Cybersecurity and digital forensics year 3

Module: individual project

Supervisor: Mrinal Sharma

Student ID: M00729531

Chapter 3 research design

1. Table of Contents

1.Development of working Hypotheses

1.1 Hypotheses: applications on the Samsung A12 have vulnerabilities related to improper handling of user inputs.

This study will attempt to uncover vulnerabilities in Samsung devices that run an android version 12 and vulnerabilities related in the handling of user inputs that could lead to unauthorize access to user personal data or manipulated user data. [1] According to the Indiatimes, it has been extensively reported that Samsung devices that run android versions 11, 12, 13 and 14 have vulnerabilities that can be used by attackers to infiltrate the device and access sensitive information by bypassing security measures of the device *(The Times of India, 2023)*. The hypothesis will focus on a variety range of mobile apps that may show signs of such vulnerabilities and provide the best possible mitigation measures for the discovered vulnerabilities.

1.2 Research goals and identified gaps:

The development of important questions that aim to uncover vulnerabilities in Samsung devices based on this research study goals and identified gaps is crucial to this study for this study to reach its full potential. The questions are as follows:

- What type of user input vulnerabilities are most common in apps on the Samsung A12 device?
- How do these vulnerabilities vary between third-party apps, system apps and high-risk apps that handle sensitive data such as financial apps?
- What are the potential negative impacts of the vulnerabilities uncovered on user data security?

1.3 Expected outcomes:

The research study will expectedly aim to shed light on how user inputs are mishandled by applications in the Samsung device. If such outcome is

reached, then the apps in the Samsung device are vulnerable to common attacks such as SQL injection, buffer overflows and cross-site scripting thus compromising the security of user data. such outcomes will be reached through rigorous testing of a wide range of applications across the board from high-risk apps that handle the most sensitive user data to the lower threat level apps that may still be susceptible to attacks.

1.4 Expected measurable and observable outcomes:

This research study expects to demonstrate that at least 50% of the apps that will be evaluated will show at least one or more vulnerability in relation to mishandling user inputs. The expected measurable outcomes will be done using MobSF (Mobile security framework) which is a tool that can perform a wide range of security test on mobile applications such as malware analysis and static analysis which will be the point of focus while evaluating various Samsung applications.

1.5 Independent and dependent variables:

- Independent variables:
  independent variables that will be examined include system apps, third-party apps and high-risk apps. A selection of application from the three categories will be crucial when making comparisons on the impacts of user inputs vulnerabilities across different Samsung applications.
- Dependent variables:
  This study will expect to show the presence of user inputs security flaws and along with the type of user input vulnerabilities by the measure of severity level from normal, warning to high risk.

Overall, the research goal for this study on vulnerabilities in application related to mishandling to user inputs is to find those vulnerabilities across various Samsung applications and more importantly on high-risk applications such as financial apps that handle sensitive user information and compare them to other type of applications in Samsung or third-party apps. This research will demonstrate and presents proof of the severity or security threat of such vulnerabilities, if found.

2. Research Approach Selection (Quantitative or Qualitative) (Theoretical or Practical Research)

This research study will approach this research in a quantitative manner which will objectively aim to measure the widespread and severity of user input vulnerabilities across various application that will be examined. MobSF (Mobile security framework) will be the tool to be used in this study that will facilitate the uncovering of user input vulnerabilities across different mobile applications in the Samsung A12.

Selection of apps:

This research to effectively examine user input vulnerabilities, this study will extract various apps from the Samsung A12, from system apps to other apps that are compatible with Samsung phones especially apps from the Google apps store. Then this study shall include third-party apps that are not available in the app store but compatible with Samsung device.

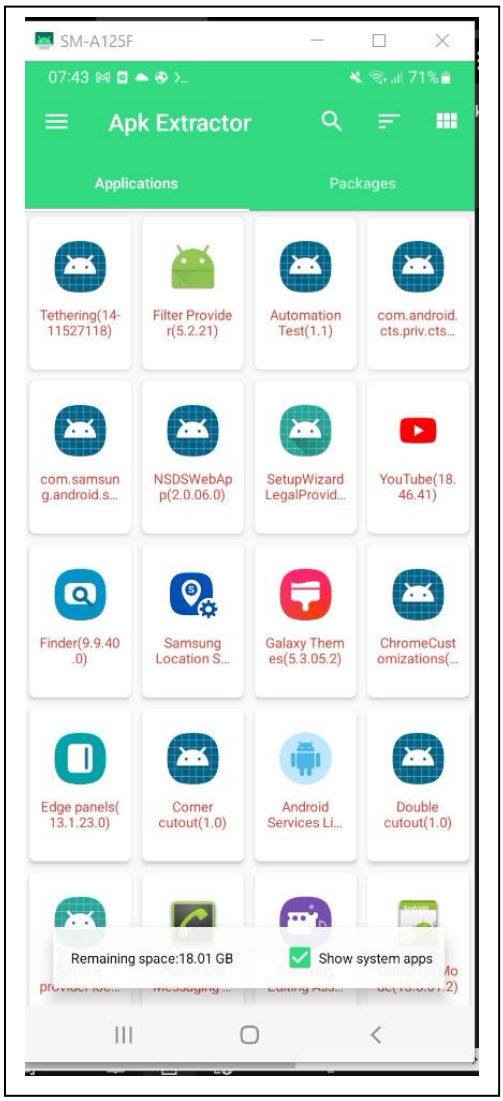
Mobile application selection categories:

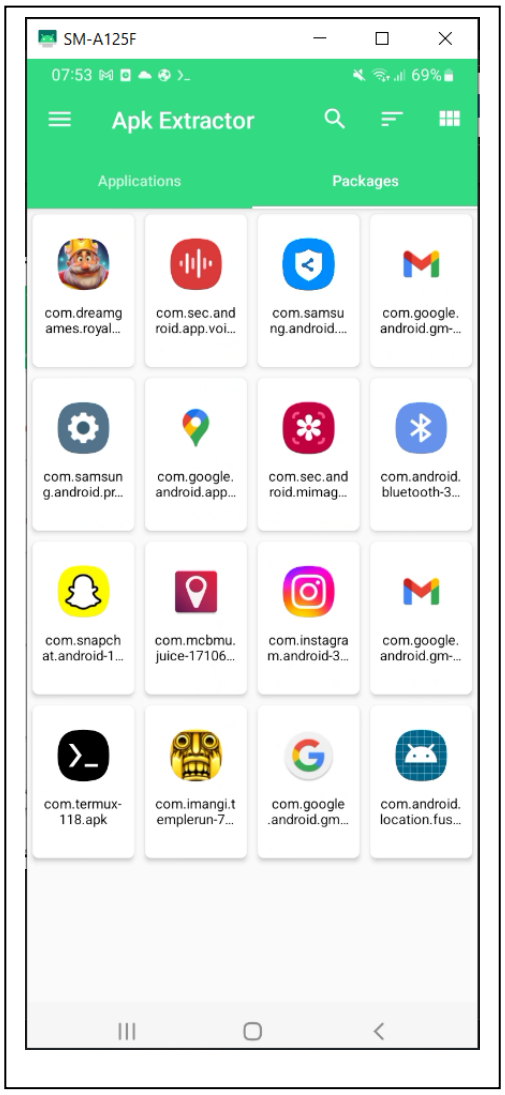
System apps	**Third-party apps**
• Bluetooth • Private share • Camera • Gallery • Contacts • Messages • Phone calls.	• Instagram • Temple run game app. • Royal match game app. • Termux • Gmail

• Phone and messaging storage. • Voice recorder • Photo editor	• MCB juice • Shein • Snapchat

2.1 Extraction of the apps:

The extraction of the apps was done through using the APK extractor that is an app that is used to extract the extract the APK (Android application package) files of the apps needed for this study. An APK extractor facilitates the process of acquire the APK files than doing it manually through a command line such as CMD.





The above figures show a selection of apps that will be examined from the android device. The random selection is crucial to the research goals that aim to presents various vulnerabilities related to mishandling of user inputs and how they impact the security of user data across different types of mobile application ranging from high-risk apps to low-level threat apps.

Setting up MobSF (Mobile security framework):

The testing tool environment is set up on a virtual machine environment. The tool will allow this study to perform various test such as static analysis, security analysis and malware analysis.

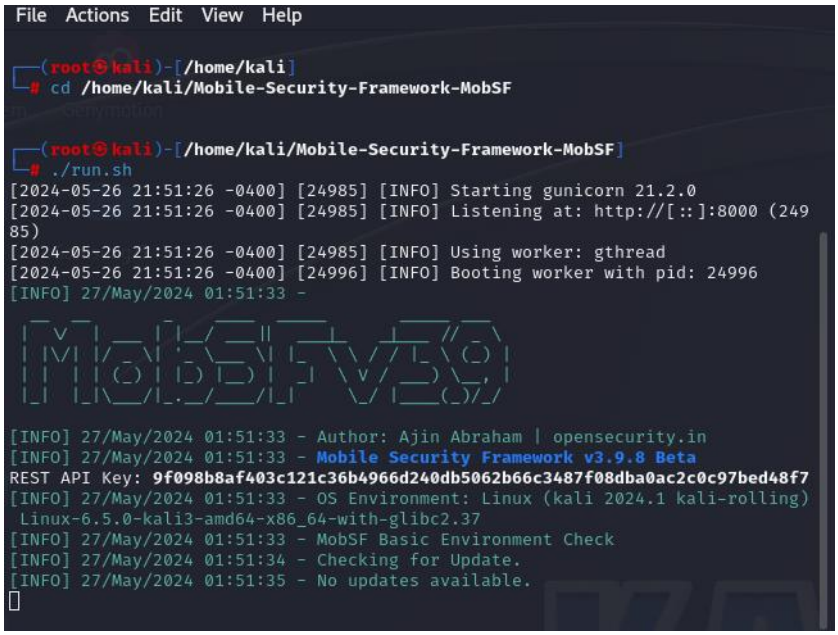


Figure 1: MobSf setup

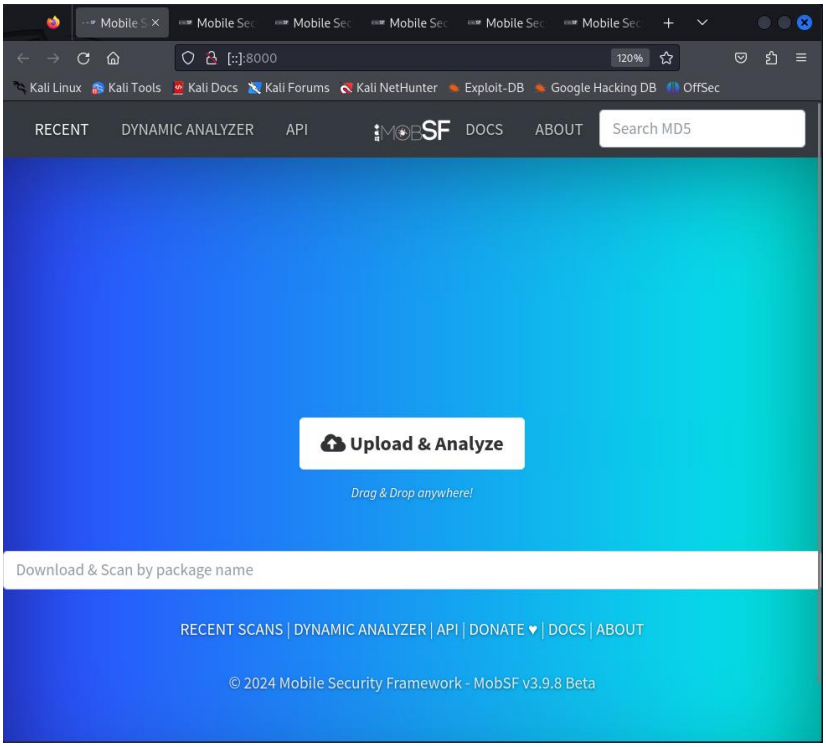


Figure 2:MobSF APK testing platform

2.2 Testing of APK files:

As shown in figure 2, the APK's shall be uploaded to the testing platform and the security shall be assessed. A total of six random selected apps that users commonly used for different purposes ranging from instant chat or browsing through social media apps to making transactions using their own bank applications. A static analysis is crucial and is one of the method used by the MobSF, it evaluates the source code of APK file and compare the codes to known industry standard for mobile applications such as the OWASP MASVS[2].

2.3 MobSF app scanning:

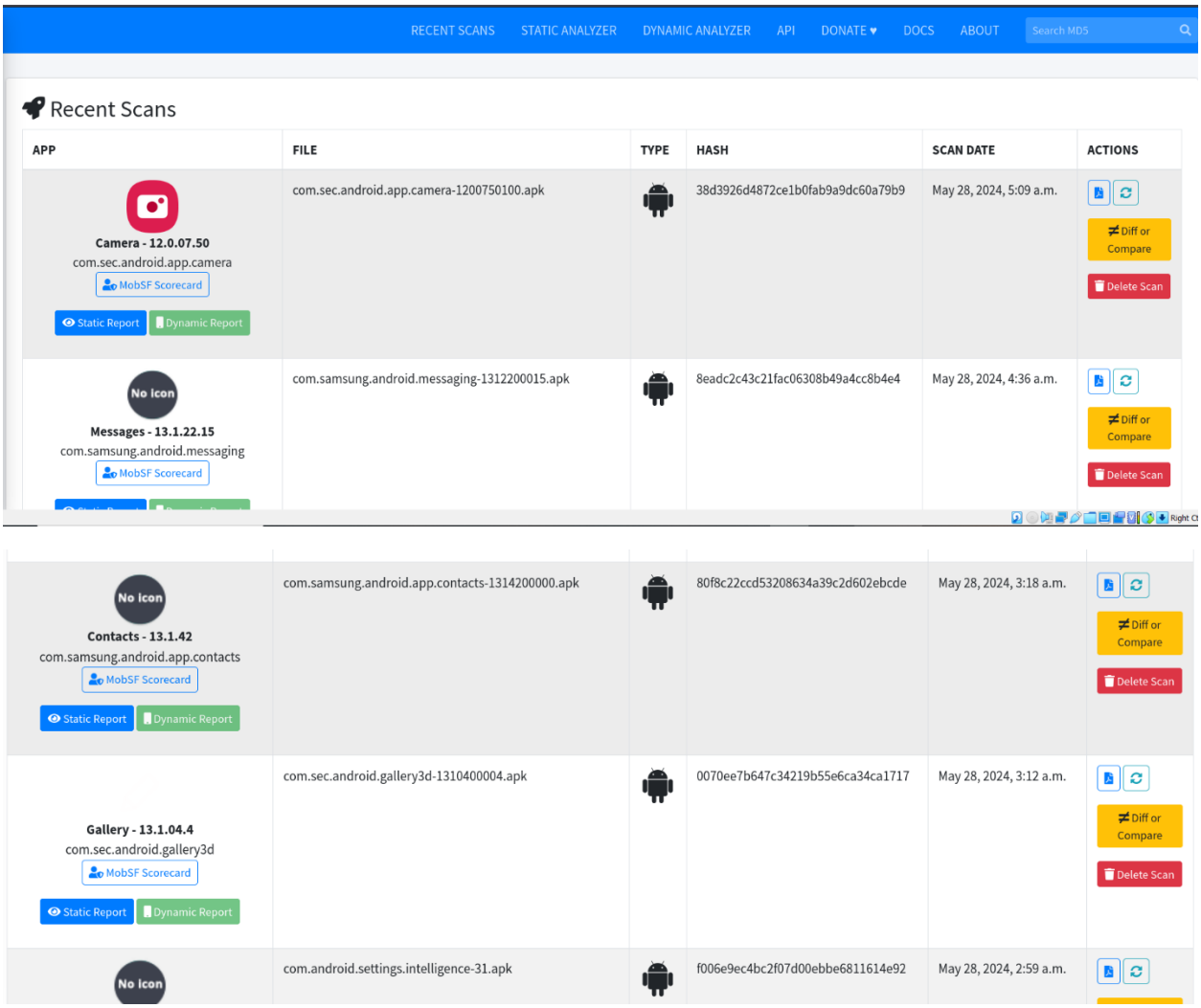


Figure 3:MobSF APK files scan.

