



Saarland University
Department of Computer Science

Understanding and Measuring Privacy Violations in Android Apps

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

von
Trung Tin Nguyen

Saarbrücken, 2023

Tag des Kolloquiums: 25.07.2023

Dekan: Prof. Jürgen Steimle

Prüfungsausschuss:

Vorsitzender: Prof. Dr. Thorsten Herfet

Berichterstattende: Prof. Dr. Michael Backes

Dr. Ben Stock

Prof. Dr. Narseo Vallina-Rodriguez

Akademischer Mitarbeiter: Dr. Christine Utz

Zusammenfassung

Die zunehmende Datenerfassung und Verfolgung von Konsumenten durch die heutigen Online-Dienste wird zu einem großen Problem für individuelle Rechte. Es wirft eine ernsthafte Frage auf, ob eine solche Datenerfassung nach der weltweiten Gesetzgebung juristisch begründet werden kann. Leider hat die Gemeinschaft keinen Einblick in diese Verstöße im mobilen Ökosystem.

In dieser Dissertation nähern wir uns diesen Problemen, indem wir eine Arbeitslinie vorstellen, die ein umfassendes Verständnis von Datenschutzverletzungen in Android-Apps in der Praxis bietet und solche Verstöße automatisch misst. Zunächst entwickeln wir ein automatisiertes Tool, das unvorhergesehene Datenzugriffe basierend auf der Nutzung der Benutzeroberfläche von Apps erkennt. Danach führen wir eine umfangreiche Studie zu Android-Apps durch, um zu verstehen, wie häufig Verstöße gegen die ausdrückliche Zustimmung der GDPR vorkommen. Schließlich hat bis jetzt keine Studie systematisch die gegenwärtig implementierten Zustimmungen und deren Übereinstimmung mit der GDPR in mobilen Apps analysiert. Daher schlagen wir einen meist automatisierten und skalierbaren Ansatz vor, um die aktuellen Praktiken von Zustimmungen zu identifizieren. Danach entwickeln wir ein Tool, das Daten erkennt, die mit unterschiedlichen Zustimmungsbedingungen ins Internet gesendet werden.

Unser Ergebnis zeigt den dringenden Bedarf an einer transparenteren Gestaltung von Benutzeroberflächen, um die Nutzer besser über den Datenzugriff zu informieren, und wir fordern neue Tools, die App-Entwickler bei diesem Unterfangen unterstützen.

Abstract

Increasing data collection and tracking of consumers by today's online services is becoming a major problem for individuals' rights. It raises a serious question about whether such data collection can be legally justified under legislation around the globe. Unfortunately, the community lacks insight into such violations in the mobile ecosystem.

In this dissertation, we approach these problems by presenting a line of work that provides a comprehensive understanding of privacy violations in Android apps in the wild and automatically measures such violations at scale. First, we build an automated tool that detects unexpected data access based on user perception when interacting with the apps' user interface. Subsequently, we perform a large-scale study on Android apps to understand how prevalent violations of GDPR's explicit consent requirement are in the wild. Finally, until now, no study has systematically analyzed the currently implemented consent notices and whether they conform to GDPR in mobile apps. Therefore, we propose a mostly automated and scalable approach to identify the current practices of implemented consent notices. We then develop an automatic tool that detects data sent out to the Internet with different consent conditions.

Our result shows the urgent need for more transparent user interface designs to better inform users of data access and call for new tools to support app developers in this endeavor.

Background of this Dissertation

This dissertation is based on the papers presented in the following, which have been accepted and presented at top peer-reviewed IT Security and Privacy conferences. I contributed to all these research projects as the main author, and was responsible for all major parts, including the methodology design, implementation, user study design, and paper writing.

Chapter 3 presents the initial work [P1] of this dissertation, i.e., presented in Asia CCS 2021. The research idea is inspired by the author’s master thesis, originally proposed by Duc Cuong Nguyen. In particular, the master thesis presents the preliminary results on unexpected data access in Android apps by comparing the app user interface (UI) and its actual behavior. However, how users perceive an app’s data access when interacting with graphical UI has not been fully studied, which is an important step in detecting unexpected data access. To fill this research gap, in this work [P1], we directly measure how end-users perceive app behaviors based on graphical UI elements via extensive user studies. The results are used to build an automated tool - GUIBAT (Graphical User Interface Behavioral Analysis Tool) - that detects unexpected data access based on user perception. Micheal Schilling contributed by helping with regression analysis that predicts participants’ expectations regarding data access and reviewing the author’s proposed user study designs. Duc Cuong Nguyen, Gang Wang, Michael Schilling, and Michael Backes were involved in general writing. In general, all authors performed reviews of the paper.

Chapter 4 outlines the second part [P2] of this dissertation which is published in USENIX Security 2021. The research idea derives from the author’s previous work [P1]. In particular, [P1] shows that while Android platforms use permission-based mechanisms to regulate data access, they fail to achieve real transparency about how users’ data is collected, used, and shared. More seriously, such data collection could violate regulatory bodies around the globe, e.g., the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). As such, in [P2], we perform the first large-scale measurement on Android apps in the wild to understand the current state of the violation of data protection laws, focusing on GDPR’s explicit consent. The author presented the idea, and the preliminary results in the Research Area Talk at CISP – Helmholtz Center for Information Security. After the talk, the author further discussed with Ben Stock regarding the topic. Ben Stock contributed by helping send app developers email notifications, and further guided the research direction. Ninja Marnau contributed by helping to identify the legally defined data controllers and carefully reviewing the paper’s legal wording. Ninja Marnau, Michael Backes, and Ben Stock were involved in general writing and performed reviews of the paper.

Chapter 5 is based on the work [P3] presented at CCS 2022. Adopted in May 2018, the European Union’s General Data Protection Regulation (GDPR) requires the consent for processing users’ personal data to be freely given, specific, informed, and unambiguous. While prior work has shown that this often is not given through automated network traffic analysis, until now, no research has systematically studied how consent notices are currently implemented and whether they conform to GDPR in mobile apps. To close this research gap, the author had a research idea to perform the first large-scale study into consent notices for third-party tracking in Android apps to

understand the current practices and the current state of GDPR's consent violations. All authors performed reviews of the paper.

Author's Papers for this Thesis

- [P1] Nguyen, T. T., Nguyen, D. C., Schilling, M., Wang, G., and Backes, M. Measuring User Perception for Detecting Unexpected Access to Sensitive Resource in Mobile Apps. In: *ACM Asia Conference on Computer and Communications Security*. 2021.
- [P2] Nguyen, T. T., Backes, M., Marnau, N., and Stock, B. Share First, Ask Later (or Never?) - Studying Violations of GDPR's Explicit Consent in Android Apps. In: *USENIX Security Symposium*. 2021.
- [P3] Nguyen, T. T., Backes, M., and Stock, B. Freely Given Consent? Studying Consent Notice of Third-Party Tracking and Its Violations of GDPR in Android Apps. In: *ACM Conference on Computer and Communications Security*. 2022.

Further Contributions of the Author

The author was also able to contribute to the following papers.

- [S1] Elbitar, Y., Schilling, M., Nguyen, T. T., Backes, M., and Bugiel, S. Explanation Beats Context: The Effect of Timing & Rationales on Users' Runtime Permission Decisions. In: *USENIX Security Symposium*. 2021.
- [S2] Wi, S., Nguyen, T. T., Kim, J., Stock, B., and Son, S. DiffCSP: Finding Browser Bugs in Content Security Policy Enforcement through Differential Testing. In: *Network and Distributed System Security Symposium*. 2023.

Acknowledgments

Pursuing a Ph.D. is a challenging journey that cannot be done without the help of others who supported and encouraged me. In the following, I would like to take the opportunity to express my most sincere thanks to those people that made this journey possible.

First of all, I would like to thank my supervisor Michael Backes for offering me this great opportunity to do my research in the Information Security & Cryptography group. He has provided a unique research environment where I have the opportunity to decide on the topics of my interests. I am grateful for his time, support, and feedback throughout my Ph.D. studies, and I am honored to have had the chance to work closely with him and learn from him.

Second, I want to express my special thank to Ben Stock for being a fantastic academic mentor, giving me valuable feedback, guiding, and motivating me in different research projects. On top of that, he has been really supportive during my Ph.D. I have learned a lot from him, and this dissertation would not have been possible without him.

I also extend my thanks to the reviewers of this thesis, Michael Backes, Ben Stock, and Narseo Vallina-Rodriguez, for their time and effort spent reviewing this dissertation. Additionally, I would like to thank Thorsten Herfet for chairing the examination board and Christine Utz for writing the protocol for my dissertation defense. Further, I am also thankful to all my collaborators: Duc Cuong Nguyen, Gang Wang, Michael Schilling, Ninja Marnau, Sven Bugiel, Yusra Elbitar, Seongil Wi, Jihwan Kim, Soel Son. It was a pleasure for me to work with every one of them.

Many thanks to my great officemates throughout my Ph.D., Jie Huang, Oliver Schranz, Sebastian Weisgerber, and Yang Zou. Thank you for all the productive research discussions, as well as chats about many interesting topics that kept me motivated. Also, I am very thankful to everybody that I have worked with at CISPA, which created an enjoyable working environment.

Last but not least, a big thanks to my wife, Anh Pham, who has always been there with me. Thank you for your endless love and support. It means so much to have you by my side through everything. I am incredibly grateful to my family for their unconditional love. Without their support and encouragement, I could not go this far.

Contents

1	Introduction	1
1.1	Contributions	5
1.1.1	Understanding User Perception for Detecting Unexpected Data Access (RQ1)	5
1.1.2	Detecting Violations of GDPR’s Explicit Consent (RQ2)	5
1.1.3	Studying Consent Notice of Third-Party Tracking and Its Violations of GDPR (RQ3)	6
1.2	Outline	7
2	Background and Related Work	9
2.1	Technical Background	11
2.1.1	Android Platform	11
2.1.2	Android Apps	13
2.1.3	Permissions on Android	14
2.1.4	App Market Store	15
2.1.5	Static and Dynamic Analysis Primer	15
2.2	Legal Background	16
2.2.1	General Data Protection Regulation (EU GDPR)	16
2.2.2	GDPR Consent	18
2.2.3	Summary	19
2.3	Related Work	19
2.3.1	Detecting Data Access that Violates User Expectations	20
2.3.2	Studying the Legislation Violations of Online Services	20
2.3.3	Summary	22
3	Automatic Detection of Unexpected Data Access	23
3.1	Measuring Users’ Perception	25
3.1.1	App and Icon Dataset	26
3.1.2	First Study: Identifying Commonly-Known Icons and The Associated Functionality	27
3.1.3	Second Study: Identifying User Expectation of Icons and Sensitive Resource Access	29
3.2	Detecting Unexpected Sensitive Resource Access	31
3.2.1	GUI Extractor	31
3.2.2	Expected Sensitive Resource Access Detector	33
3.2.3	Actual Sensitive Resource Access Detector	36

CONTENTS

3.2.4	Unexpected Sensitive Resource Access Detector	38
3.3	Evaluating GUIBAT	38
3.3.1	Comparing to Users' Expectations	39
3.3.2	Comparing to Prior Works	43
3.4	Empirical Study of Unexpected Data Access in The Wild	45
3.5	Summary	47
4	Detecting Violations of GDPR's Explicit Consent	49
4.1	Legal Analysis of GDPR's Explicit Consent	52
4.2	Methodology	53
4.2.1	App Setup and Network Traffic Collection	53
4.2.2	Traffic Log Analyzer	54
4.3	Large-Scale Analysis	57
4.3.1	App Dataset Construction	57
4.3.2	Network Traffic Analysis	58
4.3.3	Identifying Advertisement Domains	59
4.3.4	In-Depth Analysis of Violations	60
4.4	Developer Notification	65
4.4.1	Notification and Accessed Reports	66
4.4.2	Developer Responses	67
4.4.3	Updates to Notified Apps	68
4.5	Summary	69
5	Studying Consent of Third-Party Tracking and Its Violations of GDPR	71
5.1	Legal Analysis of Potential GDPR Consent Violations	73
5.1.1	Lack of Consent Notices	74
5.1.2	Sharing Data Before Given Consent	74
5.1.3	No Way to Opt Out	75
5.1.4	Non-Respect of Choice	75
5.2	Methodology	75
5.2.1	Collecting Privacy-Related User Interface	76
5.2.2	Identifying Consent Notices	77
5.3	Large-Scale Analysis	79
5.3.1	App Dataset Construction	79
5.3.2	Identifying Consent Notices In The Wild	80
5.3.3	Automating Potential GDPR Violations Detection	82
5.3.4	Observed Potential Violations	85
5.4	Developer Notification	89
5.4.1	Notification and Accessed Reports	90
5.4.2	Developer Responses	90
5.4.3	Updates to Notified Apps	91
5.5	Summary	91

6	Conclusion	93
6.1	Discussion	95
6.1.1	Transparency of Sensitive Resources Access	95
6.1.2	Transparency of Processing Users' Personal Data	96
6.1.3	Widespread Violation of GDPR Consent	96
6.1.4	Lack of Support for Developers	97
6.2	Calls to Action	97
6.2.1	Authorities Should Actively Enforce The Law	98
6.2.2	Third Parties Should Take Responsibility	98
6.2.3	App Stores Should Take Actions	99
6.2.4	Support for Developers	100
6.3	Limitations and Future Work	100
6.3.1	Detecting Unexpected Data Access	100
6.3.2	Detecting Violation of GDPR's Explicit Consent	102
6.3.3	Studying Consent Notices of Third-Party Tracking	102
6.3.4	Future Research Directions	103
6.4	Conclusion	104
A	Appendix	119
A.1	GUIBAT	120
A.1.1	Second Study	120
A.1.2	Third Study	120
A.1.3	Demographic	121
A.1.4	Evaluating of GUIBAT and GATOR	122
A.2	GDPR's Explicit Consent: Email Template	122
A.3	Manual Version Analysis	123
A.4	GDPR's Consent Requirements: Email Template	123

List of Figures

2.1	The Android software stack (adapted from [67]).	12
3.1	Overview of the methodologies to measure user’s perception.	26
3.2	Overview of the GUIBAT’s architecture.	31
3.3	Example of expected sensitive resource accesses.	33
3.4	ROC curves of the 10-Fold cross-validation.	34
3.5	Relative cumulative frequency (RCF) of accessed sensitive resources depth level of CFG in 10,000 apps.	37
3.6	<i>Unexpected SR</i> of a benign app (on Play Store).	38
3.7	Distribution of <i>unexpected SR</i> among sensitive resources in 36,115 apps. Attribution of app code and library code sum up to 100% in each sensitive resources.	45
3.8	Top 10 libraries with <i>unexpected SR</i>	46
4.1	Example of consent dialogues in Android apps.	51
4.2	Overview of our methodology to identify potential UIDs. After each step, the analysis terminates if the resulting set of candidate parameters is empty.	56
4.3	Top 10 ad domains that frequently received personal data from 24,838 apps that sent personal data to all ad-related domains.	62
4.4	Top 5 ad domains receiving personal data in each app set after applying the Exodus-Privacy filtering to the high-profile set (percentages relative to the personal data sending apps per dataset).	64
5.1	Example of various types of consent notices in Android apps.	73
5.2	Overview of the methodology to identify consent notices in Android apps and the (intermediate) results.	76
5.3	The dendrogram of our hierarchical clustering.	81
5.4	Example of the four types of consent choices.	82
5.5	Overview of our methodology to detect data sent out to the Internet with different given consent conditions.	83
5.6	Top 10 ad-domains that received personal data from 30,160 apps that have no consent notices.	86

List of Tables

3.1	The transformation techniques that are used.	27
3.2	Icon groups and user expectation of sensitive resource accesses . An icon group has multiple similar looking icons. The agreement rate is in parenthesis.	30
3.3	Sensitive Related Keywords.	36
3.4	Detected sensitive resource accesses in our sample.	39
3.5	Model comparison based on Goodness of fit.	41
3.6	Final regression model predicting users' expectation regarding sensitive resource accesses.	42
3.7	Example of <i>unexpected SR</i> that are detected by GUIBAT from DeepIntent's benign training dataset.	43
3.8	The AUC score of GUIBAT's Icon Classifier and DeepIntent's models.	44
3.9	Distribution of <i>expected SR</i> on icons/texts.	44
4.1	Overview of personal data tied to a phone.	53
4.2	Examples of the UIDs identified by our approach.	57
4.3	Types of data and number of apps sending this to any, third-party, and ad domains (percentages relative to dataset sizes).	59
4.4	Top 5 of ad domains receiving AAID along with other personal data in our two app datasets.	63
5.1	Text keywords based on each type of consent choice.	84
5.2	Types of personal data we consider in our work.	84
5.3	Types of consent notices and number of potentially violated apps (percentages relative to each violation type).	86
5.4	Types of data and number of apps sending this data to ad-related domains (percentages relative to each violation type).	87
A.1	Demographics of participants from our studies.	121
A.2	The evaluating results of GUIBAT and Gator 3.5 on a set of 34 apps. V=Views. C=Callbacks. (+)=New. (-)=Removed.	125
A.3	Companies detected as ad-related, for which our analysis of legal documents indicate they act as data controllers.	126
A.4	Companies detected as ad-related, for which our analysis of legal documents indicate they act as data controllers.	127

1

Introduction

Every time we load a page on a commercial website or use a mobile app, information about us and about what we are doing online will be broadcast to large numbers of companies, most notably for advertising purposes [81, 54]. This user tracking happens hundreds of billions of times every day, which is becoming a major problem for individuals’ rights regarding their data [54, 103].

To protect user privacy, regulatory bodies around the globe have attempted to address the user tracking problem through regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) — which mandate online services to *transparently* disclose how they handle personal data and grant users crucial data protection rights [135, 26]. A key principle of these regulations is *transparency*, i.e., users should be well-informed regarding how their data is collected, used, and shared [109]. As a result of the legislation, developers are, for example, adding or changing privacy policy, having additional consent notice popups, or removing third-party libraries [163]. However, in practice, users are still in a disadvantaged position to protect their privacy, e.g., remaining unclear for the users when (for example, when using an app function) what data access takes place. Further, it raises a serious question about whether these changes fulfill the legal conditions for collecting and processing user personal data.

In recent years, many researchers have started studying the impact of legislation on online services by proposing different techniques to detect privacy violations. However, while researchers have worked to detect and analyze legislation violations and their impact on the Web advertising and tracking industry [103, 152, 140, 151, 36, 90, 156], the community lacks insight into such violations in the mobile ecosystem. Thus, we need new studies that can provide insight into the transparency of what data is being collected, how it is used, whether it is legally compliant, and the status quo of current data protection law violations in mobile apps. In this dissertation, we approach these problems by presenting a line of work that provides a comprehensive understanding of privacy violations and then automatically measures such violations in Android apps in the wild at scale. For mobile apps, most researchers focused on analyzing the app privacy policies to identify unexpected behaviors or legislation violations, i.e., comparing an app’s actual behavior and the declared data processing in the privacy policy [4, 142, 173, 178]. However, irrespective of a privacy policy, under the GDPR [135], to be legally compliant, an app is required to obtain users’ consent before sharing personal data with third parties if such parties use the data for their own purposes (e.g., personalized advertising), i.e., “consent” packaged in terms and conditions or privacy policies is not compliant [42, 54]. Thus, with respect to the GDPR requirements, we perform the first large-scale measurements on Android apps in the wild to explore the transparency of users’ data access and further understand whether they conform to GDPR legislation — which allows us to provide a comprehensive overview of the current state of privacy violations in Android apps, and to uncover the possible reasons behind such violations.

First, we aim to explore the transparency of data access in Android apps, i.e., related to sensitive resources. Specifically, we want to detect data access that breaks user expectations by matching user expectations and the apps’ actual behaviors. While existing works have used various proxies to infer user expectations (e.g., by analyzing app descriptions), how real-world users perceive an app’s data access when interacting

with graphical user interfaces (UI) has not been fully explored. To fill this research gap, we first directly measure how end-users perceive app behaviors based on graphical user interfaces via extensive user studies. Then, the results are used to build an automated tool - GUIBAT (Graphical User Interface Behavioral Analysis Tool) - that detects data access that violates user expectations. This leads us to the first research question: *(RQ1) Does the output of GUIBAT reflect users' expectations of apps' sensitive resource access? - and if it does - How widespread is sensitive resource access that violates users' expectations in the wild?*. We believe that answering this question will shed new light on the transparency of sensitive resource access in mobile apps¹.

Subsequently, since the General Data Protection Regulation (GDPR) went into effect in May 2018, online services are required to obtain users' explicit consent before sharing users' personal data with third parties that use the data for their own purposes. While violations of this legal basis on the Web have been studied in-depth, the community lacks insight into such violations in the mobile ecosystem. Therefore, orthogonal to prior work, in the second part of this dissertation, we aim to understand how often GDPR's explicit consent mandate is violated in the mobile ecosystem, focusing on Android². Specifically, we build a semi-automated pipeline to detect data sent out to the Internet without prior consent. In addition, our research aims to answer the second question: *RQ2: How many apps send out personal data without prior consent? Of the apps which send out any data, how many send it to parties that act as data controllers under the GDPR? Are developers aware of the requirements of GDPR and the issues that might arise from not following the outlined laws?*. Answering this question allows us to derive concrete recommendations for all involved entities in the ecosystem to allow data subjects to exercise their fundamental rights and freedoms.

Third, the European Union's GDPR requires consent for processing users' personal data to be *freely given, specific, informed, and unambiguous*. While prior work has shown that this often is not given through automated network traffic analysis, no research has systematically studied how consent notices are currently implemented and whether they conform to GDPR in mobile apps. To close this research gap, we perform the first large-scale study into consent notices for third-party tracking in Android apps to understand the current practices and the current state of GDPR's consent violations. Specifically, we propose a mostly automated and scalable approach to identify the currently implemented consent notices. We then develop a tool that automatically detects users' personal data sent out to the Internet with different consent conditions based on the identified mechanisms. To this end, our next research question is: *(RQ3) Do mobile apps implement any form of consent notices? What are the common properties of such consent notices? Can these implemented consent notices be legally justified under*

¹In the Android platform, sensitive resources are permission-protected resources that are organized into groups regarding the device's capabilities or features. This work focused on the dangerous permission groups (i.e., Contact, Phone, Calendar, Camera, Location, Storage, Microphone, SMS), since these permission groups deal with user-sensitive information.

²We note that we refer to the violations as potential because we carefully worded not to make legally conclusive statements since this could amount to legal consulting strictly regulated by our national law. Therefore, only a judicial ruling can provide legal certainty about whether they are actual violations. However, in Section 4.1 and Section 5.1, we reason why they should be considered violations by directing to relevant regulations and legal precedents.

GDPR?. We believe our results can inform future research on the current practices of consent notices in Android apps and enable them to build tools that help developers comply with legislation, such as obtaining valid consent under GDPR.

1.1 Contributions

In the following, we summarize this dissertation’s major contributions, which consist of three primary parts that understand and measure privacy violations of Android apps in the wild at scale, focusing on the GDPR.

1.1.1 Understanding User Perception for Detecting Unexpected Data Access (RQ1)

We, for the first time, perform two user studies to build a semantic mapping between user expectations of sensitive resource accesses and common apps’ icon UI elements (N=459). Our results open a new perspective for identifying the relation between users’ perception of icons and the associated sensitive resource accesses. Our results can lay a concrete foundation for modeling user expectations based on UI elements, lead to better design of apps’ graphical UI, and enhance data access transparency. More importantly, we show prior works that predict the intention of apps’ icon UI elements without understanding how real-world users perceive data access, in large part, do not reflect user expectations.

We build GUIBAT (Graphical User Interface Behavioral Analysis Tool) — a new tool that accounts for users’ perception to detect *unexpected sensitive resource accesses* in Android apps, based upon the knowledge gained from a series of user studies and static control-flow analysis technique. Our evaluations showed that GUIBAT significantly outperforms prior works in identifying user expectations of sensitive resource accesses when interacting with the app’s UI (i.e., can accurately reflect users’ expectations of sensitive resource accesses in apps), and its efficiency enables the large-scale study.

Thereby, we apply GUIBAT on 100,000 Android apps to investigate the landscape of unexpected access to sensitive resources in the wild. GUIBAT identify 36,115 apps with at least one unexpected sensitive resource access. Among these apps, 13,796 (38.2%) apps have unexpected sensitive resource accesses exclusively attributed by third-party libraries. We believe our results will shed new light on the transparency of sensitive resource access in mobile apps. In particular, GUIBAT would (1) help to inform end-users about unexpected access to sensitive resources and (2) help app stores to better control the compliance of apps to the transparency policies.

1.1.2 Detecting Violations of GDPR’s Explicit Consent (RQ2)

We perform the first large-scale measurement on Android apps in the wild to understand the current state of the violation of GDPR’s explicit consent. Specifically, we build a semi-automated and scalable pipeline to detect personal data sent to the Internet by analyzing the network traffic generated by apps without the users’ explicit prior consent. Based on the domains that receive data protected under the GDPR without prior consent,

we collaborate with a legal scholar to assess the extent to which contacted domains are third-party data controllers — which require explicit consent before collecting and processing users’ personal data.

Specifically, we perform a large-scale measurement on a set of 86,163 Android apps in the wild to understand the current state of the violation of GDPR’s explicit consent. Doing so, we find 24,838 apps sent personal data towards advertisement providers that act as data controllers without the user’s explicit prior consent. To inform developers about these issues and understand the reasons behind them, we run a notification campaign to contact 11,914 affected developers and gather insights from 448 responses to our notifications.

Inspired by the responses, we further conduct an in-depth analysis of available documentation and default data collection settings of third-party SDKs. Based on the insights from both developers and our own analysis, we find that GDPR issues are widespread, often misunderstood, and require effort from advertisement providers, app stores, and developers alike to mitigate the problems. Finally, we derive concrete recommendations to all concerned parties and make an urgent call to help developers comply with GDPR.

1.1.3 Studying Consent Notice of Third-Party Tracking and Its Violations of GDPR (RQ3)

We perform a large-scale study with 239,381 Android apps available through an EEA country Play Store, allowing us to provide a comprehensive overview of the consent notices currently implemented in mobile apps in the wild, which no research has systematically studied. Specifically, we propose a mostly automated and scalable approach using image and natural language processing techniques to identify the implemented consent notices and their current practices. As a result, we recognize four widely implemented mechanisms to interact with the consent user interfaces from 13,082 apps, such as confirmation-only notices that feature a button with the text “OK” or “I agree”, or notices that provide options to either accept or decline the data sharing. We believe our results can inform future research on the current practices of consent notices in Android apps and enable them to build tools that help developers comply with legislation, such as obtaining valid consent under GDPR.

Based on the identified mechanisms, we then extend prior work to develop a tool that automatically detects users’ personal data sent out to the Internet with different consent conditions (i.e., based on the choice mechanism to interact with the consent notice). Doing so, we find 30,160 apps do not even attempt to implement consent notices for sharing users’ personal data with third-party data controllers, which mandate explicit consent under GDPR. In contrast, out of 13,082 apps that implement consent notices, we find 2,688 (20.54%) apps violate at least one of the GDPR consent requirements, such as trying to deceive users into accepting all data sharing, sharing before explicitly given consent, or even continuously transmitting data when users have explicitly opted out. While prior studies primarily focus on network traffic analysis, we are the first to analyze the app’s consent user interfaces to evidence the potential violations of GDPR. Further, to inform app developers about their implementation problems and understand

the reasons behind them, we send emails to inform 1,127 affected developers (who have implemented any form of consent notices) and gather insights from their responses.

Based on the insights from both developers and our own analysis, we show the urgent need for more transparent processing of users' personal data and further supporting developers in this endeavor to comply with strict law standards, ensuring users can make free and informed choices regarding their data.

1.2 Outline

This dissertation is organised into six chapters. The remainder of the dissertation is structured as follows. Chapter 2 provides background information to fully understand this dissertation, i.e., technical background on the Android platform, legal background of GDPR legislation. Further, we also present related work and highlight the added value of our contributions in Chapter 2. Following that, in the next three chapters, we present our contributions to answer the research questions stated in Chapter 1. Specifically, Chapter 3 introduces our approach to measure how end-users perceive app behaviors based on graphical UI elements, and to build GUIBAT that detects sensitive resource accesses that violate user expectations. Subsequently, in Chapter 4, we build a semi-automated and scalable solution to detect personal data sent to the Internet by analyzing the network traffic generated by apps without user explicit prior consent, and then perform a large-scale measurement on the mobile apps in the wild to understand the current state of the violation of GDPR's explicit consent. Chapter 5 presents our approach to identifying consent notices currently implemented in Android apps, and our large-scale analysis of Android apps and demonstrates our approaches to detecting potential GDPR consent violations. Finally, in Chapter 6 we discuss the findings, call for actions, our limitations, and future work before summarizing our contributions and concluding this dissertation.

2

Background and Related Work

This chapter provides background information to fully understand this dissertation. In particular, we first present technical information relevant to the topics throughout this work. We then describe the legal background of GDPR and GDPR consent requirements, which are used as the base for our legal analysis throughout this dissertation. Finally, we provide information about the areas of research that relate to our work and highlight the added value of our contributions.

2.1 Technical Background

In the following section, we introduce relevant technical background information for our work. First, based on the official Android documentation [65], we present a brief introduction to the Android platform, e.g., Android’s software stack, Android mobile apps, and the employed permission-based mechanisms on Android. We follow this up with a discussion of app market stores, the primary channel for distributing Android apps to the end-users, e.g., Google Play. Finally, we show general concepts of static and dynamic analysis used in our work.

2.1.1 Android Platform

Android, led by Google, is an open-source operating system (OS), a Linux-based software stack created for different devices and form factors [67]. Android has dominated the market share for smartphones and other devices for many years. In September 2022, it has a global share of 72% with more than 2.6 million active apps in the Google Play store [144, 5]. The popularity of Android apps is mostly due to the unrestricted application market, the nature of open-source platforms, and its comprehensive framework APIs (e.g., allowing develop feature-rich apps). In line with prior large-scale Android security and privacy research [47, 84, 10, 23, 30, 121, 49, 149], focusing on the popularity of the Android ecosystem allows us to regard our findings to represent the privacy violations and their effects on millions of users worldwide.

The Android platform consists of six major components (see Figure 2.1). In the following, we give a brief overview of each component (see the Android documentation [67] for more in-depth information regarding the Android software stack).

The Linux Kernel Linux Kernel is the foundation of the Android platform. This component is responsible for fundamental operating system services (e.g., threading and low-level memory management) and enables access to low-level hardware functionality (e.g., Camera, Bluetooth, network access or telephony) to upper layers.

Hardware Abstraction Layer (HAL) On top of the Linux Kernel, the HAL provides standard interfaces that expose device hardware capabilities (from the Linux Kernel) to the higher-level layers. These interfaces are grouped into different library modules based on the specific type of hardware components, such as the camera or Bluetooth module.

Android Runtime (ART) Due to the limited resources of mobile devices (e.g., low memory), ART is written to run multiple virtual machines on mobile devices by executing

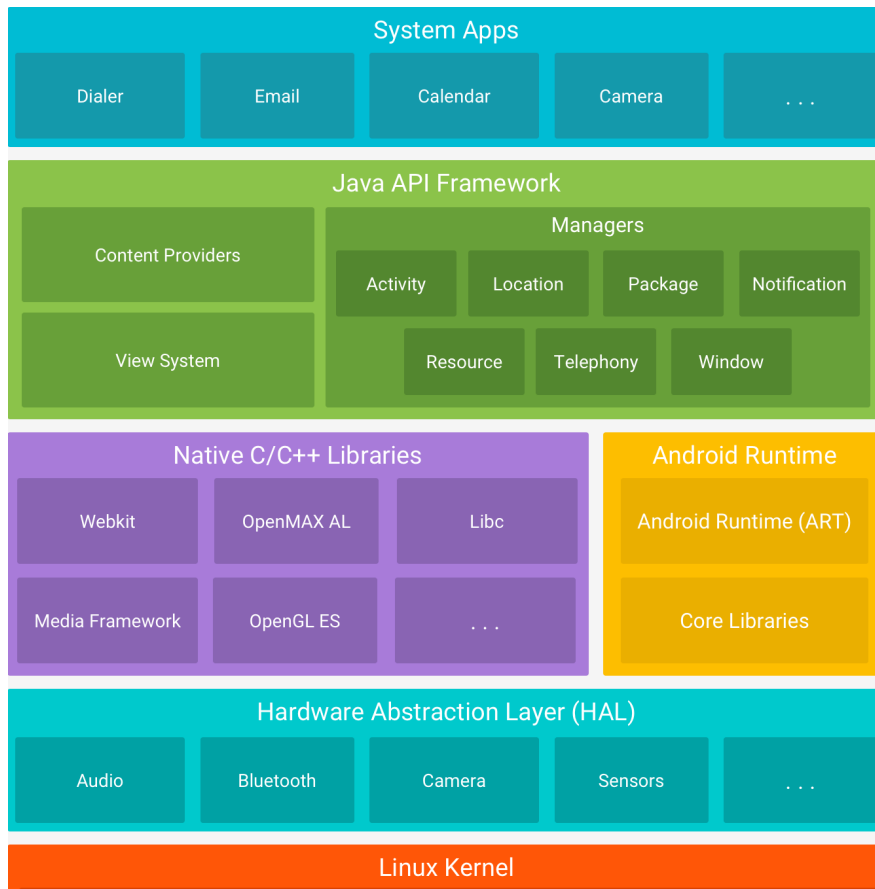


Figure 2.1: The Android software stack (adapted from (67)).

DEX files, i.e., a bytecode format that’s optimized for low-memory devices such as Android phones and tablets. Originally, ART was created especially for the Android project, i.e., ART’s predecessor is called Dalvik Virtual Machine (DVM). From Android version 5.0 (API level 21) or higher, DVM is the ART, and each app runs in its own process and instance of the ART.

Native Libraries and Java API Framework In general, the Java API Framework layer provides Application Programming Interfaces (APIs) written in the Java language for Android app developers to access the entire feature set of the Android OS. These APIs allow app developers to simplify the reuse of core, modular system components and services, e.g., a rich and extensible View System API(s) that can use to build an app’s UI. Further, most Android system components and services are built from native code written in C and C++. As such, to access the device’s hardware capabilities, such as the camera and sensors, the Java API Framework loads the corresponding library module for that hardware component from the C/C++ Native Library layer.

System Apps At the top level, by default, Android includes a set of core system apps that provide essential system capabilities, such as email, SMS messaging, calendars, internet browsing, and contacts. Further, users can use other third-party apps instead of those included with the platform, such as a third-party app that can become the user's default SMS messaging. In the next section (see Section 2.1.2), we discuss these Android apps in more detail.

2.1.2 Android Apps

Besides the system apps that come with the Android by default (e.g., pre-install apps for email, SMS messaging, calendars), the Android platform further allows users to install apps from Google Play or any websites other than Google Play (see Section 2.1.4 for more details of app market stores). To build an Android app, developers can use various programming languages, such as Kotlin, Java, and C++. From the source code, the Android SDK tools compile it and any relevant resources into an Android Package (with a .apk suffix) or an Android App Bundle (with a .aab suffix). Technically, Android-powered devices use the Android Package to install the app at runtime. On the other hand, the Android App Bundle only contains additional metadata that is not required at runtime, and is not installable.

2.1.2.1 App Components

Basically, there are four types of app components that are the essential building blocks of an Android app, and end-users can enter the app via each component as an entry point through the system [68].

- *Activities*: An activity is presented to the users as a single user interface, i.e., includes a set of organized UI elements such as buttons, labels, and text boxes. For example, an SMS app might contain one activity that shows SMS messages, an activity to compose messages, and another for reading messages.
- *Services*: To keep an app running in the background, Android provides Service components (i.e., without user interfaces) that can run in the background to perform long-running operations and processes. For example, a service might download files in the background while the user is composing an SMS message.
- *Broadcast receivers*: A broadcast receiver allows the system to deliver events to the apps, i.e., allowing the app to respond to system-wide broadcast notifications. For example, the Android sends broadcasts when system events occur (e.g., boots up, starts charging, connectivity changing, or low battery). Similarly, apps can send custom broadcasts to notify other apps (e.g., data download completed).
- *Content providers*: A content provider is used to manage a shared set of app data that can be stored in persistent storage, e.g., the file system, SQLite database. With the proper permissions, other apps can query or modify the data throughout the content provider. For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the corresponding permissions can access the content provider to read and write contact information.

2.1.2.2 The Manifest File

The Manifest XML file is an essential part of Android apps because it defines the components and metadata of the app. As such, every Android app must declare all its components in a Manifest XML file (*AndroidManifest.xml*), which is located at the root of the project directory. Besides, the manifest also does the following:

- Declares the necessary user permissions for the app, e.g., Internet access for a Web browser app.
- Declares the minimum API Level that is required by the app, e.g., the oldest Android version that is supported by the app.
- Declares hardware or software features the app uses, e.g., Bluetooth services.
- Declares additional API libraries for the app (other than the built-in Android Framework APIs), e.g., the Google Maps library.

