THE UNIVERSITY of ADELAIDE

Understanding and Measuring Privacy and Security Assertions of Mobile and VR Applications

Ruoxi Sun

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

June 2023

Contents

List of Figures

List of Tables

University of Adelaide

Abstract

Understanding and Measuring Privacy and Security Assertions of Mobile and VR Applications

by Ruoxi Sun

The emergence of the COVID-19 pandemic has catalysed a profound transformation in the way mobile applications are utilised and engaged with by consumers. There has been a noticeable surge in people relying on applications for various purposes such as entertainment, remote work, and daily activities. These services collect large amounts of users' personal information and use them in many areas, such as in medical and financial systems, but they also pose an unprecedented threat to users' privacy and security.

Many international jurisdictions have enacted privacy laws and regulations to restrict the behaviour of apps and define the obligations of app developers. Although various privacy assertions are required in app stores, such as the permission list and the privacy policies, it is usually difficult for regular users to understand the potential threats the app may pose, let alone identify undesired or malicious application behaviours.

In this thesis, I have developed a comprehensive framework to assess the current privacy practices of mobile applications. The framework first establishes a knowledge base (including datasets) to model privacy and security assertions. It then builds a sound evaluation system to analyse the privacy practices of mobile applications. Large-scale privacy evaluations were conducted on different real-world datasets, including privacy policies, contact tracing apps, and children's apps, with the aim of revealing the risks associated with mobile application privacy. Lastly, a novel approach to applying differential privacy on streamed spatial data in VR applications is proposed. This thesis provides a comprehensive guideline for the mobile software industry and legislators to build a stronger and safer privacy ecosystem.

Declaration of Authorship

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Ruoxi Sun

June 2023

Acknowledgements

I would like to offer my special thanks to my supervisors, Dr. Hsiang-Ting (Tim) Chen, Dr. Minhui (Jason) Xue, and Dr. Damith Ranasinghe for their invaluable advice, continuous support, and patience during my PhD study. Their immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life.

To my lab mates and co-authors, who have brought me regular mid-night messages and constant inspiration for many years, thanks.

Finally, thanks to my parents; you may not understand this thesis, but your pride and faith in me has been limitless.

Dedication

To my love Yibo Wang,

for helping me become the best person I could be.

Publications

This thesis contains the following works that have been published or prepared for publication:

1. **Ruoxi Sun**, Wei Wang, Minhui Xue, Gareth Tyson, Surya Camtepe, and Damith Ranasinghe. An Empirical Assessment of Global COVID-19 Contact Tracing Applications. In *proceedings of the 43rd International Conference on Software Engineering (ICSE, **CORE A***), Technical Track*, 2021

2. **Ruoxi Sun**, Minhui Xue, Gareth Tyson, Seyit Camtepe, and Surya Nepal. Not seen, not heard in the digital world! Measuring privacy practices in children's apps. In *The Web Conference (WWW, **CORE A***)*, 2023.

3. **Ruoxi Sun**, Hanwen Wang, Minhui Xue, and Tim Chen. PPVR: A privacy-preserving approach for motion and spatial data in VR. Submitted to *the 31st IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR, CORE A*)*, 2024.

In addition, I have the following publications that are ***NOT included*** but are related to the research domain of this thesis:

4. **Ruoxi Sun** and Minhui Xue. Quality assessment of online automated privacy policy generators: An empirical study. In *Proceedings of the Evaluation and Assessment in Software Engineering (EASE, **CORE A**)*, 2020

5. **Ruoxi Sun**, Wei Wang, Minhui Xue, Gareth Tyson, and Damith Ranasinghe. VenueTrace: A Privacy-by-Design COVID-19 Digital Contact Tracing Solution. In *proceedings of the 18th ACM Conference on Embedded Networked Sensor Systems (SenSys, **CORE A***), COVID-19 Response Research Track*, 2020

6. Wei Wang, **Ruoxi Sun**, Minhui Xue, and Damith Ranasinghe, An Automated Assessment of Android Clipboards. *The 35th IEEE/ACM International Conference on Automated Software Engineering (ASE, **CORE A***), Late Breaking Results Track*, 2020.

7. Xiaotao Feng, **Ruoxi Sun**, Xiaogang Zhu, Minhui Xue, Sheng Wen, Dongxi Liu, Surya Nepal, and Yang Xiang. SNIPUZZ: Black-box Fuzzing

of IoT Firmware via Message Snippet Inference. *The 28th ACM Conference on Computer and Communications Security (CCS, **CORE A***)*, 2021

8. Pingyi Hu, Zihan Wang, Ruoxi Sun, Hu Wang, and Minhui Xue. M^{4}I: Multi-modal models membership inference. In *Advances in Neural Information Processing Systems (NeurIPS, **CORE A***)*, 2022.

9. Chaoran Li, Xiao Chen, **Ruoxi Sun**, Minhui Xue, Sheng Wen, Muhammad Ejaz Ahmed, Seyit Camtepe, and Yang Xiang, Cross-Language Android Permission Specification, *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE, **CORE A***)*, 2022

10. Kunpeng Zhang, Xi Xiao, Xiaogang Zhu, **Ruoxi Sun**, Minhui Xue, and Sheng Wen. Path Transitions Tell More: Optimizing Fuzzing Schedules via Runtime Program States. *The 44th International Conference on Software Engineering (ICSE, **CORE A***), Technical Track*, 2022.

11. Hamish Spencer, Wei Wang, **Ruoxi Sun**, and Minhui Xue. Dissecting Malware in the Wild. *Australasian Information Security Conference*, 2022. (**CORE Best Student Paper Award**)

12. Matthew Crawford, Wei Wang, **Ruoxi Sun**, and Minhui Xue. Statically Detecting Adversarial Malware through Randomised Chaining. *Australasian Information Security Conference*, 2022.

13. Yifan Zhou, Zhengdong Shi, and **Ruoxi Sun**. Visualization and Attack Prevention for a Sensor-Based Agricultural Monitoring System. *Australasian Information Security Conference*, 2022.

14. Wanlun Ma, Derui Wang, **Ruoxi Sun**, Minhui Xue, Sheng Wen, and Yang Xiang. The "Beatrix" resurrections: Robust backdoor detection via Gram matrices. In *the Network and Distributed System Security Symposium (NDSS, **CORE A***)*, 2023.

15. Shuo Wang, Mahathir Almashor, Alsharif Abuadbba, **Ruoxi Sun**, Minhui Xue, Calvin Wang, Raj Gaire, Seyit Camtepe, and Surya Nepal. DoITrust: Dissecting on-chain compromised internet domains via graph learning. In *the Network and Distributed System Security Symposium (NDSS, **CORE A***)*, 2023.

16. Yuxin Cao, Xi Xiao, **Ruoxi Sun**, Derui Wang, Minhui Xue, and Sheng Wen. Stylefool: Fooling video classification systems via style transfer. In *44th IEEE Symposium on Security and Privacy (IEEE S&P, **CORE A***)*, 2023.

17. Shuo Wang, Sharif Abuadbba, Sidharth Agarwal, Kristen Moore, **Ruoxi Sun**, Minhui Xue, Surya Nepal, Seyit Camtepe, and Salil Kanhere. PublicCheck: Public integrity verification for Services of run-time deep models. In *44th IEEE Symposium on Security and Privacy (IEEE S&P, **CORE A***)*, 2023.

18. Zihan Wang, Olivia Byrnes, Hu Wang, **Ruoxi Sun**, Congbo Ma, Huaming Chen, Qi Wu, and Minhui Xue. Data hiding with deep learning: A survey unifying digital watermarking and steganography. *IEEE Transactions On Computational Social Systems*, 2023.

19. **Ruoxi Sun**, Minhui Xue, Gareth Tyson, Tian Dong, Shaofeng Li, Shuo Wang, Haojin Zhu, Seyit Camtepe, and Surya Nepal. Mate! Are you really aware? An explainability-guided testing framework for robustness of malware detectors. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE, **CORE A***)*, 2023.

20. Hongsheng Hu, Shuo Wang, Jiamin Chang, Haonan Zhong, **Ruoxi Sun**, Shuang Hao, Haojin Zhu, and Minhui Xue. A duty to forget, a right to be assured? Exposing vulnerabilities in machine unlearning services. *In the Network and Distributed System Security Symposium (NDSS, **CORE A***)*, 2024.

21. Minhui Xue, Surya Nepal, Ling Liu, Subbu Sethuvenkatraman, Xingliang Yuan, Carsten Rudolph, **Ruoxi Sun**, and Greg Eisenhauer. Rai4ioe: Responsible AI for enabling the Internet of Energy. In The Fifth IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (IEEE TPS-ISA), 2024.

Chapter 1

Introduction

The internet and mobile services provide many conveniences for our daily lives and have greatly changed people's lifestyles, especially in terms of interaction with each other. These services collect large amounts of users' personal information and use them in many areas, such as in medical and financial systems, but they also pose an unprecedented threat to users' privacy and security.

*Privacy and security statements (*e.g.*, the privacy regulations and the privacy policies provided by app developers) are used to protect users' personal information. However, the actual behaviours of an app may be inconsistent with what is asserted in their privacy policies, which will cause serious security and privacy issues, such as personal information abusing, data breach, and prone to security attacks.*

To understand and measure the privacy and security practice of mobile apps, this research built a framework , aiming to ensure that mobile users and developers can reliably recognise the privacy issues and risks on their smart devices. The datasets and tools developed in this research have been open-sourced to public. The findings of this research have been summarised into guidelines for developers and suggestions to legislators.

1.1 Introduction

With the increasing significance of mobile phones in our daily lives, the issues of information privacy and security have become a growing concern for both developers and users of mobile applications. It has become commonplace for mobile apps to access, require, or transmit users' sensitive personal information as part of their service requests. The COVID-19 pandemic has triggered a substantial shift in how consumers utilise and engage with mobile applications. Presently, there is a notable increase in people relying on applications for various purposes such as entertainment, remote work, and everyday tasks. Ninety-two percent of Australian adults now have access to a smartphone and prefer to carry out various tasks and access the internet through mobile applications, rather than laptops or computers [1]. However, among the 2,669,912 applications available in the Google Play Store, only 23 % of applications have ratings higher than 4.0/5.0 [2] (last update: 04 Dec 2021). It is not surprising that the Office of the Australian Information Commissioner (OAIC) received 2,474 privacy complaints covering 2,698 issues violating Australian Privacy Principles in the single year of 2020–2021 [3].

In order to safeguard user privacy and regulate the conduct of mobile applications, numerous countries have implemented regulations. These include the Children's Online Privacy Protection Act (COPPA) [4], the European General Data Protection Regulation (GDPR) [5], the California Online Privacy Protection Act (CalOPPA) [6], and the Privacy Act 1988 in Australia [7]. These regulations aim to promote and protect the privacy of individuals, as well as govern the handling of personal information by organisations.

These regulations impose limitations on app behaviour and establish the responsibilities of app developers. One key requirement under these privacy regulations is the mandatory inclusion of a privacy policy for an app. A privacy policy outlines the collection, usage, retention, and disclosure of user information. While app stores often require certain privacy disclosures, such as permission lists and privacy policies, it can be challenging for average users to comprehend the potential risks associated with an app, let alone identify undesired or malicious behaviours of the application [8]. Significantly, findings from the Australian Community Attitudes to Privacy Survey 2020 [9] indicate that a considerable proportion of Australians (59 %) have encountered issues regarding the management of their personal information within the previous 12 months. Furthermore,

parental concerns regarding their children's privacy are even more pronounced. The majority of parents in Australia strongly believe that children should have the right to develop and grow without being subjected to profiling and targeted marketing (84 % agree, 59 % strongly agree). This app user environment is referred to as an "ecosystem". *We, the people of the world, are recurrently suffering from such unpleasant privacy ecosystem.*

Recent works study whether or not mobile apps behaviour matches statements in privacy policies [10, 11, 12]. The actual behaviour of apps may be inconsistent with the statements claimed in privacy policies. This can lead to many privacy and security issues. The personal data could be collected and abused by an app without user consent. An app may not provide data protection asserted in its privacy policy and causes data breach. Even worse, the leaked sensitive information could be illegally used in business. For example, once health data (such as an electronic health record) is leaked to an insurance company, it may lead to an increase or even rejection of user health and life insurance [13]. Therefore, it is worth studying and debating whether the privacy and security of the user can be protected by the mobile apps assertions and behaviours.

Through this research, I develop a novel approach to assess the current privacy practices of mobile applications, **to ensure that mobile users and developers can reliably recognise the privacy issues and risks on their smart devices. The findings of this research are summarised into guidelines for developers and aim to further make suggestions to legislators.** Specifically this research aims to (*i*) reveal the current status, issues, and the risks among mobile application software, with respect to the privacy practice; and (*ii*) provide a comprehensive guideline for the mobile software industry and legislators, to build a stronger and safer privacy ecosystem.

1.2 Background & Related Work

Thanks to the success of the smart device industry, mobile software privacy has become a very promising and challenging field that aims at resolving various privacy issues and threats currently inherent in mobile applications. The protection of users' personal sensitive data is among the most common concerns among smartphone users. Although many countries restrict app behaviour through laws, regulations and app store policies, our community has not yet developed a reliable solution to automate large-scale privacy compliance.

Privacy requirements. Most app stores have released strict developer policies, along with inspection and vetting processes before app publishing, seeking to nip the security and privacy threats in the bud and improve app quality in the markets. For example, Google Play has released a set of developer policies [14] that cover 10 main categories, including "Restricted Content", "Privacy, Deception and Device Abuse", "Monetisation and Ads", and "Families". Each category stands for a type of violation that may be associated with various undesired behaviours. For example, if one of the target audiences for an application is children, the application must comply with the Families Policy Requirements, which explicitly outlines standards that must be followed for apps that are intended for children's use. Apps that that do not follow these policies cannot be published on Google Play. However, the information presentation and target audience registration in Google Play store are based on self-certification manner.

Mobile application analysis. Static and dynamic analysis techniques are widely used in the practical assessment of mobile apps. For example, MobSF [15] offers automated application penetration testing, malware analysis, and a security and privacy assessment framework. FlowDroid [16] 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 utilise syntax-based scanning and data-flows, which leads to numerous false positives that are not relevant to personal identifiable information.

Mobile app comments have been extensively studied from other perspectives, including mining user opinions, app comment filtering, and exploring other concerns. For example, Chen *et al.* [17] conduct a study on the unreliable maturity content ratings of mobile apps, which may result in inappropriate risk exposure for the children and adolescents. This research focuses more on the comparison of content ratings between iOS app store and Google Play store, rather than exposing undesired application behaviours.

Recent research by Liu *et al.* [18] targets to solve the problem of compliance analysis between GDPR and privacy policies, utilising a combination of sentence classification and rule-based analysis. However, the corpus suffers from the "imbalanced data" problem, which greatly affects the classification accuracy.

This research. In contrast to the aforementioned works, this thesis not only proposes and develops a holistic automated security and privacy assessment

tool to test apps designs but also conduct user studies to understand users' concerns and requirements to reinforce security/privacy by design, in an attempt to accommodate to the mobile software industry (Chapter 3).

In addition, based on the security and privacy assessment of mobile applications, I expand my search into the privacy policy domain, *e.g.*, to understand the privacy and security assertion and measure the actual behaviours of applications (Chapter 4).

Furthermore, taking VR application as an example where potentially more sensitive information from users are collected and share, I investigated into privacy enhancement techniques, proposing a practical approach to applying differential privacy on streamed spatial data, such as the user visiting record in a virtual home or a user's body movement trace in a virtual game, with data utility preserved (Chapter 5).

Please note that I have also conducted an investigation into existing online automatic privacy policy generators (APPGs) which heavily rely on user input and templates. This research distinguishes itself from previous work by examining both the evaluation of privacy policies and app behaviours in terms of their completeness and consistency. The results of this research [8] were published three months prior to the commencement of my PhD study. While I cannot include this study in my thesis, it serves as an indispensable component of the framework proposed by this thesis.

1.3 Objectives of my research

The project aims to enforce the privacy compliance of mobile applications. The outcomes of this research will greatly benefit both mobile application developers and users. Developers will be supported with much better tools enabling efficient development of privacy preserving mobile applications that comply with the privacy regulations and online app store polices, maximising the potential of the mobile privacy ecosystem for Australian businesses. Users will have access to better apps that have less privacy issues. To achieve this, the key objectives are to:

- **Objective 1**: Define a comprehensive taxonomy of privacy issues and regulation violations that frequently appear in applications, and establish a knowledge base (including datasets) to model such compliance and privacy issues;
- **Objective 2**: Build a sound evaluation system to analyse the privacy practice of mobile applications, as well as their privacy policies. A series of evaluation tools will be developed to interpret these privacy policies and extract application behaviours from package files and user comments;
- **Objective 3**: Conduct a large-scale privacy evaluation of mobile applications to reveal the non-compliance among the privacy policies and application behaviours;
- **Objective 4**: Research and develop a practical privacy enhancement approach for particular mobile application scenarios.

1.4 Methodology

This research focuses on Android applications. With over 70% of global market share [19], Android is the most popular mobile platform in the world. Android applications are relatively easier to be obtained and analysed, compared with other mobile operating systems, due to its open ecosystem. Nonetheless, the design and methodology proposed in this research could extend to address similar privacy evaluation on other mobile platforms, such as iOS. The objectives of this research will be achieved through **four phases** that are designed to achieve objectives 1-4 respectively, as illustrated in Figure 1.1. Please note that the results of research aiming to address Objective 1 were published three months prior to the commencement of my PhD study, marked as "prior research" in Figure 1.1). While I cannot include this study in my thesis, it serves as an indispensable component of the framework proposed by this thesis.

1.4.1 Phase 1: Regulation Modelling and Dataset Establishment (Objective 1)

The first task of my research focused on the knowledge base building and information collection, which was established before the analysis tool development stage. I extracted privacy requirements from laws, regulations, and app store

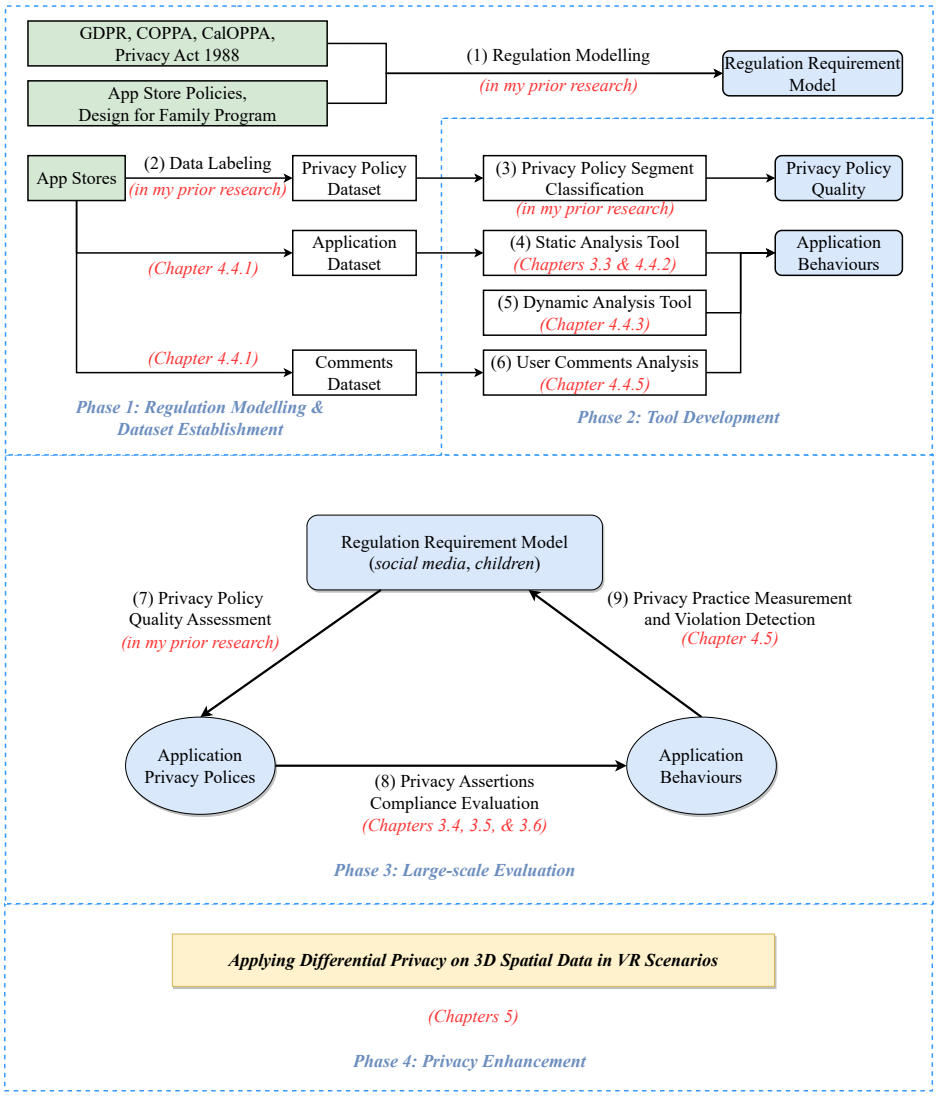


Figure 1.1. Methodology overview.

policies. A dataset was established to enable a large-scale privacy evaluation in a later stage.

Modelling the regulations and app store policies [8]. The handling of personal information is governed by legislation, such as *the Privacy Act 1988* at the federal level. App stores, *e.g.*, Google Play Store, also restrict the behaviour of mobile application with store policies and self-certificated programs. In this task, I extracted and modelled the privacy requirements. More specifically, I mapped the specific application behaviours with privacy activities required in online privacy regulations and policies.

In my work [8], I conducted an empirical study which determines what categories and items should be covered in a mobile application privacy policy. This

work has successfully assessed the auto-generated privacy policies. In this task, I further extended this work to focus more on social media applications and children's privacy domains.

Dataset establishment (Chapters 4.4.1). To enable a large-scale privacy practice evaluation, a dataset of mobile applications was established, which contains the application package files, *e.g.*, Android Package Kit (APK), user comments in app stores, and the privacy policies stated by the application developers or organisations. Concretely, I collected the APKs and user comments from Google Play Store using an account located in Australia. The privacy policies were downloaded from developers' official websites or extracted from the mobile applications.

The data practices in the privacy policy corpus was further identified according to the privacy model extracted. Specifically, each data practice in the policy text was annotated into a privacy category, such as "First-party privacy information collection", "Information sharing with third-parties", or "Children's privacy". A small group of domain experts were hired as skilled annotators and an annotation tool is developed to automate the annotation procedure. I labelled the privacy policies in a fine-grained manner, *e.g.*, annotating on sentence level instead of on paragraph-length segments, and most important, involving more specific privacy categories addressing regulations and app store policies. Despite enabling the large-scale privacy evaluation, the research community can also use this corpus to advance research in both privacy and language technologies.

1.4.2 Phase 2: Analysis Tool Development (Objective 2)

Millions of mobile applications are available on various app markets and platforms. Although app stores have enforced vetting approaches before releasing an app, a large number of low-quality or regulation violating apps still exist in the market. To identify these undesirable mobile application behaviours, especially the privacy practice in the real world, static and dynamic analysis tools were developed during this phase. Further, I took user comments as a source of information to identify violations in an accurate and timely manner, utilising state-of-the-art NLP and machine learning technologies. In this phase, a machine learning classifier was also trained to recognise what privacy requirements have been addressed in privacy policies, which was further used in the privacy policy quality evaluation in the next stage.

Privacy policy classification [8]. Based on the established privacy policy corpus, a multi-class classification machine learning model was trained to identify the privacy activities that have been covered in the application privacy policies. Specifically, each privacy policy sample, *e.g.*, a labelled policy statement sentence, was first converted into a feature vector. To achieve a suitable highly-discriminative representation, various text representation techniques were studied, *e.g.*, word mapping, count vetoriser, TF-IDF, and pre-trained transformers. Next, I leveraged multiple machine and deep learning algorithms for privacy policy classification, evaluating their effectiveness in detecting different segment-level categories. The learning algorithms include logistic regression, random forest, convolutional neural networks, deep neural networks, and fine-tuned transformers. It should be noted that transformers are increasingly the model of choice for NLP problems, which have been trained with large language datasets and can be fine-tuned for specific tasks. Therefore, the main efforts in this research task focused on the pre-processing of privacy policy samples (*e.g.*, removing stop-words, word lemmatisation and stemming) and the training or fine-tuning of models to select the best-performing learning algorithm that fits the privacy policy annotation classification.

Static analysis tool development (Chapters 3.3 and 4.4.2). Static code analysis was performed by examining source code for signs of security vulnerabilities without executing the program. Several studies [20, 21, 22, 23] have used static analysis to analyse different types of Android applications in search of malicious behaviours and privacy leaks.

In this task, I developed an automatic tool to detect undesired app behaviours and violations against regulation, such as obtaining permissions without user consent, using advertisement Software Development Kit (SDK) or trackers not certified in Google Designed for Families (DFF), and privacy information leakage to third-parties. Considering that static analysis always suffers from false-positives, a manual review of the static analysis tool was performed for quality control.

Dynamic analysis tool development (Chapter 4.4.3). In contrast to static analysis, dynamic analysis executes the code and captures the program status during run-time. Rather than solely relying on offline code analysis, it is possible to monitor vulnerabilities and program behaviours while the program is actively running, thereby gaining insights into its real-world behaviour. Dynamic analysis techniques, commonly employed for mobile applications, include

fuzzing and network traffic monitoring. These approaches enable the observation and assessment of a program's behaviour and interactions during runtime. In my research, I use dynamic approaches in privacy leakage detection [24], fuzzing optimisation [25] and IoT firmware fuzzing [26].

In this task, I first researched and developed a dynamic analysis framework adopting the mechanism of the state-of-the-art network traffic monitoring tool on Android applications, Lumen [27], which has been widely used in the privacy research community. New tracker lists and personal identifiable information (PII) keywords were involved according to the privacy requirement model extracted in Phase 1, which enabled a fine-grained analysis, particularly focused on the regulations and specific user groups, such as kids. Further, a user interface fuzzing tool was developed. Different from the existing user interface exerciser, such as Monkey [28], the fuzzing tool developed in this research fuzzes the applications according to the structure of the user interface, instead of generating pseudo-random streams of user events, to mimic the user input to mobile applications while ensuring the code coverage of fuzzing.

User comments analysis (Chapter 4.4.5). Besides the static and dynamic analysis of applications, the comments from users reflect timely feedback that may reflect the violation of regulations or policies, including user concerns. Analysis of the user comments reveals such undesired application behaviours. Recent research [29] resorts to user comments to identify violations against mobile application market policies, using a semi-automated rule-based process. In this task, I adopted state-of-the-art text mining and natural language processing (NLP) techniques, such as transformers, to interpret the application comments and further train machine learning models to categories and identify informative comments.

Specifically, comment clusters were obtained from topic modelling based on the training set of user comments at first (as shown in Figure 1.2). Manual inspection was then conducted to refine the comment clusters and build a violation model, *e.g.*, recognising the centres of clusters that indicate violations or undesired behaviours. Finally, using the violation model, the application behaviours were extracted, functioning in complement with the static and dynamic analysis results (since some application behaviours may not be identified by technical analysis, but could be reflected by users).

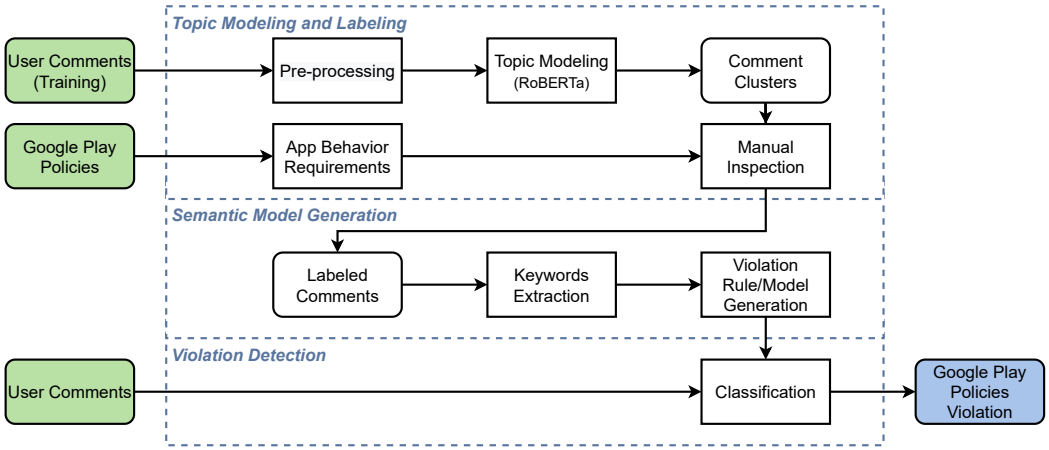


Figure 1.2. User comments analysis.

1.4.3 Phase 3: Large-scale Evaluation (Objective 3)

After the privacy requirements model and application datasets were established (Phase 1) and analysis tools and machine learning models were ready (Phase 2), I conducted a large-scale evaluation of mobile applications as the third phase of my research. Through evaluations, I assessed the compliance of privacy policies and the application behaviours based on regulations and store policies. This was achieved via the following three distinct tasks.

Privacy policy quality assessment [8]. Information privacy issues are of growing concern to developers as well as to mobile app users. Privacy policies are one of the key legal statements that are used to protected users' personal information. However, a low-quality privacy policy may miss critical legislative requirements such as providing the identity of data collectors, communicating what personal information will be collected, and how the collected information will be used. This makes it difficult for regular users to determine the trustworthiness of apps.

In this research task, I assessed the quality of privacy policies at a large scale to reveal the problems that arise in the practice of privacy regulations. Based on the privacy requirement model and the privacy policy classifier built in previous phases, I determined what categories and items should be covered in a complete privacy policy and further analysed the completeness of privacy policies. Specifically, each statement in a privacy policy was entered into a "classifier" and the output labels were summarised to indicate what items have been covered in the privacy policy. Then I compared the classification results with the privacy

policy requirements to flag any missing items. Using this information, I implemented an automatic privacy policy quality assessment tool to help mobile users and developers check and improve their privacy policies. This tool, would be the first to span the requirements of the privacy regulations and app store policies, and is stricter in terms of social media and children's privacy.

Privacy assertions compliance evaluation (Chapters 3.4, 3.5, and 3.6). In addition to assessing whether the privacy policy complies with regulations, I further assess whether the application behaviours comply with their privacy policies. Thanks to the static and dynamic tools and the user comment analysis model built in Phase 2, in this research task, I am able to extract various application behaviours, *e.g.*, the permission requested and granted, the network traffic, the collection and sharing of personal identifiable information, and other undesired behaviours reflected by user comments. The application behaviours are compared with their privacy policies (annotated by the classifier) to figure out the compliance of mobile applications. In addition to a large-scale evaluation, particular cases are studied to showcase the application behaviours that do not comply with their privacy policies.

Privacy practices measurement and violation detection (Chapter 4.5). Considering that the scope in previous research task could be limited to identify inconsistency in application privacy policies, I further investigated regulation violations by evaluating the application behaviours against the regulation requirements. Some application behaviours are not required to be explicitly stated in privacy policies but are still restricted by regulations. In this research task, I focused on additional application behaviours that are required by regulations and app store policies, such as age verification, parental consent, and the use of advertisement SDK and trackers in applications. Note that, due to the consideration of user comments, more undesired application behaviours were detected, even if these behaviours are not explicitly mentioned or restricted in the regulations or app store polices.

1.4.4 Phase 4: Privacy Enhancement (Objective 4)

In today's data-driven world, the need for effective implementation of privacy-enhancing technology is more critical than ever. Proposing practical approaches to implementing privacy-enhancing technology is crucial due to rising privacy concerns, regulatory requirements, the need to build trust, data magnetisation opportunities, and ethical considerations. In Chapter 5, I particularly focus on

proposing a practical approach to enhance streamed spatial data collected from Virtual Reality (VR) applications, to protect the user's privacy against user identification attacks.

To conclude, based on the large-scale evaluation of mobile application privacy practice in previous phases, the outputs of all research tasks is thoroughly evaluated and summarised. Manual analyses was leveraged as an important quality control mechanism to assess the precision, recall, and F-measure of the results. Typical cases were selected for further studies. In addition to real-world mobile application developers, application users were also contacted to ensure that their interests and concerns are well respected. Finally, the analysis tools and datasets developed throughout this research were released publicly to help developers and users evaluate the privacy protection during and after application development. Please see more details of legislation informing in takeaways and development guidelines in my prior research [8], Chapters 3, and 4.

1.5 Significance

This research addresses the critical problem of privacy compliance of mobile applications, aiming to strengthen mobile privacy ecosystem. At present, compliance issues in mobile applications pose a number of problems to users, developers, and businesses.

- For users, my research provides a publicly-available and easy-to-use privacy assessment platform which arms normal users fighting against these and other privacy threats. A real and ever-present risk is that using mobile applications can increase the likelihood of privacy leakage affecting their finances (*e.g.*, by data breaches in the financial services), identity (*e.g.*, by leaking users' phone number and email addresses) or even safety (*e.g.*, by tampering with users' health data). Unfortunately, with so much data passing through the internet via mobile applications, the chances of involuntary data leaks become fairly high. Savvy hackers find it quite lucrative to infiltrate personal files and threaten to release them publicly.
- For developers, privacy assessment tools and privacy practice guidelines in my research help less knowledgeable developers overcome the barriers to privacy compliance. Neglecting to safeguard privacy can lead to violations of regulations, potentially resulting in significant legal consequences. It is crucial for

developers to incorporate robust privacy protections into their everyday business practices and ensure compliance with the obligations outlined in privacy regulations. By doing so, they can mitigate the risk of privacy breaches and adhere to their legal responsibilities. However, not every app developer is a legal professional, which makes the understanding of privacy regulations a time-consuming task and implementing the privacy requirements even harder.

- For businesses, my research emphasises the importance of safeguarding consumer privacy as a core objective. There is always ongoing pressure to stay ahead of business competitors, and the growth of the mobile ecosystem is compounding the number of ways in which this competition presents itself. Businesses that prioritise and implement privacy protections will not only enhance their operations but also position themselves as preferred choices among consumers compared to competitors lacking such safeguards. When organisations demonstrate this commitment through transparent and consistently followed privacy practices, they build emotional connections with their audience, thereby improving their brand value. By showcasing their dedication to consumer privacy, these businesses establish trust, foster loyalty, and ultimately strengthen their brand reputation.

1.6 Contributions

Throughout this thesis, I built methods and evaluation frameworks for understanding and measuring privacy and security assertions of mobile apps, with specific solutions that address the limitations and challenges outlined in the sections above.

Contribution 1 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 (*i.e.*, "apps"). These apps, however, have raised security and privacy concerns. I proposed an automated security and privacy assessment tool for Android apps, 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 my findings, I undertook a user study to investigate concerns

regarding contact tracing apps. I offered concrete guidelines, and highlight gaps between user requirements and app performance. Details of this research are described in Chapter 3.

Contribution 2 While legislatures around the world have enacted regulations and laws to protect children's online privacy, and app stores have instituted various restrictive protections and requirements, privacy in mobile apps remains a growing concern for parents, as well as the wider society. I investigated 20,195 mobile apps from the Google Play store that are designed particularly for children (Family apps) or include children in their target user groups (Normal apps). The findings suggested that, despite significant attention to children's privacy, a large gap between regulatory provisions, app store policies, and actual development practices exists. Details of this research are described in Chapter 4.

Contribution 3 I developed a novel approach that leverages differential privacy to safeguard users' privacy in the context of VR applications, specifically focusing on streamed spatial data. By applying differential privacy mechanisms, I effectively mitigate the risk of user identification through attacks, thereby enhancing privacy protection. Details of this research are described in Chapter 5.

Mobile devices are playing an increasingly ubiquitous role in our lives, forcing more and more businesses to join the mobile ecosystem. At the same time, due to the large amount of revenue involved, the number of privacy breaches and malicious attacks are continuously increasing, exerting pressure on software industry to reliably deliver regulation compliance and privacy preserving mobile applications. **The outcomes of this thesis not only advanced the knowledge in the field of mobile software engineering but also brought a range of benefits to software developers, users and businesses.**

Bibliography

[1] Australian Competition & Consumer Commission. Digital platform services inquiry. https://bit.ly/3mVEOg2.

[2] Android and Google Play statistics. https://www.appbrain.com/stats.

[3] Office of the Australian Information Commissioner. Annual report 2020–21. https://bit.ly/32Mz1Cv.

[4] Children's online privacy protection act (coppa). http://www.ftc.gov/ogc/coppa1.htm, 1998.

[5] European Parliament and Council of the European Union. Regulation (eu) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data and repealing Directive 95/46/EC (general data protection regulation). *Official Journal of the European Union*, 119:1–88, 2016.

[6] California online privacy protection act (caloppa). https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=BPC&sectionNum=22575, 2003.

[7] Privacy act 1988. https://www.legislation.gov.au/Series/C2004A03712/Compilations, 1988.

[8] Ruoxi Sun and Minhui Xue. Quality assessment of online automated privacy policy generators: an empirical study. In *Proceedings of the Evaluation and Assessment in Software Engineering (EASE)*, pages 270–275, 2020.

[9] Office of the Australian Information Commissioner. Australian community attitudes to privacy survey 2020. https://bit.ly/3mVFh1M.

[10] Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D Breaux, and Jianwei Niu. Toward a framework for detecting privacy policy violations in android application code. In *Proceedings of the 38th International Conference on Software Engineering*, pages 25–36. ACM, 2016.

[11] Le Yu, Xiapu Luo, Xule Liu, and Tao Zhang. Can we trust the privacy policies of android apps? In *2016 46th Annual IEEE/IFIP International*

Conference on Dependable Systems and Networks (DSN), pages 538–549. IEEE, 2016.

[12] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel Reidenberg, N Cameron Russell, and Norman Sadeh. Maps: Scaling privacy compliance analysis to a million apps. *Proceedings on Privacy Enhancing Technologies*, 2019(3):66–86, 2019.

[13] GS McNeal. Health insurer anthem struck by massive data breach. *Forbes, http://bit. ly/2VaKTtf (30.04. 2019)*, 2015.

[14] Providing a safe and trusted experience for everyone. https://play.google.com/about/developer-content-policy/.

[15] Mobile-security-framework-mobsf. https://github.com/MobSF/Mobile-Security-Framework-MobSF.

[16] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[17] Ying Chen, Heng Xu, Yilu Zhou, and Sencun Zhu. Is this app safe for children? A comparison study of maturity ratings on android and ios applications. In *Proceedings of the 22nd international conference on World Wide Web*, pages 201–212, 2013.

[18] Shuang Liu, Baiyang Zhao, Renjie Guo, Guozhu Meng, Fan Zhang, and Meishan Zhang. Have you been properly notified? automatic compliance analysis of privacy policy text with gdpr article 13. In *Proceedings of the Web Conference 2021*, pages 2154–2164, 2021.

[19] Statecounter. Mobile operating system market share worldwide. https://bit.ly/3eNX72a.

[20] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. Iccta: Detecting inter-component privacy leaks in android apps. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 280–291. IEEE, 2015.

[21] Sen Chen, Ting Su, Lingling Fan, Guozhu Meng, Minhui Xue, Yang Liu, and Lihua 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*, pages 797–802, 2018.