2.4 Static report Findings:

This study is going to primarily focus on the static report of different APK files of apps extracted in the Samsung device. As this study hypotheses stated, it will focus in finding mishandling of user inputs and how data is impacted based on different applications ranging from system apps, popular apps, third party app and high-risk apps. The findings will mainly focus on the codebase and abused permissions of the app extracted along with its severity level. However, it is crucial to look out for vulnerabilities that might potentially impact user's data since MobSF conducts a wide range of test analysis including malware analysis.

- App name: camera
  category: system application

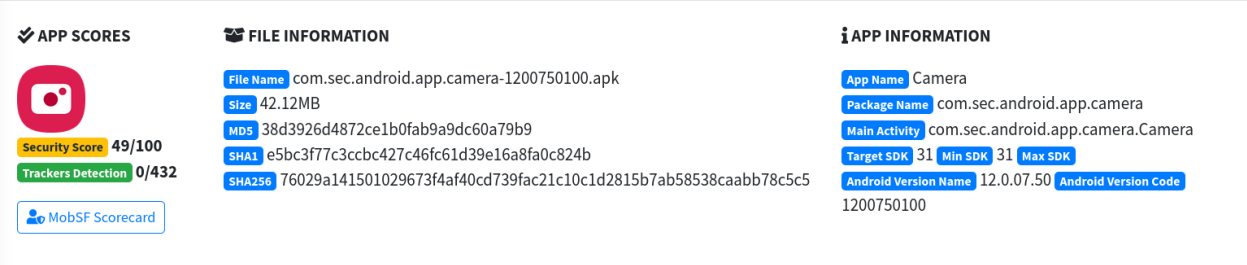


Figure 4:camera apk file information details.

The camera app detail information as shown in figure 4, it is crucial to assume that there will be vulnerabilities to check out basing on its security score of 49/100. However, it does not conclusively determine that the app is insecure based on the security score itself. There are more aspects of the app that need to be assessed.

Vulnerabilities found:

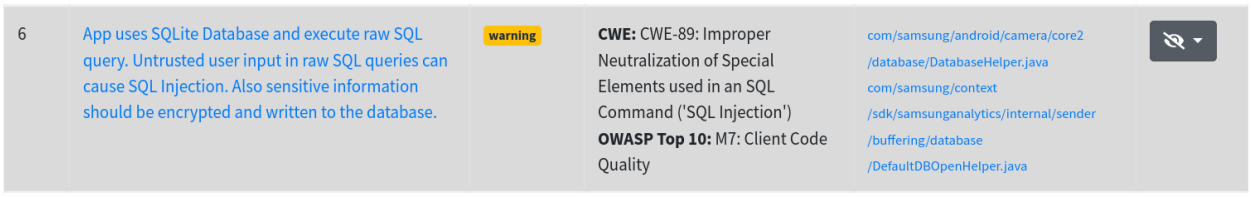


Figure 5: vulnerability of improper mishandling of user inputs in the camera app.

The vulnerability was found in the codebase analysis of the app. the app is found to be executing raw SQL queries and untrusted user inputs can open doors to SQL injection codes from attackers which can be used to execute SQL commands and potentially compromise sensitive data in the SQL database.

Mitigation is possible with such vulnerabilities. Encrypting user data before inserting into the database is crucial to protect sensitive data from unauthorized access of user data. furthermore, a good practice standard is provided by the OWASP and M7 is the standard for client code quality which provides guides on maintaining a good mobile application code quality such as maintaining consistent coding patterns and well -documented codes [3].

- App name: messages
  category: system application

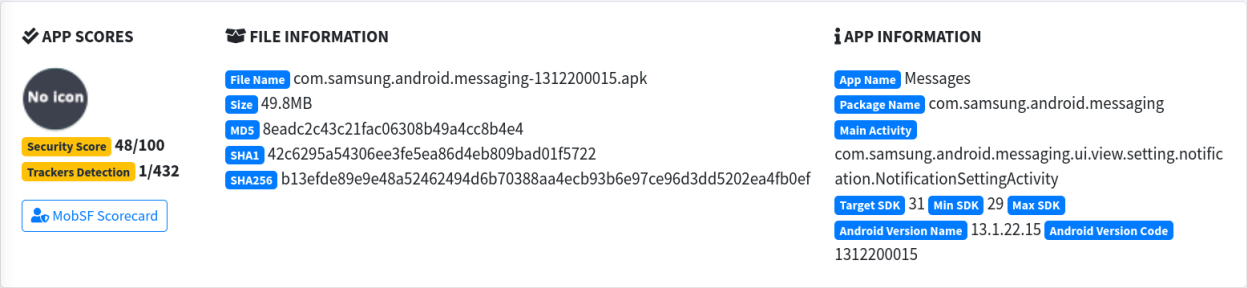
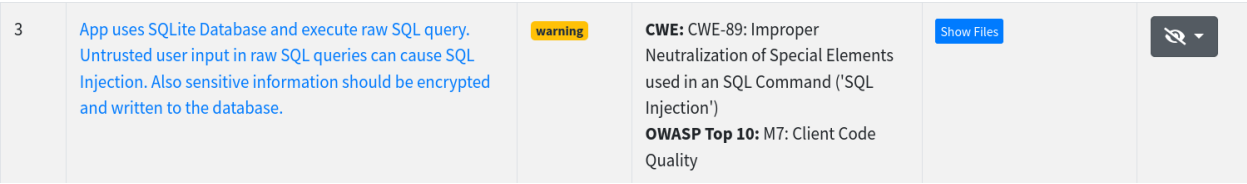


Figure 6:messages app.

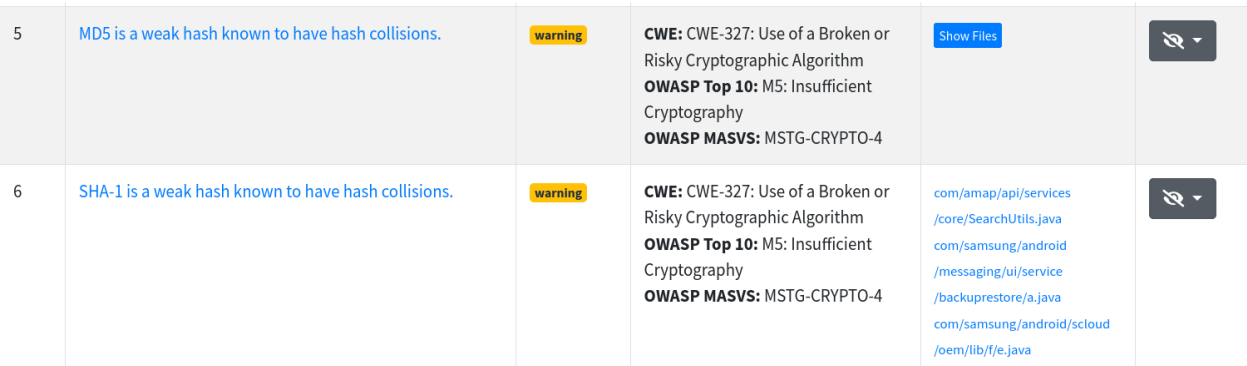
The system application is used for text messaging by using cellular phone services. according to it's security score it is assumably not well secured, but it is not conclusive without evaluating other aspects of the app. Furthermore, below the security score, there indication of trackers detection which can pose concern to privacy risk of data being collected without consent by third-party service providers. However, further analysis in the codebase might reveal whether user data might be exposed or collected unlawfully in relation to improper mishandling of user inputs.

Vulnerabilities found:



3	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	warning	**CWE:** CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') **OWASP Top 10:** M7: Client Code Quality	Show Files	

Figure 7: security flaw in the messages app.



5	MD5 is a weak hash known to have hash collisions.	warning	**CWE:** CWE-327: Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	Show Files	
6	SHA-1 is a weak hash known to have hash collisions.	warning	**CWE:** CWE-327: Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	com/amap/api/services/core/SearchUtils.java com/samsung/android/messaging/ui/service/backuprestore/a.java com/samsung/android/scloud/oem/lib/f/e.java	

Figure 8: cryptography security flaws in the messages app.

The system app has a severity level of warning which significantly raises security concerns of the mobile system app. The impact of such a security issue is that an attacker could exploit the security flaw and potentially read users confidential information from text messages or access contact details stored in the app. Furthermore, data modification is very possible once an attacker has accessed to the database or access to other apps by escalating privilege to further gain access to other applications that the app has permission to access.

The standard used to mitigate such risk of being victim to SQL injection falls under the OWASP M7 standard for producing good code quality by maintaining consistent coding patterns [3], security audits of the app code to identify security flaws in the code and personal information should use standard encryption before the data is written the database since the apps has shown weak or insufficient cryptography as shown in figure 7. Stronger hashes can be applied to mitigate the issue such as the SHA-256 hash which is currently the standard of encrypting data for mobile applications than using SHA-1 or MD5 which are vulnerable to collisions which means an attacker can produce two inputs that has the same value and can be used to swap original data for malicious data without being detected.

- App name: contacts
  App category: system app.

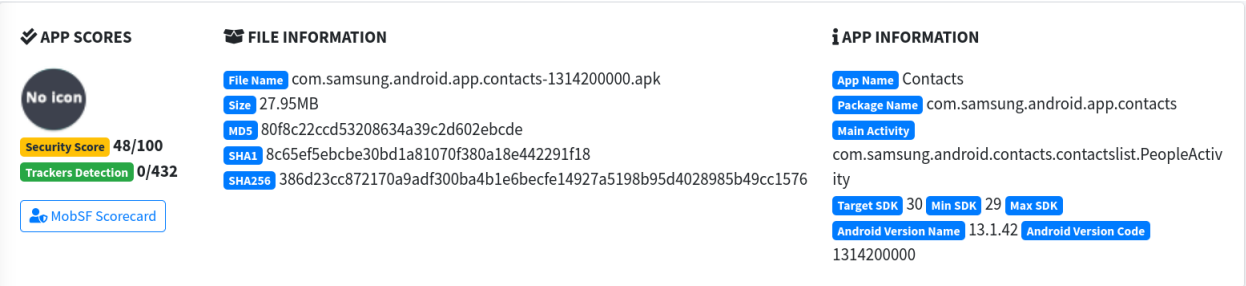
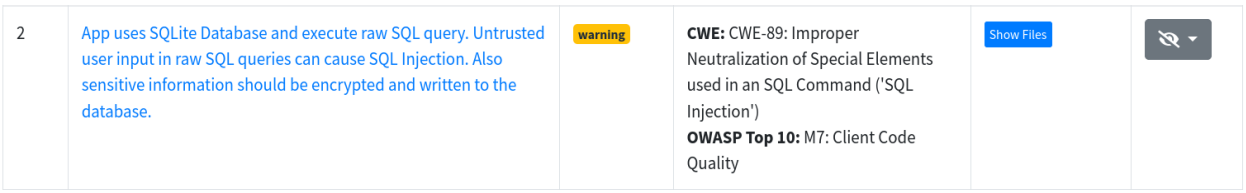


Figure 9: contacts app.

The contacts app in Samsung devices is a system app that stores contact details. The security score of the system appears to be under average which seems to be concern for a system app that stores confidential information especially contact details of other users. However, no trackers detected from other third-party service providers which is a good sign for it being secure in the part.

Vulnerabilities found:



2	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	warning	**CWE:** CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') **OWASP Top 10:** M7: Client Code Quality	Show Files	

Figure 10: security flaw found in the contacts app.

one security flaw related to mishandling of user inputs has been identified in the contacts codebase. The security flaw executes raw SQL queries which can be vulnerable to SQL injection as stated in the issue in figure 13. The same security standard provided by OWASP is required to counter such issues when found [3].

- App name: gallery
  category: system app

Figure 11: gallery app.

The gallery app is a system app that in used to stored images and based on its security score, it appears to not be secure enough. Additionally, the app has 1 tracker detected and presented as a warning in yellow which indicated there's

more to investigate. However, we shall maintain the focus of the hypotheses which focuses to identify vulnerabilities related to improper mishandling of user inputs.

Vulnerabilities found:

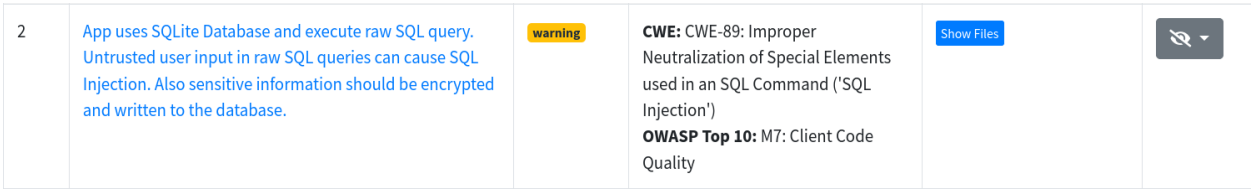


Figure 12: security flaw in the gallery app.

The gallery app has one security flaw related to improper mishandling of user inputs where it can execute raw SQL query. This is a major security issue for a system app due to the sensitive content the gallery app might contain such as videos and pictures of user which is considered personal data and must be always secured. Updates and patches can be the key to counter such as security flaws and providing regular updates too. Furthermore, encryption is a very crucial part for securing such sensitive data and encrypting data before being admitted into the SQLite database as the standard requires it from OWASP client code quality.

- App name: phone calls
  Category: system app.

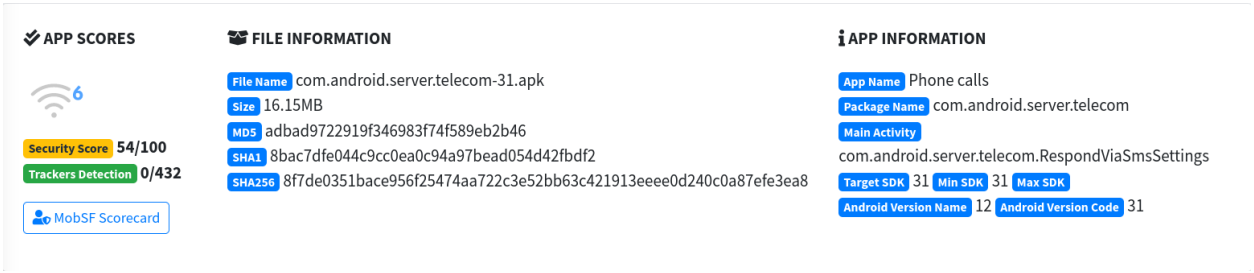


Figure 13: phone calls app.

The phone calls app being a system app has significance to this study since it can contain or store sensitive information such as contact information, call logs history and other personal information such as images. Basing on its security score in figure 12, the app should be significantly secure than having

a low score of 54. An insecure phone call app could have a huge impact on the user data security with risk of data leakage or eavesdropping on phone calls or recording of phone calls. Further investigative measures that can be taken is to evaluate its permissions that could tell if the app has access to unnecessary permissions or permissions that it should have access to.

Vulnerabilities found:

3	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	warning	**CWE:** CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') **OWASP Top 10:** M7: Client Code Quality	com/samsung/context/sdk/samsunganalytics/internal /sender/buffering/database/DbManager.java com/samsung/context/sdk/samsunganalytics/internal /sender/buffering/database/DefaultDBOpenHelper.java com/sec/android/diagmonagent/log/ged /db/GEDDatabase.java	

Figure 14: security flaw in phone calls app.

The phones calls app has one vulnerability related to improper mishandling of user data. execution of raw SQL queries can be used for SQL injections which can be impact the security of sensitive data that the device holds. Regular security audits are required for such high-risk vulnerabilities and regular updates and patches from the manufacturer are crucial to minimize such risks in the future.

