnerable Groups

Narayana Vedula
Information Systems Security
Concordia University
Quebec, Canada
narayanavedulaca@gmail.com

Divya Varshini M
Information Systems Security
Concordia University
Quebec, Canada
D_murath@live.concordia.ca

Jyothi Sree Kondru
Information Systems Security
Concordia University
Quebec, Canada
j_kondru@live.concordia.ca

Vaishnavi Kalathur
Information Systems Security
Concordia University
Quebec, Canada
Kalathurvaishnavi971@gmail.com

Gowri Annadurai
Information Systems Security
Concordia University
Quebec, Canada
g_annadu@live.concordia.ca

Kirandeep kaur
Information Systems Security
Concordia University
Quebec, Canada
Sandhukiran526@gmail.com

***Abstract*—Vulnerable populations, including individuals with disabilities, those experiencing homelessness, and those with substance use disorders, are increasingly utilizing mobile applications designed to meet their unique needs, similar to other demographic groups. These applications enhance daily activities, social connections, and access to resources . Additionally, they often provide valuable support to caregivers and service providers by offering tools to manage health and welfare-related responsibilities. However, despite their benefits, these applications typically handle a considerable amount of sensitive information, including medical records, real-time location data, and Personally Identifiable Information (PII) of both users and caregivers. There is a notable lack of research focused on applications developed specifically for these vulnerable groups, even though the security and privacy aspects of applications intended for the general public have been extensively studied. This research employs a combination of static and dynamic analyses to examine the security and privacy concerns associated with 15 prominent Android applications that cater to these populations.**

***Index Terms*—Disabled users, Homeless individuals,Vulnerable groups, People with substance abuse issues,Privacy, Android Apps Privacy and Security**

I. INTRODUCTION

Individuals with disabilities, the homeless, and those facing challenges related to substance misuse represent vulnerable populations significantly impacted by the rapid advancement of mobile technology. While these innovations offer novel opportunities to improve connectivity, accessibility, and health management, they also introduce complex security and privacy challenges. Low digital literacy, lack of confidence in technology, and limited awareness of risks make these communities more vulnerable to exploitation. Mobile applications designed for these groups provide essential services, including mental health support, resource navigation, and healthcare assistance. However, these applications frequently handle sensitive personal data, such as Personally Identifiable Information (PII), real-time geolocation, and medical records. Despite their widespread use, there is limited knowledge regarding their security and privacy vulnerabilities. Flaws in these applications can expose users to significant risks, both online and offline. Although considerable research has been conducted on privacy and security issues in mobile applications for the general population, there is a notable lack of studies focusing on vulnerable groups. Most existing research either investigates user behavior or assesses a limited range of applications, resulting in significant gaps in understanding the broader implications. This study intends to fill these gaps by performing a comprehensive analysis of 15 Android applications tailored for individuals with disabilities, the homeless, and those dealing with substance abuse. Through both dynamic and static analysis techniques, this research aims to identify and emphasize critical security and privacy issues affecting these populations.

Contributions and Notable Findings:

1) Significant security and privacy flaws in the evaluated Android apps were found by our analysis, which combined static and dynamic approaches. Inadequate authentication procedures, poor session management, the disclosure of Personally Identifiable Information (PII), data sharing with third parties, and other serious security flaws were among the shortcomings found in these programs, which were designed to assist at-risk groups.
2) Three out of four applications in a subset exhibited session management flaws, including inadequate token security, improper session expiration, and unsafe storage practices.
3) Half of the evaluated applications had personally identifiable information (PII) transmitted in an insecure manner. Inadequate handling of sensitive user data during transmission and the usage of unencrypted HTTP protocols were the primary causes of the vulnerabilities,

which raised the danger to user privacy.

4) Potential attackers were able to get around authentication procedures and obtain sensitive user data or restricted functions by exploiting flaws in two applications' authentication systems. This highlights a serious weakness in data protection and access management.
5) Furthermore, it was discovered that two applications shared user data with third parties without sufficient anonymization or explicit notice about the sharing of advertising-related data. Such actions give rise to grave worries regarding data misuse and privacy violations.
6) Inadequate validation procedures in three of the four programs revealed vulnerabilities in file path management, which attackers could exploit for privilege escalation or unauthorized file access. Attackers may use this flaw to escalate privileges within the application or obtain unauthorized access to files that are prohibited.
7) All applications had security misconfigurations, with each one exhibiting small misconfigurations such as inadequately controlled permissions, security-unoptimized default settings, and exposed services with inadequate protections. The possibility of exploitation rose as a result of these errors.

II. Identifying Privacy and Security Risks: A Threat Modeling Approach

A. Exploring Security and Privacy Vulnerabilities

Our primary focus lies in analyzing information that can be transmitted over a network, especially personally identifiable information (PII) [1]. A PII breach occurs when unauthorized disclosure involves data that uniquely identifies an individual, such as email addresses, physical addresses, passwords, birth dates, medical records, or unique device identifiers. Building on this definition, we have developed a comprehensive list of potential security and privacy vulnerabilities to assess in applications designed for senior citizens.