[22] Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, Seyit Camtepe, and Damith C Ranasinghe. An empirical assessment of global covid-19 contact tracing applications. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1085–1097. IEEE, 2021.

[23] Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, and Damith C. Ranasinghe. Venuetrace: A privacy-by-design covid-19 digital contact tracing solution: Poster abstract. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys)*, page 790–791, 2020.

[24] Wei Wang, Ruoxi Sun, Minhui Xue, and Damith C. Ranasinghe. An automated assessment of android clipboards. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, page 1249–1251, 2020.

[25] Kunpeng Zhang, Xi Xiao, Xiaogang Zhu, Ruoxi Sun, Minhui Xue, and Sheng Wen. Path transitions tell more: Optimizing fuzzing schedules via runtime program states. In *2022 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2022.

[26] Xiaotao Feng, Ruoxi Sun, Xiaogang Zhu, Minhui Xue, Sheng Wen, Dongxi Liu, Surya Nepal, and Yang Xiang. Snipuzz: Black-box fuzzing of iot firmware via message snippet inference. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 337–350, 2021.

[27] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, Phillipa Gill, et al. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS 2018)*, 2018.

[28] Ui/application exerciser monkey. https://developer.android.com/studio/test/monkey.

[29] Yangyu Hu, Haoyu Wang, Tiantong Ji, Xusheng Xiao, Xiapu Luo, Peng Gao, and Yao Guo. CHAMP: Characterizing undesired app behaviors from user comments based on market policies. In *Proceedings of 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 933–945, 2021.

Chapter 2

Literature Review

In this chapter, I review the main techniques and recent research in (i) security and privacy evaluation; (ii) static and dynamic analysis approaches, and (iii) regulations and laws compliance.

2.1 Introduction

The research on understanding and measuring privacy and security assertions of mobile apps is where several domains overlap, *e.g.*, I utilise static and dynamic analysis approaches, as well as the user comments as sources of user feedback, aiming to evaluate and expose the security and privacy issue existed in mobile apps, according to the security and privacy assertions, including privacy policies, regulations, and laws. In this chapter, I review the main techniques and recent research in (i) security and privacy evaluation; (ii) static and dynamic analysis approaches, and (iii) regulations and laws compliance. Figure 2.1 illustrates the overview of this chapter.

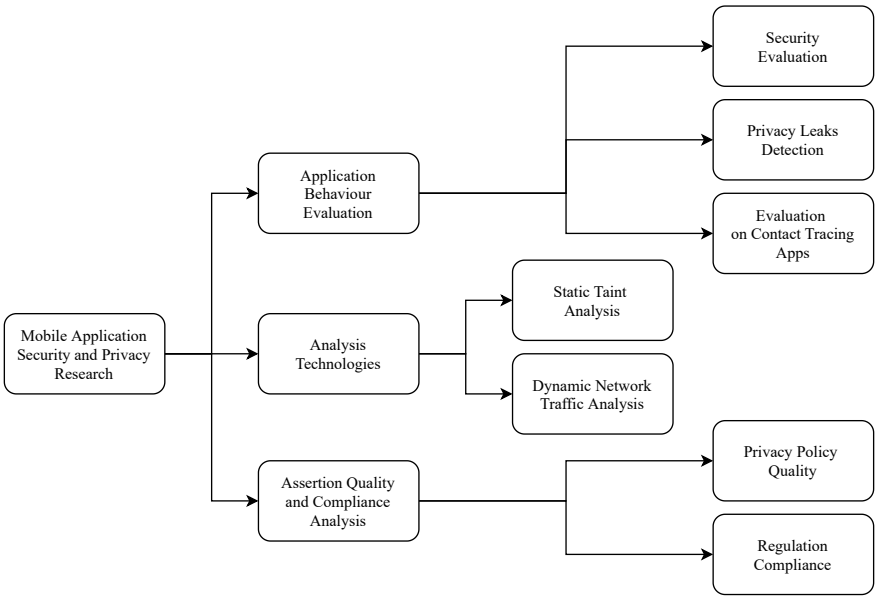


Figure 2.1. Overview of the literature review.

Specifically, I start by reviewing the work that focuses on the the privacy and security evaluation of mobile applications, exposing flaws and issues in application design and development (Section 2.2). Then I particularly focuses on the static and dynamic analysis approaches that could has been introduced to the security and privacy research domain (Section 2.3). Another approach to analyse mobile application behaviour is based on the auto-parsing and understanding of user comments. In the remaining part of this chapter, the literature related to privacy regulations and laws compliance are reviewed (Section 2.4). Finally Section 2.5 presents reflections and conclusions of this chapter.

The reason I focus on the these domains is that my research on understanding and measuring privacy and security assertions of mobile apps is where they overlap, *e.g.*, I utilise static and dynamic analysis approaches, as well as the user comments as sources of user feedback, aiming to evaluate and expose the security and privacy issue existed in mobile apps, according to the security and privacy assertions, including privacy policies, regulations, and laws.

2.2 Security and Privacy Evaluation

With the rapid development of the Internet and smart phones, mobile applications occupy an increasingly important position in people's daily life. While these applications are convenient for people's lives, they also bring new problems and risks, such as the embedding of mobile phone malware, the leakage of personal privacy data, user behaviour tracking for the purpose of personalised advertising, and even financial fraud and the spread of fake news. This makes the security and privacy evaluation of mobile applications a hot research topic in recent years. The related research mainly falls into two categories: (i) Security evaluation and malware detection and (ii) Privacy leaks and user tracking. Table 2.1 illustrates recent research in security and privacy evaluation on mobile applications.

2.2.1 Security Evaluation and Malware Detection

To evaluate the security of mobile applications, as well as the mobile ecosystem, several studies research on the bug analysis [1, 2, 17] and malware detection [3, 4, 5, 6, 18, 19, 20, 21, 22] in Android system, proposing different methods of feature extraction and approaches to enabling state-of-the-art machine learning technologies in malware detection domain. On the other hand, recent research [7, 8, 23, 24, 25, 26, 27, 28] looks into the enhancement of machine learning-based malware detectors, focusing on the generalisation issue existing in the machine learning-based Android malware classifiers over a long period, *e.g.*, concept drift). Note that there are also recent studies focus on the security of machine learning models [29, 30, 31, 32, 33, 34, 35] which may hint novel approaches in malware detection through machine learning techniques.

Software bug analysis. Software bug analysis entails the process of identifying, diagnosing, and rectifying anomalies or flaws within software applications.

Table 2.1. Recent research in security and privacy evaluations on mobile applications.

Research	Year	Domain	Description
Li *et al.* [1]	2020	Bug analysis	An empirical study of bugs in real-world WebXR projects and bug taxonomy.
ESDroid [2]	2023	Bug analysis	An Event-aware dynamic Slicing technique for AnDroid applications, combining segment-based delta debugging and backward dynamic slicing to narrow the search space to produce precise slices for Android.
StormDroid [3]	2016	Android malware detection	A machine learning-based malware detection framework, which is enhanced with a combination of contributed features and the whole malware detection process is "streaminglized" to support large-scale analysis.
Drebin [4]	2014	Android malware detection	One of the most popular machine learning-based Android malware classification model, relying on static features.
MamaDroid [5]	2016	Android malware detection	A malware detection system for Android, utilising Markov chains built from the sequence of abstracted API calls.
DroidCat [6]	2018	Android malware detection	A dynamic Android malware detection approach, using a set of dynamic features according to the method calls and inter-component communication (ICC) intents.
Transcend [7]	2017	Detectors enhancement	A novel framework to determine whether a classification model is degraded to avoid poor performance.
Tesseract [8]	2019	Detectors enhancement	A classifier robustness metric and an algorithm to tune the classifier have been introduced in this research, allowing the evaluation on mitigation strategies for time decay.
Xue *et al.* [9]	2016	Privacy leakage	This research looks into the privacy risk against the integrity of user anonymity in popular anonymous social media applications.
Recon [10]	2016	Privacy leakage	A corss-platform system, revealing personally identifiable information (PII) leaks in mobile applications.
Englehardt [11]	2016	User tracking	A measurement of online tracking with 1 million sites, proposing an open-source web privacy measurement tool.
Wang *et al.* [12]	2020	User tracking	This research investigates into China's mobile tracking behaviours, making interesting observations with respect to Advertising and Tracking Services (ATSs) popularity and community structure, user monopoly patterns, and personal private information collection.
Ferretti *et al.* [13]	2020	Contact tracing	Analysed the key parameters of pandemic spread and estimated the contribution of digital contact tracing.
Li and Guo [14]	2020	Contact tracing	Conduct a survey on the global deployment and challenges at the very beginning of the pandemic.
Baumgartner [15]	2020	Contact tracing	Demonstrate that the Google/Apple Proposal is vulnerable to several attacks that are practical in real-world.
Gvili [16]	2020	Contact tracing	Analyses the security issues in the specification, finding that significant risk to society could be introduced by this specification.

These anomalies, often denoted as bugs, have the potential to trigger unforeseen behaviours, system crashes, or other malfunctions within the software. Bug analysis constitutes a pivotal component of the software development lifecycle, typically conducted by a collaborative effort involving developers, quality assurance (QA) engineers, and occasionally, specialised bug analysts.

Recent scholarly research, as exemplified by a study conducted by Li *et al.* in 2020 [1], delved into an empirical investigation of bugs encountered in extended reality (XR) applications. The primary objective of this study was to gain insights into the distinctive attributes of these bugs. To accomplish this, the researchers collected a comprehensive dataset comprising 368 real bugs from a multitude of issue reports and release notes emanating from 33 WebXR projects hosted on GitHub. The research findings centred on an in-depth examination of the bugs' symptoms and their underlying causes, ultimately culminating in the

construction of a comprehensive bug taxonomy based on these two fundamental aspects.

In a recent study [2], the focus was on automating the debugging process for Android applications using dynamic slicing techniques. In the Android ecosystem, the execution of an app is heavily influenced by asynchronous events that trigger inter-component communications. This characteristic frequently results in a large search space when existing dynamic slicing methods are employed. To address this challenge, the researchers introduced a novel approach that combines segment-based delta debugging with backward dynamic slicing. This innovative combination aims to effectively reduce the search space and generate precise slices tailored to the Android platform.

Android malware detection. Considering the increasing trend of mobile application usage, the threats of malware attack are raising more and more significant risks to mobile devices. One critical challenge is the mobile malware detection. The StormDroid, a streaminglized machine learning-based malware detection framework, has been proposed by Chen *et al.* [3]. In this research, the researchers particularly defined "streaminglised" as a novelty of the proposed framework which is built on an open-source distributed real-time stream processing engine, which enables a large-scale analysis of a data stream.

The StormDroid is enhanced with a combination of contributed features and the whole malware detection process is streaminglized to support large-scale analysis. The framework of StormDroid consists of 3 phases: Preamble, Feature Extraction, and Classification. Resource files are prepared in Preamble phase. Four types of features are considered in the feature extraction process, including permissions requested, sensitive API calls, API call sequences, and dynamic behaviour. Particularly, different from other research, the StormDroid extracts sequence features by counting the number of sensitive API calls: if the the sum value is higher than the threshold, the corresponding sequence will be marked as 1. Therefore, a high-dimension input vector could be converted into one sequence feature.

A large-scale analysis involving more than 8,000 mobile applications has been conducted by the authors of StormDroid, reporting that the StormDroid is able to improve the malware detection accuracy by near 10%, compared with state-of-the-art antivirus engines. This work not only provides novel contributed

features for malware detection, but also suggests a new perspective on the streaminglised malware detection process.

Drebin [4], one of the most popular machine learning-based Android malware classification model, utilises 8 sets of static features, including hardware features, requested and used permissions, app components, filtered intents, restricted and suspicious API calls, and network addresses. The classifier is built on a lightweight SVM model and achieves 94% detection rate on 5,560 malware samples. The computational overhead of Drebin is quite low, which enables malware detection directly running on smartphones. However, relying on static features, Drebin inherits the limitations of static analysis, *i.e.*, attacks that cannot be detected by static analysis (such as reflection and bytecode encryption) will compromise the Drebin classifier.

In 2016, Mariconti *et al.* [5] presents MaMaDroid, a malware detection system for Android, utilising Markov chains built from the sequence of abstracted API calls. Features are further extracted based on the Markov chains, feeding to the classifier. The highlight of this work is the idea of abstracting calls, which enables the resilience to API changes and controls the feature set size. A accuracy evaluation has been conducted by Mariconti *et al.* on over 8,500 benign and 35,500 malicious applications across six years collection (which is another highlight point of this work, *i.e.*, demonstrating a good detection performance over a long periods of time). The experimental results indicate that the MaMaDroid has an effective detection capability on Adroid malware (F1-score $>$ 99%).

The most important insight yielded by MaMaDroid is the relationship between Markov chains and the behavioural model of mobile applications, which allows MaMaDroid to obtain an accurate malware detection capability. The contribution of MaMaDroid is crucial to the continuous evolution of the Android ecosystem.

A dynamic Android malware detection approach has been proposed by Cai *et al.* [6] as DroidCat, which uses a set of dynamic features according to the method calls and inter-component communication (ICC) intents. According to the experimental results, DroidCat outperforms static approaches and other dynamic detection methods that are purely based on system calls. Specifically, DroidCat distils features from behavioural characterisation study involving both benign and malicious applications.

An evaluation on 34,343 mobile applications (collected across 9 years) has been conducted by Cai *et al.*, reporting a better classification performance (97% F1-score) than two state-of-the-art techniques [20, 22], with respect to accuracy. Particularly, the literature reports its insights on the feature capturing — capturing app execution structure is more important than capturing sensitive flows. DroidCat proposes a promising solution to the main analysis challenges in Android ecosystem, such as reflection, obfuscation, and run-time permissions.

Detector enhancement. Another category of recent research especially focuses on the building of sustainable learning models to overcome the generalisation issue existing in machine learning based malware detection models. The generalisation issue could be considered as the continuation of the long period robustness problem (please refer to the efforts presented in Mariconti *et al.* [5] and Cai *et al.* [6] as discussed previously) when more and more machine learning approaches emerge in state-of-the-art anti-malware engines.

Based on the recent publications, machine learning models can achieve F1-scores up to 99%, which arguably solve the Android malware classification forever. However, as Android ecosystem, as well as the Android malware, continues to evolve, malware classifiers become less robust and even unsustainable. In this work [7], Jordaney *et al.* propose Transcend, a novel framework to determine whether a classification model is degraded to avoid poor performance. Transcend first compares two sets of samples, *i.e.*, the samples during deployment and the samples used for training, then evaluate the prediction quality of the classifier. Concretely, Transcend focuses on the identification of concept drift which could be a sound solution bridging the research gap caused by the evolving of malicious software.

Further, research by Pendlebury *et al.* [8] demonstrates two experimental bias, *i.e.*, spatial bias and temporal bias. The former one considers the gap between the distributions of a real-world deployment and the training/testing data; while the latter one focuses on the incorrect time splits of training and testing data sets (please refer to the research by Jordaney *et al.* [7] as they also focus on the generalisation issue existing in the machine learning-based Android malware classifiers over a long period, *e.g.*, concept drift).

A set of experiment constraints with respect to the two bias have been proposed by Pendlebury *et al.*. Further, a classifier robustness metric and an algorithm

(implemented as Tesseract, an evaluation framework comparing malware classifiers) to tune the classifier have been introduced, allowing the evaluation on mitigation strategies for time decay. Over 129,000 Android applications over three years have been involved in the performance evaluation, demonstrating that the earlier published results [4, 5] are biased.

My research: my recent research [36] measures the vulnerabilities of malware detectors. In this study, we present a framework that combines explainability and model-agnostic techniques to assess the effectiveness of malware detectors in the face of adversarial attacks. Our proposed framework introduces the concept of Accrued Malicious Magnitude (AMM) to identify the specific features of malware that should be manipulated to maximise the chances of evading detection. We apply this framework to evaluate the performance of various cutting-edge malware detectors in detecting manipulated malware samples.

2.2.2 Privacy Leaks and User Tracking

The collection of user privacy by mobile phone applications is common and necessary for the functions of some applications. However, it is illegal to directly or indirectly collect private information without the user's consent. Such illegal privacy theft is often With a high degree of stealthy, which is difficult for non-skilled users to detect. Therefore, many recent studies have conducted large-scale evaluations of mobile applications from several perspectives, such as location information exposure [9, 37, 38], personal identifiable information leakage [10, 39, 37, 40, 41, 42, 43, 44, 45, 46], and user tracking [11, 12, 47, 48, 49, 50, 51].

Mobile application privacy leakage. In the research by Xue *et al.* [9], privacy risk against the integrity of user anonymity has been exposed in a popular anonymous social media application which is susceptible to localisation attacks. Although the application itself neither shares user's location information nor provides indirectly location information, such as the distances between users, to any third-parties, an attack based on data collecting and supervised machine learning is demonstrated to be effective to predict the location of messages, without needing any reverse engineering of the application.

Specifically, an Android emulator is used to run the application with variable fake GPS locations generated by a third-party tool. Screenshots of the application are further automatically taken and the location-oriented information (*e.g.*,

people-nearby) are extracted. The attack is able to analyse and predict user's location based on the collected information. According to the experimental results, the accuracy of the prediction could be around 100 meters. In addition, the attack approach could also be applied to plenty of mobile applications that provide messages-nearby or people-nearby services, such as WeChat [37], which means that many apps may have the same potential localisation risks.

It is well known that mobile applications may leak user's information. Ren *et al.* [10] present ReCon, a corss-platform system, revealing personally identifiable information (PII) leaks in mobile applications. The proposed ReCon framework utilises network traffic and machine learning technologies. Specifically, ReCon first selects features and train a decision tree classifier with labelled network flows (with or without PII leaks). Notably, the classifier is retrained based on user feedback.

The evaluation on 100 most popular iOS, Android, and Windows Phone applications, as well as the user study that involves near 100 participants, indicates accurate and efficient performance of ReCon. ReCon opens a new area for mobile privacy research — the approach based on machine learning shows good accuracy and low overhead.

Other mobile application privacy research focuses on the tracking of user information, as many mobile applications are integrated with advertising or even maliciously embedded tracking services running in the background.

User tracking. In 2016, Englehardt and Narayanan [11] conducted a measurement of online tracking with 1 million sites, proposing an open-source web privacy measurement tool, OpenWPM. For the first time, this research observe the long tail distribution of online trackers and the trackers in the tail can only be found on very few sites. Researchers further quantify the impact of trackers and third parties. Such a measurement research is quite important as it makes measurement tools broadly available to minimally technically skilled analyst.

Wang *et al.* [12] investigate into China's mobile tracking behaviours, making interesting observations with respect to Advertising and Tracking Services (ATSs) popularity and community structure, user monopoly patterns, and personal private information collection. 28 billion access logs have been analysed. The researchers first identify the ATS domains and then associate ATS domains to applications, obtaining more than 190 million sessions, from which 4 million sessions are identified as containing only request from one application.

The measurement results discover a very well-established tracking network, where mobile trackers are interconnected and several tech giants in China track the majority of users. Further exploration could focuses on tracker detection, personal information collection, and the business relationships between mobile trackers (as they are well-connected).

My research: to measure the privacy practice in the wild, I have investigated into over 20K Android apps from the Google Play store, including apps that are designed particularly for children (Family apps) and apps that include children in their target user groups (Normal apps). Using both static and dynamic analysis, I measure the use of trackers and privacy leakages and found that: 3.46% Family apps request location permissions, even though collecting location information from children is forbidden by the Play store; and 13.08% of Family apps use trackers (which are not allowed in children apps). Even big players with 40+ kids apps on the Play store use ad trackers. Please find more details in Chapter 4.

2.2.3 Security and Privacy Analysis on Contact Tracing Applications During COVID-19

The pandemic of COVID-19 challenges the traditional manual contact tracing and many health authorities turn to digital contact tracing via mobile applications. Although most contact tracing applications are carefully design to preserve user's privacy as much as possible, considering the potential huge user population, even a tiny flaw may cause serious privacy disasters. Therefore, the security community has shown great interest in the security and privacy of mobile contact tracing applications [13, 52, 53, 54, 55, 14, 15, 56, 57, 58, 59, 16].

The research by Ferretti *et al.* [13] analysed the key parameters of pandemic spread and estimated the contribution of digital contact tracing, arguing that it is of crucial importance to control COVID-19 with instant digital contact tracing. In addition, they pointed out that, with digital contact tracing, the pandemic could be controlled without mass quarantines or lock-downs. However, just like a coin has two sides, the practical deployment of digital contact tracing is facing dramatic challenges, including security and privacy concerns.

Li and Guo [14] conduct a survey on the global deployment and challenges at the very beginning of the pandemic, analysing the strengths and weaknesses of two major forms of contact tracing application structures, *i.e.*, centralised

and decentralised. The study presents a geolocation mapping of the real-world deployment with respect to country, app name, installs, number of users, and the technique used (Bluetooth, GPS, QR code, and WiFi). In addition, vulnerabilities related to Bluetooth-based decentralised solutions are identified.

As pointed by the survey, compared with GPS, more digital contact tracing solutions (57% of) rely on Bluetooth technology, while a trade-off exists between data privacy and tracing effectiveness. Centralised and GPS solutions allow collect and trace the movement of population geographically, however, put user's data privacy at risk. On another hand, decentralised and non-GPS solutions protect data protection at a higher level, but less information will be provided to government or health authorities.

Several research focuses on particular contact tracing solutions, such as the Google/Apple Proposal [60], and further proposes new privacy-preserving approaches.

For example, Baumgartner *et al.* [15] demonstrate that the Google/Apple Proposal is vulnerable to several attacks that are practical in real-world, such as de-anonymizing attack and relay attack. Particularly, in downtown of the city of Darmstadt, Germany, researchers deployed 6 BLE sniffers in 6 sensitive places, such as city hall, place station, and clinic, and use the sniffers capture the Bluetooth token shared by the contact tracing users passing by. The experimental findings indicate that it is feasible to determine a user's whereabouts, including the locations they have visited, as well as the timing of their arrivals and departures from each location. Moreover, a specific type of relay attack known as a wormhole attack has been demonstrated. In this attack, an attacker captures messages at one physical location, transmits them through a network tunnel to another physical location, and subsequently retransmits the messages as if they originated from the second location originally. From the insights of this research, we can see that it is necessary to built up more sophisticated countermeasures to enhance the privacy-preserving in contact tracing applications.

Nevertheless, soon after the announcement of the Google/Apple specification, Gvili [16] analyses the security issues in the specification, finding that significant risk to society could be introduced by this specification. The research further proposes easy-to-adopt mitigation strategies for the risks. Several potential attacks have been analysed in this research, including

- *Power and storage drain attacks.* Power and storage drain attacks could be categorised as a kind of denial-of service attacks through sending large amount of messages. In such attack, a malicious attacker could send advertising messages to nearby devices. According to the Bluetooth specification, the contact tracing applications in nearby devices are forced to wake up to process each received message, which may lead to high power and storage cost, resulting in the opt out of using the contact tracing application by the user when she/he notice the drain.

- *Relay and replay attacks.* The relay attack is like the attack discussed in Baumgartner *et al.* [15] where an attacker record the messages and transfer it to another location and replay it to raise false alarms. While in a replay attack, the attacker records the messages and replay it at the later time, which is slightly different from the relay attack where a transfer to other locations is required.

- *Trolling attacks.* Different relay and replay attack, in trolling attack, the adversarial generates false messages to raise false positive alarms. The messages could be technically generated by hacking the contact tracing app, however, this could be difficult and requires professional skills or tools.

- *Linking attacks.* The linking attacks is similar to the profiling attack discussed in Baumgartner *et al.* [15], where an attacker can collect and mapping other user's information, such as the shared Bluetooth tokens and link the information to identify the user, especially when a user is diagnosed. The privacy of nearby persons is reduced due to the linkage of messages. Mitigation to linking attacks could be adding noise to the shared messages or consider a more privacy-preserving design, such as a decentralised system.

- *Tracking and de-anonymisation attacks.* In tracking and deanonymisation attacks, an attack may utilise devices to track other users' Bluetooth signal strength to determine the location information or infer locations over time. Mitigation to such attacks could be using variable Bluetooth signal with strength varies periodically or even randomly.

Although the paper focuses on the potential vulnerabilities in Google/Apple Proposal, kinds of attacks against contact tracing applications have been thoroughly discussed and analysed together with mitigation provided. The insights

presented in this research is inspiring and should always be considered during the development of any mobile applications.

My research: In my recent research [61], a false-claim attack has been discussed, where an attacker falsely claims that he/she has been diagnosed and report the diagnose status to the government or health authorities. Other users who have been contact with the adversarial will receive a false alarm. In our research, we argue that a contact tracing app should co-operate with health authorities tightly, *e.g.*, a positive report can only be upload to the system once there is a approval from health professionals. The relay attack has also been discussed and analysed in my recent research [61], which threatens most Bluetooth-based contact tracing solutions. Mitigation to relay and replay attacks could be adding temporal and spacial information into the Bluetooth tokens. A potential solution to relay attack is proposed in my recent research [62]. Please find more details in Chapter 3.

2.3 Static and Dynamic Analysis

Static analysis involves examining the code of an application without executing it, whereas dynamic analysis involves testing and evaluating an application during its run-time. Both static and dynamic analysis can effectively identify software defects, including memory and threading errors. These two approaches are complementary since each has its own strengths and limitations, and no single approach can uncover all types of errors. By utilising both static and dynamic analysis, a more comprehensive and thorough assessment of an application's quality and reliability can be achieved.

Static analysis examines all possible execution paths and variable values, including those that may not be invoked during actual execution. This makes static analysis capable of detecting errors that might not surface until much later, even weeks, months, or years after the application's release. This aspect of static analysis is particularly valuable in ensuring security, as security attacks often exploit an application in unforeseen and untested ways. On the other hand, dynamic analysis is useful in uncovering subtle defects or vulnerabilities that may have complex causes, making them difficult to detect through static analysis alone. While dynamic analysis can contribute to security assurance, its primary focus is on error detection and debugging. Table 2.2 presents several

Table 2.2. Recent research in security and privacy evaluations on mobile applications.

Research	Year	Domain	Description
Flowdroid [63]	2014	Static analysis	Pioneered in the static taint analysis on Android applications.
IccTA [64]	2015	Static analysis	Tracks flows of sensitive data from sources to sinks to identify the Inter-Component Communication (ICC) leaks.
Ausera [65]	2020	Static analysis	Propose an automated security risk assessment system, leveraging static analysis techniques and sensitive keyword identification, particularly focusing on data-ralted weakness detection.
Reardon *et al.* [66]	2019	Dynamic analysis	Utilise a dynamic approach exploring the Android permission system.
Tang *et al.* [67]	2020	Dyanmic analysis	Research into the security of network services of iOS applications with a dynamic approach.

cutting-edge studies in the field of static and dynamic analysis, highlighting their advancements in this area.

2.3.1 Static Taint Analysis

Static analysis is an automatic software analysis method which exam the source code looking for bugs or vulnerabilities without execution the code. It has been widely used in mobile industry, as well as security and privacy research domain [63, 64, 68, 65, 69, 70, 71].

Flowdroid [63] pioneered in the static taint analysis on Android applications, which is proposed by Arzt *et al.* in 2014. In taint analysis, potential malicious data flows are tracked from sources (where the sensitive "tained" information is obtained or accessed) to sinks (the leaking destination the sensitive information has been transferred to). Once a data flow containing sensitive information flows from a source to a sink, a privacy leak could be determined, as sinks are methods where the information could be shared, *e.g.*, a socket, message box, or broadcasting.

As a static taint analysis tool, Flowdroid provides a precise model of Android lifecycle, which handles callbacks invoded by the Android framework and successfully reduce the false positives. According to the experimental results, FlowDroid achieves 93% recall and 85% precision, outperforming other tools at that time. Android applications have many entry points, instead of a Main method, which are called by the Android framework, Which challenges the taint analysis on Android applications. Creatively, FlowDroid handle this problem and modelled the Android lifecycle with customise dummy main methods.

Several studies utilise such static analysis technology to evaluate the privacy among Android applications, looking for potential privacy leaks. For example,

in the IccTA proposed by Li *et al.* [64], flows of sensitive data have been tracked from sources to sinks to identify the Inter-Component Communication (ICC) leaks, *i.e.* privacy leaks between different components in Android applications. IccTA resolves the ICC problem by directly connecting the discontinuities of Android applications at the code level. Concretely, IccTA extracts the ICC links from Dalvik bytecode, then modifies the Jimple representation and enable data-flow analysis between components through connecting the components directly. At last, the taint analysis is done by a modified version of FlowDroid. The experiment on 15,000 real-world applications detects 2,395 ICC leaks in 377 applications.

Chen *et al.* [65] propose an automated security risk assessment system, Ausera, leveraging static analysis techniques and sensitive keyword identification, particularly focusing on data-ralted weakness detection. The framework of Ausera consists of three phases. The first phase is the tagging of sensitive data, in which Ausera identifies sensitive data in the user inputs and the content in user interface, and tag the sensitive data related variable according to the keyword database. The the functions related to data leakage are further identified according to a list of sinks which contains 106 manually defined vulnerable sinks.

At last, a forward taint analysis is conducted to detect potential privacy leaks using the tagged sensitive data as sources. Ausera detects 2,157 weaknesses in 693 real-world banking applications across 83 countries. Notably, 126 weaknesses have been confirmed by 21 banks. The research team also helps 7 banks, including HSBC in UK and OCBC in Singapore to improve the security of their applications.

The global distribution of weaknesses among banking applications has also been presented. The analysis shows that only 0.27 data leak weakness per application has been detected in 143 banking applications from Europe and USA, which may be attributed to the financial regulations and development guidelines, such as GDPR, which adopt strict security and privacy requirements, affecting the implementation of banking applications. Further, the development budget and developers' expertise are also considered as factors leading to weaknesses. The security status of banking applications is varied across different countries, mainly because of the shaping of economies and regulations.

2.3.2 Dynamic Network Traffic Analysis

The static analysis suffers from false positives, as it analyses the source code, rather than actually executing the applications. The run-time behaviour could be different from the static analysis outcome because there could exist dummy or dead code that will never be executed. In addition, run-time application behaviours could also remotely run on a server or be provided as a cloud service. To detect the run-time application behaviours, dynamic analysis approaches are introduced in recent research. However, dynamic analysis also has its shortcomings, such as the code coverage, *e.g.*, not all code can be triggered during the testing. Here I introduce and review some of the state-of-the-art dynamic analysis research [66, 67, 72, 73] in mobile application security and privacy domain.

Reardon *et al.* [66] utilise a dynamic approach exploring the Android permission system. They dynamically run the applications with an instrumented Android environment to monitor the run-time behaviour and network traffic. Two types of circumvention are reported: (*i*) circumvention through covert channel and (*ii*) circumvention through side channel.

In covert channel cases, communication between two mobile applications transfers information that is unauthorised to share. For example, apps may share sensitive information through the access to SD card: app A obtains sensitive information with a specific permission and write the information in hidden files on SD card, while app B can read it from the SD card without a permission specifically requesting the access to the sensitive information. In side channel cases, privileged information is obtained by an application without permission checking. For instance, an SDK is found exploiting side channels to share hashed MAC address, which may enable a MAC-address-to-location mapping and leak the user's location information. However, due the limitations of dynamic analysis, the findings of this research can only provide a lower bound of privacy leakages through the covert channel and side channel.

Tang *et al.* [67] research into the security of network services of iOS applications with a dynamic approach. They first develop a scalable application collection tool which allows them download 168,951 iOS applications. Then the researchers evaluate the characteristics of network service vulnerabilities in 1,300 applications, discovering 11 vulnerabilities in popular iOS applications. Based on the findings, they create signatures for large-scale analysis and further

look into the 168,951 applications dataset. From 92 applications, third-party libraries are reported to listen remote connections and open up the iOS device to a host of possible attacks, such as remote command execution, data leakage, and denial-of-service attacks. Although the research focuses on iOS platform which could be quite different from Android, the dynamic analysis approach is potentially to be applied on Android platform, *e.g.*, some of the vulnerabilities could also exist in Android applications.

My research: In my recent research [74], we look into the privacy practice in Android clipboards and conduct a thorough automated assessment on popular Android applications. Particularly, we analyse the clipboard access mechanism in Android system and propose a dynamic detection method to check whether an application accesses the clipboard and leaks sensitive information. We first identify the clipboard access by statically scanning the source code and further conduct a taint analysis to check whether there is sensitive information obtained by the clipboard accessing methods. The findings are further manually checked by a call graph analysis backtracking the data flows from entry points to sinks. Then dynamic analysis is conducted by hooking clipboard access methods during run-time to confirm the data leaks. We believe such semi-automated assessment method could overcome the shortcomings of static and dynamic analysis, *i.e.*, detecting potential privacy leaks through static analysis and confirm the run-time behaviours with dynamic analysis. However, although our method is able to achieve a low false positive rate, the false negatives are still existing.

2.4 Privacy Regulations and Assertions Compliance

Restrictions on mobile app behaviour go both ways. Privacy regulations and app store policies define legal mobile app behaviour and user rights from top-down. If the behaviour of the mobile application violates the relevant privacy regulations, it will be subject to corresponding penalties, such as fines, compensation or application removal. A privacy policy is a statement in simple language explaining how an organisation or institution handles users personal information. However, since mobile application developers are often not professional legal professionals, and it is difficult for ordinary users to fully understand the complex privacy policy terms (although they are required to use simple language), the specific behaviour of mobile applications is likely to violate the requirements

Table 2.3. Recent research in privacy policy quality and regulation compliance.

Research	Year	Domain	Description
Massey *et al.* [78]	2013	Privacy policy	Focus on the automatic analysis on the requirements in policy documents.
Yu *et al.* [75]	2016	Privacy policy	Automatically identifies three kinds of problems in privacy policy and conduct the first systematic study on privacy policy.
Amos *et al.* [79]	2021	Privacy policy	Develop a crawler that extracts archived privacy policies and analyse the transparency and accessibility of privacy policies.
Qu *et al.* [77]	2014	Privacy policy	Measure the description-to-permission fidelity in Android applications from Google Play store, utilising the natural-language descriptions and the permissions list provided by the store.
Xue *et al.* [83]	2016	Regulation compliance	Look into the consequences of violation RTBF, and the susceptibility to inference attacks.
Sorensen *et al.* [84]	2019	Regulation compliance	Examine the changes in third party presence and mapping the shifts in third party topology before and after GDPR.
Liu *et al.* [86]	2021	Regulation compliance	Detect compliance issues among regulations with privacy policies, and provide intuitive results for data subjects, data collection party, and the regulatory authorities.
Reyes *et al.* [87]	2017	Regulation compliance	Develop an automatic method, combining dynamic analysis with network traffic monitoring, to evaluate the mobile app's COPPA compliance.

defined in privacy regulations and assertions, which has aroused the interest of privacy research community. Privacy policy quality assessment [75, 76, 77], interpretation of privacy policy [78, 79, 77, 80], and application behaviour analysis [81, 82, 83, 84, 85, 86, 87] have become hot topics in recent years. Table 2.3 illustrates several recent studies in this research area.

2.4.1 Privacy Policy Quality and Compliance

Several recent works focused on the privacy policy quality and compliance, *e.g.*, to compare the application behaviours with the claimed privacy assertions, including privacy policies and permission request list.

Massey *et al.* [78] focus on the automatic analysis on the requirements in policy documents, *e.g.* policy documents from popular websites and large multinational corporations. Specifically, they analyse the readability of policy documents; determine that automated text mining enables the identification of software requirements expressed as privacy protections or vulnerabilities in a policy document; and provide preliminary support for the generalisability to multiple domains.

In addition, probabilistic topic models are developed to uncover hidden themes within extensive collections of documents. This approach enables analysis of scenarios that would be otherwise challenging to assess through manual annotation alone. Specifically, Latent Dirichlet Allocation (LDA) topic modelling is employed to reveal the underlying topics within the document collection. To

determine the optimal number of topics, denoted as K, a total of 35 topic models are constructed. Each model's predictive likelihood, measured by perplexity, is computed. Perplexity is a useful metric that can be applied to evaluate topic models. The experimental findings demonstrate that models with lower perplexity values on the held-out dataset exhibit stronger performance, indicating their ability to capture the latent structure of the data effectively.

It could be difficult for non-professional users to understand an application's behaviours through the required permissions. To figure out the potential privacy risks and address users' concerns, more and more applications are required to release privacy policies written in natural language. Therefore, whether these privacy policies are trustworthy or not is critical. It is worth noting that serious security and privacy issues could be raised due to a questionable privacy policy. PPChecker, a novel approach proposed by Yu *et al.* [75], automatically identifies three kinds of problems in privacy policy and conduct the first systematic study on privacy policy.

PPChecker is a tool designed to assess the quality of an app's privacy policy, description, APK file, and the privacy policies of its third-party libraries. It focuses on examining whether the privacy policy is incomplete, incorrect, or inconsistent. To evaluate the effectiveness of PPChecker, real-world applications and privacy policies are used. The experimental evaluation demonstrates that PPChecker is highly effective in identifying questionable privacy policies, achieving a high level of precision. In total, 282 apps were flagged for potential problems or vulnerabilities. This study serves as inspiration for further research aimed at improving and regulating the privacy policies of apps, ensuring better protection of user privacy.

The automated analysis of privacy policies has emerged as a fruitful area of research, as evidenced by studies such as those conducted by Amos *et al.* [79] and Qu *et al.* [77]. In their work, Amos *et al.* develop a web crawler that extracts archived privacy policies from the Internet Archive's Wayback Machine, enabling the analysis of transparency and accessibility in privacy policies. Their research reveals several noteworthy findings. First, privacy policies frequently fail to disclose the presence of tracking technologies, indicating a lack of transparency. Second, privacy policies have become increasingly challenging to read, suggesting reduced accessibility for users. Lastly, self-regulation efforts for third parties have increased, but online advertising trade associations dominate this space. These findings shed light on the impact of the General Data Protection

Regulation (GDPR) on privacy policies, demonstrating its historical significance in shaping privacy practices. This research contributes significantly to the field of privacy research, providing valuable insights into the state of privacy policies and their evolution over time.

Qu *et al.* [77] measure the description-to-permission fidelity, *i.e.*, the descriptions should reflect the need for permissions, in Android applications from Google Play store, utilising the natural-language descriptions and the permissions list provided by the store. An automatic description-to-permission fidelity assessment system, AutoCog, has been proposed, employing state-of-the-art natural language processing techniques and a newly proposed machine learning-based algorithm to investigate the relationship between app description and requested permissions. AutoCog greatly outperforms other related works in both detection performance and generalization ability to various permissions. A large-scale measurements over 45K applications demonstrates that the problem of low description-to-permission fidelity is serious and prevalent. The measurement in this study shows a generally weak description-to-permissions fidelity on the Google Play store.

2.4.2 Regulation Compliance

Another set of research focuses on the compliance of regulations, aiming to expose violations against GDPR or other privacy regulations. The privacy policy serves as a vital interface that allows users to gain an understanding of the collection and usage of their personal information. In light of the increasing importance of data privacy protection as a crucial societal concern, numerous laws and regulations have been implemented in various countries and regions. These legal frameworks aim to safeguard individuals' privacy rights and ensure responsible handling of personal data.