ABUSED PERMISSIONS

Top Malware Permissions	8/24	Other Common Permissions	3/45
android.permission.READ_CALL_LOG, android.permission.READ_EXTERNAL_STORAGE, android.permission.SEND_SMS, android.permission.VIBRATE, android.permission.WAKE_LOCK, android.permission.ACCESS_FINE_LOCATION, android.permission.ACCESS_NETWORK_STATE, android.permission.RECEIVE_BOOT_COMPLETED		android.permission.BLUETOOTH, android.permission.BLUETOOTH_ADMIN, android.permission.MODIFY_AUDIO_SETTINGS	

Figure 15: phone calls app permissions.

The phone calls app appears to have no unexpected permissions that it shouldn't have access to, nor does it have any encryption issues. However, attackers could exploit the permissions using SQL injections and potentially misuse the permissions to access personal data. Security measures that can be taken include constant security audits and ensuring that data is encrypted using the SHA256 hash for data before they are written in the database.

- App name: phone and messaging storage.
  Category: system app.

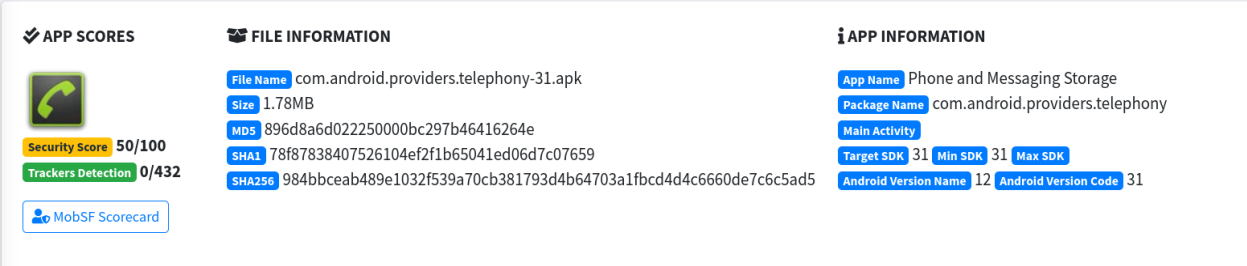


Figure 16:phone and messaging storage app.

The system app is responsible for storing data related to call logs, messages or SMS text and multimedia messages such as pictures and videos. The security score is significantly low, and any security flaws could impact severely the integrity of user data.

Vulnerabilities found:

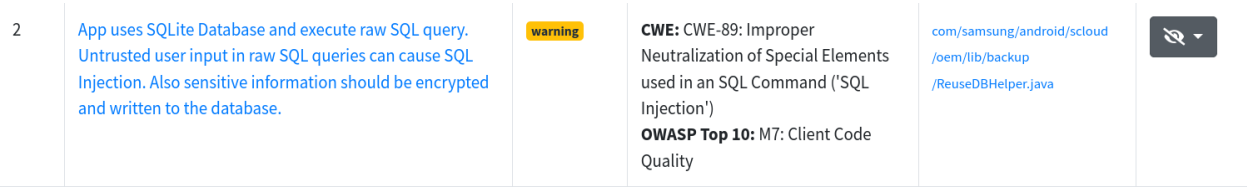


Figure 17: security flaw in phone and messaging storage app.

The system appears also have security flaws that can promptly jeopardize the integrity of user data. Execution of raw SQL queries seem to be common in system apps that were extracted in the Samsung A12. It can be that the codebase might be a legacy code or lack of awareness of potential risks by the developers of the system app [4]. Security measures that can be taken are regular security updates and patches to the codebase of the app.

- App name: photo editor.
  Category: system app.

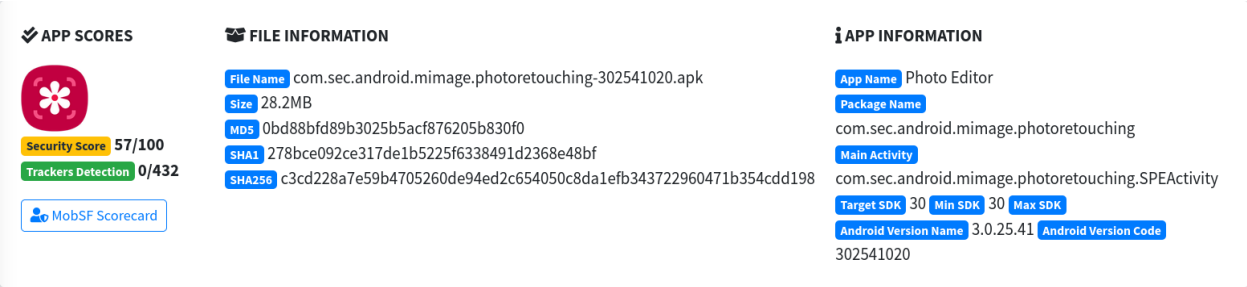


Figure 18:photo editor app.

The photo editor app is a system app of Samsung devices, and it has relevance to this study due the fact that the app can access sensitive or personal data of a user. The app is expected to be secure and it's security score does not reflect it at the highest level with it's score of 57%. A breach could severely impact the privacy of the user's data such as images and videos since it has such permissions such as access to geographic location.

PERMISSION	STATUS	INFO	DESCRIPTION	CODE MAPPINGS
android.permission.ACCESS_MEDIA_LOCATION	dangerous	access any geographic locations	Allows an application to access any geographic locations persisted in the user's shared collection.	Show Files

Figure 19: photo editor permission.

As shown in figure 25, the app has access to "any geographic locations" and is considered a dangerous permission for such an app. Upon further observation, this study can understand why such permissions to such an app that can hold personal data is a huge risk when attackers are able to breach the app through its vulnerabilities.

Vulnerabilities found:

NO	ISSUE	SEVERITY	STANDARDS	FILES	OPTIONS
1	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	warning	**CWE:** CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') **OWASP Top 10:** M7: Client Code Quality	Show Files com/samsung/context/sdk/samsunganalytics/h/h/h/b/a.java com/samsung/context/sdk/samsunganalytics/h/h/h/b/b.java com/sec/android/mimage/photoretouching/h/f/p.java com/sec/android/mimage/photoretouching/spe/controller/states/decoration/db/DecorationProvider.java	

Figure 20: security flaw in the photo editor app.

One vulnerability related to improper mishandling of user inputs. Execution of raw SQL commands in the query can lead to SQL injections. The impact to the integrity of the user's data could result in the exposure of the user's location data through the location access permissions. Security measures that are best for such security flaws can be to deny unnecessary permissions that the system app uses but does not need for its everyday tasks.

- App name: private share
  Category: system app

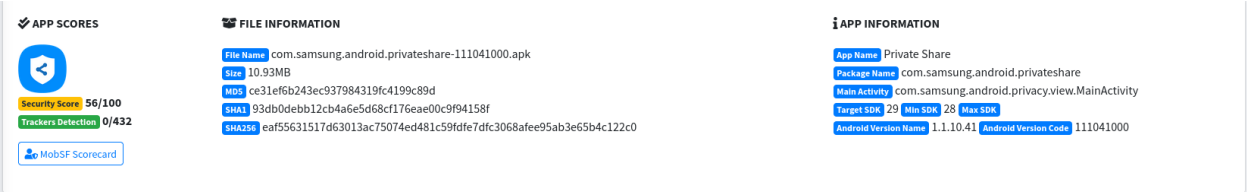


Figure 21:private share app.

The private share app is a system app of Samsung devices that allows users to securely share private files. The app uses encryption to ensure that unauthorized parties cannot access the files being shared between the

authorized parties only. The app is essential for this study to evaluate its security and potential impact of its flaws to user input data.

Vulnerabilities found:

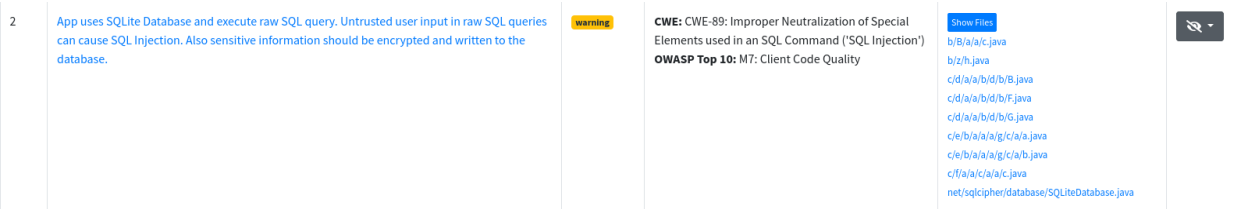


Figure 22: security flaw in the private app.

First vulnerability to address is the execution of raw SQL queries. Most of the system apps that have been evaluated so far, have shown to have that particular security flaw that can be victim to SQL injection, if the codebase is not properly safeguarded from such security mistakes [5]. Common best practices to ensure that user inputs are not passed as SQL commands or codes is to set the user inputs into parameterized queries and handle them as user data instead of SQL codes.

```
50.     @Override // b.B.a.b
51.     public void execSQL(String str) {
52.         this.f996c.execSQL(str);
53.     }
54.
```

Figure 23: insecure code using raw SQL queries.

As an example in figure 22, the piece of code shown is one example of unparameterized code that can be subject to SQL queries because it runs raw SQL queries. Line 51 in figure 22, displays a public method that takes a string and passes it as an argument. The method takes the raw strings and executes them without any checks or sanitizing the code before execution. The second line 52, is the method that executes the provided SQL string.

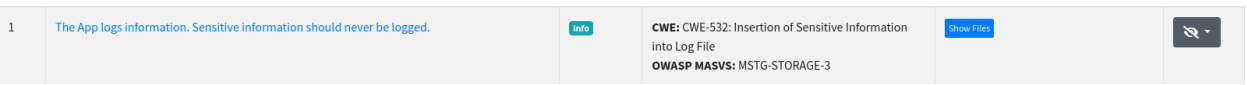


Figure 24: second user input security flaw.

Figure 23 is also a common security flaw that has been noticed in previously analyzed system apps. The security issue logs sensitive information into the log file which it should not do.

```
33.             }
34.             Log.w("SupportSQLite", "deleting the database file: " + str);
35.             try {
36.                 if (Build.VERSION.SDK_INT >= 16) {
37.                     android.database.sqlite.SQLiteDatabase.deleteDatabase(new File(str));
38.                 } else {
39.                     try {
40.                         if (!new File(str).delete()) {
41.                             Log.e("SupportSQLite", "Could not delete the database file " + str);
42.                         }
43.                     } catch (Exception e2) {
44.                         Log.e("SupportSQLite", "error while deleting corrupted database file", e2);
45.                     }
46.                 }
47.             } catch (Exception e3) {
48.                 Log.w("SupportSQLite", "delete failed: ", e3);
49.             }
50.         }
51.
52.         public void b(b.B.a.b bVar) {
53.             Log.e("SupportSQLite", "Corruption reported by sqlite on database: " + bVar.getPath());
54.             if (!bVar.isOpen()) {
55.                 a(bVar.getPath());
56.                 return;
```

Figure 25:security flaws in the log file.

Code breakdown: Line 34

- Log.w: android logging method used for log warnings
- “SupportSQLite”: this is a tag for log messages that helps in identifying the primary source of the log.
- ‘ “deleting the database file:” + str’: this log message integrates or joins a fixed string with the value of ‘str’.

Impact:

- Any user whether authorized or not authorized could access the log files when the ‘str’ variable contains sensitive information such as names or file paths. The access to such information could be easy for anyone including other applications that have permission to the app.

- App name: Bluetooth.
  Category: system app.

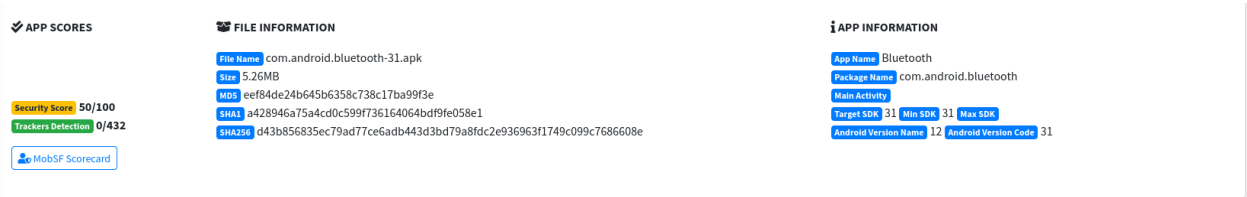


Figure 26: Bluetooth app.

The Bluetooth app is a system app that allows users to share data with other devices using radio frequencies over short distances through wireless connection. Two users must be close to each other to share data, but Bluetooth does more than sharing data. It can pair to audio speakers or even other devices that have Bluetooth. Interception of data being shared between two devices is very probable and to why the app needs regular security audits. Basing on the security score of the app, 50% is a low number for a commonly used app in many android devices. As the number of devices that uses Bluetooth grew Exponentially over 5 billion devices were set to be using the technology by the year 2021 [7]. The technology later started to be exploited by hackers as many security flaws came to light. For example, an issue was discovered that allowed attackers to take control of an android device (*The Hacker News*, 2023). The issue was an authentication bypass that attackers used to connect to devices without the user's confirmation and the attacker can inject keystrokes acting as a legitimate user of the device.

Vulnerabilities found:

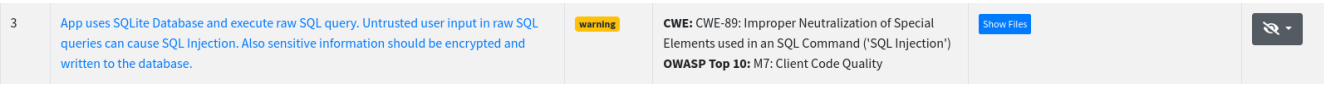


Figure 27: insecure database in the Bluetooth app.

Execution of raw SQL queries is a present user inputs mishandling vulnerability in the Bluetooth app. It is a significant issue that can result in unauthorized access to paired devices thus exposing critical data or control of the device [13].

- App name: voice recorder.
  Category: system app.

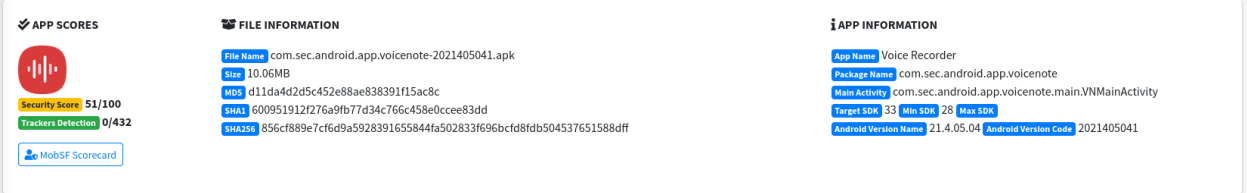


Figure 28: Voice recorder app.

Voice recorder is system app that allows users to record audio data. the audio content can be considered sensitive information of a user and crucial to this study to evaluate its security. Any breaches could severely harm the privacy of the user. observing the security of the app in figure 36, the app is arguably not secure enough and probable vulnerable to any attack.

Vulnerabilities found:

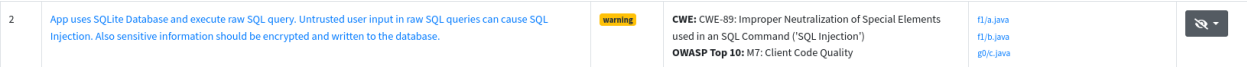


Figure 29: insecure log file and execution of raw SQL in the SQL database.

The app also shows vulnerabilities that other system apps extracted experienced execution of raw SQL queries. If the vulnerabilities are exploited, it could severely hinder the integrity of the user's data. Especially, if the attacker can access the database of recordings by code injection.

- App name: Instagram
  Category: third-party app

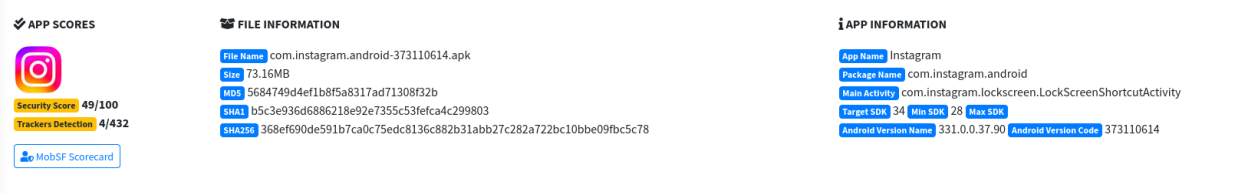


Figure 30: Instagram app.