2.1.2.3 App Resources

Besides app code, an Android app is composed of different resources, such as images, audio files, and anything relating to the visual presentation of the app. This allows developers to update various app's characteristics without modifying or compiling the app code. For example, by defining UI strings in XML resources, developers can translate the strings into multiple languages and save those strings in separate files based on a language qualifier (e.g., *res/values-fr/* for French string values). Then Android applies the appropriate language strings to the apps' UI based on the user's language setting.

2.1.3 Permissions on Android

Android offers a comprehensive Java Framework API that supports the development of third-party apps, i.e., enriching the apps' features. Further, the system allows third-party apps to access sensitive capacities of Android devices via these API(s), e.g., including user personal data information and hardware features. Therefore, Android employs a permission system that governs the security and privacy-sensitive APIs (called permission-protected APIs) to protect user privacy. Specifically, each permission-protected API is associated with predefined Android permission (e.g., `Android.permission.READ_CALENDAR`) corresponding to the accessed permission-protected resource, which are categorized into three protection levels.

- *Normal Permissions*: permissions that do not impact the user's privacy or the operations of other apps. For example, the `SET_ALARM` is normal permission that allows apps to set user alarms.
- *Dangerous Permissions*: permissions that deal with users' private information, users stored data, or affect the operations of other apps. For example, the `READ_CONTACTS` permission allows the app reads user contact information, which is sensitive data (e.g., for a list of dangerous permissions, see [66]).

- *Signature Permissions*: permissions that are only granted to the apps that are signed by the same certificate as the apps that define the signature permissions. However, several signature permissions access the most dangerous operations of the system, which are not granted to third-party apps (e.g., delete app packages).

For dangerous permissions, end users must explicitly grant access by installing time or run-time permission models. In contrast, the system automatically grants normal and signature permissions at the installation time [66]. To reduce the overwhelming technical information of permission requests, Android handles permission requests at the group level. For example, the CALENDAR group includes both the READ_CALENDAR and the WRITE_CALENDAR permissions. When an app requests the READ_CALENDAR permission, the system first notifies users that the app needs CALENDAR permission. If users approve, the app can access all permissions in the same group without asking users again.

2.1.4 App Market Store

An app store (or app market store) is a centralized distribution platform to distribute apps to the end users. Generally, each operating system has a specific platform, e.g., iOS, macOS, Windows, or Android. For Android, Google provides a centralized software distribution channel to distribute apps to the end-users, namely Google Play market stores¹. To provide a safe and trusted platform, Google requires app developers must adhere to Google's standards for privacy, security, and content [70]. In particular, developers' apps will go through automated vetting systems (e.g., quality control, censorship, security protection, malicious intentions, and privacy violations) before publishing to end users. Further, Android users can also install apps from any other websites or external sources, such as SDCard.

2.1.5 Static and Dynamic Analysis Primer

Two primary techniques have been established to analyze an Android app: *Static analysis* and *Dynamic analysis*. In the following, we briefly discuss these techniques.

Static Analysis In general, the static program analysis takes the source code of a target program (e.g., an Android app) to examine the code without executing it by checking the code structure, the sequences of statements, and how variable values are processed throughout the different function calls [93]. The main advantage of static analysis is that all the code is analyzed, which is usually more efficient and scalable. However, the technique suffers from dynamic code loading, i.e., it cannot analyze the code that is downloaded during runtime execution.

Dynamic Analysis In contrast, the dynamic program analysis actually executes the target source code to examine its runtime behaviors. As such, we may be unable to analyze all the program code with dynamic analysis due to the complex contexts of program behaviors. Therefore, dynamic program analysis is usually under-approximating,

¹<https://play.google.com/store/apps>

as it is challenging to cover all code, and thus tends to produce false negatives [93]. Whereas static analysis usually leads to over-approximations and causes false positives for multiple reasons, e.g., analyzing unreachable code. However, these two approaches can be combined to get better results.

2.2 Legal Background

In recent years, regulatory efforts around the globe, such as the European General Data Protection Regulation (GDPR) [135], the California Consumer Privacy Act (CCPA) [26], and the China Personal Information Protection Law (PIPL) [128] have been made to protect user privacy online. The key principles of these regulations are: (1) allowing an individual to exercise their rights regarding their data, e.g., a right to opt-out of the sharing of their personal data; (2) providing transparency, i.e., users should be well-informed regarding how their data is collected, used, and shared. Further, in the USA, the Children’s Online Privacy Protection Act (COPPA) [32] was designed to protect children under age 13, placing parents in control over what information is collected from their young children online.

This section describes the legal background of GDPR and GDPR consent requirements, which is used as the base for our legal analysis throughout this dissertation. Generally, the GDPR governs all personal data processing related to individuals in the EU and EEA. Besides, the ePrivacy Directive (also known as “cookie law”) further applies to how third parties obtain users’ consent to access stored information on the consumers’ devices, but this is outside our scope.

2.2.1 General Data Protection Regulation (EU GDPR)

In the European Union (EU) and the European Economic Area (EEA), data protection law aims to protect natural persons’ fundamental rights and freedoms regarding personal data processing by public and private organizations. On May 25, 2018, the EU’s General Data Protection Regulation (EU GDPR) mandated a legal justification for the processing of personal data of all European residents (**data subjects**), which applies to all companies processing data from the EU residents regardless of where they are located [135]. In particular, the GDPR grants individuals critical data protection rights, namely the right of access by data subjects and the right to rectification and erasure their data.

2.2.1.1 Scope, Controllers and Processors

This regulation protects the fundamental rights and freedoms of any individuals in the EU and EEA, particularly their right to protect personal data that is processed by a controller or processor. Under GDPR’s Article 4, **controller** means the natural or legal person, public authority, agency, or other body that, alone or jointly with others, determines the purposes and means of the processing of personal data. In contrast, **processors** means any entities that process data on behalf of the controller.

For example, app developers (first parties) process user data in order to provide the app’s functionalities and services. By deciding on the means and purposes for processing the user’s personal data, they act as **data controllers**, the legal role that is

the responsible party for data processing. Parties external to this app developer (third parties) that also receive the user’s data could act in two possible capacities. If they act purely on behalf of the first party with no data use for their own purposes and under the complete control of the first party (e.g., error logging), they act as **data processors**. If they use the user’s data for their own purposes and gains, i.e., in order to do market research, create and monetize user profiles across customers or improve their services, and are not controlled by the first party, they act as data controllers.

2.2.1.2 Definition of Personal Data

Under GDPR’s Article 4 [60], “**personal data**” means any information relating to an identified or identifiable natural person (“**data subject**”), i.e., an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person. This definition includes unique identification numbers, which may include Advertising IDs, and online identifiers (such as IP addresses) — when they can be used to identify users over a long period across different apps and services [4].

The definition of personal data under the GDPR is much broader than personal identifiable data (PII) under US laws. Instead of only including directly identifying data, GDPR also considers personal data such data that can be used alone or in combination to single out an individual in a data set.

The EU Court of Justice has already declared that even dynamic IP addresses may be considered personal data in its Breyer v. Germany ruling [25]. Android’s Advertising ID (AAID) is an interesting subject for the courts, which lacks a ruling as of yet. Google describes the ID as “*a unique, user-resettable ID for advertising, provided by Google Play services. [...] It enables users to reset their identifier or opt-out of personalized ads*” [63]. While Google itself remained vague on characterisation of the AAID as personal data, the IAB Europe GDPR Implementation Working Group already established in their 2017 Working Paper on personal data that “*Cookies and other device and online identifiers (IP addresses, IDFA, AAID, etc.) are explicitly called out as examples of personal data under the GDPR*” [73]. In May 2020, [116], a European not-for-profit privacy advocacy group, lodged a formal complaint over the AAID with Austria’s data protection authority. The complaint states that although the AAID is personal data, Google does not adhere to the requirements of valid consent. Android users have no option to deactivate or delete the tracking ID, only to reset it to a new one. Furthermore, even Google’s own brand Admob explicitly lists the AAID as personal data in their documentation about the User Messaging Platform used to deliver their ads [71]. Meanwhile, Apple has recently taken actions for mandatory prior consent for sharing of Advertising Identifiers for its iOS 14 update [7], clarifying that even dynamic advertising identifiers are considered personal data.

2.2.1.3 Legal Basis for Processing of Personal Data

GDPR Article 6 [58] contains the six general justifications for collecting and processing users' personal data. In particular, the processing shall be lawful only if and to the extent that at least one of the following applies:

- the data subject has given consent to the processing of their personal data for one or more specific purposes;
- processing is necessary for the performance of a contract;
- processing is necessary for compliance with a legal obligation;
- processing is necessary in order to protect the vital interests of the data subject;
- processing is necessary for the performance of a task carried out in the public interest or in the exercise of official authority;
- processing is necessary for the purposes of the legitimate interests pursued by the controller, except where such interests are overridden by the interests or fundamental rights and freedoms of the data subject, in particular where the data subject is a child.

2.2.2 GDPR Consent

In order to be legally compliant, consent must meet specific conditions. In particular, GDPR requires the consent for processing users' personal data to be *freely given, specific, informed, and unambiguous*. Further, the users must have given consent through a statement or by a clear *affirmative* action prior to the data processing [60, 59].

2.2.2.1 Freely Given and Specific

To satisfy the condition of being freely given, the controller should allow separate consent to be given to different personal data processing operations [126]. As such, when assessing whether consent is freely given, utmost account shall be taken of whether, among other things, the performance of a contract, including the provision of a service, is conditional on consent to the processing of personal data that is not necessary for the performance of that contract [GDPR Art. 7(4)].

In particular, the GDPR Art. 7(2) states that: “*If the data subject’s consent is given in the context of a written declaration which also concerns other matters, the request for consent shall be presented in a manner which is clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language*”. That is, personal data transfer must only occur after the user has actively agreed (e.g., by clicking accept), i.e., “consent” packaged in terms and conditions or privacy policies is not compliant [54].

2.2.2.2 Informed and Unambiguous

The GDPR further requires that consent must be informed and unambiguous, as laid out in Article 12 of the GDPR: “*The controller shall take appropriate measures to provide any information [...] relating to processing to the data subject in a concise, transparent, intelligible and easily accessible form, using clear and plain language, in particular for any information addressed specifically to a child*”.

2.2.2.3 Explicit

Article 9(2)(a) GDPR mandates that consent with regard to the processing of personal data has to be explicit. As such, the controller should obtain verbal or written confirmation about the specific processing [Recital 32 of the GDPR]. Further, consent cannot be based on an opt-out mechanism, as the failure to opt-out is not a clear affirmative action [126], e.g., data subjects have to withdraw their by default opted-in consent by turning off personalized ads through their device settings.

2.2.3 Summary

In practice, most advertising companies rely on consent or legitimate interests as the legal basis for processing users’ personal data for profiling and targeted advertising (i.e., since the legal ground necessary for the performance of a contract does not apply in these circumstances [19, 54]). However, the European Data Protection Board (EDPB) and many legal studies state that it seems unlikely these companies’ legitimate interests may claim to outweigh the fundamental rights and freedoms of the data subject and the legitimate interests grounding is not considered to be an appropriate lawful basis for the processing of personal data [54, P2, 57, 42]. Consequently, such companies have to rely on consent as the legal basis for their processing operations. In case the processing is based on consent, the GDPR requires consent to be *freely given, specific, informed, and unambiguous* (GDPR Art. 7 [59]). Further, the data subject (which is the user) must have given consent through a statement or by a clear affirmative action (GDPR Art. 4(11) [60]).

Our research focuses explicitly on these aspects of GDPR consent requirements. In particular, with respect to the regulations mentioned above, transmitting users’ personal data to a third-party data controller without *freely given, specific, informed, and unambiguous* consent for the purpose of targeted advertisement is considered violating GDPR.

2.3 Related Work

This section provides information about the areas of research that relate to the main topics discussed in this work. First, we briefly discuss prior work in detecting unexpected data access in Android apps. Next, we discuss research on studying the legislation violations of online services.

2.3.1 Detecting Data Access that Violates User Expectations

Many researchers have worked to detect sensitive resource accesses that violate user expectations, by matching *user expectations* and *apps' actual behaviors*. While app behavior can be inferred by analyzing the app code, user expectation is much more difficult to measure. Most existing works infer user expectation using certain proxies such as app descriptions (*i.e.*, users would expect apps to behave the way described on the description page) [124, 132, 72].

In particular, a related line of work aims to detect app behaviors that deviate from the app descriptions. The idea is to use app description as a proxy for user-expected behavior. Prior work by Pandita et al. [124], and Qu et al. [132] have used Natural Language Processing to extract semantic meaning from app description to identify inappropriate required permissions in the app's manifest file. On the other hand, Gorlat et al. proposed CHABADA to cluster app description and app's bytecode to detect abnormal used permission-protected APIs [72]. Follow-up, Yu et al. developed TAPVerifier to detect malware based on app description, app's privacy policy, and app's bytecode [174]. They showed that relying on the app description alone is not sufficient and could lead to large errors.

More recently, researchers have started to use app UIs as a proxy to study user expectations. For example, AsDroid was proposed to detect malicious apps by detecting the mismatches between advertised permission usage in the text of UI elements and its accessed permissions [80]. However, AsDroid only focuses on a small fixed set of permissions (*i.e.*, SMS, PHONE, HTTP connections and component installations). A follow-up work by Avdiienko et al. is an extension of AsDroid that detects arbitrary mismatches between textual UIs and the accessed permissions using text clustering [11].

Unfortunately, icons, as one of the most important UI elements for user interaction [115, 56], have been neglected in most research so far. A key reason is that it is difficult to map graphical icons to user expectations without actually performing user studies. Recently, Xiao et al. built Icontent to classify app icons using Google Image and hand labeled data as the ground truth for permission access of icons [168]. Besides, DeepIntent used icons as part of their features to detect malware [167], which demonstrated the benefits of using icons. However, Icontent did not measure end-users' expectations of icons, in stead, they used names of permission groups to search for images on Google Image to create training dataset, whereas DeepIntent used benign apps' icon-to-permission association as the "norm", and aimed to detect malicious apps that deviate from the norm.

2.3.2 Studying the Legislation Violations of Online Services

Researchers are actively and continuously studying the legislation violations of online services after GDPR went into effect in May 2018. Among others, existing works have extensively studied the cookie consent compliance on the Web.

2.3.2.1 Legislation Violations on The Web

In recent years, many researchers have started to study the impact of GDPR on the online advertising and tracking industry and proposed different techniques to detect legislation violations. A related line of work aims to study the consent notices in the Web ecosystem, which are usually presented in cookie banners. Researchers have shown that many websites potentially violate the GDPR consent requirements, such not allowing users to refuse data collection or installing tracking and profiling cookies before the user gives explicit consent [152, 140, 151, 36, 90, 156, 103].

In particular, Kampanos et al. [83] conducted a large-scale study of more than 17,000 websites, including more than 7,500 cookie banners in Greece and the UK. They show that most websites in the UK and Greece lack cookie consent notices, i.e., only roughly 45% have a cookie notice. Trevisan et al. [152] run a large-scale measurement of the EU cookie directive. The result is a shady picture in which 49% of websites do not respect the cookie laws, i.e., install cookies before any user's consent is given. Iskander et al. [140] further show a large number of websites present deceiving cookie information, making it very difficult for users to avoid being tracked. Many studies further have shown that a lot of websites do potentially violate the GDPR consent requirements, such as do not allow users to refuse data collection, installing tracking and profiling cookies before the user gives explicit consent [36, 90, 156].

Regarding the cookie consent interface, Matte et al. [103] perform the first study to compare the interface of the cookie notices shown to the users to the website behaviors. Specifically, they systematically studied IAB Europe's TCF and analyzed consent stored behind the user interface of TCF cookie banners of 1,426 European websites that have TCF banners. They find that most websites, by default, register positive consent even if the user has not made their choice and nudge the users towards accepting consent by pre-selecting options.

On the other hand, Utz et al. [155] inspected how the design of consent popups from websites nudge users into uninformed consent by conducting a study with more than 80,000 unique users on a German website. The results also show that the widespread practice of nudging greatly affects the choices users make.

2.3.2.2 Legislation Violations on Mobile Apps

While many researchers have worked to detect and analyze consent notices and their impact on the Web advertising and tracking industry after the GDPR went into effect, no study has measured the GDPR violations of explicit consent on mobile apps. For mobile apps, researchers mostly focused on analyzing the app privacy policies to identify legislation violations, i.e., determining whether an app's behavior is consistent with what is declared in the app privacy policy [4, 142, 173, 178].

Researchers have proposed different techniques to detect privacy violations by mobile apps and identify third-party advertising and tracking services. Many techniques have relied on the static program analysis of app binary code to detect malicious behaviors and privacy leaks [8, 113, 123, 9] as well as third-party library use [13, 101, 95]. While the static analysis techniques are well known for producing high false positives (e.g., do not produce actual measurements of privacy violations) [158, 93, 22], the dynamic

analysis shows precisely how the app and system behave during the test (i.e., by running the app and auditing its runtime behavior) [17, 137, 166, 172]. However, an effective dynamic analysis requires building an instrumentation framework for possible behaviors of interest, which involves extensive engineering effort [134].

Additionally, another line of work aims to analyze the app privacy policies or privacy labels to identify potential GDPR violations, i.e., determining whether an actual app's behavior is consistent with what is declared in the app privacy policy or privacy label [4, 142, 173, 178, 85]. Researchers have developed different techniques to detect privacy violations in mobile apps and to identify third-party advertising and tracking services using static analysis [8, 113, 123, 9] or dynamic analysis [17, 137, 166, 172]. On the other hand, Reyes et al. [137] presented the first insights into mobile apps' compliance with data protection law, focusing on the Children's Online Privacy Protection Act (COPPA) in the USA. Particularly, the authors performed the first large-scale analysis of COPPA compliance in the Android market using dynamic analysis.

However, little research has been done to measure the GDPR violations of consent on mobile apps. Recently, Kollnig et al. [87] tried to study the absence of consent notices to third-party tracking from a small set of Android apps (i.e., 1,297 apps) by manually inspecting each app — which is insufficient in identifying and studying the status quo of currently implemented consent notices in Android apps. Until now, there have been various ways that consent could be obtained in mobile apps. However, while existing works [87] mainly focus on analyzing apps' network traffic to detect GDPR violations, little or no research has systematically studied how the consent notices are currently implemented in mobile apps and whether they are legally obtained under GDPR.

2.3.3 Summary

Different from prior work, our first goal was to build a semantic mapping between user expectations of sensitive resource accesses and apps' icon UI elements via extensive user studies. Our results show that even benign apps can have misleading UIs that lead to violations of users' expectations. More importantly, it shows that relying on benign apps' icon-to-permission association as the "norm" to detect behavior discrepancy is not enough. Second, we perform the first large-scale study into consent notices of third-party tracking in Android apps in the wild to understand the current practices and the current state of GDPR's consent violations. While prior studies primarily focused on network traffic analysis or manually studied a small set of samples, we semi-automatically analyzed a large scale of apps' consent user interfaces to investigate whether they are *freely given*, *specific*, *informed*, and *unambiguous* with respect to GDPR consent requirements.

3

Automatic Detection of Unexpected Data Access

Most existing works infer user expectation using certain proxies such as app descriptions (i.e., users would expect apps to behave the way described on the description page) [124, 132, 72]. Unfortunately, app descriptions are too coarse-grained, and not all users would read app descriptions in practice. More recently, researchers have started to look into another source of information that directly shapes user expectations: user interface (UI) elements (e.g., texts and images) [11, 80, 168, 167]. However, while text descriptions are self-explanatory with natural language meanings, visual images (referred to as “*icon*”) often convey information in non-verbal ways, and the end-users interpret the meaning of an icon using their pre-existing experience and knowledge [165, 82]. Existing works either only look at *textual* UI elements to examine user expectations [11, 80], or try to predict the intention of apps’ icon UI elements but without understanding how real-world users perceive data access when they interact with such graphical UI(s) [168, 167] — which we found less effective in detecting data access that violates user expectations.

In this chapter, we fill this research gap by directly analyzing user expectations and making them measurable. Particularly, we aim to take into account the influence of all UI elements (*icons* and *texts*) on users’ expectations to detect unexpected data access. We first conduct two user surveys to understand how end-users perceive graphical UI(s), and what users’ expectations are (Section 3.1). The goal is to map user expectations of data access and apps’ icon UI elements. Then, we build a new tool called GUIBAT that learns from users’ perception to identify unexpected data accesses (Section 3.2). More specifically, we train a classifier using the study results to infer users’ expectations of icons on the app’s UI. For text elements, we follow similar Natural Language Processing approaches used in prior works to examine user expectation [80, 112, 161] since text elements are self-explanatory with natural language meanings. Next, we leverage static program analysis to find the app’s actual data access, which is considered *unexpected* if it deviates from user expectations. To this end, our validation of GUIBAT’s effectiveness shows that it significantly outperforms prior works and accurately reflects users’ expectations (Section 3.3). Given that, we apply GUIBAT on 100,000 Android apps to answer RQ1: *How widespread is sensitive resource access that violates users’ expectations in the wild?* (Section 3.4).

3.1 Measuring Users’ Perception

The text descriptions are self-explanatory with natural language meanings, making it possible to directly infer user expectations. However, visual images, i.e., icons, usually convey information in non-verbal ways, and users interpret the meaning of an icon using their pre-existing experience and knowledge [165, 82]. Therefore, in this step, we aimed to build a comprehensive mapping between app icons and their associated sensitive resource accesses in mobile apps. Figure 3.1 gives an overview of our approach. We first developed a crawler to obtain a large number of Android apps, and extracted the icon UI elements from their apps’ UI (Section 3.1.1). Next, we conducted a series of studies to measure how real-world users perceive an app’s data access when they interact with

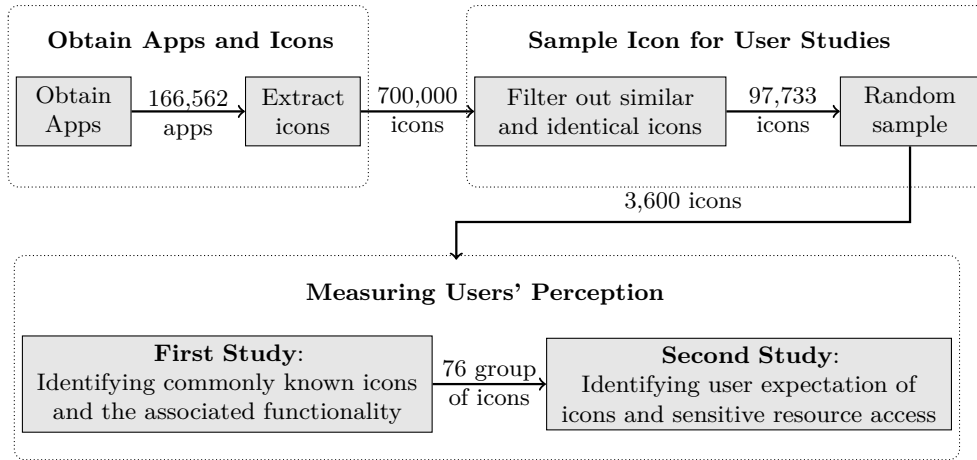


Figure 3.1: Overview of the methodologies to measure user’s perception.

such apps’ icons¹. More specifically, we divided our studies into two parts: one is to identify *commonly-known* icons (e.g., icons are frequently used) to users (Section 3.1.2), since it is practically impossible to study all available icons of apps with users because icons are extremely diverse in appearance, and not all icons are supposed to convey meaning to users [28]; the other is to measure user expectation of sensitive resource accesses of these icons (Section 3.1.3). Once we had established the expectations of end-users between the app’s icons and sensitive resource accesses, we leveraged this knowledge to build GUIBAT which we describe in Section 3.2.

3.1.1 App and Icon Dataset

App Dataset We built a crawler to crawl Android apps from the Google Play store and successfully obtained about 600,000 apps, i.e., randomly selected based on the list of apps from AndroZoo [3]. Among them, we only selected apps that request dangerous permissions because they access users’ sensitive data. Further, we used the permission mappings of PScout [10] and Axplorer [14] to identify accessed sensitive resources of a given Android app. As the Android’s permission documentation is incomplete, the combination of PScout and Axplorer provides a complete mapping from Android version 2.2 up to 7.1. We only selected apps that declare at least one dangerous permission and filtered out apps that are not supported by PScout and Axplorer based on the *min* and *max* SDK versions (min/max platform version to which the app is compatible). Besides, apps that belong to game categories were excluded since they have little to no UI elements or their UI is mainly made of drawings on canvases, which are out of the scope for this work. After filtering, our app dataset contains 166,562 apps.

¹We compensated participants based on an average hourly wage of \$14 (above US minimum wage [89]). More importantly, all user studies in this work were approved by the ethical review board of our university. Web access to the server was secured with an SSL certificate issued by the university’s computing center, and all further access was restricted to the department’s intranet and only made available to maintainers and collaborating researchers. Participants could leave the studies at any time.

Name	Range	Notes
Zoom	0-0.3	Float range for random zoom
Rotation	0-30	Degree range for random rotations
Width shift	0-0.2	Float range for random horizontal shift
Height shift	0-0.2	Float range for random vertical shift

Table 3.1: The transformation techniques that are used.

Icon Dataset From the 166,562 apps, we then leveraged existing works [11, 55] to perform static analysis on both the UI layout files and the app’s code to extract apps’ icons (see Section 3.2.1 on implementation details). This resulted in about 700,000 icons. To get an overview of these icons, we then used perceptual hashing [175] for filtering out identical and very similar icons at pixel levels.

In particular, to identify whether two icons are identical or very similar, *pHash* technique was used [175], and a *hamming distance* value of less than 11 between 2 icons is considered as similar. To select this threshold, we first randomly selected 1,000 dissimilar icons (i.e., 1,000 categories) by comparing their exact *pHash* value. For each category, we applied four transformations techniques (see Table 3.1) to generate more samples, resulted in 8,400 icons across 1,000 categories. Then, we clustered these icons into the same group with different hamming distance thresholds (ranging from 0 to 50). For example, with the threshold is 5, two icons are considered in the same group if their hamming distance is less than or equal 5. To select the best threshold for identifying very similar icons, we considered at the tradeoff between the quality of the clustering against the number of clusters. A measure that allows us to make this tradeoff is normalized mutual information or NMI (i.e., a number between 0 and 1), and a higher value is better. After running this evaluation 10 trials, we chose the value of 11 for the hamming distance threshold as the NMI’s score reached its peak at 0.88. As a result, we identified 97,733 dissimilar icons.

3.1.2 First Study: Identifying Commonly-Known Icons and The Associated Functionality

We conducted this study online via Amazon Mechanical Turk (MTurk)² to determine which icons that are commonly-known and familiar to users based on population stereotype [28] (e.g., an icon known by most of the participants is considered commonly-known). We randomly selected 3,600 icons from our set of dissimilar icons, and asked participants about their subjective feelings, if they have any concrete expectation(s) of a given icon or a similar looking icon (i.e., “Do you have a concrete expectation what could happen if you press this icon or a similar symbol?”). The participant’s responses are binary: “yes” and “no”. To identify the careless respondents, we used the instructed response items which is the most popular form of attention check [106, 159], for example, items are embedded with an obvious correct answer (e.g., “please select yes”). The survey was designed to ask each participant 90 icons and 10 more attention check icons. This resulted in 40 batches. For each batch, we tested with three different participants

²<https://www.mturk.com/>

to examine whether an icon is commonly-known. If a participant failed attention checks, the corresponding batch would be re-conducted with another participant. An icon was considered commonly-known to users if all participants know it (i.e., 100% recognizable).

The survey lasted for 2 days and we collected valid responses from 120 Turkers. Their median age is 31.5 years (65% male and 35% female, see Table A.1 in Appendix for full demographics). We identified 972 commonly-known icons and found that the majority of surveyed icons (73%) are not linked to any concrete expectations of users and therefore have no intrinsic meaning to them.

Identifying Associated Functionality We then conducted a follow-up survey to find the perceived function an icon represents. This helps us not only group icons based on their functionality but also quickly filter out icons that do not represent any functionality (given that concrete expectations can only be formed if users are somehow familiar with an object [105]). This would significantly reduce the number of icons we need to study later. To do that, we used the icon intuitiveness test to learn about users' pre-existing knowledge of known and familiar icons [114]. An icon was shown to a group of participants without contexts (e.g., without textual description). We asked participants to describe their understanding of a given icon, and their expectations of what would happen if they interact with it (i.e., *Q1: What does this icon symbolize?, Q2: What you would expect to happen if you interact with this icon or an icon that has a similar symbol?*). In this survey, we showed each participant 10 icons, and for each icon, we also asked three different participants to describe their interpretations (free-text responses).

The survey involved 294 participants and lasted 8 days. Participants' median age is 34 years (58.16% male, 41.50% female, see Table A.1 in Appendix for full demographics). We obtained 2,916 feedbacks for 972 commonly-known icons. After extracting users' responses, we constructed a mapping between commonly-known icons and the function they represent by performing description-based icons clustering.

In particular, we first applied the following widely-used text preprocessing techniques [78, 161, 100]: correcting misspelling by the *autocorrect*³ (e.g., “*imagec*” to “*image*”); normalizing and lemmatizing all words, e.g., removing punctuations, converting letters to lowercase and reducing the inflectional forms of a word (e.g., “*sent*”, and “*sending*” to “*send*”); and removing generic stop words such as “*are*” and “*the*”; lastly we employed the term-based sampling approach [100] to create our domain-specific stop words list (taking top 50 words that have the least weighted) so that we could additionally remove words that are not generic stop words but are commonly used in user responses from our survey (e.g., “*icon*”, “*symbol*”, “*button*”).

Then, using the preprocessed texts of user responses, we adopted the bag-of-words model to convert each user response into a text feature vector as follows. Let $T = \{t_1, t_2, \dots, t_n\}$ be a set of all unique terms in the corpus of user responses. A text feature of vector of the i^{th} user response is denoted as $t_i = \{t_1, t_2, \dots, t_k\}$. For example, the raw text is “*I would expect it to take a photo*”, after applying the preprocessing, the generated preliminary text vector is $\{photo\}$. Further, to resolve the problem of synonyms, we employed the hypernym strategy (*hypdepth* = 5), which is suggested

³<https://github.com/phantpiglet/autocorrect/>

by Hotho et al. [78] to enrich the text vectors with concepts from the Wordnet [108]. Specifically, we added to each term of the feature vectors all subconcepts of the 5 levels below it in the Wordnet corpus, after that, we performed word stemming on all terms (e.g., “location” and “locating” to “locat”). For instance, with the preliminary text vector above, the final text feature vector will be $\{photo, photograph, exposur, pictur, pic\}$. To this end, we converted each term, in the text vector to numeric one by using the term-frequency inverse document-frequency (TF-IDF). The TF-IDF value for each element was calculated as:

$$tfidf(t, d) = tf(t, d) * idf(t) = \frac{1 + N}{1 + df(d, t)}$$

where t refers to the selected term, d refers to the text vector, tf is the absolute frequency of a term, i.e., $tf(t, d)$ is the number of times a term t occurs in a given d , idf is the term’s inverse document frequency, N is the number of text lists in the corpus, and $df(d, t)$ returns the number of text lists that contain the target term t . Lastly, the agglomerative hierarchical clustering was used to identify which icons are commonly known by users and the function these icons represent. Specifically, Ward’s method was used [160], and the similarity score or distance between two vectors is calculated by cosine similarity:

$$similarity(\vec{v}_i, \vec{v}_j) = \cos(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{\|\vec{v}_i\| \cdot \|\vec{v}_j\|}$$

To this end, we first clustered similar icons using textual descriptions of their visual representation, since apps’ icons are extremely diverse in appearance (e.g., icons representing the same camera object may look very different). We then further clustered these clusters of icons using descriptions their functions. In the first step, we identified 77 groups of icons from 972 commonly-known icons. However, we filtered out one group that contains icons whose (participant) responses were not comprehensible (i.e., this group was also considered an outlier by hierarchical clustering). Finally, we identified 44 groups of functions from 76 groups of icons. Our results showed that although icons are extremely diverse in appearance, users will have the same expectations about the associated functionality if they represent the same object. This suggests that by focusing on the most commonly-known icons, our technique could cover most of the icons on which users have perception.

3.1.3 Second Study: Identifying User Expectation of Icons and Sensitive Resource Access

We had at this point a set of commonly-known icons and their associated functionalities. Our studies’ final step was to build a semantic mapping between user expectations of sensitive resource accesses and the apps’ icons. From 76 clusters of commonly-known icons, we randomly selected 3 icons per cluster to conduct this study. Our survey first provided the participants with an instruction page that gave a detailed explanation of each sensitive resource to minimize technical terms. We further asked participants an attention and comprehension check question to see if the participants understand these sensitive resources (see Appendix A.1.1 for details).


























Icon group	Icon group	Icon group
 Calendar (1.0)	 Calendar (0.89)	 Storage (0.88)
 Contacts (1.0)	 Contacts (0.88)	 Storage (0.75)
 Contacts (0.67)	 SMS (0.67)	 Storage (0.75)
 SMS (0.67)	 Camera (1.0)	 Storage (0.75)
 Camera (1.0)	 Storage (0.67)	 Microphone (0.67)
 Storage (0.75)	 Location (1.0)	 Storage (0.67)
 Location (1.0)	 Phone (1.0)	 Storage (0.67)
 Microphone (1.0)	 Storage (1.0)	
 SMS (0.75)	 Camera (0.86)	

Table 3.2: Icon groups and user expectation of sensitive resource accesses . An icon group has multiple similar looking icons. The agreement rate is in parenthesis.

Afterward, we provided multiple choice answers, where the participants could choose the sensitive resources that an icon could associate with (*i.e.*, “Which of the following sensitive resources that you would expect this app to access to perform the function it represents?”). Each of the sensitive resources was accompanied by a corresponding explanation taken from the Android system to leverage user prior experience (see Appendix A.1.1 for explanations of each sensitive resource). Participants could choose “NONE” if no sensitive resource is expected. Also, we asked participants to explain their answers (*i.e.*, “Why would you think the above selected sensitive resources are needed?”). This question helps us to see if participants provide meaningful responses. To minimize the bias introduced by a single participant for each icon, we asked three different participants. The survey involved 45 participants and lasted for 3 days. Participants’ median age is 33 years (68.89% male and 31.11% female, see Table A.1 in Appendix for full demographics). Finally, we got 684 feedback on the commonly-known icons and their related sensitive resources.

To build a comprehensive mapping between app icons and their associated sensitive resource accesses, we based on the majority of votes for the associated sensitive resources among the participants. Specifically, for each cluster of icons, we calculated the agreement rate on an associated sensitive resource (R_i) by the number of votes for R_i divided by the total number of votes. We considered a cluster of icons relating to R_i if its agreement rate was at least 0.66 which meets the standard of the icon recognition ISO 3864 [1] (*i.e.*, 66.7% for signs). Details of the icons and their agreement rates are listed in Table 3.2. Specifically, we identified 20 groups of icons that associate with sensitive resources from users’ perspective. In this study, we found that one specific icon could be associated with more than one sensitive resource from users’ perspective. For example, the icons that represent a camera (see Table 3.2) can be associated with CAMERA and STORAGE sensitive resources (e.g., explained by a participant “It’s an icon of a camera so it’d have to access your camera to work, and I think it’d have access to your storage to store the photos you take with the app.”). This suggests that approaches only consider

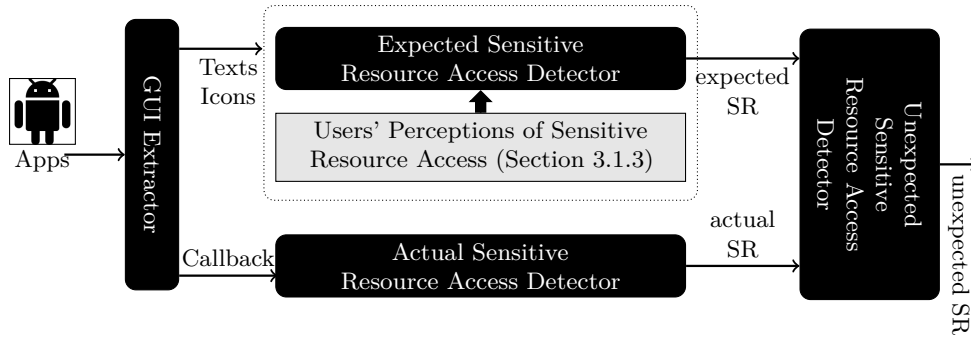


Figure 3.2: Overview of the GUIBAT’s architecture.

one-to-one mapping between icons and sensitive resources might not correctly reflect users’ expectations (e.g., `IconIntent` [168]).