For example, the recent "Right to be Forgotten" (RTBF) ruling (which enable an individual to ask Google and other search engines to delist links to web pages that contain information about that individual) has been studied by Xue *et al.* [83]. In this research, a data-driven approach has been applied to study the right to be forgotten in the traditional media outlets. The research further looks into the consequences of violation RTBF, and the susceptibility to inference attacks.

In this study, a comprehensive analysis has been conducted on over two hundred UK media pages that were known to be removed. The analysis involved a combination of manual investigation and Latent Dirichlet Allocation (LDA) to uncover insights. The authors also demonstrated how a third party can still uncover delisted URLs and the names of the requesters. Through experiments, the researchers were able to establish the link between previously unknown delisted URLs and the individuals who requested their removal. To assess the presence or absence of the Streisand effect, the study developed novel metrics and methodologies utilising publicly available datasets, such as Google Trends and Twitter data. Additionally, a demographic analysis of the known requesters was performed. The results and observations of this research can provide valuable insights to lawmakers as they continue to refine right to be forgotten laws in the future. By shedding light on the outcomes and implications of delisting requests, this study contributes to the ongoing discussions surrounding privacy and data protection.

Sorensen *et al.* [84] conduct an eight months longitudinal study on more than one thousand popular websites in Europe and US, examining the changes in third party presence and mapping the shifts in third party topology before and after GDPR. They find that the developments of third party vary in the numbers and types in different categories of websites and countries. Based on the analysis of the number of third parties over time, this research highlights a notable decline in the presence of third parties on websites, suggesting that the implementation of GDPR has resulted in reduced third-party activity. The study reveals significant variations between privately owned websites and publicly owned websites in terms of third-party involvement. Privately owned websites tend to have a higher number of embedded third parties, while publicly owned websites exhibit a relatively lower but steadily increasing presence of third parties. These findings provide valuable insights into the impact of GDPR on the dynamics of third-party engagement, emphasising the contrasting patterns observed across different types of website ownership.

Liu *et al.* [86] have developed a method to identify compliance issues between regulations and privacy policies. Their approach aims to address the challenge of analysing compliance between GDPR (Article 13) and privacy policies. The method involves two main steps: sentence classification and rule-based analysis. To train their system, the researchers collected and analysed a dataset of

36,610 labelled sentences from 304 privacy policies. Using a rule-based analysis, they were able to identify compliance issues. Additionally, they conducted a user study to assess the usability of their approach. The researchers implemented a web-based tool called AutoCompliance, which successfully detected 1,180 compliance issues. The tool provides intuitive results for data subjects, data collection parties, and regulatory authorities, thereby assisting in compliance analysis between regulations and privacy policies.

In the mobile world, there has been a significant growth in the market of games and learning apps designed for children. However, ensuring compliance with regulations that safeguard children's privacy, such as COPPA, remains a crucial concern. Many of these seemingly "free" mobile apps have the ability to access sensitive personal information. To address this issue, Reyes *et al.* [87] have introduced an automatic method that combines dynamic analysis with network traffic monitoring. Their approach enables the evaluation of mobile apps' compliance with COPPA. By conducting empirical technical observations and analysing the corresponding privacy policies, they provide a comprehensive assessment of the apps' adherence to the legal requirements. This method offers a valuable tool for assessing and ensuring COPPA compliance in the context of the growing market of games and learning apps for children in the mobile industry.

The study utilises a modified version of the Android OS, which allows for the execution of apps while recording their handling of sensitive information. This approach enables the researchers to closely monitor the apps' access to and transmission of such data. To analyze third-party involvement, the researchers leverage a crowdsourced dataset collected by the Lumen Privacy Tool [88]. This dataset provides valuable insights into the activities of various organisations, including advertising networks. The primary objective of this research is to shed light on the compliance of apps with COPPA regulations and to create a comprehensive catalog of organisations that collect sensitive user information. Preliminary results indicate the presence of several potential violations of COPPA. These violations include instances where prior consent is omitted and the active sharing of persistent identifiers with third-party services for tracking and profiling children. Through this study, the aim is to raise awareness about potential COPPA violations and emphasize the need for stricter adherence to privacy regulations in the context of children's apps.

2.5 Conclusion

This chapter conducts a comparative analysis of existing literature of mobile security and privacy-related topics. In general, while there has been a lot of research focusing on the security and privacy of mobile applications, the behaviour of mobile applications is kaleidoscopic. In specific application scenarios, more fine-grained analysis tools are required to effectively detect security risks and privacy leaks. At the same time, with the application of new technologies such as Bluetooth, virtual reality, and 5G communication on smartphones, new attack surfaces and threat models have emerged, and the privacy theft have become more stealthy. These scrutiny of mobile application security and privacy pose new challenges, and also require more attention from mobile application developers, mobile phone users, and legislators pay more attention on the understanding and measuring privacy and security assertions of mobile apps.

Bibliography

[1] Shuqing Li, Yechang Wu, Yi Liu, Dinghua Wang, Ming Wen, Yida Tao, Yulei Sui, and Yepang Liu. An exploratory study of bugs in extended reality applications on the web. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 172–183. IEEE, 2020.

[2] Hsu Myat Win, Shin Hwei Tan, and Yulei Sui. Event-aware precise dynamic slicing for automatic debugging of android applications. *Journal of Systems and Software*, 198:111606, 2023.

[3] Sen Chen, Minhui Xue, Zhushou Tang, Lihua Xu, and Haojin Zhu. Stormdroid: A streaminglized machine learning-based system for detecting android malware. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 377–388, 2016.

[4] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[5] Enrico Mariconti, Lucky Onwuzurike, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon Ross, and Gianluca Stringhini. Mamadroid: Detecting android malware by building markov chains of behavioral models. *arXiv preprint arXiv:1612.04433*, 2016.

[6] Haipeng Cai, Na Meng, Barbara Ryder, and Daphne Yao. Droidcat: Effective android malware detection and categorization via app-level profiling. *IEEE Transactions on Information Forensics and Security*, 14(6):1455–1470, 2018.

[7] Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 625–642, 2017.

[8] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 729–746, 2019.

[9] Minhui Xue, Cameron Ballard, Kelvin Liu, Carson Nemelka, Yanqiu Wu, Keith Ross, and Haifeng Qian. You can yak but you can't hide: Localizing anonymous social network users. In *Proceedings of the 2016 Internet Measurement Conference*, pages 25–31, 2016.

[10] Jingjing Ren, Ashwin Rao, Martina Lindorfer, Arnaud Legout, and David Choffnes. Recon: Revealing and controlling pii leaks in mobile network traffic. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 361–374, 2016.

[11] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1388–1401, 2016.

[12] Zhaohua Wang, Zhenyu Li, Minhui Xue, and Gareth Tyson. Exploring the eastern frontier: A first look at mobile app tracking in China. In *Passive and Active Measurement*, pages 314–328, 2020.

[13] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science*, 2020.

[14] Jinfeng Li and Xinyi Guo. COVID-19 contact-tracing apps: A survey on the global deployment and challenges. *arXiv preprint arXiv:2005.03599*, 2020.

[15] Lars Baumgärtner, Alexandra Dmitrienko, Bernd Freisleben, Alexander Gruler, Jonas Höchst, Joshua Kühlberg, Mira Mezini, Markus Miettinen, Anel Muhamedagic, Thien Duc Nguyen, et al. Mind the gap: Security & privacy risks of contact tracing apps. *arXiv preprint arXiv:2006.05914*, 2020.

[16] Yaron Gvili. Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc. Technical report, Cryptology ePrint Archive, Report 2020/428, 2020.

[17] Yuhao Gao, Guoai Xu, Li Li, Xiapu Luo, Chenyu Wang, and Yulei Sui. Demystifying the underground ecosystem of account registration bots. In

Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pages 897–909, 2022.

[18] Joshua Garcia, Mahmoud Hammad, and Sam Malek. Lightweight, obfuscation-resilient detection and family identification of android malware. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 26(3):1–29, 2018.

[19] Haipeng Cai. Assessing and improving malware detection sustainability through app evolution studies. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 29(2):1–28, 2020.

[20] Vitor Monte Afonso, Matheus Favero de Amorim, André Ricardo Abed Grégio, Glauco Barroso Junquera, and Paulo Lício de Geus. Identifying android malware using dynamically obtained features. *Journal of Computer Virology and Hacking Techniques*, 11(1):9–17, 2015.

[21] Vitalii Avdiienko, Konstantin Kuznetsov, Alessandra Gorla, Andreas Zeller, Steven Arzt, Siegfried Rasthofer, and Eric Bodden. Mining apps for abnormal usage of sensitive data. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 426–436. IEEE, 2015.

[22] Guillermo Suarez-Tangil, Santanu Kumar Dash, Mansour Ahmadi, Johannes Kinder, Giorgio Giacinto, and Lorenzo Cavallaro. Droidsieve: Fast and accurate classification of obfuscated android malware. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 309–320, 2017.

[23] Xiaohan Zhang, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 757–770, 2020.

[24] Hamish Spencer, Wei Wang, Ruoxi Sun, and Minhui Xue. Dissecting malware in the wild. In *Australasian Information Security Conference*, 2022.

[25] Matthew Crawford, Wei Wang, Ruoxi Sun, and Minhui Xue. Statically detecting adversarial malware through randomised chaining. In *Australasian Information Security Conference*, 2022.

[26] Yifan Zhou, Zhendong Shi, and Ruoxi Sun. Visualization and attack prevention for a sensor-based agricultural monitoring system. In *Australasian Information Security Conference*, 2022.

[27] Minhui Xue, Surya Nepal, Ling Liu, Subbu Sethuvenkatraman, Xingliang Yuan, Carsten Rudolph, Ruoxi Sun, and Greg Eisenhauer. RAI4IoE: Responsible AI for enabling the Internet of Energy. In *The Fifth IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (IEEE TPS-ISA)*, 2024.

[28] Hongsheng Hu, Shuo Wang, Jiamin Chang, Haonan Zhong, Ruoxi Sun, Shuang Hao, Haojin Zhu, and Minhui Xue. A duty to forget, a right to be assured? Exposing vulnerabilities in machine unlearning services. In *the Network and Distributed System Security Symposium (NDSS)*, 2024.

[29] Yuxin Cao, Xi Xiao, Ruoxi Sun, Derui Wang, Minhui Xue, and Sheng Wen. Stylefool: Fooling video classification systems via style transfer. In *44th IEEE Symposium on Security and Privacy (IEEE S&P)*, 2023.

[30] Shuo Wang, Sharif Abuadbba, Sidharth Agarwal, Kristen Moore, Ruoxi Sun, Minhui Xue, Surya Nepal, Seyit Camtepe, and Salil Kanhere. Publiccheck: Public watermarking verification for deep neural networks. In *44th IEEE Symposium on Security and Privacy (IEEE S&P)*, pages 1239–1256, 2023.

[31] Wanlun Ma, Derui Wang, Ruoxi Sun, Minhui Xue, Sheng Wen, and Yang Xiang. The "Beatrix" resurrections: Robust backdoor detection via Gram matrices. In *the Network and Distributed System Security Symposium (NDSS)*, 2023.

[32] Shuo Wang, Mahathir Almashor, Alsharif Abuadbba, Ruoxi Sun, Minhui Xue, Calvin Wang, Raj Gaire, Seyit Camtepe, and Surya Nepal. DOITRUST: Dissecting on-chain compromised internet domains via graph learning. In *the Network and Distributed System Security Symposium (NDSS)*, 2023.

[33] Yanjun Zhang, Guangdong Bai, Mahawaga Arachchige Pathum Chamikara, Mengyao Ma, Liyue Shen, Jingwei Wang, Surya Nepal, Minhui Xue, Long Wang, and Joseph Liu. Agrevader: Poisoning membership inference against byzantine-robust federated learning. In *Proceedings of the ACM Web Conference 2023*, pages 2371–2382, 2023.

[34] Aoting Hu, Renjie Xie, Zhigang Lu, Aiqun Hu, and Minhui Xue. Tablegan-mca: Evaluating membership collisions of gan-synthesized tabular data releasing. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2096–2112, 2021.

[35] Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. Hidden backdoors in human-centric language models. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3123–3140, 2021.

[36] Ruoxi Sun, Minhui Xue, Gareth Tyson, Tian Dong, Shaofeng Li, Shuo Wang, Haojin Zhu, Seyit Camtepe, and Surya Nepal. Mate! Are you really aware? An explainability-guided testing framework for robustness of malware detectors. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2023.

[37] Rongrong Wang, Minhui Xue, Kelvin Liu, and Haifeng Qian. Data-driven privacy analytics: A wechat case study in location-based social networks. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 561–570. Springer, 2015.

[38] Haizhong Zheng, Minhui Xue, Hao Lu, Shuang Hao, Haojin Zhu, Xiaohui Liang, and Keith Ross. Smoke screener or straight shooter: Detecting elite sybil attacks in user-review social networks. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS 2018)*, 2018.

[39] Matthew Joslin, Neng Li, Shuang Hao, Minhui Xue, and Haojin Zhu. Measuring and analyzing search engine poisoning of linguistic collisions. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1311–1325. IEEE, 2019.

[40] Haoyu Wang, Zhe Liu, Jingyue Liang, Narseo Vallina-Rodriguez, Yao Guo, Li Li, Juan Tapiador, Jingcun Cao, and Guoai Xu. Beyond google play: A large-scale comparative study of chinese android app markets. In *Proceedings of the Internet Measurement Conference 2018*, pages 293–307, 2018.

[41] Jingjing Ren, Daniel J Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach.

In *Proceedings of the Internet Measurement Conference*, pages 267–279, 2019.

[42] Chaoran Li, Xiao Chen, Ruoxi Sun, Minhui Xue, Sheng Wen, Muhammad Ejaz Ahmed, Seyit Camtepe, and Yang Xiang. Cross-language Android permission specification. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, page 772–783, 2022.

[43] Tian Dong, Shaofeng Li, Guoxing Chen, Minhui Xue, Haojin Zhu, and Zhen Liu. Rai2: Responsible identity audit governing the artificial intelligence. In *Proceedings of the 30th Annual Network and Distributed System Security Symposium (NDSS 2023)*, 2023.

[44] Yuantian Miao, Xue Minhui, Chao Chen, Lei Pan, Jun Zhang, Benjamin Zi Hao Zhao, Dali Kaafar, and Yang Xiang. The audio auditor: user-level membership inference in internet of things voice services. *Proceedings on Privacy Enhancing Technologies*, 2021:209–228, 2021.

[45] Pingyi Hu, Zihan Wang, Ruoxi Sun, Hu Wang, and Minhui Xue. M^4I: Multi-modal models membership inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[46] Zihan Wang, Olivia Byrnes, Hu Wang, Ruoxi Sun, Congbo Ma, Huaming Chen, Qi Wu, and Minhui Xue. Data hiding with deep learning: A survey unifying digital watermarking and steganography. *IEEE Transactions On Computational Social Systems*, 2023.

[47] Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt. Third party tracking in the mobile ecosystem. In *Proceedings of the 10th ACM Conference on Web Science*, pages 23–31, 2018.

[48] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, Phillipa Gill, et al. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS 2018)*, 2018.

[49] Suibin Sun, Le Yu, Xiaokuan Zhang, Minhui Xue, Ren Zhou, Haojin Zhu, Shuang Hao, and Xiaodong Lin. Understanding and detecting mobile Ad fraud through the lens of invalid traffic. In *Proceedings of the 2021 ACM*

SIGSAC Conference on Computer and Communications Security, pages 287–303, 2021.

[50] Tong Zhu, Yan Meng, Haotian Hu, Xiaokuan Zhang, Minhui Xue, and Haojin Zhu. Dissecting click fraud autonomy in the wild. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[51] Ruoxi Sun, Minhui Xue, Gareth Tyson, Seyit Camtepe, and Surya Nepal. Not seen, not heard in the digital world! Measuring privacy practices in children's apps. In *The Web Conference (WWW)*, 2023.

[52] Bill Chappell. Coronavirus: sacramento county gives up on automatic 14-day quarantines. 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, 2020.

[53] Siddique Latif, Muhammad Usman, Sanaullah Manzoor, Waleed Iqbal, Junaid Qadir, Gareth Tyson, Ignacio Castro, Adeel Razi, Maged N. Kamel Boulos, Adrian Weller, and Jon Crowcroft. Leveraging Data Science To Combat COVID-19: A Comprehensive Review. *IEE Transactions on Artificial Intelligence*, 2020.

[54] Kobi Leins, Chris Culnane, and Benjamin IP Rubinstein. Tracking, tracing, trust: Contemplating mitigating the impact of COVID-19 through technological interventions. *The Medical Journal of Australia*, page 1, 2020.

[55] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. Decentralized privacy-preserving proximity tracing. *arXiv preprint arXiv:2005.12273*, 2020.

[56] Joseph K Liu, Man Ho Au, Tsz Hon Yuen, Cong Zuo, Jiawei Wang, Amin Sakzad, Xiapu Luo, and Li Li. Privacy-preserving COVID-19 contact tracing app: a zero-knowledge proof approach.

[57] Ren He, Haoyu Wang, Pengcheng Xia, Liu Wang, Yuanchun Li, Lei Wu, Yajin Zhou, Xiapu Luo, Yao Guo, and Guoai Xu. Beyond the virus: A first look at Coronavirus-themed mobile malware. *arXiv preprint arXiv:2005.14619*, 2020.

[58] Haohuang Wen, Qingchuan Zhao, Zhiqiang Lin, Dong Xuan, and Ness Shroff. A study of the privacy of COVID-19 contact tracing apps. *International Conference on Security and Privacy for Communication Networks*, 2020.

[59] Hyunghoon Cho, Daphne Ippolito, and Yun William Yu. Contact tracing mobile apps for COVID-19: Privacy considerations and related trade-offs. *arXiv preprint arXiv:2003.11511*, 2020.

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

[61] Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, Seyit Camtepe, and Damith C Ranasinghe. An empirical assessment of global covid-19 contact tracing applications. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1085–1097. IEEE, 2021.

[62] Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, and Damith C. Ranasinghe. Venuetrace: A privacy-by-design covid-19 digital contact tracing solution: Poster abstract. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys)*, page 790–791, 2020.

[63] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[64] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. Iccta: Detecting inter-component privacy leaks in android apps. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 280–291. IEEE, 2015.

[65] Sen Chen, Lingling Fan, Guozhu Meng, Ting Su, Minhui Xue, Yinxing Xue, Yang Liu, and Lihua Xu. An empirical assessment of security risks of global Android banking apps. In *Proceedings of 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1310–1322, 2020.

[66] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge 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)*, pages 603–620, 2019.

[67] Zhushou Tang, Ke Tang, Minhui Xue, Yuan Tian, Sen Chen, Muhammad Ikram, Tielei Wang, and Haojin Zhu. iOS, your OS, everybody's OS: Vetting and analyzing network services of iOS applications. In *29th USENIX Security Symposium*, pages 2415–2432, 2020.

[68] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29, 2014.

[69] Sen Chen, Lingling Fan, Chunyang Chen, Ting Su, Wenhe Li, Yang Liu, and Lihua Xu. Storydroid: Automated generation of storyboard for android apps. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 596–607. IEEE, 2019.

[70] Fengguo Wei, Sankardas Roy, and Xinming Ou. Amandroid: a precise and general inter-component data flow analysis framework for security vetting of android apps. *ACM Transactions on Privacy and Security*, 21(3):1–32, 2018.

[71] Michael I Gordon, Deokhwan Kim, Jeff H Perkins, Limei Gilham, Nguyen Nguyen, and Martin C Rinard. Information flow analysis of android applications in droidsafe. In *NDSS*, volume 15, page 110, 2015.

[72] Sen Chen, Ting Su, Lingling Fan, Guozhu Meng, Minhui Xue, Yang Liu, and Lihua 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*, pages 797–802, 2018.

[73] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé Bissyandé, and Jacques Klein. MadDroid: Characterizing and detecting devious Ad contents for Android apps. In *Proceedings of The Web Conference 2020*, pages 1715–1726, 2020.

[74] Wei Wang, Ruoxi Sun, Minhui Xue, and Damith C Ranasinghe. An automated assessment of Android clipboards. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020.

[75] Le Yu, Xiapu Luo, Xule Liu, and Tao Zhang. Can we trust the privacy policies of android apps? In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 538–549. IEEE, 2016.

[76] Le Yu, Tao Zhang, Xiapu Luo, Lei Xue, and Henry Chang. Toward automatically generating privacy policy for android apps. *IEEE Transactions on Information Forensics and Security*, 12(4):865–880, 2016.

[77] Zhengyang Qu, Vaibhav Rastogi, Xinyi Zhang, Yan Chen, Tiantian Zhu, and Zhong Chen. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1354–1365, 2014.

[78] Aaron K Massey, Jacob Eisenstein, Annie I Antón, and Peter P Swire. Automated text mining for requirements analysis of policy documents. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 4–13. IEEE, 2013.

[79] Ryan Amos, Gunes Acar, Eli Lucherini, Mihir Kshirsagar, Arvind Narayanan, and Jonathan Mayer. Privacy policies over time: Curation and analysis of a million-document dataset. In *Proceedings of the Web Conference 2021*, pages 2165–2176, 2021.

[80] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. Policylint: investigating internal privacy policy contradictions on google play. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 585–602, 2019.

[81] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G Shin, and Karl Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 531–548, 2018.

[82] Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D Breaux, and Jianwei Niu. Toward

a framework for detecting privacy policy violations in android application code. In *Proceedings of the 38th International Conference on Software Engineering*, pages 25–36. ACM, 2016.

[83] Minhui Xue, Gabriel Magno, E Landulfo Teixeira P Cunha, Virgilio Almeida, and Keith W Ross. The right to be forgotten in the media: A data-driven study. In *Proceedings on Privacy Enhancing Technologies*, pages 389–402, 2016.

[84] Jannick Sørensen and Sokol Kosta. Before and after gdpr: The changes in third party presence at public and private european websites. In *The World Wide Web Conference*, pages 1590–1600, 2019.

[85] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel Reidenberg, N Cameron Russell, and Norman Sadeh. Maps: Scaling privacy compliance analysis to a million apps. *Proceedings on Privacy Enhancing Technologies*, 2019(3):66–86, 2019.

[86] Shuang Liu, Baiyang Zhao, Renjie Guo, Guozhu Meng, Fan Zhang, and Meishan Zhang. Have you been properly notified? automatic compliance analysis of privacy policy text with gdpr article 13. In *Proceedings of the Web Conference 2021*, pages 2154–2164, 2021.

[87] Irwin Reyes, Primal Wijesekera, Abbas Razaghpanah, Joel Reardon, Narseo Vallina-Rodriguez, Serge Egelman, Christian Kreibich, et al. " is our children's apps learning?" automatically detecting coppa violations. In *Workshop on Technology and Consumer Protection (ConPro 2017), in conjunction with the 38th IEEE Symposium on Security and Privacy (IEEE S&P 2017)*, 2017.

[88] Abbas Razaghpanah, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Christian Kreibich, Phillipa Gill, Mark Allman, and Vern Paxson. Haystack: In situ mobile traffic analysis in user space. *arXiv preprint arXiv:1510.01419*, pages 1–13, 2015.

Chapter 3

Privacy Assertions Compliance Evaluation

The work contained in this chapter has been published in proceedings of the 43rd International Conference on Software Engineering (ICSE).

The final publication is available at:
https://doi.org/10.1109/ICSE43902.2021.00101

Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, Surya Camtepe, and Damith Ranasinghe. An Empirical Assessment of Global COVID-19 Contact Tracing Applications. In *proceedings of the 43rd International Conference on Software Engineering (ICSE, CORE A*), Technical Track*, 2021.

Statement of Authorship

Title of Paper	An Empirical Assessment of Global COVID-19 Contact Tracing Applications
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, Seyit Camtepe, and Damith Ranasinghe, An Empirical Assessment of Global COVID-19 Contact Tracing Applications. *The 43rd International Conference on Software Engineering (ICSE, CORE A*), Technical Track*, 2021.

Principal Author

Name of Principal Author (Candidate)	Ruoxi Sun		
Contribution to the Paper	Built the conceptual idea; conducted experiments; wrote and refined the manuscript.		
Overall percentage (%)	80		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	15 Nov 2022

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Wei Wang		
Contribution to the Paper	Conducted security evaluation experiments		
Signature		Date	09/11/2022

Name of Co-Author	Minhui Xue		
Contribution to the Paper	Built the conceptual idea; refined the manuscript; supervised the development of this work		
Signature		Date	9 Nov 2022

Name of Co-Author	Gareth Tyson		
Contribution to the Paper	Wrote and refined the manuscript		
Signature		Date	9 Nov 2022

Name of Co-Author	Seyit Camtepe		
Contribution to the Paper	Participated in discussion; contributed to the revision		
Signature		Date	9 Nov 2022

Name of Co-Author	Damith Ranashinghe		
Contribution to the Paper	Wrote and refined the manuscript; supervised the development of this work Built the conceptual idea.		
Signature		Date	15-11-2022

Please cut and paste additional co-author

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 chapter, 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.*

3.1 Introduction

In this chapter, I develop 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, I have conducted a comprehensive empirical security and privacy assessment of 40 contact tracing apps. The results have identified multiple security and privacy risks, as well as threats. Naturally, my analysis has confirmed that no apps can protect users' security and privacy against *all* potential threats.

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, lock-downs, 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 [1]. 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 [2, 3, 4]. 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 [5], the public COVID-19 website in South Korea [6], and the mobile contact tracing apps released in Australia [7], Germany [8], Israel [9], Singapore [10], and United Kingdom [11]. 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 [12] 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 anonymisation of individuals' information is a challenging problem [13]. To mitigate these concerns, we develop a methodology for assessing the security and privacy weaknesses of COVID-19 contact tracing apps.

Specifically, this chapter 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 utilising 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 Tables 3.1 and 3.2); (*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?

Table 3.1. Contact tracing apps considered in our study.

#	Applications	Country	Downloads
1	COVIDSafe	Australia	1M
2	Hamagen	Israel	1M
3	TraceTogether	Singapore	500K
4	StopCovid France	France	1M
5	Next Step (DP3T)	Switzerland	N/A
6	Corona Warn App	German	5M
7	NHS Test and Tracing App	UK	10K
8	TraceCORONA	Germany	N/A
9	Private Kit	USA	10K
10	MySejahtera	Malaysia	500K
11	Smittestop	Denmark	10K
12	Covid Alert	Canada	500K
13	SwissCovid	Switzerland	500K
14	Bluezone	Vietnam	100K
15	COCOA	Japan	5M
16	Immuni	Italy	1M
17	Stopp Corona	Austria	100K
18	Aarogya Setu	India	100M
19	EHTERAZ	Qatar	1M
20	Vietnam Health Declaration	Vietnam	100K

- **RQ4:** What are the user concerns and requirements of contact tracing apps?

The main contributions of our study are as follows.

- 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

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

Table 3.2. Contact tracing apps considered in our study (continue).

#	Applications	Country	Downloads
21	STOP COVID19 CAT	Spain	500K
22	CG Covid-19 ePass	India	500K
23	StopTheSpread COVID-19	UK	100K
24	Stop COVID-19 KG	Kyrgyzstan	10K
25	BeAware Bahrain	Bahrain	100K
26	Nepal COVID-19 Surveillance	Nepal	5K
27	Stop Covid	Georgia	100K
28	Contact Tracing	USA	10K
29	Contact Tracer	USA	10K
30	Coronavirus Algérie	Algeria	100K
31	CoronaReport	Austria	10K
32	Covid19!	Czech	10K
33	Coronavirus Bolivia	Bolivia	50K
34	Coronavirus - SUS	Brazil	1M
35	COVA Punjab	India	1M
36	SOS CORONA	Mali	10K
37	Hamro Swasthya	Nepal	50K
38	COVID Radar	Netherlands	50K
39	NICD COVID-19 Case Investigation	South Africa	10K
40	Coronavirus UY	Uruguay	100K

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 decentralised architectures are not necessarily more secure than those adopting centralised 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 3.6 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.

3.2 Background and Related Work

3.2.1 Taxonomy of Contact Tracing Apps

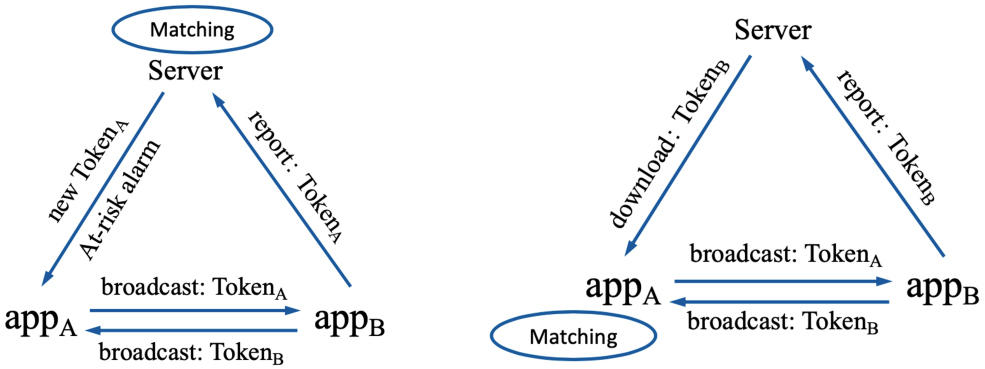
The country-centric contact tracing apps we study fall into two broad categories: (*i*) centralised and (*ii*) decentralised.

Centralised architectures. Many apps utilise a centralised 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, centralised 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 [14, 5].

For instance, *TraceTogether* [10] from Singapore and *COVIDSafe* [7] 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* [15] and the ROBERT system [16] implemented by *StopCovid France*.

Decentralised architectures. The second type of solution is decentralised, 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 decentralised apps operate in a range of ways. For example, *Hamagen* [9] relies on location information. Users download the location history of diagnosed



(a) Centralised contact tracing apps (b) Decentralised contact tracing apps

Figure 3.1. An illustration of centralised and decentralised contact tracing applications.

users from a back-end server to check if they been in contact with any infected people. Other decentralised solutions, *e.g.*, *DP3T* [17] and the Exposure Notification framework by Google and Apple (Gapple) [18], utilise 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* [11] and *Corona Warn App* [19] implement the Gapple framework. This design paradigm reduces the privacy exposure of users as only anonymous identifications are shared.

In a short word, as illustrated in Figure 3.1, the key differences between centralised and decentralised contact tracing applications lie in (i) centralised contact tracing applications collect the contact records from diagnosed users, while decentralised ones collect the token of diagnosed users; and (ii) centralised contact tracing applications evaluates health status by server, while decentralised ones evaluate health status by users.

3.2.2 Related Work

A number of prior works have utilised similar methodologies to our proposed 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 [20]. Several

studies [21, 22, 23, 24, 25, 26, 27] have used static analysis to analyse 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 [28] is a static code analysis tool designed to discover various mobile security-related vulnerabilities, either in source code or APKs. ANDROBUGS [29] is a framework that helps developers find potential mobile security vulnerabilities by pattern-matching. MOBSF [30] offers automated application penetration testing, malware analysis, and a security and privacy assessment framework. FLOWDROID [31] statically computes data flows in apps to understand which parts of the code that data may be exposed to. More details about static and dynamic analysis methodologies are discussed in Chapter 2.

Notably, these off-the-shelf tools only utilise 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, COVIDGUARDIAN 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 [32, 33] have focused on security and privacy analysis of the Bluetooth and cryptography specifications published by Apple and Google [34, 35], arguing that significant risks may be present. Other work [36, 37] has conducted a review of centralised and decentralised solutions, and proposed privacy-preserving contact tracing using a zero-knowledge protocol. Sun *et al.* [38] 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.* [39] inspected the broad implications of COVID-19 related apps, identifying the presence of malware. Wang *et al.* [40] also performed a statistical analysis of contact tracing app popularity, as well as user reviews.

Our work. In contrast to recent works on analysing contact tracing apps [41, 42, 32, 33, 43], 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 centralised and decentralised apps.

Table 3.3. 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

3.3 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 3.3). 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 3.2. 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.

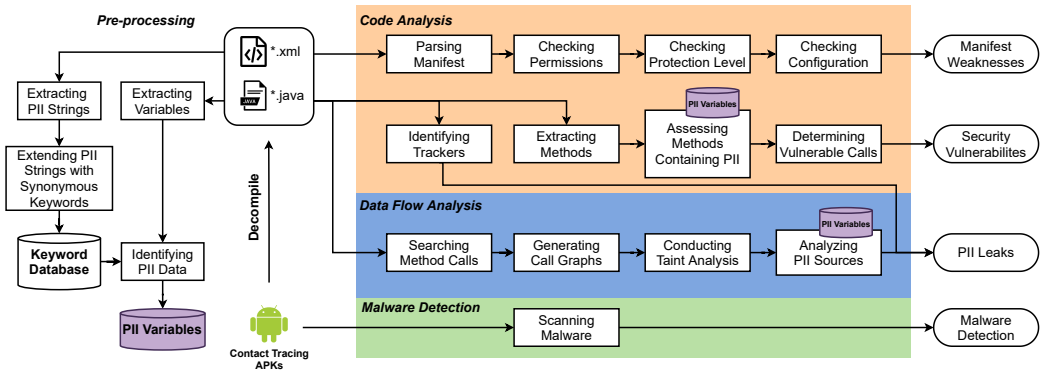


Figure 3.2. COVIDGUARDIAN: An overview of our security and privacy assessment methodology.

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 3.2 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 utilise WORD2VEC [44] to expand our keyword pool with synonymous words. To ensure the generalisation of our PII keyword database, we train the model with the text extracted from 54,371 general apps by the project, SUPOR [45]. 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

In layout file

```
<TextView android:text="Full name" />

<EditText
    android:id="@+id/personal_details_name"
    android:contentDescription="Enter full name" />
```

In Java code

```
String obj = (EditText) findViewById(R.id.personal_details_name)
    .getText().toString();
```

Figure 3.3. An example pattern of rendering a view widget.

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 3.3 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 3.2, 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 predefined 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 utilises 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 3.3.

The assessed vulnerabilities include SQL injection, IP address disclosure, hard-coded encryption keys, improper encryption, use of insufficiently random values (CWE 330) [46], 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 3.3.

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 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 unauthorised 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 unauthorised 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 unauthorised 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 artefacts in contact tracing apps. COVIDGuardian sends the APKs to VirusTotal [47], a free online service that integrates over 70 antivirus scanners. Note this has been widely adopted by the research community [25, 48, 49]. As shown in Table 3.3, 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 [50] 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 [51]. Finally, the taint analysis engine utilises the call graphs previously generated, with the list of sinks and sources, to locate private data leakage.

To perform the detection, we first summarise 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 analyse PII leakage through the AST. Finally, the static code analysis engine verifies whether the app logic invokes the data leakage functions or not.

3.4 Evaluation and Results

3.4.1 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 finalised the list of 40 contact tracing apps, as shown in Table 3.1. More detailed information, such as the versions of apps, is provided in our open source website.

3.4.2 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 analyse the security of Android apps. MobSF is recommended by OWASP [52]; 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 under-study. The detection precision results are listed in Table 3.4. Note that, as we focus on evaluate the false positives generated by tools, we only report the precision in this research. 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 3.4. Comparison of analysis results from different tools. The number of types is the sum of the number of risks an leaks identified by a tool; the precision rates are obtained through manual validation.

Tools	# Types	Precision
COVIDGuardian	315	96.19%
MobSF	213	47.41%
AndroBugs	76	80.26%
FlowDroid	201	40.32%
Qark	93	84.94%

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 analyse apps by keyword scanning. We manually analysed 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.

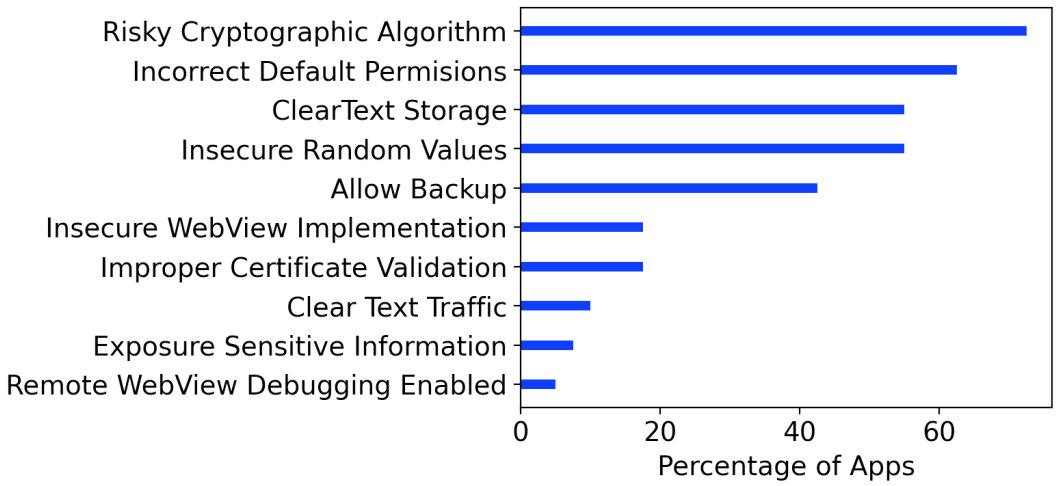


Figure 3.4. Percentages of apps that are subject to vulnerabilities based on code analysis.

3.4.3 Empirical Assessment Results

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

Code analysis results. Figure 3.4 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 "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) [53] attacks during network transmission.

Figure 3.4 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 [51] 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

Table 3.5. 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%

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 3.5, the most frequent tracker is `Google Firebase Analytics`, which is identified in more than 70% of the apps. Notably, a research study [54] argues that TraceTogether, by using Google's Firebase service to store user information, maybe leaking user information.

Data flow analysis results. Figure 3.5 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 (Israel) as an example, location information is detected and obtained by a source method called `initialise(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 [31].

According to the COVIDGuard results, we identified various PII leaks in contact tracing apps. We present the applications that have PII leaks identified according to 10 types of sinks in Figure 3.6. 60% of the 40 assessed apps have at least one PII leak found; 30% have more than 10 PII leaks detected.

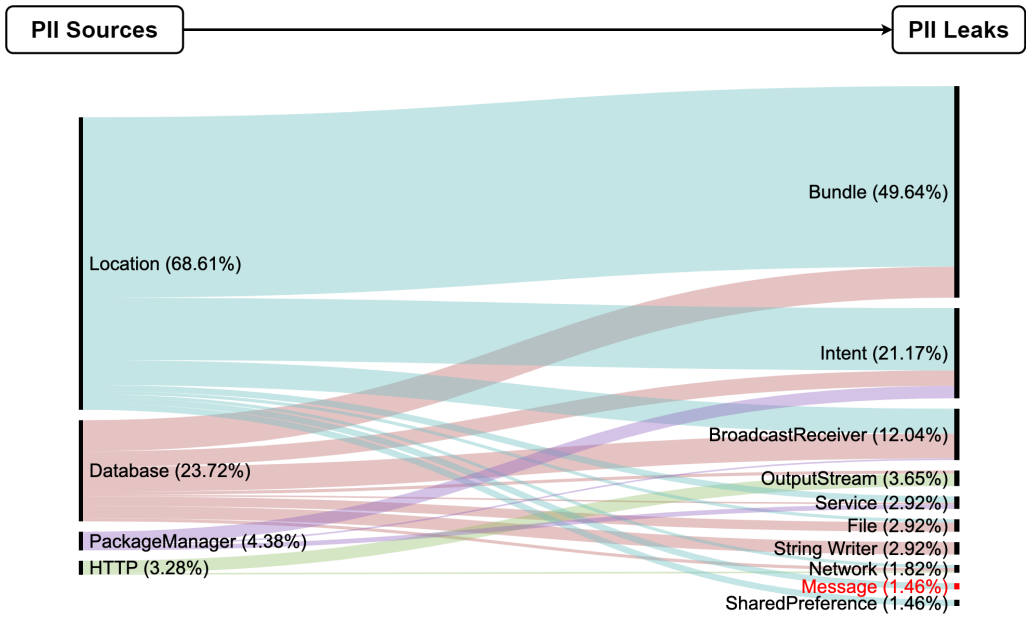


Figure 3.5. Privacy leaks detected between sources and sinks. Percentages indicate the fraction of flows originating at the sources (left) and terminating at the sinks (right).

