2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) | 978-1-6654-0296-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSE43902.2021.00101

An Empirical Assessment of Global COVID-19 Contact Tracing Applications

Ruoxi Sun*, Wei Wang*, Minhui Xue*, Gareth Tyson†, Seyit Camtepe‡, and Damith C. Ranasinghe*
*The University of Adelaide, Australia
†Queen Mary University of London, United Kingdom
‡CSIRO-Data61, Australia

***Abstract*—The rapid spread of COVID-19 has made manual contact tracing difficult. Thus, various public health authorities have experimented with automatic contact tracing using mobile applications (or "apps"). These apps, however, have raised security and privacy concerns. In this paper, we propose an automated security and privacy assessment tool—COVIDGUARDIAN—which combines identification and analysis of Personal Identification Information (PII), static program analysis and data flow analysis, to determine security and privacy weaknesses. Furthermore, in light of our findings, we undertake a user study to investigate concerns regarding contact tracing apps. We hope that COVIDGUARDIAN, and the issues raised through responsible disclosure to vendors, can contribute to the safe deployment of mobile contact tracing. As part of this, we offer concrete guidelines, and highlight gaps between user requirements and app performance.**

I. INTRODUCTION

COVID-19 is now a global pandemic affecting over 200 countries, after its first recorded outbreak in December 2019. To counter its spread, numerous measures have been undertaken by public health authorities, *e.g.*, quarantining, lockdowns, curfews, physical distancing, and mandatory use of face masks. Identifying those who have been in close contact with infected individuals, followed by self-isolation (so called *contact tracing*) has proven particularly effective [60]. Consequently, contact tracing has emerged as a key tool to mitigate the spread of COVID-19.

Unfortunately, manual contact tracing, using an army of "detectives" has proven challenging for many countries [28, 34, 48]. Therefore, authorities around the world have sought to develop contact tracing applications (or "apps") that can automate the process. A plethora of country-centric contact tracing apps and services are currently deployed. These include the Health Code in China [62], the public COVID-19 website in South Korea [19], and the mobile contact tracing apps released in Australia [33], Germany [47], Israel [58], Singapore [24], and United Kingdom [59]. These apps operate by attempting to record prolonged and close proximity between individuals by using proximity sensing methods, *e.g.*, Bluetooth. The data gathered is then used to notify people who may have come in contact with an infected person.

Proponents argue that the low cost and scalable nature of contact tracing apps make them an attractive tool for health authorities. However, they have proven controversial due to potential violations of privacy [49] and the security consequences of the mass-scale installation of (rapidly developed) apps across entire populations. Despite attempts to alleviate these concerns, it is well-known that the anonymization of individuals' information is a challenging problem [32]. To mitigate these concerns, we develop a methodology for assessing the security and privacy weaknesses of COVID-19 contact tracing apps.

TABLE I
CONTACT TRACING APPS CONSIDERED IN OUR STUDY.

#	Applications	Country	Downloads	#	Applications*	Country	Downloads
1	COVIDSafe	Australia	1M	21	STOP COVID19 CAT	Spain	500K
2	Hamagen	Israel	1M	22	CG Covid-19 ePass	India	500K
3	TraceTogether	Singapore	500K	23	StopTheSpread COVID-19	UK	100K
4	StopCovid France	France	1M	24	Stop COVID-19 KG	Kyrgyzstan	10K
5	Next Step (DP3T)	Switzerland	N/A	25	BeAware Bahrain	Bahrain	100K
6	Corona Warn App	German	5M	26	Nepal COVID-19 Surveillance	Nepal	5K
7	NHS Test and Tracing App	UK	10K	27	Stop Covid	Georgia	100K
8	TraceCORONA	Germany	N/A	28	Contact Tracing	USA	10K
9	Private Kit	USA	10K	29	Contact Tracer	USA	10K
10	MySejahtera	Malaysia	500K	30	Coronavirus Algérie	Algeria	100K
11	Smittestop	Denmark	10K	31	CoronaReport	Austria	10K
12	Covid Alert	Canada	500K	32	Covid19!	Czech	10K
13	SwissCovid	Switzerland	500K	33	Coronavirus Bolivia	Bolivia	50K
14	Bluezone	Vietnam	100K	34	Coronavirus - SUS	Brazil	1M
15	COCOA	Japan	5M	35	COVA Punjab	India	1M
16	Immuni	Italy	1M	36	SOS CORONA	Mali	10K
17	Stopp Corona	Austria	100K	37	Hamro Swasthya	Nepal	50K
18	Aarogya Setu	India	100M	38	COVID Radar	Netherlands	50K
19	EHTERAZ	Qatar	1M	39	NICD COVID-19 Case Investigation	South Africa	10K
20	Vietnam Health Declaration	Vietnam	100K	40	Coronavirus UY	Uruguay	100K

Specifically, this paper consists of the following key tasks: (*i*) we develop a tool, COVIDGUARDIAN, which uses static and dynamic program analysis, as well as a keyword database utilizing natural language processing (NLP) technology, to identify security weaknesses and personally identifiable information (PII) leakage in apps; (*ii*) we conduct a comprehensive security and privacy assessment across 40 state-of-the-practice global contact tracing apps (listed in Table I); (*iii*) based on the assessment results, we conduct a user study involving 373 participants, to investigate user concerns and the requirements of contact tracing apps. Through our study, we aim to answer the following research questions:

- **RQ1:** What is the performance of our security and privacy assessment methodology, COVIDGUARDIAN, compared to state-of-the-practice mobile app assessment tools?
- **RQ2:** What is the security and privacy status of state-of-the-practice contact tracing apps?
- **RQ3:** What is the robustness of state-of-the-practice contact tracing apps against potential security and privacy threats?
- **RQ4:** What are the user concerns and requirements of contact tracing apps?

The main contributions of our study are as follows.

978-1-6654-0296-5/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSE43902.2021.00101

- We develop COVIDGUARDIAN,[1] the *first* automated security and privacy assessment tool that tests contact tracing apps for security weaknesses, malware, embedded trackers and private information leakage. COVIDGUARDIAN outperforms 4 state-of-the-practice industrial and open-source tools.
- We assess the security and privacy status of 40 worldwide Android contact tracing apps. We discover more than 50% of the apps pose potential security risks due to: (*i*) employing cryptographic algorithms that are insecure or not part of best practice (72.5%); and (*ii*) storing sensitive information in clear text that could be potentially read by attackers (55.0%). Over 40% of apps pose security risks through Manifest weaknesses, *e.g.*, allowing permissions for backup (hence, the copying of potentially unencrypted application data). Further, we identify that approximately 75% of the apps contain at least one tracker, potentially causing privacy violations, *i.e.*, leaks that lead to exposing PII to third parties.
- By reviewing the state-of-the-art, we identify four major privacy and security threats against contact tracing apps. Our threat analysis finds that apps adopting decentralized architectures are not necessarily more secure than those adopting centralized architectures (by our measures). We also conduct a user study involving 373 participants, to investigate user concerns and requirements. The survey results indicate that the tracing accuracy and potential privacy risks are the two major concerns. Compared to users' expectations of accurate proximity recording, users are more likely to use contact tracing apps with better privacy by design.
- We have disclosed our security and privacy assessment reports to the related stakeholders on 23 May 2020. We have received acknowledgements from numerous vendors, such as MySejahtera (Malaysia), Contact Tracer (USA), and Private Kit (USA). Our re-assessments shown in Table V confirm that their updates have addressed several of the issues identified.

We believe our study can provide useful insights for government policy makers, developers, and researchers to build secure contact tracing apps, and to contain infectious diseases in the present and future.

II. Background and Related Work

A. Taxonomy of Contact Tracing Apps

The country-centric contact tracing apps we study fall into two broad categories: (*i*) centralized and (*ii*) decentralized.

Centralized architectures. Many apps utilize a centralized system in which a central server is responsible for: (*i*) collecting the contact records from diagnosed users; and (*ii*) evaluating the health status of users and selecting who to notify. For example, in China and South Korea, centralized contact tracing systems were rapidly developed and released. These systems helped health authorities to successfully control the spread of COVID-19. However, a huge amount of PII was collected [46, 62].

For instance, *TraceTogether* [24] from Singapore and *COVIDSafe* [33] from Australia rely on proximity tracing via Bluetooth broadcasts from apps. That said, designs that expose PII may not work well in countries with certain societal norms. Thus, many western countries developed solutions with no PII related information exchange, *e.g.*, *PEPP-PT* [70] and the ROBERT system [69] implemented by *StopCovid France*.

Decentralized architectures. The second type of solution is decentralized, where (*i*) the back-end server is only responsible for collecting the anonymous identifiers of diagnosed users; and (*ii*) the data is processed locally on the device to identify who to alert. This design prevents the central server from knowing at-risk persons or their contacts.