3.2 Detecting Unexpected Sensitive Resource Access

So far, we have conducted two studies to establish a semantic mapping between common apps’ icons and user expectation of apps’ sensitive resource accesses. Recall that our goal is to have a fully automatic and scalable solution to detect *unexpected sensitive resource accesses* based on user perception. We now leveraged the knowledge gained from the two user studies to build GUIBAT that estimates user expectation to detect *unexpected sensitive resource accesses* in Android apps. Figure 3.2 presents the architecture of GUIBAT. For a given app, *GUI Extractor* first extracts all the app’s UI elements (e.g., buttons, images), and their associated callbacks which handle users interaction with the app’s UI (Section 3.2.1). Then, using our study results in Section 3.1.3, the *Expected Sensitive Resource Access Detector* utilizes a classifier built on top of Natural Language Processing and Image Recognition techniques to identify the app’s expected sensitive resource accesses based on the texts and icons of the extracted UI elements (Section 3.2.2), which we refer as *expected SR*. On the other hand, for the corresponding extracted UI element, the *Actual Sensitive Resource Access Detector* leverages Static Control-flow Analysis techniques to detect the app’s actual sensitive resource accesses from sequences of callbacks by looking for permission-protected API calls (Section 3.2.3), which we refer as *actual SR*. An *actual SR* will be considered unexpected (referred to as *unexpected SR*) by the *Unexpected Sensitive Resource Access Detector* (Section 3.2.4) if it is not contained in the *expected SR*.

In this section, we describe each component in details. We then present how we designed and conducted the third user study to validate GUIBAT’s results in Section 3.3.1.

3.2.1 GUI Extractor

This component aimed to identify the app’s UI elements and their associated callbacks and to extract the associated graphical content (i.e., texts and icons) that comprise an app’s UI. Prior works have proposed different approaches to analyze the app’s GUI

(e.g., GATOR [138, 171], Backstage [11]). Among them, GATOR is a widely-used static analysis toolkit for Android that analyzes an app’s UI by providing an over-approximation algorithm to infer the relationship between app’s callbacks and app’s UI elements [167, 168]. Further, GATOR identifies both static layout files and dynamically generated UI components. Moreover, the authors showed that it achieved good precision, takes less computational memory. Therefore, to discover the apps’ UI elements and their callbacks, we extended GATOR with some improvements (i.e., better coverage and higher precision) and further overcame the following limitations:

- (1) GATOR does not consider some important Android UI components which leads to missing app’s UI elements along with their associated callbacks in app’s GUI model (*Navigation Drawer*, *Preference*, and *Fragment*). Richard et al. showed that 91% of popular apps contain fragment code in their apps which indicates that *Fragment* is widely used [21]. GUIBAT additionally considers these UI components.
- (2) GATOR’s static reference analysis that maps app’s UI elements to their callbacks introduce over-approximation due to the over-approximate of variable mappings (e.g., incorrectly map a UI element to multiple callbacks). To be more precise, GUIBAT, on the other hand, further strictly on object id (of UI elements) to correctly map callback methods to UI elements.

Second, to extract texts and icons, we followed similar approaches used in prior works [11, 167]. Specifically, *GUI Extractor* first statically parses the app’s UI layout file⁴ (i.e., layout files are in XML to define app’s UI), and then uses the XML text-related attributes (e.g., *android:text*, *android:title*) to extract UI texts, and image-related attributes (e.g., *android:drawable*, *android:icon*) to extract UI icons. Further, by analyzing app’s bytecode, *GUI Extractor* extracts UI texts and icons that are dynamically created at runtime by identifying the used of text-related APIs (e.g., *setText()*, *setTooltipText()*), or image-related APIs (e.g., *setImageResource()*, *setIcon()*). *GUI Extractor* further extracts additional information that are shown to users after they press on the app’s UI elements. It first finds all the instantiations of *Toast*, *Snackbar*, *Dialog* and *AlertDialog* objects which allow an app to send feedback messages to end-users. Then it extracts the text message based on the used API (e.g., *Toast.makeText()*). For example, clicking on a button shows the “*Start recording*” text.

Comparing GUIBAT and GATOR We selected an Android benchmark suite from the work of Choudhary et al. [30] which is widely used by other static and dynamic analysis [102, 110]. It contains 68 open source Android apps. We filtered out those apps that are not supported by GUIBAT such as apps in the games category, which resulted in 43 apps. Then we ran GUIBAT and GATOR on 43 apps to generate app’s GUI models. Among these 43 apps, those apps that have less than 200 callbacks were chosen for a comprehensive manual examination, resulted in 34 apps. From 34 apps, we compare the number of extracted UI elements, and the number of extracted callback methods

⁴<https://developer.android.com/guide/topics/ui/declaring-layout>

3.2. DETECTING UNEXPECTED SENSITIVE RESOURCE ACCESS

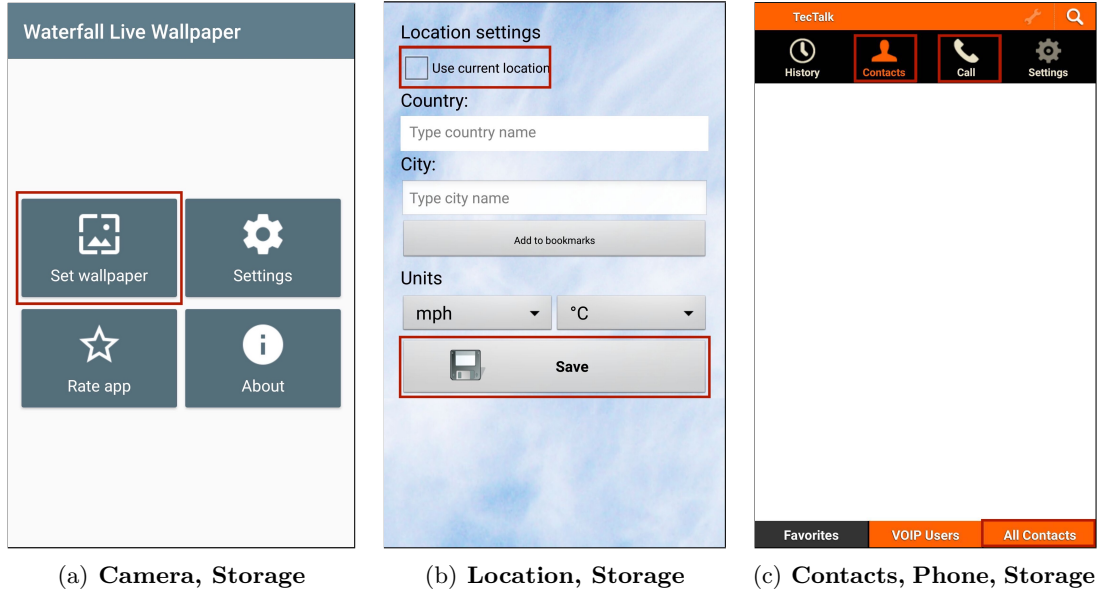


Figure 3.3: Example of expected sensitive resource accesses.

between GATOR and GUIBAT. If there are any differences between the results of GUIBAT and GATOR, we manually verify the differences (see Table A.2 in Appendix). In general, GUIBAT is better than GATOR in GUI analysis as the following: (1) in 14 (41.17%) apps, both GUIBAT and GATOR have identified the same number of UI elements and their callbacks; (2) GUIBAT correctly has removed 231 (21.81% of 1,059) of the callback methods that are made-up by the over approximation of GATOR; (3) GUIBAT successfully identified the Preference UI component and their associated callbacks in 20 apps which GATOR missed. In particular, there are 172 (15.02% of 1,145) components that GUIBAT identified while GATOR could not.

3.2.2 Expected Sensitive Resource Access Detector

After having all the extracted UI elements of a given app, we want to automatically detect if they represent any sensitive resource accesses from users' perspective. Our idea is based on the intuition that an app's UI (i.e., presented as texts and icons) depicts the user's expectation of app's behavior [80]. More importantly, we can not treat a UI element as a stand-alone element on a screen as an app activity's UI may be comprised of different UI elements, and their *expected SR* can complement each other. Therefore, from each extracted UI element, GUIBAT's *Expected Sensitive Resource Access Detector* first uses *GUI Extractor* to extract the contextual information (the texts and icons of the surrounding UI elements) where the UI element is represented. Then, it performs icons classification, in combination with the acquired user perception (Section 3.1) to automatically identify the *expected SR* of icons. Additionally, we also leverage prior works to identify *expected SR* in texts [80, 112, 161].

Examples of *expected SR* (according to our user studies) are depicted in Figure 3.3.

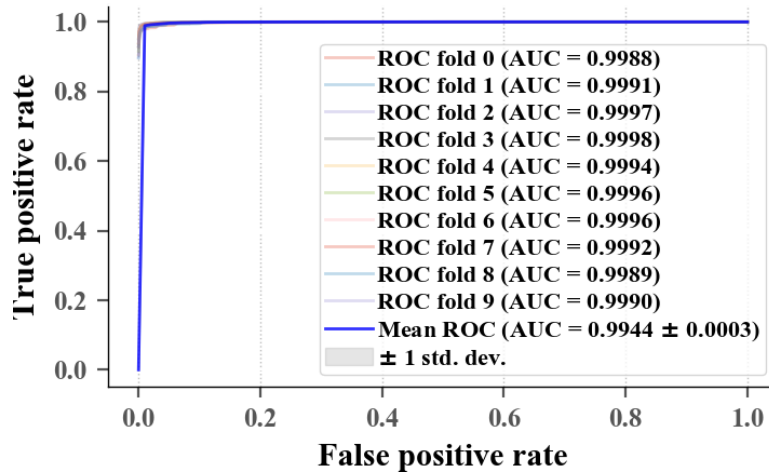


Figure 3.4: ROC curves of the 10-Fold cross-validation.

In Figure 3.3(a) the *expected SR* is accessing CAMERA, STORAGE represented by a gallery icon (see Table 3.2). In Figure 3.3(b) the *expected SRs* are LOCATION represented by the *Use current location* text (see Table 3.3), and STORAGE represented by the *Save* text with a floppy disk icon. Finally, in Figure 3.3(c) the *expected SRs* are CONTACTS represented by *Contacts* text with the human icon, and PHONE represented by *Call* text with the phone icon.

3.2.2.1 Icon Classifier

This component aimed to identify user expected sensitive resource accesses based on apps’ icons. To this end, we used Convolutional Neural Networks (CNNs), which is a deep learning model that achieves state-of-the-art results in image recognition challenges, and widely used in image classification tasks [75]. We abstained from using perceptual hashing [175] because it can only identify identical or very similar looking images, while our goal is to maximize the robustness of GUIBAT in identifying 20 groups of icons in Table 3.2, a more challenging classification task.

Our *Icon Classifier* is a multi-class CNN classifier. Given an input (i.e., an icon), the Classifier produces the probability of this icon associating with the 20 groups of icons in Table 3.2. To determine whether a given icon is associated with sensitive resources (from user perspectives), we first use the *Icon Classifier* to predict the probabilities for 20 groups of icons. If the icon’s prediction result has the highest confidence probability at group_{*i*}, and also its probability is higher than 0.99, we then consider that the icon is associated with the corresponding sensitive resource⁵. Our user studies have already collected a labeled dataset reflecting user-perceived association between icons and sensitive resource accesses. As such, we expect the Classifier to capture user expectations of icons.

⁵We exclude the icons that could not be mapped to any group in Table 3.2 (probability < 0.99). If an icon has the same probability for 2 groups (or more), we consider it to be associated with the corresponding sensitive resources behind these groups.

Constructing Training Dataset For each icon group in Table 3.2, we manually selected 500 similar icons based on the similarity of their visual feature from our icon dataset. The manually labeling method helps us enrich the generalization of the *Icon Classifier*. An icon is considered similar to one of the sensitive related icons in Table 3.2 if it represents the same object (e.g., camera). Each labeled icon was reviewed by two volunteers independently. If there was a disagreement between the two volunteers, we would ask another volunteer to join the discussion. If an agreement could not be reached, we simply excluded that icon. Further, our dataset as any real dataset contains noisy data such as icons with different sizes, or icons with different formats (e.g., RGB, RGBA) which potentially lead to low-quality models. Therefore, the following preprocessing methods were applied to obtain a quality training dataset [88, 167]: (1) resizing icons to 128x128 pixels (which is most common icon size); (2) converting the RGBA to RGB without affecting the image’s content. Finally, our training dataset has 10,000 labeled icons pertaining to sensitive resources across 20 groups of icons.

Training We employed a self-training method to leverage unlabeled data at scale, which has been widely used in image processing [169, 141, 139]. Specifically, we first implemented the SimpleNet architecture, a light-weighted deep CNN architecture that achieves high precision [75]. Then we trained the *SimpleNet* model on the labeled dataset. We used it as a “teacher” to generate pseudo labels on 700,000 unlabeled icons. Subsequently, we trained a larger *SimpleNet* on the combination of labeled and pseudo labeled icons to produce more “student” data. We iterated this process by putting back the “student” into the “teacher” data. During this process, we kept increasing the size of the “student” data to improve performance. This iterative process was stopped when the size of “student” data became stable (i.e., our final training dataset has 46,578 labeled icons across 20 groups of icons).

When putting back the “student” data to the training dataset, our training dataset became imbalanced among some classes which could significantly affect the *classifier*’s performance. Therefore, we first leveraged the Cost Sensitive Learning method in which the weights of each class (calculated based on sample frequencies) are integrated into the cost function [76]. Further, to reduce over-fitting and to build a quality image classifier, we employed data augmentation technique that helps diversitize our training dataset which is randomly zooming into images, and the zoom ranges from 0 to 10% of the original images [29].

Evaluation To validate our “teacher” model, the k-Fold cross validation was selected, together with $k = 10$ as this was shown to be the best method for cross validation by prior work [86]. Further, to evaluate the model, we used the area under the curve (AUC) of the receiver operating characteristic (ROC) [74]. The AUC is the most commonly used for evaluation of imbalanced data classification [27, 129], where the area near up to 1.0 represents a perfect model, and the area of 0.5 represents a random guessing model. Figure 3.4 shows the AUC values for 10-Fold cross validation. Our *classifier* has an AUC’s mean value of 0.9944.

Sensitive Resources	Keywords
Calendar	<i>calendar, calender, event, reminder, meeting, schedule, agenda</i>
Contacts	<i>contact, account, call</i>
Camera	<i>take picture, camera, capture, scan</i>
Location	<i>location, map, gps, track</i>
Microphone	<i>microphone, recording, record, audio, voice, mic</i>
Phone	<i>call, telephone</i>
SMS	<i>sms, mms, send, incoming, voicemail</i>
Storage	<i>storage, sd card, file, save</i>

Table 3.3: Sensitive Related Keywords.

3.2.2.2 Identifying Expected Sensitive Resource Accesses in Texts

This component aimed to identify the user-expected sensitive resource access based on texts of UI elements, since in different UI contexts, similar icons may reflect different intentions [167]. From the extracted texts of UI elements⁶, we followed prior works [112, 161] to rely on keywords to map texts to *expected SR*. Most of the visible texts on UI elements are short (combination of *verbs*, or *nouns*) but they are self-explanatory. Therefore, we first collected security and privacy relevant keywords from existing works [112, 161], then we manually examined the Android documentation⁷ regarding the permission-protected resources to expand the keyword list. The final list is shown in Table 3.3. Given an input text t (i.e., a list of words), GUIBAT first uses a set of Natural Language Processing techniques to preprocess t : normalizing and lemmatizing, removing generic stop words [78, 161, 100]. Then by searching the sensitive related keywords (see Table 3.3) in t , GUIBAT can identify the corresponding *expected SR*.

3.2.3 Actual Sensitive Resource Access Detector

To detect actual sensitive resource accesses of apps' UI elements, we first leveraged GATOR to build the control-flow graph (CFG) that includes all the reachable API calls from a UI element's callbacks. We further extended this component to support multi-threading, e.g., *AsyncTask.execute*) which was not covered by default. GATOR was then used to expand the built CFG by analyzing the inter-component communication (ICC) (e.g., permission-protected APIs may be triggered via services or broadcast receivers), since many of the vulnerabilities and malicious behavior in Android apps addressed in the literature are related to the ICC mechanism [92, 119, 118]. After identifying all reachable API calls, we used PScout and Aplorer to find out which API is permission-protected APIs (i.e., sensitive resources) [10, 14]. Accesses of content providers where apps can access sensitive resources were also detected by querying the content provider with URIs. For instance, by providing the URI content "*content://com.android.contacts*" to the *ContentResolver.Query* method, the app can access user CONTACTS.

⁶The embedded texts in icons we are also extracted using Optical Character Recognition (OCR) techniques (<https://github.com/tesseract-ocr/tesseract>).

⁷<https://developer.android.com/reference/android/Manifest.permission>

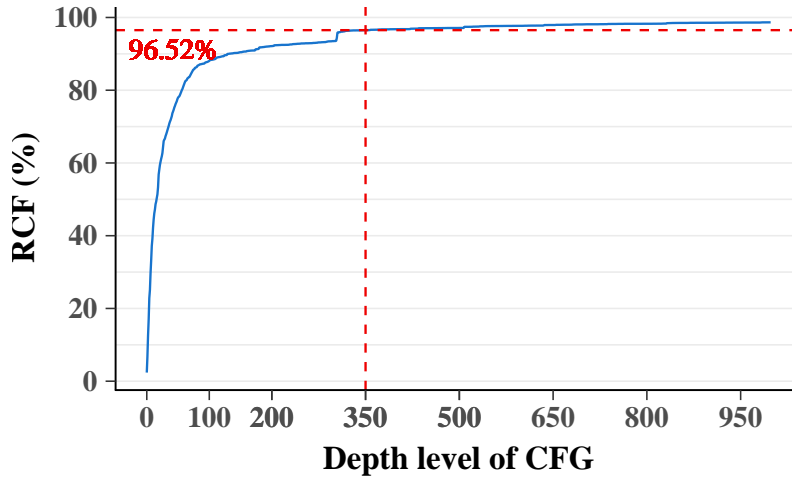


Figure 3.5: Relative cumulative frequency (RCF) of accessed sensitive resources depth level of CFG in 10,000 apps.

We further leveraged LibScout [12] to identify *actual SR* of third-party libraries (i.e., providing the information of 501 commonly used libraries). Additionally, to cover cases of unknown third-party code that LibScout could not identify, we extended its implementation to identify libraries using app package name as a heuristic [112, 11, 94]. It is not practical to build a CFG, which includes all of the reachable code statements due to execution time. Therefore, we needed to find a threshold to limit the depth of the CFG analysis counting from a callback. In our experiments, we randomly selected and analyzed 10,000 apps from the app dataset. With a maximum depth of 350 calls from the corresponding callback we could successfully identify 96.52% of the accessed sensitive resources of app’s UI elements (see Figure 3.5).

Evaluation To evaluate the precision of GUIBAT on mapping *actual SR* to app’s UI elements, we first randomly selected 200 Android apps from 32 categories (i.e., predefined categories on Google Play), and used dynamic analysis to extract runtime *actual SR* of each app’s UI element. Specifically, we used the dynamic instrumentation toolkit Frida⁸ to instrument the Android system to monitor access of sensitive resources while running the app by logging at each permission-protected API call and URI query, and used DroidBot [96] an automatic event generation tool, to randomly simulate user-interaction with the apps (performs best in comparison with similar tools [15]). This way, we could automatically create ground truth for apps’ UI elements and their *actual SR*. In this experiment, we limited the automated analysis time to 30 minutes for each app. As the results, we successfully identified 1,284 UI-sensitive resources mappings. Then we used GUIBAT’s *Accessed Sensitive Resource Detector* to extract app’s UI elements that access sensitive resources from these apps, and compare its results with the ground truth. The average precision and recall of GUIBAT is 90.19% and 96.65% respectively. We did not calculate the recall of this component for the whole apps, because it is practically impossible to calculate how many mapping relations are missed by GUIBAT

⁸<https://frida.re/>

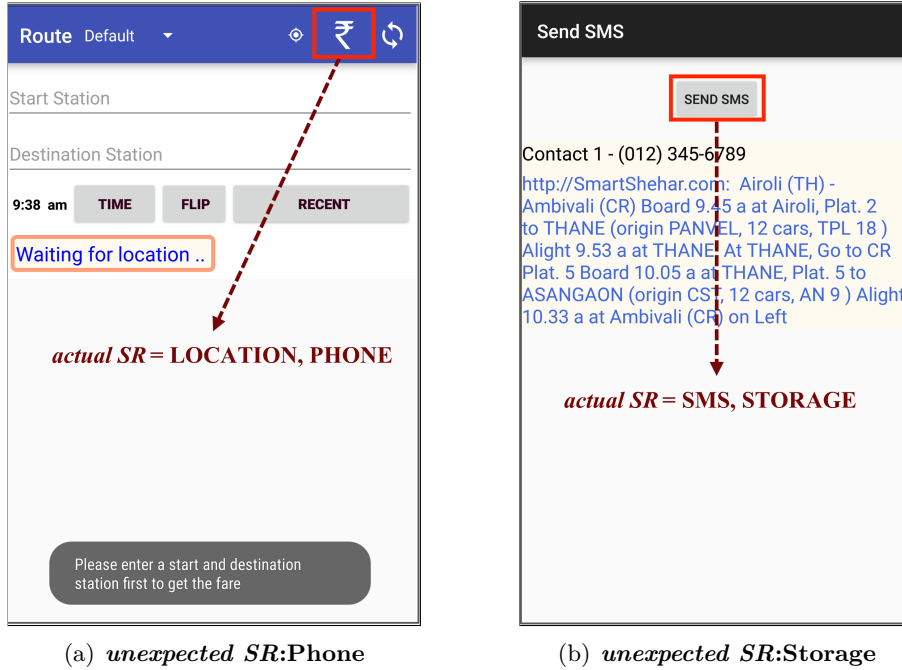


Figure 3.6: *Unexpected SR of a benign app (on Play Store).*

as there is no ground truth available, and due to the limitation of dynamic analysis which can not guarantee all the app code are analyzed.

3.2.4 Unexpected Sensitive Resource Access Detector

The final step in our work-flow (see Figure 3.2) is to find unexpected access to sensitive resources in a given app. Specifically, GUIBAT first identifies the user expectation of sensitive resource accesses from the apps' UIs (*expected SR*), then the actually accessed sensitive resources (*actual SR*) using *Expected Sensitive-Resource Access Detector*, and *Actual Sensitive-Resource Access Detector* respectively. An *actual SR* is considered unexpected if it is not contained in the *expected SR*. For example, in Figure 3.6 (a), when users click on the highlighted button, the app then accesses both LOCATION, and PHONE (e.g., to get the unique user's device ID such as the IMEI) sensitive resources. GUIBAT detects an *unexpected SR* since PHONE sensitive resource is not expected in any related context, e.g., on the surrounding UI elements, or in the feedback message at the bottom of the screen when users click on this button. Similarly, in Figure 3.6 (b), GUIBAT identifies the *Send SMS* as an *unexpected SR*, since it accesses user's phone STORAGE in addition to SMS while no further information is provided on this screen.

3.3 Evaluating GUIBAT

In this section, we present our results regarding the validation of GUIBAT. In particular, we first validated GUIBAT's effectiveness with the third user study. Our results showed

	Number of Sensitive Resource Accesses	
	Expected	Unexpected
Phone	19	104
Location	55	89
Storage	130	76
Contacts	118	16
Camera	22	13
Microphone	33	8
Calendar	18	1
SMS	108	0
Total	503	307

Table 3.4: Detected sensitive resource accesses in our sample.

that GUIBAT could accurately reflect users’ expectations of sensitive resource access in apps. Further, our evaluations showed that GUIBAT significantly outperforms prior works in terms of identifying user expectations of sensitive resource accesses when they interact with the app’s UI, and revealed the deficient of text-based only approaches.

3.3.1 Comparing to Users’ Expectations

To see whether the output of GUIBAT reflects users’ expectations of apps’ sensitive resource access, we carried out a further user study and tested users’ actual expectations while interacting with an app. The goal was to check whether GUIBAT’s output reflects these expectations. The main hypothesis was: *If our tool works as intended, sensitive resource access classified by GUIBAT as “unexpected” should be less expected by users than the one classified as “expected”.*

However, previous work showed that other variables might also influence users’ perception, attitudes, and expectations in the context of mobile apps. Particularly, the following factors have been primarily identified: 1) the users’ individual characteristics (e.g., gender, age, whether the persons have computer science background, or which mobile operating system people use) [56]; 2) the apps’ characteristics (e.g., which sensitive resources the app uses) [23, 112]; and 3) the user perception of the app descriptions [124, 132, 72]. To ensure that such effects do not overshadow our results, and, to be able to test whether GUIBAT provides added value compared to these known influential factors of user expectations, we included them in our study design as well.

Study Design and Procedure The goal of our study was to simulate the users’ interaction with an app as close to reality as possible. However, it was also important to cover different types of apps and different sensitive resources to be as representative as possible in our evaluation. After balancing these two goals, we decided on a survey design in which participants were shown an app, and then mentally put themselves (with the help of the app description and screenshots) in the situation that they were using this app.

We first presented participants the description of an app retrieved from Google Play. After reading the description, participants could select from a list of 8 sensitive resources (e.g., or none of them) those they expected the app to access. Participants were then

asked to explain briefly why they thought the app needed the selected resources (see *Q1* and *Q2* in Appendix A.1.2). This step prepared participants to put themselves in the situation of using the app, as it is the same information they see in Google Play before downloading and installing an app. This setup also allowed us to measure the effect of the app descriptions on the users' expectations of sensitive resource accesses and to control for this effect. Next, we asked participants to imagine that they were using the app and showed them a concrete screenshot taken from the corresponding app. For all *actual SR* that are detected by GUIBAT, participants should then indicate on a 7-point Likert scale to what extent they expect the app to access these resources (see *Q3* in Appendix A.1.2). We then asked participants to what extent they feel comfortable with the accessing of these resources (on a 7-point Likert scale) (see *Q4* in Appendix A.1.2). Lastly, we collected demographic data of our participants, including questions about their predominantly used mobile operating system and their background in computer science.

Selection of Apps We randomly selected 1,000 apps from the app dataset that was already used to extract the icons during the development of GUIBAT. Then we used DROIDBOT [96] to simulate user-interaction and automatically took screenshots of these apps. We limited the time for the dynamic analysis to 10 minutes for each app. Using this setting, DROIDBOT successfully analyzed 977 apps (97.7% of the total apps). Among 977 apps, we successfully extracted screenshots of *unexpected SR* in 311 apps by using GUIBAT. In a final step, we manually filtered out apps that were not in English or that were protected by FLAG_SECURE whose UI was not shown on screenshots. This resulted in a final set of screenshots of $N_{\text{apps}} = 243$ apps in which GUIBAT identified a total of 810 cases of sensitive resource access (see Table 3.4).

Survey The survey lasted 6 days and 486 Turkers had participated (for each of the selected apps, we surveyed with two participants). After collecting all the responses, we removed 41 careless responses based on attention checks. Participants' median age is 34 years (41.35% male, 57.98% female, and 0.67% others, see Table A.1 in Appendix for full demographics). Since we have included gender as a potential influencing factor in our analyses and only 3 persons indicated a non-binary gender, we unfortunately had to exclude these answers, since the model calculation based on such few data points would have been very error-prone. In the end, our final sample included $N_{\text{total}} = 1,444$ individual expectation ratings for sensitive resource access given by $N_{\text{users}} = 442$ participants and based on $N_{\text{apps}} = 243$ apps.

Data Analysis To analyze the effects of all the variables listed at the beginning of this section, we used a regression approach that predicts participants expectations regarding sensitive resource accesses. Since each participant in our survey indicated his/her expectation for several types of sensitive resource access, these data points are not independent from each other. The same applies to the data from any particular app, which was always presented to several participants. To account for this fact, we designed our analysis as a multi-level approach and included these two grouping aspects of our data as *random effects* in our model. In concrete terms, this means that the intercept

	AIC	logLik	df	<i>p</i>
Base model	4,853.8	-2,420.9		
+ Random effects	4,798.8	-2,391.4	2	<0,001
+ Users' characteristics	4,785.2	-2,380.6	4	<0,001
+ Apps' characteristics	4,624.3	-2,293.1	7	<0,001
+ Users' expectations based on app description	3,957.7	-1,958.9	1	<0,001
+ GUIBAT	3,944.2	-1,951.1	1	<0,001
+ Interaction	3,945.4	-1,950.7	1	0.369

All regression models listed above predict the users' expectations regarding apps' sensitive resource access; AIC = Akaike Information Criterion; df = degree of freedom; logLik = log likelihood; *p* quantifies statistical significance; significant *p*-values are printed bold.

Table 3.5: Model comparison based on Goodness of fit.

of our regression function is not fixed for all persons and all apps, but can vary freely. Since we measured the expectations of our participants regarding sensitive resource accesses with a 7-point Likert scale, we used an ordinal regression based on a cumulative link model with a *logit* linkage function and *flexible* thresholds between the categories. This regression method provides a regression function as well as ordered thresholds that describe the six category boundaries between the values of our 7-point Likert scale. In an iterative process, we specified different regression models and compared them using the Akaike information criterion (AIC), as well as direct comparisons of the goodness of fit using Chi²-tests to find the model with the best trade-off between complexity and fit to the empirical data [79, 104].

Results of the Evaluation We started with a base model without any independent variables and without the random effects and added successively the following effects/predictors in the next steps:

1. the random effects
2. the users' individual characteristics
3. the apps' characteristics
4. the users' expectations based on the description
5. **the classification based on GUIBAT**
6. the interaction of the previous variables

The results of this process showed that every more complex models up to step 5 could explain the empirical data significantly better than the preceding/simpler model (see Table 3.5 for the goodness of fit of each relevant step in the model building process). The interaction effects (step 6) do not add any further value and therefore Model 5, which explains 63.5% of the empirical variance is the final model for our analysis. Thereby, the significant difference between the model without the predictor based on GUIBAT's output (Model 4) and with this predictor (Model 5) shows that our tool can explain the

	Estimate (<i>b</i>)	SE	z value	<i>p</i>	
<i>Users' characteristics</i>					
Computer background (Yes)	0.384	0.223	1.722	0.090	
Age	-0.007	0.009	-0.881	0.379	
Gender (Male)	-0.190	0.180	-1.058	0.290	
Used devices (iPhone)	-0.547	0.179	-3.061	0.002	
<i>Apps' characteristics</i>					
Contacts	1.061	0.455	2.331	0.002	
SMS	1.565	0.466	3.362	0.001	
Phone	1.618	0.503	3.388	0.001	
Microphone	2.078	0.478	4.131	<0.001	
Location	2.087	0.465	4.492	<0.001	
Storage	2.621	0.456	5.751	<0.001	
Camera	2.647	0.519	5.099	<0.001	
<i>Users' expectations based on app description</i>					
Expected access (Yes)	3.637	0.173	21.046	<0.001	
<i>GUIBAT</i>					
Detected as <i>unexpected SR</i>	-0.569	0.145	-3.930	<0.001	
<i>Thresholds for category boundaries</i>					
1 2	2 3	3 4	4 5	5 6	6 7
1.980	2.756	3.354	4.004	4.787	5.775
Used devices (Baseline=Android); Sensitive resources (Baseline=Calendar); GUIBAT (Baseline= <i>expected SR</i>); SE = Std. Error=; significant <i>p</i> -values are printed bold.					

Table 3.6: Final regression model predicting users' expectation regarding sensitive resource accesses.

variance of user expectations regarding sensitive resource accesses beyond the effects on which previous research has focused on. Table 3.6 presents the final regression model's results, including the estimates and standard errors of the predictors, as well as the 6 thresholds of the Likert scales category boundaries. The proportional odds assumption as a prerequisite for a cumulative linking model was fulfilled for all predictors.

Our results showed that among the users' characteristics only the mobile operating system that people use predominantly had a significant effect on users expectations ($b = -.55, p = .002$). Particularly, iOS users were significantly less likely to expect that apps use sensitive resources than Android users. We further found differences in the prevalence of expectations regarding access to different types of sensitive resources. Thus, for instance, access to the CALENDAR was significantly less likely expected by users than access to all other sensitive resources, and was therefore picked as the baseline category in the regression model ($b_{\text{all others}} = 1.06-2.65, p_{\text{all others}} \leq .002$). This results are in line with prior work that shows that users do not consider all permission to be equally sensitive [23, 112].

We also found that expectations regarding sensitive resource accesses based on app descriptions had an impact on the expected access to sensitive resources during app usage. If participants expected access to a certain sensitive resource based on the description, they were also more likely expecting access to this sensitive resource during their further interaction with the app ($b = 3.64, p < .001$). This is in line with prior work and particularly supports the basic assumptions of research that analyzes the







Icons	Texts	<i>Actual SR</i>	<i>Unexpected SR by GUIBAT</i>
	tencent enter purchas btn take pic-tur	Camera, Location, SMS	Location, SMS
	time search	Location	Location
	default user icon	Location	Location
	holder mic	Contacts, SMS	Contacts, SMS
	post float letter refresh system friend topic new subject normal	Location	Location
	to to leg menu save	Contacts, Location, SMS, Microphone	Contacts, Location, SMS, Microphone

Table 3.7: Example of *unexpected SR* that are detected by GUIBAT from DeepIntent’s benign training dataset.

descriptions and links this content to the users’ expectations [124, 132, 72].

Most importantly, however, we found a significant effect of the predictor that represents the output of our tool. If sensitive resource access was classified by GUIBAT as unexpected, participants significantly less likely expected an app to behave in such a way, than if the sensitive resource access was classified as expected ($b = -0.57$, $p < .001$). This result supports the hypothesis that GUIBAT works as intended. Thereby, this conclusion is also supported by the results of the model comparison process, which showed that our tool can explain the users’ expectations significantly beyond previously considered influencing factors (such as app description).

3.3.2 Comparing to Prior Works

To see how effective GUIBAT is in identifying *unexpected SR*, we compare GUIBAT with the most recent work on detecting intention-behavior discrepancy namely DeepIntent, and with text-based only approaches.

Compare with DeepIntent We applied GUIBAT on the dataset that DeepIntent took as the ground truth to train their icon-behavior classifier. The dataset contains 7,691 data points that were extracted from a set of benign apps, and each data point is a triple of $\langle icon, text, actual_SR \rangle$. We found that 22.91% (1,020 out of 4,452) of the data points are *unexpected SR*⁹. All of these *unexpected SR* were then manually verified independently by two of the authors (disagreements were excluded) based on our user study results (whether the *actual SR* are justified by the associated icons/texts), and we found that 958 of them (93.92% of 1,020) indeed are *unexpected SR* (see Table 3.7 for examples). However, it is unknown why apps make these *unexpected SR* due to not having the app’s code or app’s name of these data points. Determining the root cause for such *unexpected SR* is outside the scope of this paper. Our results show that

⁹We filtered out data that was out of scope for this work, e.g., icons have embedded foreign languages.

Sensitive Resources	“unseen” dataset from		GUIBAT’s Icon
	Google Image		Training Dataset
	GUIBAT	DeepIntent icon-only	DeepIntent icon-only
Camera	0.93	0.51	0.49
Calendar	0.98	-	-
Contact	0.85	0.61	0.52
Location	0.94	0.55	0.55
Microphone	0.92	0.53	0.55
Phone	0.97	0.40	0.42
SMS	0.70	0.47	0.50
Storage	0.75	0.50	0.55
Average	0.88	0.51	0.50

Table 3.8: The AUC score of GUIBAT’s Icon Classifier and DeepIntent’s models.

	Number of UI elements	
	w/o Contexts	w/ Contexts
Exclusively by Texts	177,022	225,702
Exclusively by Icons	18,740	58,181
Both Texts & Icons	11,594	146,561
Total	207,356	430,444

Table 3.9: Distribution of *expected SR* on icons/texts.

relying on benign apps’ icon-to-permission association as the “norm” to detect behavior discrepancy is not enough. There are *actual SR* (in this case, 22.91%) of benign apps that were previously considered normal by DeepIntent but are unexpected by users.

Besides, to show the effectiveness of our *Icon Classifier* in identifying icons and their associated sensitive resource accesses, we compared it with the *icon-only* classifier of DeepIntent¹⁰. We first ran the *icon-only* classifier on our labeled dataset that reflects user perception of sensitive resource accesses (see Section 3.2.2.1). This resulted in a low accuracy (AUC=0.50, see Table 3.8 for the details). Further, for a fair comparison, we ran our *Icon Classifier* and the *icon-only* classifier on an “unseen” dataset. To create the labeled *unseen* dataset we followed the approach used in IconIntent [168]. Specifically, to obtain icons belonging to each sensitive resource, sensitive resource name (e.g., “camera”) together with the keyword “icon” were used to search for icons from Google Image. The top 100 results were then downloaded. This resulted in 800 images in total. Images that were not icons were then filtered out, resulted in 657 labeled icons. On this *unseen* dataset, our *Icon Classifier* significantly outperforms the DeepIntent’s *icon-only* classifier in identifying sensitive resource related icons when taking user perception into account (details are included in Table 3.8). In particular, our *Icon Classifier* achieved an AUC value of 0.88, while the *icon-only* had an AUC value of 0.51. Our results indicate that approaches without considering how end users perceive app’s data access is less effective in detecting sensitive resource accesses that violate user expectations.

