Mapping Risk Assessment Strategy for COVID-19 Mobile Apps' Vulnerabilities

Tanusree Sharma[1], Hunter A. Dyer[2], Roy H. Campbell[3], Masooda Bashir[4]

[1,2,3,4] University of Illinois at Urbana-Champaign, Champaign IL 61820, USA

Abstract. Recent innovations in mobile technologies are playing an important and vital role in combating the COVID-19 pandemic. While mobile apps' functionality plays a crucial role in tackling the COVID-19 spread, it is also raising concerns about the associated privacy risks that users may face. Recent research studies have showed various technological measures on mobile applications that lack consideration of privacy risks in their data practices. For example, security vulnerabilities in COVID-19 apps can be exploited and therefore also pose privacy violations. In this paper, we focus on recent and newly developed COVID-19 apps and consider their threat landscape. Our objective was to identify security vulnerabilities that can lead to user-level privacy risks. We also formalize our approach by measuring the level of risk associated with assets and services that attackers may be targeting to capture during the exploitation. We utilized baseline risk assessment criteria within the scope of three specific security vulnerabilities that often exists in COVID-19 applications namely credential leaks, insecure communication, and HTTP request libraries. We present a proof of concept implementation for risk assessment of COVID-19 apps that can be utilized to evaluate privacy risk by the impact of assets and threat likelihood.

Keywords: Mobile apps, Privacy risks, Threat likelihood, COVID-19

1 Introduction

In the wave of the recent COVID-19 pandemic, mobile apps have been considered as an important technological medium in combatting the spread of the virus. While information sharing is a key component for the functionality of COVID-19 mobile apps, it is not clear if basic security and privacy protections are being put in place in these apps while we are rushing to deploy them around the world. For example, a recent research study found that the use of Google's Firebase Analytics service by OpenTrace (which is a version of related close-source app TraceTogether) has potential scopes of privacy risks around (IP-based) location track as well as storage of user phone numbers in the Firebase authentication services [1]. Since one of Google's primary business is advertising, this type of user data collection creates a potential conflict of interest [1]. Likewise, there exists inherited denial of service vulnerabilities in COVIDSafe version 1.0 and 1.1 from OpenTrace code which allows an attacker within Bluetooth range to crash the application, potentially can cause harm to user's device [2]. Furthermore, smartphones' heterogeneous nature of having various sensor data, communication logs, and mobility of user device throughout a day, involve connections to various networks that are often not secure. Therefore, sensitive data can be transferred from client applications across unsecured channels and lead user's information to possible exploitation. In addition, there are studies that show that some android users are not using HTTPS and therefore making users vulnerable to man in the middle attacks [3] and other privacy risks. In this context, users' personal information on such apps can make organizations economically attractive source to attackers [3]. Hence, traditional risks may reappear with increased impact through COVID-19 apps. Attackers can pose security challenges that includes widened attack vectors through location surveillance in COVID-19 applications. Such threats can make traditional countermeasures ineffective and take away users' trust from technology [4], [25].

For COVID-19 applications, user trust in technology is one of the important factors to consider. In contact tracing apps, proximity tracing process is supported by a backend server that distributes anonymous exposure information where the server is trusted not to add fake exposure events and remove real exposure events [5]. In this case, the backend server is expected to act solely as a communication platform rather than processing and it is assumed to have users' privacy intact even if the server is compromised. For the purpose of this study, we illustrate privacy risks through a scenario in which several security vulnerabilities could directly put the user's data at risk within the context of a COVID-19 contact tracing application. Our aim for this illustration is to demonstrate how seemingly minor vulnerabilities can lead to very undesirable results. While these attacks are not novel, they help illustrate how existing mobile environments can facilitate some very basic attacks that can violate privacy protections. To begin this illustration, we examine exposed API keys and how an attacker could obtain the keys by manually inspect the decompiled code. After obtaining keys, network traffic could be observed for requests to be made, and to find the target within the API. This attack can then be taken in multiple directions, but we will assume that in the general case, the API has the ability to submit user information. Keys could then be used to submit a large number of new fraudulent user accounts. This could act as a denial of service attack if the volume of requests and methodology is correct. It could also open the door to further attacks on the infrastructure by attempting to simulate a large number of locations, such that if one of the fake accounts could be marked as positive for COVID-19, it could lead to many legitimate users believing they have encountered someone that tested positive for COVID-19, when in fact they did not. This attack largely leads to the availability and integrity of the application to be put under strain. There may also be capabilities of the API if it was not designed well in which user information could be extracted with only a key, leading to a large, but preventable data breach.