Instagram is an app store popular social media app owned by Meta. It was selected to this study because popular apps will tend to experience more breaches from attackers due to its popularity. It is crucial to evaluate its security due to the personal data that is shared through the app daily. Based on the security score, the score is low for such a popular app.

Vulnerabilities found:

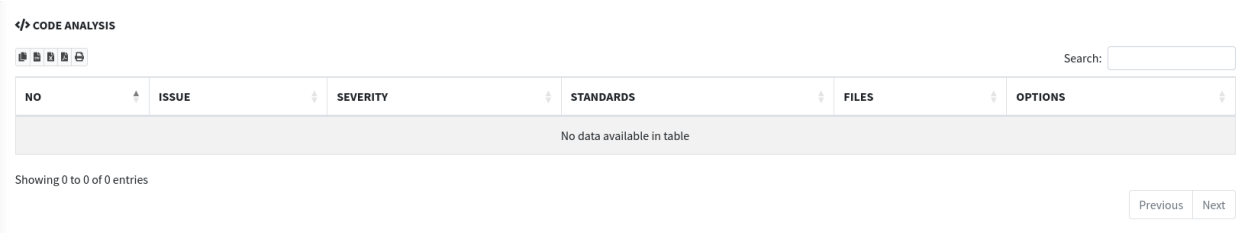


Figure 31: Instagram codebase.

The codebase showed no signs of vulnerabilities despite the low security score in figure 38. This could have many reasons as to why the codebase Is showing no security flaws. First reason, the Meta company might be protecting the codebase of the app from other parties for security reasons such as reverse engineering. Second reason, reputation damage, Instagram due to its popularity, it would be detrimental due to the company when its security flaws are exposed. Third reason, the codebase might not be available or for vulnerability scans such as MobSF. Fourth reason, the codebase could be vulnerability free and other section of the scan could have contributed to the low security score. However, while evaluating the app safety details that

are available in the Google app store, crucial information is mentioned that users are obligated to understand before downloading and using the app [9].

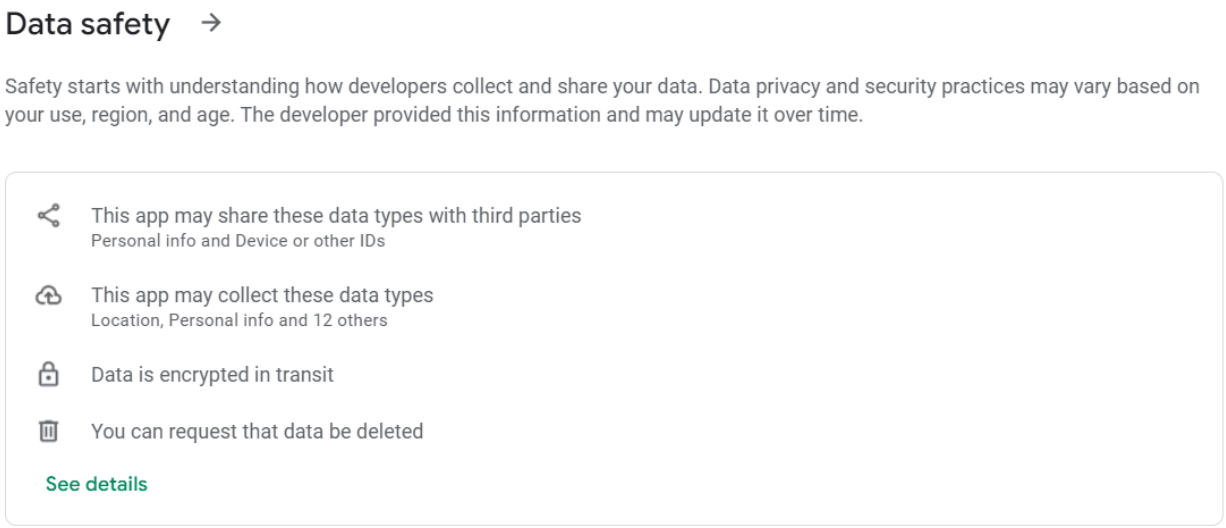


Figure 32: app store data safety details.

Important notice that data may share data with third parties including personal information. The share of personal information is an indicator that more information can be shared with third parties and the probability that those third parties can be subjected to data breaches is highly probable. However, Instagram being a major application of a company like Meta, the company employs the best developers and follows best practices for a secure codebase.

- App name: temple run game app.
  Category: third-party app

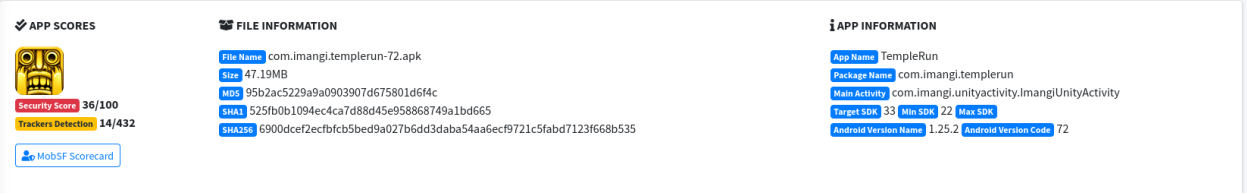


Figure 33: Temple run game app.

Temple run is an app store popular game app on many smartphone devices including Samsung devices. Besides being an app, users may not know that it can be vulnerable to attacks and user data could be jeopardized or exposed. One feature of the app allows users to login online and compete with other

players on who can attempt to get a higher score and through the process, personal information is inserting. Based on the app security score, the app security is significantly low. However, the app has no security flaw related to improper mishandling of user inputs of personal information such as vulnerabilities related to the SQL database unlike other apps that were evaluated in this study.

Vulnerabilities found:

NO	ISSUE	SEVERITY	STANDARDS	FILES	OPTIONS
1	The App logs information. Sensitive information should never be logged.	info	**CWE:** CWE-532: Insertion of Sensitive Information into Log File **OWASP MASVS:** MSTG-STORAGE-3	Show Files	
2	MD5 is a weak hash known to have hash collisions.	warning	**CWE:** CWE-327: Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	com/ironsource/sdk/controller/s.java com/mbridge/msdk/foundation/tools/aa.java	
3	The App uses an insecure Random Number Generator.	warning	**CWE:** CWE-330: Use of Insufficiently Random Values **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-6	com/applovin/exoplayer2/h/z.java com/mbridge/msdk/playercommon/exoplayer2/source/ShuffleOrder.java	
4	IP Address disclosure	warning	**CWE:** CWE-200: Information Exposure **OWASP MASVS:** MSTG-CODE-2	Show Files	
5	SHA-1 is a weak hash known to have hash collisions.	warning	**CWE:** CWE-327: Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	com/applovin/impl/sdk/utils/StringUtils.java com/imangi/googleplayservices/GameHelperUtils.java	
6	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	warning	**CWE:** CWE-312: Cleartext Storage of Sensitive Information **OWASP Top 10:** M9: Reverse Engineering **OWASP MASVS:** MSTG-STORAGE-14	Show Files	
7	Debug configuration enabled. Production builds must not be debuggable.	high	**CWE:** CWE-919: Weaknesses in Mobile Applications **OWASP Top 10:** M1: Improper Platform Usage **OWASP MASVS:** MSTG-RESILIENCE-2	com/unity/purchasing/BuildConfig.java	

Figure 34: security flaws in temple run game.

No vulnerabilities related to mishandling of user inputs were found in the temple run game app. However, the game app showed to have common issues that other scanned APKs experienced such as insertion of sensitive information into the log file, poor hashes that are common to experience collision such as MD5 and SHA-1 as shown in figure 42.

Another aspect of this evaluation of the temple run game app is the data safety details provided in the Google app store.

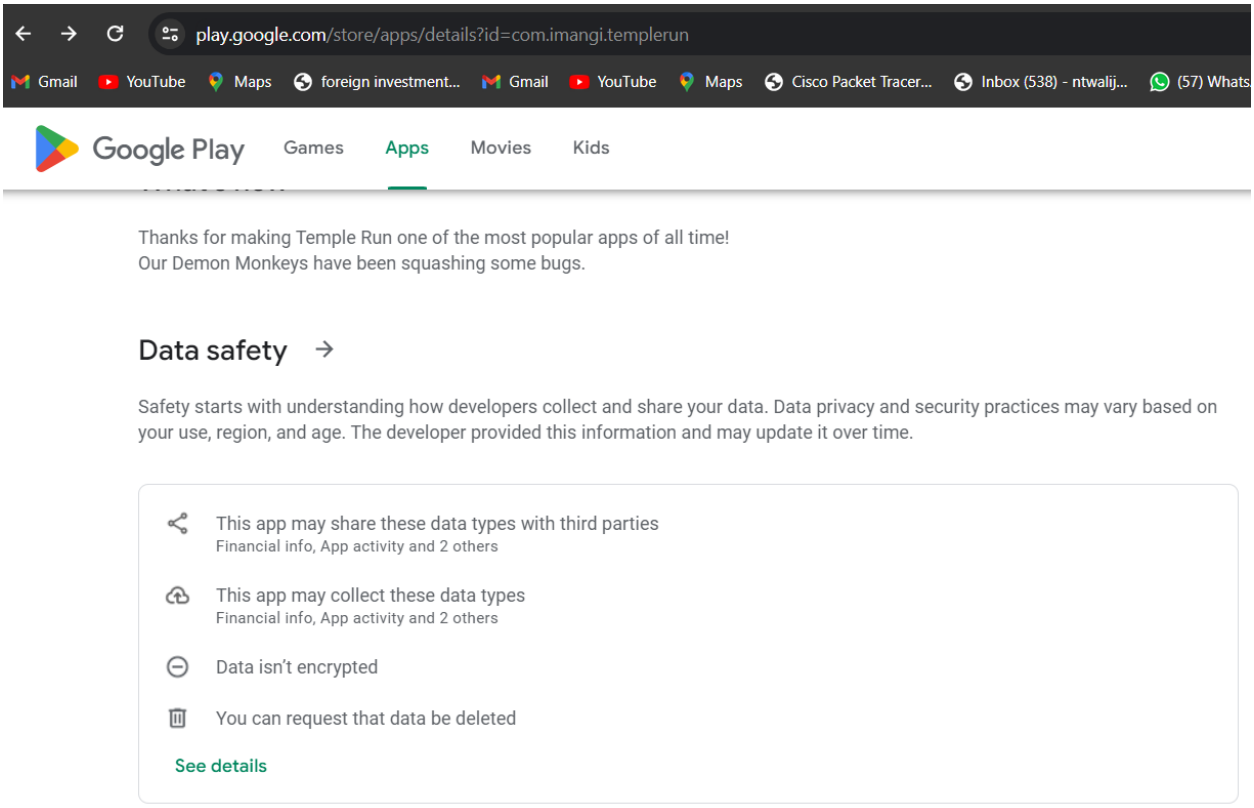


Figure 35: temple run data safety details.

The game app also shares data with third parties as other app store apps do such as Instagram. However, data isn't encrypted for game app. It is a serious vulnerability for a popular game app despite the developers mentioning the security flaw in its data safety details. It is worth mentioning the security flaw even though it is not related to improper mishandling user inputs and being an encryption issue. The Google play protect shows is strength in running safety checks for all apps and presents the findings to user before installing the app [11].

- App name: Royal match game.
  Category: third-party app.

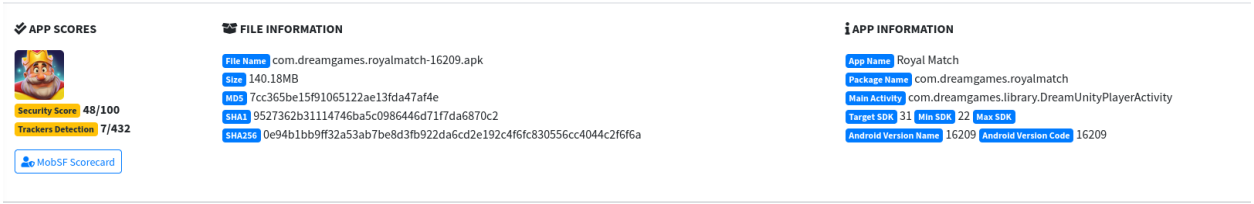


Figure 36: Royal match game app.

Royal match is a popular app store game application. The app is not as popular as temple run which has 500 million plus downloads compared to 100 million plus downloads of royal match [12] [10]. The security score of the game app is low and a high number of trackers are detected in the game.

Vulnerabilities found:

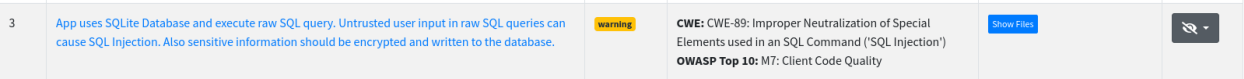


Figure 37:security flaw in the royal app game.

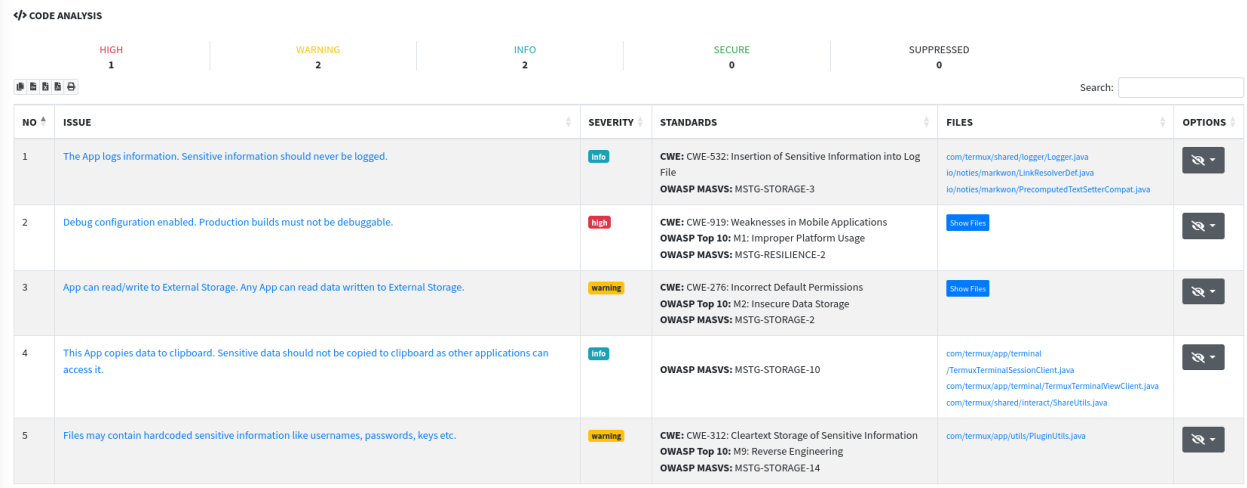
The app appears to have vulnerabilities that were discovered in the system apps of the Samsung A12 device. The execution of raw SQL queries that can be exploited by attackers by running manipulated SQL code commands that could expose personal information.

- App name: Termux
  Category: third-party application.

Figure 38: Termux app.

Termux app is a terminal emulator and Linux environment app for android devices. The termux app provides users with a Linux environment for mobile devices the same way Linux PC (personal computer) environments would be. It allows users to run the Linux distribution and run Linux commands from their android device. Furthermore, termux is usually used to set up kali Linux environments in the mobile devices and be used for penetration testing purposes. The security score of the termux app is significantly low and can be considered a potential threat for user data in the Samsung device.

