An Analysis of Sensitive User Input and Permission Requests in Android Applications

Kaan Akduman
kxa317@case.edu
Department of Computer and Data Sciences
Case Western Reserve University
Cleveland, Ohio

Andrew Duffield
ajd173@case.edu
Department of Computer and Data Sciences
Case Western Reserve University
Cleveland, Ohio

Yadira Gonzalez
ycg@case.edu
Department of Computer and Data Sciences
Case Western Reserve University
Cleveland, Ohio

Sean Twomey
spt22@case.edu
Department of Computer and Data Sciences
Case Western Reserve University
Cleveland, Ohio

ABSTRACT

In recent years, there has been an increase in the need for the protection of sensitive user information in mobile applications. UI analysis is a way for developers to analyze the issue surrounding the sensitive information in their applications and aids in formulating ways to mitigate those issues. Sensitive user inputs have been largely neglected by permissions and, as a result, may pose a higher risk to the application than other components. Moreover, permission analysis is another helpful approach in determining which permissions, if any, are posing security threats to an application. In this paper, we explore the possible relationship between sensitive user inputs and permission requests through UI and permission analysis.

1 INTRODUCTION

As time goes on smartphones continue to evolve, not only in complexity and functionality but also in the degree to which they are involved in our lives. We use our mobile devices and their applications for such a wide array of purposes whether it be communicating with friends and family, playing games, surfing the internet, buying our favorite products, even using them to secure the locks on our doors through smart home technology. With the increased presence of smartphones in so many aspects of user's lives, privacy and security have naturally become a focal point of concern. Is our data protected and not accessible to outside parties when we use our many applications? This focus on privacy has led to a plethora of research studies and efforts to improve Android application security, however, these efforts have not been entirely focused on every point of danger within our apps.

A crucial element of Android applications which research efforts have primarily neglected thus far is sensitive user inputs. Sensitive user inputs are simply defined as any sensitive information entered by a user into an application's user interface. We may not always realize it, but we often willingly input a great deal of personal information into our devices, at times without really thinking about it. Examples of types of sensitive user inputs we often may lend to our devices include personal identifying information, health information, and sensitive financial information.

Personal identifying information could include sensitive information such as a home address or a social security number, the former used very frequently when doing any kind of online shopping and the latter used for verification of identity quite often. Health information is often private in nature, and inputting a condition or subscription into a mobile pharmaceutical app for example could pose a risk. Sensitive financial information is often at risk by way of providing credit card and bank account information through our mobile banking and investing apps.

The underlying issue at hand is not the sensitive user inputs themselves, of course entering sensitive information is regularly necessary for various everyday functions and to get the most out of Android applications. A primary impetus for concern around sensitive user inputs comes from the methods in which the data is stored or transported. In the SUPOR study, the authors reference how a simple login request in an app may transmit the user ID and password through an insecure HTTP channel in a plain text format, which ultimately compromises the user's privacy [Sean1]. Users and even developers of an application containing sensitive user inputs may be unaware of insecure means being used for transmission of user data, ultimately putting the user's data privacy at risk which is an issue worth rectifying.

Another intertwined issue we wish to consider in this study is Android permissions and how they can possibly correlate with the risk posed by sensitive user inputs within applications operating with insecure means. The authors of the SUPOR paper also mention that most sensitive data managed by the smartphone and its associated APIs are permission protected. This sensitive data includes but is not limited to device identifiers, location, contacts, browser state, and more [Sean1].

Through our work in this course dealing with permission analysis, we are aware that applications request many permissions from the users and not always in an entirely safe manner. Some permissions requested are not necessary for the app to function, some after being granted will download malware unbeknownst to the user, and others operate in dangerous manners to the user. Users are not always completely cognizant of what each permission requested by an application does, and thus will often grant said requests without keeping their security in mind. A possible connection can be drawn between sensitive user inputs within an app and permissions requested by the app, are sensitive user inputs placed at higher risk and jeopardize the user's privacy to a higher degree when certain permissions are granted to the application by the user? We aim to further investigate the relationship between sensitive user inputs and permissions requested within Android applications. This analysis will serve as an ultimate effort to better understand how to increase privacy amongst Android apps with sensitive user input components.

2 BACKGROUND

Prior to discussing the steps we took to find a correlation between sensitive user inputs and android permissions, this section will introduce Android permissions and UI analysis. These two components were the main focus of our approach. A more detailed description of our approach will be given in section 3.