Next, we examined insecure communication. There are many well-known issues with HTTP traffic. For example, it can be read by anyone that observed the request in transit. The use of HTTP enables a person-in-the middle scenario in which the attacker would be able to extract all information coming into the server, because HTTP isn't encrypted. This allows a large amount of sensitive data to be leaked. Finally, the nature of HTTP libraries doesn't inherently make them an attack vector, but having multiple HTTP libraries increases the attack surface by giving more places in which a vulnerability could be found and exploited. So, if our illustrative application had multiple HTTP libraries within it, then as an attacker, there are more libraries to look for possible exploitation or possibly use an already known exploit if it were not patched within the application. As an example, there might be some exploit within one library that allows for a person-in-the-middle attack that could intercept and decrypt HTTPS traffic, which would allow a large leak of personal information that was thought to be secure.

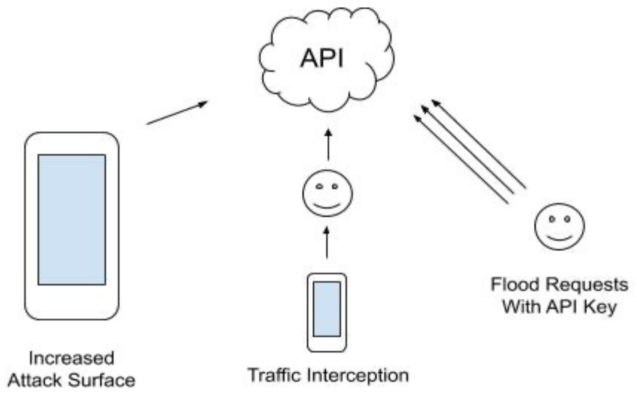


Fig. 1. Possible attack paths when security smells are present.

In COVID-19 application scenario, these exploitations and leakage of a user's information can also help build a social graph of people with which they had contact since in the system both identifiers and COVID-exposure are computed centrally and therefore, the backend server can associate uploaded ephemeral broadcast identifiers to permanent pseudo-identifiers for individual devices. Thus, the backend server can not only reconstruct a social graph, but it can reconstruct an interaction graph. COVID-19 apps and associated security threats can potentially create privacy risk. If data is manipulated or stolen due to insecure communication, then this technology will not be trustworthy to users which is a risk of authentication at user level. At the same time, if these apps are not working properly and not performing correct exposure notification, there can raise the issue of integrity of this application while credential leaks lead to a potential risk of confidentiality. Keeping previous and current security exploitation in mind and associated privacy risks, our study approach is to assess COVID-19 apps from different security parameters to identify security vulnerabilities in credential leaks, HTTP request libraries, and insecure communications. For this study, our motivation is to map these chosen security exploitations and develop assessment strategies for associated privacy risk, as well as provide suggestions for how these vulnerabilities can be addressed in the development process.

We demonstrated step by step how security vulnerabilities can be assessed and measured so that they can be quantified as risk factors. Then, we provided a risk assessment methodology that can be utilized when assessing privacy risk in mobile apps. While there are still a lack of formal risk assessment methods for determining risks at an individual level, we utilized a targeted risk assessment method to assess userspecific assets and threats that is focused on COVID-19 applications. We believe our study contributes towards the development of equations which considers users' assets and threats associated with the COVID-19 apps, as well as the likelihood of risk. Therefore, our research questions are:

RQ1: What are the possible privacy risks and security vulnerabilities that are specific to COVID-19 applications?

RQ2: How can we measure or quantify risk based on prior knowledge/likelihood of threats that considers recent privacy risk associated with those applications?

The remainder of this paper is organized as follows. Section 2 provides relevant background information. Section 3 describes our research methodology. We present our app analysis results and possible resolutions in Section 4. Section 5 presents App's Risk Assessment Strategy and then we conclude the paper with section 6 and 7 which provides a discussion of the implications, limitation, and future research directions as well as our conclusions.

2 Background

Several previous studies have shown the prevalence of smartphone privacy concerns and security threats around android permissions that can take place via privilege escalation attack that allows a malicious application to gain more capabilities leaked from a benign application [6], web API where message passing is also a medium of attack surface while providing availing inter-application collaboration. Adversarial can sniff, modify, steal or replace content with compromising user privacy [7], [19]. Again, malicious application can inject malicious messages/code which can lead to users' data breaches. Furthermore, there are security concerns around ready-to-use code without proper caution such as [8], poor authentication and authorization, and lack of proper encryption of sensitive data [9] which keeps exposing users' data to malicious actors.