Vulnerabilities found:



</> CODE ANALYSIS

HIGH	WARNING	INFO	SECURE	SUPPRESSED
1	2	2	0	0

Search:

NO	ISSUE	SEVERITY	STANDARDS	FILES	OPTIONS
1	The App logs information. Sensitive information should never be logged.	info	**CWE:** CWE-532: Insertion of Sensitive Information into Log File **OWASP MASVS:** MSTG-STORAGE-3	com/termux/shared/logger/Logger.java io/noties/markwon/LinkResolverDef.java io/noties/markwon/PrecomputedTextSetterCompat.java	
2	Debug configuration enabled. Production builds must not be debuggable.	high	**CWE:** CWE-919: Weaknesses in Mobile Applications **OWASP Top 10:** M1: Improper Platform Usage **OWASP MASVS:** MSTG-RESILIENCE-2	Show Files	
3	App can read/write to External Storage. Any App can read data written to External Storage.	warning	**CWE:** CWE-276: Incorrect Default Permissions **OWASP Top 10:** M2: Insecure Data Storage **OWASP MASVS:** MSTG-STORAGE-2	Show Files	
4	This App copies data to clipboard. Sensitive data should not be copied to clipboard as other applications can access it.	info	**OWASP MASVS:** MSTG-STORAGE-10	com/termux/app/terminal /TermuxTerminalSessionClient.java com/termux/app/terminal/TermuxTerminalViewClient.java com/termux/shared/interact/ShareUtils.java	
5	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	warning	**CWE:** CWE-312: Cleartext Storage of Sensitive Information **OWASP Top 10:** M9: Reverse Engineering **OWASP MASVS:** MSTG-STORAGE-14	com/termux/app/utils/PluginUtils.java	

Figure 39: security flaws in Termux.

the termux app codebase has no security flaws related to mishandling of user inputs as seen in previously scanned APKs. The low security score must be related to other vulnerabilities identified by MobSF such as high-level permissions that the termux apps has access to.

PERMISSION	STATUS	INFO	DESCRIPTION	CODE MAPPINGS
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.	Show Files
android.permission.DUMP	SignatureOrSystem	retrieve system internal status	Allows application to retrieve internal status of the system. Malicious applications may retrieve a wide variety of private and secure information that they should never commonly need.	
android.permission.FOREGROUND_SERVICE	normal	enables regular apps to use Service.startForeground.	Allows a regular application to use Service.startForeground.	Show Files
android.permission.INTERNET	normal	full Internet access	Allows an application to create network sockets.	Show Files
android.permission.PACKAGE_USAGE_STATS	signature	update component usage statistics	Allows the modification of collected component usage statistics. Not for use by common applications.	
android.permission.READ_LOGS	dangerous	read sensitive log data	Allows an application to read from the system's various log files. This allows it to discover general information about what you are doing with the phone, potentially including personal or private information.	
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	normal	permission for using Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS.	Permission an application must hold in order to use Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS.	
android.permission.REQUEST_INSTALL_PACKAGES	dangerous	Allows an application to request installing packages.	Malicious applications can use this to try and trick users into installing additional malicious packages.	
android.permission.SYSTEM_ALERT_WINDOW	dangerous	display system-level alerts	Allows an application to show system-alert windows. Malicious applications can take over the entire screen of the phone.	Show Files
android.permission.VIBRATE	normal	control vibrator	Allows the application to control the vibrator.	Show Files

Figure 40: Termux permissions.

The termux appears to have some dangerous permissions that can be harm the integrity of personal data. one of the permissions named “android.permission.READ.LOGS” that can read all types of data of the mobile device and could include personal information.

- App name: Gmail
  category: third party app

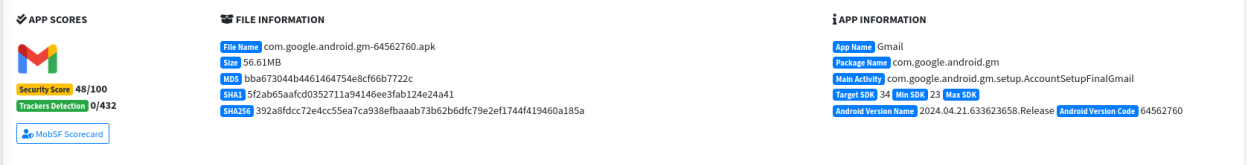


Figure 41: Gmail app.

The Gmail app is a very commonly used email application. The security score of Gmail is unexpectedly low. However, due to its popularity, the application might be victim to all sort of attacks from threat actors. Furthermore, the security score might be low due to several analysis done by MobSF and deemed not secure enough for an email mobile application. This study shall focus more on mishandling of user inputs vulnerabilities and check whether the email app has such security flaws.

Vulnerabilities found:

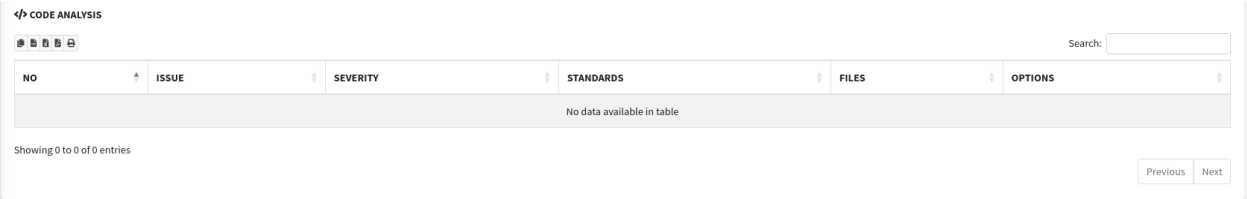


Figure 42: Gmail codebase.

The codebase shows no signs of security flaws. It is expected due to the high quality of coding by developers of the Google company. Companies such as Google hire the best mobile app developers and regular fix vulnerabilities that are discovered. Therefore, no user input vulnerabilities were found for the Gmail app.

- App name: Shein
  Category: third-party app

Figure 43:Shein app.

Shein is a popular online shopping app that is used by many due to its low pricing of goods. The security score is low unexpectedly since it is supposed to be secure because it handles financial information from users that make purchases on its app.

Vulnerabilities found:

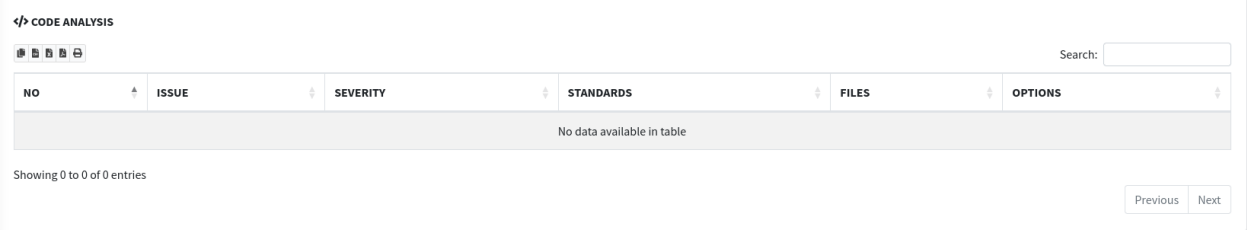


Figure 44: code analysis for the Shein app.

No vulnerabilities were found in the code analysis. Shein is a popular online shopping platform, and the company surely employs the best coding professionals and applies good coding practices. Since the app obtains financial information from customers, the app must be regularly updated and monitored from any security flaws that might arise and put the security of user data at risk.

- App name: Snapchat
  Category: third-party app.

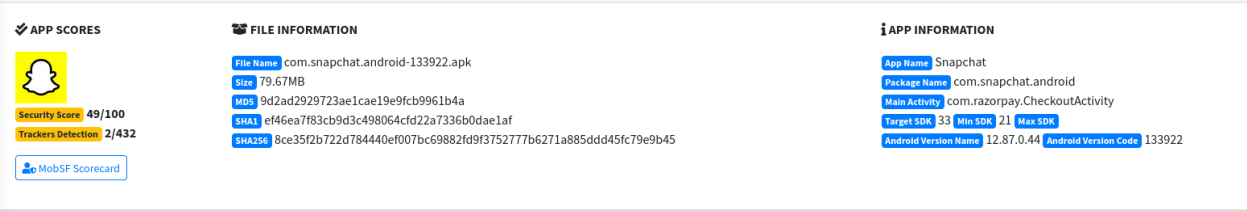


Figure 45: Snapchat app.

snapchat app is a popular social app like Instagram for instant messaging but has more user permissions than Instagram. It can access location in the background and share it with other snapchat users. However, many of the permissions are approved by the user. The security score is low for such an app that has high user permissions which is expected.

Vulnerabilities found:

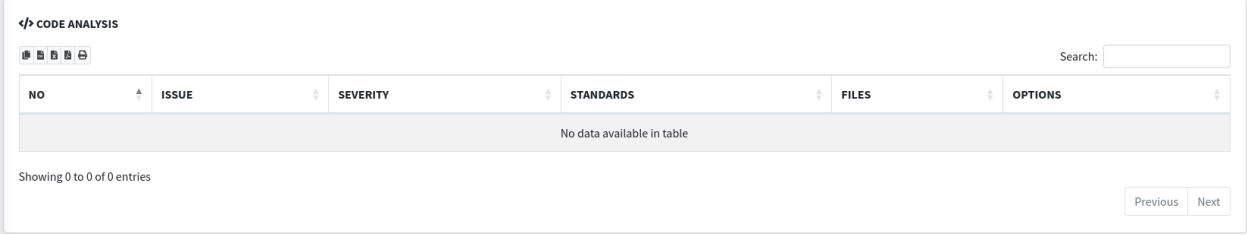


Figure 46:snapchat codebase analysis.

The snapchat app has no vulnerabilities related to mishandling of user inputs. Snapchat is a huge company. The low security score is related to other security analysis performed by MobSF. Some of the vulnerabilities detected by MobSF include dangerous permissions used by snapchat.

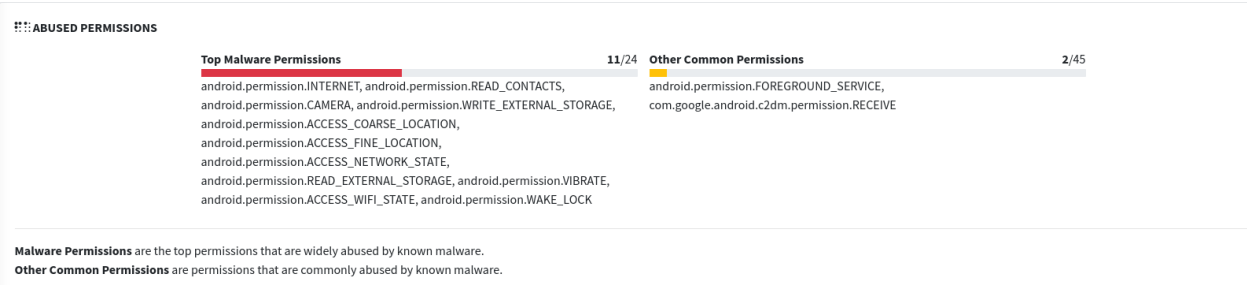


Figure 47: snapchat app permissions.

Some of the dangerous permissions include access to GPS (Global position service) in the background and sharing the location with other users. The access and read of contacts of the user's device. Such dangerous permissions are known to be exploited by malware.

- App name: MCB Juice
  Category: third-party application

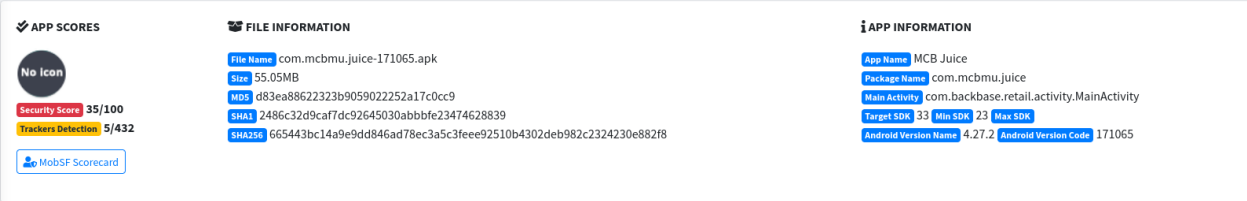


Figure 48: MCB juice app.

The MCB Juice app is a banking app of the Mauritian commercial bank used for making different kinds of transactions. It is crucial to this study since banking mobile applications are on a high rise of apps being developed and used by users to make their life more cost effective. Banking apps are expected to employ the best security measures to protect user's information. However, the security score of the app in MobSF after the scan reflects a different view. The score is significantly very low for a banking app.

Vulnerabilities found:

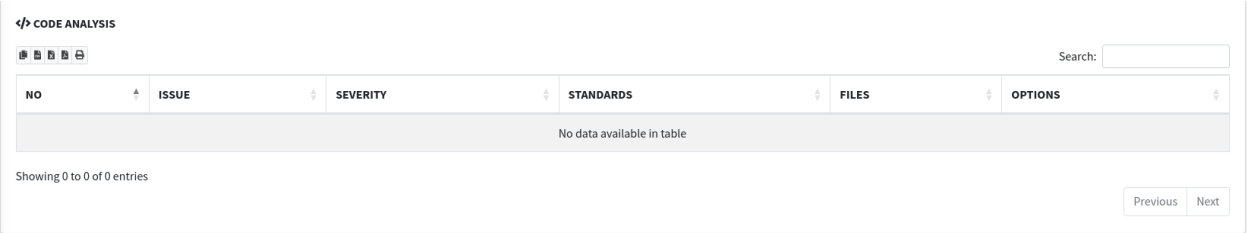


Figure 49: codebase analysis of MCB juice app.

The app has no vulnerabilities in the codebase analysis thus no vulnerabilities related to mishandling of user inputs were found. Banking apps usually employ professional mobile application developers that apply the best coding practices and regularly audit the app to fix any vulnerabilities.

3. Identification of Variables and Measures

3.1 Data structure:

Application	Android OS version	Application android version name	Security test type	Number of vulnerabilities (user inputs vulnerabilities)	Types of user inputs vulnerabilities	Severity
Bluetooth	12	12	Static analysis, security analysis	1	SQL injection	Warning
Private share	12	1.1	Static analysis, security analysis	1	SQL injection	Warning
Camera	12	12.0	Static analysis, security analysis	1	SQL injection	Warning
Gallery	12	13.1	Static analysis, security analysis	1	SQL injection	Warning
Contacts	12	13.1	Static analysis, security analysis	1	SQL injection	Warning
Messages	12	13.1	Static analysis, security analysis	1	SQL injection	Warning
Phone calls.	12	12	Static analysis, security analysis, malware analysis	1	SQL injection	Warning
Phone and messaging storage.	12	12	Static analysis,	1	SQL injection	Warning

			security analysis			
Voice recorder	12	21.4	Static analysis, security analysis	1	SQL injection	Warning
Photo editor	12	3.0	Static analysis, security analysis	1	SQL injection	Warning
Instagram	12	331.0	Static analysis, security analysis	0	None	None
Temple run game app.	12	1.25	Static analysis, security analysis	0	None	None
Royal match game app.	12	16209	Static analysis, security analysis	1	SQL injection	Warning
Termux	12	0.118	Static analysis, security analysis	0	None	None
Gmail	12	2024.04	Static analysis, security analysis	0	None	None
MCB juice	12	4.27	Static analysis, security analysis	0	None	None
Shein	12	11.0	Static analysis,	0	None	None

			security analysis			
Snapchat	12	12.87	Static analysis, security analysis	0	None	None

3.2 Independent variables:

independent variables will consist of types of security tests conducted for the mobile applications extracted from the Samsung A12 device. MobSF performs several tests including static analysis. Static analysis has sub tests of its own that include security test that is used to review the codebase of the APK file. Malware analysis is another sub test in static analysis and is mainly used to examine permissions that the mobile application use and permissions that are mostly known to be abused by malware.