¹⁰For a fair comparison, we trained these classifiers multiple times (K=10) based on the provided source code (<https://github.com/deepintent-ccs/DeepIntent>) to achieve the authors’ reported performance.

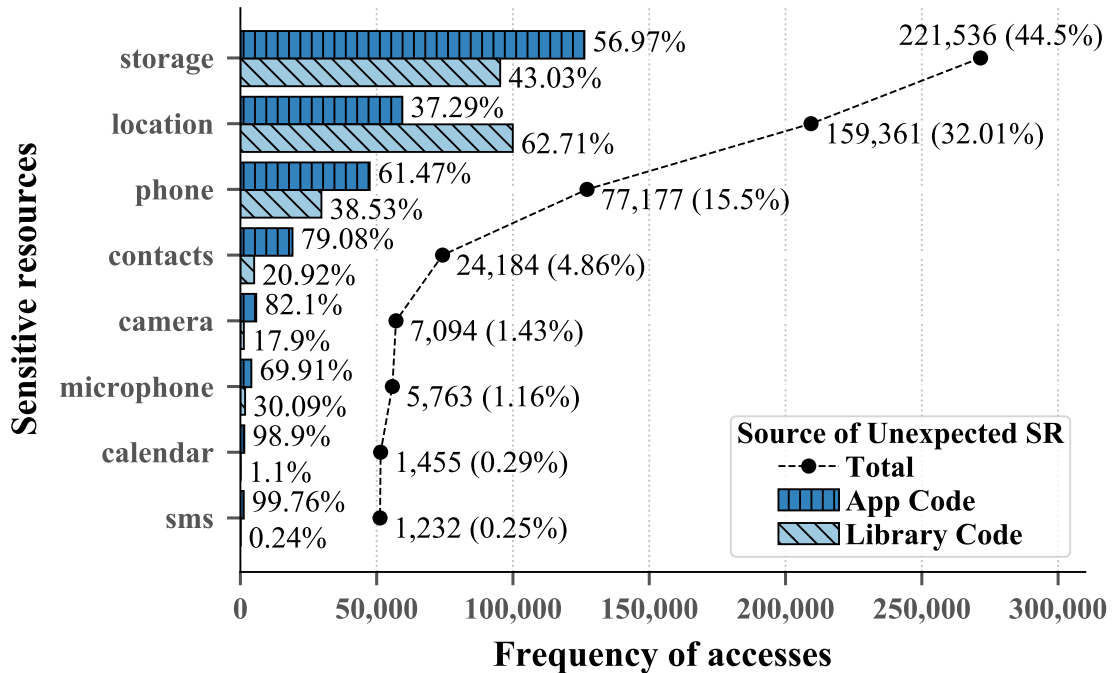


Figure 3.7: Distribution of *unexpected SR* among sensitive resources in 36,115 apps. Attribution of app code and library code sum up to 100% in each sensitive resources.

Compare with text-based only approach In Table 3.9, when we consider the contexts where UI elements are represented (e.g., including surrounding UI elements), among UI elements that advertise their sensitive resources usage, 47.56% are advertised using icons. This shows that relying solely on textual UI elements would result in a severely high number of false positive (i.e., sensitive resource accesses that are represented on UI but missed by the text-based detectors). Text-based only approaches [11, 80] did not cover such UI elements, hence their analysis would miss a significant portion of UI elements (e.g., about 47.56%) affecting the final results. Further, GUIBAT also identified 28,145 UI elements (which access sensitive resources) that associate with icons on UI screens that comprise solely of icons (i.e., no textual UI elements were presented) from 6,730 apps (14.06% of apps in our dataset). This means, text-based only approaches would completely miss such screens, hence the analysis results of the corresponding apps (e.g., 14.06%) would be affected.

3.4 Empirical Study of Unexpected Data Access in The Wild

To explore the status quo of *unexpected SR* in the wild, we utilized GUIBAT to perform a large-scale analysis on Android apps collected from Google Play. The run-time of GUIBAT’s static analysis depends on the complexity of the analyzed apps, hence it is not practically feasible to analyze all apps with-in a reasonable amount of time. Therefore, we randomly selected 100,000 apps¹¹ (see Section 3.1.1) and ran GUIBAT

¹¹These apps declare at least one dangerous permission in the apps’ manifest file.

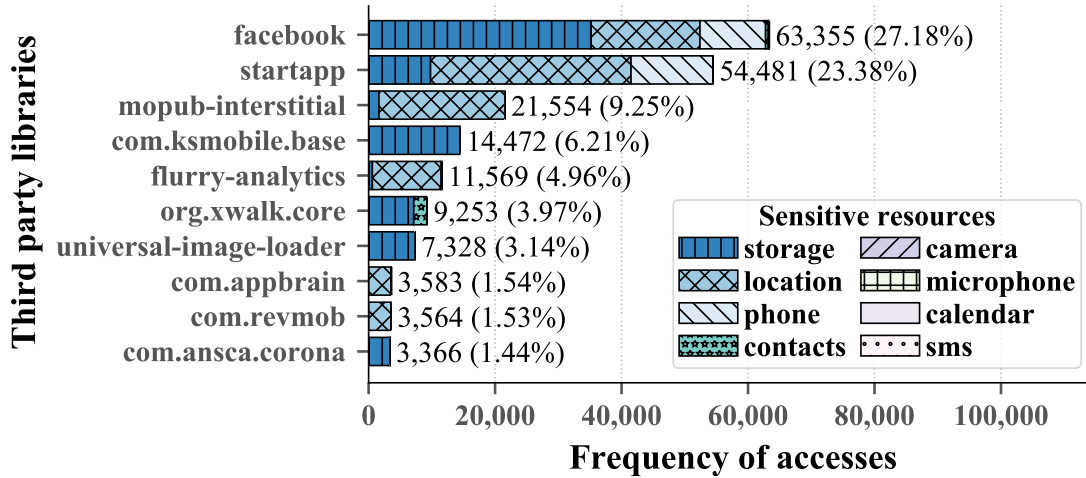


Figure 3.8: Top 10 libraries with *unexpected SR*.

on each app for 5 minutes, and then terminated it if no results were outputted. As the result, GUIBAT successfully analyzed 89,371 (89.37% of 100,000) apps. Among 89,371 apps, GUIBAT detected that only 60,485 (67.68% of 89,371) apps trigger permission-protected API calls. This means that not all declared permissions are actually used (i.e., over-privileged apps). Further, GUIBAT identified 656,110 UI elements belonging to 47,909 (79.21% of 60,485) apps that access sensitive resources.

GUIBAT detected 386,285 (58.88% of 656,110) UI elements from 36,115 (75.38% of 47,909) apps that access at least one sensitive resource which is not expected by end users. Figure 3.7 shows the distribution of *unexpected SR* by sensitive resources as well as the attribution to the *unexpected SR* (i.e., by app or by third-party libraries). We see that STORAGE sensitive resource has the highest number of *unexpected SR* accounting for 44.50% (221,536) of the total *unexpected SR*, followed by LOCATION, and PHONE resources, accounting for 32.01%, and 15.50% respectively. The *unexpected SR* pertaining to these three sensitive resources contribute to 92.01% of the total *unexpected SR*. More importantly, third-party library is the contributing factor to the high number of *unexpected SR* with 43.03% of STORAGE, 62.71% of LOCATION, and 38.53% of PHONE.

To study to which extent third-party libraries contribute to the *unexpected SR* of apps and found 178,206 (46.13% of 386,285) UI elements from 17,674 apps that have *unexpected SR* from third-party libraries. Per apps, 38.20% of 36,115 apps have *unexpected SR* that are exclusively attributed by third-party libraries. We see that LOCATION, PHONE, and STORAGE sensitive resources have the highest number of *unexpected SR* caused by using third-party library APIs (see Figure 3.8). A prior work of Book et al. showed that LOCATION and PHONE sensitive resources have constantly been the most frequently used of advertisement libraries [24]. We found that among the top 10 libraries that contribute to the total *unexpected SR* of the PHONE, LOCATION, STORAGE sensitive resources, the majority of them are advertising and analytic libraries (see Figure 3.8).

3.5 Summary

Recall that understanding users' perception of app behaviors is important in detecting data access that violates user expectations. While existing works have used various proxies to infer user expectations or try to predict the intention of apps' icon UI elements, however without understanding how real-world users perceive data access — which we found less effective in detecting data access that violates user expectations.

In this chapter, we fill this gap by directly measuring how end-users perceive app behaviors based on graphical UI elements via extensive user studies. In particular, we conducted three user studies in total (N=904). The first two user studies were used to build a semantic mapping between user expectations of sensitive resource accesses and the common graphical UI elements (N=459). We then leveraged this knowledge to build GUIBAT (Graphical User Interface Behavioral Analysis Tool) - that detects sensitive resource accesses that violate user expectations. With the third user study (N=445), we showed that GUIBAT correctly reflects users' expectations, and revealed the deficient of prior work. Having applied on a set of 47,909 apps, GUIBAT identified that 75.38% of apps have at least one *unexpected SR*. Further, GUIBAT also revealed that 46.13% of the *unexpected SR* were attributed by third-party libraries. We showed the urgent need for more transparent UI designs to better inform users of data access, and called for new tools to support app developers in this endeavor.

4

Detecting Violations of GDPR's Explicit Consent

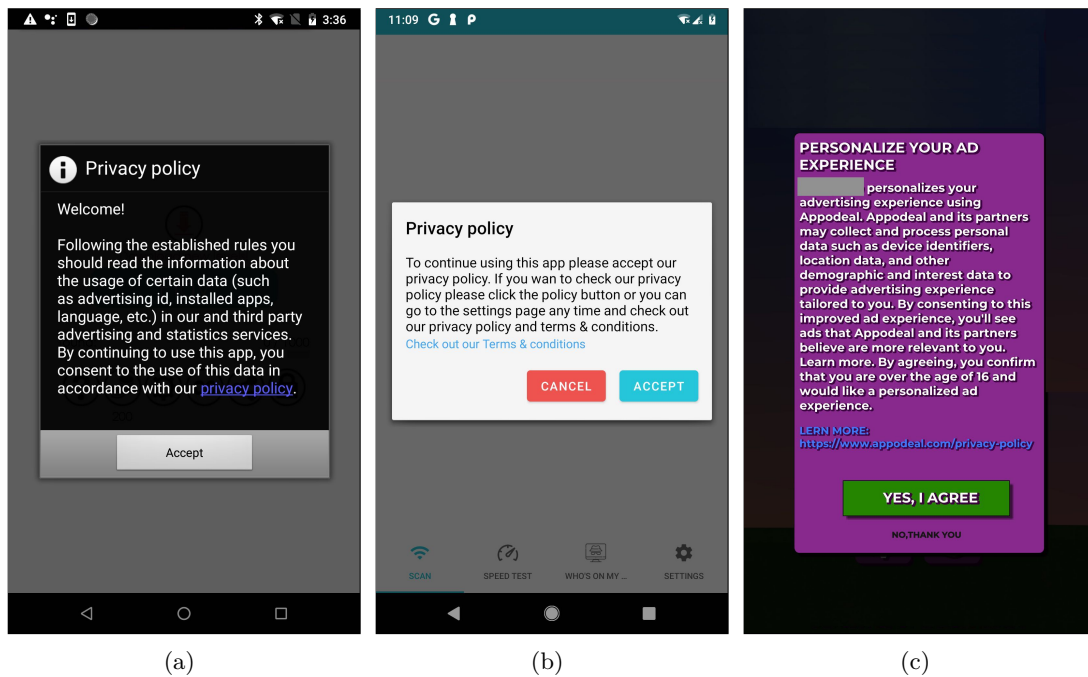


Figure 4.1: Example of consent dialogues in Android apps.

In the previous chapter, by exploring the transparency of data access in Android apps, we find a non-negligible number of apps (i.e., 75.38% of 47,909) that access user data in unexpected ways, which has implications for users’ privacy. It is unclear whether such data collection could be legally justified under GDPR. While many researchers have worked to detect and analyze consent notices (i.e., cookie banners) and their impact on the Web advertising and tracking industry after the GDPR went into effect [103, 152, 140, 151, 36, 90, 156], the community lacks insight into such violations in the mobile ecosystem. Recently, Weir et al. [163] surveyed app developers and observed that most developers’ changes were cosmetic due to the GDPR legislation (e.g., adding dialogues) — which raises a serious question about whether these changes fulfill the legal conditions for collecting valid consents. Figure 4.1 shows examples of consent dialogues that mobile users in the European Union observe on many apps they use today. Notably, neither (a) nor (b) are valid consent dialogues required before data sharing with third parties, and even dialogue (c) is meaningless if data sharing occurs before the user has the ability to reject this.

In this chapter, we aim to understand how often GDPR’s explicit consent mandate is violated in the mobile ecosystem, focusing on Android. Specifically, we answer RQ2: *How many apps send out personal data without prior consent? Of the apps which send out any data, how many send it to parties that act as data controllers under the GDPR?* Specifically, inspired by prior work [137], we build a first semi-automated and scalable pipeline to detect personal data sent to the Internet by analyzing the network traffic generated by apps without user explicit prior consent and apply this to our dataset, which consists of both high-profile and long-tail apps (Section 4.2). Then, based on

the domains that receive data protected under the GDPR without prior consent, we collaborate with a legal scholar to assess the extent to which contacted domains are third-party data controllers — which require explicit consent. Then, we conduct a large-scale study with 86,613 Android apps available through the German Play Store, allowing us to provide a comprehensive overview of the current state of the violation of GDPR’s explicit consent on mobile apps in the wild¹ (Section 4.3). To further understand the reasons behind such violations, we run a notification campaign to inform affected developers and gather insights from their responses (Section 4.4).

4.1 Legal Analysis of GDPR’s Explicit Consent

Under the GDPR, all processing of European residents’ personal data has to have a legal justification. GDPR Article 6 [58] contains the six general justifications for processing. Among others, the processing may be based on consent, the fulfillment of a contract, compliance with a legal obligation, or the data controller’s legitimate interests when such interest outweighs the fundamental rights and freedoms of the data subjects. In practice, most advertising companies rely on consent or legitimate interests as the legal basis for processing personal data for profiling and targeted advertising (i.e., since the legal ground necessary for the performance of a contract does not apply in these circumstances [19, 54]).

However, a recent study from the Norwegian Consumer Council [54] shows that data subjects do not have a clear understanding of the amount of data sharing and the variety of purposes their personal data is used for in targeted ads. A large amount of personal data being sent to various third parties, who all have their own purposes and policies for data processing, are detrimental to the data subjects’ privacy. Even if advertising is necessary to provide services free of charge, these privacy violations are not strictly necessary to provide digital ads. Consequently, it seems unlikely that these companies’ legitimate interests may claim to outweigh the fundamental rights and freedoms of the data subject. This means that many of the ad tech companies would most likely have to rely on consent as the legal basis for their processing operations. In case the data transfer in question relies on user consent, the GDPR requires the consent to be *freely given, specific, informed, and unambiguous*. Further, the data subject must have given consent through a statement or by a clear affirmative action prior to the data processing in question (GDPR Art. 4(11) [60] and Art. 7 [59]).

Unambiguous consent under the GDPR must meet certain conditions. The GDPR Art. 7(2) states that: “*If the data subject’s consent is given in the context of a written declaration which also concerns other matters, the request for consent shall be presented in a manner which is clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language*”. The user’s consent has to be easily differentiated from other declarations or even consent to other processing activities. The user has to be specifically asked to consent to data sharing and processing for

¹We note that the definition of consent can be subject to different legal interpretations by different legal scholars. Therefore, only a judicial ruling can provide legal certainty about whether they are actual violations of GDPR consent requirements.

Data Type	Description
AAID	Android Advertising ID
BSSID	Router MAC addresses of nearby hotspots
Email	Email address of phone owner
GPS	User location
IMEI	Mobile phone equipment ID
IMSI	SIM card ID
MAC	MAC address of WiFi interface
PHONE	Mobile phone's number
SIM_SERIAL	SIM card ID
SERIAL	Phone hardware ID (serial number)
SSID	Router SSIDs of nearby hotspots
GSF ID	Google Services Framework ID

Table 4.1: Overview of personal data tied to a phone.

advertising purposes and this consent must not be grouped together with, e.g., consent to download the app or consent to access certain APIs on the phone.

In order to be legally valid, consent with regard to the processing of personal data has to be *explicit*. This means that the controller should obtain verbal or written confirmation about the specific processing [Recital 32]. According to the Article 29 Working Party, consent cannot be based on an opt-out mechanism, as the failure to opt-out is not a clear affirmative action [126]. The user has to actively give their consent, i.e., by clicking “I agree” on a consent form. Merely continuing to use an app or other passive behavior does not constitute explicit consent. Lastly, the consent has to be obtained prior to the data processing to be considered valid.

Our research focuses explicitly on these aspects of user consent. In particular, with respect to the aforementioned regulations, transmitting data to an advertisement company without *prior, explicit* consent by the user for the purpose of targeted advertisement is considered violating GDPR.

4.2 Methodology

Our main goal is to have a mostly automated and scalable solution to detect personal data that is being sent to the Internet without users’ *explicit* consent, as is mandated by the GDPR. We set up an array of Android devices, on which we run each app (without any interaction) and capture the network traffic (Section 4.2.1). Based on personal data which is directly tied to the phone (see Table 4.1), we automatically detect this data in both plain and encoded form through string matching. Moreover, we derive a methodology that allows us to pinpoint data that may be other unique identifiers and manually validate whether this can be used to track the user/device (Section 4.2.2). In the following, we outline how we conduct each of the steps in more detail.

4.2.1 App Setup and Network Traffic Collection

We run each app and capture its network traffic. Here, we aim to detect apps’ network traffic without users’ explicit consent. To achieve this, we simply open the app but do

not interact with it at all. The underlying assumption is that if network traffic occurs when this app is opened without any interactions, we have naturally not consented explicitly to any type of data collection by third parties. Hence, any data being sent out must not be PD, so as not to violate GDPR. Orthogonal to that, in practice, users may not grant all the apps' permission requests, or the app may use the runtime-permission mechanism (i.e., the permissions are not granted at installation time, and users will allow/deny permission requests at runtime when using the app). As such, it may be the case that personal data (e.g., the IMEI) can only be accessed after the user consents to API usage. However, this API consent does not imply consent to have sensitive data shared with third parties. Therefore, to be legally compliant, the app must respect explicit consent even if it is authorized to access the data through a granted permission.

Recall that our goal is to analyze apps on a large scale. Hence, relying on static analysis techniques, which may produce a vast amount of false positives, is not an option [158, 93, 22]. Furthermore, we aim to have a lightweight solution to allow us to check thousands of apps in a reasonable time. Hence heavyweight instrumentation of the app itself is out of the question. Therefore, our approach is prone to miss certain instances (e.g., if we are unable to detect unique identifiers in the outgoing traffic or the app crashes in our lightweight instrumentation).

We rely on six rooted devices (Pixels, Pixel 3a, and Nexus 5) running Android 8 or 9 to analyze a given app. To intercept the TLS traffic, the devices are instrumented with our own root certificate (i.e., by using MitM proxy [33]). Further, we use *objection* to detect and disable SSL Pinning [117]. In the first step of our analysis pipeline, we aim to identify apps that send *some* data when started. To achieve that, we install the app in question and grant all requested permissions listed in the manifest, i.e., both install time and runtime permissions. Subsequently, we launch the app and record its network traffic. As our initial tests showed that apps sometimes did not load on first start, we close the app and reopen it, so as to increase the chances of observing any traffic. If an app shows no traffic in either of these starts, we discard it from further analysis.

4.2.2 Traffic Log Analyzer

Under the GDPR, personal data includes the Advertising IDs [116], location data, and online identifiers (e.g., IP addresses, any unique tracking identifiers) which can be used to identify users over a long period, potentially across different apps and services [4]. Next to data protected through OS permissions (e.g., IMEI, MAC), an app may also use other types of persisted, unique identifiers to track users. Hence, our analysis focuses on all possibly sensitive data as well as data that can be used to uniquely track a user or a specific instance of the app on their phone.

4.2.2.1 String-Matching Device-Bound Data

The first type of data we consider is such data that is tied to the phone, such as the location, the AAID, or the MAC address. Since such information is accessible by apps, we extract the relevant values from the phone through the Android debug bridge to ensure we know these values for each phone. The data selected for this (see Table 4.1) is inspired by the data used in prior work [137, 133]. Specifically, we first use simple

string matching to identify personal data that is static and known in advance. This information includes persistent identifiers bound to the phone (e.g., the IMEI, the MAC address of the WiFi interface, and the AAID) and those that are otherwise sensitive, such as the device owner’s name, email address, or phone number. For the geolocation data, we search for the precise latitude and longitude written as a floating-point number, and those values are rounded to 3, 4, and 5 decimal places.

Beyond simple string-comparison, we also search for common transformations, such as upper/lower case, hashing (e.g., MD5, SHA-1), or encoding (e.g., base64) in our analysis. Naturally, this may miss cases in which, e.g., a custom hash function is used on the sensitive data by the app. To identify such cases as well as cases in which an app creates a persistent identifier itself, we conduct a second check for potential unique tracking identifiers.

4.2.2.2 Potentially Unique Tracking Identifiers Detector

This step aims to identify parameters that could be used to track and profile an individual, but do not obviously string-match with known values such as the IMEI. We aim to cover both cases of obfuscated usage of common identifiers as well as those cases where the app generates a persistent identifier and stores it locally. For example, from Android 8.0, the Android ID scopes to $\{user, app\ signing\ key, device\}$ that does not obviously string-match with known identifiers.

More specifically, for a given app, we perform multiple runs (R_i) with a different set of devices (P_i) to monitor and record its network traffic. For each run R_i on a particular device P_i , we first install the app in question and grant all necessary requested permissions. While monitoring the app network traffic, we start the app, close the app and start it once more. By analyzing the captured traffic in run R_i , we extract all contacted hosts (domain names) as well as the GET and POST parameters (including parsing JSON if the content type of the request is set accordingly). This allows us to identify all contacted domains as well as the parameters and the values the app sent out. The output contains a set of triples $t_i = \{ \langle domain, parameter, value \rangle \}$. Each triple $\langle domain, parameter, value \rangle$ is the identified contacted domain together with its parameter and the value that is being sent in the run R_i by the analyzed apps.

To this end, we further define two functions: (1) $diff(t_i, t_j)$ outputs all triplets of $\langle domain, parameter, value \rangle$ in t_i for triples that have the same domain and parameter but different value between t_i and t_j ; (2) the function $stable(t_i, t_j)$ outputs all triplets of parameters which remained unchanged between two sets. Figure 4.2 shows an overview of our approach to detect potential unique identifiers (which we refer to as *UID* in the following). In general, four steps are involved:

1. On phone P_1 , we first perform a run R_1 to discover all the app’s network traffic. Then, by analyzing the R_1 traffic, we identify all of the contacted domains and their parameters (t_1). If there is no data sent to the Internet by the app ($t_1 = \{\}$), no further step is required.
2. On the same phone P_1 , we now conduct run R_2 (installation and two open/close cycles). In between the two runs, we uninstall the app and set the time one day

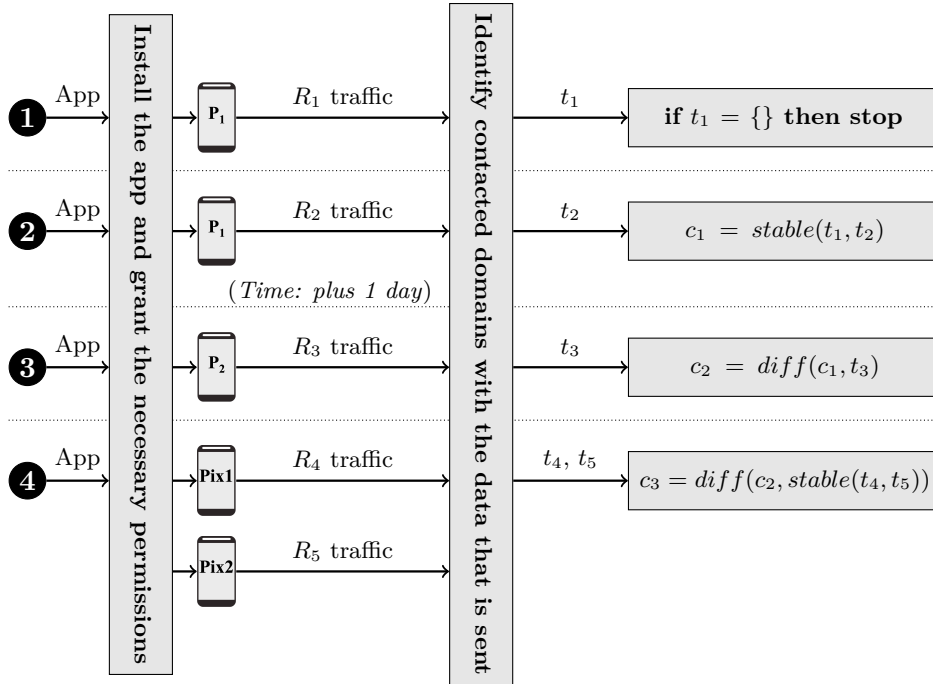


Figure 4.2: Overview of our methodology to identify potential UIDs. After each step, the analysis terminates if the resulting set of candidate parameters is empty.

into the future. The intuition here is that if an app is sending some persistent identifier, this would be the same across time and remain on the device (e.g., in persistent storage or based on some information about the phone). Again, we analyze the traffic to extract tuples (t_2). All parameters which are not stable between these runs cannot be persistent identifiers (e.g., the date) and are hence discarded. Suppose an app has any parameters with stable values across the two runs ($c_1 = \text{stable}(t_1, t_2)$). In that case, we consider a first candidate list for the next step — otherwise, we terminate the analysis (if $c_1 = \{\}$).

3. We now perform a run R_3 and extract the triplets from its traffic (t_3) on a different phone P_2 . For each parameter value that remains stable across the two phones ($\text{stable}(c_1, t_3)$), we assume the stable value is tied to the *app* (such as some app-specific token) and hence discard these. If an app has at least one parameter, for which the value remained stable between R_1 and R_2 (both on P_1), but differs between R_2 (P_1) and R_3 (P_2), we consider this app further (naturally only considering those parameters that differed), i.e., $c_2 = \text{diff}(c_1, t_3) \neq \{\}$.
4. Given the diversity in our used phones, such a difference may simply be caused by the make and model or the OS version that is being sent out. To remove such cases, we now conduct further two runs R_4 and R_5 , this time on two phones with the same make and model and OS version (Pixel 3a with Android 9). Suppose data is stable between these two runs ($\text{stable}(t_4, t_5) \neq \{\}$). In that case, we deem the corresponding parameter to be related to the device's make, model, or OS

Domains	Parameter
appsflyer.com	deviceFingerPrintId=<UUID>
branch.io	hardware_id=6fd9a2e0f2721498
tapjoy.com	managed_device_id=tjid.36cec2b4196...
unity3d.com	common.deviceid=d3d55baf21d8f31839...

Table 4.2: Examples of the UIDs identified by our approach.

version, which is not a viable tracking identifier, and hence discard the entries. The outputs of the final step is then $c_3 = \text{diff}(c_2, \text{stable}(t_4, t_5))$.

Finally, this leaves us with a highly filtered list of candidates for persistent identifiers (c_3). In the final step, we manually check the parameters identified in this fashion, to ensure that they do not contain false positives. Particularly, we removed low-entropy entries such as carriers, time zones, or LAN IPs. Moreover, we also took into account the names of the parameters, disregarding parameter names that did not indicate any identifying capabilities (such as `rs` parameter on `helpshift.com`, which has sufficient entropy but lacks the clear indication of an identifier in its name). For our analysis, which we present in detail in Section 4.3, we identified 2,113 potential parameter/domain pairs that matched the criterion in the final stage. Of those, we discarded 412, e.g., because they were related to the (different) carriers or install times. Examples of different UIDs we detected this way are shown in Table 4.2. That is, given an app, our pipeline can automatically detect the sending of personal data (such as IMEI, IMSI, UIDs) without users’ prior explicit consent. However, we have to *manually* vet the potential UIDs to avoid false-positive reports. Notably, we therefore may have missed true positives, which we nevertheless favor over a false positive.

4.3 Large-Scale Analysis

In this section, we present the results of our empirical study of Android apps on Google Play, with respect to the violation of GDPR’s explicit consent mandate. We first outline which datasets we consider and subsequently present our analysis results. We note that all technical testing was done in Germany where the GDPR applies, i.e., our geolocation is Germany and the app store is set to the German variant. Based on our findings, we manually analyze the contacted domains with the help of a legal scholar to determine which third parties are *data controllers* for which GDPR mandates explicit consent.

4.3.1 App Dataset Construction

Our analysis aims to assess the state of GDPR violations in both high-profile and long-tail apps on the Play Store, and to understand if the violations are specific to either of them. To achieve this and compare these types of apps, we sample two datasets, totaling 86,163 apps:

High-profile app dataset: We crawled the top free high-profile apps in May 2020 from the Google Play store based on the AppBrain statistic [6]. For each country and

33 categories, AppBrain lists the top 500 apps. However, for some categories, AppBrain does not provide a full list of 500 apps (e.g., Events with only 271 apps). Therefore, as a result, our crawler obtained 16,163 high-profile apps from 33 app categories.

Long-tail app dataset: Between May and September 2020, we crawled all free Android apps available in the store from Germany and successfully obtained more than 1 million apps. Rather than running the analysis of the entire dataset, we decided to filter the list of apps through two steps to reach a more manageable dataset: we first rely on *Exodus-Privacy* [43] to identify apps that have integrated tracking or advertising libraries. As a result, we obtained more than 700,000 apps with embedded tracking or advertising libraries (304 of which are detected by Exodus-Privacy) in their code. Of these apps, we randomly sampled approx. 10% of apps with at least 10,000 downloads and excluded those in the high-profile set already, yielding 70,000 distinct apps for testing.

We note that this pre-selection strategy of filtering out apps which Exodus-Privacy did not flag as containing advertising or tracking libraries results in a sampling bias compared to the high-profile apps. To account for that, when comparing the statistics later, we only compare our data against those high-profile apps that Exodus-Privacy also flagged.

4.3.2 Network Traffic Analysis

We were able to successfully analyze 72,274 (83.9%) apps, i.e., 14,975 high-profile apps and 57,299 long-tail apps. The remaining 13,889 either crashed or detected the analysis environment, making all of them potential false negatives.

Out of the 72,274 successfully analyzed apps, we identified 41,900 apps that contacted the Internet in either of the launches in R_1 . Specifically, we identified 10,290 unique fully-qualified domain names being contacted. However, we found that a single registerable domain uses many subdomains (e.g., `rt.applovin.com`, `d.applovin.com`). To normalize these hosts to their registerable domain (`applovin.com` in the above cases), we rely on the public suffix list [131]. We refer to these resolved domains as *domain names* in the following. As a result, we identified 7,384 domain names that were contacted by 41,900 apps.

Among the 7,384 domain names, we found 1,744 (23.6%) domain names that received one or more of the types of personal data listed in Table 4.1. Each time any of the relevant data is sent by an app to a domain, we count this as a case of personal data being sent out. Specifically, we identified 28,665 apps (see the first column of Table 4.3) that sent personal data to these 1,744 domain names. We now rely on the assumption that a third party would serve multiple apps and hence flag those domains as third-party domains that are contacted by at least ten different apps. This leads us to detect 337 distinct third-party domains. We found that 28,065 (97.9% of 28,665; second column in Table 4.3) apps sent personal data to 209 third-party domains. Notably, third-party domains, representing only 12.0% of domains which received PD, are responsible for a disproportionate fraction (94.7%) of cases of receiving personal data without prior consent.

This result suggests that only a negligible number of first parties collect PD. In

	Any Domains (N=28,665)		Third-Party Domains (N=28,065)		Advertisement Domains (N=24,838)	
	High-Profile Apps	Long-Tail Apps	High-Profile Apps	Long-Tail Apps	High-Profile Apps	Long-Tail Apps
AAID	5,177 (34.6 %)	22,152 (38.7 %)	5,072 (33.9 %)	21,957 (38.3 %)	4,366 (29.2 %)	19,904 (34.7 %)
BSSID	86 (0.6 %)	107 (0.2 %)	71 (0.5 %)	88 (0.2 %)	16 (0.1 %)	12 (0.0 %)
EMAIL	48 (0.3 %)	113 (0.2 %)	42 (0.3 %)	108 (0.2 %)	—	—
GPS	459 (3.1 %)	1,151 (2.0 %)	363 (2.4 %)	946 (1.7 %)	136 (0.9 %)	244 (0.4 %)
GSF	4 (0.0 %)	3 (0.0 %)	3 (0.0 %)	1 (0.0 %)	—	—
IMEI	107 (0.7 %)	611 (1.1 %)	51 (0.3 %)	444 (0.8 %)	36 (0.2 %)	356 (0.6 %)
IMSI	22 (0.1 %)	26 (0.0 %)	8 (0.1 %)	6 (0.0 %)	—	—
MAC	68 (0.5 %)	126 (0.2 %)	30 (0.2 %)	41 (0.1 %)	27 (0.2 %)	17 (0.0 %)
PHONE	1 (0.0 %)	4 (0.0 %)	1 (0.0 %)	—	—	—
SERIAL	49 (0.3 %)	158 (0.3 %)	17 (0.1 %)	91 (0.2 %)	3 (0.0 %)	3 (0.0 %)
SIM_SERIAL	9 (0.1 %)	29 (0.1 %)	5 (0.0 %)	19 (0.0 %)	—	—
SSID	73 (0.5 %)	108 (0.2 %)	67 (0.4 %)	78 (0.1 %)	17 (0.1 %)	15 (0.0 %)
UID	1,044 (7.0 %)	4,471 (7.8 %)	938 (6.3 %)	4,236 (7.4 %)	679 (4.5 %)	3,533 (6.2 %)
Any	5,455 (36.4%)	23,210 (40.5%)	5,276 (35.2%)	22,789 (39.8%)	4,415 (29.5%)	20,423 (35.6%)

Table 4.3: Types of data and number of apps sending this to any, third-party, and ad domains (percentages relative to dataset sizes).

contrast, the majority of personal data was sent to third parties, which developers heavily rely on for a variety of purposes such as monetization (e.g., personalized ads), error logging, analytic services, user engagement, or social network integration. We note that GDPR mandates explicit consent in case such a third party acts as a data controller (rather than a data processor, that does not itself benefit from processing the data). Hence, in the following, we specifically focus on domains for which we can unequivocally determine that they control data for their own purposes, namely advertisement.

4.3.3 Identifying Advertisement Domains

Under the GDPR, all personal data processing has to have a legal justification. The first party acting as a data controller may rely on several potential legal justifications for their data processing: fulfillment of a contract, legitimate interest, or consent. This legal justification extends to any third party acting as a data processor for the app developer. Since the third party acts completely under the app developer’s control they are viewed as in the same legal domain as the first party. Meanwhile, a third party acting as a data controller would need its own legal justification to receive and process the user’s PD. As such, they cannot rely on the original controller (app developer) to be the only responsible party to obtain a valid legal basis for their processing operations, or to ensure compliance with other obligations under the GDPR, particularly regarding the exercise of data subjects’ rights. [54].

As the most prominent business case of third parties receiving and processing

user data for their own business purposes, we chose (targeted) advertising to have a conservative lower bound for the cases of GDPR violations in the wild. An app which relies on external data controllers for targeted advertising needs to explicitly ask for the user's consent to share her personal data with the third party. We found that third-party domains received 94,7% of all personal data being sent out to the Internet. In order to analyze whether this data transfer would most likely require the user's prior consent, we first need to identify whether a third party is an advertising company, and second need to differentiate between those third parties that act as data processors and those that act as data controllers.

To determine whether a party is a potential advertisement company, we first rely on Webshrinker's categorization to identify the main topic of a domain [162] for all 209 third-party domains that received personal data in our analysis. For all domains *not* flagged as ad-related, we manually review the Web pages of the domains to assess if the domain is related to a company offering in-app advertising services. For example, while Facebook is categorized by Webshrinker as a social network, they are also an advertising company, which relies on `graph.facebook.com` for advertising and tracking [133]. In this fashion, we identified 69 domains which are operated by ad-related companies. However, not all these domains actually act as data controllers under the GDPR. To distinguish between data controllers and processors, we analyzed the legal documents provided by the third parties.

Particularly, we manually analyzed the terms of service, privacy policies, developer guidelines and contracts, if available. The GDPR requires companies processing personal data to transparently provide their processing purposes and justification. We relied on the third party's legal self-assessment whether they describe themselves and their data use as a data controller or data processor. If they described their data use as mainly for their own company's gain, e.g., assembling and selling user profiles across several different apps, we would classify them as data controllers. If they limit their described data use as purely on behalf of and instructed by the app developer and if they would provide the additional necessary data processor agreement documents, we classify them as data processors. If a company's legal statements were too vague or they offered services as both data controller *and* processor, we classified them as data processors in order to conservative estimate the number of potential GDPR violations and to not unjustly notify app developers that commissioned these companies as data processors.