These attack scenarios are not novel to COVID-19 apps. The urgency in testing basic security and privacy vulnerabilities within these apps are ever more vital because all types of organizations and governments around the world are rapidly working on developing and deploying contact tracing apps to track and mitigate the spread of COVID-19. While the developments of these apps are extremely important amid the COVID-19 pandemic, security and privacy researchers must address these basic anticipated vulnerabilities and the risks they may pose to users' personal information. One such research study analyzed Singapore's OpenTrace app and its use of Google firebase services to manage user data and deployment of reversible encryption and found that such method can be vulnerable to secret key disclosure [1]. Other researchers are proposing different security and privacy aware frameworks that are better at anticipating associated risk and adversarial attacks. Some of the proposed or implemented designs include decentralized proximity tracing [5]; construction of token in Bluetooth contact tracing [10]; distributed hash table to build a decentralized messaging system for infected users [11]; minimizing re identification risk strategy [12], user-centric privacy design [24] to ensure privacy is well preserved with formal security model mechanisms.

Generally, organizations and IT sectors implement security standards and controls, as well as risk assessments, to decrease the likelihood of risk for their assets and services from the evolving landscape of exploitation [11]. Qualitative and quantitative research has been done regarding decision making on security measures implementation based on the estimation of attack paths and security costs by attack graphs, tress, and other models [2], [13]. Our approach of assessing various assets and their associated risks as it pertains to COVID-19 apps where we consider users' personal information as an asset is valuable and unique. Previous research has identified different types of assets like personal data, financial assets, and personal and political reputation based on the privacy risks [14]. With these prior studies as our motivation, we have designed our experiment considering three security vulnerabilities in COVID-19 apps to assess risks and articulate assessment strategies for investigated vulnerabilities based on user level asset and threat likelihood.

3 Method

Following previous research papers' approach in examining privacy violations, we focused our study on specific threats, namely sensitive data exposure, broken authentication, and security misconfiguration. While there are many other security threats that can be evaluated, we believe these three make up the most common security issues and it is supported by previous studies [17], [2], [20]. Therefore, in terms of COVID-19 apps, we focus on Credential leaks as one of the security exploitations since it involves sensitive data including financial, healthcare, and PII to be compromised if there are not enough security measures, such as encryption for the different keys and data storage. Next, we considered insecure communication and HTTP request libraries which can lead to insecure default configurations, incomplete or ad-hoc configurations, misconfigured HTTP headers, and verbose error messages containing sensitive information. Therefore, we believe these three specific security vulnerabilities selected in our assessment of COVID-19 apps constitutes the most common vulnerabilities [2], [17], [21] that can be encountered in those apps. Thus, guarding against these vulnerabilities can ensure timely and secure configuration of COVID-19 apps.

App selection: We selected 7 apps from the recent CVE report on apps having frequent exploitation report [2]. Initially, we assessed apps by automated frameworks for understanding fundamental security requirements. This step is not necessarily a part of this study, but rather was used as our preliminary step towards our initial understanding of those apps. Our goal is to examine security vulnerabilities in terms of privacy violation.

Selecting security parameters to check: After having our initial inspection, we emphasize on credential leaks; HTTP Request Libraries and insecure communication based on security exploitations [15] [16] that have been identified by previous studies. In addition, our review of the national security vulnerabilities database [2] suggest that when evaluating security vulnerabilities, we are to place an increased priority on patching the most commonly known vulnerabilities and controls to avoid risks. The purpose of this database is to provide security professionals with an update on the most common and exploited security issues so that they can prioritize patching and place security accordingly. As outlined below, we evaluated each selected COVID-19 app for the presence of the three specific vulnerabilities. Next, we recorded how frequently these vulnerabilities occurred in our selected apps. In addition, we recorded potential user privacy violations that might take place from such vulnerabilities. More specifically, we performed the following steps:

- We selected 7 COVID-19 apps as described above to examine.
- We examined each of the 7 apps for the presences of three types of security vulnerabilities (credential leaks, insecure communication and HTTP request libraries).
- If the app had any of the 3 security vulnerabilities, then we analyzed/recorded how frequently those vulnerabilities occurred. For example, an app can have 3 credential leaks such as API key, password, and username leaks.
- We also investigated how these 3 security vulnerabilities can lead to privacy violations.

Mapping Risk Assessment Strategy for COVID-19 apps: After analyzing each of the 7 apps for security vulnerabilities as described above, we mapped associated privacy risks for users in respect to assets and services provided by those apps. To conduct this risk assessment, we measured the severity of privacy/security risk based on previous literature [18]. It is important to note that our measurement was not entirely based on performing quantitative evaluations but rather building and proposing an exploratory strategy to assess risk that can be tested/verified in future studies. More specifically, the following steps were performed for this mapping.