3.3 Dependent variables:

Dependent variables consist of the number of vulnerabilities found when examining mobile applications and types of vulnerabilities. In the data structure table, the data collected is classified by the number of vulnerabilities found for each app related to improper mishandling of user inputs, the type of vulnerability found such as SQL injection security flaws and the severity level of the vulnerability. Classification is done using the CWE (common weakness enumeration) that classifies the vulnerability found. Example, vulnerabilities related to the execution of raw SQL queries are classified as SQL injection or information exposure for issues such as IP disclosure.

4. Sample Size and Sampling Methodology (ensure good representation of the population)

The selection of mobile applications to test was based on system apps and third-party apps that are not pre-installed but are available in the app store of the mobile device. The selection was also based on ensuring to tests the two types of apps (system and third-party apps) in search of vulnerabilities related to user inputs which were found in both types of apps. The most common security flaw was SQL injection which was present in almost all system apps and some third-party apps. The selection ensured a good representation of testing to reach good conclusive evidence on the state of the mobile device which ran an android 12 version. As stated in the hypotheses, this study was able to find at least 50% of apps scanned to have improper mishandling of user inputs in the codebase.

5. Methods of Data Collection (surveys, interviews, observations, experiments, etc.)

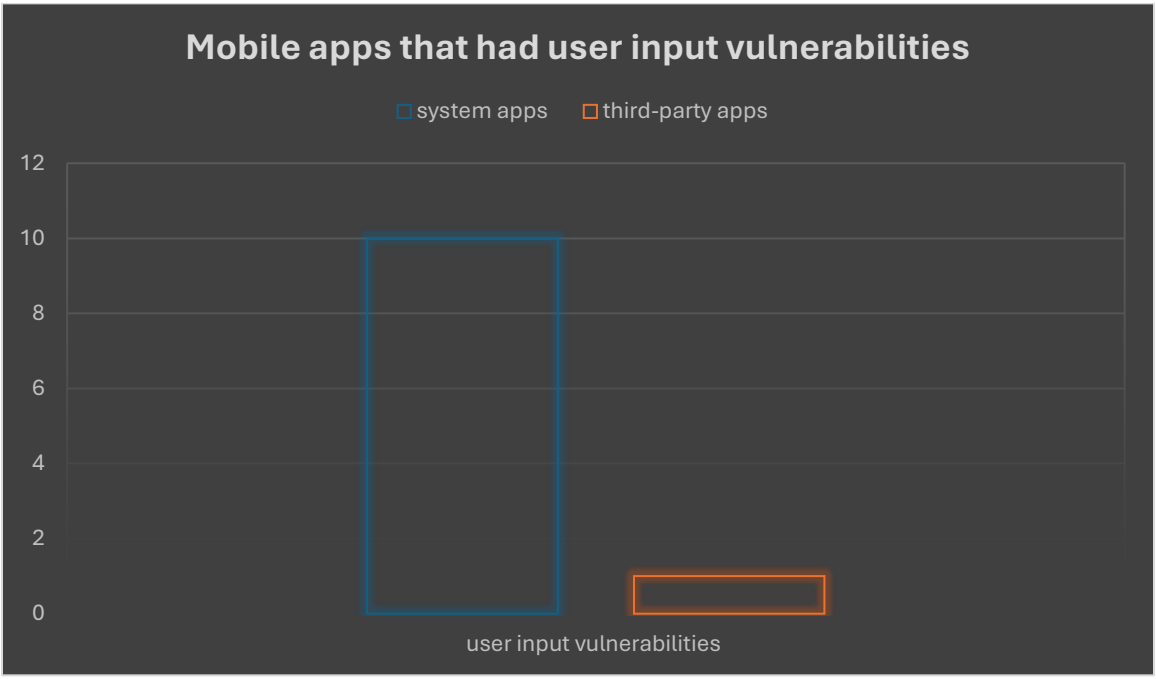
Collection of data consisted of using a security scanning tool named MobSF (Mobile security framework) which examined sourced code of the extracted APK files from the android Samsung device. Static analyzer was a method of data collection which consisted of information about the APK file, security evaluation and the malware analysis in among several tests conducted by the static analyzer. Some of the other tests performed by the static analyzer included evaluating permissions used by the app and classifying the severity of permissions.

6. Analysis Techniques

The analysis aspect of this study will include a focus on the prevalence of vulnerabilities found related to mishandling of user inputs such as SQL

injection that was observed in security scans done by MobSF (mobile security framework). This study shall examine the prevalence of user-inputs using the selection of apps to determine which mobile types of apps show to have a consistency in having vulnerabilities related to mishandling of user inputs. This analysis is determined to show the contrast aspect of apps between system apps and third-party apps since system apps may not receive regular updates and patches than third-party apps. A comparison between the two types of apps should give a clear view on which apps are likely to have vulnerabilities of improper handling of user inputs on the android version 12.

6.1 Number of apps that have user inputs vulnerabilities



Samsung's device system apps upon observation, have shown to have the highest number of mishandle of user input vulnerabilities than third-party apps from the few selected. Due to lack of updates and patches in time, different apps can be exploited and can lead to data breaches. However, the discovery of such security vulnerabilities in system apps of the Samsung device does not mean that any data breaches occurred but means that the vulnerabilities discovered are among the security flaws documented to have

caused data breaches. Documentation such as the CVE (common vulnerability and exposures) report in details on the vulnerability with its potential impact by using the base score ranging from 0 t0 10 to characterize its severity level which is called the CVSS (common vulnerability scoring system) [14].

6.2 Relationship between apps and the potential impact of SQL injections:

Relationships between apps such as contacts, messages and phone calls can aggravate the risk of exposing more information at once than a threat actor attacking one app independently. The relationship lies in app permission that the apps share and some of the app permission include permission to read external storage and call logs.

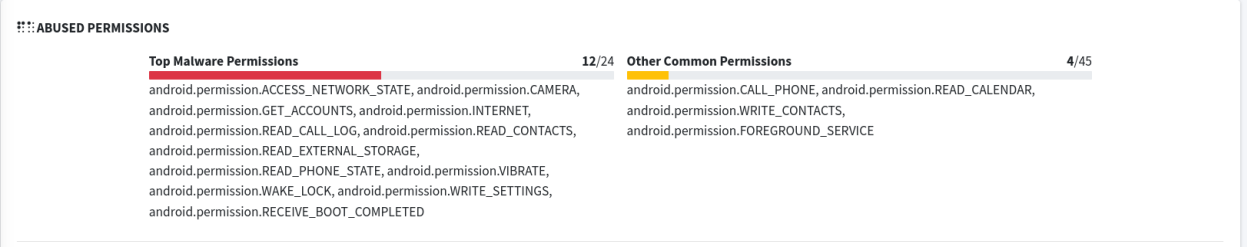


Figure 50: app permissions for the contacts app.

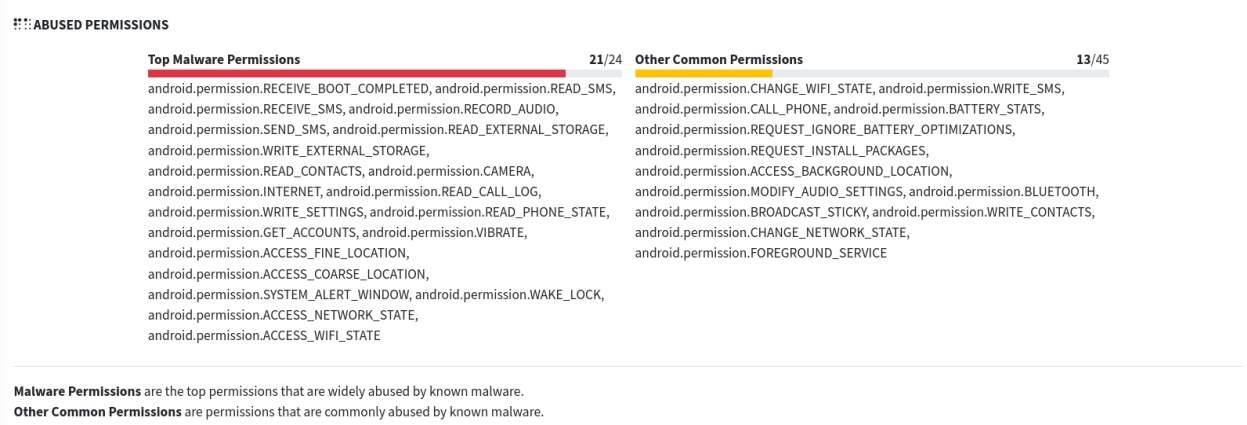


Figure 51: app permissions for the messages app.

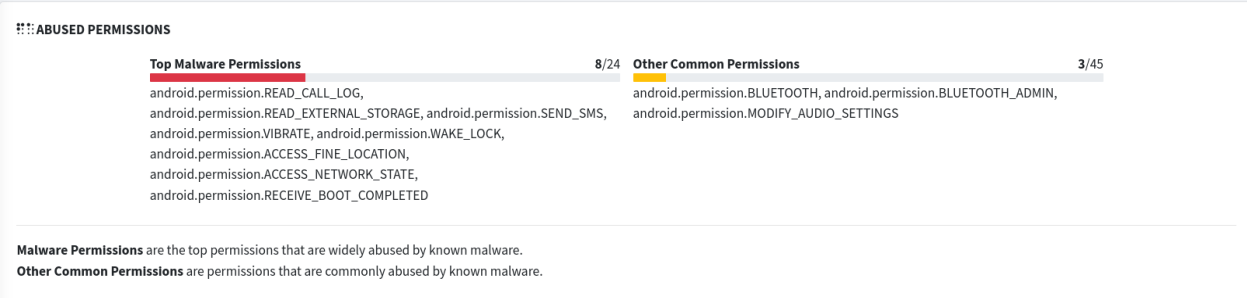


Figure 52: app permissions for the phone calls app.

App name	Shared permission
Contacts app	Call log
Messages app	Call log
Phone calls app	Call log

Call logs can store all information related to telephone call to contacts and messages. Normally, a user can have the ability to check the recent calls or even instant messages with another user in the contacts through to use of a call log that registers it a history. A call log is a database which is managed by the operating system of the mobile phone. As observed, most of the pre-installed system apps have improper handling of executing raw SQL queries. Given that the three mentions apps (figure 50, 51, 52) have access to the call log, the potential risk of the call log being breached is high and could expose information such as recent calls, recent messages and contact information.

7. Ethical Consideration:

All data collected for this research study have been collected from a Samsung A12 device that is owned by the conductor of this research study. Information exposure has been a crucial part of this study to not expose any sensitive information. This research study did not collect any information from any participant through surveys or questionnaires. The data collected for this study was data collected from a mobile application testing tool MobSF (Mobile security framework) by using applications extracted from the Samsung A12 device.

No attempts were made to reverse engineer the app or expose information about other users in the Samsung phone. Vulnerability testing on the extracted app was done legally since no information about any user were going to be involved in the study. The vulnerability assessment done on mobile applications was done in a safe manner due to vulnerability assessment tool used which tests a mobile application by using its APK file. Private apps such as third-party apps that were tested in this study were only included for comparative arguments in the analysis part to set a contrast between system apps and third-party apps. Furthermore, no negative arguments were made to any third-party apps and the information detailed in the study was only broad on the surface to not expose any flaws of the mobile application.

The mobile applications owners of the third-party apps were sent an email request for consent but due to the high number of emails they received, it is advised that a response would not be received in time for companies such as Instagram and snapchat [15]. This study has taken considerable measures to not mishandle in type sensitive information from the apps extracted from the Samsung device and only focused on information that was necessary for the success of this research study.

8. Evaluation Plan

8.1 Effectiveness of mitigations measures:

This part of the study, this study shall identify the best strategies to mitigate the vulnerabilities identified in this research study and apply the mitigation measures. The approach shall measure the effectiveness of the measures applied by re-testing some of apps tested in Mobile security framework and determine the effectiveness of the measures applied.

8.2 Identification of mitigation methods:

One of the best practices or industry standards will be regular updates provided by operating systems manufacturers. Currently, the Samsung A12 runs an android 12 version.

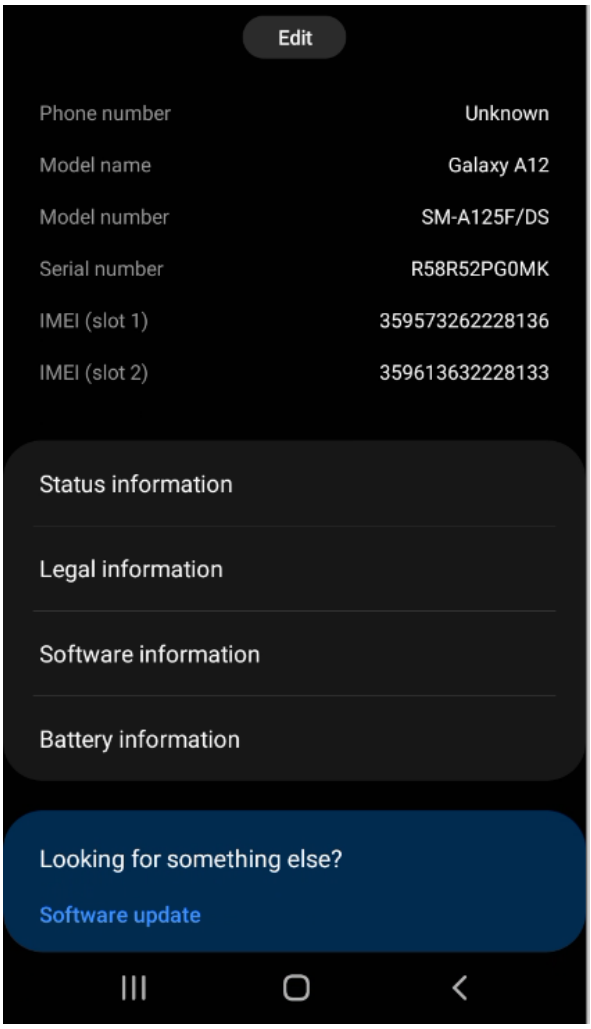


Figure 53:phone details.

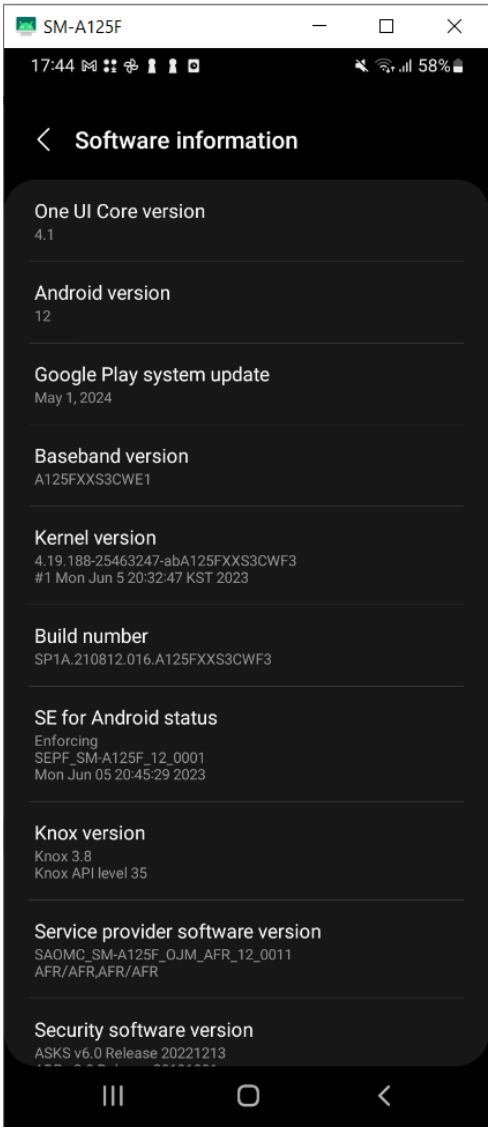


Figure 54:Samsung A12 software information.