1) **Improper Authentication Management:** Refers to vulnerabilities that allow unauthorized actors to gain access to user accounts, potentially leading to unauthorized logins and data breaches.
2) **Inadequate Access Control:** Occurs when an attacker can view or access other users' data within a platform without proper authorization, compromising user privacy.
3) **Deficient Input Validation**: The absence or inadequacy of input validation mechanisms, enabling potential injection attacks, such as SQL or code injection, which may expose or compromise sensitive user information.
4) **Exploitable Backend Infrastructure:** Involves the use of outdated, misconfigured, or unauthenticated server-side software, including services like Firebase, which can be remotely exploited by attackers.
5) **Transmission of Authentication Secrets in Plain-text:** Authentication credentials, such as passwords or session identifiers, are transmitted in plain text, making them vulnerable to interception by network attackers seeking unauthorized access.
6) **Unencrypted Transmission of PII and Device Information:** Personally Identifiable Information (PII), device data, and usage patterns are transmitted without encryption (e.g., over plain HTTP), increasing the risk of data interception.
7) **Data Sharing with Third Parties:** Refers to the transmission of PII, device details, and usage information to third-party domains, trackers, or external library providers, often without user knowledge or consent.
8) **Inadequate Security Configurations:** Includes misconfigurations in backend HTTP web servers, such as the absence of Cross-Origin Resource Sharing (CORS) policies or improper implementation of flash cross-domain policies, exposing applications to large-scale attacks.
9) **Excessive and Dangerous Permissions:** Includes scenarios where apps automatically acquire dangerous permissions (e.g., write access to external storage or fine location data) through third-party libraries or malicious applications sharing the same signed certificate as the app developer.

B. Threat Model

On-path Attacker

This type of attacker functions within the network, situating itself between the user's device and the backend server of the application. By serving as an intermediary, the on-path attacker is capable of intercepting, modifying, or manipulating communications. This facilitates actions such as eavesdropping on confidential information, injecting harmful content, or interrupting the communication flow between the application and the server.

Applications are tested in controlled environments, focusing on the following:

Dynamic Analysis

Applications are tested in controlled environments, focusing on the following:

- Monitoring network traffic via a proxy to assess data flow , detect data leaks and enumerate usernames and passwords.
- Inspecting GraphQL endpoints for potential data leaks or backend security flaws.

Static Analysis

The analysis of static elements includes:

- Reviewing app permissions to identify excessive or high-risk access rights.
- Examining third-party libraries for known security issues.
- Evaluating backend configurations, such as Firebase setups, to identify misconfigurations or outdated components.

III. Methodology

This section covers the approach used to find and choose the applications for Android that formed part of our review, as well as the technique used for both the static and dynamic analysis of the chosen applications.

A. App Selection

Using relevant keywords, a Google Play Store search was conducted to find mobile applications made especially for the senior population. A total of 15 applications were chosen for evaluation based. The review focused on applications tailored to individuals with disabilities, the homeless, and those dealing with substance abuse. Applications that required confirmed identities or required financial account information, including bank or credit card details, were not taken into consideration. Every application was manually examined to ensure that it met the necessary inclusion requirements. In conclusion, an improved dataset with 15 apps served as the focus of our investigation.

B. Dynamic Analysis

This section describes the dynamic testing we do under simulated real-world settings to assess the expected features of the chosen applications. In order to find any security and privacy flaws, the network traffic produced by the applications is recorded and examined during this process. We use two comprehensive tools for assessing the security of web and mobile applications, Burp Suite,GraphQl and Mobile Security Framework (MobSF), for this investigation. Together with other extensions, their wide range of features provides a thorough and dynamic assessment of the selected apps.

Testing across several device configurations was made possible via the emulator, which functioned as a mimicked Android environment allowing testing across various device configurations. We used the Android Studio Emulator, specifically a Google Pixel 4 running Android Version 12 (x86 64, API 31), configured through the Android Virtual Device (AVD). This setup provided flexibility to test application behavior under different system conditions. ADB, a command-line tool, facilitated communication between the host machine and the emulator, enabling operations such as file transfers, system state observation, and controlled access for testing purposes. This configuration made it possible to test how the application behaved in various system scenarios. By enabling the connection between the emulator and the host computer, the command-line program, ADB made it possible to do tasks like file transfers, system state monitoring, and controlled access for testing.

1) Burpsuite: We installed applications from the Google Play Store on an emulator, created test user accounts in the apps and using Burpsuite Proxy, which is the main functionality in Burpsuite,we turned on interception and monitored the network traffic.