These decentralized apps operate in a range of ways. For example, *Hamagen* [58] relies on location information. Users download the location history of diagnosed users from a back-end server to check if they been in contact with any infected people. Other decentralized solutions, *e.g.*, *DP3T* [71] and the Exposure Notification framework by Google and Apple (Gapple) [4], utilize Bluetooth beacons. In this design, users periodically download the anonymous identifiers of infected people and compare them against previously encountered beacons to compute the risk of exposure. Note that *NHS COVID-19* [59] and *Corona Warn App* [64] implement the Gapple framework. This design paradigm reduces the privacy exposure of users as only anonymous identifications are shared.

B. Related Work

A number of prior works have utilized similar methodologies to COVIDGUARDIAN. We discuss them below.

Security and privacy analysis for mobile apps. COVIDGUARDIAN relies on static code analysis. This is performed by examining source code for signs of security vulnerabilities without executing the program. In contrast, dynamic analysis executes the code. Whereas static analysis often suffers from false positives, dynamic analysis is limited by the execution coverage [63]. Several studies [29, 30, 52, 55, 68, 75, 78] have used static analysis to analyze different types of software in search of malicious behaviours and privacy leaks. Static analysis techniques are also widely used in the practical assessment of mobile apps. For example, QARK [12] is a static code analysis tool designed to discover various mobile security-related vulnerabilities, either in source code or APKs. ANDROBUGS [1] is a framework that helps developers find potential mobile security vulnerabilities by pattern-matching. MOBSF [9] offers automated application penetration testing, malware analysis, and a security and privacy assessment framework. FLOWDROID [21] statically computes data flows in apps to understand which parts of the code that data may be exposed to.

Notably, these off-the-shelf tools only utilize syntax-based scanning and data-flows. Therefore, they cannot verify any identified vulnerabilities, which leads to numerous false positives that are not relevant to PII data leakage. Thus, COVID-

[1]Publicly available at https://covid-guardian.github.io/.

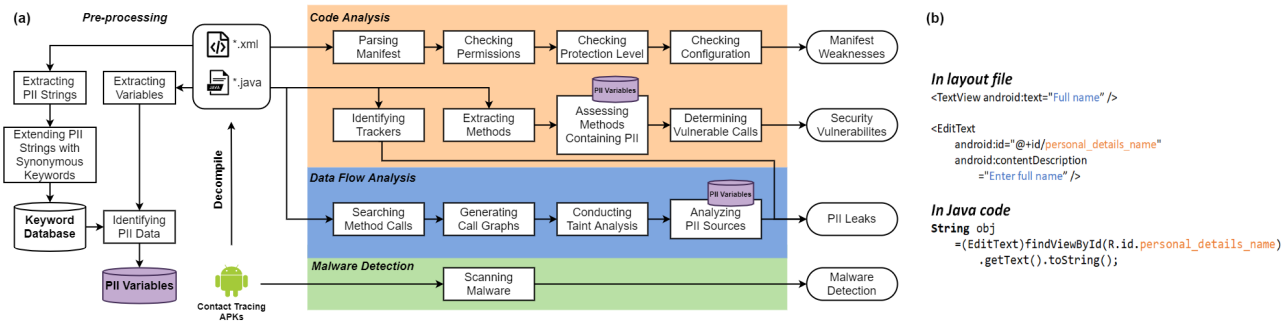

Fig. 1. (a) COVIDGUARDIAN: An overview of our security and privacy assessment methodology; (b) An example pattern of rendering a view widget.

GUARDIAN complements this with other methodologies, including the use of third party malware detection and techniques for data flow analysis.

Contact tracing apps analysis. Several works [23, 36] have focused on security and privacy analysis of the Bluetooth and cryptography specifications published by Apple and Google [42, 43], arguing that significant risks may be present. Other work [51, 54] has conducted a review of centralized and decentralized solutions, and proposed privacy-preserving contact tracing using a zero-knowledge protocol. Sun *et al.* [67] proposed a privacy-by-design solution, termed VENUETRACE, which enables contact tracing of users based on venues they have visited. VENUETRACE preserves user's privacy by avoiding information exchanges between users and no private data is exposed to back-end servers. He *et al.* [38] inspected the broad implications of COVID-19 related apps, identifying the presence of malware. Wang *et al.* [74] also performed a statistical analysis of contact tracing app popularity, as well as user reviews.

Our work. In contrast to recent works on analyzing contact tracing apps [23, 31, 36, 73, 73, 76], we not only propose and develop a holistic automated security and privacy assessment tool, COVIDGUARDIAN, but also undertake user studies to perceive users' concerns and requirements to reinforce security and privacy by design. COVIDGUARDIAN works for both centralized and decentralized apps.

III. METHODOLOGY OF COVIDGUARDIAN

In this section, we propose an automated security and privacy assessment tool, COVIDGUARDIAN, to identify security and privacy risks in contact tracing apps. Using COVIDGUARDIAN, we conduct a security and privacy assessment of 40 contact tracing apps from Google Play Store and evaluate their security performance against four categories: (*i*) manifest weaknesses; (*ii*) general security vulnerabilities; (*iii*) data leaks (with a focus on PII); and (*iv*) malware detection (see Table II). Further, we compare COVIDGUARDIAN with several state-of-the-practice tools to evaluate its ability and performance.

An overview of COVIDGUARDIAN is shown in Figure 1(a). We first compile a set of PII items, then perform: (*i*) code analysis to detect Manifest weaknesses and security risks; (*ii*) data flow analysis to reveal the privacy leaks in contact tracing apps; and (*iii*) malware detection.

PII identification (pre-processing). Considering the large-scale use of COVID-19 contact tracing apps and the concerns about PII data collection, we pre-process the contact tracing apps to construct a PII keyword database. Figure 1(a) shows the process of PII keyword database construction and the approach to identifying variables that contain PII (defined as *PII Variables*).

To construct the keyword database, we first focus on the PII input by users through widgets in the user interface, *e.g.*, `EditText`. We decompile the Android APKs and extract all strings defined in layout files and resource files, including the widget ID name of any `EditText` components, the hint text, and the text in the `TextView`. We manually filter the strings and keep the ones related to PII as seed keywords, *e.g.*, name, phone number, postcode, and password. We then utilize WORD2VEC [57] to expand our keyword pool with synonymous words. To ensure the generalization of our PII keyword database, we train the model with the text extracted from 54,371 general apps by the project, SUPOR [40]. WORD2VEC presents each word as a vector, and the vectors of synonymous words will have a small cosine distance. After training, for each of the seed keywords, we identify the top 5 synonymous words, and then feed them into the keyword database after manual validation. Other numbers of synonyms could be applied in practice, but a larger number will increase the workload of manual review. Using Google translate, we check that all apps that have other languages also have contexts (strings) in English, and that all variable names in source code are in English. Considering other languages may increase the efforts of manual filtering, and we therefore only extract English keywords.

After the keyword database is built, we identify which variables in the code are related to PII. We do this by keyword matching, which binds semantics to variables. Concretely, we first extract variables related to the widgets in the user interface, *e.g.*, `EditText` for user input, and `TextView` for hints or display text. We then determine a variable as a PII Variable if it matches with any keyword in the database. For example, in Figure 1(b) we present a typical layout XML file for a widget and the Java code to access it. The text attributes in the layout file can help users quickly understand the usage of a widget, *e.g.*, the string "Full name" for an input text box. When the app is compiled, the widget is

referenced by an integer ID which is typically assigned in the layout XML file as a string, in the `id` attribute, *e.g.*, `personal_details_name`. By keyword matching (*e.g.*, "name") we tag the variable `obj` as a PII Variable, as it gets content from the `EditText` widget where a user will input their full name. Although the tool is tailored to focus on contact tracing apps, our synonym compilation and keyword matching framework can also be adapted to other contexts.

Code analysis. We next perform static analysis on the Android Package (APK) binary files. We first decompile the APK of each app to its corresponding *class* and *xml* files. As shown in Figure 1(a), the de-compiled `AndroidManifest.xml` file is first parsed to extract essential information about the app, such as `Permission`, `Components`, or `Intents`. Then, we assess requested permissions and examine whether all `Components` (*e.g.*, `Service`, `Receiver`, `Activity`, `Provider`) are protected by at least one permission explicitly requested in Manifest files. Other attribute configurations, such as the `allowBackup`, `debuggable`, and `networkSecurityConfig` flags, will also be checked.

The Extracting Method module matches methods in decompiled files with pre-defined rules to extract potentially vulnerable methods. For example, if a method contains the keyword `.hashCode()`, it relies on the Java Hash Code (a weak hash function). The Assessing Methods Containing PII module utilizes the PII Variable database to identify methods containing potential PII. However, as a weakness could be defined in a third-party API, the vulnerable method inspected may never actually be executed during run-time. To address this, the Determining Vulnerable Calls module assesses whether a vulnerable method is actually called and determines whether the PII data is accessed. COVIDGUARDIAN records all the vulnerabilities listed in the *Manifest Weaknesses* and *Vulnerabilities* categories in Table II.

The assessed vulnerabilities include SQL injection, IP address disclosure, hard-coded encryption keys, improper encryption, use of insufficiently random values (CWE 330) [6], insecure hash functions, and remote WebView debugging being enabled. To increase accuracy, COVIDGUARDIAN not only relies on the detection of a vulnerable method being called, but also employs PII data matching. For example, a vulnerable method detection rule may use keywords "log" or "print" to locate the method calls related to "data logging", *e.g.*, `Log.v()` and `System.out.print()`. We further check whether the logged data contains PII by matching the inputs with the PII Variables.