2.1 Android Permissions

In order to manage the data resources of an application, the Android system adopts its own Android permissions framework. Android permissions dictate if resources such as camera, GPS location, and SMS data, can be accessed by the application. The type of Android permissions within an application can be broken down as

follows: install-time permissions, runtime permissions, and special permissions [Yadira 1].

Install-time permissions are those permissions that are granted when the user downloads the application; the application's system will automatically grant the application with these permissions. Moreover, two subtypes of install-time permissions include normal and signature. Normal permissions generally present a low risk to a user's data; these are permissions that allow the application to access data and actions that it cannot get within the scope of the application. Signature permissions, on the other hand, are permissions that require application signatures from two applications; if both applications do not have the signature then the permission cannot be enabled. Each of these types of permissions are assigned "normal" and "signature" level protection, respectively [Yadira 1].

The second type of Android permissions are runtime permissions, also known as dangerous permissions. Runtime permissions are particularly important because they give applications access to restricted data and allow applications to perform restricted actions. As a result, runtime permissions can only be enabled by the user. In short, an application must declare permissions in its manifest file. If the permission is a runtime permission, then a permission request will be given to the user at runtime. The user can then decide whether to accept the permission request or not [Yadira 1].

The last type of Android permission are special permissions. Special permissions dictate specific application operations and are therefore defined by the platform and OEMs. Ultimately, special permissions are put into place in order to protect dangerous actions such as drawing over other applications [Yadira 1].

Moreover, the Android permission system was built on the idea of protecting user privacy. As a result, it follows the following components: control, transparency, and data minimization. In regards to control, the user is given autonomy over which permissions it allows. By having the user explicitly give permission to what data can be accessed (through runtime permissions), the user is given control over their personal data. In regards to transparency, applications must provide the user with what data the application will use and why that data is necessary for the application. By providing this kind of transparency on what is done by the application and why those specific permissions are needed, the user will be able to make more educated choices about the permissions they allow an application to use. Lastly, in regards to data minimization, an application should only use data that is required for the application to function or was allowed by the user. Minimizing the data usage of an application aids in the integrity of the application as well as minimizing data leakage [Yadira 1].

Understanding the Android permissions system was important to our team so we can understand the framework through which potential data leakage can occur. As stated in the previous section, the current framework still allows for possible security risks as users may not be fully cognizant of the permissions they are allowing an application to use. Moreover, our project primarily focuses on sensitive user input, which has been largely neglected by permissions. Understanding the current framework behind Android permissions has given our team a clearer understanding of why that could possibly be. When inserting sensitive user information into an application's user interface, the Android permissions system does not require the application to send a permission request. As a result, these sensitive user inputs are at high risk.

2.2 UI Analysis

The primary way through which users interact with an application is through the application's user interface (UI). An application's UI design is particularly important to our project because sensitive user inputs are inserted through an application's UI. In order to understand an application's UI, an in-depth analysis of the UI must be performed.

UI analysis is important in that it helps outline if the intentions delineated by the application's interface are correct; it essentially helps determine if the application is using data in the way that the application says it will. Generally, UI analysis is broken down into the following steps: obtaining UI elements, modeling UI control flow, obtaining UI element contents, and analysis of the previous three steps. For the purpose of this project, our team primarily focused on the obtainment of UI elements and the analysis of those UI elements.

Within android applications, UI elements are generally defined in an application's layout files. In order to obtain the elements of an application, an APK of the application must be used. The APK file is essentially an Android zip file that contains bytecode and resources, specifically any and all layout files that contain the declared UI elements of the application. In order to unzip these files, an APK tool can be used. An APK tool is used to reverse engineer Android APK files; it decodes resources to their original form. Once an APK file has been disassembled by an APK tool, the UI elements may be extracted from the layout files. The UI elements that are extracted can then be used for further analysis. For instance, UI elements can carry tags relating to the type of information that is obtained from those elements. These tags allow us to determine if the information can be deemed sensitive or not [Yadira 2].

Obtaining these UI elements was important to our project because it helped our team obtain the UI elements that are relevant to sensitive user information. Extracting the UI elements necessary to obtaining sensitive user inputs could potentially allow us to draw a correlation between permissions used within an application and the intentions outlined by the application's interface. The following section will outline the steps our team took to implement an approach that considers both sensitive user inputs and android permissions.

3 APPROACH

Our approach for this study was three-fold. It first consisted of sensitive user input retrieval, then respective permissions requested retrieval, and concluded with the in-depth analysis of the previous data collected. We will describe each of these procedures in greater detail as well as the various tools and practices used.