Repeater in BurpSuite: This allowed us to resend requests with manual modifications, testing the server's responses. We used it to detect insecure session management, unencrypted PII or device information transmission. Testing involved modifying requests to access unauthorized data. Apps were marked as having improper access control if, for example, using our test account,we were able to retrieve many user's data by altering request parameters.

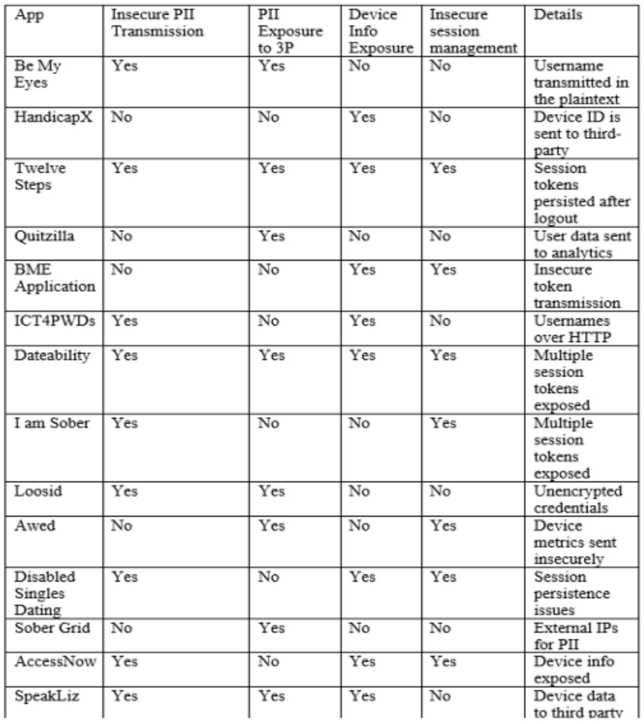
Intruder in BurpSuite: A fuzzer to test input points by iterating over multiple values, assessing vulnerabilities brute-force risks, and weak API parameters. Using this functionality we were able to enumerate usernames, passwords. Decoder: This tool decoded encoded data, such as Base64 or URL formats, revealing potentially hidden or sensitive information within parameter values or headers.

Applications, such as *Dateability*, were installed from the Google Play Store and tested for insecure data transmission. For instance, *Dateability* exposed sensitive user data, including email addresses, usernames, user IDs, photo URLs, and Firebase Cloud Messaging (FCM) tokens, in plaintext, making it susceptible to interception. Poor session management was also evident, with authentication tokens transmitted unencrypted, increasing the risk of unauthorized access.

In parallel, application code, permissions, and backend configurations were examined as part of static analysis to supplement dynamic findings. Risky permissions were found, such Wheelmap giving CAMERA access, which might result in unwanted access to private photos if handled improperly. Likewise, HandicapX employed WRITE_EXTERNAL_STORAGE, which, in the absence of adequate validation, raised the possibility of file manipulation attacks. Misconfigurations in Firebase were also noted; for example, Dateability showed user identities and public FCM tokens because of inadequate access controls.

2)GraphQL: The security and privacy issues of mobile applications that cater to vulnerable populations, like the homeless, those with disabilities, and those battling substance misuse, are examined in this paper. The investigation finds flaws such as unsafe data transmission, insufficient access control, and PII exposure to third parties through the use of GraphQL server administration, vulnerability assessments, and network traffic monitoring. To handle and query app data, including IDs, names, sensitive data, and query logs, a bespoke GraphQL server was put into place. Apollo GraphQL server is launched by server.js, types, queries, and mutations are defined by schema.js, data operations are managed by resolvers.js, and use trends and data sensitivity are examined by analysis.js. The modular architecture makes it simple to add additional queries or analytical logic, offering suggestions and insights to improve the application's security.

A dynamic framework for handling and evaluating app data is offered by the deployment of a GraphQL server, with an emphasis on security, sensitivity, and usage patterns. As the entry point, the server.js file sets up the GraphQL server using Apollo Server, binds the resolvers and schema, and configures the API to execute on a specified port. Through Apollo Playground, the server may be accessed via 'http://localhost:4000', enabling queries and modifications. A smooth connection between the server and the underlying logic for API functionality is guaranteed by this initialization.