The app that have the most PII leak paths identified is COVA Punjab (40 leaks identified). The PII information is leaked through Bundle, File, and Intent in this app. Notably, we also discover that some apps transmit location information through SMS messages. Considering Hamagen (Israel) as an example, location information is detected and obtained by a source method `initialize(Context,Location,e)` and 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 Android SMS service [31].

Malware detection results. We discover only one app with malware, Stop COVID-19 KG (Kyrgyzstan) [55]. 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 [56]. 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 [57] disclosed that new ransomware has targeted the contact tracing app in Canada even before its public release.

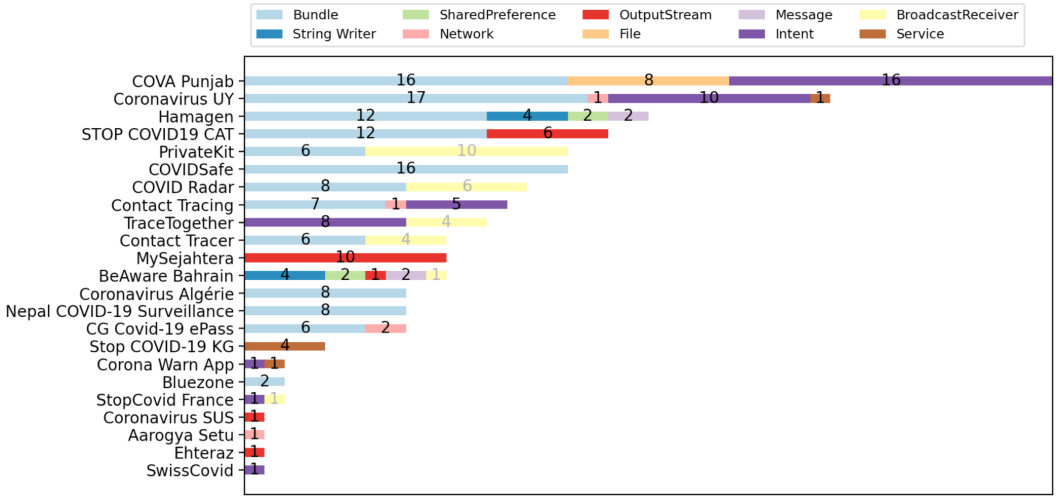


Figure 3.6. PII Leaks in contact tracing apps.

Feedback from developers. After alerting all developers of ours findings, we re-checked the apps by regression testing. Table 3.6 summarises 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 3.6). 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 [58] 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.

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.

Table 3.6. 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

3.4.4 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 [59] 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 [60]. When a user inputs confidential data, including passwords, in a debug-enabled WebView, attackers may be able to inspect all elements in the webpage [61]. 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 [2]), the long duration of their operation, and the fact that 7.6% of Android users already have rooted their devices [62]. 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:

- Implement database encryption [63]
- Enable root detection [64] and confidential data protection [65] 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 [66]. Therefore, if enough data in a region were collected, contact IDs and timestamps could be used to analyse movements [67].

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 [67]. 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 [68] framework, which enhances the SQLite database by making it more suitable for encrypted local data storage. It also introduces the conscrypt [69] 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.

3.5 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.

3.5.1 Threat Model

We consider four attackers in our threat model in addition to the user groups.

Application users. Those who install contact tracing applications on their mobile phones will receive information about COVID-19, *e.g.*, an at-risk alarm. A regular user may reveal their private information, *e.g.*, name, gender, phone number, national ID, home address, and location history, to the contact tracing

systems, as well as discover other users' private information from the system — pubic information or broadcasts from other users.

Health authorities. The actors are responsible for diagnosing infections and collecting health information from Application users. They may learn or deduce private information about at-risk users. Health authorities will also help the diagnosed users record or upload information to the contact tracing system.

Governments. These actors work with technology providers and are often responsible for operating the contact tracing system. They may access the data stored in a central server. In our threat model, we suppose the Government (and even the cloud operator) is "untrusted", that is, they may use the collected data for purposes beyond the pandemic.

Malicious adversaries. These adversaries have access to local app information. They follow the defined algorithms, but wish to learn more than the allowed information. They may have the capability to access the local log of contact tracing applications, but hacking the back-end server or another user's device is out of the scope of their capabilities. They may utilise some devices, such as a Bluetooth broadcaster or receiver, to attack the system or gain extra information. They may also modify the app and impersonate a legitimate user to access the system, which is difficult to prevent unless remote attestation is applied.

3.5.2 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 3.1. 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 summarised in Table 3.7 and Table 3.8, all 27 assessed apps have user privacy exposure to some extent. We determine user exposure level as IV in some centralised 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 decentralised Bluetooth-based apps are categorised as a lower-level exposure. This is because they utilise 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.

3.5.3 Security and Privacy Threats

As discussed previously, the privacy of users is hard to preserve in a contact tracing system. To introduce potential privacy threats, we will let Alice be an at-risk user, and let Bob be a diagnosed user who has been in contact with

Table 3.7. User privacy exposure and threats to apps.

#	Apps (27 of 40 apps assessed; 13 apps do not provide adequate information)	Privacy Exposure		
		Generic	At-risk	Diagnosed
1	CovidSafe	▪▪	▪▪▪▪	▪▪▪▪
2	HaMagen	▪	▪	▪▪▪▪▪
3	TraceTogether	▪▪	▪▪▪▪	▪▪▪▪
4	StopCovid France	▪▪	▪▪▪	▪▪▪
5	Next Step (DP3T)	▪▪	▪▪▪	▪▪▪
6	Corona Warn App	▪▪	▪▪	▪▪▪
7	NHS Test and Tracing App	▪▪	▪▪	▪▪▪
8	TraceCorona	▪▪	▪▪	▪▪▪
9	Private Kit	▪▪	▪▪	▪▪▪
10	MySejahtera	▪▪▪▪	▪▪▪▪	▪▪▪▪
11	Smittestop	▪▪	▪▪▪	▪▪▪
12	COVID Alert	▪▪	▪▪	▪▪▪
13	SwissCovid	▪▪	▪▪	▪▪▪
14	Bluezone	▪▪	▪▪	▪▪▪
15	COCOA	▪▪	▪▪	▪▪▪

▪: No data is shared with servers or users;

▪▪: Token shared with proximity users; ▪▪▪: Token shared with the server;

▪▪▪▪: PII shared with the server; ▪▪▪▪▪: PII is released to public.

Alice. *Mallory* will be a malicious attacker, and Grace will be the government server (or other authority). Here we discuss four potential privacy and security threats. According to our threat model in Section 3.5.1, if an attacker is not able to re-identify a user or inject fake reports to a contact tracing system through a specific privacy attack, we try to determine the system as well-protected against such a threat; otherwise, the system will be considered as at-risk. The assessment results are summarised in Table 3.9 and Table 3.10.

To further examine security and privacy threats against contact tracing apps, we systematically search and review the state-of-the-art [70, 42, 33, 32]. 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

Table 3.8. User privacy exposure and threats to apps (continue).

#	Apps (27 of 40 apps assessed; 13 apps do not provide adequate information)	Privacy Exposure		
		Generic	At-risk	Diagnosed
16	Immuni	▪▪	▪▪	▪▪▪
17	Stopp Corona	▪▪	▪▪	▪▪▪
18	Aarogya Setu	▪▪▪▪	▪▪▪▪	▪▪▪▪
19	EHTERAZ	▪▪▪▪	▪▪▪▪	▪▪▪▪
20	Vietnam Health Declaration	▪▪▪▪	▪▪▪▪	▪▪▪▪
21	STOP COVID19 CAT	▪	▪▪▪▪	▪▪▪▪
22	CG Covid-19 ePass	▪▪▪▪	▪▪▪▪	▪▪▪▪
23	StopTheSpread COVID-19	▪	▪▪▪▪	▪▪▪▪
24	Stop COVID-19 KG	▪▪▪▪	▪▪▪▪	▪▪▪▪
25	BeAware Bahrain	▪	▪▪▪▪	▪▪▪▪
26	Nepal COVID-19 Surveillance	▪	▪▪▪▪	▪▪▪▪
27	Stop Covid	▪▪	▪▪	▪▪▪▪

▪: No data is shared with servers or users;

▪▪: Token shared with proximity users; ▪▪▪: Token shared with the server;

▪▪▪▪: PII shared with the server; ▪▪▪▪▪: PII is released to public.

synthesise four dominant security and privacy threats: (*i*) server link attacks; (*ii*) user link attacks; (*iii*) false positive claims; and (*iv*) relay attacks. We summarise the nature of these threats in Figure 3.7, Figure 3.8, Figure 3.9, and Figure 3.10, and discuss them in turn below.

Linkage attacks by servers. In centralised systems, the major privacy concern is metadata leakage by the server. For example, in TraceTogether and COVIDSafe, a central server is used to collect PII information and to evaluate at-risk individuals. 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 deduce the social connections of Alice. Even for StopCovid France, a centralised Bluetooth system with solutions to avoid PII collection, the re-identifiable threats still exist. For example, from the server side, Grace is able to link ephemeral IDs to the corresponding permanent app identifier and thus trace Alice based on IDs observed in the past, as well as tracing future movements. Thus, no centralised solutions in Table 1 can prevent linkage attacks by the server.

Table 3.9. User privacy exposure and threats to apps.

#	Apps (27 of 40 apps assessed; 13 apps do not provide adequate information)	Threats				Architecture
		Linkage-Server	Linkage-User	False-Claim	Relay Attack	
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

○: the system is well protected ●: the system is at-risk C: centralised D: Decentralised

(-): Inadequate information to conduct an assessment.

In contrast, for decentralised Bluetooth solutions, Alice's privacy is protected as her PII will not be sent to a central server by a diagnosed user and her health status is evaluated on her own device. Thus, decentralised Bluetooth systems are able to protect users' privacy against linkage attacks by the server. However, in location-based decentralised systems (*e.g.*, Hamagen), the server also learns users' PII, which can compromise such such decentralised systems too (as marked in Table 3.9 and Table 3.10).

Privacy guideline 1: To protect users' privacy against linkage attacks by a server, a contact tracing app should:

- Avoid sharing PII with central points or
- Implement a decentralised design.

Table 3.10. User privacy exposure and threats to apps (continue).

#	Apps (27 of 40 apps assessed; 13 apps do not provide adequate information)	Threats: Linkage-Server	Threats: Linkage-User	Threats: False-Claim	Threats: Relay Attack	Architecture
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: centralised D: Decentralised

(-): Inadequate information to conduct an assessment.

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 3.9 and Table 3.10 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 [42]. Note that Mallory is able to extend such attacks to Sybil

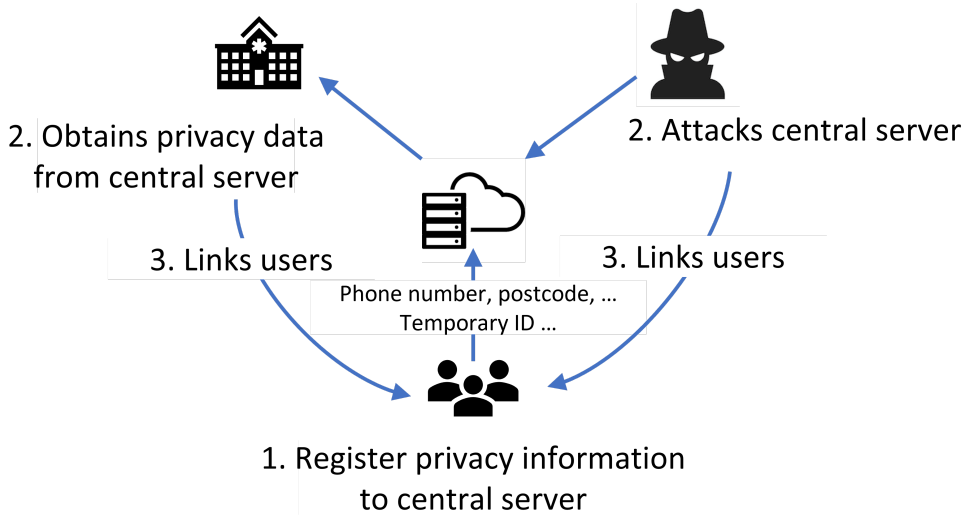


Figure 3.7. Linkage attacks by a server: in centralised systems, a key privacy concern is metadata leakage by the server.

attacks to enable the identification and the tracing back of multiple targets at the same time. Even worse, if Mallory distributes multiple broadcast receivers, which could be also considered as a Sybil attack, in a large area with some layout, *e.g.*, honeycomb, 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:

- Avoid data sharing between users or
- Ensure privacy protections exist for any published data.

False positive claims. In some systems, such as Coronavirus Australia, Bob can register as infected and upload data through the contact tracing app to the server, which enables Alice to receive an at-risk alarm. However, 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 authorisation process, *i.e.*, Bob is only permitted to upload data after receiving a one-time-use permission code generated by the server. Without the permission code, Mallory is not allowed to claim they are infected

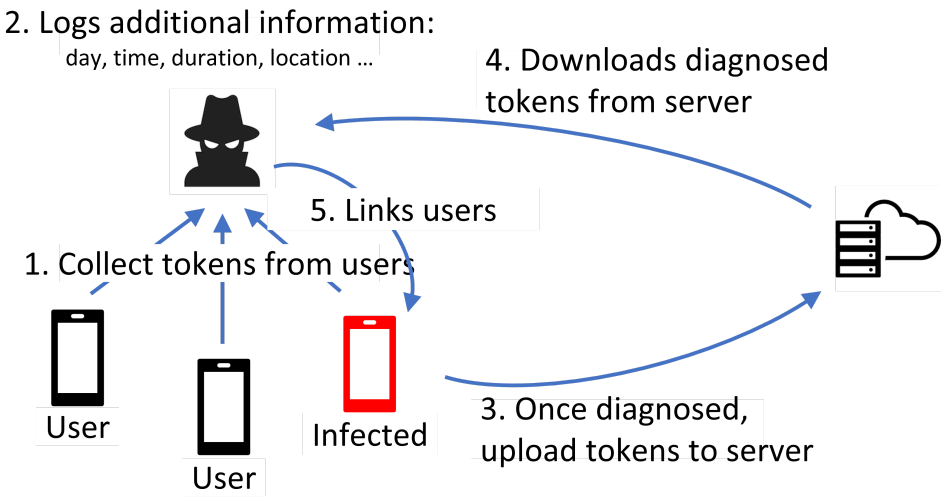


Figure 3.8. Linkage attacks by users: in systems based on information exchange between users, attackers could re-identify users using data published or received from users.

and Alice will always receive a true at-risk alarm. Only two solutions, *i.e.*, DP3T and PACT [70], have no authorisation process implemented.

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

Relay attacks. To apply such an attack, Mallory could collect existing broadcast messages exchanged between users, then replay it at another time or forward it through proxy devices to a remote location and replay the messages. Due to the lack of message validation in solutions that utilise information broadcasts, a user will not be able to determine whether a received broadcast is from a valid source or from a malicious device. Any received broadcast will be recorded as a contact event, even though no actual contact exists. A malicious attacker can potentially redirect all the traffic from one place to another, resulting in a targeted area being incorrectly locked-down by replaying fake information.

For example, suppose Mallory records the broadcasts from Bob and then replays it hours later, or transmits it to a remote location and replays the messages to Alice. Alice's device, although not actually being in contact with Bob, will receive and record the replayed broadcast in its local log. Once Bob is diagnosed, Alice will receive an at-risk alarm even though she has never been in contact with Bob. Such an attack will falsely enlarge the contact range of

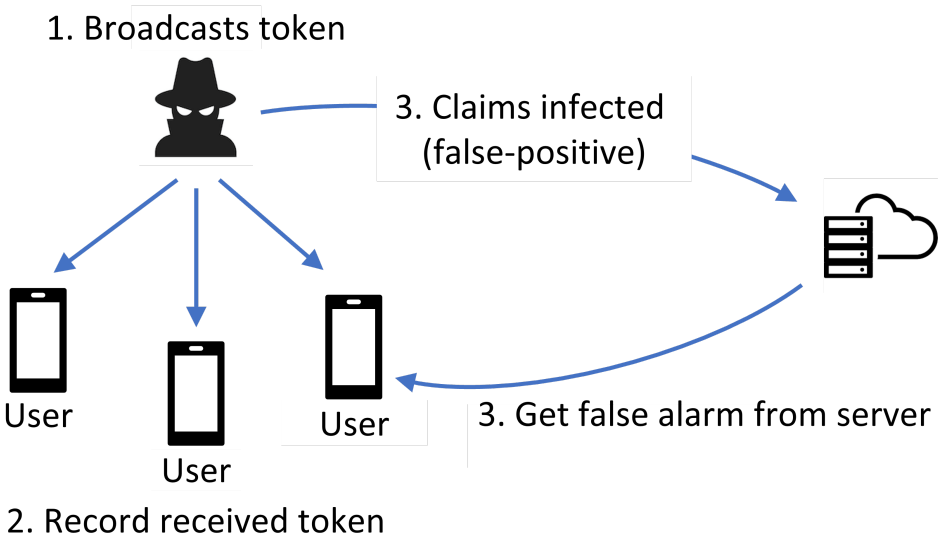


Figure 3.9. False positive claim: attackers can incorrectly register as infected, which will generate false at-risk alerts.

Bob and create a large amount of false-positive alarms, which may cause panic among citizens. Solutions that do not utilise information broadcasts, such as Coronavirus Disease-19 and Hamagen, can avoid relay attacks.

In DP3T, a solution is provided to limit the replay attack by including temporal information in the broadcast ephemeral identifiers. However, it cannot effectively prevent replay attacks occurring at the same moment. Another promising solution is to use an ambient physical sensing approach, *e.g.*, ambient audio. This has been shown to secure proximity detection by comparing the ambient information embedded in the broadcast messages with the local ambient. It allows a receiver to validate whether the source is nearby as the range of Bluetooth broadcast is generally within 50 m.

Privacy guideline 4: To protect a system against relay attacks, a solution should:

- Either avoid utilising information broadcast or
- Implement a validation approach.

Summary. The linkage attacks by a server are the overarching threats to centralised 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

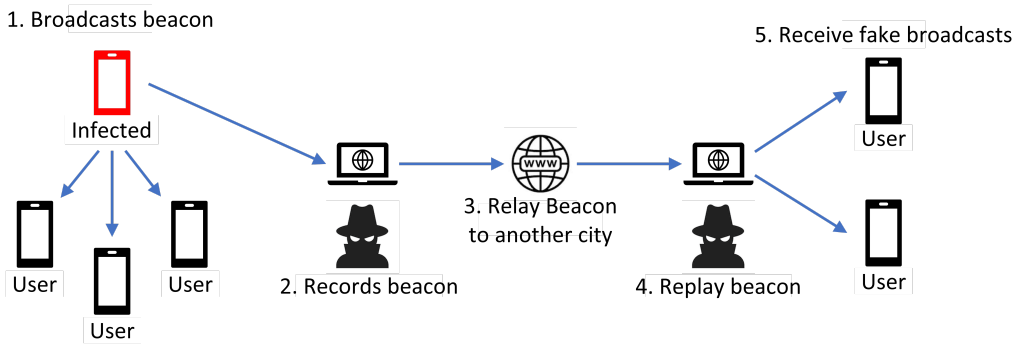


Figure 3.10. Relay attacks: in Bluetooth based apps, a malicious attacker can potentially redirect/replicate Bluetooth broadcasts from one place to another, thereby generating false at-risk alerts.

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 authorisation to allow only positive users to upload diagnosed data to the server, there are still some apps failing to implement essential authorisation. 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 categorise 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 decentralised architectures are necessarily more secure than those adopting centralised architectures, *e.g.*, Hamagen, a decentralised location-based system.

Table 3.11. Types of contact tracing apps investigated in the survey.

Type	Architecture	PII Collected?	Example App	Questions
A	Centralized	Yes	COVIDSafe	Q9, Q16
B	Centralized	No	StopCovid France	Q13, Q16
C	Decentralized	Yes	Hamagen	Q15
D	Decentralized	No	Corona Warn App	Q12, Q17

3.6 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*).

In our survey, 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. For example, COVIDSafe is covered by questions related to "phone number and postcode are shared with governments or health authorities" and "upload proximity data if being tested positive", and Corona Warn App or other apps that implement Google and Apple (Gapple) framework are covered by "anonymous identifiers are shared with other users" and "upload anonymous tokens if being tested positive". We present the detailed Type to questions relationship in Table 3.11. For detailed questionnaire, please refer to Supplementary Section 3.6.4.

3.6.1 Ethical Considerations

The Human Research Ethics Committee of lead author's affiliation determined that the study was exempt from further human subjects review, and we followed best practice for ethical human subjects survey research, *e.g.*, all questions were optional and we did not collect unnecessary personal information. Participant consented for their answers to be used for academic research. All participants are over 18 years old, regularly use a smartphone, and are able to complete the survey in English.

3.6.2 Participant Recruitment

The proportion of youth aged 15-24 years with COVID-19 has increased six-fold from 24 February through 12 July 2020 [71]. It was reported by Reuters [72] 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.

3.6.3 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 3.5.

All questions, excluding demographics, are five-point Likert scale [73] 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.* [74] and Kaptchuk *et al.* [75]. We provide the survey design and questionnaire in our open source website and summarise 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 3.2. 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 [76]. 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.

3.6.4 User Study Questionnaire

This is a survey on Coronavirus (COVID-19) and contact tracing applications by researchers at the University of Adelaide. We will only collect non-identifiable personal information, *e.g.*, gender, age range and nationality, and will not put you at any risk for harm.

You do not have to answer any question that makes you uncomfortable. In order to participate, you must be at least 18 years old, regularly use a smartphone, and are able to complete the survey in English. We expect this survey will take about 15-20 minutes to complete.

[Please check] I am over 18 years old. I acknowledge that my gender, age range, and nationality could be collected by this survey and my answers to this survey may be published in an academic research.

1. What is your age? (a. Under 20 years old; b. 20-24 years old; c. 25-29 years old; d. 30-34 years old; e. 35-39 years old; f. 40-44 years old; g.

45-49 years old; h. 50-54 years old; i. 55-59 years old; j. 60-64 years old; k. 65-69 years old; l. 70 years or older.)

2. What is the highest degree or level of school you have completed? If currently enrolled, the highest degree received. (a. Did not graduate high school; b. Graduated high school; c. Graduated high school.)
3. To which gender identify do you most identify? (a. Female; b. Male; c. Other; d. Prefer not to say)
4. What is your nationality?
5. How likely would you be to install and use an app that is able to accurately detect and notify you that you were exposed, but actually you were not? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)
6. How likely would you be to install and use an app that **may have a chance to incorrectly (false positively)** detect and notify you that you were exposed, but actually you were not? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)
7. How likely would you be to install and use an app that **may have a chance to fail (false negatively)** to detect that you were exposed and not notify you in time? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)
8. How likely would you be to install and use an app that shares your **location data with other users** for the purposes of studying or mitigating the spread of COVID-19? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)
9. How likely would you be to install and use an app that shares your **location data with your government or health authorities** for the purposes of studying or mitigating the spread of COVID-19? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)
10. How likely would you be to install and use an app that shares your **phone number and postcode with other users** for the purposes of studying

or mitigating the spread of COVID-19? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

11. How likely would you be to install and use an app that shares your **phone number and postcode with your government or health authorities** for the purposes of studying or mitigating the spread of COVID-19? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

12. How likely would you be to install and use an app that shares your **anonymous identification with other users** for the purposes of studying or mitigating the spread of COVID-19? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

13. How likely would you be to install and use an app that shares your **anonymous identification with your government or health authorities** for the purposes of studying or mitigating the spread of COVID-19? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

14. How likely would you be to install and use an app that shares **no information with other users or your government** for the purposes of studying or mitigating the spread of COVID-19? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

15. If you tested positive of COVID-19, how likely would you be to **upload your location data** for the past two weeks? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

16. If you tested positive of COVID-19, how likely would you be to **upload your proximity data** (who have been in the proximity of your phone) for the past two weeks? (1 - Extremely likely; 2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

17. If you tested positive of COVID-19, how likely would you be to **upload your anonymous tokens** for the past two weeks? (1 - Extremely likely;

2 - Somewhat likely; 3 - Neither likely nor unlikely; 4 - Somewhat unlikely; 5 - Extremely unlikely)

18. When you use a contact tracing application, how concerned are you about **battery drain** on your phone? (1 - Very unconcerned; 2 - Somewhat unconcerned; 3 - Neither unconcerned nor concerned; 4 - Somewhat concerned; 5 - Very concerned)

19. When you use a contact tracing application, how concerned are you about **privacy issues**? (1 - Very unconcerned; 2 - Somewhat unconcerned; 3 - Neither unconcerned nor concerned; 4 - Somewhat concerned; 5 - Very concerned)

20. When you use a contact tracing application, how concerned are you about whether it is **easy-to-use**? (1 - Very unconcerned; 2 - Somewhat unconcerned; 3 - Neither unconcerned nor concerned; 4 - Somewhat concerned; 5 - Very concerned)

21. When you use a contact tracing application, how concerned are you about its **accuracy**? (1 - Very unconcerned; 2 - Somewhat unconcerned; 3 - Neither unconcerned nor concerned; 4 - Somewhat concerned; 5 - Very concerned)

22. When you use a contact tracing application, how concerned are you about **storage drain** on your phone? (1 - Very unconcerned; 2 - Somewhat unconcerned; 3 - Neither unconcerned nor concerned; 4 - Somewhat concerned; 5 - Very concerned)

3.6.5 Data Analysis and Results

After gathering the answers from participants, we combine the answers that related to one type of apps together. We take the more negative answer to the likelihood questions, that is, keep the answer with larger value of Likert scale, as in our study design, 5 means "Extremely unlikely", and 1 means "Extremely likely". For example, if a user answered "1 - Extremely likely" to question 11, "How likely would you be to install and use an app that shares your phone number and postcode with your government or health authorities for the purposes of studying or mitigating the spread of COVID-19?", and selected "5 - Extremely unlikely" to question 16, "If you tested positive of COVID-19, how likely would you be to upload your proximity data (who have been in the proximity of your

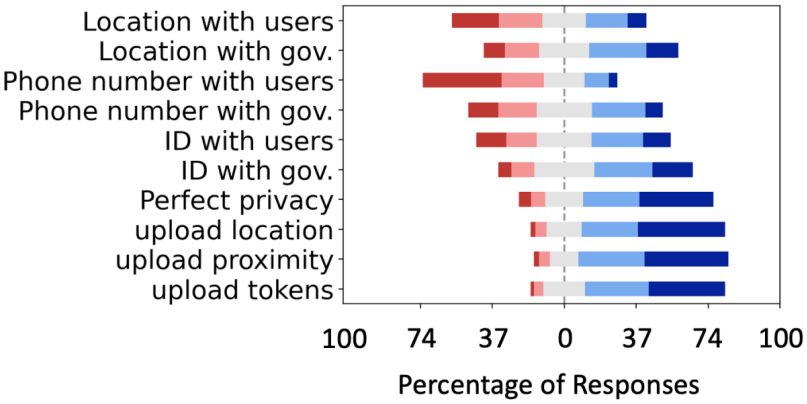


Figure 3.11. Raw data of answers from question 8 to 17.

phone) for the past two weeks?", we will consider that the user's likelihood of use this app is "5 - Extremely unlikely". After combining questions, we use a Python library, `SciPy.stats` to conduct Mann-Whitney U-tests. 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 tracking apps and consider the difference is significant. The raw data of answers from question 8 to 17 is provided in Figure 3.11.

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 [77] 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 3.5.3).

We calculate the U-value and p-value of each pair of contact tracing solutions among centralised solutions with PII collected (Type A), centralised solutions with non-PII collected (Type B), decentralised solutions with PII collected (Type C), and decentralised 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 decentralised system. As shown in Figure 3.12, if a contact tracing app can accurately detect proximity and

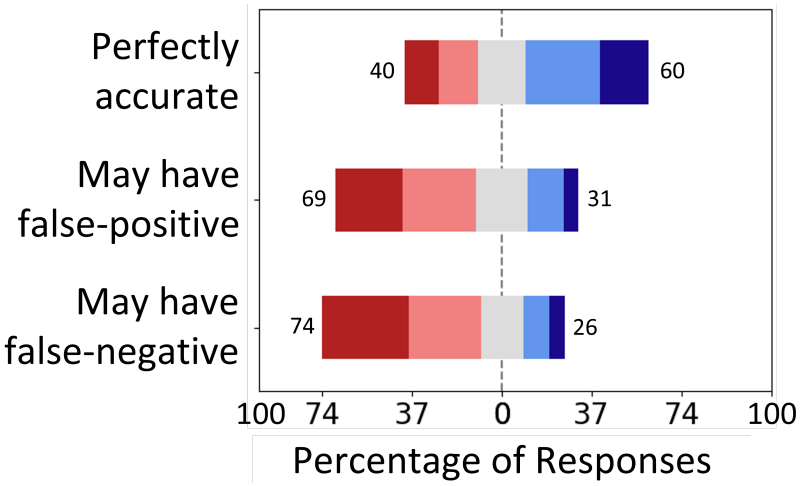


Figure 3.12. Participants' likelihood of using contact tracing apps *vs.* the accuracy of proximity detection.

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 well with a survey of COVID-19 app usage taken in the USA [75].

Figure 3.13, shows how likely users are to use a contact tracing app in different scenarios. Here, 50% of participants listed positive responses to decentralised apps with non-PII identification collected and shared (Type D). This resonates with the fact that Bluetooth-based decentralised systems preserve user privacy more than centralised systems. However, decentralised 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 3.12 presents the Mann-Whitney U-tests comparing pairs of contact tracing apps. Our results indicate that the differences between the likelihood of using decentralised apps with PII collected (Type C) and other apps is statistically significant. This is perhaps due to the privacy design in a decentralised 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

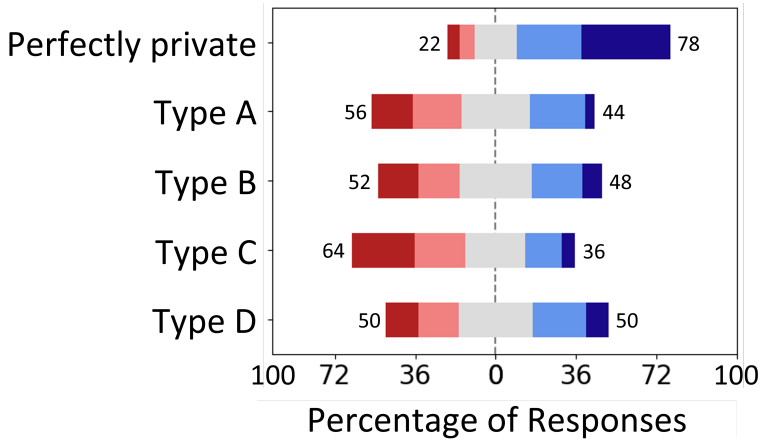


Figure 3.13. Participants' likelihood of using apps across different privacy-preserving and data sharing scenarios. ■ : 5 = extremely unlikely, ■ : 1 = extremely likely.

in the likelihood of users using these apps. However, from Figure 3.13, we can infer that it is more likely for users to accept and use the decentralised apps without PII collection (Type D).

Additionally, as evidenced in Figure 3.12 and Figure 3.13, 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 3.14, 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.

3.6.6 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 [78] (*i.e.*, participants and their responses may have different characteristics from the general population of

Table 3.12. 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: centralised system with PII collection
Type B: centralised system without PII collection
Type C: Decentralised system with PII collection
Type D: Decentralised system without PII collection

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 centralised solution in contrast to a decentralised solution that collects PII data.

3.7 Conclusion

This chapter has developed a security and privacy assessment tool, COVID-Guardian. This tool can evaluate the security weaknesses, vulnerabilities, potential privacy leaks, and malware in contact tracing apps. Using COVID-Guardian, 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

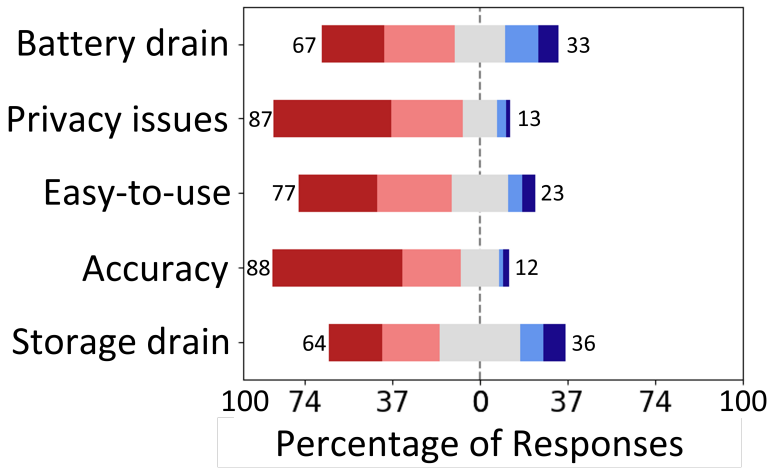


Figure 3.14. Participants' concerns about contact tracing apps. ■ : 5 = extremely concerned, ■ : 1 = extremely unconcerned.

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.

Bibliography

[1] World Health Organization. Operational considerations for case management of COVID-19 in health facility and community. https://apps.who.int/iris/bitstream/handle/10665/331492/WHO-2019-nCoV-HCF_operations-2020.1-eng.pdf, 2020.

[2] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science*, 2020.

[3] Bill Chappell. Coronavirus: sacramento county gives up on automatic 14-day quarantines. 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, 2020.

[4] Siddique Latif, Muhammad Usman, Sanaullah Manzoor, Waleed Iqbal, Junaid Qadir, Gareth Tyson, Ignacio Castro, Adeel Razi, Maged N. Kamel Boulos, Adrian Weller, and Jon Crowcroft. Leveraging Data Science To Combat COVID-19: A Comprehensive Review. *IEE Transactions on Artificial Intelligence*, 2020.

[5] Raymond Zhong Paul Mozur and Aaron Krolik. In Coronavirus fight, China gives citizens a color code, with red flags. https://www.nytimes.com/2020/03/01/business/china-coronavirus-surveillance.html, 2020, accessed: 2020-05-07, 2020.

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

[7] Australia Department of Health. COVIDSafe. https://www.health.gov.au/resources/apps-and-tools/covidsafe-app, 2020, accessed: 2020-04-23, 2020.

[8] Robert Koch-Institut. Corona Warn App. https://www.coronawarn.app/en/, 2020, accessed: 2020-08-17, 2020.

[9] Israel Ministry of Health. Hamagen. https://govextra.gov.il/ministry-of-health/hamagen-app/, 2020, accessed: 2020-04-23, 2020.

[10] Jason Bay, Joel Kek, Alvin Tan, Chai Sheng Hau, Lai Yongquan, Janice Tan, and Tang Anh Quy. Bluetrace: A privacy-preserving protocol for

community-driven contact tracing across borders. *Government Technology Agency-Singapore, Tech. Rep*, 2020.

[11] Department of Health and Social Care. Next phase of NHS coronavirus (COVID-19) app announced. https://www.gov.uk/government/news/next-phase-of-nhs-coronavirus-covid-19-app-announced, 2020, accessed: 2020-08-17, 2020.

[12] Kobi Leins, Chris Culnane, and Benjamin IP Rubinstein. Tracking, tracing, trust: Contemplating mitigating the impact of COVID-19 through technological interventions. *The Medical Journal of Australia*, page 1, 2020.

[13] Chris Culnane and Kobi Leins. Misconceptions in privacy protection and regulation. *Law in Context. A Socio-legal Journal*, 36(2):1–12, 2019.

[14] Min Joo Kim and Simon Denyer. A 'travel log' of the times in South Korea: Mapping the movements of coronavirus carriers. https://www.washingtonpost.com/world/asia_pacific/coronavirus-south-korea-tracking-apps/2020/03/13/2bed568e-5fac-11ea-ac50-18701e14e06d_story.html.

[15] The PEPP-PT team. PEPP-PT high level overview. https://github.com/pepp-pt/pepp-pt-documentation/blob/master/PEPP-PT-high-level-overview.pdf, 2020.

[16] PRIVATICS team and Fraunhofer AISEC. ROBERT: Robust and privacy-preserving proximity tracing. https://github.com/ROBERT-proximity-tracing/documents/blob/master/ROBERT-specification-EN-v1_0.pdf.

[17] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. Decentralized privacy-preserving proximity tracing. *arXiv preprint arXiv:2005.12273*, 2020.

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

[19] Jens Helge Reelfs, Oliver Hohlfeld, and Ingmar Poese. Corona-Warn-App: Tracing the start of the official COVID-19 Exposure Notification App for germany. *arXiv preprint arXiv:2008.07370*, 2020.

[20] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge 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)*, pages 603–620, 2019.

[21] Sen Chen, Lingling Fan, Guozhu Meng, Ting Su, Minhui Xue, Yinxing Xue, Yang Liu, and Lihua Xu. An empirical assessment of security risks of global Android banking apps. In *Proceedings of 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1310–1322, 2020.

[22] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. Iccta: Detecting inter-component privacy leaks in android apps. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 280–291. IEEE, 2015.

[23] Minhui Xue, Cameron Ballard, Kelvin Liu, Carson Nemelka, Yanqiu Wu, Keith Ross, and Haifeng Qian. You can yak but you can't hide: Localizing anonymous social network users. In *Proceedings of the 2016 Internet Measurement Conference*, pages 25–31, 2016.

[24] Sen Chen, Ting Su, Lingling Fan, Guozhu Meng, Minhui Xue, Yang Liu, and Lihua 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*, pages 797–802, 2018.

[25] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé Bissyandé, and Jacques Klein. MadDroid: Characterizing and detecting devious Ad contents for Android apps. In *Proceedings of The Web Conference 2020*, pages 1715–1726, 2020.

[26] Zhushou Tang, Ke Tang, Minhui Xue, Yuan Tian, Sen Chen, Muhammad Ikram, Tielei Wang, and Haojin Zhu. iOS, your OS, everybody's OS: Vetting and analyzing network services of iOS applications. In *29th USENIX Security Symposium*, pages 2415–2432, 2020.

[27] Wei Wang, Ruoxi Sun, Minhui Xue, and Damith C Ranasinghe. An automated assessment of Android clipboards. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020.

[28] Qark. https://github.com/linkedin/qark.