3.1 Sensitive User Input Retrieval

Our study began by using a modified version of SUPOR in combination with two python programs. We selected our random subset of 250 Android APKs out of the provided 1500 distributed through this course using the script runner.py. Runner.py simply took the overarching set of APKs and returning a random subset of them using Python's pseudo-random functionalities.

We then wished to retrieve various sensitive user inputs which were used in these APKs by leveraging SUPOR. To do this we utilized another script, runs.sh. This script would take the randomized subset of APKs generated by runner.py as input, and then return individual outputs revealing sensitive user inputs evident in each APK. Each output was in the form of an individual CSV file. These results were placed in the file supor_ouput. We now had access to an array of sensitive user inputs in each APK and

could move on to our second step where we would hone in on permissions requested in these APKs.

3.2 Permissions Requested Retrieval

The second step in our approach was to gather the permissions requested in each APK from our subset. This operation was performed through a modified version of the DumpApkPackname.java file we used in lab 2. We modified the file and then ran the main method which returned us all the permission requests for the specific APK. The results across the APKs were collated into the file app_permissions_final.csv. At this point, we had successfully obtained the sensitive user inputs and permissions requested from each APK in our subset, and we were ready to then analyze the data we had retrieved and find a correlation.

3.3 Sensitive User Input and Permissions Requested Analysis

To effectively analyze our data and produce corresponding graphs of said data, we leveraged two more python scripts. Our analysis began by placing our part 1 file supor_ouput and our part 2 file app_permissions_final.csv into the same directory. A script csv_creator.py was run on the individual CSV files within suport_ouput from our first step dealing with sensitive user inputs, to output the final CSV detailing permissions and sensitive user inputs with appropriate statistics.

The final action we took in our procedure was to run another script titled data_analyser.py to produce our graph which would display the number of sensitive user inputs vs the number of permission requests across all APKs in our subset. This graph would also include a line of best fit, to aid in revealing any potential correlation between these two variables. The results of our study are displayed in the following section.

4 RESULTS AND ANALYSIS

In this section, we outline the results retrieved from our approach in section 3. It is important to note that many of the results that we gathered from our approach do not align with the hypotheses we developed at the start of our project. We will go into further detail as to why that was and explain the results that we did obtain.

4.1 Correlation Analysis

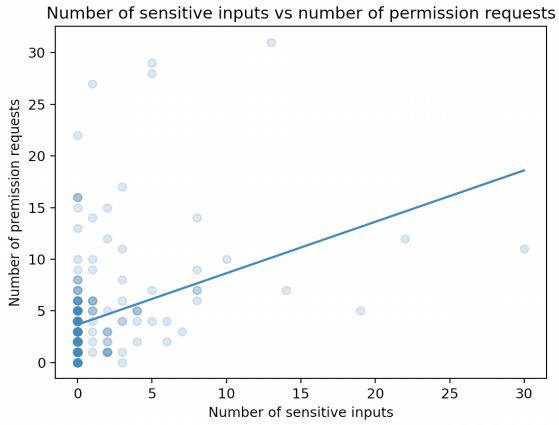

Figure 1

Figure 1 shows the correlation between the number of permissions requested and the number of sensitive inputs in an apk. Each point on the graph represents one or more apks from our sample of 250. The darker points represent multiple apks. We plotted a best fit line to the data and found the graph to have an r-value of 0.35 which means that the number of permissions requested increases with an increase in the number of sensitive inputs. Although this isn't a strong correlation, it is concerning nonetheless because apps that have many sensitive user inputs and also request many permissions, or specifically dangerous permissions, pose a security and privacy risk to the user.

Essentially, the danger lies in the fact that dangerous permission requests being granted by an app that contains sensitive user inputs open the door for the user's sensitive information to be placed at risk of being compromised. For example, there exists a multitude of attacks that can be brought upon a user through an application and its corresponding permissions. A study conducted by two researchers through the Department of Computer Science at Christ University outlined several attacks users can be subjected to through apps and associated permissions. They discuss threats posed to users such as permission escalation attack, collision attack, time of check, and time of use attack.