App	Insecure PII Transmission	PII Exposure to 3P	Device Info Exposure	Insecure session management	Details
Be My Eyes	Yes	Yes	No	No	Username transmitted in the plaintext
HandicapX	No	No	Yes	No	Device ID is sent to third-party
Twelve Steps	Yes	Yes	Yes	Yes	Session tokens persisted after logout
Quitzilla	No	Yes	No	No	User data sent to analytics
BME Application	No	No	Yes	Yes	Insecure token transmission
ICT4PWDs	Yes	No	Yes	No	Usernames over HTTP
Dateability	Yes	Yes	Yes	Yes	Multiple session tokens exposed
I am Sober	Yes	No	No	Yes	Multiple session tokens exposed
Loosid	Yes	Yes	No	No	Unencrypted credentials
Awed	No	Yes	No	Yes	Device metrics sent insecurely
Disabled Singles Dating	Yes	No	Yes	Yes	Session persistence issues
Sober Grid	No	Yes	No	No	External IPs for PII
AccessNow	Yes	No	Yes	Yes	Device info exposed
SpeakLiz	Yes	Yes	Yes	No	Device data to third party

Fig. 1. Observed results from GraphQL server analysis

The GraphQL API's structure is defined by the schema.js file, which also describes data types, queries, and modifications. App and AppAnalysis, which contain application data and risk assessment metrics, are essential components. While app analysis retrieves facts about a particular app by its ID, queries like GetAllApps obtain information about all apps. Mutations such as incrementQueries monitor query use for every app, while createApp permits the creation of new app records with sensitive data columns. In addition to organizing app data, this schema offers a framework for evaluating data sensitivity and examining usage patterns.

The resolvers.js file links the schema to real data operations by implementing the logic for queries and changes. While app analysis offers in-depth analysis for a single app, queries like getAllApps employ helper methods to gather and analyze data from the data store. Data changes, such as adding new apps or changing query counts, are handled via mutations like createApp and increase queries. The resolvers use analytical functions to compute metrics like usage patterns and risk levels, bridging the gap between the backend logic and the API.

Based on characteristics like sensitive data and query frequency, the analysis.js file assesses the risk level and usage patterns of each app. Apps containing sensitive data, for instance, are labeled "High Risk," but those with few inquiries are labeled "Low Usage." Features like getAppAnalysis and getAllAppAnalyses help stakeholders identify high-risk applications and track use trends by offering comprehensive visibility into both individual and aggregate app performance.

App IDs, names, sensitive data, query counts, and calculated risk metrics are among the important details that are gathered by the getAllApps query. This query is crucial for guaranteeing app security and compliance since it analyzes data sensitivity, tracks use patterns and identifies high-risk apps by assessing parameters like riskLevel and queriesRisk.

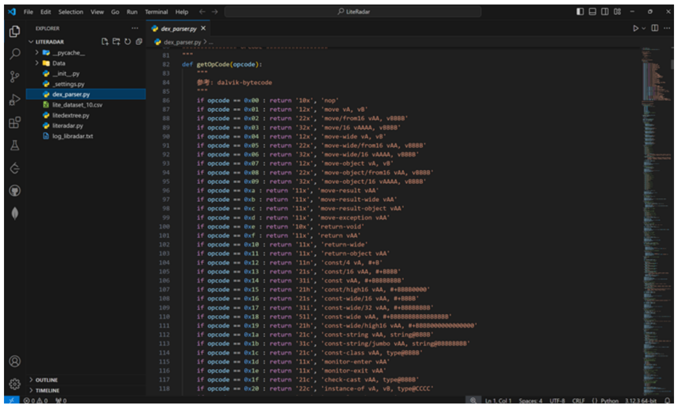
As demonstrated in fig 1, the security and privacy flaws of 15 applications that serve vulnerable populations are highlighted in this table. The vulnerability of Personally Identifiable Information (PII) to third parties and its insecure transmission are major problems. Poor session management techniques, such as keeping session tokens after logging out, and the leakage of device information were also common. Applications such as "Dateability" and "Twelve Steps" exhibit several flaws, such as exposed session tokens and unsafe PII processing. On the other hand, some, such as "Quitzilla" and "BME Application," reduce the vulnerability of third-party PII, but they still have issues with token management or device information security. Overall, there are serious privacy and security hazards due to unsafe behaviors in several apps.

The link for graphql source code can be found in the GitHub Repository .

C. Static Analysis

By providing information about the anticipated operational flow of applications, our static analysis improves the dynamic testing procedure. This allows us to discover any suspicious actions or inadequate security policies by correlating findings with dynamic testing results. Issues like inadequate input sanitization, unsafe Firebase setups, and other flaws that might compromise security or privacy are covered in the report. This analysis's components include:

1) **Data Sensitivity, Risk, Library and Dependency Analysis:** The examination carried out with LiteRadar concentrated on analyzing Android applications to uncover security vulnerabilities and assess data management practices. LiteRadar achieves this by decompiling Android APK files and scrutinizing their .dex files, which house the application's bytecode. This bytecode is processed using the dexparser.py module to extract essential information regarding the app's components, including methods, libraries, and permissions. The extracted data is then organized into a hierarchical DEX tree through the litedextree.py module, facilitating a systematic evaluation of dependencies and relationships within the application. The entire operation is coordinated by the literadar.py module, which manages data parsing, dependency mapping, and the formulation of security assessments.
   Comprehensive insights into usage trends and security are made possible by a thorough query of application fields in LiteRadar. To organize and retrieve applications during analysis, the appid and appname are crucial identifiers. To help developers prioritize updates or replacements for problematic components, the libraryDependencies field offers an overview of third-party libraries



```python
    """
def getOpCode(opcode):
    """
    参考: dalvik-bytecode
    """
    if opcode == 0x00 : return '10x', 'nop'
    if opcode == 0x01 : return '12x', 'move vA, vB'
    if opcode == 0x02 : return '22x', 'move/from16 vAA, vBBBB'
    if opcode == 0x03 : return '32x', 'move/16 vAAAA, vBBBB'
    if opcode == 0x04 : return '12x', 'move-wide vA, vB'
    if opcode == 0x05 : return '22x', 'move-wide/from16 vAA, vBBBB'
    if opcode == 0x06 : return '32x', 'move-wide/16 vAAAA, vBBBB'
    if opcode == 0x07 : return '12x', 'move-object vA, vB'
    if opcode == 0x08 : return '22x', 'move-object/from16 vAA, vBBBB'
    if opcode == 0x09 : return '32x', 'move-object/16 vAAAA, vBBBB'
    if opcode == 0xa : return '11x', 'move-result vAA'
    if opcode == 0xb : return '11x', 'move-result-wide vAA'
    if opcode == 0xc : return '11x', 'move-result-object vAA'
    if opcode == 0xd : return '11x', 'move-exception vAA'
    if opcode == 0xe : return '10x', 'return-void'
    if opcode == 0xf : return '11x', 'return vAA'
    if opcode == 0x10 : return '11x', 'return-wide'
    if opcode == 0x11 : return '11x', 'return-object vAA'
    if opcode == 0x12 : return '11n', 'const/4 vA, #+B'
    if opcode == 0x13 : return '21s', 'const/16 vAA, #+BBBB'
    if opcode == 0x14 : return '31i', 'const vAA, #+BBBBBBBB'
    if opcode == 0x15 : return '21h', 'const/high16 vAA, #+BBBB0000'
    if opcode == 0x16 : return '21s', 'const-wide/16 vAA, #+BBBB'
    if opcode == 0x17 : return '31i', 'const-wide/32 vAA, #+BBBBBBBB'
    if opcode == 0x18 : return '51l', 'const-wide vAA, #+BBBBBBBBBBBBBBBB'
    if opcode == 0x19 : return '21h', 'const-wide/high16 vAA, #+BBBB000000000000'
    if opcode == 0x1a : return '21c', 'const-string vAA, string@BBBB'
    if opcode == 0x1b : return '31c', 'const-string/jumbo vAA, string@BBBBBBBB'
    if opcode == 0x1c : return '21c', 'const-class vAA, type@BBBB'
    if opcode == 0x1d : return '11x', 'monitor-enter vAA'
    if opcode == 0x1e : return '11x', 'monitor-exit vAA'
    if opcode == 0x1f : return '21c', 'check-cast vAA, type@BBBB'
    if opcode == 0x20 : return '22c', 'instance-of vA, vB, type@CCCC'
```

Fig. 2. VS Code showing a Python script for Dalvik bytecode mapping

used within the application, evaluating permissions, vulnerabilities, and data sensitivity. Applications that may need more security measures, such as encryption or restricted access, are flagged by the sensitiveDataAccess field, which shows whether an application interacts with sensitive data, such as location or personal information. As shown in fig 2 By tracking how frequently sensitive data fields are visited, the usage frequency field helps identify applications that are highly exposed and may require more stringent data management and rules. Additionally, this field shows usage patterns, which helps with proactive monitoring to stop possible data leaks. Last but not least, the security assessment field summarizes the app's security posture by combining important discoveries into a risk level and vulnerability score. Applications deemed high-risk are marked for prompt consideration to reduce the possibility of breaches. By offering actionable insights, this structured querying solution enables teams to prioritize security improvements, concentrate on important areas, and guarantee adherence to data protection regulations.

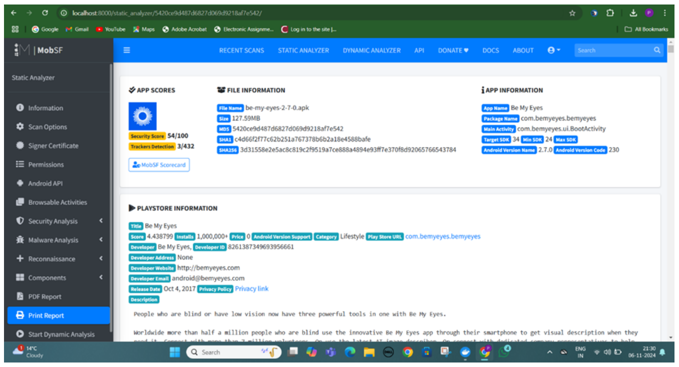
We have observed critical security concerns in some applications. Many of these apps handled sensitive user information, including personal data and geolocation, yet frequently lacked sufficient protective measures. For example, certain applications transmitted sensitive data over unencrypted channels like HTTP rather than utilizing secure HTTPS connections. LiteRadar identified these vulnerabilities, highlighting potential risks of data interception. Additionally, some applications were discovered to lack strong authentication mechanisms, rendering sensitive endpoints susceptible to unauthorized access.