- Our risk strategy considered 3 security vulnerabilities and a list of associated risks that can impact user's and industries' assets. Assets include personal information, health information, location information, financial information that can impact availability, confidentiality, and integrity in any security infrastructure. For this study, we assessed impact on assets that would be compromised by these selected 3 security vulnerabilities.
- Then we considered the likelihood of privacy risks associated with those assets listed above. We adopted the Likelihood metrics from previous literature that included measures of a) historical risk data, (b) statistics of the known vulnerabilities, and (c) existing controls to reduce vulnerabilities. For example, in our study, credential leaks are considered a vulnerability. In this case, threat likelihood is calculated on the basis of, a) the likelihood of

credential leaks in the COVID-19 smartphone platform, b) the threat incident likelihood from previous incidents, i.e. statistics on threat incidents in the platform or previous incidents experienced by the user, and c) the relevant security control existence/absence.

- Therefore, we propose to calculate risk by the combination of impact on user privacy and likelihood of vulnerabilities. We followed this method in our risk calculation that has been proposed in previous risk assessment studies [13,14].

4 Result & Possible Solutions

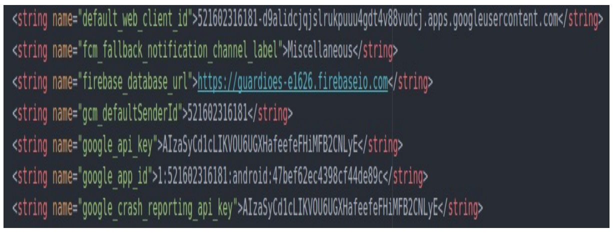
In this section, we reported our conducted analysis specifically for three security exploits for 7 apps. Three focus areas for security measurements are credential leaks, insecure communication, and HTTP request libraries.

4.1 Credential Leaks

The development of mobile applications often hinges upon using web applications and services through the application's programming interface, also known as an API, in order to serve dynamic content, collect data, or perform other complex operations that may not be otherwise appropriate or possible on users' devices through HTTP or HTTPS requests. Most APIs use keys for authentication within the headers of the requests. The consequences of keys being wrongfully obtained might mean private information is leaked, or financial repercussions based on billing agreements for the API which also fits under a broader topic of credential leaks where credentials are revealed to unauthorized entities through unintended channels. This could include emails, passwords, or tokens, among other sensitive information that might be used for authentication purposes. The primary focus of this assessment was to find types of credentials like API keys.

One email was found but it seemed to be a remnant of code that was meant for testing an upload of data. Within the apps examined, there were multiple instances in which API keys were hardcoded and easily accessible by decompiling the application, as was done within the study. This practice is often considered insecure partially due to the possibility of wrongfully obtaining keys. Inspection of these applications was done manually; only the code that would have been written by the developers was inspected. This not only made the manual inspection more feasible, but also allowed us to assess the extent of the credential leaks that the developers would have been directly responsible for. Due to the variance in length, possible characters, and forms that API keys may take, we searched for terms that would possibly appear near these credentials (terms like key, token, etc.) and inspected the files in which matches showed up. Results were counted if they were directly labeled as a key or token, which may not fully account for every single key hardcoded based on naming scheme or associated labels. Notably, the parameters for which we counted keys or tokens omitted pieces of data that may be labeled as "ids". There are some cases in which a value labeled ID would be equivalent to a key, and those scenarios would be excluded from this analysis.

Most of the API keys found in each case were found in the resources/res/values/string.xml file after decompiling as can be seen in an excerpt from CoronaSUS in fig. 2. These keys were mostly related to Google libraries and services.



```xml
<string name="default_web_client_id">521602316181-d9alidcjqjslrukpuuu4gdt4v88vudcj.apps.googleusercontent.com</string>
<string name="fcm_fallback_notification_channel_label">Miscellaneous</string>
<string name="firebase_database_url">https://guardioes-e1626.firebaseio.com</string>
<string name="gcm_defaultSenderId">521602316181</string>
<string name="google_api_key">AIzaSyCd1cLIKVOU6UGXHafeefeFHiMFB2CNLyE</string>
<string name="google_app_id">1:521602316181:android:47bef62ec4398cf44de89c</string>
<string name="google_crash_reporting_api_key">AIzaSyCd1cLIKVOU6UGXHafeefeFHiMFB2CNLyE</string>
```

Fig. 2. Example snippet of credential leak within CoronaSUS

Results in Table 1 denoted with a * were situations in which two of the leaked credentials were related to Google libraries, and had security features available to mitigate or prevent potential unauthorized use. These two credentials were uniformly named across the apps that they occurred in and were labeled google api key and google crash reporting api key. If we omit these findings, then we are left with three applications that had credentials leaked, Aarogya Setu, COVIDSafe, and האפליקציה.