A permission escalation attack is when a malicious application collaborates with other applications to access critical resources without requesting the explicit permissions associated with the resources [Sean2]. A collision attack is described as "If application A has permissions to READ_CONTACTS, READ_PHONE_STATUS and B has permissions to READ_MESSAGES, LOCATION_ACCESS, if both the applications use the same user id SHAREDUSERID, then it is possible for application A to use the permissions granted to itself and the permissions granted to B." [Sean2]. A time of check and time of use attack (TOCTOU) is derived from a naming collision. Permissions are stored as strings, and thus any permission with the same name is treated as equivalent even if they belong to different applications [Sean2].

The important takeaway for the correlation is despite it being on the lower side, it still should yield concern amongst users as applications that take in their sensitive information tend to also request a larger number of permissions which puts the aforementioned sensitive information at risk for security breaches and various attacks. The permissions discussed in the next section will shed some light on which permissions may be more frequently requested across the APKs we analyzed which contain sensitive user inputs.

4.2 Permission Analysis in Sensitive Apps

As stated in our approach section, our team was interested in seeing the correlation, if any, between permissions requested and sensitive user inputs in an application. Although our correlation analysis showed a weak connection between the two, we obtained additional information regarding the types of permissions requested by each application in our dataset. We then attempted to relate permission requests to sensitive user inputs in each application. Our findings are outlined in Figure 2.

Permission	Total (%)	Sensitive (%)	Difference (%)
ACCESS_NETWORK_STATE	40.78	66.1	25.32
READ_PHONE_STATE	38.43	62.71	24.28
READ_CONTACTS	7.84	23.73	15.89
VIBRATE	25.49	40.68	15.19
CALL_PHONE	6.27	18.64	12.37
WRITE_CONTACTS	4.71	16.95	12.24
WAKE_LOCK	19.61	30.51	10.9
INTERNET	87.45	98.31	10.86
WRITE_EXTERNAL_STORAGE	28.24	38.98	10.74
RECEIVE_BOOT_COMPLETED	5.88	15.25	9.37
ACCESS_FINE_LOCATION	12.94	22.03	9.09
RECORD_AUDIO	4.71	13.56	8.85
CAMERA	3.14	11.86	8.72
RECEIVE_SMS	2.75	10.17	7.42
SEND_SMS	2.75	10.17	7.42
MODIFY_AUDIO_SETTINGS	4.71	11.86	7.15
GET_TASKS	6.67	13.56	6.89

WRITE_SETTINGS	4.31	10.17	5.86
READ_LOGS	4.31	10.17	5.86
RESTART_PACKAGES	2.75	8.47	5.72
PROCESS_OUTGOING_CALLS	2.75	8.47	5.72
BROADCAST_STICKY	1.57	6.78	5.21

Figure 2

Figure 2 shows the breakdown of permission requests with respect to the percentage of applications used. The first column of the table shows the percentage of the total apps used that requested that permission. The second column of the table shows the percentage of applications with sensitive user inputs that requested that permission. The third column of the table shows the difference between the first and second columns; it shows the likelihood of an application that has sensitive user inputs to request that permission. For instance, row one of the table shows that 40.78% of our dataset requested ACCESS_NETWORK_STATE, 66.1% of the applications of our dataset that contained sensitive user inputs requested ACCESS_NETWORK_STATE, and the applications with sensitive user input within our dataset were 25.32% more likely to request ACCESS_NETWORK_STATE. The rest of this section will consist of an analysis of the results obtained from Figure 2.

From this analysis, we found that applications that included sensitive user input were up to 25% more likely to ask for certain permissions. As shown in Figure 2, the highest difference was ACCESS_NETWORK_STATE, where an application with sensitive user input was 25.32% more likely than a random sample of applications to request that permission. This may suggest that an application that has sensitive user inputs is more likely to send data over the internet than a random application. This could pose a potential security risk as sensitive user inputs are more likely to be leaked if that data is sent across a network, even if the application's intention is not to leak that information. Factors such as insecure connections, leaked private keys and database hacks can lead to compromised sensitive user data. However, given that the majority of applications need ACCESS_NETWORK_STATE and INTERNET in order to function properly, we cannot deduce that accessing these permissions is likely to lead to leakage of sensitive user information. It is clear from these findings that we need to do more research regarding permissions in correlation to sensitive user inputs to determine if it is indeed a risk.