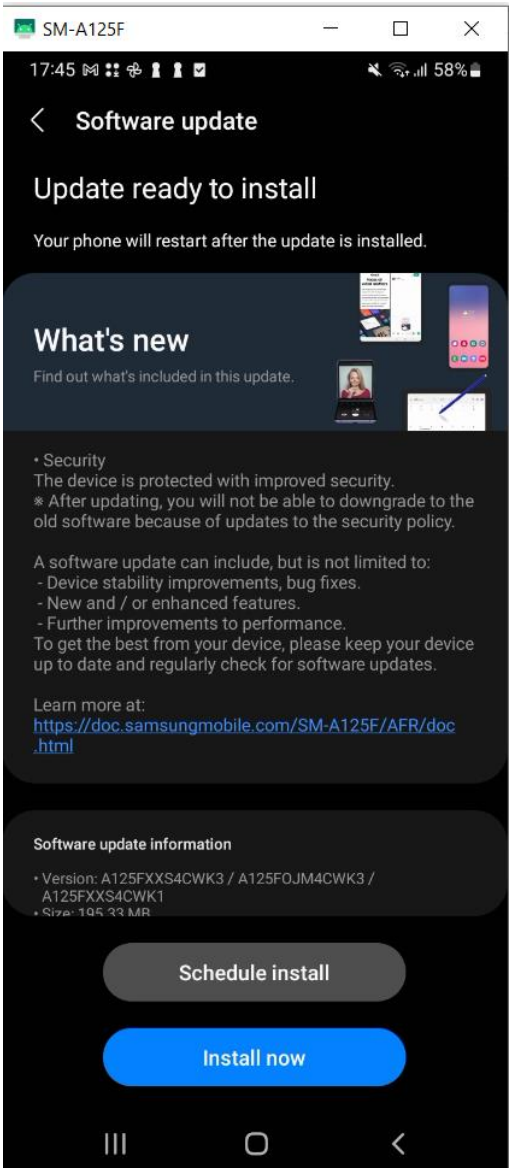


Figure 55:software update for the Samsung A12.

As observed in figure 55, the mobile device has some pending software system updates from Google. As a first step, it is best to update the mobile device since software updates are based on vulnerabilities identified and the system manufacturer provides updates and patches to fix vulnerabilities that can potentially be exploited by attackers.

8.3 Implementation of mitigation methods:

The procedure is to download the latest updates and patches and install the downloaded packages. After a successful system update for the android device, re-testing the system apps that were part of the selection of the vulnerability assessment to check whether the user input vulnerabilities are still present or not.

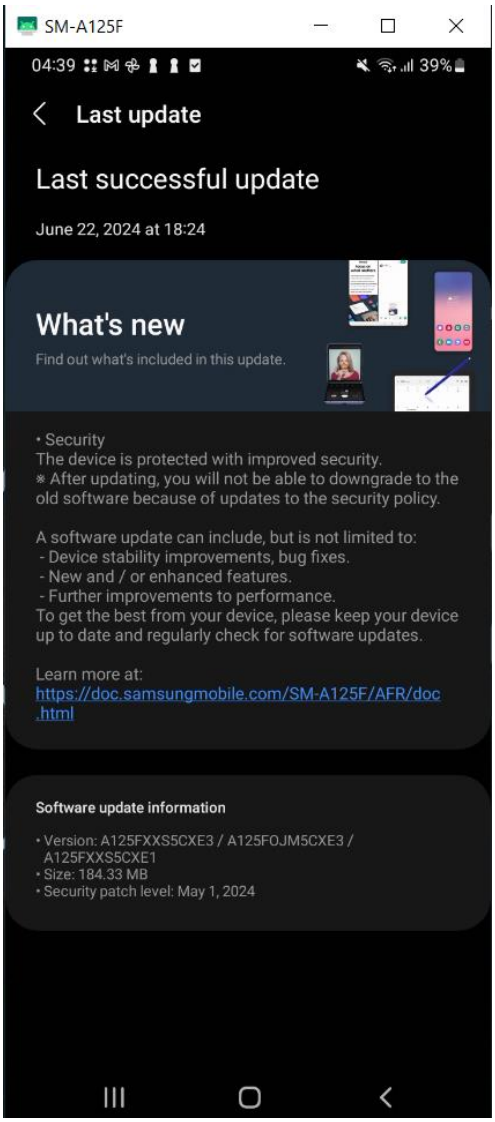


Figure 56: latest system updates successful.

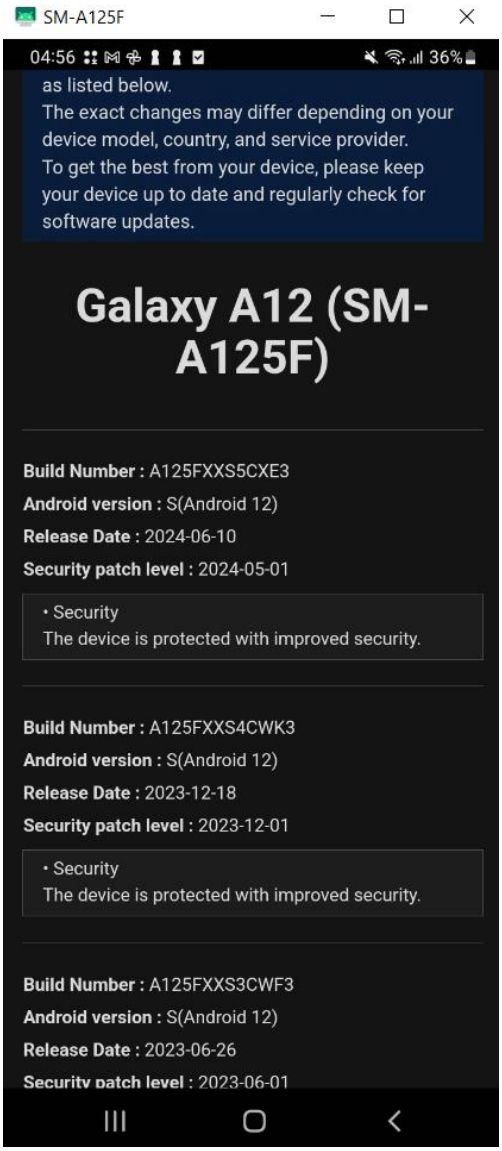


Figure 57:system update information.

The system update was successful. However, this android device was expected to not receive the latest version of android after its system update due to the model of the device. According to the system update, the update provides enhance security and testing the enhanced security would be necessary to test the effectiveness of this method which is primarily essential as a first step to take to clear out vulnerabilities embedded in the mobile device.

8.4 Re-test for vulnerabilities:

This part shall re-assess some of the vulnerabilities identified and check whether they are still present in some of the system apps which are primarily concerned than third party apps that get updates from the manufacturers. This study shall select a few of the system apps that were previously selected in search for vulnerability assessment. The common vulnerability of improper mishandling of user inputs was the execution of raw SQL queries that could potentially be exploited by attacker by SQL injection in the SQL database. the reason to select a few system apps is because system updates packages are applied to all system apps. Testing one or two apps should confirm the effectiveness of the system updates and patches.

8.5 Selection of system apps:

- Bluetooth
- Contacts
- Gallery

8.6 Re-testing of system apps:

This study shall re-extract the newly updated system apps using an APK extractor.

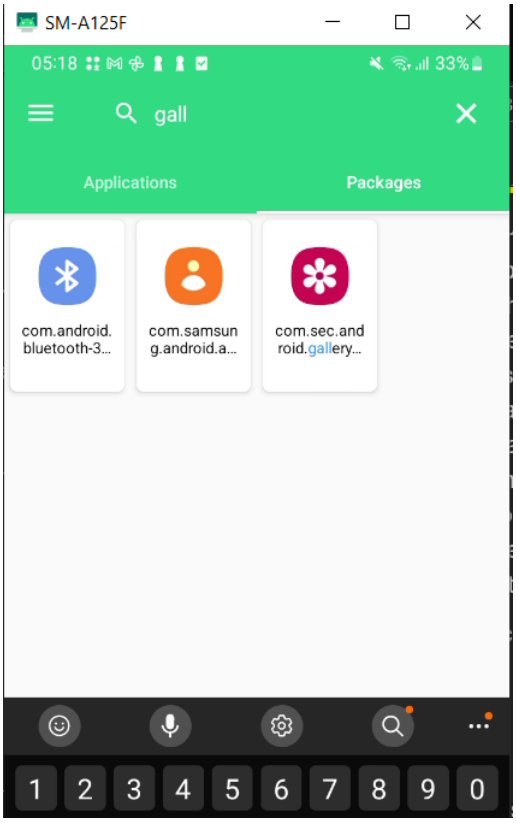


Figure 58:APK extraction.

Apk scans:

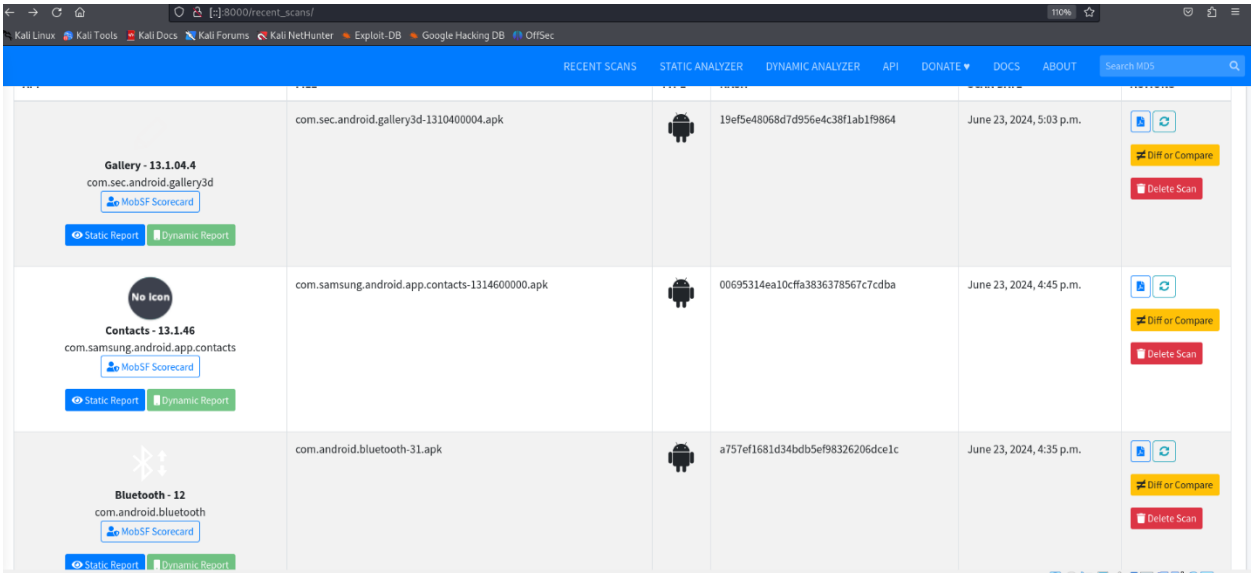


Figure 59: new APK file scans after system update.

8.7 Findings:

- App name: gallery.
  Category: system app.

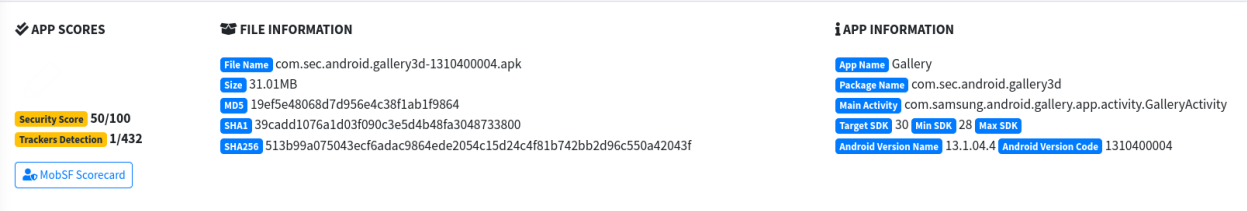


Figure 60: Gallery app.

Vulnerabilities found:

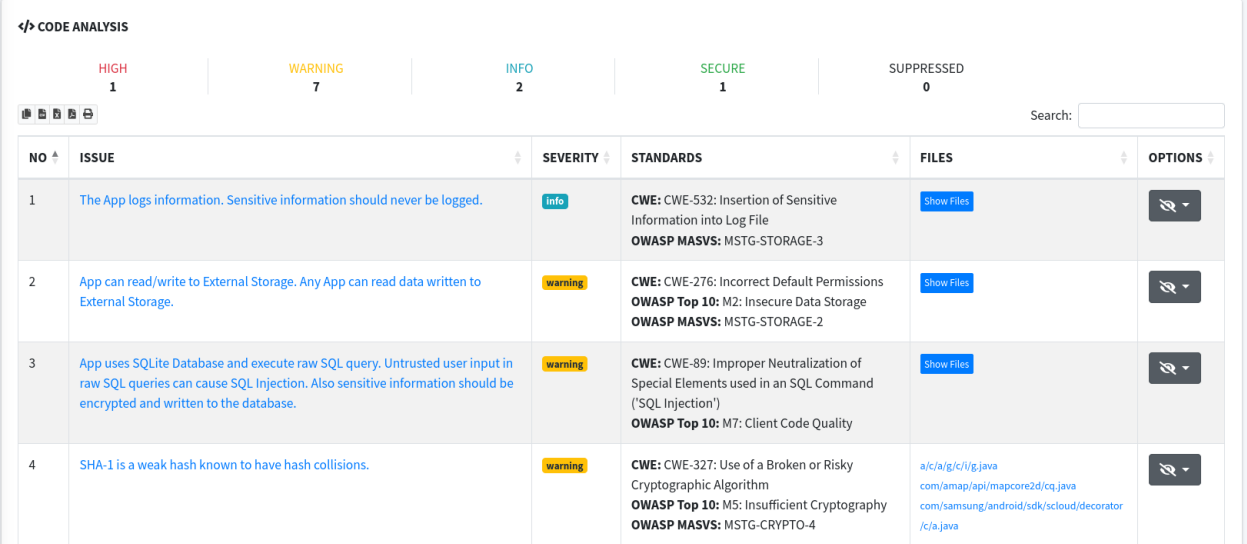


Figure 61:codebase vulnerabilities in the Gallery app.

Based upon the new scan of the gallery app after the system update, the execution of raw SQL queries vulnerability is still present in the app as shown in figure 61.

- App name: contacts.
  Category: system app.

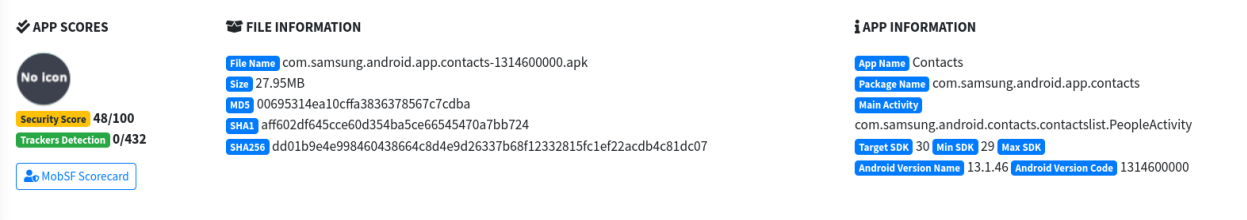


Figure 62: contacts app.