Finally, the trackers in apps (*e.g.*, Google Firebase Analytics, Facebook Analytics, and Microsoft Appcenter Analytics) are detected by the Tracker Identification module and recorded in the *Privacy Leaks* category in Table II.

Data flow analysis. We next conduct a data flow analysis to identify high risk privacy leaks. The data flow analysis extracts the paths from data sources to sinks, and the code statements transmitting the data outside of the app. We define *sources* as calls to any PII data we identify, *e.g.*, `getViewById(int)` and `getText()`. Furthermore, we also consider methods

TABLE II
SECURITY AND PRIVACY ASSESSMENT CATEGORY.

Assessment Category	Security and Privacy Risks	COVIDGUARDIAN	MOBSF	ANDROBUGS	QARK	FLOWDROID
Manifest Weaknesses	Insecure flag settings (e.g., app data backup allowed)	●	○	○	○	
	Non-standard launch mode	●	○		○	
	Clear text traffic	●	○	○		
Security Vulnerabilities	Sensitive data logged	●	○		○	
	SQL injection	●	○	○		
	IP address disclosure	●				
	Uses hard-coded encryption key	●	○	○	○	
	Uses improper encryption	●	○	○	○	
	Uses insecure SecureRandom	●	○			
	Uses insecure hash function	●	○	○		
	Remote WebView debugging enabled	●	○		○	
PII Leaks	Trackers	●	○			
	Potential Leakage Paths from Sources to Sinks	●				○
Malware Detection	Viruses, worms, Trojans and other kinds of malicious content	●				

● Automated flow analysis, ○ Syntax-based scanning

that may obtain personal information without user input, *e.g.*, `getLatitude()` for geographic location information and `database.Cursor.getString()` for database queries. Note that, although unauthorized users cannot directly access sources (only sinks), PII data may still leak during storage or transmission activities. For example, an app may not have permission to access location data directly (the source), but it could obtain that information from the SMS outbox (the sink). If PII data flows into a code point where unauthorized users or apps can access it (*e.g.*, via local storage, external storage or SMS), the confidentiality of the PII is broken. Here we define *sinks* as methods that may leak sources through specific channels, *e.g.*, `SharedPreferences$Editor.putString()`, `Bundle.putAll()`, and `SmsManager.sendTextMessage()`.

Thus, we next search the app for lifecycle and callback methods. Using this, we generate a call graph. Starting at the detected sources, the analysis tracks taints by traversing the call graph. If PII data flows from a source to a sink, it indicates that there is a potential privacy leak path. To reduce false-positives, we conduct a backward flow analysis. If the vulnerable code is reachable (*i.e.*, not dead code) and contains PII Variables, we determine it is a potential valid privacy leak. For example, if we find there is a PII Variable that flows into a sink (*e.g.*, Bundle, Log output, SMS) where unauthorized users can access, we will trace it backwards to its source and confirm whether the source is reachable. If reachable, we consider it as a privacy leak.

Malware detection. To complement the code analysis, we rely on malware scanners to flag malicious artifacts in contact tracing apps. COVIDGUARDIAN sends the APKs to VIRUSTOTAL [18], a free online service that integrates over 70 antivirus scanners. Note this has been widely adopted by the research community [39, 41, 55]. As shown in Table II, the results of malware detection identify viruses, worms, Trojans, and other malicious content embedded in the apps.

Implementation. COVIDGUARDIAN includes two components: static code and PII data-flow analysis engines. The static code analysis engine employs JADX [7] to decompile the `dex` byte code of APK files to Java code. This allows the engine to generate an abstract syntax tree (AST) and create call graphs. Then, the engine scans the given source code to find risks listed in the OWASP [10]. Finally, the taint analysis engine utilizes the call graphs previously generated, with the list of sinks and sources, to locate private data leakage.

To perform the detection, we first summarize 220 types of (suspected) function calls, which we then use to identify vulnerabilities in the source codes. Then 195 taint sources and sinks are collected to analyze PII leakage through the AST. Finally, the static code analysis engine verifies whether the app logic invokes the data leakage functions or not.

IV. EVALUATION AND RESULTS

A. Selection of apps under-Study

To evaluate COVIDGUARDIAN, and explore the risks associated with popular contact tracing apps, we curate a list of apps to study. To achieve this, we first search for keywords in the Google Play Store, *e.g.*, "contact tracing", "Covid", and "tracing coronavirus". We also search for known official apps from countries, *e.g.*, the COVIDSafe recommended by the Australian government. After a contact tracing app is found, we assess its functionality by reading the app description and select those with in excess of 10,000 downloads. We also include two beta apps. Subsequently, we include the app into the set and look for new apps through the recommendation links in the app store. We repeat this until there are no more contact tracing apps found. At last, we finalized the list of 40 contact tracing apps, as shown in Table I. More detailed information, such as the versions of apps, is provided in our open source website.

B. RQ1: Evaluation of COVIDGUARDIAN

Comparison with the state-of-the-practice tools. To evaluate the effectiveness and accuracy of COVIDGUARDIAN, we compare against four industrial and open-source state-of-the-practice security assessment tools. These tools are selected based on the number of stars on GitHub (*e.g.*, MobSF 7.6K, Qark 2.4K), their update frequency, usability, and their ability to analyze the security of Android apps. MobSF is recommended by OWASP [2]; Qark and AndroBugs are widely used open-source security assessment tools for Android apps; FlowDroid is a classic tool for static taint analysis. We therefore believe they offer an effective baseline to compare against.

We apply all four tools to the 40 contact tracing apps understudy. The detection precision results are listed in Table III. The number of *types* is the sum of the number of risks and leaks identified by COVIDGUARDIAN. The precision rates are obtained through manual validation (*i.e.*, filtering out all false positives). The worst performing tool is FLOWDROID, which generates a large number of false positives that fall into "Log" related sinks. This is because FLOWDROID marks all log methods as sinks without considering whether the logged data is PII or not. For instance, in TraceTogether, error messages such as `SQLiteException` (from the stack trace) are logged by the `Log.e` method which matches the keywords "log" and is falsely identified as a privacy leak.

TABLE III
TOOL COMPARISON.

Tools	# Types	Precision
COVIDGUARDIAN	315	96.19%
MOBSF	213	47.41%
ANDROBUGS	76	80.26%
FLOWDROID	201	40.32%
QARK	93	84.94%

TABLE IV
IDENTIFIED TRACKERS.

Trackers	# Apps	Percentage
Google Firebase	25	71.4%
Google CrashLytics	6	17.1%
Other Google trackers	4	11.4%
Facebook trackers	3	8.6%
Other trackers	9	25.7%

In contrast, COVIDGUARDIAN is able to identify most log related false positives through analyzing the PII Variables (precision of 96.19%). It is also optimized to filter out non-private data, such as `Locale.getCountry`. Therefore, most false positives are removed automatically, and only eight false positives (which are from database query results but are not sensitive) are found.

Another type of false positive found in COVIDGUARDIAN is related to SQL Injections. For instance, TraceTogether encapsulates all SQL manipulation methods to limit the input to the SQL query. Both MOBSF and ANDROBUGS regard them as at risk of SQL Injections, since they analyze apps by keyword scanning. We manually analyzed all 40 apps and found that ten false positives (of which inputs are limited to several constants in the app but are still regarded as injected) fall into this category in COVIDGUARDIAN.

Threats to validity. Considering that both the code analysis and data flow analysis rely on keyword matching, a potential cause of false negatives is poorly chosen of keywords. This could mean that some vulnerabilities are not defined in the analysis rules. Similarly, in our data flow analysis, although we update the sources and sinks extracted by SuSi with newly defined ones based on the PII data we identify, there may exist PII leakage that does not match any sources or sinks. We aim to improve the false negatives by updating the rules and keywords database in the future. Currently, our empirical assessment accentuates the identified vulnerabilities and privacy leakage paths.

> **Answer to RQ1.** State-of-the-practice tools are less effective (*i.e.*, lower precision) compared to COVIDGUARDIAN. Our assessment methodology can be used to identify security weaknesses with high precision.

C. RQ2: Empirical Assessment Results

We next explore the presence of security vulnerabilities among the 40 considered apps using COVIDGUARDIAN.

Code analysis results. Figure 2 shows the percentage of contact tracing apps that have security weaknesses found via our code analysis. We observe that 42.5% of apps do not set the flag `allowBackup` to `False`. Consequently, users with enabled USB debugging can copy application data from the device. Other weaknesses identified are related to

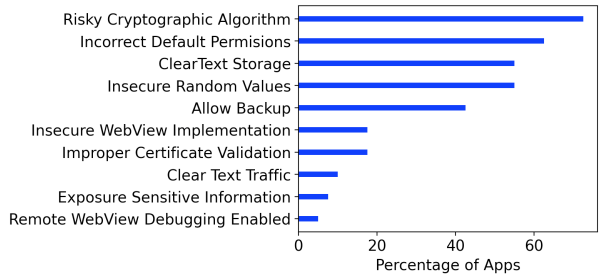

Fig. 2. Percentages of apps that are subject to vulnerabilities based on code analysis.