Table 1. Leaked Credentials Total Breakdown by Application

Security Smell: 1	
Application	***Instances of leaked credentials***
Aarogya Setu	3*
ABTraceTogether	0
CoronaSUS	2*
COVIDSafe	1
Protego	2*
TraceTogether	2*
האפליקציה	3*

Privacy Risks With consideration of these occurrences not being disclosed to developers, discussion of further leaked credentials is discussed in aggregate. 4 keys found could be classified broadly as an application API key and 2 keys related to 3rd party services that handle application engagement and location management. Notably, the key that directly relates to the API could have direct consequences to the privacy of other users if data can be exfiltrated from the server through different endpoints. Inversely, it could also act as a way to deny service to users if an excess of requests is made. These two consequences are on opposite sides of the spectrum of things that can be done with keys in related circumstances. Oftentimes attacks are dependent implementations of the server, along with the possible utilities that the API provides. These attacks are also dependent on proper permissions and control of the API key not being implemented. Further, with the location service, there may be a risk that a malicious actor is able to gather user locations in the event where API is not designed well, or could possibly flood the API with excess data, which would be problematic for offering intended service of the mobile application it is used in. API implementations that are unsecured and don't have proper access control implementations in place, such as a restriction on requesting IP addresses can be subject to serious attacks if the proper care is taken with the key that is used, especially in a time in which decompiling an app is a readily available service with relatively low barrier to use.

Proposed resolution: In terms of solving the specific problem of leaving API keys hardcoded within applications is to make it more difficult to obtain such keys or key could not be obtained solely through decompiling the application. This can mean that if an adversary wanted to obtain the key, they might have to use methodologies for examining the memory of the application or other debugging tools. Alternatively, employing methodologies similar to what was seen with Google libraries in which key owners have the ability to strictly limit when and where the keys can be used, but this may open up other possible attacks.

4.2 Insecure Communication

With many mobile applications relying on web services in order to provide services, it is very important for the communication channels in which this information is conveyed to be secure. The predominant method for this communication is HTTP and HTTPS. HTTP is transmitted through plaintext methods, whereas HTTPS is transmitted through encrypted methods. The consequence of not using HTTPS is that it severely endangers the privacy for the user of the application of data being stolen or manipulated.

Within our analysis of URLs that were hardcoded into the applications, we only examined domains that would have been directly coded by developers of the app. URLs within 3rd party libraries or URLs included as part of the Android application build process were omitted. Notably, this excludes URLs from the schemas.android.com subdomain, as this is used to host base resources for most Android applications. These include domains such as apache.org. URLS that were hard-coded but did not contain the prefix of http or https were counted under undetermined as it would likely be determinant on the HTTP library used to make the request or how the string is later manipulated in code. The counts below are the number of unique URLs to avoid double counting if a URL is hardcoded in more than one place [22].

Table 2. Breakdown of URLs found through Manual Inspection

Application	Security Smell: 2		
	#HTTP	***#HTTPS***	***Undetermined***
Aarogya Setu	0	5	2
ABTraceTogether	0	5	0
CoronaSUS	1	5	2
COVIDSafe	0	5	1
Protego	0	7	2
TraceTogether	0	5	2
האפליקציה	0	78	2

Since the hardcoded URLs were inspected manually, it wasn't feasible at this time to find the exact purpose and behavior of each link through the code. However, insight to the URL's purpose can be found from its name and general form. The http: http://mobileapps.saude.gov.br/coronavirus and following the link in a browser only returns text in JSON format, which might indicate this is an API URL, though it is not clear what information might be served with requests to this URL, but if it is sensitive information, then it is a threat to user privacy as discussed earlier. Within links that were HTTPS, they could be grouped into two main descriptions. They were either linked to external websites or resources, such as FAQs, privacy agreements, or EULAs, or the URLs were used for API requests. The former is the driving reason for "היצקילפאה" to have had 78 hardcoded HTTPS links. Collectively, there was one definite case in which this code smell was present, while the undetermined URLS might be potential cases.

Proposed Resolution: Unfortunately, there aren't as many flexible fixes to this problem as there are with the API keys. One possible solution is to ensure that HTTP Request libraries used in your application (also discussed in the next section) are capable of making HTTPS requests, and that this functionality is used. The use of HTTPS is all dependent on the server being communicated with having the capability as well. So, in the event that developers are using an API they built and are hosting, they should take the time to ensure that they have properly set up HTTPS capabilities on their server to attempt to protect the inbound and outbound communications.

4.3 HTTP Request Libraries

It is common practice to use 3rd party libraries within software development to facilitate and simplify the development cycle. This can certainly prove to be efficient and effective for completing development, it can introduce other concerns in regard to potential credit leaks and cohesiveness of the application. While the implications of this security vulnerability may not be as significant as others, it is still worth considering in its relation to the other two security vulnerabilities.