[29] AndroBugs. https://github.com/AndroBugs/AndroBugs_Framework.

[30] Mobile-security-framework-mobsf. https://github.com/MobSF/Mobile-Security-Framework-MobSF.

[31] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[32] Yaron Gvili. Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc. Technical report, Cryptology ePrint Archive, Report 2020/428, 2020.

[33] Lars Baumgärtner, Alexandra Dmitrienko, Bernd Freisleben, Alexander Gruler, Jonas Höchst, Joshua Kühlberg, Mira Mezini, Markus Miettinen, Anel Muhamedagic, Thien Duc Nguyen, et al. Mind the gap: Security & privacy risks of contact tracing apps. *arXiv preprint arXiv:2006.05914*, 2020.

[34] Apple Inc and Google Inc. Exposure notification-bluetooth specification. https://www.blog.google/documents/58/Contact_Tracing_-_Bluetooth_Specification_v1.1_RYGZbKW.pdf.

[35] Apple Inc and Google Inc. Exposure notification-cryptography specification. https://www.blog.google/documents/56/Contact_Tracing_-_Cryptography_Specification.pdf.

[36] Jinfeng Li and Xinyi Guo. COVID-19 contact-tracing apps: A survey on the global deployment and challenges. *arXiv preprint arXiv:2005.03599*, 2020.

[37] Joseph K Liu, Man Ho Au, Tsz Hon Yuen, Cong Zuo, Jiawei Wang, Amin Sakzad, Xiapu Luo, and Li Li. Privacy-preserving COVID-19 contact tracing app: a zero-knowledge proof approach.

[38] Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, and Damith C Ranasinghe. VenueTrace: a privacy-by-design COVID-19 digital contact tracing solution. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 790–791, 2020.

[39] Ren He, Haoyu Wang, Pengcheng Xia, Liu Wang, Yuanchun Li, Lei Wu, Yajin Zhou, Xiapu Luo, Yao Guo, and Guoai Xu. Beyond the virus: A first look at Coronavirus-themed mobile malware. *arXiv preprint arXiv:2005.14619*, 2020.

[40] Huiyi Wang, Liu Wang, and Haoyu Wang. Market-level analysis of government-backed covid-19 contact tracing apps. *arXiv preprint arXiv:2012.10866*, 2020.

[41] Haohuang Wen, Qingchuan Zhao, Zhiqiang Lin, Dong Xuan, and Ness Shroff. A study of the privacy of COVID-19 contact tracing apps. *International Conference on Security and Privacy for Communication Networks*, 2020.

[42] Serge Vaudenay. Analysis of DP3T between scylla and charybdis. 2020.

[43] Hyunghoon Cho, Daphne Ippolito, and Yun William Yu. Contact tracing mobile apps for COVID-19: Privacy considerations and related trade-offs. *arXiv preprint arXiv:2003.11511*, 2020.

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[45] Jianjun Huang, Zhichun Li, Xusheng Xiao, Zhenyu Wu, Kangjie Lu, Xiangyu Zhang, and Guofei Jiang. {SUPOR}: Precise and scalable sensitive user input detection for android apps. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 977–992, 2015.

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

[47] Virustotal. https://www.virustotal.com/.

[48] Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, Noha Loizon, and Roya Ensafi. The chain of implicit trust: An analysis of the web third-party resources loading. In *The World Wide Web Conference*, pages 2851–2857, 2019.

[49] Yangyu Hu, Haoyu Wang, Li Li, Yao Guo, Guoai Xu, and Ren 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)*, pages 332–343. IEEE, 2019.

[50] Jadx. https://github.com/skylot/jadx.

[51] OWASP. https://owasp.org/www-project-mobile-top-10/.

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

[53] Man-in-the-middle attack. https://en.wikipedia.org/wiki/Man-in-the-middle_attack.

[54] Douglas J Leith and Stephen Farrell. Coronavirus contact tracing app privacy: What data is shared by the Singapore OpenTrace app? 2020.

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

[56] Tara Gould, Gage Mele, Parthiban Rajendran, and Rory Gould. Anomali threat research identifies fake COVID-19 contact tracing apps used to download malware that monitors devices, steals personal data. https://www.anomali.com/blog/anomali-threat-research-identifies-fake-covid-19-contact-tracing-apps-used-to-monitor-devices-steal-personal-data.

[57] Charlie Osborne. New ransomware masquerades as COVID-19 contact-tracing app on your android device. https://www.zdnet.com/article/new-crycryptor-ransomware-masquerades-as-covid-19-contact-tracing-app-on-your-device/.

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

[59] Zhiyun Qian Hang Zhang, Dongdong She. Android root and its providers: A double-edged sword. 2015.

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

[61] Meggin Kearney. Remote debugging webviews. https://developers.google.com/web/tools/chrome-devtools/remote-debugging/webviews.

[62] Roman Unuchek. Rooting your Android: Advantages, disadvantages, and snags. https://www.kaspersky.com/blog/android-root-faq/17135/, 2017.

[63] Using the SQLite encryption extension. https://sqlite.org/android/doc/trunk/www/see.wiki.

[64] Android root detection techniques. https://blog.netspi.com/android-root-detection-techniques/.

[65] Support direct boot mode. https://developer.android.com/training/articles/direct-boot.

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

[67] Yilin Shen, Fengjiao Wang, and Hongxia Jin. Defending against user identity linkage attack across multiple online social networks. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 375–376, 2014.

[68] Sqlcipher. https://www.zetetic.net/sqlcipher/.

[69] conscrypt. https://github.com/google/conscrypt.

[70] Justin Chan, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob Sunshine, et al. PACT: privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv preprint arXiv:2004.03544*, 2020.

[71] WHO. Coronavirus disease situation report - 198. https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200805-covid-19-sitrep-198.pdf?sfvrsn=f99d1754_2, 2020, accessed: 2020-08-20, 2020.

[72] Ankur Banerjee and Stephanie Nebehay. Proportion of youth with COVID-19 triples in five months: WHO. 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, 2020.

[73] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[74] Lucy Simko, Ryan Calo, Franziska Roesner, and Tadayoshi Kohno. Covid-19 contact tracing and privacy: Studying opinion and preferences. *arXiv preprint arXiv:2005.06056*, 2020.

[75] Gabriel Kaptchuk, Daniel G Goldstein, Eszter Hargittai, J Hofman, and Elissa M Redmiles. How good is good enough for covid19 apps. *The influence of benefits, accuracy, and privacy on willingness to adopt.*, 2020.

[76] SoGoSurvey. https://www.sogosurvey.com.

[77] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[78] J Brassey, KR Mahtani, EA Spencer, and C Heneghan. Volunteer bias. *Catalogue Of Bias*, 2017.

Chapter 4

Measuring Privacy Practices and Application Behaviours

The work contained in this chapter has been published in
The Web Conference (WWW 2023).

The final publication is available at:
https://dl.acm.org/doi/pdf/10.1145/3543507.3583327

Ruoxi Sun, Minhui Xue, Gareth Tyson, Shuo Wang, Seyit Camtepe, and Surya Nepal. Not seen, not heard in the digital world! Measuring privacy practices in children's apps. In *The Web Conference (WWW, CORE A*)*, 2023.

Statement of Authorship

Title of Paper	Not Seen, Not Heard in the Digital World! Measuring Privacy Practices in Children's Apps
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Ruoxi Sun, Minhui Xue, Gareth Tyson, Shuo Wang, Seyit Camtepe, and Surya Nepal. Not seen, not heard in the digital world! Measuring privacy practices in children's apps. *The Web Conference 2023 (WWW, CORE A*)*, 2023

Principal Author

Name of Principal Author (Candidate)	Ruoxi Sun		
Contribution to the Paper	Built the conceptual idea; conducted experiments; wrote and refined the manuscript.		
Overall percentage (%)	90		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	30 Oct 2022

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Minhui Xue		
Contribution to the Paper	Built the conceptual idea; refined the manuscript; supervised the development of this work		
Signature		Date	9 Nov 2022

Name of Co-Author	Gareth Tyson		
Contribution to the Paper	Wrote and refined the manuscript, participated in discussion		
Signature		Date	9 Nov 2022

Name of Co-Author	Shuo Wang		
Contribution to the Paper	Refined the manuscript, participated in discussion		
Signature		Date	9 Nov 2022

Name of Co-Author	Seyit Camtepe		
Contribution to the Paper	Refined the manuscript, participated in discussion		
Signature		Date	9 Nov 2022

Name of Co-Author	Surya Nepal		
Contribution to the Paper	Refined the manuscript, participated in discussion		
Signature		Date	10/11/2022

Please cut and paste additional co-author panels here as required.

The digital age has brought a world of opportunity to children. Connectivity can be a game-changer for some of the world's most marginalised children. However, while legislatures around the world have enacted regulations and laws to protect children's online privacy, and app stores have instituted various restrictive protections and requirements, privacy in mobile apps remains a growing concern for parents, as well as the wider society. In this research, we aim to explore potential privacy issues and threats that exist in these apps. With this in mind, we investigate 20,195 *mobile apps from the Google Play store that are designed particularly for children (Family apps) or include children in their target user groups (Normal apps). Using both static and dynamic analysis, we find that 3.46% Family apps request location permissions, even though collecting location information from children is forbidden by the Play store; and 13.08% of Family apps use trackers (which are not allowed in children apps). Even big players with 40+ kids apps on the Play store use ad trackers. Furthermore, we find that most permission request notifications are not well designed for children. 21.69% apps have inconsistent ratings across different rating authorities. Our findings suggest that, despite significant attention to children's privacy, a large gap between regulatory provisions, app store policies, and actual development practices exists. Our research sheds light for government policymakers, app stores, developers. The dataset and source code are publicly available at* https://github.com/children-privacy.

4.1 Introduction

In this chapter, I focus on the privacy practices of apps that are designed for children or have target users that include children. I have measured the use of permissions and trackers, investigated inconsistency in content ratings, and analysed user comment feedback. The measurement results illustrate that, despite many privacy protection regulations and the strict requirements imposed by the app store, children still experience privacy threats.

The last decades has seen a dramatic increase in society's reliance on mobile services for everyday activities. At the start of the 2000s, there were 740 million cell phone subscriptions worldwide. Two decades later, that number has surpassed 8 billion, meaning there are now more cellphones in the world than people [1]. Particularly due to COVID-19, more and more young people are spending time at home using mobile applications for entertainment, remote

work, and day-to-day tasks [2]. Unfortunately, the Internet is home to a vast amount of content, potentially harmful to children, such as uncensored sexual imagery, violent content, and strong language [3, 4]. Though many parents find YouTube videos to be entertaining and educational, more than four-in-ten parents (46%) say their child aged 11 or younger who uses this platform has encountered videos on YouTube that were inappropriate for their age [5]. Furthermore, children may expose sensitive data online that could eventually fall in the hands of predators, or that could trigger conflicts with their peers. Of the 60 countries covered by The Economist Intelligence Unit's 'Out of the Shadows' index, only 9 have established legislation for mandatory reporting, content blocking, deleting and record-keeping of child sexual abuse material [6]. More than a third of young people in 30 countries report being a victim of online bullying [7].

Due to these concerns, many international jurisdictions have enacted privacy laws and regulations to promote and protect the privacy of individuals and to regulate how organisations handle personal information. These include the Children's Online Privacy Protection Act (COPPA) and its implementation, Children's Online Privacy Protection Rule (COPPR) which "imposes certain requirements on operators of websites or online services directed to children under 13 years of age" [8, 9], the European General Data Protection Regulation (GDPR) [10], and the Privacy Act 1988 [11] in Australia.

These regulations restrict the behaviour of apps and define the obligations of app developers. For example, GDPR Art. 8, COPPA, and COPPR §312 require that the applications should provide notice and obtain verifiable parental consent prior to collecting personal information from children. Apps must make reasonable efforts to ensure that the notification is received by a parent of a child. Although various privacy assertions are required in app stores (such as the permission list and the privacy policies), it is usually difficult for regular users to understand the potential threats an app may pose, let alone identify undesired or malicious application behaviours. Notably, according to a survey by Pew Research Center [12], the majority of Americans report being concerned about the way their data is being used by companies (79%); roughly three-in-ten Americans (28%) say they have suffered at least one major identity theft problem in the past 12 months.

In order to help parents determine age-appropriate mobile apps for their children, app stores have released strict developer policies, along with inspection

and vetting processes before app publishing, seeking to nip the aforementioned threats in the bud and improve app quality in the markets. Critical app information (such as the number of installs, requested permissions, a rating score, the name of developer, and the comments by other users) is provided to help users know the app before using it. Furthermore, every app that is sold through the Google Play store is rated for age-appropriateness. These rating systems recognise that mobile apps now run the entire gamut from interactive picture books for toddlers, through to graphic adult content. Hence, parents can use ratings to help with making app purchasing decisions. In 2015, Google Play launched the "Designed for Families" program [13], which allows app publishers to opt into an additional review in order have their apps labelled as being family-friendly, aiming at highlighting pre-approved, child-safe apps. Further, in 2020, Google Play added a "Teacher Approved" section in its app store, in which the Play store consults with teachers and specialists rating Designed for Families apps based on design, appeal, age appropriateness and the appropriateness of ads [14]. However, considering that most information of apps is provided by app developers' self-report (*e.g.*, by filling a form or answering questionnaires), a centralised rating system is still missing.

We argue that the complexity of these different policies and systems challenge the ability of many parents (and children) to make informed decisions. To gain an understanding of this complexity, we provide the first comprehensive measurement study of privacy practices in children's applications. We inspect apps from both a technical and user-available information perspective, with the aim to expose improper children app development practices and privacy threats. We particularly focus on apps that are (*i*) designed primarily for children under 13 and listed on the Children tab (Family apps); and (*ii*) designed for everyone, including children (Normal apps). From the technical side, we conduct static and dynamic analysis of 20,195 apps, including 3,627 Family apps and 16,568 Normal apps. From the users' perspective, we collect and analyse content ratings from 5 different ratings authorities and 13,132,577 comments from 11,831 apps. **The main findings of our measurement study are as follows.**

- Through static and dynamic analysis, we find that 3.46% Family apps request location permissions (which is forbidden by the Play store); the proportion of Normal apps requesting dangerous per-missions is 6.07% to 32.16% higher than that of Family apps. However, no apps except the YoutubeKids have permission notification specifically request consent

from parents (manually inspection on 500 apps); 13.08% Family apps use trackers that are not allowed to be used in children apps. Worryingly, even major players (having more than 40 kids apps published in Play store) do not follow the Play store policies.

- We compare the content ratings given by various agencies (these tag the suitability of apps for different age groups). We identify significant inconsistency among these content ratings. 19.25% apps have such issues, with 13.25% Family apps and 8.99% Normal apps have severe inconsistencies. We conclude that greater transparency should be introduced to help inform parents.
- From users' comments, we find that Family app users complain more about app content, but less about privacy and security. Highly-rated apps have a larger number of complaints at the same time. We conclude that a more efficient mechanism to report and check inappropriate app content should be established.

Our findings suggest that despite significant attention paid to children's privacy by legislatures, app stores, and society, a large gap between regulatory provisions, app store policies, and actual development practices still exists. We argue that app stores should establish more effective supervision mechanisms, provide more detailed information, reduce their reliance on developers' self-certified information, and respond more actively to user feedback. We believe our study can provide useful insights for government policy makers, app stores, developers, and researchers to build privacy-preserving apps for children, in the present and future.

Ethical considerations. All the apps and user comments in this research are collected from public available resources. All the app and developer names mentioned in this chapter are anonymised with *s. All user personal information, (usernames, timestamps) is removed from the database. We have disclosed all the findings to Play store and related entities.

4.2 Legal Background and Play Store Policies

This section describes the legal background for online children's apps that are subject to regulations or policies, such as COPPA, GDPR, and the Google

Play Store policies. We further briefly outline our legal analysis of potential violations in Android children apps.

4.2.1 The Scope of Children Apps

The definition of the term of "child" is varied across regulations issued in different territories and jurisdictions. According to COPPA and its implementation, Children's Online Privacy Protection Rule [9], the term "child" means an individual under the age of 13. In EU and EEA, the GDPR Art. 8 requires that "the processing of the personal data of a child shall be lawful where the child is at least 16 years old", while in UK GDPR [15], a child means anyone under the age of 18. In addition, as claimed by Play store, "all apps and games on the Children tab are expert-approved" which are further categorised into four age groups, *i.e.*, All ages up to 12, Ages up to 5, Ages 6-8, and Ages 9-12. Note that, based on our observation, apps without expert-approved badges can also be found on Children tab. Therefore, in our research, we define the apps that are categorised as FAMILY in Play store (*i.e.*, can be found at `https://play.google.com/store/apps/category/FAMILY`) as "Family apps", while the "Normal apps" in our research refer to apps that include children under 13-years old in their target user groups.

4.2.2 Families Policy Requirements

To better serve users, the Play store requires developers to provide accurate information about their apps. In addition to filling out the content rating questionnaire, developers also must provide details about their app's target audience and content. Depending on the target audience selections the developers make, the app will be subject to additional Google Play policies, to ensure that apps for children have appropriate content, show suitable ads, and handle personal and sensitive information correctly.

For apps that are designed primarily for children under 13, they must participate in the Designed for Families [13] and comply with Google Play's Families Policy Requirements [16]. For any apps that have at least one target audience age group that includes children, developers must comply with Google Play's Families Policy Requirements. In short, for any apps that have target users that include children, Google Play's Families Policy Requirements are compulsory. Developers are responsible for ensuring their apps are appropriate for children

and compliant with all relevant laws. Failure to satisfy the requirements may result in an app's removal or suspension.

4.2.3 Consent from Parents

Children merit specific protection with regard to their personal data, as they may be less aware of the risks, consequences and safeguards concerned and their rights in relation to the processing of personal data [17]. According to COPPR §312.4 [9], it shall be the obligation of the operator to provide notice and obtain verifiable parental consent prior to collecting, using, or disclosing personal information from children. Apps with target users under 13 must make reasonable efforts, taking into account available technology, to ensure that a parent of a child receives *direct notice* of the apps' practices with regards to the collection, use, or disclosure of personal information from children. As regulated in GDPR Art. 8, the processing of personal information is lawful only if the consent is given by the holder of parental responsibility over the child, meaning that users who are 15 years or younger need parental consent where applicable (member states can choose a younger age down to 13). Developers will need to prove that consent is valid, that it is informed and granular, and that they have methods in place to allow parents to exercise their rights in relation to children (also required by COPPR §312.4(b)). Although the Privacy Act 1988 in Australia does not specify an age after which an individual can make their own privacy decision, it requires that, for their consent to be valid, an individual must have capacity to consent. If they lack maturity it may be appropriate for a parent or guardian to consent on their behalf.

Therefore, companies must obtain verifiable parental consent before gathering data from children below the age limit (13 years of age for COPPA, 16 for GDPR). This legal requirement implies that informing parents or legal tutors about data collection practices via the privacy policy is not enough, especially if the app disseminates sensitive data to third-party services. With this in mind, we would like to determine whether apps collect private data without a user consent. As a result, any sensitive or personal data, particularly unique identifiers or geolocations, uploaded by the app to third parties without a user consent may be a potential violation of COPPA and GDPR. Note that, in Play store policy, apps that solely target children are not allowed to access location permissions.

4.2.4 Ads SDKs Allowed for Children

According to the Designing Apps for Children and Families policy by Play Store [13], apps designed specifically for children must participate in the Designed for Families program. This requires that apps can only use Google Play certified ad SDKs listed in Table 4.1. In order for an ad SDK to be included on the list, the ad SDK must self-certify that they are compliant with Play's Families Ad Program policies and all applicable local laws and regulations. Apps that include children and adults in their target audience, but are not in the Designed for Families program, are allowed to use non-certified ad SDKs for serving ads only to users above the age of 13. Therefore, any non-certified ad SDK used in Family apps violates the Play Store policy. Normal apps that have non-certified ad SDKs but do not implement a neutral age screen may also violate the policy.

4.2.5 Content Ratings

A content rating (also known as maturity rating) rates the suitability of TV broadcasts, movies, comic books, or video games for its audience [18, 19]. This usually places a media source into one of a number of different categories, to show which age group is suitable to view the media. In the Google Play store, the app developers are responsible for completing a rating questionnaire about the nature of the apps' content. The ratings assigned to the app, displayed on Google Play, are determined by the questionnaire responses. Misrepresentation of an app's content may result in removal or suspension [20]

The ratings are intended to help consumers (especially parents) identify potentially objectionable content that exists within an app. Considering that, in different territories, rating standards can have differences and each rating authority uses its own methodology, an app can earn different ratings. In Figure 4.1, we list a few rating authorities, including the Entertainment Software Rating Board (ESRB) [18] in Americas, the Pan European Game Information (PEGI) [19] in Europe and the Middle East, Unterhaltungssoftware Selbstkontrolle (USK) [21] in Germany, the Australian Classification Board (ACB) [22] in Australia, and the International Age Rating Coalition (IARC) [23]. However, for some apps, their content ratings across different territories are inconsistent and confusing, even if they contain the same content. For example, the app, `sg***ve`, earns a content rating of "PEGI 12" in Australia, while in Germany,

Table 4.1. Ad SDKs that participate in Play's Families Ads Program.

Ad SDKs	Code Signature	Network Signature
AdColony	com/adcolony/, com/jirbo/adcolony/	adcolony.com
AppLovin	com/applovin	applovin.com, applvn.com
Chartboost	com/chartboost/sdk/	chartboost.com
Google AdMob	com/google/ads/, com/google/android/gms/ads/, com/google/android/ads/, com/google/unity/ads/, com/google/android/gms/admob	2mdn.net, google.com, dmtry.com, doubleclick.com, doubleclick.net, mng-ads.com
InMobi	com/inmobi, in/inmobi/	inmobi.com, inmobicdn.net, inmobi.cn
ironSource	com/ironsource/	ironsrc.co
Kidoz	com/kidoz/sdk	kidoz.net/kidoz-sdk
SuperAwesome	tv/superawesome/sdk, tv/superawesome/lib/	superawesome.com
Unity Ads	com/unity3d/services, com/unity3d/ads	unity3d.com
Vungle	com/vungle/publisher/, com/vungle/warren/	vungle.com

it is rated as "USK 16+". Note that, in the requirements of "PEGI 12", an app could contain "slight violence towards fantasy characters", "non-graphic violence towards human-looking characters", and "mild bad language and no sexual expletives". However, an "USK 16+" app is allowed to contain "realistic violence", "shock and horror elements", "consistently explicit language", and "erotic or sexual focus", which obviously cannot fall into the category "PEGI 12", even the gap between suitable age ranges of the two ratings is only 1 year ("PEGI 12" is suitable for age group 12 to 15 and "USK 16+" is suitable for age group 16 to 17).

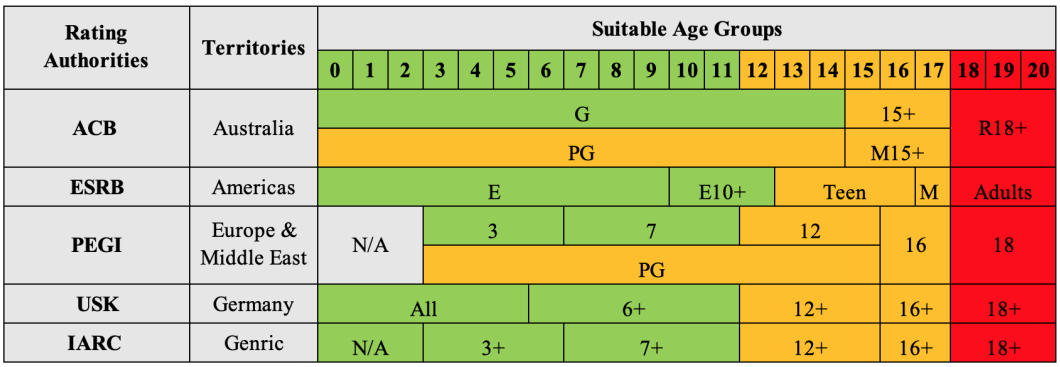


Figure 4.1. Age groups of content ratings from different rating authorities.

4.3 Related Work

4.3.1 Evaluation of Children Apps

Previous efforts have studied COPPA and children's apps from various perspectives. Previous work examined the risks posed by third-party components bundled in children's apps, with a focus on targeted advertisements [24, 25]. Massey *et al.* examined privacy policies [26] according to the requirements of COPPA, as well as Reyes *et al.* [27, 28] which analysed mobile apps' compliance with COPPA. Other research has focused on methods aiding developers to make their apps more child-friendly in terms of content and privacy [29, 30]. Several works focus on the evaluation of app content rating. For example, Chen *et al.* [31] conducted a study on the unreliable maturity content ratings of mobile apps, which may result in inappropriate risk exposure for the children and adolescents. However, this research focuses more on the comparison of content ratings between iOS app store and Google Play store, rather than exposing undesired application behaviours.

4.3.2 Static and Dynamic Analysis

Static code analysis techniques are widely used in the assessment of mobile apps [32, 33, 34, 35, 36, 37, 38, 39]. For example, MobSF [40] offers automated application penetration testing, malware analysis, and a security and privacy assessment framework. FlowDroid [41] 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 utilise syntax-based scanning and data-flows, which

leads to false positives that are not relevant to personal identifiable information. In contrast, dynamic analysis executes the code. Whereas static analysis often suffers from false positives, dynamic analysis is limited by the execution coverage [42]. Most work in this category focuses on analysing apps' network traffic [43, 44, 45, 46, 47]. In our work, we utilise static analysis for identifying permission requests and third-party trackers, while we trace privacy leakage using network traffic through a dynamic network monitoring tool [45]

4.3.3 User Comments analysis

Mobile app review comments have been extensively studied from other perspectives, including mining user opinions [48, 49, 50, 51], app comment filtering [52, 53], and exploring other concerns [54, 55]. Chen *et al.* [53] pioneered the prioritisation of user comments with AR-Miner. Chen *et al.* [55] conducted a study on the fraudulent campaigns to boost apps' rankings, which will result in inappropriate risk exposure for the children and adolescents. Besides user comments, NLP techniques have been widely adopted to study app descriptions, privacy policies, and other meta text related to mobile apps. Autocog [56] adapts NLP techniques to characterise the inconsistencies between app descriptions and declared permissions. PPChecker [57] is a system for identifying the inconsistencies between privacy policy and the sensitive behaviours of apps. Recent research by Liu *et al.* [58] tries to solve the problem of compliance analysis between GDPR and privacy policies, utilising a combination of sentence classification and rule-based analysis. However, the corpus suffers from an imbalanced data problem, which negatively affects the classification accuracy. We build on these techniques to investigates the children's app behaviour through comments analysis.

4.4 Test Environment and Analysis Pipeline

Figure 4.2, Figure 4.3, and Figure 4.4 present an overview of our analysis pipeline. We combine the advantages of both static and dynamic analysis, as well as performing using content rating and user comment analysis. Our goal is to triage suspicious apps and analyse their behaviours in depth. Specifically, we evaluate apps (with targeted users including children) to find evidence of violations against privacy regulations and app store policies. Our pipelines covers 20,195 different Android apps, and we check for improper use of trackers

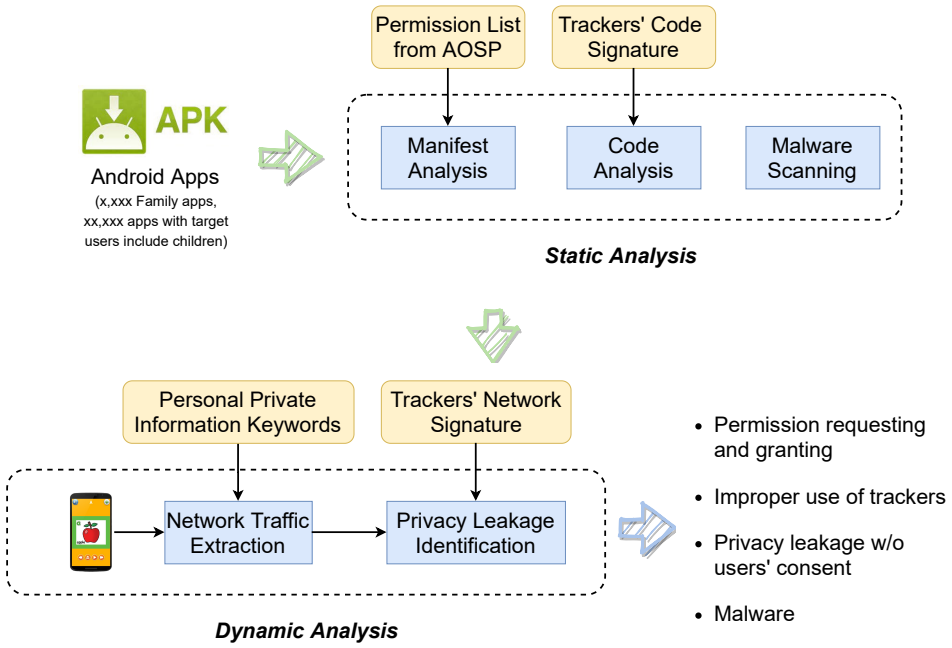


Figure 4.2. An overview of static and dynamic analysis.

in children's apps, privacy leakage without user consent, inconsistent content ratings from different countries, and complaints by users. All of the apps are downloaded from the Google Play Store using a Google Play scraper.

Our pipeline performs several tasks. We first decompile the APK file of the Android app under-study, and conduct a static analysis to determine what permissions have been requested and what trackers are used in the app. Next, we execute each app version individually on a physical mobile phone with a network monitor, which allows us to observe apps' run-time behaviours. We further collect the content ratings using different country settings and identify improper ratings according to inconsistent levels. Finally, we analyse users' comments with both NLP-based topic modeling and rule-based detection techniques to identify undesired app behaviours from users' complaints (especially improper content for children).

4.4.1 Data Collection

We collect apps and their content ratings, as well as the user comments from Google Play store. Here we describe how we collect the data.

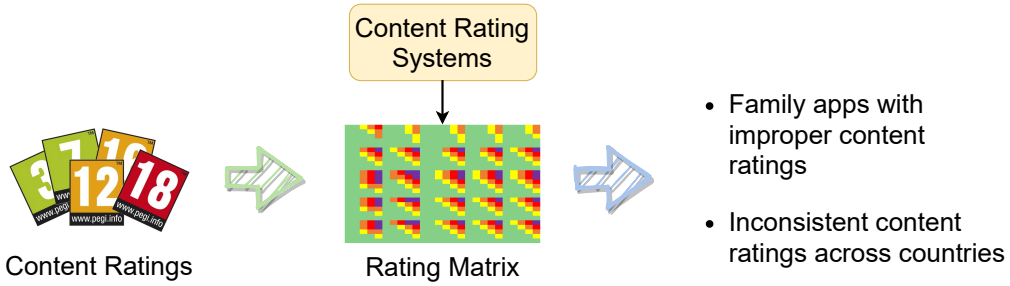


Figure 4.3. An overview of content rating analysis.

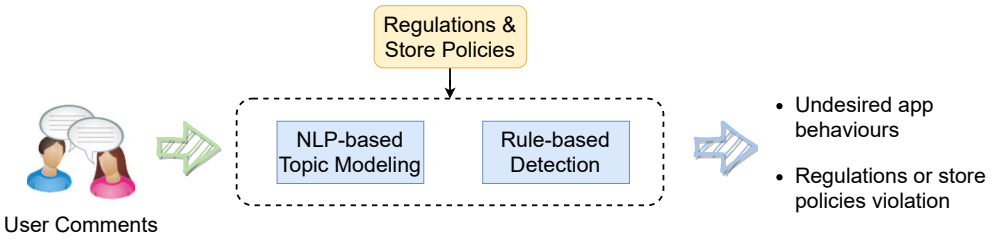


Figure 4.4. An overview of user comment analysis.

App collection. We wrote a Google Play Store scraper to download the most popular children apps under each category. We collect 3,627 apps that participate in the Design for Families program. We refer to this group as "Family apps" as their target users are purely children and families (categorise as "Children" in Play Store). We also collect 16,568 apps that are not particularly designed for children, but include children as target users. We refer to these as "Normal apps" (with content ratings of "Everyone", "Everyone 10+", and "Teen"). Note, we download all apps from the U.S. Google Play Store.

Because the popularity distribution of apps is long tailed, our analysis of the 20,195 most popular apps is likely to cover most of the apps that people currently use (from Mar 2021 to Dec 2021). We instrument the scraper to inspect the Google Play Store to obtain application executable APK files, and their associated metadata (*e.g.* number of installs, category, developer information, content rating, *etc.*).

Content ratings collection. We collect the content ratings of each app in the dataset from different countries by changing the country setting in the URL of each app when visiting the Play Store. We gather data for Australia, France,

Germany, UK, and USA. For example, the URL https://play.google.com/store/apps/details?id=[package_name]&hl=en&gl=US will lead us to the Play Store in the USA and if we change"gl=US" to "gl=FR", it will present the app information to users from France. The reason we selected these five countries is that they are using different content rating standards that cover the main choices around the world, including ACB (Australia - games only), ESRB (North & South America), PEGI (Europe & Middle East), USK (Germany), and IARC (other countries).

For a fair comparison, we only keep the content ratings of an app if it is available in all five countries. However, as we download all the apps from the U.S. Play Store, there is no Normal App rated as "Mature" and "Adults Only" in ESRB in our dataset. Finally, we successfully collect content ratings for 9,453 apps.

User comments collection. For each app, we further collect their 3,000 most recent comments. Note, as we aim to find undesired app behaviours and policy violations from user comments, we only collect the comments with rating stars below 2. This focuses our analysis on negative feedback. We further remove comments with fewer than 5 words. Finally, we successfully collect 13,132,577 comments for 11,831 apps.

4.4.2 Static Analysis

Our static analysis focuses on three parts: Manifest Analysis, Code Analysis, and manual inspection. We use these to measure the use of dangerous and signature permissions, as well as trackers in children apps. To 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 using AndroGuard [59].

Permissions requested. The de-compiled `AndroidManifest.xml` file is parsed to extract the permissions requested by the app. We further categorise the permissions into dangerous, signature, and normal permissions, according to the permission list of Android 11 (API level 30) from the Android Open Source Project (AOSP).[1] We then compare the Family apps and Normal apps with respect to the most popular permissions and the numbers of permissions requested by each app. For top cases that have requested large number of dangerous or signature permissions, we further look into their functionality and

[1] Available at https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/AndroidManifest.xml

descriptions in the app store to determine that whether the permission requested are reasonable.

Through Manifest Analysis, we collect the dangerous and signature permissions requested by the apps. We classify permissions as follows. (*i*) Dangerous permissions are permissions that could potentially affect the user's privacy or the device's operation. The user must explicitly agree to grant such permissions. These include accessing the camera, contacts, location, microphone, sensors, SMS, and storage. However, children may not fully understand the description of permissions and grant them unaware. Therefore, we would like to find cases that request a large amount of dangerous permissions and compare the permission requests between Family apps and Normal apps. (*ii*) Signature permissions are designed for applications to communicate with each other (which may not pose a risk to the user's privacy). There are several signature permissions that need users' consent to be granted. We would like to investigate whether they are properly implemented in the children's apps.

Trackers used. In code analysis, we extract all the class names in the decompiled source code and match them with trackers' code signatures. We adopt a list of tracker from Exodus Privacy with 693 known trackers[2] and extend the list with 32 more trackers reported in other online resources. We then search for class names that contain any code signatures from the tracker list. For example, an app that contains `com/adcolony` in its source code will be detected as containing tracker "AdColony", which matches its signature `com/adcolony/`, `com/jirbo/adcolony/`.

Note, our static approach may report false positives. It cannot determine whether the trackers are active or not during run-time. However, we argue that such static analysis still reflects potential tracking of users, especially when an app contains a large amount of trackers. We further compare the use of trackers in Family and Normal apps. Finally, we manually review the Normal apps with most trackers to check whether there are age screens implemented.

4.4.3 Dynamic Analysis

Our static analysis focuses on detecting the embedded third-party SDKs that potentially collect and disseminate personal children data to the Internet. We

[2] Available at https://etip.exodus-privacy.eu.org/trackers/all

compliment this with dynamic analysis, to collect evidence of personal data dissemination.

App execution. To analyse whether personally sensitive information is leaked to third-parties without user consent, we rely on the automatic method previously proposed by Feal *et al.* [60]. We launch each app and run it for 5 minutes without interacting with it. This implies that we do not actively consent to data collection and we do not carry out any of the children actions, opting instead to leave the app running with no input.

For this, we implement the dynamic testing environment described in Figure 1.1, which consists of four Xiaomi Notebook 9 Android phones running a rooted Android 10. This allows us to comprehensively monitor the network traffic of each of the 20,195 Android apps. We execute each app automatically using the Android Automator Monkey [61] to achieve scale by eliminating any human intervention. The Monkey is a UI fuzzer which injects a pseudo-random stream of simulated user input events into the app. Concretely, we start each app with the Android Debug Bridge (ADB) command, `adb shell monkey -p [package name] n`, where the "`-p`" parameter specifies the package to run, and `n` indicates the number of events. Here we set `n` as 1. After 5 minutes, we force stop and uninstall the app. The logs are then cleared and the device is ready to be used for the next test. We store the resulting network traffic in a database for offline analysis, which we discuss in detail later.

Network monitoring. We monitor all network traffic, including TLS-secured flows, using a network monitoring tool, Lumen Privacy Monitor [45]. This has shown to be effective in several prior research activities [62, 60, 63, 42]. The network monitoring module leverages Android's VPN API to redirect all the device's network traffic through a localhost service that inspects the traffic, regardless of the protocol used. It reconstructs the network streams and ascribes them to the originating app by mapping the app owning the socket to the UID as reported by the `proc` filesystem. Furthermore, it also performs TLS interception by installing a root certificate in the system trusted certificate store. This technique allows it to decrypt TLS traffic unless the app performs advanced techniques, such as certificate pinning, which can be identified by monitoring TLS records and proxy exceptions [45].

Personal information in network flows. We define "personal information" as any piece of data that could potentially identify a specific individual and

Table 4.2. Summary of personal identifiable information checked in network transmission data.

PII	Description	Risk
Device Model	Identifies device model and manufacturer.	Low
Brand	Identifies phone brand. When combined with other information, it can be used to identify a user uniquely.	Low
Board Info	Identifies hardware and the phone model.	Low
Build number	Identifies uniquely the Android OS and the version.	Low
MAC Address WLAN0	Identifies uniquely the WiFi AP users are connecting to, leaking users' activities and location.	Mid
Private IP	By leaking the private IP address of your device, an ad network, tracker or application developer can better identify unique users.	Mid

distinguish them from another. Online companies, such as mobile app developers and third-party advertisement networks, want this type of information in order to track users across devices, websites, and apps (to better target ads). For this reason, we are primarily interested in examining apps' access to the persistent identifiers that enable long-term tracking, as well as their geolocation information.

We focus our study on detecting apps that access specific types of sensitive data without user consent, including persistent identifiers and geolocation information. Notably, the unauthorised collection of geolocation information in Android has been the subject of prior regulatory action [16]. Table 4.2 and Table 4.3 list the different types and risk levels of personal information that we look for in network transmissions. Note that, some of the PII (*e.g.*, device model) may be needed for proper app functionality and is generally not considered sensitive within the industry. Therefore, we consider them as low risk.