"Clear Text Traffic", such as plaintext HTTP, FTP stacks, DownloadManager, and MediaPlayer. These may enable a network attacker to implement man-in-the-middle (MITM) [8] attacks during network transmission.

Figure 2 shows that the most frequent weakness identified by code analysis is the "Risky Cryptography Algorithm". Over 72.5% of apps use at least one deprecated cryptographic algorithm, *e.g.*, MD5 and SHA-1. For instance, in the app MySejahtera (Malaysia), the parameters in `WebSocket` requests are combined and encrypted with MD5. These will be compared with the content from requests in the class `Draft_76` in order to confirm the validity of connections. Although this has been listed in the top 10 OWASP [10] mobile risks 2016, the results show that it is still a common security issue. Another frequent weakness is "Clear Text Storage" (*i.e.*, creation of files that may contain hard-coded sensitive information like usernames, passwords, keys, *etc.*). For example, in the `DataBaseSQL` class of the COVID-19 (Vietnam) app, the SQLite database password is stored in the source code without encryption; CG Covid-19 ePass (India) also hard-coded its encryption key in its `Security` class.

In total, 20 trackers have also been identified, including `Google Firebase Analytics`, `Google CrashLytics`, and `Facebook Analytics`. Approximately 75% of the apps contain at least one tracker. In the most extreme case, one app, Contact Tracing (USA), contains 8 trackers. As shown in Table IV, the most frequent tracker is `Google Firebase Analytics`, which is identified in more than 70% of the apps. Notably, a research study [50] argues that TraceTogether, by using Google's Firebase service to store user information, maybe leaking user information.

Data flow analysis results. Figure 3 presents the potential privacy leakage between sources and sinks. This is counted by the number of source-to-sink paths found in each app. The top sources of PII data are methods calling from `Location` and `database.Cursor`. These may obtain PII from a geographic location sensor or from a database query. Most of the PII data will be transferred to sinks, such as `Bundle`, `Intent`, and `BroadcastReceiver`, which may leak PII out of the app.

As discussed above, sending PII to the `Bundle` object may reveal PII data to other activities. Notably, we also discover that some apps transmit location information through SMS messages. Considering Hamagen

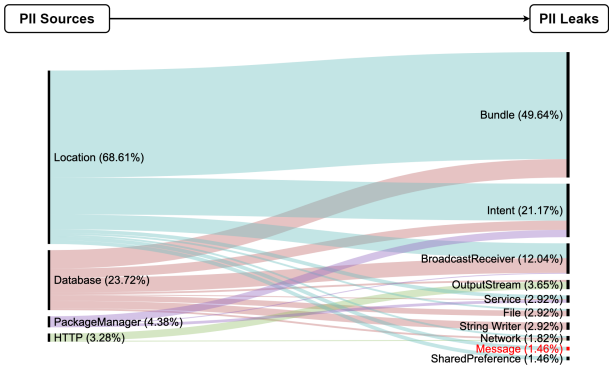

Fig. 3. Privacy leaks detected between sources and sinks. Percentages indicate the fraction of flows originating at the sources (left) and terminating at the sinks (right).

(Israel) as an example, location information is detected and obtained by a source method called `initialize(Context,Location,e)`. This then flows to a sink method where `Handler.sendMessage(Message)` is called. This is a potential vulnerability, as malware could easily intercept the outbox of the Android SMS service [21].

Malware detection results. We discover only one app with malware, Stop COVID-19 KG (Kyrgyzstan) [15]. Two risks are identified: a variant Of `Android/DataCollector.Utilcode.A` and an Adware (0053e0591). This aligns with the finding of COVID-19 apps' threats reported elsewhere [35]. Considering the limited use of Stop COVID-19 KG (roughly 10K downloads), we conclude that the vast majority of contact tracing apps downloaded from Google Play Store are free of malware. That said, the rise of contact tracking apps has also attracted the interest of malicious developers, *e.g.*, a recent report [61] disclosed that new ransomware has targeted the contact tracing app in Canada even before its public release.

Feedback from developers. After alerting all developers of ours findings, we re-checked the apps by regression testing. Table V summarizes the regression testing results. We find that all potential sources of privacy leakage on three apps — TraceTogether (Singapore), BlueZone (Vietnam), STOP COVID19 CAT (Spain) — have been fixed. Additionally, the trackers in MySejahtera (Malaysia) have been removed and the vulnerable app, Contact Tracer (USA), is no longer available.

Meanwhile, new vulnerabilities are identified in the updated versions of several apps (see Table V). For example, COVA Punjab (India) enables popup windows in the WebView setting. STOP COVID19 CAT (Spain) allows clear text traffic in the Manifest, and some apps have more trackers identified. This may mean that the urgency of app development has impacted quality assurance procedures. We note that a study of 493 iOS apps [20] confirms that security issues are prevalent in iOS too. Investigating the root cause of the security issue and their urgency is beyond the scope of our research, yet it is an interesting future research topic.

TABLE V
RESULTS OF REGRESSION TESTING OF APPS.

Applications	Version	Issues Patched
TraceTogether	2.0.15	✓ Disabled Allow Backup; ✓ Fixed potential privacy leakage.
STOP COVID19 CAT	2.0.3	✓ Fixed Insufficient Random issue as they do not use Microsoft package anymore; ✓ Fixed potential privacy leakage; ✗ Allows clear text traffic in manifest; ✗ New tracker detected: Google CrashLytics.
MySejahtera	1.0.19	✓ Fixed the incorrect launch mode of an activity; ✓ Removed three trackers: Google Analytics, Google CrashLytics, and Google Tag Manager.
BlueZone	2.0.2	✓ Fixed potential privacy leakage.
COVA Punjab	1.3.11	✗ New WebView weakness is detected, which could enable popup windows; ✗ New potential privacy leakage path found; ✗ New tracker detected: Google CrashLytics and Google Ads.
Coronavirus UY	4.3.2	✗ New tracker detected: Google CrashLytics.
Contact Tracer	N/A	✓ No longer available in Google Play Store

✓: Fixed, ✗: New vulnerabilities found

Answer to RQ2. The most frequent weaknesses found in contact tracing apps are risky cryptography algorithms use (72.5%), incorrect default permissions (62.5%), clear text storage (55.0%), insecure random values (55%), and allowing backup (42.5%). Meanwhile, trackers and potential privacy leakage also pose risks to users' personal information.

D. Cases and Implications

From our curated list of 40 apps, we select five typical apps around the world to further highlight key lessons we can learn with respect to security and privacy. The cases are TraceTogether, Next Step (DP3T), Private Kit, COVIDSafe, and Corona Warn App.

TraceTogether. According to the COVIDGUARDIAN analysis results, root detection [37] has been implemented in TraceTogether. This potentially prevents SQL injection and data breaches, thereby reducing the risk to a certain extent. For example, in `o/C3271ax.java`, root detection logic is implemented by detecting the existence of specific root files in the system, *e.g.*, `/system/app/Superuser.apk` and `/system/xbin/su`. By assessing their integrity, the app can detect whether a device is rooted and subsequently block users from either logging in or opening it.

However, TraceTogether also includes a third-party customer feedback library, ZENDESK SDK, in which remote WebView debugging is enabled. This potentially allows attackers to dump the content in the WebView [25]. When a user inputs confidential data, including passwords, in a debug-enabled WebView, attackers may be able to inspect all elements in the webpage [45]. Fortunately, as per the static analysis, the only WebView with debugging mode enabled is to display articles; therefore, it does not contain confidential data.

Security guideline 1: Never leave WebView with debugging mode enabled in the app release.

Next Step (DP3T). Next Step's database is not encrypted, and data is saved in plain text. In contrast to TraceTogether, the app does not implement any root detection capabilities. Although, the leakage of user's information via a rooted device may not be remotely exploitable, local malware may be able to gain root access. We therefore argue that root detection is necessary due to the large-scale deployment of such apps (over 60% of the population [34]), the long duration of their operation, and the fact that 7.6% of Android users already have rooted their devices [72]. Further, in a rooted device, a malicious app could possibly access the database and manipulate COVID-19 contact records.

Security guideline 2: To protect the database from being dumped and prevent data breaches, a solution should:
1) Implement database encryption [17]
2) Enable root detection [3] and confidential data protection [16] at app startup.

In addition, as the database records timestamps and contact IDs, the leakage of the database from a rooted device could be exploited to mount linkage attacks by adversaries [11]. Therefore, if enough data in a region were collected, contact IDs and timestamps could be used to analyze movements [65].

Private Kit. Similar to Next Step (DP3T), Private Kit does not encrypt the database and contains plaintext data. Additionally, the app creates temporary JSON files to store users' location data. Without any encryption and root detection, the temporary JSON files could be dumped from a rooted device.

Security guideline 3: To prevent potential data breaches, confidential data must not be stored in temporary files in plain text.