The use of multiple HTTP libraries in each application can also increase the potential attack surface of the app, as it presents opportunities for more vulnerabilities to arise. This can pose a security threat to the application's infrastructure while also posing a threat to the users of the application. It is worth noting that the consequences of an attack on an HTTP library would be potentially reliant upon which URLs the HTTP library serves. The libraries used were determined by manual inspection of the 3rd party libraries that were included within each application. If the library was self-described within official documentation or webpages as a library for making generic HTTP or API related requests, then it was included within the survey of libraries below. 3rd party services that offered a library specifically for their own owned service were not included as these libraries are often specialized and provide other services. Libraries were surveyed based on the inclusion of library folders in the decompiled application.

In some cases, the libraries could not be fully assessed due to code obfuscation. Two applications, TraceTogether and היצקילפאה, have some obfuscation and files were not named in conventional naming conventions and had names of the form similar to C0000a.java, where the number could be a four-digit number and the last letter could be any letter of the alphabet. Some of the consequences of our assessment could be due to the compiler that we used. Results were gathered from the folders whose naming structure remained intact. The distribution of HTTP libraries is dominated by OKHTTP3 and has some other inclusions. As discussed earlier, there is some risk with using multiple libraries, and it seems that this selection of apps abides by this. The use of libraries that are written for a specific product in mind can be advantageous in that it should ideally take the burden of security considerations from the developers and places them onto the companies that have a better working knowledge of their product, and assumedly have more incentive, time, and resources to devote to such considerations compared to the developers using their library. Using HTTP libraries that are meant to flexibly handle different HTTP requests can place this burden back on the developers as it may not directly promote good practices for the purposes of API requests or other services that require key authentication.

5 App's Privacy Risk Assessment Strategy

In this study of COVID-19 app assessment, we have demonstrated associated risks for users' privacy. In this section, we specify our proposed risk assessment strategy for COVID-19 smartphone apps that is described in section 4. Our risk assessment strategy considers associated risks that can impact users'/industries' assets and the likelihood of those risks. We have found that our selected security vulnerabilities can lead to personal information leaks, data sharing/storage risks, and insecure communication between server and client application. Therefore, in our risk assessment strategy, we considered credential leaks, HTTP libraries, and insecure communication as attack vectors to do our analysis. The frequency of those attacks and the associated impact on users' privacy due to those vulnerabilities while using COVID-19 apps are shown in the results section. While our risk assessment strategy is somewhat exploratory, we believe that our proposed approach for evaluating COVID-19 apps is robust, as it not only considers the current privacy impact, but also includes statistics from past exploitations.

To assess COVID-19 smartphone apps' privacy risk, our first criteria was to evaluate its assets. Assets identified were related to COVID-19 apps' components, for example, personal information, health information, location information, financial information. Based on that, impact of such insecurity scenarios (loss of availability, confidentiality and integrity) can be assessed. For instance, the impact of data type is inferred to their associated data sources. This means that if there are 'personal' data types that those apps are dealing with, then the disclosure impact for the data source are: "exposure notification", "data Processing" and "Data storage/sharing" can be calculated, as follows:

Impact (Data Sharing) = max {Impact (personal data), Impact (health data)}
$\mathrm{I} = \sum_{a=0}^{n} \mathrm{avg}(\mathrm{I}_{\mathrm{Pd}}, \mathrm{I}_{\mathrm{hd}})$

Where each vulnerability here, such as, credential leaks, insecure communication and HTTP request library will be present and considered as a single vulnerability vector:

$$V_i = \{0,1\} \; for\ all\ i \, , i = 1,2,3,4, \dots , n$$

So, the overall COVID-19 smartphone apps' data sharing impact is the max impact related to personal data and health data. Some other attacks can happen to other types of assets, i.e., the device, data, applications, and connectivity.

The other variable that can be measured is threat likelihood for risk assessment that is assessed on the basis of: (a) experience and applicable statistics, (b) vulnerabilities, and (c) existing controls to reduce vulnerabilities. Each threat, for example, credential leaks is grouped in the appropriate attack vector dimension. Asset refers to the topics targeted by the threat. For example, Credential leaks are considered as the threat. In this case, threat likelihood is valuated on the basis of a) the likelihood of credential leaks in the COVID-19 smartphone platform, b) the threat incident likelihood from previous incidents, i.e.statistics on threat incidents in the platform or previous incidents experienced by the user, and c) the relevant security control existence (for mobile apps data management and security). Incident Information from the organization's historical database recorded in system log files can be used in modeling which can predict threats likelihood. In this case, existing model, such as, attack graph can be used for new incoming threats.

$$T_j = \{0,1\} \; for\ all\ j, \qquad j = 1,2,3,4, \dots , m$$

Where each threat here, like users' information leaks, data stolen/manipulations are considered as threat vector. where T_j represents an individual threat. Value 1 indicates the presence of this threat in the information systems and otherwise 0. So, within vulnerability and threat to individual level, likelihood of incoming threats can be measured by $L_{ji} = (T_j, V_i)$ where threat acting over a vulnerability [14].