Out of 69 third-party domains which are operated by ad-related companies, we identified 45 domains of data *controllers* (full list in Appendix, Table A.3), which would require explicit consent to receive data. In the next section, based on these 45 ad-related domains, we present our analysis on the GDPR compliance of apps regarding consent requirements.

4.3.4 In-Depth Analysis of Violations

We now focus on the set of apps which contacted any of the aforementioned 45 domains which we determined to be ad-related data controllers. Based on these domains, we find that the vast majority of apps that contact third parties with personal data in fact send this to ad-related domains (24,838/28,065 or 88.5%, as shown in the third

column of Table 4.3). Moreover, 86.6% (24,838/28,665) of apps which sent out *any* data do so towards advertisement domains. Relative to the number of apps we could successfully analyze, this means that 34.4% of them sent out personal data to third-party data controllers, thereby violating GDPR’s mandated consent to such collection. We note that this is in light of a mere 45/1,774 (2.5%) contacted domains being flagged as advertisement domains, which shows the significant skew towards apps sending out personal data to advertisement companies without user’s explicit prior consent. Notably, there is a significant skew towards the AAID as being the most frequently transferred piece of PD. However, according to both the advertising industry [73] and Google’s own brand [71], the AAID is considered personal data and regularly requires consent before being collected by third-party data controllers.

Identifying Major Players We now turn to analyze which are the most frequent parties that receive PD. Figure 4.3 shows the top 10 ad-related domains that received personal data in our dataset, counting the number of apps that sent data towards them. We find that more than half of the apps which sent data without consent sent data to (at least) Facebook. It is noteworthy that Facebook makes GDPR compliance particularly tough for developers to implement. According to their own documentation [44], their SDK defaults to assuming user consent². That is, a developer must actively disable the automated transmission of personal data and implement their own consent dialogue. In the case of Facebook, they operate in multiple capacities (e.g., social media integration *and* advertisement), yet their terms allow data sharing between the different services in their privacy policies. Specifically, the Facebook Graph API can share its data with Facebook Ads [4], which in turn can again be used to optimize advertisement.

The current version of the SDK of the second most-prevalent recipient of data, namely Unity, supports two variants of consent [153]: either, the developer provides consent through an API call (naturally after having acquired explicit consent) or the app developer can rely on Unity’s solution which asks the user when the first ad is shown. However, as per their legal documentation, this is an *opt-out* mechanism rather than opt-in [154]. We believe this to be the major driving force behind the large number of requests towards Unity, as their ads first transmit data and then ask users to opt-out.

As for the third-largest recipient, we note that Flurry also supports a consent API, but the documentation is unclear about the default behavior and lacks important details about the proper implementation [53]. More notably, Flurry delegates the responsibility to acquire consent to the app developer. Moreover, they explicitly state that they assume any data is only sent after the developer has received consent. Overall, this implies that library providers make it very cumbersome for developers to be compliant with GDPR.

Combining Multiple Identifiers As our data indicates, the vast majority of apps send out the AAID. While this in itself is already problematic with respect to the GDPR,

²We note that not all SDKs (libraries) connect to a specific domain, and not all domains are associated with third-party SDKs or libraries. However, in our study, these identified third-party SDKs have been associated with at least one domain name, e.g., Facebook SDKs connected to the `facebook.com` domain.

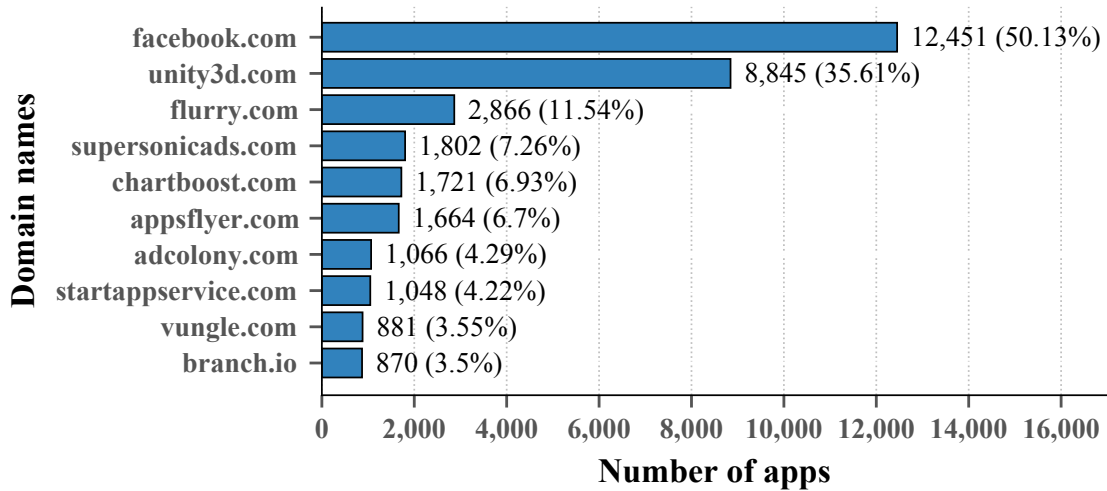


Figure 4.3: Top 10 ad domains that frequently received personal data from 24,838 apps that sent personal data to all ad-related domains.

apps sending out any other information alongside the AAID also violate the Google policy for AAID usage. In particular, according to said policy [62, 2], the AAID must only be used for advertising and user analytics. For both purposes, the AAID may not be connected to persistent device identifiers (e.g., SSID, IMEI). The AAID may only be connected to other personally-identifiable information with the user's explicit consent. In our dataset, we found that a total of 3,840 apps combined the AAID with some other type of personal information. Hence, all these apps not only infringe on the explicit consent required by GDPR, but also violate Google's policy, which means they could be removed from the app store.

For each app, we investigated to which ad-related domain they sent out the combination of the AAID and other PD. The results of this analysis are shown in Table 4.4. Note that, on purpose, we do not include the UID here, as we cannot identify whether a particular unique ID is just the AAID (e.g., hashed with an unknown algorithm). The results indicate that there are numerous domains that receive the combination of AAID and some other identifiers. Specifically, for cases such as the 190 apps that sent out the AAID with the IMEI to Flurry, Google can remove the apps without prior notice from the app store. To further understand this violation of Google's policy (combined with the fact that only relatively few apps conduct this practice), we analyzed the versions of SDKs used in apps which sent out the data to these top 5 ad-related domains. To that end, we rely on a two-step approach. First, based on the collected traffic, we identify SDK version numbers from the requests, such as GET or POST parameters (see Section A.3 for details). For apps which lack such version information in the HTTP traffic, we instead rely on a more involved analysis through LibScout [13]. We chose not to apply LibScout to all apps given the significant runtime overhead this would have caused (99.47 seconds per app for 100 randomly tested apps on macOS/Core-i7/16GB RAM).

Out of the 353 apps which contacted Flurry, we were unable to extract SDK version information for 202 apps from their traffic. For these 202 apps, LibScout successfully detected SDK versions for 53 of the apps. In particular, it detected SDK versions for 45

Domains	Data Types	No. Apps
flurry.com	IMEI	190
	GPS	156
	SERIAL	4
	SSID	2
	MAC	1
my.com	BSSID;GPS;MAC;SSID	22
	MAC	17
	GPS;MAC	2
	BSSID;GPS;IMEI;MAC;SSID	1
amazon-adsystem.com	GPS	30
unity3d.com	IMEI	29
vungle.com	GPS	26

Table 4.4: Top 5 of ad domains receiving AAID along with other personal data in our two app datasets.

of the 190 apps which sent the IMEI, all of which were pre-GDPR versions. We note that based on the release notes of Flurry [52], the feature for the IMEI collection was removed already in 2016. Since we are unable to download Flurry SDKs before version 6.2.0 (released in November 2015), it is highly likely that LibScout’s failure to detect the version stems from pre-6.2.0 versions being used by the apps in question. Hence, we believe that the IMEI collection can be attributed to extremely old versions of the SDK still in use in the apps we tested. For the versions which sent the serial, LibScout detected two out of the four SDK versions, both of which ran pre-GDPR versions. For the single detected case of sending out the MAC address, LibScout detected a version with GDPR support. Finally, for the class of apps that sent out GPS, we could detect SDK versions for 151/156 based on the traffic, and LibScout successfully detected the SDK version for the remaining five. Notably, all these apps used current versions of the Flurry SDK. However, based on the Flurry manual, it appears that if an app has GPS permissions, Flurry defaults to sending this unless the developer explicitly opts-out [51].

For the 42 cases in which `my.com` received (at least) the MAC with the AAID, we found that 23 ran SDK versions which support GDPR. However, the documentation is sparse [111] and it remains unclear if the default behavior is to collect such data (or the developer has to set `setUserConsent` to true first). For the remaining 19 cases, they all used outdated SDK versions without GDPR support. Considering the data sent out to Amazon, we find that 20/30 apps are running a current version of the Mobile Ads SDK. For the 29 cases of apps which sent the AAID along with the IMEI to Unity, these all used outdated SDKs (released before 2018 when GDPR came into effect). For Vungle, 16/26 apps which sent out GPS with the AAID ran pre-GDPR versions of the library (added in version 6.2.5 [157]). Yet, for the remaining ten, the version numbers indicated GDPR support; i.e., in these cases developers opted into sending said data.

Further, out of 24,838 apps that sent personal data to ad-related domains, we found that 2,082 (8.4%) of these apps have a pre-GDPR update date (before May 2018). We note from this analysis that the most egregious violations can be attributed both to

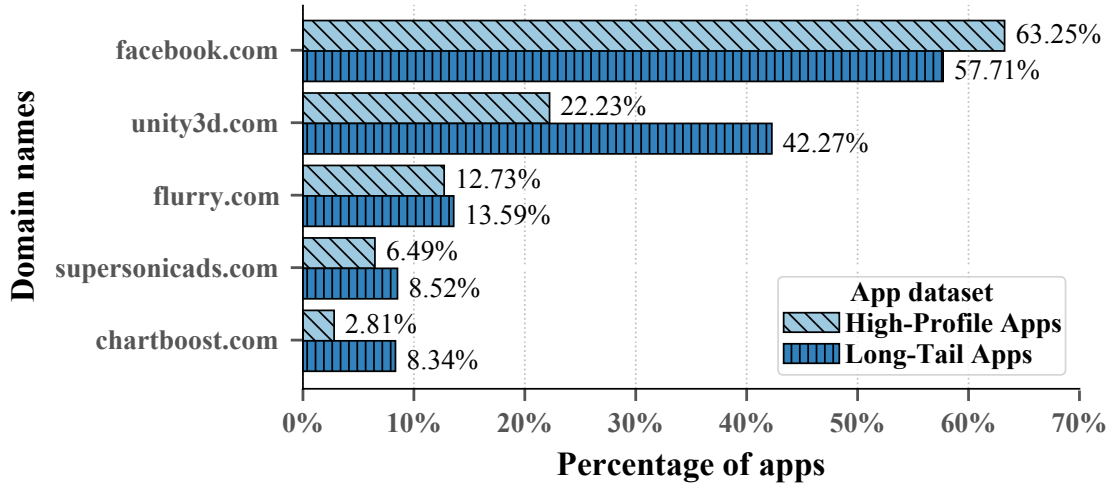


Figure 4.4: Top 5 ad domains receiving personal data in each app set after applying the Exodus-Privacy filtering to the high-profile set (percentages relative to the personal data sending apps per dataset).

extremely old versions of libraries (e.g., developers often neglect SDK updates when adding functionality [37]), but also to the complex configuration required to make apps GDPR (and Play Store)-compliant. This is particularly obvious for the collected GPS coordinates in Flurry’s SDK, which seems to be enabled by default unless developers opt-out. In the following, we aim to understand if the violations discussed thus far are specific to either high-profile or long-tail apps.

Comparing the Datasets A natural question that arises is about potential differences between the datasets. As mentioned earlier, we filtered the long-tail apps through Exodus-Privacy, which introduces a selection bias. To account for that, before comparing the datasets, we apply the same filtering to the high-profile apps. After that, we find that only 10,799 of the 14,975 high-profile apps we could successfully analyze would have passed this filtering step. Notably, we would have missed 888 high-profile apps which sent out data if we had prefiltered the high-profile apps. Assuming similar distributions of undetected ad-related libraries in the long-tail set, this is an obvious limitation of our sampling approach and should be kept in mind for future work. After applying the filtering to the high-profile set, we consider 3,527/10,799 (32.6%) high-profile apps which sent out data as well as 20,423/57,299 (35.6%) apps from the long-tail dataset in the following.

We first compare the number of apps in each dataset which send out personal data to ad-related domains. Our null hypothesis H_0 is that there is no difference between high-profile and long-tail apps in terms of sending out PD. By applying χ^2 , we find that with $p < 0.01$, H_0 is rejected. However, computing Cramer’s V ($v = 0.0228$) we find that the effect is negligible [41]. Next, we investigate to what extent the apps in the datasets differ in terms of domains to which they send PD. For this, we apply the Kruskal-Wallis test, which rejects H_0 of no difference between the sets with $p < 0.01$; however, computing the effect size $\epsilon^2 = 0.0178$, there is again only a small to negligible

effect [41]. Similarly, for the number of different types of personal data sent out, Kruskal-Wallis shows $p = 0.022$, but $\epsilon^2 = 0.0002$, i.e., again significant difference, yet negligible effect.

In addition to the overall trend, we also analyzed whether we can observe differences in the parties which are contacted by apps in each category. Figure 4.4 shows the most frequently contacted domains across both datasets together with the percentage of apps that sent personal data to them. In particular, we consider the percentages relative to the apps after the Exodus filtering step. We note that `facebook.com` is the most prevalent in both sets, yet occurs more often in the high-profile than the long-tail apps. Contrary to that, we find that `unity3d.com` is more pronounced in the long-tail apps.

By analyzing the categories of the apps, we found that Unity is frequently used in games, which are at the core of Unity’s business model. Notably, AppBrain combines all sub-categories of games into a single category, meaning our high-profile apps set contains only 500 such games. In contrast, in the long-tail apps set, almost 20% of the apps are related to games, explaining the significant skew towards Unity in that dataset. Generally, out of 72,274 successfully analyzed apps, the top 5 categories that have more violating apps than others in both app sets are *game* (73.29%), *comics* (64.97%), *social* (41.39%), *shopping* (37.04%), and *weather* (36.59%).

Overall, the results of our comparative analysis lead us to conclude that the phenomenon of sending out personal data without prior explicit consent occurs as frequently and with as many parties in both high-profile and long-tail apps. While we did observe statistically significant differences, the associated effect size was negligible. And while there certainly exists a difference between the two datasets in terms of *who* receives data, we cannot observe a difference that would warrant the hypothesis that high-profile apps violate GDPR less than long-tail ones.

Manually Analyzing Consent Dialogues To investigate whether developers may have merely misunderstood the concept of consent (or the GDPR requirements thereof), we randomly sampled 100 apps which sent out data in our experiment and checked the screenshots (which we had taken as part of our analysis to show on the Web interface). Specifically, we checked for both implicit consent dialogues (such as those indicating that by using the app, we consent to data being sent out) or explicit opt-out dialogues. We note here that even having an opt-out dialogue which – after negative confirmation – stops collecting of data still meant the app sent out data *before* asking for the user’s consent. Among these 100 apps, we found only 25 apps present any type of consent notices to users. Of these, only 11 apps provide an option to reject the data collection, while the remaining 14 apps ask users to accept the data collection to use without options to reject the data collection and sharing. Overall, this indicates that the vast majority of apps do not even attempt to achieve GDPR compliance, nor do they add meaningful ways of rejecting consent after the fact.

4.4 Developer Notification

In addition to the technical analyses described thus far, we also notified affected developers. This had two main goals: first, to inform them about the potential breach

of GDPR regulations, which may lead to severe fines [35]. Second, we wanted to gain insights into the underlying reasons that caused the observed phenomena in the first place. Since disclosing the findings to authorities (e.g., regulators, Google) might cause financial harm to developers, we consciously decided not to involve authorities but rather notify developers directly to remedy compliance issues. We note that our institution's ethics guidelines do not mandate approval for such a study.

To notify the developers, we extracted the email addresses used to upload the apps into the Play Store. To assess how many developers actually received our reports, rather than including the technical details about the issues in the email, we sent developers a link to our web interface. On this, we briefly explained our methodology of testing and showed the developers information about which hosts received which type of data (see the previous section). In addition, in our email, we asked recipients if they had been aware of the potential violation of their apps, their general understanding of what is personal data under GDPR, and their plans to address the issues as well as proposals for tool support (see Section A.2 for the full email). We decided to have this rather than a full-fledged survey, as we wanted to keep the overhead for respondents as low as possible to prompt more responses. We note that the notification was carefully worded not to make legally conclusive statements, since this could amount to legal consulting which is strictly regulated by German law.

4.4.1 Notification and Accessed Reports

Out of the 24,838 apps for which we had discovered some potential GDPR issue, 7,043 had been removed from the Play Store by the time we conducted our notification. For the remaining 17,795 apps, we sent out notifications in two batches, each with a reminder. The first batch of apps were notified on December 15, 2020, with a reminder on January 5, 2021. In this batch, we only included such apps that had not been updated since our download from the Play Store until the day of the notification, totalling 8,006 apps. We took this step to ensure that we would not notify developers who had removed the problematic code between our dataset download and notification date. For these 9,789 apps with recent changes, we re-downloaded the latest version, conducted our analysis again to confirm our findings, and added those apps to the second batch which still had some issues. The second batch of notifications was sent on January 6, 2021, with reminders on January 20, 2021. Note that we decided to give an additional week between notification and reminder for round 1 of the notifications, given the overlap with the Christmas vacation. In both cases, we grouped emails to developers (i.e., if a single developer had more than one app in the store, they only received one email with multiple links). We followed best practices established by prior work [39, 91, 146, 145] allowing developers to opt-out and not send reminders for those apps for which we had previously seen an access to the report.

In total, we notified 11,914 developers responsible for 17,795 apps. Of those developers, eight asked to be removed from our experiment. Until February 1, 2021, we saw 2,199 accessed reports. Notably, some accesses were related to spam checking (e.g., from Barracuda's IP range or clients not downloading subresources like CSS files), which we ignore in our analysis. This leaves us with accessed reports for 2,083 apps. Notably,

considering that a single owner may have multiple apps affected by the same issue, we count the overall number of apps for which their developer accessed *some* report, totalling 2,791 (15.7%) apps for which we reached their owner.

4.4.2 Developer Responses

In addition to the accessed reports and the updated apps, we also analyzed the responses we received from developers. In total, this amounted to 448 distinct senders that we classified emails for. Based on an initial set of responses, three coders developed an initial code book and then separately analyzed the entire set of responses. For all cases in which their assessment of an email/thread differed, they discussed the cases in a group until they agreed on a classification. Note that not all respondents answered the stated questions from our email ³.

Of the 448 respondents, 114 acknowledged receipt of our email and wanted to take it under advisement. 54 stated that they required further investigation, either within their respective companies or their third-party SDK vendor. 48 further inquired with us about potential solutions to the problem, such as adding privacy policies to explain the data collection. We faithfully answered these emails while stating that we cannot provide conclusive individual legal assessments. Notably, 20 respondents argued that the EU was not their main market, and that hence either GDPR would not apply to them or they did not feel the need to implement consent, either being unaware of their app being downloadable from an EEA country Play Store or that having users resident in the EU leads to applicability of the GDPR.

On the other side of the spectrum, 116 respondents disagreed with our assessment. These ranged from comments like *“i am not aware that my app might not be GDPR compliant”*, *“I show my privacy policy at the start of the app”*, *“where seems to be the problem”* to simple claims that their apps do not transmit any user information, but also argued that their advertisement libraries first ask for user consent before transmitting any data. This highlights a misconception that having a privacy policy supersedes the need to have explicit consent under the GDPR. Notably, as also highlighted by our manual analysis, many ad-related libraries sent out data before showing the consent dialogue. In the most extreme cases, developers also argued that it was infeasible for them to fully support GDPR, with one developer stating: *“I haven’t done anything wrong in my eyes. I show adverts in my Apps from BIG F***** COMPANIES like Google and Facebook and it’s up to them to tell developers what they collect so we can then pass that on to our users. Like how the hell am I supposed to know what a black box SDK from Google does with data in the App I publish to people???”*.

When asked about the data collection, 70/151 respondents (46%) said they were aware of the types of data being collected and 53/122 (43%) said they knew this data was protected by GDPR. Of the 139 respondents who answered our question regarding reasons for lacking explicit consent, 66 (47%) argued they rely on a third-party app builder or SDK to make their apps compliant and 40 (29%) believed their app to be compliant already. Ten explained their app was outdated, seven noted that they lacked resources for proper implementation, and seven said this was a bug. Finally, nine

³All developers we quote in this paper have given their explicit consent.

respondents stated there was no particular reason.

When asked about their plans to change their apps (218 answers), 136 (62%) stated to update their app, with another 29 (13%) claiming to plan to remove the app from the Play Store altogether or make it inaccessible from EEA countries. Eleven said to conduct additional research into GDPR and their responsibilities as the developers, and 17 said they did not feel the necessity to take any steps. Our data is heavily skewed towards those developers that plan to take action; we attribute this to the fact that developers that disagreed with our assessment rarely answered our follow-up questions.

Regarding our final question about developer support, we received 72 answers. Of those, 44 wanted to have an automated tool like ours to analyze their apps for compliance, while 19 asked for better documentation around how to implement GDPR compliance. Finally, nine respondents argued that third-party tools should be compliant by default (e.g., *“requiring ad providers to take responsibility for all the compliance to this unnecessarily complicated law”*). Naturally, the skew towards automated tooling is not surprising, given that we notified developers after applying our automated toolchain. It is noteworthy, though, that fewer developers answered this final question, implying that it is not even clear to them how they could be better supported in this issue.

4.4.3 Updates to Notified Apps

To assess our notification's impact on the affected apps, we downloaded new versions of all apps that had looked at our reports at least one by April 06, 2021. We re-ran our pipeline for each app with an updated version to assess if the changes were related to the reported GDPR infringement. For the 2,791 apps for which we reached a developer (i.e., they looked at at least one report for their apps), 91 apps were removed from Google Play, and 8 apps were no longer available to download from Germany. We found 1,075 apps with updates since our notification for the remaining apps. By rerunning our pipeline on these 1,075 apps, we observed that 250 (23%) apps no longer sent personal data to ad-related domains without prior consent. Considering those 136 respondents that claimed to plan to update their apps to incorporate proper GDPR consent, we found that 92 apps (for which the respondents were responsible) had been updated until the end of our experiment. Notably, though, only 43 were updated in such a fashion that they did not send out any data without interaction.

We note here that the overall number of apps which addressed the issue is low. Based on the responses we received, we believe this to have two core reasons. First, many apps are developed by small teams (if not individuals) who would rather focus on functionality updates. Second, as shown in our analysis of popular SDKs such as the one from Facebook, they do not provide a consent dialogue, but rather put the burden on the developer to integrate a new UI to ask for consent, which is then passed to the SDK. Hence, we believe the number of apps which address this issue will rise over time and the seemingly small change in overall numbers can be attributed to a lack of time to properly address the issue.

4.5 Summary

Since the General Data Protection Regulation (GDPR) went into effect in May 2018, online services are required to obtain users' explicit consent before sharing users' personal data with third parties that use the data for their own purposes. While violations of this legal basis on the Web have been studied in-depth, the community lacks insight into such violations in the mobile ecosystem.

In this chapter, we performed the first large-scale measurement on Android apps in the wild to understand the current state of the violation of GDPR's explicit consent. Specifically, we built a semi-automated pipeline to detect data sent out to the Internet without prior consent and apply it to a set of 86,163 Android apps. Based on the domains that receive data protected under the GDPR without prior consent, we collaborated with a legal scholar to assess if these contacted domains are third-party data controllers. Doing so, we found 24,838 apps send personal data towards data controllers without the user's explicit prior consent. Further, to understand the reasons behind this, we ran a notification campaign to inform affected developers and gather insights from their responses. We then conducted an in-depth analysis of violating apps as well as the corresponding third parties' documentation and privacy policies. Based on the insights from both developers and our own analysis, we found that GDPR issues are widespread, often misunderstood, and require effort from advertisement providers, app stores, and developers alike to mitigate the problems.

5

Studying Consent of Third-Party Tracking and Its Violations of GDPR

5.1. LEGAL ANALYSIS OF POTENTIAL GDPR CONSENT VIOLATIONS

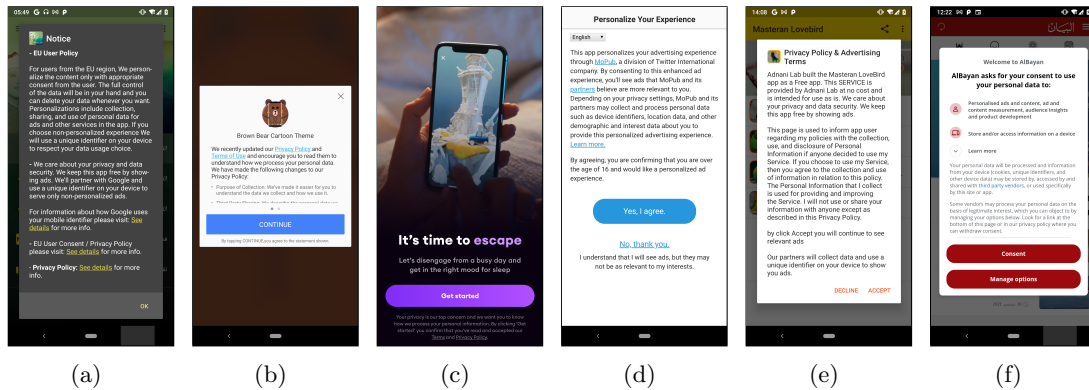


Figure 5.1: Example of various types of consent notices in Android apps.

Adopted in May 2018, the European Union’s General Data Protection Regulation (GDPR) requires the consent for processing users’ personal data to be *freely given, specific, informed, and unambiguous*. While the previous chapter has shown that this often is not given through automated network traffic analysis, no research has systematically studied how consent notices are currently implemented and whether they conform to GDPR in mobile apps.

In this chapter, we close this research gap by performing the first large-scale study into consent notices for third-party tracking in Android apps to understand the current practices and the current state of GDPR’s consent violations. In particular, we answer RQ3: *Do mobile apps implement any form of consent notices? What are the common properties of such consent notices? Can these implemented consent notices be legally justified under GDPR? Are developers aware of the GDPR consent requirements and the violations of their implementation?* To answer these research questions, we perform a large-scale study with 239,381 Android apps available through an EEA country Play Store, allowing us to provide a comprehensive overview of the consent notices currently implemented in mobile apps in the wild. Specifically, we propose a mostly automated and scalable approach using image processing and natural language processing techniques to identify the implemented consent notices and their current practices. Based on the identified mechanisms, we then extend prior work to develop a tool that automatically detects users’ personal data sent out to the Internet with different consent conditions (i.e., based on the choice mechanism to interact with the consent notice).

5.1 Legal Analysis of Potential GDPR Consent Violations

As a result of in-depth legal analysis, we aim to systematically study the following potential legal violations in Android apps specific to GDPR consent requirements. In addition, we cited expert-generated legal sources to argue whether the declared practices violate the aforementioned regulations.

5.1.1 Lack of Consent Notices

Lack of Consent Notices *Apps transmit users' personal data with third-party data controllers for advertising purposes without implementing explicit consent notices.*

This practice violates the requirement of GDPR consent which mandates mobile apps to obtain users' explicit consent before sharing users' personal data with third-party data controllers [42]. As such, due to a lack of consent notices, these apps do not have a valid legal basis for processing personal data under GDPR. Therefore, collecting and processing users' personal data without implemented consent notices is violated GDPR. The user has to be explicitly asked to consent to personal data processing for advertising purposes through a statement or by a clear affirmative action, and this consent must not be grouped with, e.g., consent to download the app or consent to access certain APIs on the phone, or "consent" packaged in terms and conditions or privacy policies.

Further, GDPR requires consent to be *informed*. In particular, it must fulfill certain conditions: "*The controller shall take appropriate measures to provide any information [...] relating to processing to the data subject in a concise, transparent, intelligible and easily accessible form, using clear and plain language, in particular for any information addressed specifically to a child [GDPR Art.12(1)]*". Notably, Article 29 Working Party [126] states that providing information to data subjects prior to obtaining their consent is important to enable them to make informed decisions, understand what they agree to, and exercise their right to withdraw their consent — if the controller does not provide accessible information, user control becomes illusory, and consent will be an invalid basis for processing.

Besides, the consent must be *unambiguous*: "*If the data subject's consent is given in the context of a written declaration which also concerns other matters, the request for consent shall be presented in a manner which is clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language [GDPR Art. 7(2)]*". Therefore, the user's consent must be easily differentiated from other declarations or even consent to other processing activities.

5.1.2 Sharing Data Before Given Consent

Sharing Data Before Given Consent *Apps implement consent notices to obtain users' consent for sharing personal data, but the apps transmit data before any given explicit consent by users.*

This practice violates the requirement of *explicit consent*. Under GDPR, the data controller should obtain verbal or written confirmation about the specific processing [Recital 32]. Article 29 Working Party states that consent can not be based on an opt-out mechanism (e.g., users have to withdraw their by default opted-in consent by turning off personalized ads through their device settings), as the failure to opt-out is not a clear affirmative action [126]. Instead, the user has to actively give their consent, i.e., by clicking "I agree" on a consent form. Merely continuing to use an app or other passive behavior does not constitute explicit consent.

Lastly, this practice further violates the requirement of *prior consent* which requires that the consent has to be obtained prior to any processing activity of personal data to be considered valid [126]. For example, neither of the consent dialogues in Figure 5.1

is valid consent under GDPR if the data sharing occurs before the user has explicitly given their consent.

5.1.3 No Way to Opt Out

No Way to Opt Out *The consent notice does not offer a way to refuse consent. The most common case is a consent notice that simply informs the users about the app’s data share.*

This practice violates the requirement of *unambiguous* consent, which specifies that the users must have given consent through a statement or by a clear affirmative action (GDPR Art. 4(11) [60] and Art. 7 [59]). According to Article 29 Working Party [126], as a general rule, the GDPR prescribes that if the data subject has no real choice, feels compelled to consent, or will endure negative consequences if they do not consent, then consent will not be valid. Accordingly, consent will not be considered “free” if the data subject is unable to refuse or withdraw their consent without detriment. Furthermore, any element of inappropriate pressure or influence upon the data subject (which may be manifested in many different ways) which prevents a data subject from exercising their free will, shall be invalid consent.

As shown in Figure 5.1 (a), Figure 5.1 (b), and Figure 5.1 (c) none of the apps provide any meaningful ways of giving or refusing consent to the sharing of personal data. The only option to use the app is to agree to share personal data as described in the consent dialogues or packaged in the privacy policies. Since users cannot use the app without consenting to these purposes (has to choose between giving consent or uninstalling the app), the consent cannot be considered as being “*freely given*”.

5.1.4 Non-Respect of Choice

Non-Respect of Choice *Apps transmit users’ personal data with third-party data controllers for advertising purposes after users explicitly rejecting/opting-out consent.*

The legal analysis of Matte et al. [103] shows that this practice violates the lawfulness principle established in Articles 5(1)(a) and 6(1) of the GDPR: for the processing to be lawful, it must be based on a legal ground. The EDPB further specifies that “*if the individual decided against consenting, any data processing that had already taken place would be unlawful*” due to lacking legal basis for processing. For example, the apps in Figure 5.1 (d) or (e) will be violated GDPR legislation, if data sharing still occurs after the user has explicitly rejected the data sharing.

5.2 Methodology

Our goal is to systematically study the current practices of consent notices implemented in the mobile ecosystem and then examine whether they conform to GDPR. Although identifying cookie consents (i.e., ePrivacy Directive) on the Web has been studied in-depth, little research has been done to systematically study GDPR consent implementations in the Android apps. Recently, Kollnig et al. [87] first attempted to study the absence of consent notices to third-party tracking from a small set of Android apps (i.e., 1,297 apps) by manually inspecting each app’s user interfaces. However, such

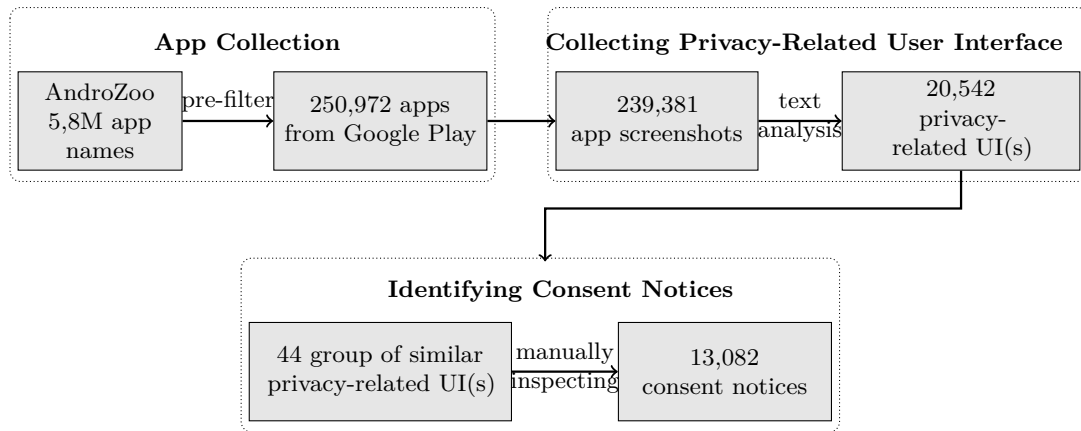


Figure 5.2: Overview of the methodology to identify consent notices in Android apps and the (intermediate) results.

a manual process is insufficient in identifying and studying the status quo of currently implemented consent notices and the potential GDPR consent violations in Android apps at scale. Furthermore, consent notices currently found vary in their appearances (e.g., the consent dialogues display the same notices for sharing data may look very different in many apps) and the underlying functionality in mobile apps (see Figure 5.1). For example, the consent notices in Figure 5.1 (a), (b), and (c) only display a notice that informs users about the collecting and sharing of users’ data without further information, and there are no ways to reject the data sharing from these consent notice user interfaces; the consent dialog in Figure 5.1 (f) shows other types of consent mechanisms from third-party providers that offer complex opt-in choices (e.g., consent management platforms).

More specifically, we first propose a mostly automated and scalable solution to identify currently implemented consent notices in Android apps in the wild. We use real Android devices to run each app (without interactions with the app user interface) and take the app screenshots (apps’ user interfaces). The underlying assumption is that the app has to show the consent notices before sharing data to be legally compliant, which is the first time users open the app. From the collected app screenshots, we then perform the image processing and natural language processing techniques to identify the privacy-related user interfaces, which potentially are consent notices for sharing users’ personal data with third parties (Section 5.2.1). Then we conduct the clustering analysis on these privacy-related user interfaces to group them by their similarity. Finally, based on the clustering results, we carefully manually verify each group to identify any form of consent notices (Section 5.2.2). In the following, we outline how we conduct each of the steps in more detail.

5.2.1 Collecting Privacy-Related User Interface

We aim to cover all different forms of consent notice user interfaces currently implemented in the Android apps in the wild such as self-implemented consent notices by app developers, consent management platforms (CMP), or consent SDKs from third-party

providers. Therefore, it would be insufficient if we based on any specific consent notice designs or user interfaces to identify the currently implemented consent notices. In fact, most of the consent notices contain a certain keyword that is related to privacy, such as “*privacy*”, “*privacy policy*”, “*data policy*”, or “*gdpr*” [36, 155]. However, not all apps’ user interfaces with such keywords are supposed to be consent notices (e.g., app developers simply added the “privacy policy” text that navigates the user to the privacy policy page, which is not a consent notice). Therefore, our first step aims to collect such privacy-related user interfaces, which potentially contain consent notices for sharing the users’ personal data with third-party services.

Specifically, to collect privacy-related user interfaces in Android apps, we first install the app in question and then open it but do not interact. We then take the app screenshot (i.e., an app’s user interface is everything the user can see and interact with) after waiting for five seconds for the app to be fully initialized. The underlying assumption is that the app should show the consent notices before sharing data, which is the first time users open the app. To do that, we rely on *six* rooted devices (Pixel¹, Pixel 3a, and Pixel 6) running Android 9 or 12 to analyze a given app. Recall that our goal is to systematically study the current practices of consent notices in the wild at scale. Hence, relying on static analysis techniques, which may produce a vast amount of false positives and even could not understand what the actual user interface of such consent notices are, is not an option [158, 93, 22].

Finally, we extract the text from the collected apps’ screenshots by using optical character recognition (OCR) [150]. We then perform the string-matching with the privacy-related keyword list from the prior work Degeling et al. [36] (i.e., which contained phrases from all 24 official languages, plus four EU languages) to identify whether the screenshots are privacy-related user interfaces.