COVIDSafe. COVIDSafe 1.0.11 stores all tracing histories (including contacted device IDs and timestamps) into an SQLite database using plain text. Since the app does not implement root detection logic, tracing histories may be leaked from root devices and therefore potential linkage attacks could be implemented [65]. However, in the latest version, COVIDSafe fixed this issue by encrypting the local database with a public key.

Corona Warn App. We find two security features worth highlighting. Corona Warn App applies the SQLCIPHER [14] framework, which enhances the SQLITE database by making it more suitable for encrypted local data storage. It also introduces the CONSCRYPT [5] framework, which uses BoringSSL to provide cryptographic primitives and Transport Layer Security for Java applications on Android, during data transmission.

Security guideline 4: To protect local databases, introducing professional security frameworks is useful, *e.g.* CONSCRYPT and SQLCIPHER are open-source, well-documented, frequently-updated, widely-used, and their code is frequently reviewed.

V. RQ3: Reviewing and Assessing Security and Privacy Threats

In this section, we review the major security and privacy threats facing contact tracing apps, based on our prior analysis.

A. User Privacy Exposure

We envisage three groups of contact tracing app users, based on their health status:

- **Generic user.** A typical user of the contact tracing system, who is healthy or has not been diagnosed yet.
- **At-risk user.** A user who has recently been in contact with an infected user. Ideally, an at-risk user will receive an at-risk alarm from the app.
- **Diagnosed user.** A diagnosed patient who will be asked to reveal their information as well as the information of at-risk users to the health authorities, *e.g.*, the diagnosis of their infection, their movement history, the persons they have been in contact with.

Based on our prior analysis, we further define five broad categories of apps:

- **Level I** ▪: "No data is shared with a server or users", the most secure level.
- **Level II** ▪ ▪: "Tokens are shared with proximity users", a medium exposure level with only tokens containing no PII being exchanged between users.
- **Level III** ▪ ▪ ▪: "Tokens are shared with the server", a medium exposure level with tokens exposed to the server.
- **Level IV** ▪ ▪ ▪ ▪: "PII is shared with a server", a high risk exposure level.
- **Level V** ▪ ▪ ▪ ▪ ▪: "PII is released to public", the highest risk exposure level.

Based on the in-app instructions, documents provided in apps' official websites, and the privacy policies, we assess user privacy exposure and threats posed by the 40 apps listed in Table I. We translated to English the materials not written in English with Google Translate. For 13 out of the 40 apps, we were not able to collect adequate information to conduct the privacy assessment. *Significantly, 13 out of the 40 apps lacking transparent documentation were also identified as the ones with higher than the average number of security and privacy risks using* COVIDGUARDIAN. *Considering the lack of transparency of such apps with more than two million downloads, our findings demonstrate the pressing need for a holistic security and privacy assessment tool.*

As summarized in Table VI, all 27 assessed apps have user privacy exposure to some extent. We determine user exposure level as IV in some centralized apps, such as COVIDSafe, TraceTogether, and apps from #18 to #27. This is primarily because the central servers request users' PII (*e.g.*, name, phone number, postcode, or even location information) during registration or execution. This could be fixed by better privacy by design, exemplified by #4 StopCovid France, which does not collect such PII. Meanwhile, most decentralized Bluetooth-based apps are categorized as a lower-level exposure. This is because they utilize non-identifiable tokens, instead of directly using PII in contact tracing. However, we determine the privacy exposure level of diagnosed users in Hamagen as Level V, as their location information could be accessed by third-party users, allowing the diagnosed users to be potentially re-identified.

TABLE VI
USER PRIVACY EXPOSURE AND THREATS TO APPS.

#	Apps (27 of 40 apps assessed; 13 apps do not provide adequate information)	User Privacy Exposure: Generic	User Privacy Exposure: At-risk	User Privacy Exposure: Diagnosed	Threats: Linkage-Server	Threats: Linkage-User	Threats: False-Claim	Threats: Relay Attack	Architecture
1	CovidSafe	▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	●	○	●	C
2	HaMagen	▪	▪	▪ ▪ ▪ ▪ ▪	●	●	○	○	D
3	TraceTogether	▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	●	○	●	C
4	StopCovid France	▪ ▪	▪ ▪ ▪	▪ ▪ ▪	●	●	○	●	C
5	Next Step (DP3T)	▪ ▪	▪ ▪ ▪	▪ ▪ ▪	○	●	(-)	●	D
6	Corona Warn App	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	●	D
7	NHS Test and Tracing App	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	●	D
8	TraceCorona	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	●	D
9	Private Kit	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	●	D
10	MySejahtera	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	○	○	C
11	Smittestop	▪ ▪	▪ ▪ ▪	▪ ▪ ▪	○	●	○	●	C
12	COVID Alert	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	○	D
13	SwissCovid	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	●	D
14	Bluezone	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	●	D
15	COCOA	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	○	●	D
16	Immuni	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	(-)	●	D
17	Stopp Corona	▪ ▪	▪ ▪	▪ ▪ ▪	○	●	●	●	D
18	Aarogya Setu	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	●	●	●	C
19	EHTERAZ	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	○	○	C
20	Vietnam Health Declaration	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	●	○	C
21	STOP COVID19 CAT	▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	●	○	C
22	CG Covid-19 ePass	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	○	○	C
23	StopTheSpread COVID-19	▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	●	○	C
24	Stop COVID-19 KG	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	(-)	○	C
25	BeAware Bahrain	▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	●	●	○	C
26	Nepal COVID-19 Surveillance	▪	▪ ▪ ▪ ▪	▪ ▪ ▪ ▪	●	○	●	○	C
27	Stop Covid	▪ ▪	▪	▪ ▪ ▪ ▪	●	●	●	●	C

○: the system is well protected ●: the system is at-risk C: Centralized D: Decentralized
(-): Inadequate information to conduct an assessment.

B. Security and Privacy Threats

To further examine security and privacy threats against contact tracing apps, we systematically search and review the state-of-the-art [23, 27, 36, 73]. We use Google Search, Google Scholar, and Twitter to discover research papers and Twitter comments or media outlets referring to published or arXiv papers. To achieve this, we use a set of associated search keywords (*e.g.*, security and privacy attacks, COVID-19 contact tracing apps, vulnerabilities, effectiveness, safety). To expand this set, we further examine papers in the bibliographies of each keyword-filtered paper. We then manually exclude irrelevant topics and synthesize four dominant security and privacy threats: (*i*) server link attacks; (*ii*) user link attacks; (*iii*) false positive claims; and (*iv*) relay attacks. We summarize the nature of these threats in Figure 4 and discuss them in turn below.

Linkage attacks by servers. In centralized systems, the major privacy concern is metadata leakage by the server. Consequently, Grace will be able to collect a large amount of PII, such as names, phone numbers, contact lists, post code, home addresses, location trails. Therefore Grace is able to

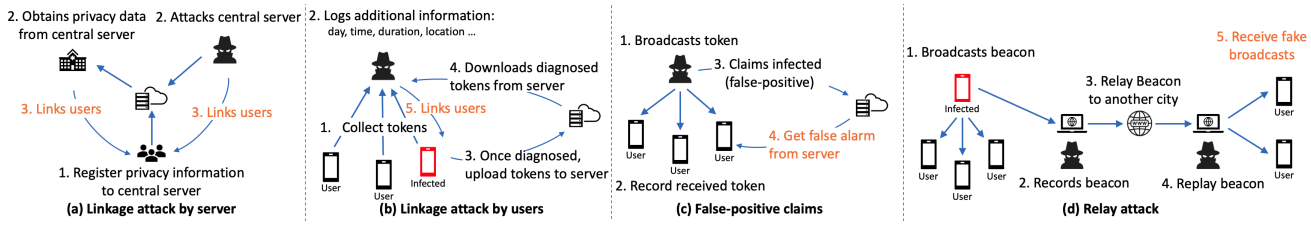

Fig. 4. Four types of potential security and privacy threats: (a) **linkage attacks by a server:** in centralized systems, a key privacy concern is metadata leakage by the server; (b) **linkage attacks by users:** in systems based on information exchange between users, attackers could re-identify users using data published or received from users; (c) **false positive claim:** attackers can incorrectly register as infected, which will generate false at-risk alerts; (d) **relay attacks:** in Bluetooth based apps, a malicious attacker can potentially redirect/replicate Bluetooth broadcasts from one place to another, thereby generating false at-risk alerts.

deduce the social connections of Alice. However, in some decentralized systems (*e.g.*, Hamagen), the server also learns users' PII, which can compromise such such decentralized systems too (as marked in Table VI).

Privacy guideline 1: To protect users' privacy against linkage attacks by a server, a contact tracing app should:
1) Avoid sharing PII with central points or
2) Implement a decentralized design.