2) **Third-Party Libraries:** Android applications usually use third-party libraries to integrate external services and enhance functionality. We use LiteRadar to perform a comprehensive analysis of the libraries used in the chosen applications. This tool is controlled by a specially written Python script that automatically extracts library data from the APK files, such as names, types, and permissions. The libraries are assessed relative to their intended purposes and permissions to identify potential risks related to their use.

3) **Authentication Mechanism:** By examining the Firebase authentication options available on the application, we have examined the authentication systems found in the chosen mobile applications. We checked the configuration files for authentication rule mismanagement using the FirebaseScanner tool. For example, we attempted to confirm whether these applications required authentication before granting the user access to the cloud or database. The application of Firebase's rules in our situation. To check whether the right steps were taken to limit access, such as granting permissions relevant to the user's position or permitting anonymous users access, a JSON file was needed. Through this procedure, we were able to confirm that the authentication techniques were implemented efficiently and by security guidelines.

4) **Misconfiguration Detection:**We evaluated the authentication systems implemented in the selected applications by examining the Firebase authentication settings. We examined the configuration files using Firebase Scanner to find any instances of authentication protocol mishandling. We checked to see if the apps required user authentication before allowing access to the database or cloud storage. We thoroughly reviewed the Firebase guidelines as part of our study .json file to verify the implementation of appropriate access control policies, such as limiting rights based on user roles and preventing access for unauthenticated users. Through this analysis, we were able to confirm that the authentication procedures were carried out efficiently and by security best practices.

5) **App Permissions & Certificate Validation**: Using concepts such as MobSF [10] and Docker, the described steps guarantee detailed assessments in an efficient and repeatable manner, in an environment free from external threats. The static analysis starts by starting the MobSF framework in a Docker container to be able to gain isolation and be reproducible. The Docker integration also helps in maintaining a synchronized analysis environment wherever the setup is done and also resolves the problem of conflicts of dependencies that may occur when the packages are to be installed directly. For instance, the MobSF container is launched using the command:
"docker run -it -p 8000:8000 open security/mobile security framework mobs "
This generates a safe environment for testing the application that can only be accessed from within a web browser; the developer of the app under analysis, in this case, is "Be My Eyes," which uploads the APK file for analysis. After you have uploaded the APK, MobSF [10] proceeds to decompile an application to analyze its components. Several important factors are discussed in the case, starting with the Security Score,



APP SCORES

Security Score 54/100
Trackers Detection 3/432

FILE INFORMATION

File Name be-my-eyes-2-7-0.apk
Size 127.59MB
MD5 5420ce9d487d6827d069d9218af7e542
SHA1 c4d66f2f77c62b251a767378b6b2a18e4588bafe
SHA256 3d31558e2e5ac8c819c2f9519a7ce888a4894e93ff7e370f8d92065766543784

APP INFORMATION

App Name Be My Eyes
Package Name com.bemyeyes.bemyeyes
Main Activity com.bemyeyes.ui.BootActivity
Target SDK 34 Min SDK 24 Max SDK
Android Version Name 2.7.0 Android Version Code 230

PLAYSTORE INFORMATION

Title Be My Eyes
Score 4.438799 Installs 1,000,000+ Price 0 Android Version Support Category Lifestyle Play Store URL com.bemyeyes.bemyeyes
Developer Be My Eyes, Developer ID 8261387349693956661
Developer Address None
Developer Website http://bemyeyes.com
Developer Email android@bemyeyes.com
Release Date Oct 4, 2017 Privacy Policy Privacy link
Description

```
People who are blind or have low vision now have three powerful tools in one with Be My Eyes.

Worldwide more than half a million people who are blind use the innovative Be My Eyes app through their smartphone to get visual description when they
```

Fig. 3. "Be My Eyes" Apk - static analysis

which gives a quantitative measure of an app's security. For instance,As shown in Fig. 3 the 'Be My Eyes' app in the analysis has been awarded only 54/100, which means that they have great possibilities to respond to possible vulnerabilities. Furthermore, it is discovered that MobSF also detects third-party trackers which may be privacy intrusive. Indeed, in this case, the tool was able to identify 3 out of the 432 trackers involved in the app. Also, the permissions required by the app are scrutinized to categorize them into dangerous, normal, or unnecessary, and have features pointing to specific risks regarding the apps using sensitive permission like camera, location, and storage among others.
Static analysis on the other hand offers an initial and broad assessment of mobile applications and negligence to it exposes the applications to high risks. The analysis is made robust, reproducible, and efficient by the integration of MobSF and Docker. Nonetheless, as a result of identifying security issues that are vital for the improvement of general safety and privacy of mobile applications, static analysis finds its place as one of the significant steps toward improving the applications' safety.