5.2.2 Identifying Consent Notices

After collecting the privacy-related user interfaces, we now want to detect if they are any form of privacy notices or consents for sharing data with third-party services. However, not all privacy-related user interfaces are consent notices (e.g., the app presents only the “privacy policy” text on the screen). Moreover, it is practically impossible to manually analyze all collected privacy-related user interfaces to identify the consent notices. Therefore, we apply a mostly automated and scalable approach using natural language processing techniques to identify the consent notices at scale. Generally, we first perform the clustering technique based on the extracted texts from these privacy-related user interfaces to group them by their visual representation and content. Based on these groups, we then carefully manually inspect each group to identify any form of privacy notices or consents and systematically study the current practices of such consent notices. This approach allows us to verify a large number of apps in manageable ways, and allows us to provide a comprehensive overview of the consent notices currently implemented on mobile apps at scale.

More specifically, we apply several text processing methods to analyze extracted natural language text from collected privacy-related user interfaces. These methods are

¹The first generation of Pixel smartphones.

broadly classified into three primary tasks: text preprocessing, feature extraction, and clustering. In the following, we outline how we conduct each step in more detail.

5.2.2.1 Text Preprocessing

We first apply the following widely-used text preprocessing techniques [78, 161, 100, 113]: correcting misspelling from ORC errors through *autocorrect* [48] (e.g., "*imagec*" to "*image*"); normalizing and lemmatizing all words, e.g., removing punctuations, converting letters to lowercase and reducing the inflectional forms of a word (e.g., "*sent*", and "*sending*" to "*send*"); and removing generic stop words such as "*are*" and "*the*"; lastly we remove words that are not generic stop words but are specific to the app such as app name, package name, number, and date time.

5.2.2.2 Feature Extraction

We then use a bag-of-words model to extract features from the preprocessed texts of privacy-related user interfaces [113, 143]. In particular, let $T = \{t_1, t_2, \dots, t_n\}$ be a set of all unique terms in the corpus of privacy-related user interfaces. A text feature vector of the i^{th} privacy-related user interface is denoted as $t_i = \{t_1, t_2, \dots, t_k\}$. For example, the raw text is "*We use device identifiers to personalise content and ads*", after applying the preprocessing, the generated preliminary text vector is: $t = \{\text{"device"}, \text{"identifier"}, \text{"use"}, \text{"ad"}, \text{"personalise"}, \text{"content"}\}$.

Additionally, we employ the hypernym strategy to resolve synonyms and introduce more general concepts for identifying related topics [78] (i.e., added to each term of the feature vectors all subconcepts of the five levels below it based on Wordnet corpus [108]). Finally, we perform word stemming on all terms (e.g., "*personalise*" and "*personalising*" to "*personalis*"). For instance, with the text vector $\{\text{"personalise"}\}$, the final text feature vector will be $\{\text{"personalis"}, \text{"person"}, \text{"individu"}, \text{"individualis"}\}$.

Finally, we convert each term, in the text vector to numeric one by using the term-frequency inverse document-frequency (TF-IDF). The TF-IDF value for each element is calculated as:

$$tfidf(t, d) = tf(t, d) * idf(t) = \frac{1 + N}{1 + df(d, t)}$$

where t refers to the selected term, d refers to the text vector, tf is the absolute frequency of a term, i.e., $tf(t, d)$ is the number of times a term t occurs in a given d , idf is the term's inverse document frequency, N is the number of text lists in the corpus, and $df(d, t)$ returns the number of text lists that contain the target term t .

5.2.2.3 Hierarchical Clustering Analysis

Lastly, we use agglomerative hierarchical clustering to identify similar privacy-related user interfaces. Specifically, we use Ward's method [160], and the similarity score or distance between two vectors is calculated by cosine similarity:

$$\text{similarity}(\vec{v}_i, \vec{v}_j) = \cos(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{\|\vec{v}_i\| \cdot \|\vec{v}_j\|}$$

5.2.2.4 Manually Identify Consent Notices

Finally, this leaves us with groups of privacy-related user interfaces for identifying consent notices. We now manually inspect each group and classify any form of information about data practices as a privacy notice and any affirmative user agreement and action to data practices as consent (i.e., based on the current practice in the field [87]). In fact, the GDPR consent requirements are far more specific, and strict standards must be adhered to. However, this is an intended option to increase the objectivity of our classification to identify the implemented consent notices. Further, we argue that it does not affect the validity of our results regarding the identified potential GDPR consent violations in Section 5.3.

5.3 Large-Scale Analysis

This section presents the results of our empirical study of Android apps on Google Play regarding the currently implemented consent notices and whether they are obtained legally under GDPR requirements. More specifically, we first outline how we construct the app dataset for our analysis (Section 5.3.1) and subsequently present our consent analysis results (Section 5.3.2). We note that we performed all technical testing and experiment in EEA country where the GDPR applies, i.e., our geolocation is EEA country and the Play store is set to the EEA country variant accordingly. Based on the identified consent notices, to empirically study the potential violations of GDPR consent requirements (outlined in Section 5.1), we further perform a network traffic analysis to detect apps actually send users' personal data to third-party data controllers for advertising purposes (see Section 5.3.3). This allows us to ensure the potential violations indeed took place. Finally, we report the observed potential violations in Section 5.3.4.

5.3.1 App Dataset Construction

Our analysis aims to assess the state of potential GDPR violations in Android apps in the wild. Therefore, we crawled all free Android apps from October 2021 to March 2022 on the Google Play store (EEA country location-based) based on the list of apps from AndroZoo [3] (which has 5,8M of Android apps' names). As such, we cannot generalize our findings to paid apps by limiting our analysis to free apps. However, this is also in line with previous large-scale Android security research [121, 49, 149, P2]. We further applied the following filter to get the most relevant apps to our study (e.g., filtered out apps that developers do not maintain; focusing on apps that have the capacity to access users' personal data, such as persistent unique identifiers on users' devices). Notably, we consider those apps that meet the following conditions:

- Apps have at least 10,000 downloads (i.e., the popularity of the apps). This factor allows us to regard our findings to represent widespread practices of the potential

GDPR consent violations and their effects on millions of users.

- Apps request sensitive permission such as GPS location, contact [69]. These apps have the capacity to access and share highly sensitive information (which are considered personal data under GDPR) to third-party services.
- Apps have the latest update later than May 2018, when GDPR went into effect, i.e., app developers have not maintained the outdated apps, and most of them violated the legislation [P2]. Therefore, our study does not include those apps with the latest update before May 2018.

As a result, we obtained 250,972 Android apps. In the next step, we present how to identify the consent notices on these apps.

5.3.2 Identifying Consent Notices In The Wild

Figure 5.2 shows an overview of our methodology for identifying consent notices in 250,972 Android apps and the intermediate results. More specifically, our technique suffers from certain limitations which keep us from analyzing all apps in the dataset. In particular, we successfully analyzed about 239,381 (95.38% of 250,972) apps by using dynamic analysis (i.e., install the app, open the app, take a screenshot). Unfortunately, the remaining 11,591 either crashed or detected the analysis environment, which potentially affects the completeness of our results.

Out of the 239,381 successfully analyzed apps, we identified 20,542 privacy-related user interfaces using the keyword matching techniques. Furthermore, based on these 20,542 collected privacy-related user interfaces, we identified 44 groups of similar user interfaces. From 44 groups, we then manually inspected each group to filter out those without consent notices (e.g., UI has only the “privacy policy” text). Finally, for the remaining 36 groups, we manually inspected each UI to identify consent notices (e.g., any form of information about data practices, any affirmative user agreement). As a result, we identified 13,082 privacy-related user interfaces that are any form of consent notices.

To select the best threshold for identifying similar privacy-related user interfaces, we considered the tradeoff between the quality of the clustering against the number of clusters. By manually inspecting the cluster quality and testing with different distances ranging from 50 to 150 based on the dendrogram of our hierarchical clustering in Figure 5.3, we have the best quality result at the height 62. The best quality cluster means that the majority of privacy-related user interfaces in the same group would be more similar to each other than in another group. Besides interpreting the dendrogram, we used the silhouette score to evaluate the quality of clusters. By inspecting the silhouette plot, we found that the first highest was at 9 clusters, and the second highest was at 44 clusters. Then the number of clusters started to increase significantly in correlation with the value of the silhouette score.

Recall that our goal is first to understand the current practices of implemented consent notices in the Android app market (which no study has systematically analyzed before) and then to examine whether they conform to GDPR. Therefore, we not only reported whether an app displayed a consent notice (based on 36 groups), but also

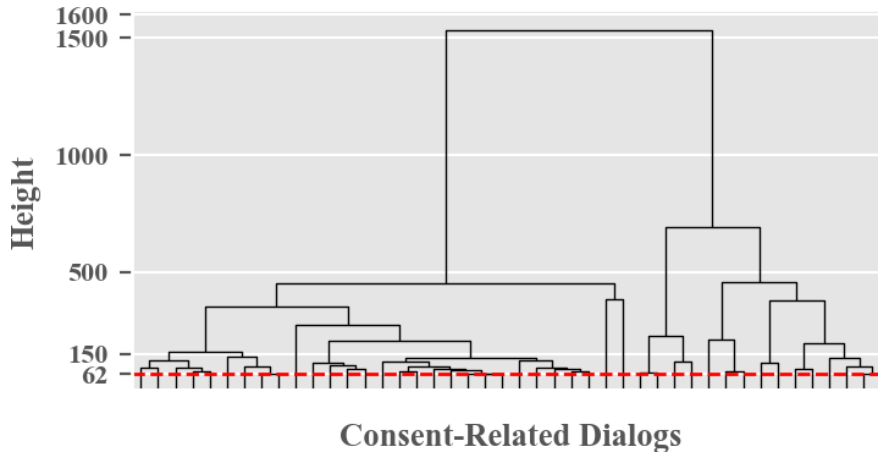


Figure 5.3: The dendrogram of our hierarchical clustering.

analyzed and categorized the types of consent notices based on their interaction options. In particular, by manually inspecting these screenshots of identified consent notices, we identified the four mechanisms for user interaction that are currently widely implemented by Android apps² — the classification is based on the definition of prior work on the Web area [36]. Figure 5.4 shows some example of each mechanism. Based on the choice mechanism to interact with the consent notice, we then built an automatic tool to empirically study the potential violations of GDPR (Section 5.3.3).

- **Confirmation-only:** 5,740 (43,87%) consent notices display a UI element (e.g., a button, checkbox) with an affirmative text such as “OK”, “I agree”, “Start”, clicking on which is interpreted as an expression of user consent.
- **Opt-out personalized ads:** 3,949 (30,19%) consent notices implement a simple choice to refuse consent limit to personalized advertising only, but this does not necessarily prevent the tracking.
- **Binary choices:** 2,871 (21,95%) consent notices have two or more UI elements, but mainly offer two main functionalities which allow users to either accept or decline the data sharing on the app.
- **Complex choices:** 522 (4%) consent notices provide complex opt-in choices such as consent management platform (CMP), consent SDKs from third-party services.

Overall, our classification results provide a comprehensive understanding of the kinds of consent notices in the current Android app market. First, as our data indicate, the confirmation-only consent notices are widely implemented in Android apps (i.e., 43,87%), in which the users have no ways to reject the data sharing. The only option is

²We note that the clustering of similar privacy-related user interfaces is based on the consent notices’ full content, i.e., including the explanation text and the consent choices. As such, when we manually categorized 36 groups of consent notices based on their interaction options, the same consent interaction mechanism may contain multiple groups (e.g., the things that make it different are the text of the choices, such as the choices of the confirmation-only group could be “Start”, “Agree”, “Next”).

CHAPTER 5. STUDYING CONSENT OF THIRD-PARTY TRACKING AND ITS VIOLATIONS OF GDPR

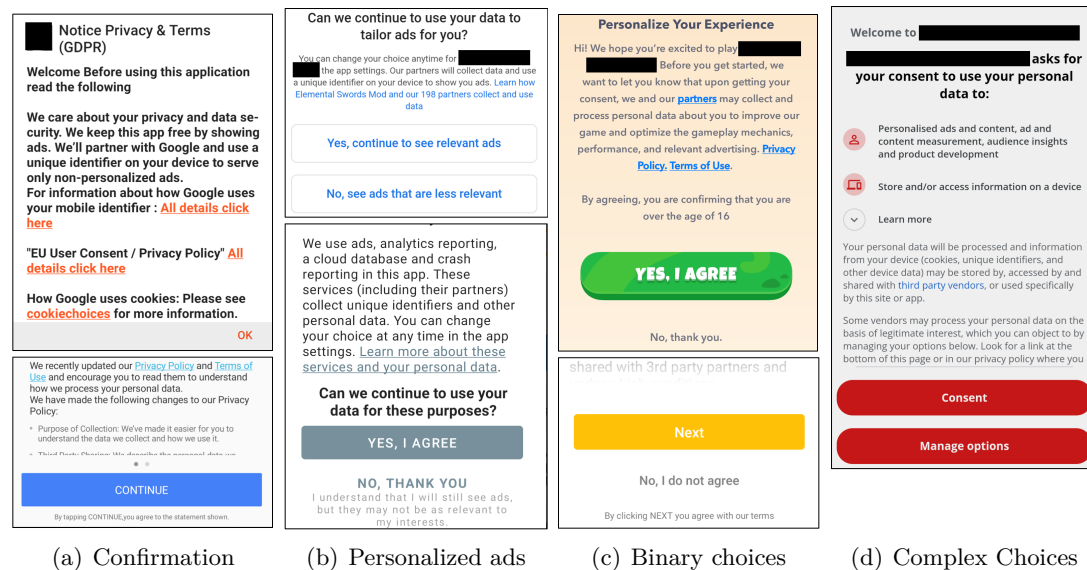


Figure 5.4: Example of the four types of consent choices.

to agree with data sharing as described in the consent notices, privacy policies, or terms and conditions. If not, users have to uninstall the app, such a practice which cannot be considered “*freely given*” under GDPR. Secondly, 30,19% of apps implemented the opt-out personalized ads consent notices. However, such a choice might make users incorrectly assume that refusing to see personalized ads prevents all tracking [87]. Notably, there are not so many apps that provide users with reject entirely the data sharing (i.e., 21,95% of binary notices, and 4% of the complex choices, which also not easy to reject all of the data sharing at one time).

GDPR Article 6 contains six general justifications for collecting and processing users’ personal data. As such, even though those 13,082 apps implemented any form of consent notices, they may be based on other legal grounds for processing users’ personal data. Thus they do not legally require consent under GDPR. In order to justify whether those consent notices are legally compliant, only legal experts and authorities can make the decision, and we exclude such discussions from this work. Instead, we specifically focus on those apps that implement consent notices and send users’ personal data to third-party advertising data controllers, which mandate users’ consent under GDPR. Doing so, we ensure that a potential violation indeed took place and avoid the case where the apps relied on using legitimate interests as a legal basis for processing users’ personal data. Hence, in the following, we further perform a network traffic analysis to detect apps that send users’ data to third-party data controllers for advertising purposes.

5.3.3 Automating Potential GDPR Violations Detection

To empirically study the potential violations of GDPR consent requirements in Android apps (outlined in Section 5.1), we further perform a network traffic analysis to detect

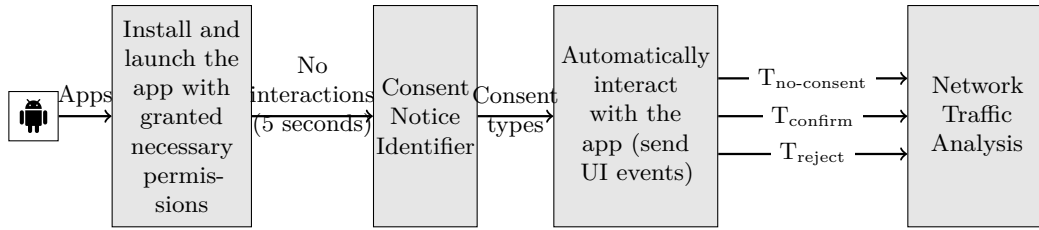


Figure 5.5: Overview of our methodology to detect data sent out to the Internet with different given consent conditions.

apps actually send users’ personal data to third-party data controllers for advertising purposes. In particular, we extend prior work ([P2]) to build an automatic dynamic analysis tool that detects apps sending data to the Internet with different given consent conditions. Based on the collected network traffic and the identified consent notices, we then perform a legal analysis of potential violations of GDPR consent requirements (Section 5.3.4). Generally, we measure the following conditions: (1) we look for apps that send users’ personal data to third-party data controllers for advertising purposes to see whether they implement any consent notices; (2) for those that have implemented consent notices, we further investigate whether they send data before any given consent; (3) whether they allow users to refuse the consent; (4) and finally, those apps where we explicitly opt-out from consent and they still share data.

We followed best practices established by prior work [136, 133, 77, P2] to collect network traffic for identifying third-party services and privacy leaks in Android apps. In order to intercept the TLS traffic, our six rooted devices were instrumented with our own root certificate (i.e., by using MitM proxy [33]), and the given app was instrumented to detect and disable SSL Pinning by using *objection* [117]. To interact with the app automatically, we then extended the lightweight test input generator that sends random or scripted input events/UI interactions to the app based on our configurations (i.e., DroidBot [97]). The collected network traffic will be stored in our database for later analysis.

Figure 5.5 shows an overview of our methodology to detect data sent out to the Internet with different given consent conditions. Generally, we first install the app and then grant all apps’ requested permissions listed in the manifest, i.e., install and runtime permissions. Subsequently, we launch the app, run it up to 150 seconds based on different given consent conditions, and record its network traffic. We note that, between each condition, we uninstall the app and clear all of the app stored data to ensure the apps show consent notices each time (i.e., the consent notice may not show again when the users have already made their choices).

In the following, we outline how we analyze the network traffic of 13,082 apps (that implemented consent notices, in Section 5.3.2) based on each consent condition.

- **No consent:** First, we aim to detect apps’ network traffic without users’ explicit consent ($T_{\text{no-consent}}$ traffic). To achieve this, we simply open the app but do not interact with it at all. The underlying assumption is that if network traffic occurs when this app is opened without any interactions, we have naturally not consented explicitly to any type of data collection by third parties.

Choices	Text	Keyword
Confirm	“yes”, “agree”, “ok”, “continue”, “consent”, “confirm”, “understand”, “start”, “okay”, “enable”, “got it”, “go”, “join”, “next”, “understood”, “play”, “allow”	
Reject	“no, thank you”, “decline”, “no, see ads”, “no, thanks you”, “no, thank”, “cancel”, “opt out”, “disagree”, “no”, “not”, “refuse”, “deny”, “exit”, “cancel”, “postpone”, “later”, “do not”, “reject”	

Table 5.1: Text keywords based on each type of consent choice.

Data Type	Description
AAID	Android Advertising ID
BSSID	Router MAC addresses of nearby hotspots
Email	Email address of phone owner
GPS	User location
IMEI	Mobile phone equipment ID
IMSI	SIM card ID
MAC	MAC address of WiFi interface
PHONE	Mobile phone’s number
SIM_SERIAL	SIM card ID
SERIAL	Phone hardware ID (serial number)
SSID	Router SSIDs of nearby hotspots
GSF ID	Google Services Framework ID

Table 5.2: Types of personal data we consider in our work.

- **Confirm consent:** Second, we detect apps’ network traffic after accepting the consent (T_{confirm} traffic). Based on the identified consent notices in Section 5.3.2, our tool will first automatically identify the current user interface is consent notice or not and then click on the UI element that indicates the acceptance on the consent user interface (see the first row in Table 5.1) by using the string-matching technique and then afterward randomly navigating the apps.
- **Reject consent:** Third, we also aim to detect apps’ network traffic after rejecting the consent (T_{reject} traffic). We now configure our tool to click on the UI element that indicates to reject consent (see the second row in Table 5.1) and then record its network traffic while running.

Additionally, we further want to detect apps that transmit users’ personal data with third-party data controllers for advertising purposes without implementing any form of consent notices, which is potentially violated the outlined “*Lack of Consent Notices*” in Section 5.1. Therefore, from the 239,381 successfully analyzed apps, we excluded those apps that are in the set of 13,082 apps, and then automatically analyzed the remaining apps to collect their network traffic.

Finally, we extend the code from the previous chapter (Chapter 4) to search for the users’ personal data in the outgoing network streams originating from each device

(Table 5.2 listed types of personal data), and to identify third-party domains operated by ad-related companies (i.e., based on list of 45 advertisement domains of data controllers in Chapter 4).

5.3.4 Observed Potential Violations

In this section, we present the results of our empirical study of 239,381 Android apps on Google Play regarding the potential violations of GDPR consent requirements.

5.3.4.1 Overview of Network Traffic Analysis

Out of the 239,381 successfully analyzed apps, we identified 101,484 (42.39% of 239,381) apps that contacted to 20,968 unique fully-qualified domain names by either sending or receiving some data in our experiment. Is it known that a single registerable domain may use many subdomains (e.g., `api2.branch.io`, `api.branch.io`). Therefore, to normalize these hosts to their registerable domain (`branch.io` in the above cases), we rely on the public suffix list [131]. As a result, we identified 14,209 registerable domains (referred to as “domain names”) that were contacted by those 101,484 apps. Notably, we identified 41,639 (41.03% of 101,484) apps sent users’ personal data to 1,484 domain names.

Those 1,484 domain names that receive personal data may operate with their own privacy policies and further share the data with their partners, which could be broadcast to large numbers of different companies. Therefore, to understand how personal data may be processed, the legal basis for processing, and whether the processing is compliant with GDPR, we have to read the entire privacy policies of all the involved partners of those services. However, it is infeasible to conduct such an in-depth analysis of hundreds of privacy policies. Therefore, in the following potential GDPR consent violations analysis, we primarily focus on the list of 45 domains from [P2] that are operated by advertising companies and hence definitely act as data controllers. We leave an automated analysis of privacy policies and assessment of potential data controllers to future work. In that case, our framework would be able to detect more potential violations. By using the conservatively established list from prior work, we are confident to not suffer from any false positive, yet our results naturally only serve as a lower bound of potential violations.

In the following subsections, we will now elaborate on the analysis results of each potential violation outlined in Section 5.1 in greater detail.

5.3.4.2 Lack of Consent Notices

We detected 32,341 apps sent users’ personal data to 43 of those 45 third-party ad-related domains (data controllers) from [P2], which would require explicit consent to receive users’ personal data. However, we identified a significant number of 30,160 (93.26% of 32,341) apps have not implemented any form of consent notices. Figure 5.6 shows the top 10 ad-related domains that received personal data without consent notices in our dataset, counting the number of apps that sent data to them. The potentially violated apps occur across different app categories, such as the top 5 categories that have more

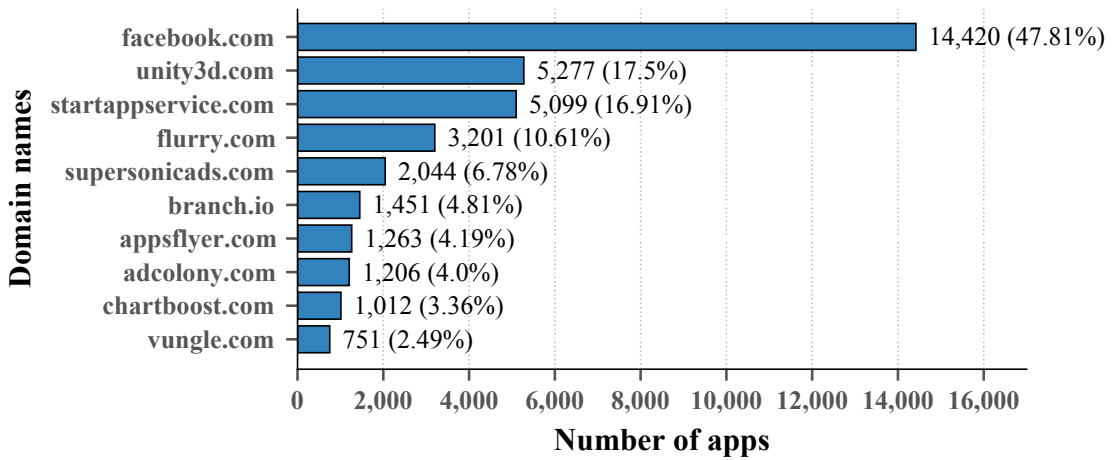


Figure 5.6: Top 10 ad-domains that received personal data from 30,160 apps that have no consent notices.

	Sharing Data Before Given Consent (N=2,181)	No Way to Opt-Out (N=1,084)	Non-Respect of Choice (N=134)
Confirmation-only	929 (42.60%)	1,084 (100%)	–
Opt-out of personalized ads	609 (27.92%)	–	48 (35.82%)
Binary choices	466 (21.37%)	–	28 (20.90%)
Complex choices	177 (8.12%)	–	58 (43.28%)

Table 5.3: Types of consent notices and number of potentially violated apps (percentages relative to each violation type).

potential violating apps than others are GAME (20.51%), ENTERTAINMENT (8.10%), EDUCATION (5.65%), PERSONALIZATION (5.13%), and BOOKS & REFERENCE (4.96%).

Our results show a significant number of apps that potentially violate GDPR consent requirements, specifically due to a lack of legal basis for processing. For this, we provide empirical evidence of a widespread *lack of consent notices* in the Android app market, which is not given through automated network traffic analysis on GDPR violations by prior work [P2]. Overall, our data indicate that the vast majority of apps do not even attempt to achieve GDPR compliance. It could well be that developers are unaware of the data sharing and the need to obtain user consent for such data sharing and collection.

5.3.4.3 Sharing Data Before Given Consent

Out of 13,082 apps that implemented consent notices, we identified 3,007 (23%) apps that sent users’ personal data to the Internet before any given consent (in $T_{no-consent}$). Notably, 2,181 (16.67% of 13,082) apps sent personal data to third-party data controllers for advertising purposes which mandates explicit consent. It could be that the developers bundle the data sharing with consent to use the app (i.e., confirmation-only consents).

	Lack of Consent Notices (N=30,160)	Sharing Data Before Given Consent (N=2,181)	No Way to Opt Out (N=1,084)	Non-Respect of Choice (N=134)
AAID	29,952 (99.31%)	2,166 (99.3%)	1,077 (99.4%)	134 (100%)
BSSID	30 (0.1%)	—	—	—
EMAIL	1 (0.0%)	—	—	—
GPS	524 (1.74%)	23 (1.1%)	13 (1.2%)	2 (1.5%)
IMEI	336 (1.11%)	23 (1.1%)	10 (0.9%)	3 (2.2%)
MAC	214 (0.71%)	39 (1.8%)	27 (2.5%)	3 (2.2%)
SERIAL	3 (0.0%)	—	—	—
SSID	31 (0.1%)	—	—	—

Table 5.4: Types of data and number of apps sending this data to ad-related domains (percentages relative to each violation type).

We found that 42.6% of 2,181 apps implemented the confirmation-only consent type. However, for such cases, in Section 5.1, we show that personal data transfer must only occur after the user has actively consented (e.g., by clicking accept), such “consent” packaged in terms and conditions or privacy policies are not compliant, which render invalid consent under GDPR.

On the contrary, a large number of 57.4% potential violating apps implemented other consent types (i.e., including 27.92% opt-out of personalized ads, 21.37% binary choices, and 8.12% complex choices). However, even though they provide choices for the users to reject the data sharing, but the apps behave differently compared to the consent notice user interfaces. Weir et al. [163] surveyed app developers and observed that most developers’ changes were cosmetic due to the GDPR legislation (e.g., adding dialogues). We confirm and evidence the existence of this widespread problem among app developers and further highlight that such cosmetic changes do not fulfill the legal conditions for collecting valid consent under GDPR. Further, this may be caused by app developers’ misconfiguration of third-party libraries. The other likely reason is that third-party consent platforms and services employ the opt-out mechanism rather than opt-in. Their services first transmit data and then ask users to opt-out, or send data along with a flag indicating the opt-out. However, at the very least, there is no technical reason why this would even be necessary, and this opt-out mechanism is also not compliant with the GDPR consent requirements.

5.3.4.4 No Way to Opt Out

From 5,740 apps that implemented the confirmation-only consent type, we identified 1,387 of them sent personal data to the Internet. Another 1,084 of them sent to third-party advertising data controllers. For such apps, the only option to use the app is to agree to the sharing of personal data as described in the privacy policies. If the data subject has to choose between giving consent to third-party processing for profiling and behavioural advertising purposes, or uninstalling the app, the consent cannot be considered to be “*freely given*”.

5.3.4.5 Non-Respect of Choice

We found 134 apps that still sent users' personal data to third-party advertising data controller after explicitly opting-out the data sharing from the consent user interface. Also, surprisingly, these types of potential violations can even occur in popular apps with millions of installs. For example, we find an app that has more than 5M installs that sent the AAID along with other persistent identifiers (i.e., IMEI, MAC) to the same third-party advertising data controllers. Other apps with more than 100M installs still shared the AAID with the third-party data controller for advertising purposes after explicitly rejecting the consent.

5.3.4.6 Summary of Observed Potential Violations

In summary, we perform a large-scale analysis of the potential violations of GDPR consent requirements on a set of 239,381 Android apps in the wild. Doing so, we identified 30,160 apps do not even attempt to implement consent notices for sharing users' personal data with third-party data controllers, which mandate explicit consent under GDPR. In contrast, out of 13,082 apps implemented consent notices, we identified 2,688 (20.54%) apps violate at least one of the GDPR consent requirements (i.e., an app could violate more than one GDPR consent requirement). In particular, 2,181 (16.67% of 13,082) apps sent personal data to third-party data controllers before given explicit consent. Among these 2,181 apps, 42.6% of apps are from the confirmation-only group (see the first column of Table 5.3). On the other hand, 1,084 (8.28% of 13,082) apps sent to third-party data controllers in which their consent notices do not offer a way to refuse consent. Notably, all of them are from the confirmation-only group (see the second column of Table 5.3). Further, 134 apps that still sent data after explicitly opting out of the data sharing from the consent user interface. 58 out of 134 apps are from the complex choices group (see the third column of Table 5.3).

Interestingly, a significant number of potential violations are related to Android's Advertising ID (AAID), i.e., nearly 99% of apps at least sharing this personal data (Table 5.4 shows the type of personal data that we detected). According to Google, an AAID is *"a unique, user-resettable ID for advertising, provided by Google Play services. ... It enables users to reset their identifier or opt-out of personalized ads"* [63]. Furthermore, even Google's brand Admob explicitly lists the AAID as personal data in their documentation for delivering ads [71]. While Google itself remained vague on the characterization of the AAID as personal data, the IAB Europe GDPR Implementation Working Group already established in their 2017 Working Paper on personal data that *"Cookies and other device and online identifiers (IP addresses, IDFA, AAID, etc.) are explicitly called out as examples of personal data under the GDPR"* [73]. In May 2020 NOYB – European Center for Digital Rights [116], a European not-for-profit privacy advocacy group, lodged a formal complaint about the AAID with Austria's data protection authority. The complaint states that although the AAID is personal data Google does not adhere to the requirements of valid consent. Android users have no option to deactivate or delete the tracking ID, only to reset it to a new one.

More recently, Google has taken a first action regarding this matter. As part of the

Google Play services update³, the advertising ID will be removed when users opt-out of personalization using the advertising ID in Android Settings (i.e., any attempts to access the identifier will receive a string of zeros instead of the identifier) [64]. However, Article 29 Working Party states that consent can not be based on an opt-out mechanism (e.g., users have to withdraw their by default opted-in consent by turning off personalized ads through their device settings), as the failure to opt-out is not a clear affirmative action [126]. This indicates that none of the controllers who claim that data subjects can withdraw their by default opted-in consent by turning off personalized ads through their device settings, have a valid consent to process personal data [54]. In contrast, Apple has recently taken active action for mandatory prior consent for sharing of Advertising Identifiers for its iOS 14 update [7] explaining that even dynamic advertising identifiers are considered personal data.

Not Easy to Withdraw Consent GDPR Article 7(3) specifies that the controllers must ensure the data subject can withdraw their consent as easily as giving consent and at any given time. Notably, Article 29 Working Party [126] states that data subjects must be able to withdraw consent via the same interface, as switching to another interface for the sole reason of withdrawing the consent would require undue effort. Furthermore, the controller must make the withdrawal of consent possible free of charge or without lowering service levels [125]. To investigate whether developers provide the easy options to withdraw consent, we randomly sampled 100 apps that potentially violated at least one of the consent requirements and checked the app functionality. Specifically, we checked for options to allow withdrawal consent in the app settings. Among these 100 apps, we found only 16 apps present any options to withdraw consent notices. Overall, this indicates that most apps may not provide meaningful ways to withdraw consent, which is also necessary for valid consent under GDPR. However, further studies need to be conducted to confirm this problem with larger samples. Finally, we note that finding privacy-related app settings is challenging due to the difficulty in locating them from an app's user interface. The more challenging is to automatically detect those related to consent. We leave this challenge for future work.

5.4 Developer Notification

To enable developers to address the incorrect consent implementations, we notified affected developers, focusing mainly on the potentially violated apps that have implemented consent notices. On the one hand, this enables them to address the issues before other parties might take any legal actions (e.g., being fined for breaching data protection law [35, 147]). Second, we wanted to gain insights into the underlying reasons that caused the observed phenomena in the first place. In addition, we informed all notified developers about the study purpose, our methodology, and contact information (i.e., email address, phone number) to contact in case they had questions or concerns. We note that our institution's ethics guidelines do not mandate approval for such a study.

³The Google Play services' updates will affect Android 12 starting in late 2021 and then will expand to affect apps running on all devices starting April 1, 2022

Based on the developers' detailed contact information in the Play Store, we extracted the publicly available email addresses to send the notifications. Similar to the work of [P2], to access how many developers received our reports, rather than including the technical details in the email, we further sent developers a link to our Web interface. Specifically, in our Web report, we briefly explained our testing methodology, showed the developers information about potential violations, accompanied by the corresponding legal references (i.e., GDPR consent requirements), and detailed which hosts received which type of data. Further, to gain some insights into the underlying reasons that caused the identified problems, we asked participants if they had been aware of the potential violations of their apps, their general understanding of the personal data that their app sent to third-party services, and their understanding of GDPR consent requirements as well as proposals for tool support (see Appendix A.4 for the full email). We decided to have this rather than a full-fledged study, as we wanted to keep the overhead for respondents as low as possible to prompt more responses.

5.4.1 Notification and Accessed Reports

The potentially violated apps may have been updated (i.e., changed the problematic code, removed from the Play store) between our download and notification date. Thus, we further checked their availability and last update time before sending our notifications. Doing so, out of the 2,688 apps that implemented consent notices and potentially violated at least one of the GDPR consent requirements, we find 829 apps had been removed or updated with a newer version by the time we conducted our notification on April 04, 2022. We took this step to ensure that we would not notify developers who had removed the problematic code between our dataset download and notification date. Also, a single developer may have responsibility for more than one app in the store. Therefore, to ensure we do not send multiple emails to developers, we grouped emails to developers to receive only one email with multiple report links. We followed currently best practices established by existing work [39, 146, P2] allowing developers to opt-out of our study.

In total, we notified 1,127 developers responsible for the remaining 1,859 potentially violated apps. Of those developers, only one asked to be removed from our experiment and do not wish to be involved in further study. Until April 26, 2022, we saw 505 accessed reports for 225 apps. Notably, considering that a single developer may have multiple apps affected by the same issue, we count the overall number of apps for which their developer accessed *some* report; totaling 266 (14.31% of 1,859) apps for which we reached their developer.

5.4.2 Developer Responses

In addition to the accessed reports and the updated apps, we also analyzed the responses we received from developers to understand the underlying reasons that caused the problems. In total, this amounted to 43 distinct developers for which we classified emails. Note that not all respondents answered the stated questions from our email notification⁴. The response rate for our questionnaires was low, as might be expected.

⁴All notified developers were informed that their responses are pseudonymous and could be used in our paper.

However, it is in line with prior work [P2] which received 448 responses from 11,914 notifications.

Of the 43 respondents, 25 acknowledged receipt of our email and wanted to take it under advisement. Five stated that they required further investigation within their respective companies. Two have mentioned they updated the apps and further inquired with us to recheck their apps. Notably, two respondents argued that the EU was not their primary market and inquired us about potential solutions to the problems. We faithfully answered all of the emails while stating that we cannot provide conclusive individual legal assessments. On the other side, three respondents disagreed with our assessment.

When asked about the data collection, 9/14 respondents said they were not aware of the types of data being collected, and 5/14 said they knew GDPR protected this data. Of the 13 respondents who answered our question regarding being aware of GDPR consent requirements, 9 said yes and passed all the flags to the SDK, and four are not aware. Three explained their app was outdated, and four said this was a bug.

Regarding our final question about developer support, we received seven answers. Of those, six wanted to have an automated tool like our to analyze their apps for compliance, while two asked for better documentation around how to implement GDPR compliance. Finally, three respondents argued that third-party tools should be compliant by default, e.g., “a *one-stop solution*” as *Unity3D plugin that ensures to fully cover all current (and future) GDPR requirements*”.