Linkage attacks by users. Linkage attacks performed by users (Mallory), try to re-identify Alice or Bob. In contact tracing systems that directly publish users' PII, Bob is obviously at risk of privacy leakage. For other apps listed in Table VI that rely on information exchange between users (*e.g.*, DP3T, which implements an ephemeral ID design), Mallory is still able to identify Bob using more advanced attacks. For example, if Mallory places a Bluetooth receiver near Bob's home or working place and ensures that the device will only receive Bluetooth broadcasts from Bob. Once Bob is diagnosed, Mallory will receive an at-risk alarm and immediately acknowledge that the infected patient is Bob. In addition, Mallory can log the timestamp and the received ephemeral ID when in contact with Bob, which could be done by modifying the app or developing a customised app using the open-sourced contact tracing framework. Once Bob is diagnosed, Mallory will be able to trace back the source of recording and re-identify Bob and potentially infected users. Similar attacks were described as *Paparazzi Attacks* and *Nerd Attacks* in [73]. Even worse, if Mallory distributes multiple broadcast receivers in a large area they could potentially trace the movement of Bob by tracing the records on each device.

Privacy guideline 2: To protect users' privacy against linkage attacks by an adversary, a solution should:
1) Avoid data sharing between users or
2) Ensure privacy protections exist for any published data.

False positive claims. In some systems, Bob can register as infected via the app. If Mallory exploits such a mechanism and registers as a (fake) infected user, Alice will receive a false-positive at-risk alarm, which may cause social panic or negatively impact evidence-driven public health policies. Most solutions mitigate this issue by implementing an authorization process, *i.e.*, a one-time-use permission code.

Privacy guideline 3: To protect a system against false-positive-claim attacks, a solution should establish an authorization process.

Relay attacks. To apply a relay attack, Mallory could collect existing broadcast messages exchanged between users, then replay them at another time, or forward them through proxy devices to a remote location. Due to the lack of message validation in solutions that utilize information broadcasts, a user will not be able to determine whether a received broadcast is from a valid source or from a malicious device. Thus, any received broadcast will be recorded as a contact event. A malicious attacker can potentially redirect all the traffic from one place to another, resulting in a targeted area being incorrectly locked-down.

Privacy guideline 4: To protect a system against relay attacks, a solution should:
1) Either avoid utilizing information broadcast or
2) Implement a validation approach.

Summary. The linkage attacks by a server are the overarching threats to centralized systems, as third-party attackers (or the server itself) are able to re-identify users (if PII is exposed to the central server). Although StopCovid France is robust to such a threat, it is still susceptible to re-identification risks since the information from the server enables linkage between anonymous IDs and the corresponding permanent app identifier. This permits the tracing of users based on IDs observed in the past, as well as future movements.

Additionally, apps that use Bluetooth broadcasts are exposed to linkage attacks by users. We note that, although the false positive claims could be mitigated by authorization to allow only positive users to upload diagnosed data to the server, there are still some apps failing to implement essential authorization. Furthermore, the relay attack is another threat causing false positives in Bluetooth-based apps. Notably, although TraceCORONA is rated as at-risk against relay attacks, its specific design provides protection against one-way-relaying unlike other apps.

Answer to RQ3. We manually rate the level of user privacy exposure for each contact tracing app. We further categorize the robustness of apps against security and privacy threats. We ascertain that the apps in question are vulnerable to at least one of the four threats. Not all decentralized architectures are necessarily more secure than those adopting centralized architectures, *e.g.*, Hamagen, a decentralized location-based system.

VI. RQ4: User Study

In this section, we present a survey, exploring the user perceptions of contact tracing apps. Our objective is to query the *likelihood* and *concerns* associated with using the apps. Through the study we aim to ascertain user concerns and requirements of contact tracing apps (*RQ4*).

A. Participant Recruitment

The proportion of youth aged 15-24 years with COVID-19 has increased six-fold from 24 February through 12 July 2020 [77]. It was reported by Reuters [22] that young people who are visiting nightclubs and beaches are causing a rise in fresh cases with potential consequences for more vulnerable age groups. Therefore, we focus our user study on 18-29 year olds. We have recruited 373 volunteers and asked them standard demographic questions, *i.e.*, age range, education, gender, and nationality. All participants reside in Australia but have various cultural backgrounds (58% from Oceania, 20% from Asia, and 14% from other geographic regions; 30 participants did not indicate their nationalities). 39% of participants are male (59% female). One participant identified themselves as other gender and six participants refrained. 67% of participants are university graduates and 30% are high school graduates.

B. Survey Protocol

To calibrate the scope of our survey without introducing bias, we provide vignettes describing user privacy exposure levels using the security and privacy threats reviewed in Section V.

All questions, excluding demographics, are five-point Likert scale [53] questions. The Likert scale is a quantitative, fine-grained, and user-friendly method of collecting data from users. The scales ask participants to indicate how much they are likely (1) or unlikely (5), to be unconcerned (1) or concerned (5) about using contact tracing apps. The survey adopts a bespoke model and integrates questions from related survey instruments used by Simko *et al.* [66] and Kaptchuk *et al.* [44]. We provide the survey design and questionnaire in our open source website and summarize the two item categories below.

(a) Likelihood of using contact tracing apps. To investigate the usability requirements of contact tracing apps, we asked participants their likelihood of using contact tracing apps under different scenarios: (*i*) functionality scenarios, *i.e.*, if the accuracy of proximity contact detection and at-risk alarm generation is not perfect (false positives or false negatives exist); and (*ii*) privacy scenarios, *i.e.*, if personal data (location, phone number, postcode, or anonymous identifier) will be shared with other entities (other users or governments), or if the authorities require them to provide data (location, proximity data, or anonymous tokens) after being diagnosed.

Notably, *to ensure the survey is designed for both users and non-users, we did not explicitly mention any specific contact tracing app*, but still covered all contextual privacy concepts of state-of-the-practice contact tracing apps discussed in Section II. For example, COVIDSafe is covered by scenarios in which a "phone number and postcode are shared with governments or health authorities" and if the app will "upload proximity data if tested positive". Similarly, the Corona Warn App or others that implement the Google and Apple (Gapple) framework are covered by scenarios where "anonymous identifiers are shared with other users" and the app will "upload anonymous tokens if being tested positive".

(b) Concerns about use of contact tracing apps. We also asked the participants about their concerns regarding contact tracing apps. We focused mainly on three aspects: (*i*) the usability of contact tracing apps, including battery drain, storage drain, and ease-of-use; (*ii*) the effectiveness of contact tracing apps, *i.e.*, to what extent users are concerned about the accuracy of contact tracing; and (*iii*) the concerns about privacy. We conducted the survey through both pencil-and-paper and SoGoSurvey [13]. 3.4% (15) of participants used the paper version of the survey and there is no significant impact on the study results. During the paper survey, we did not receive any queries from participants.

C. Data Analysis and Results

Our variables are ordinal and the responses to each question are not expected to be normally distributed. Therefore, we use a Mann-Whitney U-test [56] for statistical significance testing to ascertain the key factors that impact user concerns and requirements (or expectations). Concretely, we aggregate item responses to assess the participants' likelihood of using contact tracing apps across *four* different privacy-preserving and data sharing scenarios (as described in the Section V-B).

We calculate the U-value and p-value of each pair of contact tracing solutions among centralized solutions with PII collected (Type A), centralized solutions with non-PII collected (Type B), decentralized solutions with PII collected (Type C), and decentralized solutions with non-PII collected (Type D). If the p-value is smaller than 0.05, we reject the null hypothesis that users are equally likely to use any of the two contact tracing apps. We use SciPy.stats for our analysis.

(a) The accuracy of contact tracing impacts the likelihood of app use. Users are sensitive to sharing PII in a decentralized system. As shown in Figure 5(a), if a contact tracing app can accurately detect proximity and notify users who may be at-risk, more than 60% of participants are likely to use it. However, in reality, it is hard to eliminate false-positives and false-negatives. When the tracing accuracy concern is considered, the proportion of users likely to use contact tracing apps drops to 31% and 26%, respectively. Our results align

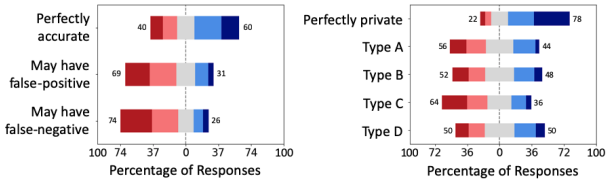

Fig. 5. (a) Participants' likelihood of using contact tracing apps *vs.* the accuracy of proximity detection; (b) Participants' likelihood of using apps across different privacy-preserving and data sharing scenarios. : 5 = extremely unlikely, : 1 = extremely likely.

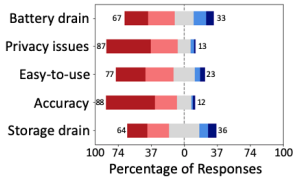

Fig. 6. Participants' concerns about contact tracing apps. : 5 = extremely concerned, : 1 = extremely unconcerned.

TABLE VII
MANN-WHITNEY RESULTS ON THE LIKELIHOOD OF USING CONTACT TRACING APPS.

Solution 1	Solution 2	U-value	p-value
Type A	Type B	66020.0	0.1072
Type A	Type C	60756.0	<0.01**
Type A	Type D	62714.0	0.0819
Type B	Type C	57646.0	<0.001***
Type B	Type D	66364.5	0.131
Type C	Type D	54472.5	<0.001***

Type A: Centralized system with PII collection
Type B: Centralized system without PII collection
Type C: Decentralized system with PII collection
Type D: Decentralized system without PII collection

well with a survey of COVID-19 app usage taken in the USA [44].