6) **Data Encryption:** To verify compliance with security best practices, we evaluated the encryption of data transmitted between the client and Firebase. To ensure the confidentiality of data in transit, we confirmed that all communications were carried out over HTTPS by using FirebaseScanner and manually analyzing network traffic. Intercepted network requests were examined to identify any unencrypted transfers and the applications were tested in controlled environments. Potential vulnerabilities were identified in cases of HTTP usage or unsecured data exchanges. We also looked into the security of the transmission of sensitive data, including API keys and user credentials. Our findings made clear how important it is to use HTTPS to prevent illegal access and data interception.
7) **Library and Dependency Analysis:** LiteRadar serves as a specialized tool for Android application analysis, focusing on locating and evaluating the third-party libraries that are included in these apps. Because these libraries might offer both functional enhancements and potential security flaws, this study is crucial to understanding the dependencies that a program depends on. LiteRadar helps developers make well-informed decisions that minimize potential hazards by enabling them to identify essential libraries as well as those that provide risks. This tool's ability to organize dependencies using the litedextree.py file in a structured tree manner is one of its noteworthy features. The program's architecture is represented hierarchically by this DEX tree, which arranges dependencies based on how they relate to one another within the application.

Result

The study's dynamic and static evaluations identified serious flaws in all of the assessed applications. These results point to several security and privacy threats [1] linked to incorrect data processing, shoddy authentication systems, and configuration errors.

Sensitive user data was transmitted insecurely, which was one notable problem. Numerous applications used unencrypted HTTP channels to send Personally Identifiable Information (PII), such as email addresses, session tokens, photo URLs, and usernames. For example, the Dateability app was discovered to disclose authentication tokens and user passwords in plaintext, leaving them open to hacker interception. There is a serious risk of data compromise and illegal access due to this absence of encryption.

Vulnerabilities in authentication were another major worry. Bad session management techniques were shown by apps such as Dateability, which exposed tokens in plaintext in network traffic logs. Such actions jeopardize user privacy and platform security by giving hackers the chance to access user accounts without authorization.

Many applications, such as HandicapX, allowed third-party access to user data, including advertising-related permissions, raising significant privacy concern.Several programs were found to have input validation problems, making them more vulnerable to injection attacks. For instance, if user input was not carefully verified, HandicapX's usage of permissions like WRITE_EXTERNAL_STORAGE could result in file path manipulation attacks. Attackers may be able to maliciously utilize program resources thanks to such flaws.

Another level of risk was introduced by backend setup errors. Apps like Dateability were found to have misconfigured Firebase services, where inadequate database security permitted sensitive user data, including tokens and identifiers, to be accessed without the necessary authentication measures. Critical backend systems are vulnerable to possible exploitation due to these misconfigurations.

Table below represents overall results for 15 apps : with Empty circle : On-device Attacker,Full circle : Remote attacker ,Half Circle : On-path attacker.

All things considered, the analysis's findings show that the assessed apps have serious security and privacy issues.

The results emphasize that to protect user data and preserve application integrity, developers must give top priority to secure transmission techniques, put strong authentication and session management systems in place, make sure that input is validated correctly, and get rid of backend misconfigurations.

Conclusion

This study clarifies the significant security and privacy issues that Android apps make for vulnerable groups—such as people with impairments, the homeless, and those battling drug addiction phase. Using both static and dynamic approaches, we thoroughly examined 15 Android apps finding serious flaws in the authentication processes, session management, data handling, and general security setups. Because of these vulnerabilities, apps that target groups that are already more vulnerable to exploitation because of their unique circumstances urgently need to implement strong security procedures.

Among the most alarming discoveries was the extensive application of unsafe session management techniques. Vulnerabilities including incorrect token storage, insufficient expiration policies, and weak session security were found in three of the four assessed applications. These vulnerabilities seriously jeopardize the confidentiality of sensitive user data by putting users at risk of session hijacking and illegal access. Furthermore, in half of the applications examined, inappropriate transmission of Personally Identifiable Information (PII) was found. Serious questions about user privacy and data protection were raised by the usage of unencrypted HTTP protocols and inadequate security measures during data transmission.

The authentication systems of two of the tested applications had serious weaknesses that allowed attackers to get around security measures and access restricted features or sensitive data without authorization. Developers must immediately adopt strong authentication procedures, particularly for applications that handle sensitive user data, as highlighted by these access control flaws. Furthermore, it was discovered that two applications shared user data with outside parties without anonymization or user authorization. Such actions present serious ethical and legal issues about data privacy, especially when connected to analytics or advertising services.