4.4.4 Content Rating Evaluation

Measuring rating inconsistency. To investigate inconsistent content ratings, we measure the inconsistency with levels from 0 to 4, depending on the distance of suitable age groups between each categories. Specifically, we consider the inconsistency between two ratings from authority A and B in the

Table 4.3. Summary of personal identifiable information checked in network transmission data (continue).

PII	Description	Risk
Device Fingerprint	A fingerprint of user device which can be used by analytics and ad services to track user.	High
Location	The location data of a user.	High
Timezone	Identifies the current timezone.	High
IMEI	The IMEI (International Mobile Station Equipment Identity) identifies the device uniquely, which could be used to track user's traffic and online behaviour.	High
Serial number	Allows ad networks and online trackers to identify a user uniquely for tracking, surveillance or advertising purposes.	High
Advertising ID	A unique string of characters that identifies the user's device, for purposes like measuring app usage and ad penalisation.	High

following manner. (*i*) We define the suitable age group for rating A a range $[a_1, a_2]$, where a_1 is the minimal allowed age of rating A and a_2 is the minimal allowed age of the next rating level minus 1. For example, the suitable age group for "USK 6+" is $[6, 11]$, as the next rating level is "USK 12+" for which the minimal allowed age is 12. (*ii*) We determine the inconsistent level between two ratings A and B, according to the gap between two age groups, *i.e.*, $b_1 - a_2$, as shown in Equation 4.1.

$$Inconsistent\ Level = \begin{cases} 0, & if\ b_1 - a_2 \leq 0, \\ \lfloor \frac{(b_1 - a_2)}{T} \rfloor + 1, & if\ 9 \geq b_1 - a_2 > 0, \\ 4, & otherwise. \end{cases} \tag{4.1}$$

where T is the threshold between two inconsistent levels (we increase the level by 1 for each T years gap). We set T as 3 in the measurement as for most content ratings the interval between neighbour levels is around 3. For example, if the two age groups are overlapped, *i.e.*, $b_1 - a_2 < 0$, we determine the inconsistent level as 0; while for a 9 years gap, we determine the inconsistent level as 4. (*iii*) We manually increase the inconsistent levels by 1, if rating A is suitable

Rating from Authority A			Rating from Authority B																												
			ACB					ESRB					PEGI						USK					IARC								
			G	PG	M	M15+	R18+	E	E 10+	Teen	M 17+	Adults	3	PG	7	12	16	18	All	6+	12+	16+	18+	3+	7+	12+	16+	18+				
ACB	G		0	0	1	1	2	0	0	0	2	2	0	0	0	0	1	2	0	0	0	1	2	0	0	0	2	2				
	PG		0	0	1	1	2	0	0	0	2	2	0	0	0	0	1	2	0	0	0	1	2	0	0	0	2	2				
	M		0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1				
	M15+		0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1				
	R18+		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
ESRB	E		0	0	3	3	4	0	1	2	3	4	0	0	0	2	3	4	0	0	2	3	4	0	0	2	3	4				
	E 10+		0	0	2	2	3	0	0	1	2	3	0	0	0	0	2	3	0	0	0	2	3	0	0	0	2	3				
	Teen		0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1				
	M 17+		0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1				
	Adults		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
PEGI	3		0	0	4	4	4	0	2	3	4	4	0	0	1	3	4	4	0	0	3	4	4	0	1	3	4	4				
	PG		0	0	0	0	2	0	0	0	1	2	0	0	0	0	1	2	0	0	0	1	2	0	0	0	1	2				
	7		0	0	2	2	3	0	0	1	3	3	0	0	0	1	2	3	0	0	1	2	3	0	0	1	3	3				
	12		0	0	0	1	2	0	0	0	1	2	0	0	0	0	1	2	0	0	0	1	2	0	0	0	1	2				
	16		0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1				
	18		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
USK	All		0	0	4	4	4	0	2	3	4	4	0	0	1	3	4	4	0	1	3	4	4	0	1	3	4	4				
	6+		0	0	2	2	3	0	0	1	3	3	0	0	0	1	2	3	0	0	1	2	3	0	0	1	3	3				
	12+		0	0	0	1	2	0	0	0	1	2	0	0	0	0	1	2	0	0	0	1	2	0	0	0	1	2				
	16+		0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1				
	18+		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
IARC	3+		0	0	4	4	4	0	2	3	4	4	0	0	1	3	4	4	0	0	3	4	4	0	1	3	4	4				
	7+		0	0	2	2	3	0	0	1	3	3	0	0	0	1	2	3	0	0	1	2	3	0	0	1	3	3				
	12+		0	0	0	1	2	0	0	0	1	2	0	0	0	0	1	2	0	0	0	1	2	0	0	0	1	2				
	16+		0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1				
	18+		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Figure 4.5. Inconsistency level matrix for content rating evaluation.

for age under 16 and rating B is "ACB M15+", as "ACB M15+" is more restrict than "ACB M" although they both have suitable age group as $[15, 17]$.

After assigning the inconsistency levels, we manually adjust the levels according to the descriptions provided by the rating authorities. For example, we increase the inconsistency level between "IARC 3+" (suitable for ages 3 to 6) and "ESRB 10+" (suitable for ages 10 to 12) from 1 to 2, as mild violence and bad language are not permitted in "IARC 3+" but "ESRB 10+" allows "mild violence, mild language and/or minimal suggestive themes" [23, 18].

To summarise the above, Figure 4.5 is a matrix showing the level assigned to each combination of ratings. Using this, we later compare the content ratings of each app. Further, for any that have inconsistency levels larger than 3, we manually review them (including their description and Android packages provided across different territories) .

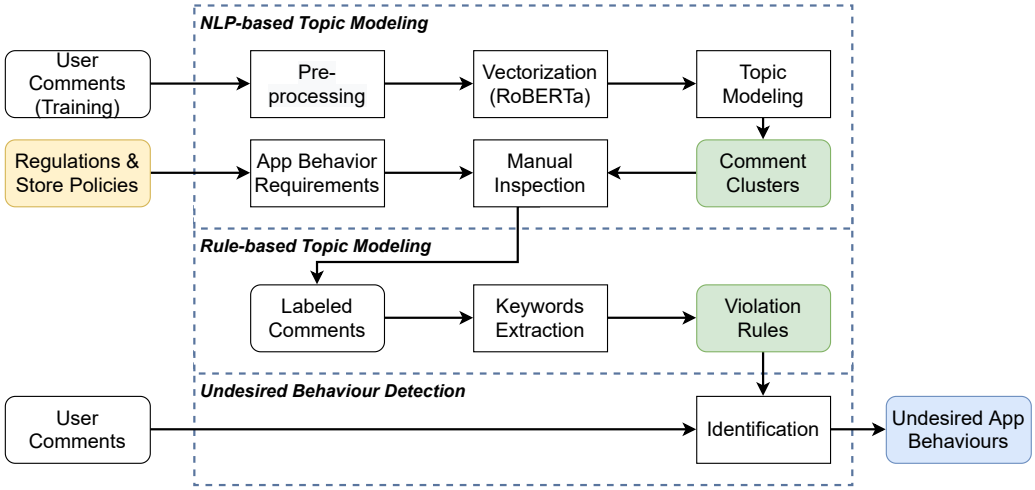


Figure 4.6. Summary of user comment analysis pipeline

4.4.5 User Comments Analysis

Beyond the static and dynamic analysis of apps, the comments left by users may also highlight the violation of regulations or policies, including user concerns. Recent research [64] relies on user comments to identify violations against mobile application market policies, using a semi-automated rule-based process. Figure 4.6 presents an overview of our comment analysis pipeline. We adopt the state-of-the-art text mining and natural language processing (NLP) techniques (such as transformers) to interpret the application comments and train machine learning models to categorise and identify informative comments.

First, 18 comment clusters are obtained from K-means clustering based on the training set of user comments; next, we extract keyword lists using TF-IDF [65] from each cluster and manually select and merge the lists into 15 keywords sets, representing 15 user complain topics; at last, 19 semantic rules are extracted based on the keywords sets and manually labelled comments (50 comments from each topic). Such comment analysis complement the static and dynamic analysis results (since some application behaviours may not be identified by technical analysis, but could be better identified by users). The rules will be further used in the recognition of regulation violations or undesired app behaviours.

Pre-processing and embedding. We filter out comments with a length less than 5 words. We further transfer emojis into words, such as transferring the

emoji `ud83d` (smiling face with open mouth) to "smiley", using the `emoji` Python package.[3] We utilize the pre-trained RoBERTa [66] embedding to vectorize each comment into a 300-dimensional vector. RoBERTa (Robustly optimized BERT approach) is a retraining of BERT [67] with an improved training methodology and 10x more training data to improve the BERT performance. It achieves state-of-the-art results (2% to 20% improvement over BERT) [68].

Comment clustering. The next step of user comments analysis is to cluster the comments to tease out the different concerns that users describe. These include complaints about functionality, performance, advertisement, personal data collection, vulgar content, violence, and payment deception. We refer to these as *undesired behavior topics*. As the Play Store user comments do not have fine-grained labels for these topics, we use unsupervised learning to cluster comments into types.

We rely on K-means clustering, to identify the user concerns. We use K-means clustering as our experiments expose good results; we leave exploration of other clustering solutions to future work. Without knowing how many distinct topics users may write about, the challenge of applying K-means in our clustering task is how to determine a proper k value (*i.e.* the number of target clusters). Recent work by Nema *et al.* [69] proposes a Summarisation Metric as shown in Equation 4.2, aiming for a k that results in well separated clusters and a high number of compact clusters.

$$Summarization\ Metric = dist_k * M_k, \tag{4.2}$$

where $dist_k$ is the minimum of the cosine distance between all pairs of k cluster centres, and M_k is the number of compact clusters. A compact cluster is a cluster in which at least 30% of the samples have silhouette scores higher than the average silhouette score in this cluster.

We iterate through $k = 5, 10, 15, ..., 100$ and chose k for which the summarisation score is highest, as we want to increase both $dist_k$ (distance between cluster centres) and M_k (number of compact clusters). We consider the range of k as 5 to 100 as we would like to focus on dominant user concerns at first. A larger

[3] Available at https://github.com/carpedm20/emoji.git

k may include more undesired behaviour topics in the clustering results, but could also lead to a too heavy workload in the later manual inspection.

To summarise the topic of each cluster, we further conduct a manual inspection on the clustering results. Specifically, we select the top 20 (per cluster) representative comments that can be considered as representative of the entire cluster. We rank a comment's representativity using their silhouette scores. We carefully analyse these representatives manually across all clusters and determine the topic of each cluster. Considering that not all user comments focus on the expression of concerns or complaints, we only keep the 18 clusters with topics related to undesired app behaviours, and drop the remaining 22 clusters.

Semantic rules extraction. If we directly use K-means to recognise users' concerns, it may lead to a high false positive rate. This is because we can only classify each comment based on its distances to the cluster centres, and we have to assign a label even if it may not be relevant. Therefore, based on the clustering results, we apply a rule-based approach which assigns topics to comments according to semantic rules, *i.e.*, if a comment is matched to a semantic rule, the comment will be recognised as belonging to the corresponding topic. Specifically, for each cluster, we first remove meaningless words (the stop-words according to the NLTK English stop-words list [70]) and sort the remaining words in descending order based on TF-IDF weighting [65] to generate a word list. Then we manually select keywords for each word list and merge the word lists that usually appear in the same topics. We finally obtain 15 topics in 4 categories as shown in Table 4.4. For each topic, we obtain a keyword set containing one or more representative keywords. At last, we generate semantic rules in a form of $\{w_1, w_2, d, t\}$ for each keyword set, where w_1 and w_2 are two keywords, d is the distance constrain between w_1 and w_2, t is the topic that a matched comment will be assigned to. Concretely, we first manually select 50 representative comments from each topic as labelled samples. Then we traverse each keyword set by calculate F1-scores for each single keyword (set w_2 as `null` and d as 0) and each pair of keywords under different distance constraints (from 1 to 20), and select the rules with F1-score larger than 0.8. For example, a semantic rule could be $\{kid, improper, 2, not\_proper\_for\_kids\}$, which detects any comments that contains the keywords `kid` and `improper`, and the distance between the two keywords are smaller than 2 words. Manual inspections are further conducted on a pilot experiment on 5,000 labelled comments and rules

Table 4.4. List of user comment topics.

Categories	Topic Descriptions
Content	• The apps contain violence, blood or scaring contents that are not proper for kids. • The apps include sexual content that is not allowed for kids. • The app content encourages the use of tobacco or drugs. • The apps expose inappropriate language to children. • The apps present depicting criminal activities to kids.
Ads	• The app provides too many advertisements. • The users are disrupted by ads. • Ads shortcuts in launching menu or notification bar. • Redirection or drive-by download by ads.
Privacy	• The app leaks or steals users' private information. • The app abuse the permissions (e.g., requesting unnecessary permissions). • The app collects unnecessary private data. • The app shares data with third-parties or other users without user's consent.
Security	• The apps contain virus or malware. • The app is suspicious to payment fraud.

with error rate larger than 10% are removed. At last, we obtained 19 semantic rules for user comment analysis.

We note that, as the semantic rules are extracted from representative comments in clusters, this approach may still focus more on the representative comments and can only provide a lower bound of the detection of users' complaints on undesired app behaviours or regulation violations.

4.5 Results

In this section, we present the measurement results from applying our analysis pipeline to 3,627 children apps that participate in the Google Family project (the "Family Apps"), and 16,568 normal apps with kids included in the target user group (the "Normal Apps").

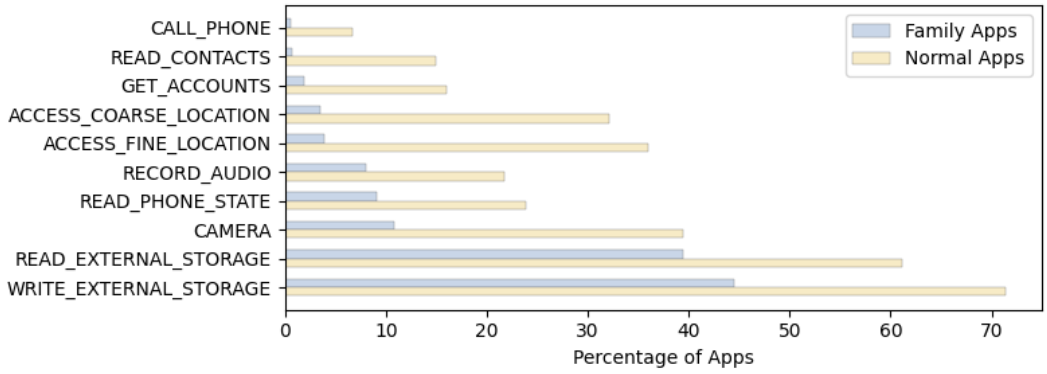


Figure 4.7. Top-10 dangerous permissions requested.

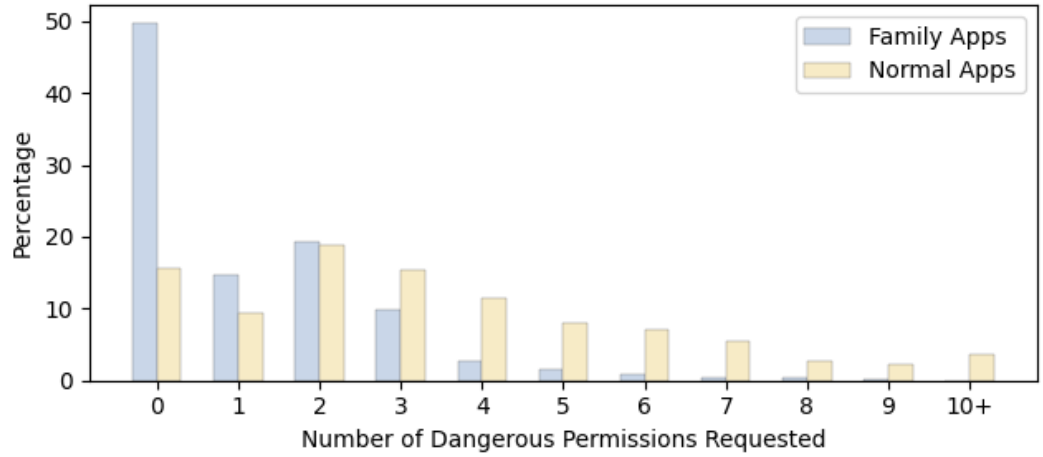


Figure 4.8. Distribution of number of dangerous permissions.

4.5.1 Permissions

We start by examining the use of permissions, based on our static analysis.

Dangerous permissions. The results of top 10 most requested dangerous permissions in Family apps and the distribution of numbers of dangerous permissions requested per app are shown in Figure 4.7 and 4.8.

From Figure 4.7, we see that Normal apps request dangerous permissions more frequently than Family apps. For the top 10 most frequently requested dangerous permissions, the proportion of Normal apps requesting these permissions is 6.07% to 32.16% higher than that of Family apps (per permission). Notably, 3.46% of Family apps request location-related permissions, which is potentially violating the Design for Family policy. Further, from Figure 4.8, a larger fraction of Normal apps request over 3 dangerous permissions. 29.15% of Normal

apps request more than 5 dangerous permissions, compared to just 3.56% in Family apps. These results show that dangerous permissions are being used more carefully in the Family apps. However, note that all the Normal apps in our research have target users that include children (less than 12 years old). We argue that developers should be equally cautious about using dangerous permissions as long as the target users include children.

Among these permissions, some are directly related to sensitive personal information, such as `CALL_PHONE`, `ACCESS_FINE_LOCATION`, `RECORD_AUDIO`, `RECORD_AUDIO`, and `CAMERA`. We have manually checked 200 apps randomly sampled from Family and Normal apps, and did not find any permission request notifications specifically designed for children users (*i.e.*, a notification to ensure that a parent receives a *direct notice*, as required by COPPA and GDPR). This may mean that children simply click "yes" buttons, without understanding what information the developer or third-parties will collect. An appropriate design should provide guidance to children with a graphical and textual interface, ensuring that the children's PII is only obtained with parental consent.

Signature permissions. Figure 4.9 and 4.10 presents the permission results for the Normal apps. We see that Normal apps request more signature permissions than Family apps do. Optimistically, one might argue that a signature permission will only be granted when the requesting app is signed with the same certificate as the app that declared the permission, and a signature permission is limited to sharing specific features (*e.g.*, communication). However, abuse of signature permissions could still harm the privacy of users, especially children users. For example, `SYSTEM_ALERT_WINDOW`, the most frequently requested signature permission, allows an app to display a window over any other app, with no notification for the user. This functionality can be abused to display fraudulent ads, phishing scams, click-jacking, and overlay windows. These are common with banking Trojans that create windows identical to a banking app's login page, as well as ransomware that creates a persistent on-top screen. Although an app has to have explicit, manual approval from the user to use these signature permissions, again, children user need more protection against the abuse. We argue that a list of (dis)allow permissions in children's apps should be established, especially for the permissions that are (*i*) related to PII; and (*ii*) easier to be exploited when user is a child.

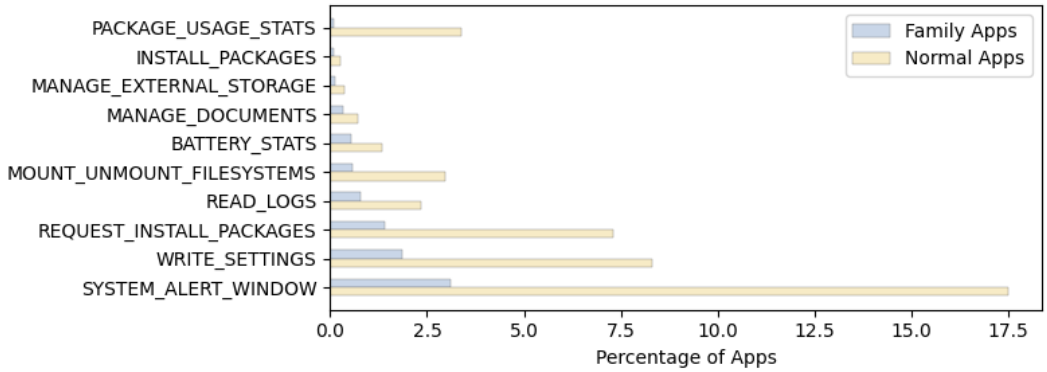


Figure 4.9. Top-10 signature permissions requested.

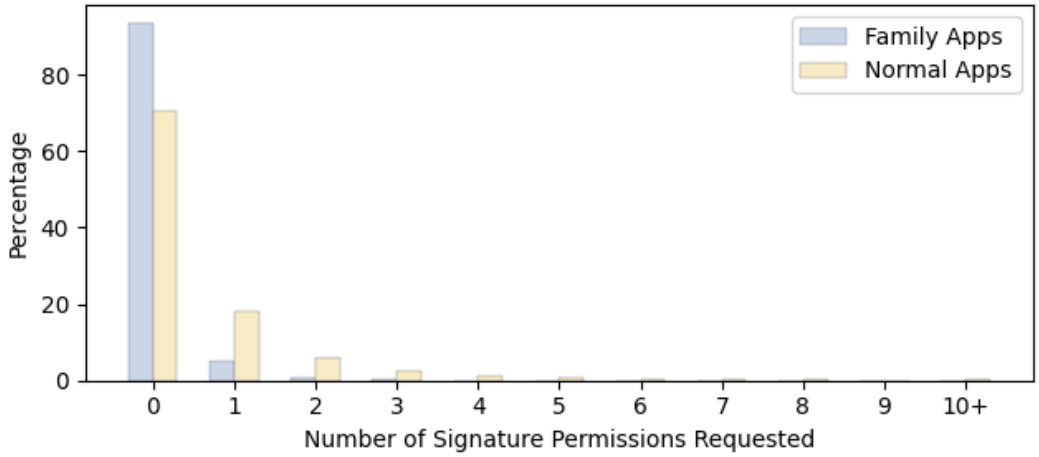


Figure 4.10. Distribution of number of signature permissions.

Exemplar case study: YouTube Kids. The YouTube Kids app [71], a Google product built with children in-mind, aims to make it safer and easier for children to explore videos. This provides a good example of permission request notifications for children. As shown in Figure 4.11(a), when the app is opened for the first time, a simple and clear notification is presented to users, noting that a parent is required to unlock the app. If the user selects "I'M A KID", the app requires the child to get their parent, and the only choice is going back to the previous screen (Figure 4.11(b)). Only after the "I'M A PARENT" button has been clicked, the app starts showing a introduction video (Figure 4.11(c)) and obtaining permissions from parent user (Figure 4.11(d)). Although a child may fool this process and login with their parent's account, the app tries its best to ask consent from guardians. Through our manual checks on 500 randomly selected Family and Normal apps, we did not find other permission notification designed for children users in other apps.

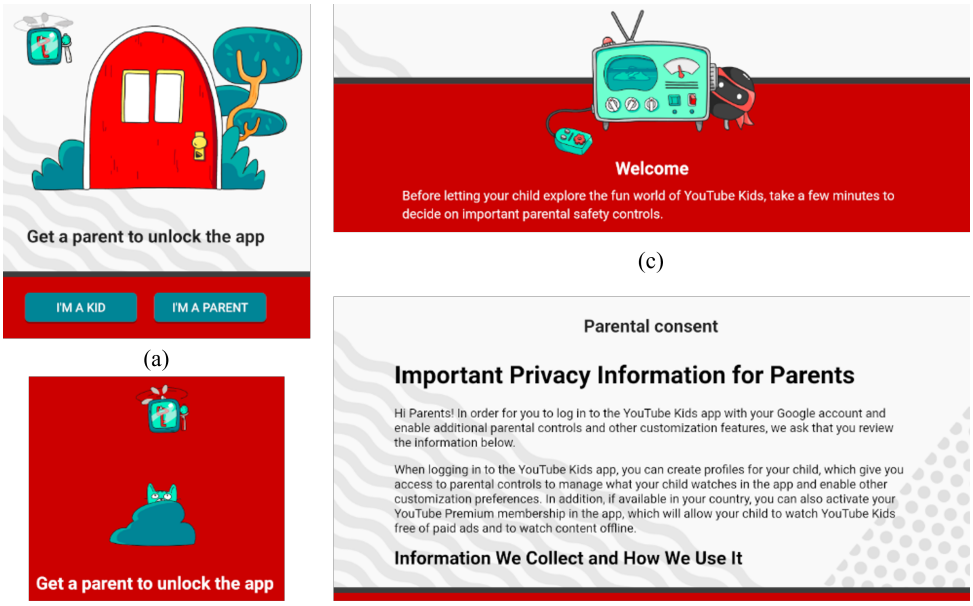


Figure 4.11. Examples of YouTube Kids notifications.

Finding summary:

- 3.46% of Family apps request location-related permissions, which is potentially violating the Design for Family policy.
- The proportion of Normal apps requesting dangerous permissions is 6.07% to 32.16% higher than that of Family apps. However, only x out of xxx have permission notification specifically request consent from parents.
- More restriction should be applied on permission requests when target user includes children, *e.g.*, strictly requiring consents from parents and establishing a list of (dis)allow permissions for children apps.

4.5.2 Third-party Trackers

To determine the usage and the potential privacy leakage through third-party trackers, we next leverage both static and dynamic analysis.

The usage of third-party trackers. The static analysis results on the use of third-party trackers in both Family and Normal apps are presented in Figure 4.12. The trackers on the left side of the vertical red dash-line are allowed by the Design for Family program. `Google AdMob` is the most frequently used

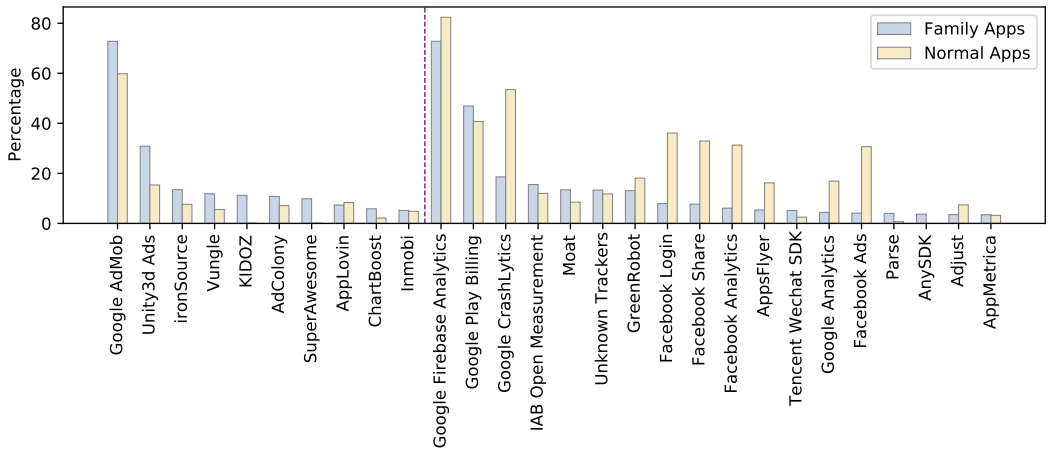


Figure 4.12. Distribution of apps with respect to number of non-kids trackers used.

ad SDK in both Family and Normal apps. Not surprisingly, Family apps use the kids-allowed SDKs more frequently, especially for SDKs such as `KIDOZ`, `SuperAwesome`, and `ChartBoost`. This could indicate that the Play store's Design for Family program contributes positively to app development — more developers consider using the recommended SDKs when their target users are children. However, as shown by the right side of the vertical red dash-line, we find that quite a few Family apps still use SDKs that are not allowed for children apps, *i.e.*, not in the Google Play certified ad SDKs in Table 4.1. For example, more than 72.80% Family apps use `Google Firebase Analytics` and the number for `Google Play Billing` is 46.91%; while other tracker SDKs such as `Google CrashLytics`, `Facebook Login`, and `ApppsFlyer` are more frequently used in Normal apps. Considering that some tracker SDKs are very popular among both Normal apps and Family apps, and they may provide irreplaceable functionalities to apps, we argue that a list of kids-allowed tracker SDKs is essential to protect the privacy of children users. This will likely also benefits the app developers quite a lot.

As shown in Figure 4.13 shows the distribution of the number of trackers per app. Worryingly, only 18.75% of Family apps did not use a disallowed ad or tracker SDK, whereas 95.43% of Normal apps use SDKs that are disallowed when the target users include children. Even when we exclude SDKs from Google and Facebook, the numbers are still high: 38.49% of Family apps and 28.25% of Normal apps use at least one non-kids SDK. In fact, 1.63% of Family apps and 5.54% of Normal apps use over 10 non-kids SDKs. From these results, we confirm that the Design for Family requirements are *not* followed by all app

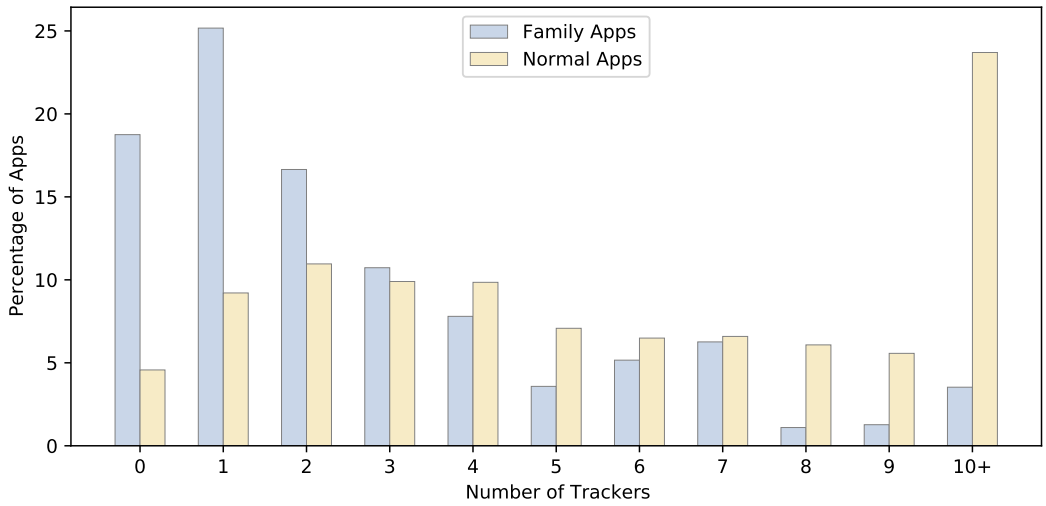


Figure 4.13. Distribution of apps with respect to number of non-kids trackers used.

developers. We note that, through static analysis, we cannot determine which ad or tracker SDKs are active during run-time. However, dynamic analysis may still be incomplete as it is hard to catch all tracking. We posit if the trackers are not necessary, the app developers should remove the SDKs from a children app.

Privacy leakage. We further analyse the specific privacy leakages via both tracker SDKs and first-party APIs. We identify that 3.79% Family apps leak the device model, brand, builder, and other PII to third-party trackers without any consent from users. Table 4.2 presents the top 5 leaked PII that has been marked as Mid or High risk in. Figure 4.14 further shows the destination of this PII. We observe that `Google` collects a large amount of `Private IP`; and `Supersonic Ads` obtains `Advertisement ID`, `Location Information`, and `Android Serial` without users' consents. Furthermore, `Location Information` is sent to `Facebook`, and `Babybus` collects the `Android Serial`. We cannot check how the leaked PII is used, but emphasize that the leaking of PII has already violated the GDPR and other privacy regulations (which is forbidden no matter if the target users are children or not). We note again that our results only present a lower-bound of the PII leakage without users' consents.

Case study: Excessive tracking. During the experiments, we found an interesting phenomenon: apps from the same developer tend to have similar trackers, even when the apps' functionalities or content are different. For example, we found that 6 apps from `Tu***ns`, a large player in the childrens game

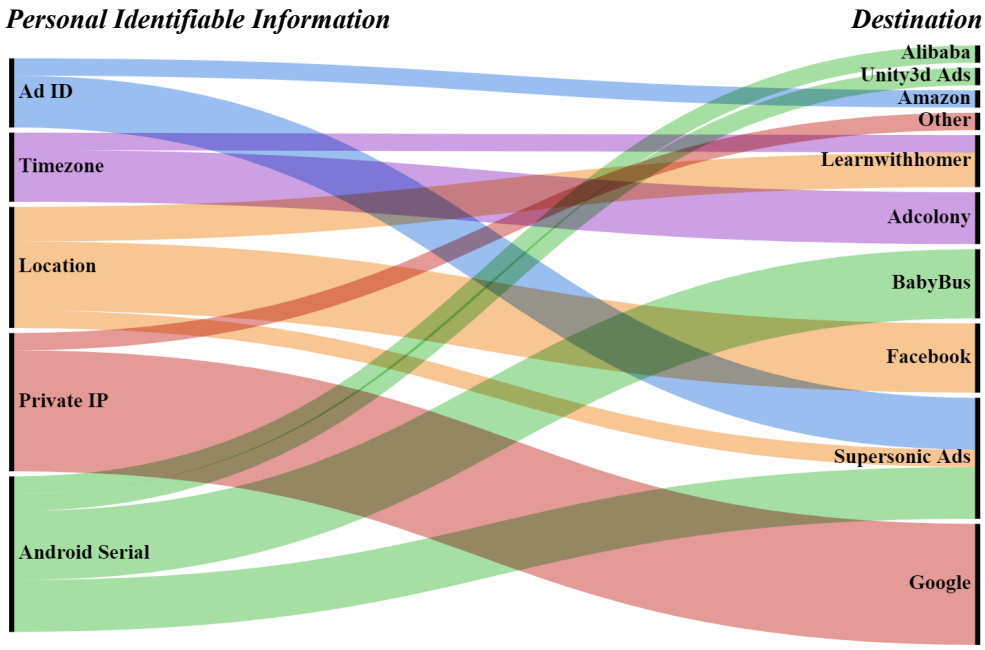


Figure 4.14. Sankey diagram representing the flow of PII to different destinations.

market (published over 100 games and achieved more than 950 million downloads according to its official website), has **47** trackers. Although `Tu***ns` has a privacy policy which listed 17 ad or tracker SDKs, there are still 30 trackers that have not been mentioned. It is therefore unclear if parents or children acknowledge that their personal information could be shared with 17 (or even 47) third-party trackers.

We investigate more apps to see whether there exists large players who have large amount (*e.g.*, more than 40) of children apps listed in Play store that embed high numbers of trackers. In Figure 4.15, we pick 4 developers from our static results and list the distribution of numbers of trackers in their products. 142 out of the 150 apps (94.7%) from `Ba***us` have 5 to 6 trackers; 105 out of 133 apps (78.9%) from `Hi***es` have 20 trackers; and 29 out of 43 apps (67.4%) have 14 trackers. The distribution of numbers of trackers is less concentrated among apps from `Tu***ns`: 6 out of 62 `Tu***ns` apps have 47 trackers; 14 apps have 34 trackers; and 16 trackers are detected in the rest 41 apps. The reason behind this phenomenon could be that developers directly apply the same tracker configurations while developing different apps, instead of setting trackers for each app according to what personal information should be collected and shared. Further, we conclude that most apps from these 4 big players contains trackers not allowed in children apps.

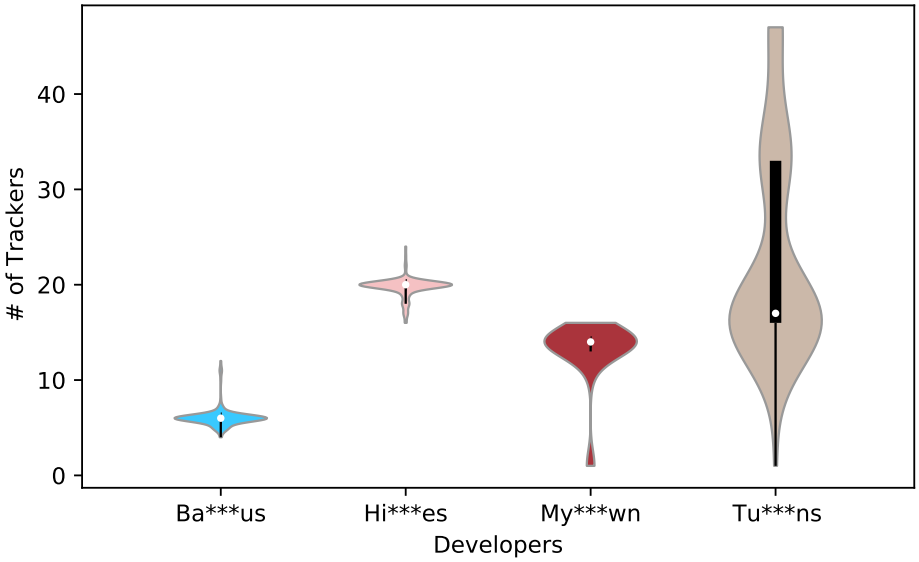


Figure 4.15. Number of trackers used in children apps from 4 big players.

Finding summary:

- 81.25% of Family apps use trackers that are not allowed for children; disallowed trackers are also frequently used in 95.43% Normal apps whose target user includes children.
- 1.63% of Family apps and 5.54% of Normal apps use over 10 ad or tracker SDKs that are not allowed to be used in children apps.
- Even big players do not follow the Play store policy. We found 4 developers who have 5 to 47 disallowed trackers in each of the 43 to 150 children apps they listed on Play store.

4.5.3 Inconsistent Content Ratings

We next investigate into the content (*i.e.*, maturity) ratings of the 9,453 apps across all 5 rating authorities. These are ACB for Australia, ESRB for Americas, PEGI for Europe and the Middle East, USK for Germany, and IARC for generic countries. We seek to check if the content ratings given by these different agencies are inconsistent. Indeed, we find inconsistent ratings in 19.25% apps, and 9.99% apps have an inconsistency level above 3. This indicates that content rating inconsistencies among the various rating authorities are prevalent.

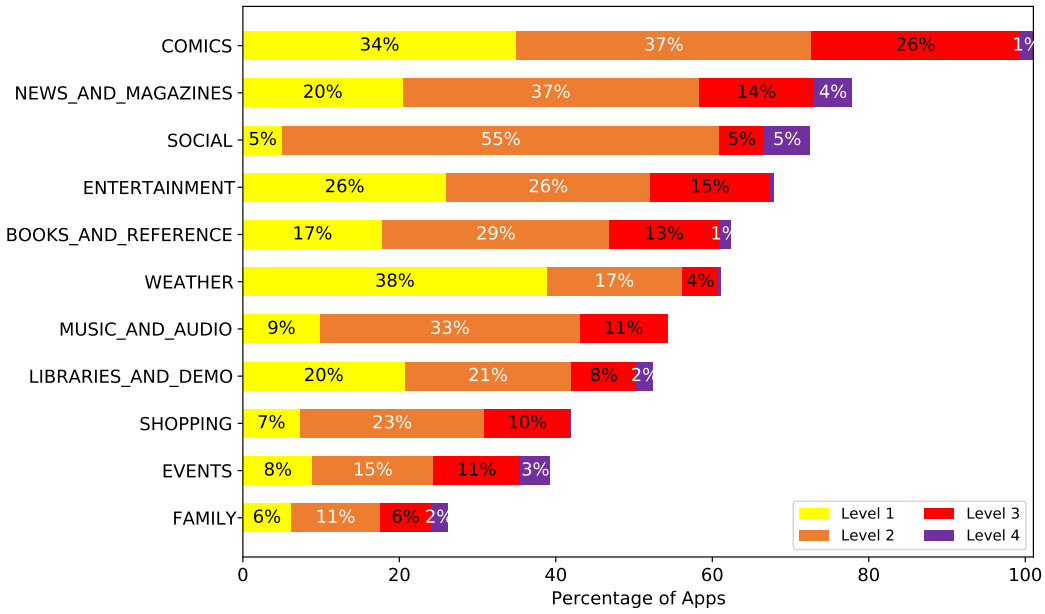


Figure 4.16. Percentage of apps (per category) with inconsistent content ratings

Figure 4.16 presents the content rating results across the top-10 categories of Normal apps *vs.* Family apps. Note, as one app may have multiple inconsistent rating pairs across different levels detected, the sum of apps could be higher than 100%.