Figure 5(b), shows how likely users are to use a contact tracing app in different scenarios. Here, 50% of participants listed positive responses to decentralized apps with non-PII identification collected and shared (Type D). This resonates with the fact that Bluetooth-based decentralized systems preserve user privacy more than centralized systems. However, decentralized apps with PII collected (Type C) are not as popular with users. More than 64% of participants said that they are unlikely to use such an app. It can be seen that even this young cohort of users is sensitive to sharing PII.

Table VII presents the Mann-Whitney U-tests comparing pairs of contact tracing apps. Our results indicate that the differences between the likelihood of using decentralized apps with PII collected (Type C) and other apps is statistically significant. This is perhaps due to the privacy design in a decentralized PII-collected system. That is, diagnosed users' information (*e.g.*, location information) will be collected and shared with other users, while the other types of apps do not share such information between users. When compared with the other three types, we had p-values greater than 0.05. Hence, despite the different levels of privacy protection, there is no statistically significant difference in the likelihood of users using these apps. However, from Figure 5(b), we can infer that it is more likely for users to accept and use the decentralized apps without PII collection (Type D).

Additionally, as evidenced in Figure 5, we find that more than 78% of participants are likely to use perfectly private contact tracing apps. This indicates a much higher likelihood than those who prefer a perfectly accurate contact tracing app (60%). The difference is statistically significant (p-value $<$ 0.0001), indicating that users are more likely to accept and use contact tracing apps that satisfy privacy protection requirements.

(b) User concerns focus on privacy and tracing accuracy. As shown in Figure 6, more than 55% of participants are extremely concerned about the tracing accuracy of apps, and more than 49% of participants are extremely concerned about privacy issues.

D. Threats to Validity of Our User Study

External validity. The participants of the survey were all from one country, and the duration of the survey was three weeks. Further, our survey may be subject to volunteer bias and non-response bias [26] (*i.e.*, participants and their responses may have different characteristics from the general population of interest). To mitigate the volunteer bias, we designed the survey questionnaire to be completed in 10-15 minutes and anonymized the responses.

Internal validity. Considering the nature of the user study, participants may find that some questions are confusing or unclear. This will lead to incorrect or inconsistent responses. To mitigate this threat, we conducted a pilot survey with 45 computer science students and updated the questions based on feedback. In addition, as the responses are anonymous, there could be participants who took the survey multiple times.

Answer to RQ4. Privacy design and tracing accuracy impact the likelihood of app use. Furthermore, compared to users' expectations of tracing accuracy, users are more likely to accept and use apps with better privacy by design. Interestingly, if PII data is collected, users prefer a centralized solution in contrast to a decentralized solution that collects PII data.

VII. CONCLUSION

This study has developed a security and privacy assessment tool, COVIDGUARDIAN. This tool can evaluate the security weaknesses, vulnerabilities, potential privacy leaks, and malware in contact tracing apps. Using COVIDGUARDIAN, we have conducted a comprehensive empirical security and privacy assessment of 40 contact tracing apps. Our results have identified multiple security and privacy risks, as well as threats. Naturally, our analysis has confirmed that no apps can protect users' security and privacy against *all* potential threats. To understand the perception of users, we have also performed a survey involving 373 participants. This has further consolidated our observations of user concerns. In the future, we plan to extend our study to obtain user feedback from a wider geographic and demographic range. Examining network traffic originating from contact tracing apps is also worth further exploration.

ACKNOWLEDGMENTS

We would like to thank the reviewers for their valuable feedback. This work was supported in part by the Australian Research Council (ARC) Discovery Project (DP210102670) and the EPSRC (EP/S033564/1).

REFERENCES

[1] "AndroBugs." [Online]. Available: https://github.com/AndroBugs/AndroBugs_Framework

[2] "Android basic security testing." [Online]. Available: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05b-Basic-Security_Testing.md

[3] "Android root detection techniques." [Online]. Available: https://blog.netspi.com/android-root-detection-techniques/

[4] "Apple and Google partner on COVID-19 contact tracing technology." [Online]. Available: https://www.apple.com/au/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/

[5] "conscrypt." [Online]. Available: https://github.com/google/conscrypt

[6] "CWE-330: Use of insufficiently random values." [Online]. Available: https://cwe.mitre.org/data/definitions/330.html

[7] "Jadx." [Online]. Available: https://github.com/skylot/jadx

[8] "Man-in-the-middle attack." [Online]. Available: https://en.wikipedia.org/wiki/Man-in-the-middle_attack

[9] "Mobile-security-framework-mobsf." [Online]. Available: https://github.com/MobSF/Mobile-Security-Framework-MobSF

[10] "OWASP." [Online]. Available: https://owasp.org/www-project-mobile-top-10/

[11] "Protecting against linkage attacks that use 'anonymous data'." [Online]. Available: https://www.marklogic.com/blog/protecting-linkage-attacks-use-anonymous-data/

[12] "Qark." [Online]. Available: https://github.com/linkedin/qark

[13] "SoGoSurvey." [Online]. Available: https://www.sogosurvey.com

[14] "Sqlcipher." [Online]. Available: https://www.zetetic.net/sqlcipher/

[15] "Stop COVID-19 KG." [Online]. Available: https://play.google.com/store/apps/details?id=kg.cdt.stopcovid19

[16] "Support direct boot mode." [Online]. Available: https://developer.android.com/training/articles/direct-boot

[17] "Using the SQLite encryption extension." [Online]. Available: https://sqlite.org/android/doc/trunk/www/see.wiki

[18] "Virustotal." [Online]. Available: https://www.virustotal.com/

[19] "Coronavirus Desease-19," 2020. [Online]. Available: http://ncov.mohw.go.kr,2020,accessed:2020-03-23

[20] J. Albright, "The pandemic app ecosystem: Investigating 493 covid-related iOS apps across 98 countries," 2020. [Online]. Available: https://d1gi.medium.com/the-pandemic-app-ecosystem-investigating-493-covid-related-ios-apps-across-98-countries-cdca305b99da

[21] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps," *Acm Sigplan Notices*, vol. 49, no. 6, pp. 259–269, 2014.

[22] A. Banerjee and S. Nebehay, "Proportion of youth with COVID-19 triples in five months: WHO," 2020. [Online]. Available: https://www.reuters.com/article/us-health-coronavirus-youth/proportion-of-youth-with-covid-19-triples-in-five-months-who-idUSKCN2502FS,accessed:2020-08-20

[23] L. Baumgärtner, A. Dmitrienko, B. Freisleben, A. Gruler, J. Höchst, J. Kühlberg, M. Mezini, M. Miettinen, A. Muhamedagic, T. D. Nguyen *et al.*, "Mind the gap: Security & privacy risks of contact tracing apps," *arXiv preprint arXiv:2006.05914*, 2020.

[24] J. Bay, J. Kek, A. Tan, C. S. Hau, L. Yongquan, J. Tan, and T. A. Quy, "Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders," *Government Technology Agency-Singapore, Tech. Rep*, 2020.

[25] A. Bhatia, "Android security: Don't leave WebView debugging enabled in production," 2019. [Online]. Available: https://dev.to/ashishb/android-security-don-t-leave-webview-debugging-enabled-in-production-5fo9

[26] J. Brassey, K. Mahtani, E. Spencer, and C. Heneghan, "Volunteer bias," *Catalogue Of Bias*, 2017.

[27] J. Chan, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine *et al.*, "PACT: privacy sensitive protocols and mechanisms for mobile contact tracing," *arXiv preprint arXiv:2004.03544*, 2020.

[28] B. Chappell, "Coronavirus: sacramento county gives up on automatic 14-day quarantines," 2020. [Online]. Available: https://www.npr.org/sections/health-shots/2020/03/10/813990993/coronavirus-sacramento-county-gives-up-on-automatic-14-day-quarantines,2020,accessed:2020-03-23

[29] S. Chen, L. Fan, G. Meng, T. Su, M. Xue, Y. Xue, Y. Liu, and L. Xu, "An empirical assessment of security risks of global Android banking apps," in *Proceedings of the 42st International Conference on Software Engineering. IEEE Press*, 2020, pp. 596–607.

[30] S. Chen, T. Su, L. Fan, G. Meng, M. Xue, Y. Liu, and L. Xu, "Are mobile banking apps secure? what can be improved?" in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 797–802.

[31] H. Cho, D. Ippolito, and Y. W. Yu, "Contact tracing mobile apps for COVID-19: Privacy considerations and related trade-offs," *arXiv preprint arXiv:2003.11511*, 2020.

[32] C. Culnane and K. Leins, "Misconceptions in privacy protection and regulation," *Law in Context. A Socio-legal Journal*, vol. 36, no. 2, pp. 1–12, 2019.

[33] A. Department of Health, "COVIDSafe," 2020. [Online]. Available: https://www.health.gov.au/resources/apps-and-tools/covidsafe-app,2020,accessed:2020-04-23

[34] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser, "Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing," *Science*, 2020.