5.4.3 Updates to Notified Apps

To assess our notification’s impact on the affected apps, we downloaded new versions of all apps that had looked at our reports at least one by April 26, 2022. Then, we re-ran our pipeline for each app with an updated version to assess if the changes were related to the reported GDPR infringement. For the 266 apps for which we reached a developer, 8 apps were removed from Google Play, 111 apps were no longer available to download from EEA country, and 147 apps have been updated. Of those 147 apps, 84 still potentially violated at least one of the GDPR consent requirements, leaving the remaining 63 apps which fixed the problems.

We note that the overall number of apps that addressed the issue is low, as might be expected from sending unsolicited emails to prospective participants. In fact, app developers are more likely to take action when they receive such notifications from their service providers [127], e.g., “*THE only person who can claim anything is my service provider-Play Store*”. Besides, we believe that the seemingly minor change in overall numbers can be attributed to a lack of time to address the issue properly.

5.5 Summary

GDPR requires the consent for processing users’ personal data to be *freely given, specific, informed, and unambiguous*. While we have shown that this often is not given through automated network traffic analysis ([P2]), no research has systematically studied how

consent notices are currently implemented and whether they conform to GDPR in mobile apps.

To close this research gap, in this chapter, we performed a systematic study into consent notices of third-party tracking in 239,381 Android apps in the wild to understand the current practices and the current state of GDPR's consent violations. Specifically, we proposed a mostly automated and scalable approach using image processing and natural language processing techniques to identify the implemented consent notices and their current practices. As a result, we recognized four widely implemented mechanisms to interact with the consent user interfaces from 13,082 apps, such as confirmation-only notices that feature a button with the text "OK" or "I agree". Based on the identified mechanisms, we then extended our previous work to develop a tool that automatically detects users' personal data sent out to the Internet with different consent conditions (i.e., based on the choice mechanism to interact with the consent notice).

We found 30,160 apps do not even attempt to implement consent notices for sharing users' personal data with third-party data controllers, which mandate explicit consent under GDPR. In contrast, out of 13,082 apps implemented consent notices, we identified 2,688 (20.54%) apps potentially violate at least one of the GDPR consent requirements, such as trying to deceive users into accepting all data sharing or even continuously transmitting data when users have explicitly opted out. We sent notification emails to inform affected developers and gather insights from their responses. Our study showed the urgent need for more transparent processing of personal data and supporting developers in this endeavor to comply with legislation, ensuring users can make free and informed choices regarding their data.

6

Conclusion

In this chapter, we first discuss our findings regarding data access transparency and privacy violations in Android apps. Subsequently, we derive concrete recommendations to all concerned parties and make an urgent call for more transparent processing of users' personal data. We then discuss our limitations, and future work that could be conducted to improve our current approaches and results before concluding this dissertation.

6.1 Discussion

Our results thus far have shown that many apps do not respect users' privacy, i.e., accessing user data in unexpected ways, and neither attempt to implement consent notices for sharing users' personal data with third-party data controllers for advertising purposes, despite the GDPR requirements. Given these insights, we now discuss further the problems in this section.

6.1.1 Transparency of Sensitive Resources Access

GUIBAT revealed that 36,115 apps (75.38% of 47,909) have at least one unexpected sensitive resource (*unexpected SR*). This is a non-negligible number given its implications to user's privacy. Further, this shows that even seemingly benign apps (*i.e.*, apps from Google Play) can have misleading UIs that lead to *unexpected SR*. Users already expressed concerns about the permissions they grant to apps, and the most common permissions that users worried about and were uncomfortable after they granted apps access to were STORAGE, PHONE, and LOCATION [23, 112]. This is also the same set of sensitive resources that were most frequently associated with *unexpected SR*. Besides, Micinski et al. showed that user actions such as pressing a button could be interpreted as authorization [107]. Thus, it is important to inform users whether such actions would lead to *unexpected SR*.

Further, in the third study (see Section 3.3), we have observed a significant correlation ($z = 19.7$, $p < 0.001$) between the users' expectation and comfort ratings. In other words, users' expectations of apps' *actual SR* are directly connected to their subjective feelings. Users are uncomfortable when an app's *actual SR* is not expected on the app's UI (inline with a prior study [98]). Tools like GUIBAT could support users in making decision to protect their privacy by learning about app's *unexpected SR*.

GUIBAT can also help developers to self-assess the informativeness of their apps' UIs. Future work can look into building new tools to support developers in this endeavor (*e.g.*, by suggesting changes). Besides, developers may also use privacy policies to inform users about their access to sensitive user data. However, privacy policies are often long and complicated that is difficult for users to understand [120]. Further, in practice, the majority (71%) of apps lack privacy policy even though they are obligated to have one [177]. Hence, they would have a limited impact if only a few users would actually read and understand the privacy policy.

Runtime permission The goal of using runtime permission is to improve the permission decision-making and avoid undermining users' expectations [46]. It enables apps to embed their permission requests in contexts, so that the end-users can understand

better. Our analysis found 21,843 apps with runtime permission (60.48% of 36,115 apps) that have at least one *unexpected SR*. We found a significant difference (Kruskal-Wallis, $\chi_2 = 197.99$, $df = 1$, $p < .001$) between the number of *unexpected SR* of apps with install-time permission (mean = 10.99) and of apps with runtime permission (mean = 10.05). This suggests that apps with runtime permissions have significantly smaller numbers of *unexpected SR*. However, we believe that the runtime permission is not the panacea for resolving the transparency issues of accessing sensitive data. Particularly, in this analysis, the effect size of runtime permissions is negligible (*epsilon - squared* = 0.00413). An app behavior may defy users' expectations, depending on whether the app provides users enough semantics to justify the access — not merely whether the app was authorized to receive data the first time it asked for it [122].

The Impact of Third-party Libraries GUIBAT revealed that 38.20% of apps have *unexpected SR* that are exclusively attributed by third-party libraries. This suggests that developers should be extra careful about the *actual SR* by libraries. More specifically, developers should be informed when they use an API that accesses sensitive resources. Library developers should also provide information on what the library does, which sensitive resources will be accessed under which conditions. Future work can further analyze libraries to provide permission-protected API mappings. This could put app developers in a better position in informing their users about the apps' permission access. Central repositories such as Jcenter and Maven also play an important role in making permission accesses of libraries more transparent. For example, these central repositories can enforce third-party libraries to follow a permission-transparency policy, *e.g.*, having an explanation for usages of permission-protected API(s).

6.1.2 Transparency of Processing Users' Personal Data

We find that the majority of apps have not asked for informed consent from the users for tracking and profiling third-party services (i.e., 30,160 apps do not even attempt to implement consent notices). Thus, it is practically impossible for users to know which third parties receive and process their data. Moreover, even if users had the time and knowledge to read and understand privacy policies, these documents are excessively complicated and obtuse, and the majority (71%) of apps lack privacy policies even though they are obligated to have one [178, 54]. Therefore, we show the urgent need for more transparent processing of users' personal data and further allowing them to exercise their fundamental rights and freedoms. In mobile apps, users can take only a few actions to limit or prevent tracking and data sharing, while there are various tools to prevent tracking in the Web browsers [103, 20].

6.1.3 Widespread Violation of GDPR Consent

On the legal side, EU regulators have already been active. Recently, the Norwegian Data Protection Authority (DPA) imposed a fine of \$7.17M on Grindr [38] for obtaining invalid consent under GDPR (i.e., users had to agree to the entire privacy policy but not to a specific processing operation, and do not have the choice not to consent). The France DPA (CNIL) fined Google \$170M and Facebook \$68M for breaching French and

GDPR laws (i.e., these tech giants were using manipulative dark patterns to try to force consent) [147]. In late October 2018, the CNIL also made a ruling decision on an advertising company, which suggests that bundling consent to partner processing in a contract is not valid consent under the GDPR [148]. The CNIL stated that controllers have to implement a compliant consent mechanism (i.e., it must be *freely given, specific, informed, and unambiguous*) and ensure that any personal information is collected and processed lawfully. This means that when receiving personal data from a partner company, the receiving party must demonstrate that the transmitting party also relied on legally compliant consent mechanisms [54]. However, our results show a significant skew toward apps sending out personal data to advertisement companies without valid consent under GDPR, i.e., 30,160 apps do not even attempt to implement consent notices, 2,688 apps that implemented a form of consent notice but potentially violate at least one of the GDPR consent requirements. Those apps did not present the user with legally compliant consent mechanisms under GDPR, which has consequences for the validity of consent for any third parties acting as controllers, e.g., those advertising companies that received personal data.

In practice, many third-party services claim that they operate based on consent passed on through contractual terms with their customers, which would be the app developers in this case (e.g., Facebook required developers to obtain appropriate legal basis consent before sending data via their SDK [44], while it by default automatically collects data). However, those third-party data controllers can neither demonstrate valid legal consent nor a legitimate interest that overrides the consumer’s fundamental right to privacy for behavioral profiling and targeted advertising [54].

6.1.4 Lack of Support for Developers

Is it known that developers are currently in a disadvantaged position, where third parties make it cumbersome for developers to comply with GDPR [P2]. However, as first-party data controllers, developers are legally responsible for ensuring that the users’ data is lawfully processed. Based on the received responses, there is a clear need for better information and documentation from third-party services and assurance tools that help developers comply with strict law standards, e.g., *“I would love there to be a standardised implementation requirement for this. It seems every 3rd party SDK we have uses different ways of implementing consent. The documentation can be quite unclear, so having a tool to analyse app traffic would be incredibly useful.”*. Therefore, we strongly call on providing developers with clear requirements or guidance for how GDPR consent has to be legally obtained.

6.2 Calls to Action

Our results thus far have shown that the sharing of personal data with third-party data controllers is very pronounced in the datasets we tested. More than one-third of all apps we tested sent out personal data before any users’ interaction. More notably, we could not find a significant difference between high-profile and long-tail apps, i.e., the problem affects both high-profile and long-tail apps. Given these insights, we now discuss which

involved parties can take which steps to remedy the situation.

6.2.1 Authorities Should Actively Enforce The Law

Generally, the enforcement of GDPR obligations falls under the jurisdiction of national data protection authorities (DPAs) appointed by each EU member state. DPAs are independent public authorities that investigate and handle complaints that may have breached the law within their jurisdiction [31]. In some cases, DPAs in different EU countries need to collaborate to determine who has the ultimate authority to conduct an investigation, e.g., companies within the EU whose business affects many EU data subjects [34]. At the EU level, the European Data Protection Board (EDPB) ensures that GDPR and the Data Protection Law Enforcement Directive are consistently applied across the EU, and provides general guidance to clarify the GDPR, e.g., guidelines, recommendations, and best practices [40]. However, based on the results of our studies, enforcement seems to be lacking since there is evidence of potential violations of data protection legislation. Therefore, there is an urgent need for authorities to actively monitor and investigate any potential violations of data protection legislation in order to ensure that consumers and citizens are protected online. As such, our tools could benefit authorities in investigating potential violations and bringing enforcement action.

6.2.2 Third Parties Should Take Responsibility

Today, digital content is largely funded by advertising, which means that companies monetize our behavior, attention, and personal data rather than us paying for services with money [54]. To maximize revenue, advertising services heavily rely on continuous data collection and tracking personal data from users [103]. Our results show a significant skew towards apps sending out personal data to advertisement companies without user's explicit prior consent (i.e., 86.6% of all apps that sent personal data to the Internet) — which is the most prominent business case of third parties receiving and processing user data for their own business purposes. However, we found that these third parties make it cumbersome for developers to comply with GDPR or shift the responsibility to app developers.

For example, Facebook required developers to obtain users' consent before sending data via the SDK [44], whereas the default behavior is automatically collecting user personal data such as AAID [45]. Our insights further show that such popular companies play key roles in the widespread receiving of personal data without users' explicit consent, i.e., more than half of apps that sent data without consent sent it to (at least) Facebook. However, many developers believed that their apps are compliant by default when using these popular companies' services, as noted by one respondent as *“These third party SDKs are from industry leading ad networks that only accept those apps that are GDPR compliant. So a GDPR compliance is must before the app is being approved by these advertising networks (i.e. Facebook & Admob). So our app is a GDPR complaint”* (sic). In addition, some respondents claimed to be aware of GDPR-relevant data, but were surprised by our reports which showed that the SDKs collected information; *“We were already aware of this topic and we were already working on it. We did not send any events to Facebook (we eliminated this feature long ago) – the SDK itself sent out data*

to Facebook without any trigger from our side”. While this lack of knowledge does not absolve the first party of their responsibility, the lack of clear guidelines and safe defaults for GDPR-compliant data collection by the advertisement industry inevitably puts their customers, i.e., the app developers, at risk of the draconian fines which can be imposed for GDPR violations [54].

Our findings show the urgent need for advertisement companies and third parties (data controllers) to make comprehensive changes to help app developers comply with European regulations and users exercise the data subject’s fundamental rights and freedoms. Notably, third parties should limit the data collection to respect privacy by design principles. For obtaining user consent, third parties can provide the consent mechanism that automatically shows the consent dialogue to users and explicitly ask for opt-in to data sharing and collection, without forcing the developers to implement this mechanism in a legally compliant way. However, in practice, third parties usually have different versions of consent notices with different legal implications. As such, it may negatively affect usability with multiple consent notices for different purposes. Further, it is difficult to determine who is responsible for violating GDPR consent requirements, either the third-party services or the app developers, since the developers have a part of the responsibility if they choose non-compliant services [103]. The global consent mechanism (i.e., the user consent is shared between multiple services) is the other approach that is still under discussion by legal scholars, which has shown users’ lack of knowledge that consent is global and that opting out is near impossible [103, 130]. Further, to support developers, third parties should make their documentation transparent and easy to access, including explicit discussions of implications of violating GDPR.

6.2.3 App Stores Should Take Actions

App stores, e.g., Google Play, are primary channels for developers to distribute their apps to the end-users. Given that the app stores are not legally responsible for developers’ privacy abuses and are not responsible for enforcing regulations such as GDPR, CCPA, or COPPA rules. However, we believe that app stores play an important role in supporting developers to be informed about each territory’s related regulations and protect user privacy. In this work, we found that developers lack such support, e.g., *“This game was designed to Brazil and we published the game to Europe in March 2019 to expand our potencial [sic] customers, targeting Portugal. GooglePlay allows this, checking a button, without any restrictions. So, I feel protected by Google somehow”*. Therefore, we strongly suggest that app stores should take more decisive actions in this area. For example, when developers upload their apps, the store should tell them about the selected countries’ associate regulations.

Besides a large number of apps that sent personal data to ad-related domains without users’ explicit consent (24,838 apps), we further detected a total of 3,840 apps that combined the AAID with some other type of personal data. Hence, all these apps not only infringe on the explicit consent required by GDPR, but also violate Google’s policy [62, 2]. Such behaviors happened due to developers’ opt-in or the usage of outdated libraries that do not support GDPR. Given that, app stores could also employ

such techniques as our to identify the potential violations of GDPR explicit consent, or the usage of outdated SDK by or LibScout [13], and then inform developers before delivering the apps to end-users. To support this effort, we make our analysis pipeline available as open-source [61].

6.2.4 Support for Developers

Obviously developers play a major role in making their apps compliant with the GDPR. Our findings show that they are currently put in a disadvantaged position. Out of the responses we received, more than half noted that they were unaware of what counts as *personal* data under GDPR. From the received responses, there is a clear need for better information and documentation as well as tools which help developers avoid such pitfalls. Further, based on our in-depth analysis of third party’s developer and legal documentation, we observed that third parties make it cumbersome for developers to comply with GDPR. We therefore strongly call on third-party vendors for better documentation and transparency in legal documents, which should in turn be thoroughly checked by developers when building their apps.

6.3 Limitations and Future Work

In this section, we discuss the limitations that impact our results presented throughout this dissertation. We further discuss how future work could be done to improve our current approaches and results.

6.3.1 Detecting Unexpected Data Access

Similar to any other static analysis approaches, GUIBAT is over-approximation in identifying associated callbacks of UI elements. Besides, while LibScout is resilient against common obfuscation techniques (e.g., identifiers renaming, API hiding), it would fail when apps leverage more advanced obfuscation techniques (e.g., package flattening, class repackaging). Besides LibScout, we additionally used package names (directories) to identify third-party libraries by comparing their package names. This heuristic allows us to cover cases of unknown third-party code that LibScout could not identify. However, relying on package names has low coverage of third-party libraries since package names can be modified in many ways, e.g., package name obfuscation. Hence, this would affect our results in attributing the source of unexpected sensitive resources (*unexpected SR*) in apps. Future work could adopt more advanced de-obfuscation tools [18, 170] to pre-process these obfuscated apps. Note that Wemker et al. showed that only 24.92% apps on Google Play are obfuscated by developers [164]. Specifically related to UI, apps could also obfuscate their UI, e.g., by using UIObfuscator [176]. This would affect our results in statically extracting UI elements. Future work could look into developing de-obfuscators that address the challenges that UIObfuscator introduces, for example, by removing invisible layout from view hierarchies, resolving Java reflection [16], and dynamically instrumenting and analyzing apps to unveil apps’ UI. Especially, with the techniques used in GUIBAT, we could build an icon classifier that learnt from the unobfuscated apps (majority of apps on Google Play [164]), and combine it with

dynamic analysis (and instrumentation techniques) to analyze the UI of obfuscated apps.

Further, we did not consider UI elements on webviews or other dynamic app content, which would potentially affect the number of the identified UI elements. These limitations might affect the precision of GUIBAT in detecting the *unexpected SR*. Besides, we did not consider SENSORS permission as only 0.09% (319 of 600,000) of apps in our dataset request this permission. In our analysis, we did not consider inter-app interaction (e.g., an app can launch another app). For such cases, users have to choose the provider apps (receiving intent) explicitly. Thus, further analysis of such providers are needed. This would affect the number of false positive in our analysis. We leave these challenges for future work.

To examine the coverage of the icons in our studies, we extended our *Icon Classifier* with the results of the first studies (see Section 3.1) to identify *all* commonly-known icons in our dataset (700,000 icons). With the confidence probability higher than 0.5 (random guessing), the *Extended Icon Classifier* successfully identified 72.35% of the icons across 76 groups of commonly-known icons. Applying the *Extended Icon Classifier* on the Material Design Icons [50], we successfully identified 83.22% of the icons (352 of 423). We focused on the commonly-known icons (72.35%), and ignored icons that users have not seen before since it is not possible to formulate user’s mental models of such icons. This limits us from studying the user’s perception of *not yet seen* icons. Thereby, our detection result can only serve as a lower-bound of *unexpected SR* in mobile apps. Further, our studies were conducted online which might suffer from opt-in bias and the inherent bias from MTurk’s users.

In Chapter 3, we referred to prior works [112, 161] and Android’s documentation to create the mapping for sensitive related keywords (see Table 3.3). This mapping might be incomplete, and hence, could limit GUIBAT in identifying *expected SR* of textual UI elements. One alternative approach would be to build the mapping by mining the app dataset and then picking the most common keyword associated with *actual SR*. However, even with a complete mapping, the text-based detector is deficient for identifying *expected SR* of UI elements that comprise solely of icons (14.06% of apps in our dataset).

Further, to detect the actual sensitive resource accesses of Android apps (Section 3.2.3), we used PScout and AExplorer to find out which API is permission-protected API (i.e., accessing users’ sensitive data) [10, 14]. However, the mappings of PScout and AExplorer only support up to Android version 7.1, which may not cover potential changes in the newer version of the Android permission model. To see how these mappings affect our proposed approach, we randomly crawled all the app metadata (i.e., including the required Android version to run the app) on Google Play in August 2023, resulting in 1,452,382 Android apps. We found that 96.70% of apps still support Android 7.1 and the older version of the Android permission model. Given that, we believe GUIBAT still effectively detects unexpected sensitive resource access in the wild.

Finally, an interesting direction for future work is to identify the root causes of the *unexpected SR* in mobile apps, e.g., by conducting an in-depth study with developers and end users to characterize the identified *unexpected SR*.

6.3.2 Detecting Violation of GDPR's Explicit Consent

Our approach naturally suffers from certain limitations, some of which are desired. As an example, an app might show a welcome screen unrelated to data collection consent and only send out data once the user interacts with the app. Our framework would miss such cases of incorrect consent handling. We consciously decided to allow these false negatives, as understanding whether or not the welcome screen asks explicit consent and opt-out is infeasible to be done automatically.

Second, it might be possible that the consent notices are part of the runtime permissions request (e.g., apps have rationales that indicate data collection and use). By automatically granting all apps' permission requests, our approach might have false positives for such cases. However, in practice, Liu et al. [99] show that most developers do not provide rationales for permission requests (less than 25% of apps in their study). Moreover, before Android 6, only install-time permissions existed, meaning that any app compatible with Android 5 or lower *could not* ask for consent in the permission request. Out of the apps that we detected to send personal data (see Section 4.3), about 96% support Android prior to 6.

Third, given that we rely on software that attempts to bypass security mechanisms (in particular SSL pinning), the apps may be able to detect such manipulation, e.g., by checking which CA is the root of the trust chain. Similarly, an app may also simply not start on a rooted device. Moreover, apps may not be supported on the Android 8 devices, which means they might not start and hence cannot be analyzed. Generally speaking, all these are potential causes for false negatives.

Next, similar to other heuristic approaches for identifying third-party hostnames, our approach also has potential shortcomings. In particular, app developers can use CDNs, and cloud services, e.g., Cloudfront, AppEngine, or Azure, which may support first parties using pseudo-random domains. For example, `xyz.cloudfront.com` could be a first-party domain, which could be incorrectly identified as a third party. To avoid such cases, we assume a third party would serve multiple apps and only flag those domains as third-party domains when at least ten different apps contact them. Further, in Section 4.3.3, by manually analyzing the legal documents of those identified third-party domains, we only select the ones that are actually third-party advertising data controllers. As such, our results do not suffer from false positives regarding identifying third-party advertising data controllers.

Finally, an app may also transmit a persistent identifier in some encrypted form with changing encryption keys or use a custom serialization format. Naturally, this is not something we can account for, and we would miss the parameter (as we could not decode the serialization protocol or, in case of the encryption case, its values already change between R_1 and R_2). However, we argue that if any app is detected to send out personal data in our automated system, we have never granted explicit consent; hence we do not suffer from false positives.

6.3.3 Studying Consent Notices of Third-Party Tracking

We naturally suffer from certain limitations from OCR and dynamic analysis, in which the collected privacy-related user interfaces and the identified consent notices may be

incomplete by our approach. First, the text content may be missed when extracting from app screenshot to text by using OCR, or the taken screenshot could be only a small part of the consent notices.

However, relying on static analysis techniques, which are well known for producing unsound results, is not an option [158, 93, 22]. On top of that, with the static analysis, we could not see the actual appearance of such privacy-related user interfaces, which is necessary for our analysis to identify the consent notices and their current practices. To demonstrate the insufficiency of static analysis, we first randomly sampled 1,000 apps from those 20,542 apps that have privacy-related user interfaces (see Section 5.3.2). We then performed a static analysis to extract the text from the *strings.xml* resource of these apps (where the apps store UI text), and then apply the same keyword matching to the extracted text (i.e., identifying privacy-related user interfaces). As a result, the static analysis approach only identified 345 (34.5% of 1,000) apps that have privacy-related text. We found that the static analysis has missed the majority (65.5%) of privacy-related user interfaces.

Therefore, we argue that the static analysis is insufficient to identify privacy-related user interfaces (potentially containing consent notices). However, similar to any other static analysis, our approach also has drawbacks. In particular, we statically analyzed the *strings.xml* resource to identify the privacy-related text (which could potentially be used by the consent notice user interfaces). In practice, the app may dynamically load consent notices from the Web API, consent CMP(s), third-party SDK(s), or dynamic content. Further, developers may add hardcoded text into the UI layout files or the app's code, which could be obfuscated (i.e., require more advanced de-obfuscation tools to analyze [170, 18]). As such, our static analysis could miss such cases, which can only be analyzed by dynamic analysis (i.e., actually running the app).

Further, to estimate the false negatives of our proposed approach, we randomly sampled 100 non-flagged apps (identified as not having any form of consent notices by our approach) and manually checked each app. Among them, we found that four apps crashed, one app had a consent notice, and 95 apps had no consent notices.

Finally, we only consider consent notices written in English, supported by the natural language processing techniques that we used, to ensure that we understand the actions we perform, such as measuring the quality of privacy-related user interface clustering and categorizing the consent notices in Section 5.2.2. As such, our results have not covered 1,996 (9.72% of 20,542) privacy-related user interfaces that are not in English but instead, e.g., German, Spanish. There are some limitations future research could take into consideration.

6.3.4 Future Research Directions

We again note some notable future research directions for this dissertation. Regulatory efforts around the globe, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), have been made to protect users' fundamental rights and freedoms. With respect to the GDPR legislation, we first attempted to explore the transparency of users' data access and further understand whether they conform to GDPR consent requirements. To provide a comprehensive

overview of the current state of privacy violations in Android apps, further research could also look at other legal justifications for processing personal data. For example, under GDPR, the processing also may be based on the fulfillment of a contract, compliance with a legal obligation, or the data controller’s legitimate interests. Additionally, future research could be conducted to examine other data protection rights under GDPR or CCPA, such as the right of access by data subjects, the right to rectification and erasure their data.

With the complexity of tracking and profiling consumers through many online services, especially in the advertising industry, users are impossible to make informed choices regarding how their personal data is collected, shared, and used. However, developers are also currently in a disadvantaged position. Inspired by our findings, future research could be extended to develop services and assurance tools that help developers comply with strict law standards.

6.4 Conclusion

Increasing data collection and tracking of consumers by today’s online services is becoming a major problem for individuals’ rights regarding their personal data, e.g., users are secretly tracked and profiled. As such, users cannot make informed choices regarding how their personal data is collected, shared, and used. In recent years, regulatory efforts have been made to mandate online services to disclose *transparently* how they handle personal data and grant users crucial data protection rights. However, the community lacks insight into whether such data processing is legally compliant, and what are the possible reasons behind such legislation violations in the mobile ecosystem.

In this dissertation, we discussed our research on understanding and measuring privacy violations in Android apps, focusing on the GDPR legislation. In Chapter 2, we discussed the relevant technical background, and presented the legal background of GDPR and GDPR consent requirements. Subsequently, we briefly discussed prior work in detecting unexpected data access in Android apps and prior research on studying the legislation violations of online services.

We then introduced our approach to measuring how end-users perceive app behaviors based on graphical UI elements and built GUIBAT that detects sensitive resource accesses that violate user expectations in Chapter 3. Specifically, we performed a series of user studies to build a semantic mapping between user expectations of sensitive resource accesses and common apps’ icon UI elements (N=459). Based upon the knowledge gained from two user studies and static control-flow analysis, we then built GUIBAT that accounts for users’ perception to detect unexpected data accesses. Our evaluations showed that GUIBAT significantly outperforms prior works in identifying user expectations of sensitive resource accesses when interacting with the app’s UI (i.e., can accurately reflect users’ expectations of sensitive resource accesses in apps), and its efficiency enables the large-scale study. We applied GUIBAT on 100,000 Android apps to look at the landscape of unexpected access to sensitive resources in the wild. GUIBAT identify 47,909 apps with UI elements accessing sensitive resources, and 75.38% of the apps have at least one unexpected sensitive resource access. Among these apps, 38.20% have unexpected data accesses exclusively attributed by third-party libraries.

In Chapter 4, we then performed the large-scale measurement on Android apps in the wild to understand the current state of the violation of GDPR’s explicit consent. Specifically, we built a semi-automated pipeline to detect data sent out to the Internet without prior consent and applied it to a set of 86,163 Android apps. Based on the domains that receive data protected under the GDPR without prior consent, we collaborated with a legal scholar to assess if these contacted domains are third-party data controllers. Doing so, we found 24,838 apps send personal data to data controllers without the user’s explicit prior consent. To understand the reasons behind this, we ran a notification campaign to inform affected developers and gather insights from their responses. We then conducted an in-depth analysis of violating apps as well as the corresponding third parties’ documentation and privacy policies.

Next, we performed a large-scale study into consent notices for third-party tracking in Android apps to understand the current practices and the current state of GDPR’s consent violations in Chapter 5. Specifically, we proposed a mostly automated and scalable approach to identify the currently implemented consent notices and apply it to a set of 239,381 Android apps. As a result, we recognized four widely implemented mechanisms to interact with the consent user interfaces from 13,082 apps. We then developed a tool that automatically detects users’ personal data sent out to the Internet with different consent conditions based on the identified mechanisms. Doing so, we found 30,160 apps do not even attempt to implement consent notices for sharing users’ personal data with third-party data controllers, which mandate explicit consent under GDPR. In contrast, out of 13,082 apps implemented consent notices, we identified 2,688 (20.54%) apps violate at least one of the GDPR consent requirements, such as trying to deceive users into accepting all data sharing or even continuously transmitting data when users have explicitly opted out. To allow developers to address the problems, we further sent emails to notify affected developers and gather insights from their responses.

Given these insights, we discussed further the problems in Chapter 6. We believe our results will shed new light on the transparency of sensitive resource access in mobile apps. GUIBAT would help to inform end-users about unexpected access to sensitive resources and help app stores to better control the compliance of apps to the transparency policies. Further, our study shows the urgent need for more transparent processing of personal data and supporting developers in this endeavor to comply with legislation, ensuring users can make free and informed choices regarding their data. Based on the responses and our analysis of available documentation, we derive concrete recommendations for all involved entities in the ecosystem to allow data subjects to exercise their fundamental rights and freedoms.

Bibliography

Author's Papers for this Thesis

- [P1] Nguyen, T. T., Nguyen, D. C., Schilling, M., Wang, G., and Backes, M. Measuring User Perception for Detecting Unexpected Access to Sensitive Resource in Mobile Apps. In: *ACM Asia Conference on Computer and Communications Security*. 2021.
- [P2] Nguyen, T. T., Backes, M., Marnau, N., and Stock, B. Share First, Ask Later (or Never?) - Studying Violations of GDPR's Explicit Consent in Android Apps. In: *USENIX Security Symposium*. 2021.
- [P3] Nguyen, T. T., Backes, M., and Stock, B. Freely Given Consent? Studying Consent Notice of Third-Party Tracking and Its Violations of GDPR in Android Apps. In: *ACM Conference on Computer and Communications Security*. 2022.

Other Papers of the Author

- [S1] Elbitar, Y., Schilling, M., Nguyen, T. T., Backes, M., and Bugiel, S. Explanation Beats Context: The Effect of Timing & Rationales on Users' Runtime Permission Decisions. In: *USENIX Security Symposium*. 2021.
- [S2] Wi, S., Nguyen, T. T., Kim, J., Stock, B., and Son, S. DiffCSP: Finding Browser Bugs in Content Security Policy Enforcement through Differential Testing. In: *Network and Distributed System Security Symposium*. 2023.

Other references

- [1] (ISO), I. S. O. *International standard for safety colours and safety signs: ISO 3864*. 1984.
- [2] Agreement, G. D. D. *Developer Distribution Agreement*. <https://play.google.com/about/developer-distribution-agreement.html#use>. 2021/01/17. 2021.
- [3] Allix, K., Bissyandé, T. F., Klein, J., and Le Traon, Y. Androzoo: collecting millions of android apps for the research community. In: *MSR*. 2016.
- [4] Andow, B., Mahmud, S. Y., Whitaker, J., Enck, W., Reaves, B., Singh, K., and Egelman, S. Actions speak louder than words: entity-sensitive privacy policy and data flow analysis with polichack. In: *USENIX Security*. 2020.

BIBLIOGRAPHY

- [5] AppBrain. *Android apps on Google Play*. <https://www.appbrain.com/stats/number-of-android-apps>. 2022/10/02. 2022.
- [6] appbrain. *Google Play Ranking*. <https://www.appbrain.com/stats/google-play-rankings/>. 2021/05. 2021.
- [7] Apple. *User Privacy and Data Use*. <https://developer.apple.com/app-store/user-privacy-and-data-use/>. 2021/02/01. 2021.
- [8] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., and McDaniel, P. Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices* (2014).
- [9] Au, K. W. Y., Zhou, Y. F., Huang, Z., and Lie, D. Pscout: analyzing the android permission specification. In: *CCS*. 2012.
- [10] Au, K., Zhou, Y., Huang, Z., and Lie, D. Pscout: analyzing the android permission specification. In: *CCS*. 2012.
- [11] Avdiienko, V., Kuznetsov, K., Rommelfanger, I., Rau, A., Gorla, A., and Zeller, A. Detecting behavior anomalies in graphical user interfaces. In: *ICSE-C*. 2017.
- [12] Backes, M., Bugiel, S., and Derr, E. Reliable third-party library detection in android and its security applications. In: *CCS*. 2016.
- [13] Backes, M., Bugiel, S., and Derr, E. Reliable third-party library detection in android and its security applications. In: *CCS*. 2016.
- [14] Backes, M., Bugiel, S., Derr, E., McDaniel, P., Octeau, D., and Weisgerber, S. On demystifying the android application framework: re-visiting android permission specification analysis. In: *USENIX Security*. 2016.
- [15] Bao, L., Le, T. B., and Lo, D. Mining sandboxes: are we there yet? In: *SANER*. 2018.
- [16] Barros, P., Just, R., Millstein, S., Vines, P., Dietl, W., d’Amorim, M., and Ernst, M. Static analysis of implicit control flow: resolving java reflection and android intents (t). In: *ASE*. 2015.
- [17] Bhoraskar, R., Han, S., Jeon, J., Azim, T., Chen, S., Jung, J., Nath, S., Wang, R., and Wetherall, D. Brahmastra: driving apps to test the security of third-party components. In: *USENIX Security*. 2014.
- [18] Bichsel, B., Raychev, V., Tsankov, P., and Vechev, M. Statistical deobfuscation of android applications. In: *CCS*. 2016.
- [19] Board, E. D. P. *Guidelines 2/2019 on the processing of personal data under Article 6(1)(b) GDPR in the context of the provision of online services to data subjects*. https://edpb.europa.eu/sites/edpb/files/files/file1/edpb_guidelines-art_6-1-b-adopted_after_public_consultation_en.pdf. 2019/02. 2019.
- [20] Bollinger, D., Kubicek, K., Cotrini, C., and Basin, D. Automating cookie consent and gdpr violation detection. In: *USENIX Security*. 2022.

-
- [21] Bonett, R., Kafle, K., Moran, K., Nadkarni, A., and Poshyvanyk, D. Discovering flaws in security-focused static analysis tools for android using systematic mutation. In: *USENIX Security*. 2018.
- [22] Bonett, R., Kafle, K., Moran, K., Nadkarni, A., and Poshyvanyk, D. Discovering flaws in security-focused static analysis tools for android using systematic mutation. In: *USENIX Security*. 2018.
- [23] Bonné, B., Peddinti, S., Bilogrevic, I., and Taft, N. Exploring decision making with android’s runtime permission dialogs using in-context surveys. In: *SOUPS*. 2017.
- [24] Book, T., Pridgen, A., and Wallach, D. Longitudinal analysis of Android ad library permissions. In: *MOST*. 2013.
- [25] *Breyer v. Germany*. 2021.
- [26] CCPA. *California Consumer Privacy Act (CCPA)*. <https://oag.ca.gov/privacy/ccpa>. 2021/02/01. 2021.
- [27] Chawla, N. Data mining for imbalanced datasets: an overview. In: *Data Mining and Knowledge Discovery Handbook*. Ed. by Maimon, O. and Rokach, L. 2005.
- [28] Cheng, H. and Patterson, P. Iconic hyperlinks on e-commerce websites. *Applied Ergonomics* (2007).
- [29] Chollet, F. Building powerful image classification models using very little data. *Retrieved December 13* (2016).
- [30] Choudhary, S., Gorla, A., and Orso, A. Automated test input generation for android: are we there yet? In: *ASE*. 2015.
- [31] Commission, E. *What are Data Protection Authorities (DPAs) and how do I contact them?* https://commission.europa.eu/law/law-topic/data-protection/reform/rights-citizens/redress/what-are-data-protection-authorities-dpas-and-how-do-i-contact-them_en. 2023/12/01. 2023.
- [32] COPPA. *Children’s Online Privacy Protection Act*. <https://www.ftc.gov/legal-library/browse/rules/childrens-online-privacy-protection-rule-coppa>. 2023/07/01. 2023.
- [33] Cortesi, A., Hils, M., Kriechbaumer, T., and contributors. *mitmproxy: A free and open source interactive HTTPS proxy*. 2010.
- [34] Daigle, B. and Khan, M. The eu general data protection regulation: an analysis of enforcement trends by eu data protection authorities. *J. Int’l Com. & Econ.* (2020).
- [35] Datatilsynet. *Intention to issue EUR 10 million fine to Grindr LLC*. 2021/02/04. 2021.
- [36] Degeling, M., Utz, C., Lentzsch, C., Hosseini, H., Schaub, F., and Holz, T. We value your privacy... now take some cookies: measuring the gdpr’s impact on web privacy. In: *NDSS*. 2019.