Although Family apps have a lower ratio of inconsistent ratings than the other 10 Normal app categories, at least 6% Family apps have an inconsistency level higher than 3. 26% of `COMICS` and 15% of `ENTERTAINMENT` apps have inconsistency leve higher than 3; and at least 55% `SOCIAL` apps have a rating inconsistency level 2. We conjecture that the reason for highly inconsistent ratings in these categories may be because the rules to flag sensitive contents could be different across the rating authorities. However, we argue it is still not reasonable to have an app rated as 18+ in one territory and 3+ in another (level 4). This will confuse parents and increase the risk to children (*e.g.*, allowing a child to play a game rated 18+ in another country). Therefore, we argue that, rather than only displaying the content ratings in the user's current territory, the app store should also provide ratings from other authorities an an option. Briefly listing the reason why the app is rated will also help parents and children to make a decision.

Case study: Highly Inconsistent App Ratings. Here, we show case some inconsistent ratings to highlight key concerns. For example, `com.my***in.app`,

com.my a plant identification app with professional care guides (1,000,000+ installs), has been rated as "PEGI 3" and "Rated for 3+" in IARC. However, when we visit the Play store using `gl=de` and `gl=us`, the ratings change to "USK 12+" and "ESRB Mature 17+". The only explanation we can find is "Drug use". If the app truly contains information related to drugs, it is unsuitable for 3-years old children, however, if there is no drug use content (we manually check the app and only find drug-related information for plant diseases), the labels in PEGI 3 and IARC are unnecessary.

Another example is `com.di***rd`, an instant messaging and digital distribution platform. As of 2021, the service has over 350 million registered users and over 150 million monthly active users (over 100,000,000 installs from Play store). As a social app, it is quite reasonable to be rated as "Parental guidance" in PEGI (as such app may not always have predefined content that can be classified beforehand). Nevertheless, in USK, the app is rated as "18+", which hints that the app contain "drug use, realistic and explicit violence". However, there is no explanation on the app page why the app is rated "18+". If it contains "18+" content, will a parent in the US acknowledge this before they download the app for their teenager, as it is rated as "Teen" in ESRB? Here we argue again, there should be better descriptions that explain why the app is rated, at least for apps not recommended for children.

Finding summary:

- 21.69% of apps have inconsistent content ratings across different rating authorities; this is commonplace.
- There are many examples of highly confusing and inconsistent content ratings, including among Family apps. 6% of Family apps have an inconsistently level above 3.
- The app store should provide ratings from other authorities as references for parents and children users, and explanations of ratings should be more transparent.

4.5.4 User Complaints

Finally, we analyse 13,132,577 comments downloaded from 11,831 apps in the Play store. We strive to measure users' complaints across 4 categories: app content, advertisement, privacy, and security.

Table 4.5. Comment analysis results.

Categories	Family Apps		Normal Apps	
	% Comments	% Apps	% Comments	% Apps
Content	2.12%	30.62%	0.92%	33.86%
Ads	1.01%	25.77%	0.97%	34.81%
Privacy	0.09%	6.91%	0.52%	29.05%
Security	0.16%	12.89%	0.43%	23.76%

Table 4.5 presents the percentage of comments and that have been detected as complaints in each category, alongside the percentage of corresponding apps that they refer to. In 2.12% comments of Family apps, we recognize at least one user complaints related to the app content, while the number is just 0.92% for Normal apps. This indicates that the users of Family apps complain more about the content, which is quite reasonable. However, the percentage of apps are quite similar, 30.62% *vs.* 33.86%. This reflects that the developers of Family app may not apply special controls on app content when the users are children. The ratios of comments related to advertisement complaints are similar in both Family and Normal apps, although this impacts more Normal apps than Family apps (34.81% *vs.* 25.77%). Further, fewer Family apps are complained about with respect to privacy and security than the other two categories. This indicates that the users of Family apps may focus more on app content and advertisements, rather than thinking about privacy and security in Family apps.

Impact of popularity. We next seek to understand the impact of these negative reviews. Figure 4.17 correlates the number of complaints received by an app with both its app rating and number of installations. The x-axis presents the log of the number of installs, the y-axis presents the rating scores. The color presents the log of the number of total comments, and the size of bubbles reflect the number of complain comments.

To focus more on popular apps, we only present the top 100 apps that have most complain comments with a rating score higher than 3.00 and over 1,000,000 installations. A popular app with more installs, more comments, and a higher rating score will be located near to the upper-right corner with a light color. Intuitively, one might expect a popular app may have fewer complaints (so that the bubbles near to the upper-right corner should be smaller), while the sizes of bubbles should increase from the upper-right corner to the lower-left corner. However, this is not true. According to our measurement result, 45 of

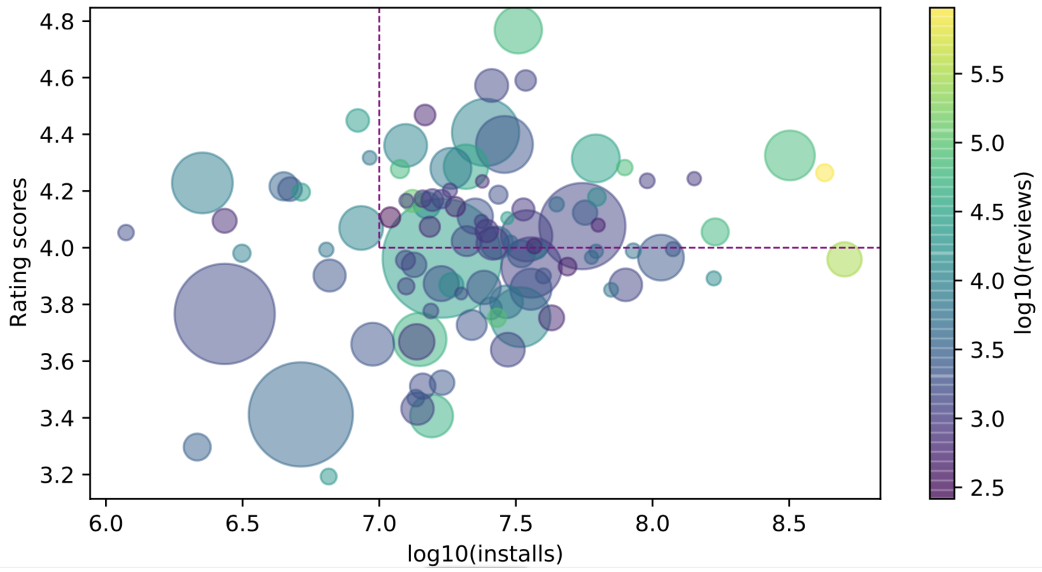


Figure 4.17. Correlation between the number of complaints received by an app with both its rating and number of installations; the colour represents the log of the number of comments, and the size of the bubbles reflect the number of complain comments.

the top-100 apps have rating scores above 4 and over 10,000,000 installations. Such inconsistency between the number of complained comments and the rating score suggests that the app store should expose more negative comments to help users understand the app more comprehensively.

Case study: Apps with many complaints. We first extract the app with the largest number of content complaints: `com.to***en` is an educational app for children. The app has participated in the Design for Family program (marked as "Ages 6-8"), achieving over 10,000,000 installations from the Play store. However, 342 out of the 3,000 comments we downloaded from Play store report that the game is "terrifying" or "scary".

We list some of the comments below and mark the keywords we used in our rules with square brackets. Note that we did not find any response to the comments from developers. *"The game honestly is [terrifying], the characters look [inappropriate for kids], they don't have teeth, the chewing sounds are very [disturbing], I would not recommend this to kids, ...""It's super [scary]...as a sixth grader I came back to play *** and do not recommend installing. It's like a [horror] game and [NOT a good fit for the kids]. So the chewing sounds [disturbing] and the faces are like from little misfortune ..."*

By reading these comments, a user would likely feel that the app is not fit for children. However, the Teacher approved "Ages 6-8" badge, the 4.2 rating score, and the large number of installs are far more obvious than the comments. Parents and children require a better way to acknowledge other users' complaints, rather than scrolling through thousands of comments. We argue that our methodology could be applied to automatically flag these comment-based concerns These could, for example, be presented as a set of word tags on an app's page in the Play Store.

Finding summary:

- 30.62% and 25.77% of Family apps have content-based and ad-based complaints, respectively. Users report privacy and security issues on Family apps far less often though (6.91% and 12.89%, respectively).
- Even highly popular and well rated apps accumulate many complaints — among the top 100 apps that have most complain comments, 45 apps have app rating scores above 4 and over 10,000,000 installs.
- Comments offer an opportunity to automatically generate content warnings for Family apps.

4.6 Discussion & Limitations

False positives and false negatives. Although we validate the measurements via manual inspection, false positives and false negatives may still exist in the reported results, which are inherent in the static and dynamic analysis. Apart from this, there exists other factors that may contribute to false positives and false negatives. For example, our comment analysis assumes users report complaints accurately. However, malicious users (*e.g.* competing app developers) may create fake complaints (false positives). This may hinder the ability to automatically extract content ratings from the comment text. That said, we believe that this should be rare and our results still reflect the undesired behaviours in kids apps. In addition, more privacy leakages may exist in the apps (false negatives) that we have detected. This is because we have focused on detecting privacy leakage without users' operation (*e.g.*, granting consents), thus only providing a lower bound of privacy leakage.

Scalability and manual efforts. In our measurements, dynamic analysis and manual inspection play an important role. Unfortunately, this limits scalability. It is possible to use static analysis methods to improve the detection of privacy leakage without user consent, such as taint analysis, but it may lead to new challenges (*e.g.*, false positives and dead code) and more engineering efforts. In the future, manual inspection in the rule-based comment analysis could be reduced if more robust models were built with the development of NLP technology. We would like to leave these interesting topics in the future work.

Potential improvements to app stores. App stores have established several programs to protect children's privacy while using applications, such as the Designed for Families and Teacher Approved in the Play store. However, according to our results, there is still room for improvement. In the short-term, as suggested in Section 4.5, app stores should provide more information to display and explain content ratings. In addition, we argue manual-based app content assessment mechanisms, (*e.g.* crowd-sourcing or user feedback-driven) should complement the current self-certification rating system before realizing a reliable large-scale automated app analysis system. Finally, we believe family requirements should be embedded in the app development cycle (*e.g.* via SDKs). For example, this could provide a permission notification interface designed for children or disable the forbidden permission and ads if the target users include children.

4.7 Conclusion

In this chapter, we have focused on the privacy practices of apps that are designed for children or have target users that include children. We have measured the use of permissions and trackers, investigated inconsistency in content ratings, and analysed user comment feedback. Our measurement results illustrate that, despite many privacy protection regulations and the strict requirements imposed by the app store, children still experience privacy threats. This is caused by things like permission requests without child-friendly notifications, abuse of ad trackers, confusing and inconsistent content ratings, as well as privacy leakage without users consent. Ultimately, we conclude that the existing self-certification-based content rating mechanism must be improved immediately.

Bibliography

[1] Madeleine Hillyer. How has technology changed - and changed us - in the past 20 years?, 2020.

[2] Gregorio Serra, Lucia Lo Scalzo, Mario Giuffrè, Pietro Ferrara, and Giovanni Corsello. Smartphone use and addiction during the coronavirus disease 2019 (covid-19) pandemic: cohort study on 184 italian children and adolescents. *Italian Journal of Pediatrics*, 47(1):1–10, 2021.

[3] The United Nations Children's Fund. Child protection advocacy brief. https://www.unicef.org/media/96691/file/Online-Protection-Advocacy-Brief-2021.pdf, 2020.

[4] Joshua Barrie. This is how much of the internet is porn, 2017.

[5] Brooke Auxier, Monica Anderson, Andrew Perrin, and Erica Turner. Parenting children in the age of screens, 2020.

[6] Economist Intelligence Unit. Out of the shadows: Shining light on the response to child sexual abuse and exploitation, 2018.

[7] The United Nations Children's Fund. Unicef poll: More than a third of young people in 30 countries report being a victim of online bullying, 2019.

[8] Children's online privacy protection act (coppa). http://www.ftc.gov/ogc/coppa1.htm, 1998.

[9] National Archives. Part 312 - children's online privacy protection rule. https://www.ecfr.gov/current/title-16/part-312, 2022.

[10] European Parliament and Council of the European Union. Regulation (eu) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data and repealing Directive 95/46/EC (general data protection regulation). *Official Journal of the European Union*, 119:1–88, 2016.

[11] Privacy act 1988. https://www.legislation.gov.au/Series/C2004A03712/Compilations, 1988.

[12] Brooke Auxier, Lee Rainie, Monica Anderson, Andrew Perrin, Madhu Kumar, and Erica Turner. Americans and privacy: Concerned, confused and feeling lack of control over their personal information, 2019.

[13] Play Store. Creating apps and games for children and families. https://play.google.com/console/about/families/, 2015.

[14] Mindy Brooks. Find high-quality apps for kids on google play, 2020.

[15] Information Commissioner's Office. Children and the UK GDPR. https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/children-and-the-uk-gdpr/, 2022.

[16] Play Store. Designing apps for children and families. https://support.google.com/googleplay/android-developer/answer/9893335, 2015.

[17] European Parliament and Council of the European Union. Special protection of children's personal data. https://gdpr-info.eu/recitals/no-38/, 2022.

[18] Entertainment Software Association. Esrb ratings guide. https://www.esrb.org/ratings-guide/, 2022.

[19] Pan European Game Information. PEGI age ratings. https://pegi.info/page/pegi-age-ratings, 2022.

[20] Play Store. https://support.google.com/googleplay/android-developer/answer/9898843, 2022.

[21] Unterhaltungssoftware Selbstkontrolle. USK age categories, 2022.

[22] Australian Goverment. The advisory categories for films and computer games. https://www.classification.gov.au/classification-ratings/what-do-ratings-mean, 2022.

[23] International Age Rating Coalition. How IARC works. https://www.globalratings.com/how-iarc-works.aspx, 2022.

[24] Ravi Bhoraskar, Seungyeop Han, Jinseong Jeon, Tanzirul Azim, Shuo Chen, Jaeyeon Jung, Suman Nath, Rui Wang, and David Wetherall. Brahmastra: Driving apps to test the security of {Third-Party} components. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1021–1036, 2014.

[25] Minxing Liu, Haoyu Wang, Yao Guo, and Jason Hong. Identifying and analyzing the privacy of apps for kids. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 105–110, 2016.

[26] Aaron K Massey, Jacob Eisenstein, Annie I Antón, and Peter P Swire. Automated text mining for requirements analysis of policy documents. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 4–13. IEEE, 2013.

[27] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, Serge Egelman, et al. "won't somebody think of the children?" examining coppa compliance at scale. In *The 18th Privacy Enhancing Technologies Symposium (PETS 2018)*, 2018.

[28] Irwin Reyes, Primal Wijesekera, Abbas Razaghpanah, Joel Reardon, Narseo Vallina-Rodriguez, Serge Egelman, Christian Kreibich, et al. " is our children's apps learning?" automatically detecting coppa violations. In *Workshop on Technology and Consumer Protection (ConPro 2017), in conjunction with the 38th IEEE Symposium on Security and Privacy (IEEE S&P 2017)*, 2017.

[29] Ilaria Liccardi, Monica Bulger, Hal Abelson, Daniel J Weitzner, and Wendy Mackay. Can apps play by the coppa rules? In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pages 1–9. IEEE, 2014.

[30] Bing Hu, Bin Liu, Neil Zhenqiang Gong, Deguang Kong, and Hongxia Jin. Protecting your children from inappropriate content in mobile apps: An automatic maturity rating framework. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1111–1120, 2015.

[31] Ying Chen, Heng Xu, Yilu Zhou, and Sencun Zhu. Is this app safe for children? A comparison study of maturity ratings on android and ios applications. In *Proceedings of the 22nd international conference on World Wide Web*, pages 201–212, 2013.

[32] Sen Chen, Lingling Fan, Guozhu Meng, Ting Su, Minhui Xue, Yinxing Xue, Yang Liu, and Lihua Xu. An empirical assessment of security risks of global Android banking apps. In *Proceedings of 2020 IEEE/ACM 42nd*

International Conference on Software Engineering (ICSE), pages 1310–1322, 2020.

[33] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. Iccta: Detecting inter-component privacy leaks in android apps. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 280–291. IEEE, 2015.

[34] Minhui Xue, Cameron Ballard, Kelvin Liu, Carson Nemelka, Yanqiu Wu, Keith Ross, and Haifeng Qian. You can yak but you can't hide: Localizing anonymous social network users. In *Proceedings of the 2016 Internet Measurement Conference*, pages 25–31, 2016.

[35] Sen Chen, Ting Su, Lingling Fan, Guozhu Meng, Minhui Xue, Yang Liu, and Lihua 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*, pages 797–802, 2018.

[36] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé Bissyandé, and Jacques Klein. MadDroid: Characterizing and detecting devious Ad contents for Android apps. In *Proceedings of The Web Conference 2020*, pages 1715–1726, 2020.

[37] Zhushou Tang, Ke Tang, Minhui Xue, Yuan Tian, Sen Chen, Muhammad Ikram, Tielei Wang, and Haojin Zhu. iOS, your OS, everybody's OS: Vetting and analyzing network services of iOS applications. In *29th USENIX Security Symposium*, pages 2415–2432, 2020.

[38] Wei Wang, Ruoxi Sun, Minhui Xue, and Damith C Ranasinghe. An automated assessment of Android clipboards. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020.

[39] Ruoxi Sun, Wei Wang, Minhui Xue, Gareth Tyson, Seyit Camtepe, and Damith C Ranasinghe. An empirical assessment of global covid-19 contact tracing applications. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1085–1097. IEEE, 2021.

[40] Mobile-security-framework-mobsf. https://github.com/MobSF/Mobile-Security-Framework-MobSF.

[41] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[42] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge 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)*, pages 603–620, 2019.

[43] Catherine Han, Irwin Reyes, Amit Elazari Bar On, Joel Reardon, Álvaro Feal, Serge Egelman, Narseo Vallina-Rodriguez, et al. Do you get what you pay for? comparing the privacy behaviors of free vs. paid apps. In *Workshop on Technology and Consumer Protection (ConPro 2019), in conjunction with the 39th IEEE Symposium on Security and Privacy, 23 May 2019, San Francisco, CA, USA.*, 2019.

[44] Anh Le, Janus Varmarken, Simon Langhoff, Anastasia Shuba, Minas Gjoka, and Athina Markopoulou. Antmonitor: A system for monitoring from mobile devices. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*, pages 15–20, 2015.

[45] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, Phillipa Gill, et al. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS 2018)*, 2018.

[46] Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. Nomoads: Effective and efficient cross-app mobile ad-blocking (the andreas pfitzmann best student paper award). In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, volume 2018, 2018.

[47] Max Van Kleek, Ilaria Liccardi, Reuben Binns, Jun Zhao, Daniel J Weitzner, and Nigel Shadbolt. Better the devil you know: Exposing the data sharing practices of smartphone apps. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5208–5220, 2017.

[48] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. Release planning of mobile apps based on user reviews. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 14–24. IEEE, 2016.

[49] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. Ardoc: App reviews development oriented classifier. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, pages 1023–1027, 2016.

[50] Ehsan Noei, Feng Zhang, and Ying Zou. Too many user-reviews! what should app developers look at first? *IEEE Transactions on Software Engineering*, 47(2):367–378, 2019.

[51] Phong Minh Vu, Hung Viet Pham, Tam The Nguyen, and Tung Thanh Nguyen. Phrase-based extraction of user opinions in mobile app reviews. In *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*, pages 726–731, 2016.

[52] Washington Luiz, Felipe Viegas, Rafael Alencar, Fernando Mourão, Thiago Salles, Dárlinton Carvalho, Marcos Andre Gonçalves, and Leonardo Rocha. A feature-oriented sentiment rating for mobile app reviews. In *Proceedings of the 2018 World Wide Web Conference*, pages 1909–1918, 2018.

[53] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Boshen Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th international conference on software engineering*, pages 767–778, 2014.

[54] Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. Short text, large effect: Measuring the impact of user reviews on android app security & privacy. In *2019 IEEE symposium on Security and Privacy (SP)*, pages 555–569. IEEE, 2019.

[55] Hao Chen, Daojing He, Sencun Zhu, and Jingshun Yang. Toward detecting collusive ranking manipulation attackers in mobile app markets. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 58–70, 2017.

[56] Zhengyang Qu, Vaibhav Rastogi, Xinyi Zhang, Yan Chen, Tiantian Zhu, and Zhong Chen. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1354–1365, 2014.

[57] Le Yu, Xiapu Luo, Xule Liu, and Tao Zhang. Can we trust the privacy policies of android apps? In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 538–549. IEEE, 2016.

[58] Shuang Liu, Baiyang Zhao, Renjie Guo, Guozhu Meng, Fan Zhang, and Meishan Zhang. Have you been properly notified? automatic compliance analysis of privacy policy text with gdpr article 13. In *Proceedings of the Web Conference 2021*, pages 2154–2164, 2021.

[59] Andro guard. https://github.com/androguard/androguard, 2012.

[60] Álvaro Feal, Paolo Calciati, Narseo Vallina-Rodriguez, Carmela Troncoso, Alessandra Gorla, et al. Angel or devil? a privacy study of mobile parental control apps. *Proceedings of Privacy Enhancing Technologies (PoPETS)*, 2020, 2020.

[61] Ui/application exerciser monkey. https://developer.android.com/studio/test/monkey.

[62] Catherine Han, Irwin Reyes, Álvaro Feal, Joel Reardon, Primal Wijesekera, Narseo Vallina-Rodriguez, Amit Elazar, Kenneth A Bamberger, and Serge Egelman. The price is (not) right: Comparing privacy in free and paid apps. *Proceedings on Privacy Enhancing Technologies*, 2020(3), 2020.

[63] Julien Gamba, Mohammed Rashed, Abbas Razaghpanah, Juan Tapiador, and Narseo Vallina-Rodriguez. An analysis of pre-installed android software. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1039–1055. IEEE, 2020.

[64] Yangyu Hu, Haoyu Wang, Tiantong Ji, Xusheng Xiao, Xiapu Luo, Peng Gao, and Yao Guo. CHAMP: Characterizing undesired app behaviors from user comments based on market policies. In *Proceedings of 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 933–945, 2021.

[65] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[66] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[67] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[68] Aastha Singh. Evolving with BERT: Introduction to RoBERTa. https://medium.com/analytics-vidhya/evolving-with-bert-introduction-to-roberta-5174ec0e7c82, 2021.

[69] Preksha Nema, Pauline Anthonysamy, Nina Taft, and Sai Teja Peddinti. Analyzing user perspectives on mobile app privacy at scale. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022.

[70] Sean Bleier. NLTK's list of English stopwords. https://gist.github.com/sebleier/554280, 2009.

[71] YouTube. Youtube kids. com.google.android.apps.youtube.kids, 2022.

Chapter 5

Enhancing VR Users' Privacy with Differential Privacy

The work contained in this chapter is prepared to be submitted to a conference.

Ruoxi Sun, Hanwen Wang, Minhui Xue, and Tim Chen. PPVR: A privacy-preserving approach for 3D body motion data in VR. Will be submitted to submit to *the 31st IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR, CORE A*)*, 2024.

Statement of Authorship

Title of Paper	PPVR: A privacy-preserving approach for motion and spatial data in VR
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☑ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Ruoxi Sun, Hanwen Wang, Minhui Xue, and Tim Chen. PPVR: A privacy-preserving approach for motion and spatial data in VR. Plan to submit to the 31st IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR) 2024.

Principal Author

Name of Principal Author (Candidate)	Ruoxi Sun		
Contribution to the Paper	Built the conceptual idea; conducted experiments; wrote and refined the manuscript.		
Overall percentage (%)	80%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	10 Jun 2023

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Hanwen Wang		
Contribution to the Paper	Implemented the tool; conducted experiments		
Signature		Date	

Name of Co-Author	Minhui Xue		
Contribution to the Paper	Built the conceptual idea; supervised the development of this work.		
Signature		Date	

Name of Co-Author	Tim Chen		
Contribution to the Paper	Built the conceptual idea; refined the manuscript; supervised the development of this work.		
Signature		Date	15/6/2023

Please cut and paste additional co-author panels here as required.

The increasing popularity of immersive environments creates a pressing challenge of reconciling service providers' desire to collect user behaviour data with users' privacy concerns. Striking a balance between data collection and privacy protection becomes crucial in this context. In this study, we investigate the use of differential privacy (DP) algorithms in VR applications to enable statistical analysis of 3D spatial motion data while protecting against re-identification attacks. Two datasets were used to conduct the experiments. First is an indoor activities data with simulated agents ($N = 7$). The second is a public 3D motion capture dataset of users playing VR sport games ($N = 16$). We assessed the efficacy of DP on both the original 3D spatial data and its cumulative heat map representation. Experimental results reveal that our approach effectively preserves data utility (with threshold $RSE \leq 1$) while reducing the accuracy of the re-identification attack model from 93.89% to 48.03% (through window-slicing) and from 96.53% to 55.37% (through heat map conversion). This study underscores the utility of the DP algorithm in the context of 3D body motion data and highlights its potential for widespread adoption in diverse VR applications.

5.1 Introduction

The global VR market was valued at 28 billion US dollars in 2022 and is projected to grow at a compound annual growth rate (CAGR) of 13% from 2023 to 2030 [1]. Leading tech companies invest heavily in VR, with Meta introducing its cutting-edge VR headset, the Meta Quest Pro [2], and reports suggesting that Apple has a team of 3,000 employees dedicated to their upcoming VR headset [3]. Consumer interest in applications such as gaming, fitness, and social experiences has steadily grown over the past decade. Notably, during the COVID-19 pandemic [4], VR has played a pivotal role in driving innovation in fields like art, healthcare, and education.

As VR devices become more widely adopted, researchers express increasing concerns about privacy issues in many VR applications [5, 6, 7]. On one hand, VR technology is rapidly evolving, and VR device companies understandably seek to continuously monitor user behaviour data to improve user experiences and the performance of VR systems [8]. On the other hand, it can be argued that the privacy risks associated with VR applications are potentially more severe than those of mobile applications. This is due to the extensive collection of users' personal information by various input/output devices and sensors [9]. For example,

the seemly standard data from VR headsets and controllers can inadvertently disclose users' biometric details, such as height and body shape [10]. Furthermore, the dynamic data captured by motion sensors can even unveil users' preferences and opinions towards virtual content, as evidenced by the analysis of eye-tracking data [11]. Moreover, recent studies have revealed that users can be identified with an accuracy exceeding 90% when compared to a database of over 50,000 individuals, based on just 100 seconds of motion recorded during VR gaming sessions [12]. Striking a delicate balance between data collection and privacy preservation is imperative to enable the ongoing enhancement of VR technologies while respecting users' privacy and security.

Differential privacy (DP) is a mathematical framework that provides a strong guarantee of privacy by allowing data to be analysed without revealing sensitive information about any individual in the dataset [13]. DP has been applied across diverse domains, such as concealing demographic census data [14] and safeguarding the privacy of Uber drivers and riders during analysis [15]. However, there is limited research on the use of DP in VR applications. Nair *et al.* [16] proposed the pioneering concept of "VR Incognito Mode", which uses DP to obscure sensitive user data attributes, such as user height, wingspan, or room size. However, it is unclear how to apply DP on 3D body motion data, such as head or hand movement, which has been shown vulnerable to re-identification attacks [12, 10].

To address this knowledge gap, we present two simple yet effective approaches to apply DP mechanism on 3D body motion data from VR applications. Our goal is to safeguard user privacy by preventing re-identification attack while retaining valuable statistical information, such as the average head and hand movements of users. This data is vital for enhancing the gaming experience and gaining insights into user preferences in virtual social or shopping spaces. Our first approach involves the direct application of DP onto the 3D body motion data while our second approach transforms the spatial data into 2D heat maps before applying DP. We evaluated the effectiveness of our privacy protection method using two VR datasets. The first dataset emulates indoor activities collected by the VirtualHome simulator [17], while the second dataset captures user motion in a VR sports game [18]. The experimental findings demonstrate that our method successfully maintains data utility (with an RMSE threshold set to 1) while diminishing the accuracy of the re-identification attack model from

93.89% to 48.03% (via window-slicing) and from 96.53% to 55.37% (through heat map conversion), indicating that heat map conversion is more effective.

Contribution. We summarise our main contributions as follows.

- We conduct experiments to apply Differential Privacy (DP) to 3D body motion data, aiming to protect user privacy while preserving data utility.
- We suggest a novel approach of transforming 3D body motion data into heat maps prior to applying DP, which enhances the efficacy of DP.
- We assess our method using both synthetic and real-world body motion datasets, specifically focusing on their resilience against re-identification attacks.

Our findings indicate that, despite the wealth of data being collected, user privacy can be preserved while still permitting data analysis for various purposes within an immersive environment. Given the rapid evolution of VR and sensor technologies, and the increasing collection of user behavior data for analysis, there is a pressing need for more research dedicated to privacy-preserving approaches, such as DP.

5.2 Related Work

Recent studies have revealed the ease with which attackers can identify [19, 20, 21] and create profiles of VR users [22, 23] using just a few minutes of data streaming. Furthermore, these studies have highlighted that the extent and magnitude of data collection in VR surpass the capabilities of current internet platforms. The immersive nature of VR contributes to these vulnerabilities, as it can make users more susceptible to self-disclosure[24] and social engineering [25].

In contrast to current Internet platforms, where users have options like Tor, VPNs, proxies, and incognito mode to protect against user tracking and profiling, there is a lack of equivalent and robust defense mechanisms to address the unique threats in VR. Existing literature provides a fragmented collection of privacy defenses that are still in the proof-of-concept stage, with limited application in commercial-grade solutions. Moreover, industry practices in the VR domain are not reassuring. Vulnerabilities in VR devices have been identified, some developers disregard their own privacy policies, and updates tend to prioritize increased data collection [26].

In our threat model, privacy breaches occur when attackers gather and infer sufficient information to consistently identify and extensively profile a user across multiple usage sessions in VR applications (referred to as tracking). Attackers achieve identification (*i*) by distinguishing the user from others in a unique manner and (*ii*) profiling users by associating unwarranted information with their characteristics, such as demographics, preferences, and browsing history [22].

For example, two primary entities pose threats to user privacy in this context: the developers of client-side applications running on VR devices (referred to as Application Adversaries [8]) and content creators (known as Content Adversaries [25]). Content adversaries have the ability to create immersive experiences that incorporate misleading, manipulative, and deceptive content. On the other hand, application adversaries can access input data through system APIs and manipulate the rendered frames and signals sent to VR devices, as well as the information streamed to external servers. While one server-side, server adversaries may have control over the external server. As a result, they possess the ability to manipulate and process the networked data they receive before streaming it to other users' devices in any desired manner.

We argue that without a well-designed privacy protection technique, users' privacy is threatened by both client-side and server-side adversaries. In other words, these adversaries have the capability to access users' private data and easily re-identify them.

5.3 Use Cases

Before diving into the methodology, we would like to introduce two use cases of our proposed privacy protection approach.

Case 1: VR Gaming

In VR gaming, the goal of privacy protection is to secure players' personal and sensitive information while providing a safe and enjoyable gaming experience. However, developers, such as gaming companies, often collect user data to analyse gameplay and enhance the gaming experience.

Consider a VR archery game where the objective is to hit the bull's eye. The bow is attached to the user's left hand, while the right hand draws the string, and arrows are loaded by moving the bow towards a virtual quiver. Developers

may enhance the gaming experience by analysing players' behaviour using body movement data collected from sensors on VR devices. This data can encompass eye tracking, hand controller positioning, and derived features like user height, reaction speed, and arm stability. Without adequate privacy protection, it could be possible to re-identify a specific user from this data. However, it's important to note that developers may not necessarily need precise body movement data from each user, as they are typically more interested in the average behaviour of all users.

Case 2: Virtual shopping mall

In another scenario, a virtual shopping mall provides an immersive and interactive environment, allowing shoppers to browse and purchase products from the comfort of their homes. This virtual environment closely mirrors a real shopping experience. Shoppers utilise VR headsets and controllers to navigate the mall, interact with objects, and even try on virtual clothing and accessories.

In this context, the company is interested in collecting spatial interaction data, such as movements within the virtual shopping mall or interactions with virtual shelves, to identify popular areas and modify the virtual environment to enhance user engagement. However, akin to the first use case, while this spatial interaction data is crucial for improving user experience, it could potentially be used to re-identify users, thus raising privacy concerns.

5.4 Methodology

In this section, we first introduce the background of differential privacy, followed by our method of applying differential privacy onto 3D body motion data through (*i*) window slicing and (*ii*) converting the data into heat maps. Our approach aims to safeguard data privacy while preserving its utility.

5.4.1 Background of Differential Privacy

Differential privacy [27, 28] is a formally recognized concept of privacy that can be mathematically validated for data releases. Unlike k-anonymity, which is a property attributed to data, DP is a property attributed to algorithms. This implies that we can prove an algorithm's compliance with DP requirements. To assert that a dataset adheres to DP, we need to show that the algorithm used to generate it satisfies DP's principles..

Plain differential privacy. Formally, a mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (plain) differential privacy if for all neighbouring datasets $d, d' \in \mathcal{D}$ and for all possible outputs $S \subseteq \mathcal{R}$ it have

$$Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(d') \in S]. \tag{5.1}$$

Specifically, two datasets $d, d' \in \mathcal{D}$ are considered neighbours if they only vary in the information of a single individual. It's important to note that $\mathcal{M}$ is usually a randomised function, producing multiple possible outputs for the same input. As a result, the probability distribution describing its outputs is not a singular point distribution.

The crucial implication of this definition is that the output of $\mathcal{M}$ will remain largely unchanged, regardless of the inclusion or exclusion of any specific individual's data. In other words, the level of randomness incorporated into $\mathcal{M}$ should be sufficient to prevent an observed output from revealing whether the input was d or d'. For instance, if an individual data is present in d but not in d' an adversary would be unable to determine which of the two was the input to $\mathcal{M}$. As a result, the adversary would have no means of determining whether an individual's data was included in the input data, let alone obtaining any detailed information about that particular data.

The privacy parameter or privacy budget in the definition is denoted as ϵ. It serves as a control to adjust the "degree of privacy" provided by the mechanism. Smaller values of ϵ that should produce highly similar outputs for similar inputs, thereby offering stronger privacy protection. On the other hand, larger values of ϵ allow for greater variability in the outputs, resulting in reduced privacy.

Laplace mechanism. The most direct method to achieve DP is by incorporating random noise into the response. The primary challenge is to add enough noise to meet DP's requirements, while ensuring the answer remains meaningful and not excessively distorted. To streamline this process, the DP field has developed fundamental mechanisms that precisely outline the type and level of noise to be used.

Laplace mechanism [28] is a commonly used approach. Specifically, according to the Laplace mechanism, the following definition of $\mathcal{M}$ satisfies ϵ-differential privacy.

$$\mathcal{M}(d) = m(d) + Lap(\frac{s}{\epsilon}) \tag{5.2}$$

where s is the sensitivity of m which represents the amount of m's output changes when its input changes by 1 (recall the neighbouring datasets d and d'), and Lap denotes sampling from the Laplace distribution with centre 0 and scale $\frac{s}{\epsilon}$.

Approximate differential privacy. In this study, we employ the notion of approximate differential privacy, also called (ϵ, δ)-differential privacy, which is commonly used in machine learning and defined as below.

A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with a domain $\mathcal{D}$ and a range $\mathcal{R}$ achieves (ϵ, δ)-differential privacy if, for any pair of neighbouring inputs d and $d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$, the following condition is satisfied:

$$Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(d') \in S] + \delta \tag{5.3}$$

where the privacy parameter δ represents the "failure probability" associated with the definition. With a probability of $1-\delta$, the privacy guarantee provided is equivalent to pure differential privacy, while with a probability of δ, no guarantee is provided. In other words, with a probability of $1 - \delta$, we have the inequality $\frac{Pr[\mathcal{M}(d) \in S]}{Pr[\mathcal{M}(d')]} \leq e^{\epsilon}$. Due to this, it is typically required for δ to be very small, usually less than or equal to $\frac{1}{n^2}$, where n represents the size of the dataset.

5.4.2 Preparing Body Motion Data for DP

Outliers Dropping. Considering that each sample in the datasets represents a repetition of a motion, which can have varying lengths and some samples may exhibit significant deviations, it becomes challenging to unify the shape of the data without removing such outliers. Moreover, allowing these outliers to remain could enable a machine learning model to learn and classify the execution length of a motion, potentially leading to user identification and privacy breaches. Furthermore, similar to the outliers in the temporal domain mentioned earlier, outliers in the spatial domain should also be taken into account.

Therefore, prior to inputting the data into a machine learning model, we first eliminate outliers using the standard deviation of the dataset. Specifically, we calculate the upper and lower boundaries of the dataset by adding or subtracting three standard deviations from the mean of the values. Any sample that falls

outside these boundaries is marked as an outlier and subsequently removed from the dataset.

It is worth noting that in practical applications, there may be alternative methods for handling outliers, such as pre-padding the data with zeros to achieve temporal invariance. However, the selection of an optimised outlier approach may depend on the specific application scenario and is beyond the scope of our research.

Sequence Alignment. Dynamic time warping (DTW) is an algorithm for measuring similarity between two temporal sequences, which is widely used in time series analysis [29]. DTW is good at synchronizing two sequences by optimally aligning within them through appropriate translations, expansions and contractions, ultimately minimizing the distance between aligned series. DTW has been applied to temporal sequences of video, audio, and graphics data — indeed, any data that can be turned into a one-dimensional sequence can be analyzed with DTW.

The principle is that a cos-matrix is created which is record the shortest distance between longer sequence A and shorter sequence B. Then, the returning result is a sequence B' which has the same length of the sequence A and some repeated values of sequence B. The sequence B' is the final results from sequences trajectories. For motion sequences, DTW preserves the distinct features of longer sequences when compared to fixed segmentation method. Also, in contrast to average fill-up method, Dynamic Time Warping ensures that the unique data features of shorter sequences are retained.

Applying DTW for aligning each motion sequence becomes imperative to guarantee the successful computation of the average curve for motion sequences. Otherwise, the trailing end of long sequences will be not computed the average values. However, it is essential to note that due to the computationally intensive nature, the application in data pre-processing for large datasets could be limited.

In practice, we use the Dynamic Time Warping (DTW) method to align sequences, thereby preserving their distinct features. Specifically, we employ a function from the 'DTAIdistance' library [30] to find values corresponding to the time length of the longest sequence and implement dynamic alignment using our user-defined function.