[35] T. Gould, G. Mele, P. Rajendran, and R. Gould, "Anomali threat research identifies fake COVID-19 contact tracing apps used to download malware that monitors devices, steals personal data." [Online]. Available: https://www.anomali.com/blog/anomali-threat-research-identifies-fake-covid-19-contact-tracing-apps-used-to-monitor-devices-steal-personal-data

[36] Y. Gvili, "Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc," Cryptology ePrint Archive, Report 2020/428, Tech. Rep., 2020.

[37] Z. Q. Hang Zhang, Dongdong She, "Android root and its providers: A double-edged sword," 2015. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2810103.2813714

[38] R. He, H. Wang, P. Xia, L. Wang, Y. Li, L. Wu, Y. Zhou, X. Luo, Y. Guo, and G. Xu, "Beyond the virus: A first look at Coronavirus-themed mobile malware," *arXiv preprint arXiv:2005.14619*, 2020.

[39] Y. Hu, H. Wang, L. Li, Y. Guo, G. Xu, and R. He, "Want to earn a few extra bucks? a first look at money-making apps," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2019, pp. 332–343.

[40] J. Huang, Z. Li, X. Xiao, Z. Wu, K. Lu, X. Zhang, and G. Jiang, "{SUPOR}: Precise and scalable sensitive user input detection for android apps," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 977–992.

[41] M. Ikram, R. Masood, G. Tyson, M. A. Kaafar, N. Loizon, and R. Ensafi, "The chain of implicit trust: An analysis of the web third-party resources loading," in *The World Wide Web Conference*, 2019, pp. 2851–2857.

[42] A. Inc and G. Inc, "Exposure notification-bluetooth specification." [Online]. Available: https://www.blog.googl

e/documents/58/Contact_Tracing_-_Bluetooth_Specification_v1.1_RYGZbKW.pdf
[43] ——, "Exposure notification-cryptography specification." [Online]. Available: https://www.blog.google/documents/56/Contact_Tracing_-_Cryptography_Specification.pdf
[44] G. Kaptchuk, D. G. Goldstein, E. Hargittai, J. Hofman, and E. M. Redmiles, "How good is good enough for covid19 apps," *The influence of benefits, accuracy, and privacy on willingness to adopt.*, 2020.
[45] M. Kearney, "Remote debugging webviews." [Online]. Available: https://developers.google.com/web/tools/chrome-devtools/remote-debugging/webviews
[46] M. J. Kim and S. Denyer, "A 'travel log' of the times in South Korea: Mapping the movements of coronavirus carriers." [Online]. Available: https://www.washingtonpost.com/world/asia_pacific/coronavirus-south-korea-tracking-apps/2020/03/13/2bed568e-5fac-11ea-ac50-18701e14e06d_story.html
[47] R. Koch-Institut, "Corona Warn App," 2020. [Online]. Available: https://www.coronawarn.app/en/,2020,accessed:2020-08-17
[48] S. Latif, M. Usman, S. Manzoor, W. Iqbal, J. Qadir, G. Tyson, I. Castro, A. Razi, M. N. K. Boulos, A. Weller, and J. Crowcroft, "Leveraging Data Science To Combat COVID-19: A Comprehensive Review," *IEE Transactions on Artificial Intelligence*, 2020.
[49] K. Leins, C. Culnane, and B. I. Rubinstein, "Tracking, tracing, trust: Contemplating mitigating the impact of COVID-19 through technological interventions," *The Medical Journal of Australia*, p. 1, 2020.
[50] D. J. Leith and S. Farrell, "Coronavirus contact tracing app privacy: What data is shared by the Singapore OpenTrace app?" 2020.
[51] J. Li and X. Guo, "COVID-19 contact-tracing apps: A survey on the global deployment and challenges," *arXiv preprint arXiv:2005.03599*, 2020.
[52] L. Li, A. Bartel, T. F. Bissyandé, J. Klein, Y. Le Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, and P. McDaniel, "Iccta: Detecting inter-component privacy leaks in android apps," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 280–291.
[53] R. Likert, "A technique for the measurement of attitudes." *Archives of psychology*, 1932.
[54] J. K. Liu, M. H. Au, T. H. Yuen, C. Zuo, J. Wang, A. Sakzad, X. Luo, and L. Li, "Privacy-preserving COVID-19 contact tracing app: a zero-knowledge proof approach."
[55] T. Liu, H. Wang, L. Li, X. Luo, F. Dong, Y. Guo, L. Wang, T. Bissyandé, and J. Klein, "MadDroid: Characterizing and detecting devious Ad contents for Android apps," in *Proceedings of The Web Conference 2020*, 2020, pp. 1715–1726.
[56] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.
[57] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[58] I. Ministry of Health, "Hamagen," 2020. [Online]. Available: https://govextra.gov.il/ministry-of-health/hamagen-app/,2020,accessed:2020-04-23
[59] D. of Health and S. Care, "Next phase of NHS coronavirus (COVID-19) app announced," 2020. [Online]. Available: https://www.gov.uk/government/news/next-phase-of-nhs-coronavirus-covid-19-app-announced,2020,accessed:2020-08-17
[60] W. H. Organization, "Operational considerations for case management of COVID-19 in health facility and community," 2020. [Online]. Available: https://apps.who.int/iris/bitstream/handle/10665/331492/WHO-2019-nCoV-HCF_operations-2020.1-eng.pdf
[61] C. Osborne, "New ransomware masquerades as COVID-19 contact-tracing app on your android device." [Online]. Available: https://www.zdnet.com/article/new-crycryptor-ransomware-masquerades-as-covid-19-contact-tracing-app-on-your-device/
[62] R. Z. Paul Mozur and A. Krolik, "In Coronavirus fight, China gives citizens a color code, with red flags," 2020. [Online]. Available: https://www.nytimes.com/2020/03/01/business/china-coronavirus-surveillance.html,2020,accessed:2020-05-07
[63] J. Reardon, Á. Feal, P. Wijesekera, A. E. B. On, N. Vallina-Rodriguez, and S. Egelman, "50 ways to leak your data: An exploration of apps' circumvention of the android permissions system," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 603–620.
[64] J. H. Reelfs, O. Hohlfeld, and I. Poese, "Corona-Warn-App: Tracing the start of the official COVID-19 Exposure Notification App for germany," *arXiv preprint arXiv:2008.07370*, 2020.
[65] Y. Shen, F. Wang, and H. Jin, "Defending against user identity linkage attack across multiple online social networks," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 375–376.
[66] L. Simko, R. Calo, F. Roesner, and T. Kohno, "Covid-19 contact tracing and privacy: Studying opinion and preferences," *arXiv preprint arXiv:2005.06056*, 2020.
[67] R. Sun, W. Wang, M. Xue, G. Tyson, and D. C. Ranasinghe, "VenueTrace: a privacy-by-design COVID-19 digital contact tracing solution," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 790–791.
[68] Z. Tang, K. Tang, M. Xue, Y. Tian, S. Chen, M. Ikram, T. Wang, and H. Zhu, "iOS, your OS, everybody's OS: Vetting and analyzing network services of iOS applications," in *29th USENIX Security Symposium*, 2020, pp. 2415–2432.
[69] P. team and F. AISEC, "ROBERT: Robust and privacy-preserving proximity tracing." [Online]. Available: https://github.com/ROBERT-proximity-tracing/documents/blob/master/ROBERT-specification-EN-v1_0.pdf
[70] T. P.-P. team, "PEPP-PT high level overview," 2020. [Online]. Available: https://github.com/pepp-pt/pepp-pt-documentation/blob/master/PEPP-PT-high-level-overview.pdf
[71] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli *et al.*, "Decentralized privacy-preserving proximity tracing," *arXiv preprint arXiv:2005.12273*, 2020.
[72] R. Unuchek, "Rooting your Android: Advantages, disadvantages, and snags," 2017. [Online]. Available: https://www.kaspersky.com/blog/android-root-faq/17135/
[73] S. Vaudenay, "Analysis of DP3T between scylla and charybdis," 2020. [Online]. Available: https://eprint.iacr.org/2020/399.pdf
[74] H. Wang, L. Wang, and H. Wang, "Market-level analysis of government-backed covid-19 contact tracing apps," *arXiv preprint arXiv:2012.10866*, 2020.
[75] W. Wang, R. Sun, M. Xue, and D. C. Ranasinghe, "An automated assessment of Android clipboards," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020.
[76] H. Wen, Q. Zhao, Z. Lin, D. Xuan, and N. Shroff, "A study of the privacy of COVID-19 contact tracing apps," *International Conference on Security and Privacy for Communication Networks*, 2020.
[77] WHO, "Coronavirus disease situation report - 198," 2020. [Online]. Available: https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200805-covid-19-sitrep-198.pdf?sfvrsn=f99d1754_2,2020,accessed:2020-08-20
[78] M. Xue, C. Ballard, K. Liu, C. Nemelka, Y. Wu, K. Ross, and H. Qian, "You can yak but you can't hide: Localizing anonymous social network users," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 25–31.