Vulnerabilities found:

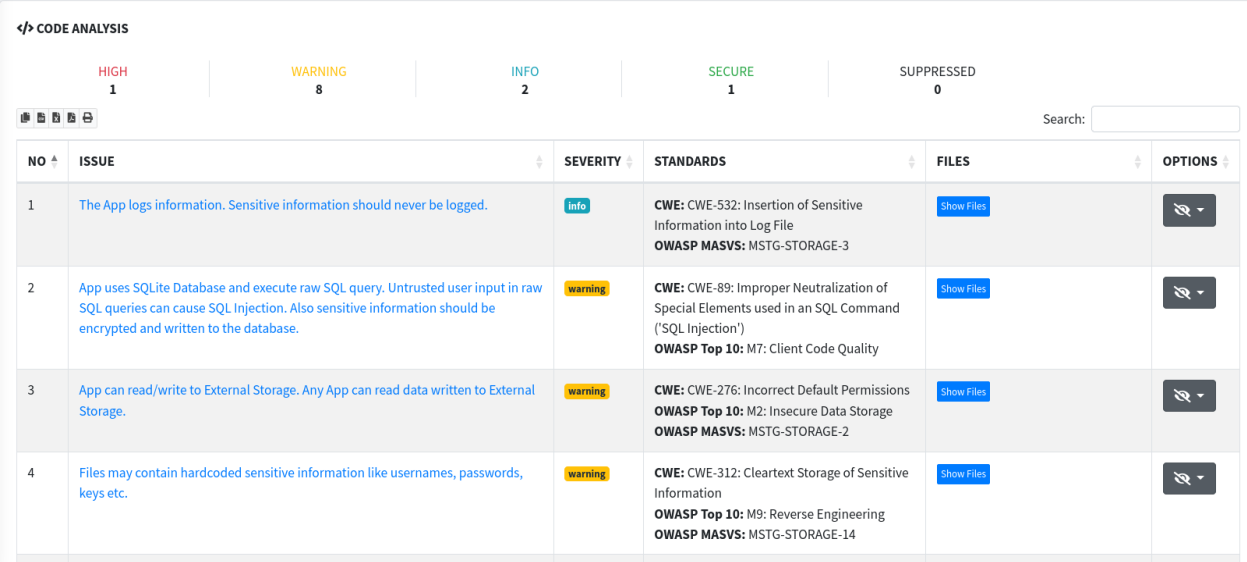


Figure 63: codebase security flaws in the contacts app.

the execution of raw SQL queries is still a present vulnerability in the contacts app as shown in figure 63.

- App name: Bluetooth.
  Category: system app.

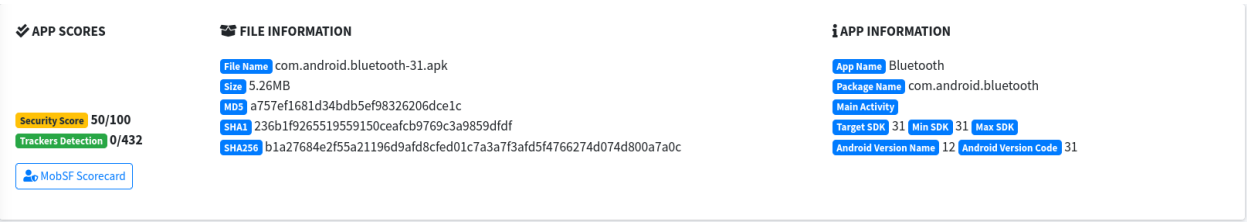
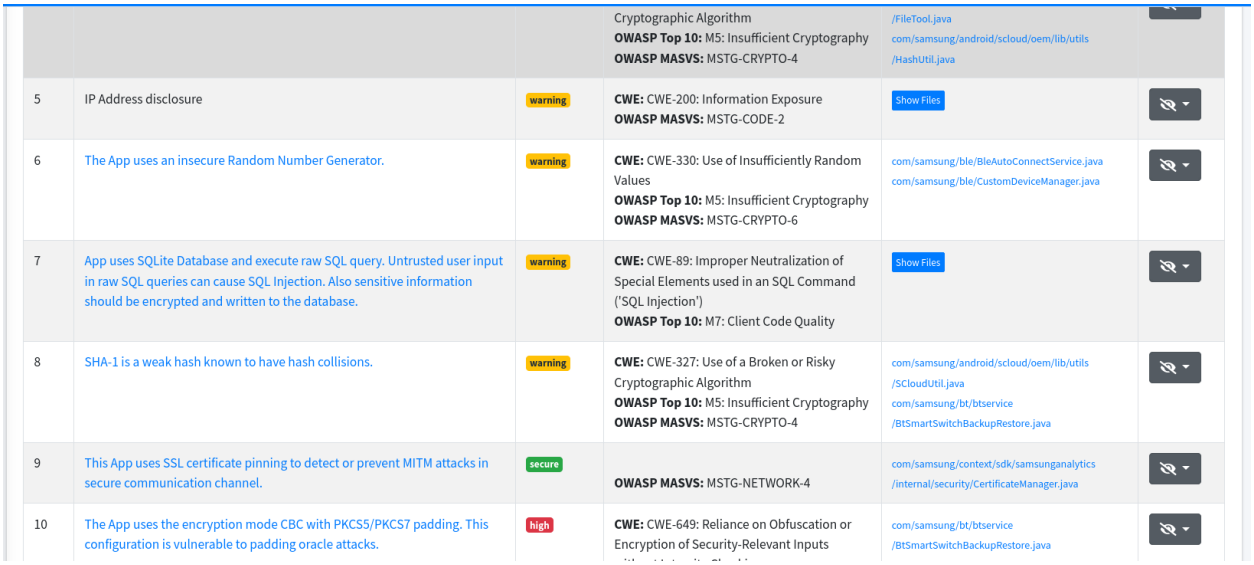


Figure 64: Bluetooth app.

Vulnerabilities found:



			Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	/FileTool.java com/samsung/android/scloud/oem/lib/utils /HashUtil.java	
5	IP Address disclosure	warning	**CWE:** CWE-200: Information Exposure **OWASP MASVS:** MSTG-CODE-2	Show Files	
6	The App uses an insecure Random Number Generator.	warning	**CWE:** CWE-330: Use of Insufficiently Random Values **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-6	com/samsung/ble/BleAutoConnectService.java com/samsung/ble/CustomDeviceManager.java	
7	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	warning	**CWE:** CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') **OWASP Top 10:** M7: Client Code Quality	Show Files	
8	SHA-1 is a weak hash known to have hash collisions.	warning	**CWE:** CWE-327: Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	com/samsung/android/scloud/oem/lib/utils /SCloudUtil.java com/samsung/bt/btservice /BtSmartSwitchBackupRestore.java	
9	This App uses SSL certificate pinning to detect or prevent MITM attacks in secure communication channel.	secure	**OWASP MASVS:** MSTG-NETWORK-4	com/samsung/context/sdk/samsunganalytics /internal/security/CertificateManager.java	
10	The App uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is vulnerable to padding oracle attacks.	high	**CWE:** CWE-649: Reliance on Obfuscation or Encryption of Security-Relevant Inputs	com/samsung/bt/btservice /BtSmartSwitchBackupRestore.java	

Figure 65: security issues in codebase of the Bluetooth app.

Execution of raw SQL queries vulnerability is still present in the Bluetooth app as shown in figure 65.

8.8 Measure of effectiveness:

Even though the system update may have improved other areas such as phone performance and improved security of the phone but not all issues were addressed by the system update. The system update may not have included system updates fixes for specific vulnerabilities which continue to be a security issue and an advantage for threat actors that may see potential in exploiting vulnerabilities that are not addressed. A phone such as the Samsung A12 tend to be not a priority unlike the latest Samsung phones which have the latest android version such as the android 14 version. As reported, the Samsung A12 receives security patches updates twice a year

and still may not receive the latest updates but only updates available depending on the model of the phone [16]. The effectiveness of the system update did not reach the success point which should have addressed the vulnerabilities in the system apps.

Raising awareness on such issues is key to avoiding any data breaches. System updates may not address the vulnerabilities in the smartphone depending on the model and give users the chance to know the security strength structure of their mobile device. However, raising awareness may not be as effective as system updates which should be the primary key to flash out any type of vulnerability. Users tend to delay system updates due to the amount of time it takes for the phone to fully update, and the user is impatient. This study encourages users of Samsung devices to download and install the system updates every chance they get or set the phone to automatically download and install the system updates which is an option already available in smartphones.

9. Recommendation

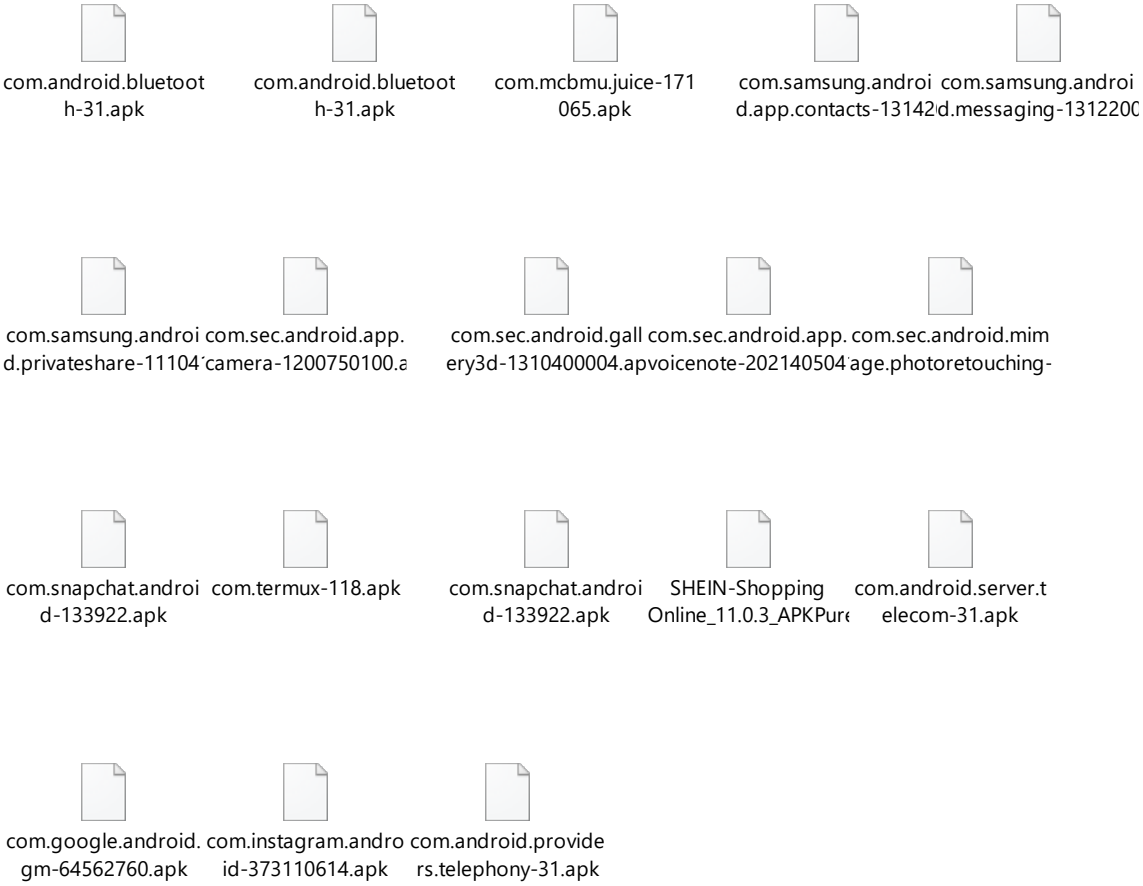
Old Samsung models such as the Samsung A12 model that had it's initial release in the year of 2020 or low budget Samsung models that do not have the same specifications as advanced models such as the Galaxy note, are often left out with insufficient system updates and patches that do not address all vulnerabilities. This study aims to address developers of Samsung mobile application developers to prioritize all Samsung models by increasing or producing even system updates and patches to all Samsung models. This study findings, especially on the findings after re-testing system apps that have been received system updates and patches has shown that some apps are ignored, and performance and other new security features are prioritize. It is the unaddressed vulnerabilities that remain as a critical issue and an open opportunity to hackers that can exploit such vulnerabilities. Data security must always be a priority regardless of the model or cost of the phone. Developers should address the vulnerabilities exposed in this study and produce sanitized line of codes that can better secure SQL databases that may hold sensitive data.

10. Limitations

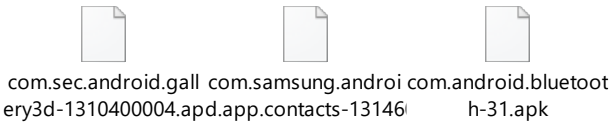
This research study was not able to test the APK files in real-time to evaluate how information is shared and how secure it is when information is transferred from one point to another. The MobSF vulnerability assessment has a dynamic analyzer that requires an emulator that acts as a mobile device and can be used to test different types of apps. The emulator was called genymotion which is an android emulator since this study could not use the Samsung A12 device in real-time and capture network traffic using MobSF [17]. The setup of genymotion had various issues, including lack of proper information on how to run genymotion and MobSF at the same time. The aim was to test the user-experience and how secure is data over the traffic.

11. Appendix

11.1 APK files



11.2 New APK file after system update:



12. References:

[1] "Government has 'warning' for Galaxy S23, other Samsung smartphone users," *The Times of India*, Dec. 16, 2023. Accessed: Jun. 30, 2024. [Online]. Available: https://timesofindia.indiatimes.com/gadgets-news/government-has-warning-for-galaxy-s23-other-samsung-smartphone-users/articleshow/105997045.cms

[2] "OWASP Mobile Application Security | OWASP Foundation." Accessed: Jun. 30, 2024. [Online]. Available: https://owasp.org/www-project-mobile-app-security/

[3] "M7: Poor Code Quality | OWASP Foundation." Accessed: Jun. 30, 2024. [Online]. Available: https://owasp.org/www-project-mobile-top-10/2016-risks/m7-client-code-quality.html

[4] G. P, "What is Legacy Code and How to Deal with It [Updated 2021]." Accessed: Jun. 30, 2024. [Online]. Available: https://enkonix.com/blog/legacy-code

[5] M. Chinta, "SQL Injection: Understanding the Risks and Protecting Your Code," CodeNx. Accessed: Jun. 30, 2024. [Online]. Available: https://medium.com/codenx/sql-injection-understanding-the-risks-and-protecting-your-code-dd09aa042f90

[6] markingmyname, "sp_executesql (Transact-SQL) - SQL Server," May 2024, Accessed: Jun. 30, 2024. [Online]. Available: https://learn.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-executesql-transact-sql?view=sql-server-ver16

[7] "More than 5 billion Bluetooth devices by 2021." Accessed: Jun. 30, 2024. [Online]. Available: https://blog.softtek.com/en/more-than-5-billion-bluetooth-devices-by-2021

[8] "New Bluetooth Flaw Let Hackers Take Over Android, Linux, macOS, and iOS Devices," The Hacker News. Accessed: Jun. 30, 2024. [Online]. Available: https://thehackernews.com/2023/12/new-bluetooth-flaw-let-hackers-take.html

[9] "Instagram - Apps on Google Play." Accessed: Jun. 30, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.instagram.android&hl=en_US

[10] "Temple Run - Apps on Google Play." Accessed: Jun. 30, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.imangi.templerun&hl=en_US

[11] "Use Google Play Protect to help keep your apps safe & your data private - Google Play Help." Accessed: Jun. 30, 2024. [Online]. Available: https://support.google.com/googleplay/answer/2812853?hl=en

[12] "Royal Match - Apps on Google Play." Accessed: Jun. 30, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.dreamgames.royalmatch&hl=en_US

[13]

"CVE-2021-25427 : SQL injection vulnerability in Bluetooth prior to SMR July-2021 Release 1 allows unauthorized access to paired device in." Accessed: Jun. 30, 2024. [Online]. Available: https://www.cvedetails.com/cve/CVE-2021-25427/

[14] "What is CVE and CVSS | Vulnerability Scoring Explained | Imperva," Learning Center. Accessed: Jun. 30, 2024. [Online]. Available: https://www.imperva.com/learn/application-security/cve-cvss-vulnerability/

[15] A. A. D. Antonelli William, "How to contact Instagram support for help with your account," Business Insider. Accessed: Jun. 30, 2024. [Online]. Available: https://www.businessinsider.com/guides/tech/how-to-contact-instagram

[16] PURU, "Samsung distributes April 2024 security patch to Galaxy A12 Nacho." Accessed: Jun. 30, 2024. [Online]. Available: https://samlover.com/2024/04/22/samsung-distributes-april-2024-security-patch-to-galaxy-a12-nacho/

[17] "Genymotion - Android Emulator in the Cloud and for PC & Mac," Genymotion. Accessed: Jun. 30, 2024. [Online]. Available: https://www.genymotion.com/