Combining the impact assessment of assets/services and threat likelihood can be used to derive the risk for a particular security vulnerability in COVID-19 applications. Therefore, we can assess risk based on the current attack landscape and previous attack history assuming there exists controls against those threats. $C_k = \{0,1\}\, for\ all\ k, k = 1,2,3, \dots, p$

$$Risk = \sum_{j=1}^{m} \sum_{i=1}^{n} I.L_{ji}$$

6 Discussion

The advancement of ICTs and its critical role in society in recent times requires us to develop timely and robust test methods, reference data, proof of concept implementations, and technical analyses. The use of COVID-19 apps throughout the world as one mechanism to combat the pandemic demonstrates the urgent need for better data privacy/security management that accounts for not only known privacy risks but also for unintended privacy risk for both users and organizations. While COVID19 apps provide a timely vehicle for tackling the spread of the virus, it also provides a unique and global scenario for users' loss of privacy. Mobile platforms continue to be one of the main sources of personal information exchange and storage and therefore more vulnerable to un/expected attacks that can violate users' privacy [15]. To be able to protect against these risks, a risk assessment strategy is crucial. In our study, we propose a risk assessment strategy for COVID-19 apps based on our COVID-19 app assessment. Our assessment strategy includes the impact on users' privacy and the likelihood of those specific privacy vulnerabilities.

While the proposed assessment strategy is exploratory, we believe that our strategy is the initial and essential step towards building a more comprehensive privacy/security assessment framework for mobile applications. We hope to improve this strategy in subsequent iterations by assessing more mobile apps and validating our approach in order to develop better methods for evaluating and managing privacy risks.

Additionally, the complexities and innovative features of the mobile platforms will continue to introduce privacy and security risks that will need to be assessed quickly, cost-effectively, and easily, in order to reduce cyber risks. Therefore, including risk assessments as one part of the system evaluation process can be an effective approach that can facilitate decision making at all risk assessment hierarchy including organizational levels, mission/business process level, and information system level [5]. Furthermore, this type of risk assessment can be made part of the system development life cycle that involves pre-system acquisition and post system acquisition. Perhaps our proposed strategy can be utilized by software developers in their Software Development

Life Cycle (SDLC) to make decision on the proper privacy requirements based on the impact and likelihood score of the vulnerabilities.

As mentioned above, risk assessment strategy is an iterative development processes and it changes overtime with new threat landscape. Our proposed model provides the flexibility to add new security attack vectors and the associated privacy risk by the impact and likelihood variable. These two risk assessment variables consider both system risk and users' risk. Further research is needed to verify this initial proposed strategy and to validate our proof of concept framework.

While there are various studies related to COVID-19 apps' privacy and security, our study provides a new approach towards assessment of risk which is not present in the current literature of COVID-19 apps. Our proof of concept approach can be used on Covid19 mobile applications to assess privacy risks that maybe easily overlooked. While we believe our approach is a critical step forward, it is important to note that our security assessment considered only 3 main types of security vulnerabilities which does pose a limitation. However, in future studies additional vulnerabilities can be added to the list to broaden the scope and the privacy risks. Another limitation of this study is that our initial analysis considered only 6 COVID-19 mobile apps and therefore in order to confirm these findings and extend the scope of our findings additional Covid-19 apps needs to be considered and analyzed.

For our future research, we will be examining a larger and diverse set of COVID-19 apps for security vulnerabilities to conduct our experiment. In addition, we will be utilizing our risk strategy on a publicly available risk incident data set in order to expand and focus our strategy.

7 Conclusion

Protecting user's privacy and security against adversaries and unauthorized access continues to be a challenging task in the age of technological innovation. Mobile devices and applications make this task even more difficult due to its nature of constant information sharing that includes a wide range of user's personal information. In our paper, we demonstrate specific security vulnerabilities relate to COVID-19 apps and provide proof of concept strategy that can be utilized for privacy and security risk assessment. We believe that our proposed risk assessment strategy can be expanded and developed further into a framework that can provide an efficient and novel approach for assessing security and privacy risks in mobile application. In our future work, we plan to develop this risk assessment framework further by including more training and testing data from public database.