Window Slicing. Window slicing is a prevalent technique for training machine learning models on time series spatial data, operating on a rolling window with a predefined size, denoted as n, and a step size, denoted as m. This method iterates through each user's data recording, generating new samples at each step.

We use a window size of $N = 10$ and the step size of $M = 1$ for our training and validation set for the evaluation. To ensure consistency and comparability, the same pre-processing steps and sliding window approach are applied to the validation data. This allows for fair evaluation and assessment of the model's performance.

From an adversary's perspective, the application of the window slicing technique involves a majority vote among the labels of all sliced windows. This approach is frequently used as it enables the attack model to identify temporal patterns and dependencies within the data, and potentially minimise noise or inaccuracies in the prediction.

Heat map Creation. Alongside window slicing, we explore transforming the 3D body motion data into heat maps. This approach is frequently employed in the visualisation of spatial data and consequently helps to preserve the utility of the data.

Our proposed method comprises a series of steps. Initially, we calculate the range of data values on each axis, such as the X axis, represented by [X_{max}, X_{min}]. Subsequently, we define the resolution of the heat map as r, which determines the number of data points along each axis. By having r data points on each axis, the total number of data points in a heat map becomes $r \times r$. For each sample $x \in X$, we determine its position within the heat map using Equation 5.4. The "ceil" function, denoted as "$\lceil\ \rceil$", is applied to round each sample's position to the nearest integer, ensuring its placement within the heat map grid.

$$x_h = \lceil \frac{x - X_{min}}{X_{max} - X_{min}} * r \rceil \tag{5.4}$$

The value of each data point in the heat map is the number of samples that fall into the corresponding grid position. This provides a measure of the density or frequency of occurrences at that specific location. Lastly, to ensure consistency and comparability, we normalize the data values in the heat map to fall within

the range of $[0, 1]$. This step facilitates a standardized representation of the data across various heat maps.

Heat map utility: one disadvantage of converting body motion data into heat maps is the loss of temporal information, limiting the analysis to spatial movement tracing and frequency information alone. However, as described in the use cases earlier, the omission of temporal data when sharing with vendors or third-parties have little impact the data utility. This is particularly true in scenarios such as virtual homes or virtual shopping, where the primary focus for application vendors may be recording movement traces and identifying the most frequently visited places by users. Similarly, in VR scenarios involving natural interactions like playing virtual archery, vendors may only need to collect users' average movement traces to analyse user behaviours and enhance the user experience of VR applications. Hence, for these common use cases, heat maps provide sufficient information to vendors or third-parties.

5.4.3 Applying DP on Body Motion Data with Utility Preserved

We then apply different privacy mechanism onto both the original motion data and the converted heat maps. To compare, we also conduct experiments applying DP on data with window slicing.

Differential privacy tool. In this research, we leverage the IBM differential privacy library[1], Diffprivlib [31]. Diffprivlib is a comprehensive library specifically designed to address the challenges of differential privacy and machine learning. Its primary objective is to provide researchers and practitioners with a versatile platform for conducting experiments, simulations, and implementing differentially private models. By utilising a unified codebase and a collection of fundamental building blocks, Diffprivlib facilitates seamless exploration and deployment of differential privacy techniques across various domains and applications. This library serves as a valuable resource for researchers seeking to incorporate differential privacy into their studies and implementations. Particularly, we utilise Diffprivlib's implementation of Laplace mechanisms, which were initially proposed by Dwork *et al.* [28]. This implementation also includes support for (relaxed) (ϵ, δ)-differential privacy [32]. We leverage these mechanisms

[1]The library is available at https://github.com/IBM/differential-privacy-library

to apply differential privacy to our datasets or heat maps, ensuring privacy protection while preserving valuable information.

Utility preserving. Known as the privacy parameter or privacy budget, ϵ represents the maximum allowable distance between a query performed on two neighbouring databases (d and d'), where $d,\ d' \in \mathcal{D}$ differ by only 1 change in data (*i.e.*, the addition or removal of a single entry, as per Equation 5.1). In other words, ϵ serves as a measure of privacy loss, ensuring that for any pair of adjacent databases $d,\ d'$ and all possible outputs $\mathcal{M}$, an adversary cannot differentiate between them based on observing the output.

Particularly, when ϵ has smaller values, the outputs generated for similar inputs need to be very similar, thereby providing higher levels of privacy. Conversely, larger values of ϵ allow for greater dissimilarity in the outputs, resulting in reduced privacy. Essentially, a smaller ϵ implies that more noise must be added to adequately protect the dataset's privacy.

While our primary objective is to utilize the Laplace mechanism to introduce noise and prevent the re-identification of individuals in the dataset, we also need to consider the balance between privacy protection and data utility as described in Section 5.3. To achieve this, we introduce a data utility threshold to control the level of noise added through the application of differential privacy. This threshold helps us avoid excessive noise that could potentially compromise the usefulness of the data while still ensuring an acceptable level of privacy.

Specifically, we can establish the data utility threshold empirically using quantitative or qualitative metrics, or a combination of both. For instance, in the case of a heat map, a qualitative threshold could be defined as "the heat map, after applying differential privacy, should still provide clear visibility of the users' movement traces". Based on this qualitative threshold, we can further determine a quantitative threshold, such as "the Relative Squared Error (RSE) between the heat maps before and after applying differential privacy should be lower than t". In our study, we identify the optimal privacy budgets as the largest ϵ value that produce the output below an acceptable utility threshold, based on the ϵ-RSE chart. The pilot experiment allows us to empirically assess the impact of different privacy budgets on the data utility. By applying varying levels of noise and measuring the resulting data utility, we can identify the optimal privacy budget that strikes the right balance between privacy protection and data usability.

By setting such utility thresholds, we can effectively limit the amount of noise introduced by the differential privacy mechanism. These thresholds serve as objective measures to ensure that the data utility is maintained while adequately protecting privacy. Through a careful balance of quantitative and qualitative assessments, we can strike the right balance between privacy preservation and data utility in our analysis.

5.5 Experiment Setup

In this section, we introduce the datasets, user identification models, and the evaluation metrics used in our study.

5.5.1 Experiment Environment

Our experimental environment consists of a PC workstation equipped with 64GB of RAM, an AMD Ryzen 3750X 8-core CPU, and Linux Mint 20.1 Cinnamon operating system. Additionally, we have allocated a 256GB swap partition to enhance system performance. The experiments are conducted using Python 3.9.9.

5.5.2 Datasets

VirtualHome dataset. A technique for simulating household activities using programs, employing sequences of atomic actions and interactions as a higher-level representation of complex tasks, was introduced by Puig *et al.* [17]. The proposed simulator, VirtualHome[2], empowers users to generate a comprehensive dataset of activity videos with detailed ground-truth information, facilitating the training and evaluation of video understanding models. An example of an agent is watching TV which is generated by the simulator and demonstrated in Figure 5.1(a). In our study, we use some interaction sequences as the simulator input and select all existed agents as the interacted subjects. Then, we utilise the simulator to generating interacted videos and 3D spatial interacted data of virtual agents performing kinds of tasks in household scenarios (*i.e.*,7 agents acting 852 tasks) and collect the data as our experimental dataset. For example, in one scenario, multiple agents' activities can be generated using the following description: "Go watch TV on the couch. Turn the TV off and grab the coffee pot. Put the coffee pot on the table and go turn the light on" [17]. We collect

[2]The simulator is available at https://github.com/xavierpuigf/virtualhome_unity.

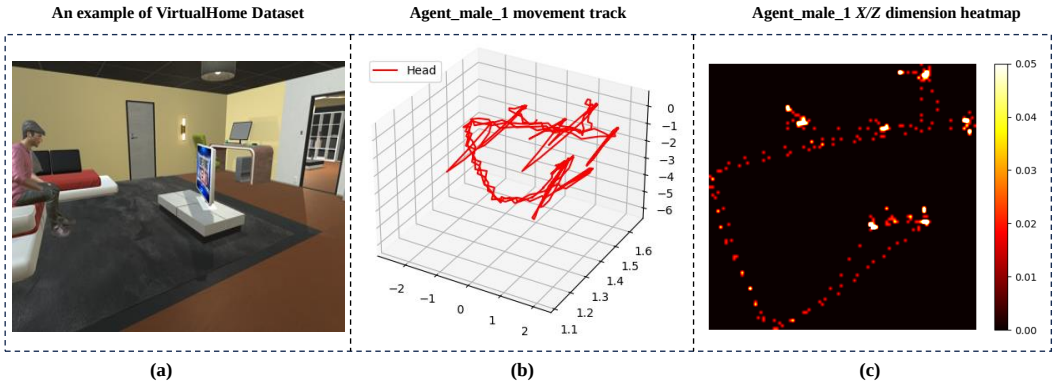


Figure 5.1. A visualisation of one sample in VirtualHome dataset [18]: (a) the virtual environment, (b) an illustration of movement track, and (c) the heat map conversion.

the motion data of the agents from their starting points to the TV, then to the table, and finally to the light switch. Figure 5.1(b) illustrates that the movement track of agent male 1 in a 3D space. Since the head is the cent er of the body and has more actions than other body parts, the position of data collection is the head of each agent. After that, the results of the heat map conversion method is utilised on interacted motion data, which is shown in Figure 5.1(c). In this scenario, we define the data utility as 'the average motion trace of multiple users', which could be collected by the app vendor or third-parties for further data analysis.

Body Movement dataset. Liebers *et al.* [18] conducted a laboratory study involving 16 participants to investigate the accuracy of user identification. The researchers simulated two task-driven scenarios using common VR games, *i.e.*, Bowling and Archery. In these VR games, users engage in natural interactions with the game, based on their spatial movement. An illustration of the Archery game to generate Body Movement dataset is demonstrated in Figure 5.2(a). Spatial motion data was collected using a consumer-grade head-mounted display (HMD) and hand-held controllers.

Specifically, the data recording includes Euler Angles, timestamps, and motion stages as extended data features. These features are set into distinct experimental groups to investigate their impact on identification accuracy. Furthermore, researchers have introduced a novel normalisation technique which is aimed at adjusting the height and arm length ratios between users and virtual players.

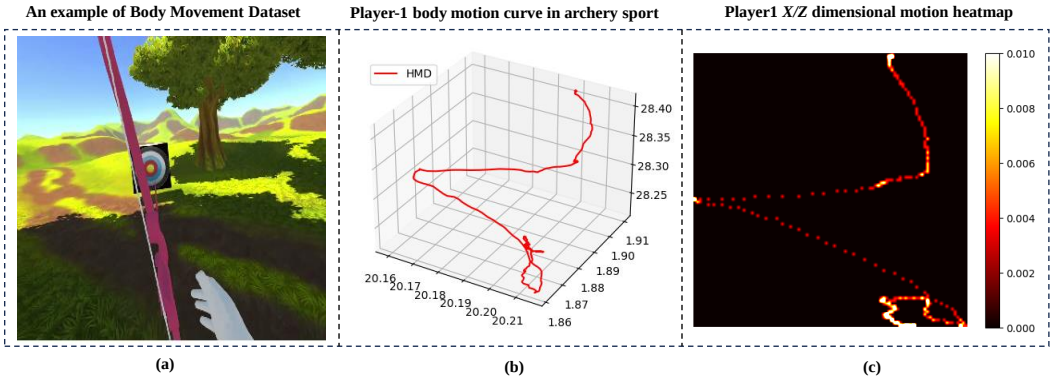


Figure 5.2. A visualisation of one sample in Body Movement dataset [18]: (a) the virtual environment, (b) an illustration of movement track, and (c) the heat map conversion.

This adjustment is also a part of the experimental setup. In this study, researchers argue that implementing the proposed height-normalisation approach on spatial motion data only would generally increase the identification rate.

In our research, we adopt their dataset[3] and particularly extract the users' motion data after body normalization applied. Figure 5.2(b) shows an example of body movement data in a 3D space; also, the heat map converted results are shown in Figure 5.2 (c).

5.5.3 User Identification Models

As described in Section 5.4, we explore two approaches for applying differential privacy to 3D body motion data. In concrete, we process the data with two different techniques (*i.e.*,window-slicing and converting into heat map). Then we conduct user identification attacks using two distinct models based on the features of the processed data. Specifically, following the experimental setup in datasets and recent studies [18, 17], we apply a Recurrent Neural Network (RNN) models on window-slicing data and Convolutional Neural Network (CNN) models on heat map data, respectively.

LSTM on window-slicing data. The LSTM model consists of three Long Short-Term Memory layers, and each layer has one hundred units. The activation function is selected as the default 'sigmoid'. Other hyper-parameters are

[3]The dataset is available at https://www.hci.wiwi.uni-due.de/en/publikationen/understanding-user-identification-in-virtual-reality-through-behavioral-biometrics-and-the-effect-of-body-normalization/.

set as: 200 epochs, Adam optimiser, and 1e-4 learning rate. Additionally, a majority voting is applied to determine the prediction label a sample. Specifically, according to the labels predicted on all sub-samples (a sample could be sliced into several sub-samples during window-slicing), the most frequent label is assigned as the final predicted label. Due to the specificity of the dataset, which includes repetitions over two days, we divided the data into two parts: the motion data from the first day is set as the training set, while the data from the second day is set as the testing set. The advantage of splitting the dataset by date is that it helps avoid high repeatability between each sub-sample after window-slicing pre-processing.

CNN on heat map data. We establish a CNN network with two convolution layers and three full connection layer. In each convolution layer, there is a pooling layer. In the first two full connection layers, we add a drop layer with 0.5 dropping rate. For each layer, we use 'ReLU' as the activation function. The output of the model is a n-dimension vector, where n is the number of users in the dataset. The model is trained on 80% of the samples and tested on the remaining 20% samples.

5.5.4 Evaluation Metrics

In our study, we evaluate whether a privacy-enhancing approach is capable of safeguarding users' privacy while maintaining sufficient data utility. We use the following two metrics in experiments.

Relative squared error (RSE). To measure and control the error introduced by differential privacy, we utilise relative squared error (RSE) to calculate the average squared difference between the original data and the DP-enhanced data. The output value of RSE is expressed in terms of ratio. Specifically, as formalised in Equation 5.5, RSE calculates the relative squared error, which normalises the total squared error (*i.e.*,MSE) and normalises it by the square of the difference between the actual and the mean of the data.

$$RSE = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x}_i)^2}, \tag{5.5}$$

where x_i is a data point from spatial data or a heat map, $\bar{x}_i$ is the mean of all data points, and $\hat{x}_i$ is the corresponding data point after applying DP. A RSE value can range from 0 to 1. A good model should have a value close to 0 while a model with a value greater than 1 is not reasonable.

The reason we use RSE as the error matric is because it is less influenced by the total data volume, compared to Mean Squared Error (MSE). Specifically, for window-slicing data, we directly compute two motion average curves between the temporal dimension and each dimension of feature, both before and after applying DP (*i.e.*, calculating the average curve by x-dimension data of the headset and time stamp); for heat map data, we calculate RSE between two 100*100 images.

Specifically, we use RSE as a quantitative threshold of data utility, to ensure that the data is still usable after applying DP onto it. The threshold of RSE will further determine the privacy budget ϵ which controls how much noise would be added to the data. For example, we can select the Relative Squared Error (RSE) threshold according to the specific utility scenario. A default threshold could be set at 1, and the smallest value of ϵ that can meet this threshold will be chosen for different privacy settings. For convenience in practice, we round up the value of ϵ to an integer. For example, if a chosen value of ϵ is 6.4, it will be rounded up to $\epsilon = 7$.

Identification accuracy. To evaluate the data privacy performance against user identification attacks, we utilise identification accuracy as the metric. As illustrated in Equation 5.6, the attack accuracy is defined as the number of user classifications the attack model correctly predicts divided by the total number of predictions made. Specifically, a lower identification accuracy indicates higher robustness of data privacy. If the accuracy of a user identification attack falls below that of random guessing (*i.e.*, below 50% in a binary classification), we classify the attack as failed and consider the data privacy to be robust.

$$Identification\ Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{5.6}$$

5.6 Experimental Results

In this section, we present the experimental results of the data utility evaluation and the performance of our approach in protecting privacy against user identification attacks.

5.6.1 Data Utility Evaluation

We first evaluate the data utility through both qualitative and quantitative manners.

Qualitative data utility. As previously described in Section 5.4 and 5.5, we first evaluate the data utility through a qualitative method. This involves ensuring that the movement traces are still clear and recognisable to the human eyes after adding differential privacy noise. We demonstrate the data utility by plotting the original spatial data in x/z space, along with its corresponding heat map.

Specifically, to illustrate how we compare the data utility in different ϵ settings, we randomly select two samples from each dataset and plot their spatial data and corresponding heat maps, as shown in (Figure 5.3(a) and Figure 5.4(a)). Then compare with data with differential privacy applied. This is done for ϵ settings of 1, 3, and 10, as shown in Figure 5.3(b-d) and Figure 5.4(b-d). From the experimental results, it is evident that higher privacy budgets (*e.g.* when $\epsilon = 3$ and 10) introduce less noise into the data, which is expected as higher privacy budgets are applied.

Here we note that, in practice, the purpose of conducting qualitative utility evaluation is to obtain an approximate proper range of ϵ, to saving computational cost in the quantitative utility control, since a proper utility threshold may lead to selecting a very large or small ϵ in specific cases. For example, an ϵ value in the range of 3 to 10 provides sufficient data utility in VirtualHome dataset, while an ϵ value less than 3 introduces too much noise, making it difficult to recognise the body movements or the visiting trace in the virtual home.

Quantitative data utility controlling. Based on the qualitative results, we further determine the data utility using RMSE thresholds. In Figure 5.5, we present the RMSE between the original data and the differential privacy-enhanced data with various ϵ settings. Specifically, we vary the values of ϵ from 1 to 10 with a step size of 1. It can be observed that the RMSE decreases as ϵ increases. Moreover, there is an elbow point in the RSE-ϵ curve, indicating that an appropriate RSE threshold could be chosen near this point. In this region, increasing ϵ has less impact on the RSE, allowing us to select a privacy budget that is sufficiently tight.

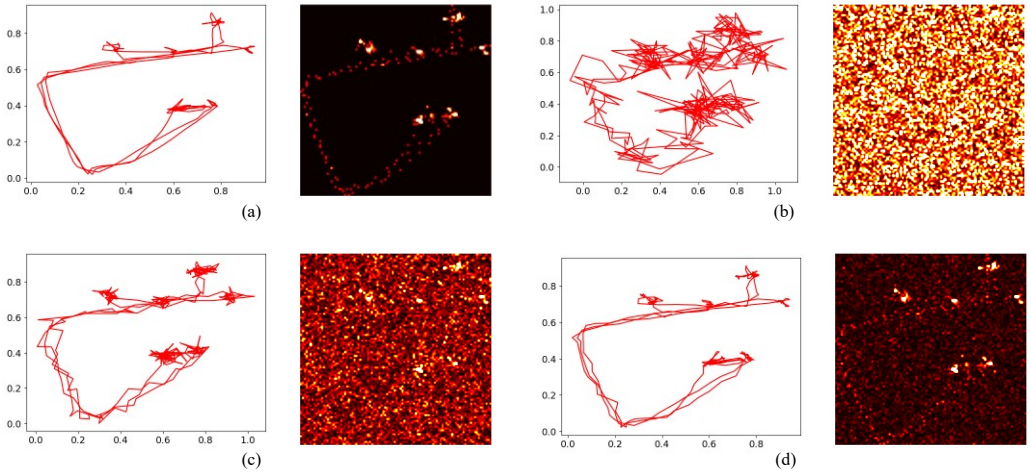


Figure 5.3. Qualitative evaluation of data utility (an example from VirtualHome dataset). (a) original data and corresponding heat map; (b-d) data visualisation after applying Laplace differential privacy with $\epsilon = 1, 3, 10$, respectively. Higher privacy budgets introduce less noise into the data. To maintain the data utility, a privacy budget higher than 3 should be selected, since when $\epsilon = 3$ the noise level is still high.

According to Section 5.4, we selected a threshold of $RMSE \leq 1$ near the elbow point for our experiments, which led to different values of ϵ being chosen (*i.e.* $\epsilon = 4$ for window-slicing data and $\epsilon = 7$ for heat map data). This difference arises because, when applying differential privacy to heat maps, perturbations are introduced to all points in the heat map (in our case, 100*100 points); whereas differential privacy applied to window-sliced data only introduces noise to individual time series data points, resulting in a smaller amount of noise being added with the same privacy budget, compared to the heat map case.

5.6.2 Privacy Performance against User Identification Attacks

We further evaluate the privacy protection performance of applying differential privacy to streamed spatial data and its corresponding heat map. We conduct user identification attacks on the original data and privacy-enhanced data using various privacy budget settings (*i.e.*, $\epsilon = 1, 3, 10$), as well as an ϵ value selected based on the data utility threshold determined in the previous experiments.

As shown in Table 5.1, the attack success rates experience a significant drop when differential privacy is applied. For instance, in the VirtualHome (Body

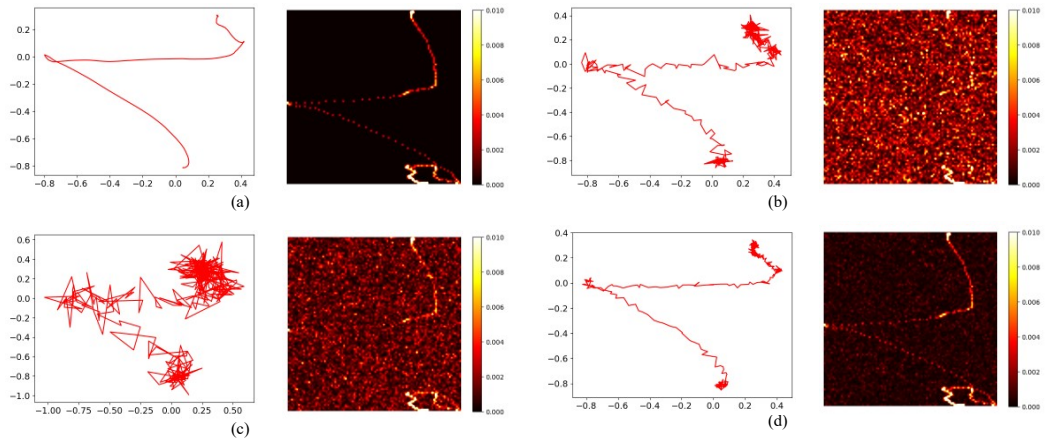


Figure 5.4. Qualitative evaluation of data utility (an example from Body Movement dataset). (a) original data and corresponding heat map; (b-d) data visualisation after applying Laplace differential privacy with $\epsilon = 1, 3, 10$, respectively. Higher privacy budgets introduce less noise into the data. To maintain the data utility, a privacy budget around 3 could be selected, since when $\epsilon = 3$ the noise level is acceptable and a stronger privacy protection can be pursued.

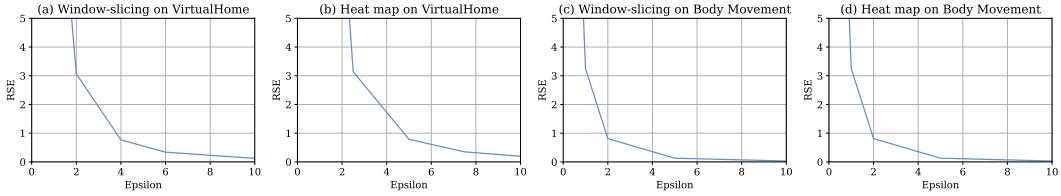


Figure 5.5. RSE between the original data and the differential privacy-enhanced data with various ϵ settings. The query is the averaged user visiting trace.

Movement) dataset, the identification rate decreases in the range of from 41.16% to 45.86% (from 22.44% to 27.09%), compared to the baseline attack success rates when no privacy enhancement is applied.

5.7 Discussion

Our findings demonstrate that DP can effectively mitigate the risk of re-identification attacks, while still preserving the utility of understanding average 3D body motion. This balance between privacy and utility is crucial in various applications. For instance, in gaming, understanding players' body motions can lead to enhancements in the gaming experience. Similarly, in a virtual shopping context,

Table 5.1. Privacy enhancement against user identification attack.

Datasets	Methods	Models	User Identification Accuracy					
			Original	Privacy Enhanced with Differential Privacy				
				$\epsilon = 0.0005$	$\epsilon = 0.001$	$\epsilon = 0.002$	$\epsilon = 0.005$	ϵ controlled by utility threshold
Body Movement	Window-slicing	LSTM	85.42%	15.63	31.25	61.97	79.17	58.33%
Body Movement	Heatmap	CNN	66.67%	14.09	24.83	63.09	87.92	44.23%
			Original	Privacy Enhanced with Differential Privacy				
				$\epsilon = 0.1$	$\epsilon = 0.25$	$\epsilon = 0.5$	$\epsilon = 0.75$	ϵ controlled by utility threshold
VirtualHome	Window-slicing	LSTM	93.89%	27.15	36.51	53.07	75.91	48.03%
VirtualHome	Heat map	CNN	96.53%	14.09	24.83	63.09	87.92	55.37%

understanding shoppers' movements and interactions can help improve the virtual shopping experience. Thus, the application of DP not only ensures user privacy but also contributes to the refinement of user experiences in virtual environments.

Our result also suggest that the transformation of a user's 3D body motion data into heat maps can effectively enhance user privacy. When the same privacy budgets are assigned to both the raw 3D body motion data and the heat maps, the heat maps approach can incorporate more noise into the data, thereby increasing the level of privacy protection. Importantly, this increase in noise does not significantly compromise the utility of the data. This means that important patterns and trends within the data can still be identified, which is vital for analysing user behaviour and improving virtual experiences.

5.8 Limitations and Future Work

In this study, we utilized two VR application datasets to investigate user visiting tracing in virtual home scenarios and body movement in an interactive VR game. Although our choice of datasets is limited, we acknowledge the inherent limitations in terms of representing the entire scope of VR scenarios. It is important to note that the field of VR is vast and diverse, encompassing various applications and user interactions. Despite such limitation, we contend that our approach holds general applicability across a wide range of scenarios involving the collection of users' spatial data.

Another limitation of our study could be the number of attack models involved in the experiments. We acknowledge that there exist more complex models that could be used for user identification attacks. However, the primary goal of

our study is to demonstrate that the application of differential privacy mechanisms can significantly reduce the success rate of user identification attacks, thus enhancing user privacy. In future research, we aim to conduct more comprehensive investigations into how the complexity or structure of attack models can influence privacy protection. This includes exploring whether a more complex privacy protection method or a tighter privacy budget is necessary when facing stronger attack models.

Additionally, it is crucial to explore the delicate balance between data utility and privacy protection. Further studies can shed light on how to optimise this trade-off and develop strategies that effectively preserve data utility while ensuring robust privacy protection. By addressing these aspects, we can advance the understanding of privacy protection in the context of VR applications and offer valuable insights into the design of more resilient and efficient privacy-preserving mechanisms.

5.9 Conclusion

In this chapter, we introduced a novel approach that leverages differential privacy to safeguard users' privacy in the context of VR applications, specifically focusing on streamed spatial data. By applying differential privacy mechanisms, we effectively mitigate the risk of user identification through attacks, thereby enhancing privacy protection.

Furthermore, we have demonstrated the efficacy of our proposed heat map method, which surpasses the direct application of differential privacy to streamed spatial data. The heat map method provides a more robust privacy protection mechanism by introducing perturbations to all points in the heat map, rather than solely focusing on individual data points. This approach allows for a more comprehensive and effective privacy enhancement while preserving data utility. Through our experiments and evaluation, we have shown that applying differential privacy to heat map data can significantly reduce the success rates of user identification attacks. This validates the effectiveness of our approach in enhancing user privacy within the VR domain.

Overall, our work contributes to the growing body of research on privacy preservation in VR applications, offering insights into the application of differential privacy mechanisms and highlighting the benefits of the heat map method for

robust privacy protection. These findings pave the way for further advancements in privacy-preserving techniques in the field of VR, ensuring that users' privacy is safeguarded while enjoying immersive and interactive experiences.

Bibliography

[1] Virtual reality market size, share & trends analysis report. https://www.grandviewresearch.com/industry-analysis/virtual-realityvr-market, 2021.

[2] Meta connect 2022: Meta Quest Pro, more social vr and a look into the future. https://about.fb.com/news/2022/10/meta-quest-pro-social-vrconnect-2022/, 2022.

[3] David Heaney. Apple reportedly has 3000 people working on its upcoming headset. https://uploadvr.com/apple-3000-staff-ar-vr-headset/, 2022.

[4] Immersive media technologies: The acceleration of augmented and virtual reality in the wake of COVID-19. https://www.weforum.org/reports/immersive-media-technologies-the-acceleration-of-augmented-and-virtual-reality-in-the-wake-of-covid-19, 2022.

[5] Jaybie A De Guzman, Kanchana Thilakarathna, and Aruna Seneviratne. Security and privacy approaches in mixed reality: A literature survey. *ACM Computing Surveys (CSUR)*, 52(6):1–37, 2019.

[6] Lauren E Buck and Bobby Bodenheimer. Privacy and personal space: Addressing interactions and interaction data as a privacy concern. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 399–400. IEEE, 2021.

[7] Jingdong Jia and Wenchao Chen. The ethical dilemmas of virtual reality application in entertainment. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 1, pages 696–699. IEEE, 2017.

[8] Vivek Nair, Gonzalo Munilla Garrido, and Dawn Song. Exploring the unprecedented privacy risks of the metaverse. *arXiv preprint arXiv:2207.13176*, 2022.

[9] Devon Adams, Alseny Bah, Catherine Barwulor, Nureli Musaby, Kadeem Pitkin, and Elissa M Redmiles. Ethics emerging: the story of privacy and security perceptions in virtual reality. In *SOUPS@ USENIX Security Symposium*, pages 427–442, 2018.

[10] Ken Pfeuffer, Matthias J Geiger, Sarah Prange, Lukas Mecke, Daniel Buschek, and Florian Alt. Behavioural biometrics in vr: Identifying people from body motion and relations in virtual reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[11] Thomas Germain. Meta's new headset will track your eyes for targeted ads. https://gizmodo.com/meta-quest-pro-vr-headset-track-eyes-ads-facebook-1849654424, 2022.

[12] Vivek Nair, Wenbo Guo, Justus Mattern, Rui Wang, James F O'Brien, Louis Rosenberg, and Dawn Song. Unique identification of 50,000+ virtual reality users from head & hand motion data. *arXiv preprint arXiv:2302.08927*, 2023.

[13] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[14] Ruobin Gong, Erica L. Groshen, and Salil Vadhan. Harnessing the Known Unknowns: Differential Privacy and the 2020 Census. *Harvard Data Science Review*, 2022.

[15] Chuck Kapelke. Using differential privacy to harness big data and preserve privacy. https://www.brookings.edu/articles/using-differential-privacy-to-harness-big-data-and-preserve-privacy/, 2021.

[16] Gonzalo Munilla Garrido, Vivekand Nair, and Dawn Song. Going incognito in the metaverse. *arXiv preprint arXiv:2301.05940*, 2023.

[17] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[18] Jonathan Liebers, Mark Abdelaziz, Lukas Mecke, Alia Saad, Jonas Auda, Uwe Gruenefeld, Florian Alt, and Stefan Schneegass. Understanding user identification in virtual reality through behavioral biometrics and the effect of body normalization. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2021.

[19] Mark Roman Miller, Fernanda Herrera, Hanseul Jun, James A Landay, and Jeremy N Bailenson. Personal identifiability of user tracking data during observation of 360-degree vr video. *Scientific Reports*, 10(1):1–10, 2020.

[20] Ilesanmi Olade, Charles Fleming, and Hai-Ning Liang. Biomove: Biometric user identification from human kinesiological movements for virtual reality systems. *Sensors*, 20(10):2944, 2020.

[21] Howe Yuan Zhu, Hsiang-Ting Chen, and Chin-Teng Lin. The effects of a stressful physical environment during virtual reality height exposure. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 468–469. IEEE, 2021.

[22] Pier Paolo Tricomi, Federica Nenna, Luca Pajola, Mauro Conti, and Luciano Gamberi. You can't hide behind your headset: User profiling in augmented and virtual reality. *IEEE Access*, 11:9859–9875, 2023.

[23] Cong Shi, Xiangyu Xu, Tianfang Zhang, Payton Walker, Yi Wu, Jian Liu, Nitesh Saxena, Yingying Chen, and Jiadi Yu. Face-mic: inferring live speech and speaker identity via subtle facial dynamics captured by ar/vr motion sensors. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 478–490, 2021.

[24] Philipp Sykownik, Divine Maloney, Guo Freeman, and Maic Masuch. Something personal from the metaverse: Goals, topics, and contextual factors of self-disclosure in commercial social vr. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2022.

[25] Nadisha-Marie Aliman and Leon Kester. Malicious design in aivr, falsehood and cybersecurity-oriented immersive defenses. In *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 130–137. IEEE, 2020.

[26] Rahmadi Trimananda, Hieu Le, Hao Cui, Janice Tran Ho, Anastasia Shuba, and Athina Markopoulou. {OVRseen}: Auditing network traffic and privacy policies in oculus {VR}. In *31st USENIX security symposium (USENIX security 22)*, pages 3789–3806, 2022.

[27] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*, pages 1–12. Springer, 2006.

[28] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[29] Athanassios Kassidas, John F MacGregor, and Paul A Taylor. Synchronization of batch trajectories using dynamic time warping. *AIChE, 44, 864–875*, 1998.

[30] Wannes Meert; Kilian Hendrickx; Toon Van Craenendonck. wannesm/dtaidistance v2.0.0. *Zenodo*, 2020.

[31] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. Diffprivlib: the IBM differential privacy library. *ArXiv e-prints*, 1907.02444 [cs.CR], July 2019.

[32] Naoise Holohan, Douglas J Leith, and Oliver Mason. Differential privacy in metric spaces: Numerical, categorical and functional data under the one roof. *Information Sciences*, 305:256–268, 2015.

Chapter 6

Conclusion

This chapter summaries the material covered in our thesis and discusses possible future work in this area.

6.1 Summary of Thesis

Throughout this thesis, we proposed novel methods and evaluation frameworks for understanding and measuring privacy and security assertions of mobile apps, with specific solutions that address the limitations and challenges existing in the security and privacy research domain. The outcomes of this research not only advance the knowledge in the field of mobile software engineering but also bring a range of benefits to software developers, users, and businesses. Here we summarise how the objectives of this research have been addressed.

- (Addressing Objective 1) Requirements of key privacy regulations are extracted and modelled. The regulation model is used in our privacy policy quality assessment and app behaviour violation detection. The knowledge base to model privacy assertions has been established (Chapters 3.4 and 4.5), containing more than 20K mobile apps (20,195 children's apps and 40 contact tracing apps) and their corresponding privacy policies. The dataset and data collection scripts are open-sourced and publicly available.

- (Addressing Objective 2) A comprehensive security and privacy evaluation system is built to enable large-scale analysis on mobile applications. A series of evaluation tools, including privacy policy segment classification, static analysis (Chapters 3.4.2 and 4.4.2), dynamic analysis (Chapter 4.4.3), and user comments analysis (Chapter 4.4.5), have been developed and open-sourced. Novel methods and technologies are proposed with validated prototypes.

- (Addressing Objective 3) Large-scale security and privacy evaluations are conducted on different datasets and scenarios, including online privacy policy generators, contact tracing apps (Chapter 3.4), and children's apps (Chapter 4.5). The findings of these evaluation successfully reveal the non-compliance among the privacy assertions and application behaviours.

- (Addressing Objective 4) A novel approach is proposed and evaluated, which leverages differential privacy to safeguard users' privacy in the context of VR applications, specifically focusing on streamed spatial data. By applying differential privacy mechanisms, we effectively mitigate the risk of user identification through attacks, thereby enhancing privacy protection (Chapter 5).

- Based on the large-scale evaluation of mobile application privacy practice, the outputs of all research tasks is thoroughly evaluated and summarised into takeaways and development guidelines. In addition to real-world mobile application developers, application users were also contacted to ensure that their interests and concerns are well respected.

Chapter 3 develops 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. COVIDGuardian and the issues raised through responsible disclosure to vendors, can contribute to the safe deployment of mobile contact tracing.

Chapter 4 focuses on the privacy practices of apps that are designed for children or have target users that include children. We have measured the use of permissions and trackers, investigated inconsistency in content ratings, and analysed user comment feedback. Our measurement results illustrate that, despite many privacy protection regulations and the strict requirements imposed by the app store, children still experience privacy threats. This is caused by things like permission requests without child-friendly notifications, abuse of ad trackers, confusing and inconsistent content ratings, as well as privacy leakage without users consent. Ultimately, we conclude that the existing self-certification-based content rating mechanism must be improved immediately.

Chapter 5 introduces a practical approach for enhancing the privacy of streamed spatial data collected from VR applications, specifically targeting user identification attacks. Our proposed method involves converting the streamed spatial data into heatmaps, representing the data with frequencies. This approach ensures a comprehensive and effective privacy enhancement while maintaining data utility. Through the development of our method and the subsequent evaluation, we contribute to the growing body of research on privacy protection in VR environments. Our work highlights the potential of applying differential privacy to heatmap data as an efficient and reliable means of safeguarding user privacy. By adopting our proposed approach, VR applications can better

ensure the protection of user identities and maintain a secure environment for user interactions.

6.2 Future Work

The privacy-enhancing technologies and mobile application behaviour evaluation approaches proposed in this thesis advance the knowledge and software development in the field of mobile security and privacy research domain. However, much work still remains for further study. In this section, we discuss the improvements that could be made to our proposed technologies, as well as a more general area for future research.

First, to extend our study on contact tracing applications, we plan to

(*i*) Obtain user feedback from a wider geographic and demographic range. User feedback could be a source of concerns and complaints that potentially expose vulnerabilities or design flaws. However, it is challenging to automatically understand such information on a large scale, considering the breadth of topics discussed by users, the noise in the feedback comments (such as slang, grammatical errors, and abbreviations), and the performance of the state-of-the-art NLP technologies.

(*ii*) Examine network traffic originating from contact tracing apps. The current evaluation did not involve network traffic information which could be monitored by a dynamic testing framework. Dynamic tests can further confirm the privacy leaks detected by the static approach. However, the false negatives may influence the evaluation results quite a lot, *i.e.*, not every privacy leak could be confirmed in network traffic.

(*iii*) Extend the assessment to a broader scope. The security and privacy evaluation could be extended to other mobile applications, beyond contact tracing applications. The security and privacy guidance and suggestions can also be evaluated and further applied to a broader scope in the software industry.

Second, our study on the privacy practice of applications designed for children could be further improved with the following aspects considered. First, a user study could collect users' feedback more fine-granularly. One foreseeable challenge in such a user study could be the recruitment of children as participants in the user study. Furthermore, the app store policy involved in the current

research limits to Google Play store Designed for Family policy. It would be interesting to extend to other app stores, checking the application compliance with other app store family policies, and comparing the behaviour of applications across different venues. In addition, a notification framework could be developed and validated to ensure a children's application only access the private information of children users with consent from their legal guardians.

Last but not least, to provide more comprehensive analysis and evaluation on privacy enhancement with differential privacy, we would like to further evaluate our proposed heatmap method on more datasets and other VR scenarios (*e.g.*, other types of data) and involve more complex attacking models.