In addition to ACCESS_NETWORK_STATE, we found three more dangerous permissions which were more likely to be requested by an app that contains sensitive user inputs. These permissions are ACCESS_FINE_LOCATION, RECORD_AUDIO, and CAMERA which were 9.09%, 8.85%, and 8.72% more likely to be requested by apps containing sensitive inputs, respectively. The presence of any of these permissions in apps with sensitive inputs is concerning because sensitive data can be even more dangerous if connected to the user's location, recordings of their conversations, and photos of their surroundings. Seeing that these permissions were more likely to be requested in apps with sensitive inputs is especially concerning because it could signify that these apps are using the data maliciously. However, as stated earlier, the data we collected cannot definitively determine if the use of these permissions affects sensitive user input. More research would need to be conducted to determine how and when these permissions are being used in relation to sensitive user inputs.

Moreover, it is also important to discuss how the data that we collected from the table corresponds

to our correlation analysis from the previous section. The correlation between the number of sensitive user inputs and the number of permissions requested is weak, with an r-value of 0.35 as previously stated. We were not expecting this result, because we had hypothesized that applications that contain a larger number of sensitive user inputs would also request a larger number of permissions. This hypothesis was driven by the notion that an application that needs to take in and transmit sensitive information will likely need permission to access the internet, network, read or write to the phone, and other potentially dangerous permissions.

We believe the lower correlation may be partially due to the fact that it is a difficult task to differentiate between an application containing sensitive user inputs requesting dangerous permissions for malicious reasons and an app requesting those permissions simply because those permissions are necessary for the app and to facilitate the storage or transmission of the sensitive inputs. For example, just because an application requests INTERNET and ACCESS_NETWORK_STATE when submitting the user's username and password for login does not mean those login credentials are going to be sent off to unapproved third parties. Those permissions are requested simply because the application needs to verify the user's credentials and perhaps interface with a database that will require internet and network access.

The lower correlation still reveals a correlation, but it does not exactly state that applications containing sensitive user inputs will automatically request the most dangerous permissions and then use them to exploit the user's privacy. These applications of course could potentially be misusing dangerous permissions, but there is a grey area there alongside the apps requesting necessary permissions to facilitate the usage of the sensitive inputs. This distinction is cumbersome to determine, and worth more investigation.

5 RELATED WORK

Huang et al. created a static analysis tool called SUPOR to identify input fields in the UI that are sensitive user inputs. Huang et al. applied the tool to 16,000 popular Android applications and found that SUPOR achieves an average precision of 97.3% in detecting sensitive user inputs and has a false positive rate of 8.7% [Sean1].

Andow et al. created an automated approach called UIRef that determines the semantics of user inputs requested by Android applications. UIRef was used on a study of 50,162 applications and achieved an overall accuracy of 95.0% at semantics resolution and 82.1% at disambiguation [Kaan1].

6 CONCLUSION

In an ever-increasing digital and app-driven world, users want to receive a positive and safe experience from the applications they use on a day-to-day basis. Thus it is essential that before apps are released to the market, they are tested thoroughly for proper behaviors and performance in response to interactions with the user interface. In this paper, we discussed possible ways for sensitive user input to be leaked.

We analyzed 250 apps to find the correlation between the number of permissions requested and the number of sensitive inputs in an apk. Our r-value of 0.35 indicates that apps with more sensitive inputs seem to request more permissions. We also looked at those apps to find the breakdown of permission requests with respect to the percentage of applications. We found that apps that contain sensitive inputs

were 5.21 to 25.32 percent more likely to request each of the 22 analyzed input fields.

It is important to note that even a benign app with the best of intentions can unintentionally leak sensitive user information when combined with dangerous permissions. We believe that if an app requests sensitive user input, then dangerous permissions should be granted only if absolutely necessary to avoid the possibility of a data leak. We leave the task of finding the presence of sensitive user input leaks to future work.

7 REFERENCES

[Sean1] Jianjun Huang, Zhichun Li, Xusheng Xiao, Zhenyu Wu, Kangjie Lu, Xiangyu Zhang, and Guofei Jiang. 2015. SUPOR: Precise and Scalable Sensitive User Input Detection for Android Apps. In Proceedings of the USENIX Security Symposium.

[Sean2] Karthick, S. and Binu, S. 2017. Android security issues and solutions. *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. (2017).

[Kaan1] Andow, B., Acharya, A., Li, D., Enck, W., Singh, K. and Xie, T. 2017. UiRef. *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. (2017).

[Yadira 1] Permissions on Android: Android Developers: https://developer.android.com/guide/topics/permissions/overview

[Yadira 2] X. Xiao, Class Lectures, Topic: ”Module 11: UI Analysis”, Case School of Engineering, Case Western Reserve University, Cleveland, OH