References

1. Leith, D., & Farrell, S. (2020). Coronavirus Contact Tracing App Privacy: What Data Is Shared By The Singapore OpenTrace App?

2. NIST, National vulnerability database, automating vulnerability management, security measurement and compliance checking, http://nvd.nist.gov/home.cfm
3. Wei, X., & Wolf, M. (2017). A survey on HTTPS implementation by Android apps: issues and countermeasures. *Applied Computing and Informatics*, *13*(2), 101-117.
4. Sharma, T., Bashir, M. Use of apps in the COVID-19 response and the loss of privacy protection. *Nat Med* (2020).
5. Troncoso, C., Payer, M., Hubaux, J. P., Salathé, M., Larus, J., Bugnion, E., ... & Barman, L. (2020). Decentralized privacy-preserving proximity tracing. arXiv preprint arXiv:2005.12273.
6. Davi, L., Dmitrienko, A., Sadeghi, A. R., & Winandy, M. (2010, October). Privilege escalation attacks on android. In international conference on Information security (pp. 346-360). Springer, Berlin, Heidelberg.
7. Chin, E., Felt, A. P., Greenwood, K., & Wagner, D. (2011, June). Analyzing interapplication communication in Android. In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 239-252).
8. Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., & Fahl, S. (2017, May). Stack overflow considered harmful? the impact of copy&paste on android application security. In 2017 IEEE Symposium on Security and Privacy (SP) (pp. 121136). IEEE.
9. Krishnan, M. (2015). Survey on Security Risks in Android OS and an Introduction to Samsung KNOX. International Journal of Computer Science and Information Technologies, 6(4), 3965-3967.
10. Cho, H., Ippolito, D., & Yu, Y. W. (2020). Contact tracing mobile apps for COVID19: Privacy considerations and related trade-offs. arXiv preprint arXiv:2003.11511.
11. Viduto, V., Maple, C., Huang, W., & LóPez-PeréZ, D. (2012). A novel risk assessment and optimisation model for a multi-objective network security countermeasure selection problem. *Decision Support Systems*, *53*(3), 599-610.
12. Sharma, T., Bambenek, J. C., & Bashir, M. (2020). Preserving Privacy in Cyberphysical-social Systems: An Anonymity and Access Control Approach.
13. Wang, L., Noel, S., & Jajodia, S. (2006). Minimum-cost network hardening using attack graphs. *Computer Communications*, *29*(18), 3812-3824.
14. Theoharidou, M., Mylonas, A., & Gritzalis, D. (2012, June). A risk assessment method for smartphones. In *IFIP International Information Security Conference* (pp. 443-456). Springer, Berlin, Heidelberg
15. Stevens, R., Gibler, C., Crussell, J., Erickson, J., & Chen, H. (2012, May). Investigating user privacy in android ad libraries. In *Workshop on Mobile Security Technologies (MoST)* (Vol. 10). Citeseer.
16. Zhou, Y., Wu, L., Wang, Z., & Jiang, X. (2015, June). Harvesting developer credentials in android apps. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (pp. 1-12.
17. Wichers, D. (2013). Owasp top-10 2013. *OWASP Foundation, February*.
18. Hiller, J. S., & Russell, R. S. (2017). Privacy in crises: The NIST privacy framework. *Journal of Contingencies and Crisis Management*, *25*(1), 31-38.

19. Sharma, T., Bashir, M. (2020, July). Privacy apps for smartphones: An assessment of users' preferences and limitations. In *International Conference on Human-Computer Interaction* (pp.533-546). Springer, Cham.
20. Sharma, T., Bashir, M. (2020). Are PETs (Privacy Enhancing Technologies) Giving Protection for smartphones?--A Case Study. arXiv preprint arXiv: 2007 04444.
21. Sun, R., Wang, W., Xue, M., Tyson, G., Camptepe, S., & Ranasinghe, D. (2020). Vetting security and privacy of global COVID-19 contact Tracing applications. arXiv preprint arXiv: 2006.10933.
22. Santa Maria Shithil, T.K.S., & Sharma, T. A Dynamic Data Placement Policy for Heterogeneous Hadoop Cluster.
23. Lewis, T.L., & Wyatt, J.C. (2014). mHealth and mobile medical apps: a framework to assess risk and promote safer use. *Journal of medical Internet research, 16(9),* e210.
24. Sharma, T., Wang, T., & Bashir, M. (2020). Advocating for Users' Privacy Protections: A Case study of COVID-19 apps.
25. Sharma, T., & Bashir, M. (2020, July). An analysis of phishing emails and how the human vulnerabilities are exploited. In *International Conference on Applied Human Factors and Ergonomics* (pp. 49-55). Springer, Cham.