Three programs were found to have file path manipulation vulnerabilities as a result of inadequate input validation. These flaws emphasize the significance of thorough validation in application development by giving attackers the ability to escalate privileges or access restricted files. Security misconfigurations, including permissive default settings, poorly managed permissions, and exposed services with insufficient safeguards, were also present in all of the examined apps. Despite their seeming insignificance, these problems greatly raise the possibility of exploitation.

To mitigate these risks, developers should adopt a multi-layered security approach, incorporating robust input validation, encrypted data transmission, and strong authentication protocols. It is also crucial to have clear data-sharing procedures that include user consent and appropriate anonymization. By taking these precautions, developers will be able to safeguard consumers and keep these apps dependable.

References

[1] Pranay Kapoor, Rohan Pagey, Mohammad Mannan, and Amr Youssef:Silver Surfers on the Tech Wave: Privacy Analysis of Android Apps for the Elderly.

[2] Choi, H., Kim, Y.: Large-scale analysis of remote code injection attacks in Android apps. Security and Communication Networks 2018, 1–17 (04 2018). https://doi.org/10.1155/2018/2489214

[3] Shalini Saini, Dhiral Panjwani, Nitesh Saxena.:Mobile Mental Health Apps: Alternative Intervention or Intrusion?(2022).https://ieeexplore.ieee.org/document/9851975

[4] Sajjad Pourali, Nayanamana Samarasinghe, Mohammad Mannan.: Hidden in Plain Sight: Exploring Encrypted Channels in Android apps.https://arxiv.org/pdf/2209.15107

[5] Nora McDonald and Andrea Forte .: Privacy and Vulnerable Populations.

[6] Leonardo Horn Iwaya · M. Ali Babar Awais Rashid · Chamila Wijayarathna.:On the Privacy of Mental Health Apps An Empirical Investigation and Its Implications for Apps Development(Jan 2022).https://arxiv.org/pdf/2201.09006

[7] Frik, A., Nurgalieva, L., Bernd, J., Lee, J.S., Schaub, F., Egelman, S.: Privacy and security threat models and mitigation strategies of older adults. In: Proceedings of the Fifteenth USENIX Conference on Usable Privacy and Security. p. 21–40. SOUPS'19, USENIX Association, USA (2019)

[8] Razaghpanah, A., Nithyanand, R., Vallina-Rodriguez, N., Sundaresan, S., Allman, M., Kreibich, C., Gill, P., et al.: Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In: The 25th Annual Network and Distributed System Security Symposium (NDSS 2018) (2018)

[9] Ren, J., Lindorfer, M., Dubois, D.J., Rao, A., Choffnes, D., Vallina-Rodriguez, N., et al.: Bug fixes, improvements,... and privacy leaks. In: The 25th Annual Network and Distributed System Security Symposium (NDSS 2018) (2018)

[10] Shirke, K.: Mobile security framework (mobsf) static analysis (Jan 2019), online article. https://medium.com/@kshitishirke/mobile-security-framework-mobsfstatic-analysis-df22fcdae46e

[11] https://ecsa.citizen-science.net/wp-content/uploads/2021/06/Guide-to-ICTs-data-and-vulnerable-people-PUBLIC.pdf

[12] A Scalable Inclusive Security Intervention to Center Marginalized & Vulnerable Populations in Security & Privacy Design,online article-https://dl.acm.org/doi/10.1145/3633500.3633508

[13] Slane, A., Pedersen, I., Hung, P.C.K.: Involving seniors in developing privacy best practices: Towards the development of social support technologies for seniors. in: Office of the privacy commissioner of canada., online article (2020). https://www.priv.gc.ca/en/opc-actions-and-decisions/research/funding-forprivacy-research-and-knowledge-translation/completed-contributions-programprojects/2019-2020/p 2019-20 03/

[14] ZhiLiang Bu; Chunlei Zhao; Liangyi Gong; Yan Wang; Yi Yang; Xi Wang.: ML-based Privacy Leakage Behavior Detection in Android Apps at Scale(2022),https://ieeexplore.ieee.org/document/10189644

[15] Online article,https://bmcmedicine.biomedcentral.com/articles/10.1186/s12916-015-0444-y

[16] Jingjing Ren , Martina Lindorfer , Daniel J. Dubois, Ashwin Rao, David Choffnes and Narseo Vallina-Rodriguez, Bug Fixes, Improvements, ... and Privacy Leaks, https://recon.meddle.mobi/papers/appversion.pdf

[17] Maaß, W.: The Elderly and the Internet: How Senior Citizens Deal with Online Privacy, pp. 235–249. Springer Berlin Heidelberg (2011)

[18] Kakulla, B.N.: Older adults keep pace on tech usage. AARP Research (2020), https://www.aarp.org/research/topics/technology/info-2019/2020-technologytrends-older-americans.html