BIBLIOGRAPHY

- [37] Derr, E., Bugiel, S., Fahl, S., Acar, Y., and Backes, M. Keep me updated: an empirical study of third-party library updatability on android. In: *CCS*. 2017.
- [38] Digital Rights, N. -. E. C. for. *NCC & noyb GDPR complaint: Grindr fined € 6.3 Mio over illegal data sharing*. <https://noyb.eu/en/ncc-noyb-gdpr-complaint-grindr-fined-eu-63-mio-over-illegal-data-sharing>. 2022/04/28. 2022.
- [39] Durumeric, Z., Wustrow, E., and Halderman, J. A. Zmap: fast internet-wide scanning and its security applications. In: *USENIX Security*. 2013.
- [40] EDPB. *The European Data Protection Board (EDPB)*. https://european-union.europa.eu/institutions-law-budget/institutions-and-bodies/search-all-eu-institutions-and-bodies/european-data-protection-board-edpb_en. 2023/12/01. 2023.
- [41] Ellis, P. D. *The essential guide to effect sizes: Statistical power, meta-analysis, and the interpretation of research results*. Cambridge university press, 2010.
- [42] europa.eu. “*Opinion 06/2014 on the notion of legitimate interests of the data controller under Article 7 of Directive 95/46/EC (Article 29 Working Party)*”. https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp217_en.pdf. 2020/09/02. 2020.
- [43] Exodus-Privacy. *Exodus*. <https://github.com/Exodus-Privacy/exodus-standalone>. 2020/09/14. 2020.
- [44] Facebook. *Best Practices for GDPR Compliance*. <https://developers.facebook.com/docs/app-events/gdpr-compliance/>. 2021/02/01. 2021.
- [45] Facebook. *Get Started - Android*. <https://developers.facebook.com/docs/app-events/getting-started-app-events-android#auto-events>. 2021/02/01. 2021.
- [46] Felt, A., Egelman, S., Finifter, M., Akhawe, D., Wagner, D., et al. How to ask for permission. *HotSec* (2012).
- [47] Felt, A., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. Android permissions: user attention, comprehension, and behavior. In: *SOUPS*. 2012.
- [48] Filyp. *autocorrect*. <https://github.com/filyp/autocorrect>. 2022/04/28. 2022.
- [49] Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., and Fahl, S. Stack overflow considered harmful? the impact of copy&paste on android application security. In: *SP*. 2017.
- [50] Flaticon. *Material*. <https://www.flaticon.com/packs/material-design>. 2020/04/30. 2020.
- [51] Flurry. *Android SDK Integration*. <https://developer.yahoo.com/flurry/docs/integrateflurry/android-manual/>. 2021/02/01. 2021.

-
- [52] Flurry. *Android SDK Release Notes*. <https://developer.yahoo.com/flurry/docs/releasenotes/android/#version-6-3-0-03-22-2016>. 2021/02/01. 2021.
- [53] Flurry. *Flurry Monetization and GDPR*. <https://developer.yahoo.com/flurry/docs/publisher/gdpr/>. 2021/02/01. 2021.
- [54] forbrukerradet.no. *OUT OF CONTROL*. <https://fil.forbrukerradet.no/wp-content/uploads/2020/01/2020-01-14-out-of-control-final-version.pdf>. 2020/09/02. 2020.
- [55] GATOR. *GATOR: Program Analysis Toolkit For Android*. <http://web.cse.ohio-state.edu/presto/software/gator/>. 2018.
- [56] Gatsou, C., Politis, A., and Zevgolis, D. From icons perception to mobile interaction. In: *FedCSIS*. 2011.
- [57] GDPR. *Opinion 03/2013 on purpose limitation (WP 203), adopted on 2 April 2013*. https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2013/wp203_en.pdf. 2022/04/21. 2013.
- [58] GDPR. *Art. 6 Lawfulness of processing*. 2021/02/01. 2021.
- [59] GDPR. *Art. 7 Conditions for consent*. 2021/02/01. 2021.
- [60] GDPR. *Art. 4 Definitions*.
- [61] *GDPR-Consent*. 2021.
- [62] Google. *Ads*. <https://support.google.com/googleplay/android-developer/answer/9857753>. 2021/01/17. 2021.
- [63] Google. *Play Console Help for Android Developers - Advertising ID*. <https://support.google.com/googleplay/android-developer/answer/6048248?hl=en>. 2021/02/02. 2021.
- [64] Google. *Advertising ID*. <https://support.google.com/googleplay/android-developer/answer/6048248?hl=en>. 2022/04/28. 2022.
- [65] Google. *Android Documentation*. <https://developer.android.com/>. 2022/10/02. 2022.
- [66] Google. *Android Manifest Permission*. <https://developer.android.com/reference/android/Manifest.permission>. 2022/10/02. 2022.
- [67] Google. *Android Platform Architecture*. <https://developer.android.com/guide/platform>. 2022/09/28. 2022.
- [68] Google. *Application Fundamentals*. <https://developer.android.com/guide/components/fundamentals>. 2022/10/02. 2022.
- [69] Google. *Permission*. <https://developer.android.com/reference/android/Manifest.permission>. 2022/04/28. 2022.
- [70] Google. *Policy Center*. <https://support.google.com/googleplay/android-developer/topic/9858052>. 2022/12/01. 2022.

BIBLIOGRAPHY

- [71] Google. *Obtaining Consent with the User Messaging Platform*.
- [72] Gorla, A., Tavecchia, I., Gross, F., and Zeller, A. Checking app behavior against app descriptions. In: *ICSE*. 2014.
- [73] Group, I. E. -. G. I. *The definition of Personal Data - Working Paper 02/2017*. https://iabeurope.eu/wp-content/uploads/2019/08/20170719-IABEU-GIG-Working-Paper02_Personal-Data.pdf. 2017.
- [74] Hanley, J. and McNeil, B. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology* (1982).
- [75] HasanPour, S., Rouhani, M., Fayyaz, M., and Sabokrou, M. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *CoRR* (2016).
- [76] He, H. and Garcia, E. Learning from imbalanced data. *TKDE* (2009).
- [77] Hornyack, P., Han, S., Jung, J., Schechter, S., and Wetherall, D. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In: *CCS*. 2011.
- [78] Hotho, A., Staab, S., and Stumme, G. Ontologies improve text document clustering. In: *ICDM*. 2003.
- [79] Hox, J., Moerbeek, M., and Van de Schoot, R. *Multilevel analysis: Techniques and applications*. 2017.
- [80] Huang, J., Zhang, X., Tan, L., Wang, P., and Liang, B. Asdroid: detecting stealthy behaviors in android applications by user interface and program behavior contradiction. In: *ICSE*. 2014.
- [81] ICCL. *Landmark litigation*. <https://www.iccl.ie/rtb-june-2021/>. 2022/03/08. 2022.
- [82] Isherwood, S. Graphics and semantics: the relationship between what is seen and what is meant in icon design. In: *EPCE*. 2009.
- [83] Kampanos, G., Shahandashti, S. F., and Name, N. Accept all: the landscape of cookie banners in greece and the uk. In: *IFIP SEC*. 2021.
- [84] Kelley, P., Consolvo, S., Cranor, L., Jung, J., Sadeh, N., and Wetherall, D. A conundrum of permissions: installing applications on an android smartphone. In: *FC*. 2012.
- [85] Koch, S., Wessels, M., Altpeter, B., Olvermann, M., and Johns, M. Keeping privacy labels honest. *PoPETs* (2022).
- [86] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *IJCAI*. 1995.
- [87] Kollnig, K., Dewitte, P., Van Kleek, M., Wang, G., Omeiza, D., Webb, H., and Shadbolt, N. A fait accompli? an empirical study into the absence of consent to third-party tracking in android apps. In: *SOUPS*. 2021.
- [88] Krizhevsky, A., Sutskever, I., and Hinton, G. Imagenet classification with deep convolutional neural networks. In: *NIPS*. 2012.

-
- [89] LABOR, U. D. O. *Minimum Wage*. <https://www.dol.gov/general/topic/wages/minimumwage>. 2020/01/01. 2020.
- [90] Leenes, R. and Kosta, E. Taming the cookie monster with dutch law—a tale of regulatory failure. *Computer Law & Security Review* (2015).
- [91] Li, F., Durumeric, Z., Czyz, J., Karami, M., Bailey, M., McCoy, D., Savage, S., and Paxson, V. You’ve got vulnerability: exploring effective vulnerability notifications. In: *USENIX Security*. 2016.
- [92] Li, L., Bissyand, T., Papadakis, M., Rasthofer, S., Bartel, A., Ocateau, D., JK, and Traon, L. Static analysis of android apps: a systematic literature review. *IST* (2017).
- [93] Li, L., Bissyandé, T. F., Papadakis, M., Rasthofer, S., Bartel, A., Ocateau, D., Klein, J., and Traon, L. Static analysis of android apps: a systematic literature review. *Information and Software Technology* (2017).
- [94] Li, L., Bissyandé, T., Klein, J., and Le Traon, Y. An Investigation into the Use of Common Libraries in Android Apps. In: *Technique Report*. 2015.
- [95] Li, M., Wang, W., Wang, P., Wang, S., Wu, D., Liu, J., Xue, R., and Huo, W. Libd: scalable and precise third-party library detection in android markets. In: *ICSE*. 2017.
- [96] Li, Y., Yang, Z., Guo, Y., and Chen, X. Droidbot: a lightweight ui-guided test input generator for android. In: *ICSE-C*. 2017.
- [97] Li, Y., Yang, Z., Guo, Y., and Chen, X. Droidbot: a lightweight ui-guided test input generator for android. In: *ICSE-C*. 2017.
- [98] Lin, J., Amini, S., Hong, J. I., Sadeh, N., Lindqvist, J., and Zhang, J. Expectation and purpose: understanding users’ mental models of mobile app privacy through crowdsourcing. In: *UbiComp*. 2012.
- [99] Liu, X., Leng, Y., Yang, W., Wang, W., Zhai, C., and Xie, T. A large-scale empirical study on android runtime-permission rationale messages. In: *VL/HCC*. 2018.
- [100] Lo, R., He, B., and Ounis, I. Automatically building a stopword list for an information retrieval system. In: *DIR*. 2005.
- [101] Ma, Z., Wang, H., Guo, Y., and Chen, X. Libradar: fast and accurate detection of third-party libraries in android apps. In: *ICSE*. 2016.
- [102] Mao, K., Harman, M., and Jia, Y. Sapienz: multi-objective automated testing for android applications. In: *ISSTA*. 2016.
- [103] Matte, C., Bielova, N., and Santos, C. Do cookie banners respect my choice?: measuring legal compliance of banners from iab europe’s transparency and consent framework. In: *SP*. 2020.
- [104] McCullagh, P. Regression models for ordinal data. *Journal of the Royal Statistical Society: Series B (Methodological)* (1980).

BIBLIOGRAPHY

- [105] McDougall, S. and Isherwood, S. What's in a name? the role of graphics, functions, and their interrelationships in icon identification. *Behavior research methods* (2009).
- [106] Meade, A. and Craig, S. Identifying careless responses in survey data. *Psychological methods* (2012).
- [107] Micinski, K., Votipka, D., Stevens, R., Kofinas, N., Mazurek, M., and Foster, J. User interactions and permission use on android. In: *CHI*. 2017.
- [108] Miller, G. Wordnet: a lexical database for english. *COMMUN ACM* (1995).
- [109] Momen, N., Hatamian, M., and Fritsch, L. Did app privacy improve after the gdpr? *IEEE Security & Privacy* (2019).
- [110] Moran, K., Tufano, M., Bernal-Cárdenas, C., Linares-Vásquez, M., Bavota, G., Vendome, C., Di Penta, M., and Poshyvanyk, D. Mdroid+: a mutation testing framework for android. In: *ICSE*. 2018.
- [111] myTarget. *Privacy and GDPR*. <https://target.my.com/help/partners/privacygdpr/en>. 2021/02/01. 2021.
- [112] Nguyen, D., Derr, E., Backes, M., and Bugiel, S. Short text, large effect: measuring the impact of user reviews on android app security & privacy. In: *SP*. 2019.
- [113] Nguyen, T. T., Nguyen, D. C., Schilling, M., Wang, G., and Backes, M. Measuring user perception for detecting unexpected access to sensitive resource in mobile apps. In: *ASIA CCS*. 2020.
- [114] Nielsen, J. and Sano, D. Sunweb: user interface design for sun microsystem's internal web. *Computer Networks and ISDN Systems* (1995).
- [115] Norman, D. *Things that make us smart: Defending human attributes in the age of the machine*. 2014.
- [116] NOYB – European Center for Digital Rights. *Google: If you don't want us to track your phone – just get another tracking ID!* <https://noyb.eu/en/complaint-filed-against-google-tracking-id>. 2021/01/17. 2021.
- [117] objection. *Runtime Mobile Exploration*. <https://github.com/sensepost/objection>. 2021/01/17. 2021.
- [118] Octeau, D., Luchau, D., Dering, M., Jha, S., and McDaniel, P. Composite constant propagation: application to android inter-component communication analysis. In: *ICSE*. 2015.
- [119] Octeau, D., McDaniel, P., Jha, S., Bartel, A., Bodden, E., Klein, J., and Le Traon, Y. Effective inter-component communication mapping in android: an essential step towards holistic security analysis. In: *USENIX Security*. 2013.
- [120] Oltramari, A., Piraviperumal, D., Schaub, F., Wilson, S., Cherivirala, S., Norton, T., Russell, N., Story, P., Reidenberg, J., and Sadeh, N. Privonto: a semantic framework for the analysis of privacy policies. *Semantic Web* (2017).
- [121] Oltrogge, M., Huaman, N., Amft, S., Acar, Y., Backes, M., and Fahl, S. Why eve and mallory still love android: revisiting {tls}({in} security) in android applications. In: *USENIX Security*. 2021.

-
- [122] Pan, X., Cao, Y., Du, X., He, B., Fang, G., Shao, R., and Chen, Y. Flowcog: context-aware semantics extraction and analysis of information flow leaks in android apps. In: *USENIX Security*. 2018.
- [123] Pan, X., Cao, Y., Du, X., He, B., Fang, G., Shao, R., and Chen, Y. Flowcog: context-aware semantics extraction and analysis of information flow leaks in android apps. In: *USENIX Security*. 2018.
- [124] Pandita, R., Xiao, X., Yang, W., Enck, W., and Xie, T. Whyper: towards automating risk assessment of mobile applications. In: *SEC*. 2013.
- [125] Party, D. P. W. *Opinion 4/2010 on the European code of conduct of FEDMA for the use of personal data in direct marketing*. https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2010/wp174_en.pdf. 2022/04/28. 2010.
- [126] Party, D. P. W. *Guidelines on Consent under Regulation 2016/679 (wp259rev.01)*. https://ec.europa.eu/newsroom/article29/item-detail.cfm?item_id=623051. 2020/09/04. 2016.
- [127] Peddinti, S. T., Bilogrevic, I., Taft, N., Pelikan, M., Erlingsson, U., Anthonysamy, P., and Hogben, G. Reducing permission requests in mobile apps. In: *Proceedings of ACM Internet Measurement Conference (IMC)*. 2019.
- [128] PIPL. *Personal Information Protection Law of the People's Republic of China*. http://en.npc.gov.cn.cdurl.cn/2021-12/29/c_694559.htm. 2023/07/01. 2023.
- [129] Prati, R., Batista, G., and Monard, M. Class imbalances versus class overlapping: an analysis of a learning system behavior. In: *MICAI*. 2004.
- [130] Privacy-International-NGO. *Most cookie banners are annoying and deceptive. This is not consent*. <https://privacyinternational.org/explainer/2975/most-cookie-banners-are-annoying-and-deceptive-not-consent>. 2023/12/01. 2023.
- [131] publicsuffixlist. *publicsuffixlist*. <https://github.com/ko-zu/psl>. 2021/05. 2021.
- [132] Qu, Z., Rastogi, V., Zhang, X., Chen, Y., Zhu, T., and Chen, Z. Autocog: measuring the description-to-permission fidelity in android applications. In: *CCS*. 2014.
- [133] Razaghpanah, A., Nithyanand, R., Vallina-Rodriguez, N., Sundaresan, S., Allman, M., Kreibich, C., and Gill, P. Apps, trackers, privacy, and regulators: a global study of the mobile tracking ecosystem. In: *NDSS*. 2018.
- [134] Reardon, J., Feal, Á., Wijesekera, P., On, A. E. B., Vallina-Rodriguez, N., and Egelman, S. 50 ways to leak your data: an exploration of apps' circumvention of the android permissions system. In: *USENIX Security*. 2019.
- [135] Regulation, G. D. P. 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. *OJEU* (2016).

BIBLIOGRAPHY

- [136] Ren, J., Rao, A., Lindorfer, M., Legout, A., and Choffnes, D. Recon: revealing and controlling pii leaks in mobile network traffic. In: *MobiSys*. 2016.
- [137] Reyes, I., Wijesekera, P., Reardon, J., On, A. E. B., Razaghpanah, A., Vallina-Rodriguez, N., and Egelman, S. “won’t somebody think of the children?” examining coppa compliance at scale. *PETS* (2018).
- [138] Rountev, A. and Yan, D. Static reference analysis for gui objects in android software. In: *CGO*. 2014.
- [139] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In: *NIPS*. 2016.
- [140] Sanchez-Rola, I., Dell’Amico, M., Kotzias, P., Balzarotti, D., Bilge, L., Vervier, P.-A., and Santos, I. Can i opt out yet? gdpr and the global illusion of cookie control. In: *Asia CCS*. 2019.
- [141] Scudder, H. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory* (1965).
- [142] Slavin, R., Wang, X., Hosseini, M. B., Hester, J., Krishnan, R., Bhatia, J., Breaux, T. D., and Niu, J. Toward a framework for detecting privacy policy violations in android application code. In: *ICSE*. 2016.
- [143] Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., and Demirbas, M. Short text classification in twitter to improve information filtering. In: *ACM SIGIR*. 2010.
- [144] Statcounter. *Mobile Operating System Market Share Worldwide*. <https://gs.statcounter.com/os-market-share/mobile/worldwide>. 2022/09/28. 2022.
- [145] Stock, B., Pellegrino, G., Li, F., Backes, M., and Rossow, C. Didn’t you hear me? - towards more successful web vulnerability notifications. In: *NDSS*. 2018.
- [146] Stock, B., Pellegrino, G., Rossow, C., Johns, M., and Backes, M. Hey, you have a problem: on the feasibility of large-scale web vulnerability notification. In: *USENIX Security*. 2016.
- [147] Techcrunch. *France slaps Google \$170M, Facebook \$68M over cookie consent dark patterns*. <https://techcrunch.com/2022/01/06/cnil-facebook-google-cookie-consent-privacy-breaches/>. 2022/04/28. 2022.
- [148] Techcrunch. *How a small French privacy ruling could remake adtech for good*. <https://techcrunch.com/2018/11/20/how-a-small-french-privacy-ruling-could-remake-adtech-for-good/>. 2022/04/28. 2022.
- [149] Tendulkar, V. and Enck, W. An application package configuration approach to mitigating android ssl vulnerabilities. *MoST* (2014).
- [150] tesseract-ocr. *Tesseract OCR*. <https://github.com/tesseract-ocr/tesseract>. 2019.

-
- [151] Traverso, S., Trevisan, M., Giannantoni, L., Mellia, M., and Metwalley, H. Benchmark and comparison of tracker-blockers: should you trust them? In: *TMA*. 2017.
- [152] Trevisan, M., Traverso, S., Bassi, E., and Mellia, M. 4 years of eu cookie law: results and lessons learned. *PETS* (2019).
- [153] Unity3d. *GDPR Compliance*. <https://docs.unity3d.com/Packages/com.unity.ads@3.3/manual/LegalGdpr.html>. 2021/02/01. 2021.
- [154] Unity3d. *Privacy Policy*. <https://unity3d.com/legal/privacy-policy>. 2021/02/01. 2021.
- [155] Utz, C., Degeling, M., Fahl, S., Schaub, F., and Holz, T. (un) informed consent: studying gdpr consent notices in the field. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, 973–990.
- [156] Vallina, P., Feal, Á., Gamba, J., Vallina-Rodriguez, N., and Anta, A. F. Tales from the porn: a comprehensive privacy analysis of the web porn ecosystem. In: *IMC*. 2019.
- [157] Vungle. *Vungle SDK for Android*. <https://github.com/Vungle/Android-SDK/blob/master/CHANGELOG.md>. 2021/02/01. 2021.
- [158] Wang, Y., Zhang, H., and Rountev, A. On the unsoundness of static analysis for android guis. In: *PLDI*. 2016.
- [159] Ward, M. and Pond III, S. Using virtual presence and survey instructions to minimize careless responding on internet-based surveys. *Computers in Human Behavior* (2015).
- [160] Ward Jr, J. H. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* (1963).
- [161] Watanabe, T., Akiyama, M., Sakai, T., and Mori, T. Understanding the inconsistencies between text descriptions and the use of privacy-sensitive resources of mobile apps. In: *SOUPS*. 2015.
- [162] webshrinker. *webshrinker*. <https://www.webshrinker.com/>. 2021/05. 2021.
- [163] Weir, C., Hermann, B., and Fahl, S. From needs to actions to secure apps? the effect of requirements and developer practices on app security. In: *USENIX Security*. 2020.
- [164] Wermke, D., Huaman, N., Acar, Y., Reaves, B., Traynor, P., and Fahl, S. A large scale investigation of obfuscation use in google play. In: *ACSAC*. 2018.
- [165] Wiedenbeck, S. The use of icons and labels in an end user application program: an empirical study of learning and retention. *Behaviour & Information Technology* (1999).
- [166] Wijesekera, P., Baokar, A., Hosseini, A., Egelman, S., Wagner, D., and Beznosov, K. Android permissions remystified: a field study on contextual integrity. In: *USENIX Security*. 2015.

BIBLIOGRAPHY

- [167] Xi, S., Yang, S., Xiao, X., Yao, Y., Xiong, Y., Xu, F., Wang, H., Gao, P., Liu, Z., Xu, F., et al. Deepintent: deep icon-behavior learning for detecting intention-behavior discrepancy in mobile apps. In: *CCS*. 2019.
- [168] Xiao, X., Wang, X., Cao, Z., Wang, H., and Gao, P. Iconintent: automatic identification of sensitive ui widgets based on icon classification for android apps. In: *ICSE*. 2019.
- [169] Xie, Q., Hovy, E., Luong, M., and Le, Q. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252* (2019).
- [170] Xue, L., Zhou, H., Luo, X., Yu, L., Wu, D., Zhou, Y., and Ma, X. Packergrind: an adaptive unpacking system for android apps. *IEEE Trans. Softw. Eng.* (2020).
- [171] Yang, S., Yan, D., Wu, H., Wang, Y., and Rountev, A. Static control-flow analysis of user-driven callbacks in android applications. In: *ICSE*. 2015.
- [172] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., and Wang, X. S. Appintent: analyzing sensitive data transmission in android for privacy leakage detection. In: *CCS*. 2013.
- [173] Yu, L., Luo, X., Liu, X., and Zhang, T. Can we trust the privacy policies of android apps? In: *DSN*. 2016.
- [174] Yu, L., Luo, X., Qian, C., and Wang, S. Revisiting the description-to-behavior fidelity in android applications. In: *SANER*. 2016.
- [175] Zauner, C. Implementation and benchmarking of perceptual image hash functions (2010).
- [176] Zhou, H., Chen, T., Wang, H., Yu, L., Luo, X., Wang, T., and Zhang, W. Ui obfuscation and its effects on automated ui analysis for android apps. In: *ASE*. 2020.
- [177] Zimmeck, S., Wang, Z., Zou, L., Iyengar, R., Liu, B., Schaub, F., Wilson, S., Sadeh, N., Bellovin, S., and Reidenberg, J. Automated analysis of privacy requirements for mobile apps. In: *NDSS*. 2017.
- [178] Zimmeck, S., Wang, Z., Zou, L., Iyengar, R., Liu, B., Schaub, F., Wilson, S., Sadeh, N. M., Bellovin, S. M., and Reidenberg, J. R. Automated analysis of privacy requirements for mobile apps. In: *NDSS*. 2017.

A

Appendix

A.1 GUIBAT

A.1.1 Second Study

In mobile apps, sensitive resources are the resources that involve your private information, or could potentially affect your stored data or the operation of other apps. For example, your contacts is a sensitive resource. In this study, we focus on the following sensitive resources:

- **Contacts:** *involve the contacts information such as modify your contacts, find accounts on the device, read your contacts, etc;*
- **Phone:** *involve the telephone features such as read call log, read phone status, access your phone number, directly call phone numbers, write call log, reroute outgoing calls, etc;*
- **Calendar:** *involve calendar information such as read calendar events, add or modify calendar events, etc;*
- **Camera:** *access the camera such as taking pictures and videos, turning the flashlight on, etc;*
- **Location:** *access device location such as precise location (GPS and network-based), approximate location (network-based), etc;*
- **Storage:** *involve the storage such as read the contents (e.g., photos, media, or files) of your device, modify or delete the contents of your SD card, etc;*
- **Microphone:** *access the microphone such as record audio, etc;*
- **SMS:** *involve SMS information such as read your text messages (SMS or MMS), receive text messages, send and view SMS messages, read cell broadcast messages, etc.*

Q. A mobile app that accesses the camera to take photos and then saves them to SD card in your phone, what are sensitive related functions that the app involves?

A.1.2 Third Study

Q1. Which of the following sensitive resources would you expect this app to access? (*a list of 8 sensitive resources and their descriptions*)

Q2. Please explain briefly why in your opinion the app requires the sensitive resources you have selected (Free text).

Q3. To serve the app's functionality on the current screen, on a scale from 1 to 7 how much do you expect this app to access the following data? (*a list of sensitive resources that are detected by GUIBAT*)

Q4. Suppose you are pressing the highlighted function (red rectangle), how much do you feel comfort letting this app to access the following data? (*a list of sensitive resources that are detected by GUIBAT*)

A.1.3 Demographic

	Study 1.1	Study 1.2	Study 2	Study 3
Number of participants	120	294	45	445
<i>Gender</i>				
Female	42 (35.0%)	122 (41.5%)	14 (31.11%)	258 (57.98%)
Male	78 (65.0%)	171 (58.16%)	31 (68.89%)	184 (41.35%)
Other	0	1 (0.34%)	0	2 (0.45%)
No answer	0	0	0	1 (0.22%)
<i>Age group</i>				
18–30	52 (43.33%)	103 (35.03%)	17 (37.78%)	155 (34.83%)
31–40	43 (35.83%)	117 (39.8%)	23 (51.11%)	173 (38.88%)
41–50	14 (11.67%)	47 (15.99%)	3 (6.67%)	66 (14.83%)
51–60	6 (5.0%)	17 (5.78%)	1 (2.22%)	39 (8.76%)
61–70	5 (4.17%)	8 (2.72%)	1 (2.22%)	11 (2.47%)
>=71	0	2 (0.68%)	0	1 (0.22%)
<i>Computer science background</i>				
Yes	25 (20.83%)	69 (23.47%)	12 (26.67%)	76 (17.08%)
No	95 (79.17%)	225 (76.53%)	33 (73.33%)	369 (82.92%)
<i>Primary phone</i>				
Android	44 (36.67%)	199 (67.69%)	31 (68.89%)	275 (61.80%)
iPhone	75 (62.5%)	93 (31.63%)	14 (31.11%)	170 (38.20%)
Other	1 (0.83%)	2 (0.68%)	0	0
<i>Education level</i>				
Less than high school	0	1 (0.34%)	0	1 (0.22%)
High school graduate	19 (15.83%)	31 (10.54%)	5 (11.11%)	38 (8.54%)
Some college, no degree	32 (26.67%)	65 (22.11%)	13 (28.89%)	87 (19.55%)
Associate’s degree	9 (7.5%)	38 (12.93%)	2 (4.44%)	56 (12.58%)
Bachelor degree	51 (42.5%)	128 (43.54%)	17 (37.78%)	189 (42.47%)
Master degree	6 (5.0%)	29 (9.86%)	7 (15.56%)	59 (13.26%)
Ph.D	1 (0.83%)	1 (0.34%)	0	6 (1.35%)
Graduate/prof. degree	2 (1.67%)	1 (0.34%)	1 (2.22%)	8 (1.80%)
Others	0	0	0	1 (0.22%)
<i>Ethnicity</i>				
White/Caucasian	91 (75.83%)	218 (74.15%)	30 (66.67%)	333 (74.83%)
Black/African American	9 (7.5%)	33 (11.22%)	6 (13.33%)	42 (9.44%)
Asian	11 (9.17%)	22 (7.48%)	5 (11.11%)	41 (9.21%)
Hispanic/Latino	4 (3.33%)	12 (4.08%)	1 (2.22%)	17 (3.82%)
Native American/Alaska	1 (0.83%)	2 (0.68%)	1 (2.22%)	3 (0.67%)
Native Hawaiian/Pacific	0	0	0	1 (0.22%)
Islander	0	0	0	0
Middle Eastern	1 (0.83%)	0	0	1 (0.22%)
Other	3 (2.5%)	7 (2.38%)	2 (4.44%)	7 (1.57%)

Table A.1: Demographics of participants from our studies.

A.1.4 Evaluating of GUIBAT and GATOR

Table A.2 shows the differences between the results of GUIBAT and GATOR by manually verifying.

A.2 GDPR's Explicit Consent: Email Template

Dear **\$developer** team,

We are a team of academic researchers from the **\$affiliation**, conducting a research project on user consent and GDPR (EU General Data Protection Regulation) compliance of mobile apps. Please note that this email is part of an academic research project and is not meant to sell any products or services.

As part of our analysis, we investigate the sharing of users' personal information (e.g., user IP address, persistent identifiers, tracking identifiers) with third-party services to show personalized or behavioral advertising. Based on our analysis, your app shares some personal user information to such services without obtaining prior explicit consent from users. We have prepared a detailed report on the analysis methodology, the data being sent out, and the parties involved. You can access this through our (password-protected) Web interface at **\$report_url** (please do not publish this URL as it is personalized for your app). By analyzing the legal documents (e.g., the terms of service, privacy policies, developer guidelines, and contracts) provided by the third-party services in question, we concluded that your app might be non-compliant with the consent requirements by the GDPR [1]. In most cases, in order to be legally compliant, an app is required to obtain explicit consent from users situated in the European Union before sharing users' personal data with third parties for personalized ads, if those third parties act as a data controller. Please note that we do not offer a conclusive legal assessment or consultancy on an individual app's compliance as there might be an alternative lawful basis present for data sharing with a third party other than consent. As this email is part of a research project in which we are trying to understand the reasons for GDPR compliance issues of mobile apps in the wild, it would be immensely helpful to provide us with feedback regarding your apps.

(1) Were you aware of the types of data that are being collected and transmitted when you include third-party SDK(s) into your apps? Were you aware that these types of data could be considered personal data under the GDPR?

(2) Are there specific reasons why your app does not implement explicit consent?

(3) Are there any changes you plan to apply to remedy the outlined issues? What type of support (e.g., documentation or automated tools) would be beneficial for you?

Should you have further questions or wish not to receive any further communication, please contact us, and we will diligently follow the request.

Best regards

\$researchers

[1] (The full-text reference of [135] was added in the email.)

A.3 Manual Version Analysis

We first analyze the app network traffic to find parameters that indicate the SDK version. These are: `unity3d.com` (*x-unity-version*, *sdkversion*, *sdk_version_name*, *sdk_ver*, *sdkversion*); `flurry.com` (*fl.sdk.version.code*); `vungle.com` (*user-agent*, *sdk*); `my.com`: (*mytracker_ver*); `amazon-adsystem.com` (*adsdk*). Further, we manually verify the results with the identified SDK(s) release notes. We then used this knowledge to detect versions all apps that sent personal data.

A.4 GDPR's Consent Requirements: Email Template

Dear **\$developer**,

We are a team of researchers from the **\$affiliation**, studying the GDPR (EU General Data Protection Regulation) consent for sharing users' personal data with third-party data controllers (use the data for their own purposes) in mobile apps. We are reaching out to you because of a potential GDPR violation in your app(s).

This email is a part of an academic research project. We do not sell any products or services.

Under GDPR[1], an app is required to obtain consent from users situated in the EU for sharing users' personal data with third-party data controllers. Based on our analysis, your app shares some user personal data with such third-party services (e.g., ads), and it has implemented a consent notice for such data sharing. However, in our automated tests, your app shares data without any given consent, or even after rejecting that consent/opting out, your app seemingly still shared data with third-party data controllers.

We have prepared a detailed report on the analysis methodology, the data being sent out, the parties involved, and all other details. You can access this through our (password-protected) Web interface at **\$web_url** (please do not publish this URL as it is personalized for your app).

Note that we are not threatening legal action or plan to involve regulators, or offer a conclusive legal assessment or consultancy. We aim to enable developers to address the issues before other parties might take any legal actions. Since we are also trying to understand the reasons for GDPR compliance issues of mobile apps in the wild, it would be immensely helpful to provide us with feedback regarding your app(s).

- Were you aware that personal data – based on the GDPR – (i.e., **\$sent_data**) is being collected and transmitted when you include third-party SDK(s) into your apps?
- Are you aware of the GDPR consent requirements? For example, that the consent will not be considered "free" if the user is unable to refuse or withdraw their consent.

- What type of support (e.g., documentation or automated tools) would benefit you to address such issues?

Please contact us if you have further questions or wish not to receive further communication. We will diligently follow the request.

Best Regards, **\$researchers**

Disclaimer: This study is part of a research project of the **\$affiliation**. The collected information will be used for scientific purposes only. Your responses are pseudonymous. We do NOT collect any personal information, publicize or perform any actions against your apps, and your company. [1] (The full-text reference of [135] was added in the email.)

A.4. GDPR'S CONSENT REQUIREMENTS: EMAIL TEMPLATE

App	GUIBAT/GATOR 3.5		GUIBAT's Refinement upon GATOR		
	V	C	(+) V	(+) C	(-) C
AardDictionary 1.4.1	218/218	105/105	0	0	0
FTP Server 1.0.4	14/0	25/0	14	25	0
Battery Circle 1.4.1	30/30	7/19	0	0	12
LolcatBuilder 0.2.1	9/1	5/0	8	5	0
SpriteMethodTest 2.0a	4/4	0/0	0	0	0
Alarm Clock 1.4.1	46/40	20/32	6	2	14
Manpages 0.2.1	13/13	7/7	0	0	0
Auto Answer 1.4.1	10/0	8/0	10	8	0
RandomMusicPlayer 2.0a	9/9	6/6	0	0	0
AnyCut 1.4.1	15/15	5/5	0	0	0
HNDroid 1	65/62	105/106	3	0	1
DivideAndConquer 1	1/1	0/0	0	0	0
Photostream 2.0a	7/2	0/0	5	0	0
Multi SMS 0.2.1	47/47	30/30	0	0	0
World Clock 2.0a	24/24	12/18	0	0	6
Ringdroid 2.0a	34/34	56/68	0	0	12
Yahtzee 2.0a	30/30	12/12	0	0	0
aagtl 2.8.11	60/60	44/44	0	0	0
WhoHasMyStuff 2.0a	40/40	58/58	0	0	0
Mirrored 0.2.1	30/21	15/15	9	0	0
WeightChart 2.0a	65/32	115/21	33	100	6
ADSDroid 1.4.1	6/6	2/2	0	0	0
myLock 0.2.1	33/16	3/9	17	0	6
LockPatternGenerator 0.2.1	19/10	19/13	9	6	0
MunchLife 0.2.1	20/14	20/19	6	1	0
CountdownTimer 1.0.31	45/35	8/10	10	0	2
NetCounter 0.1.1	28/20	10/6	8	4	0
TippyTipper 2.0a	95/83	28/188	12	0	160
BatteryDog 1.4.1	15/15	32/32	0	0	0
Dialer2 1	55/45	57/57	10	0	0
DalvikExplorer 2.11	134/134	126/126	0	0	0
Blokish 2.2	38/32	13/13	6	0	0
ZooBorns 2.0a	12/12	8/8	0	0	0
Wordpress_394 2.0a	46/40	21/30	6	3	12
Total	1,317/1,145	982/1,059	172 (15.02% of 1,145)	154 (21.81% of 1,059)	231

Table A.2: The evaluating results of GUIBAT and Gator 3.5 on a set of 34 apps. V=Views. C=Callbacks. (+)=New. (-)=Removed.

No.	Name	Domain Names	GDPR Solution	Earliest consent support SDK version	Note
1	Facebook	facebook.com	Not require consent	—	Under GDPR, developer are required to obtain end User consent before sending data via our SDK
2	Unity	unity3d.com	Consent API	3.3.0	
3	Flurry	flurry.com	Consent API	10.0.0	
4	AppsFlyer	appsflyer.com	Not require consent	—	Providing APIs for opt-in and opt-out
5	Chartboost	chartboost.com	Consent API	7.3.0	
6	SuperSonic	supersonicads.com	Consent API	6.7.9	
7	StartApp	startappservice.com	Consent API	1.2.0	
8	AdColony	adcolony.com	Consent API	3.3.4	
9	Branch	branch.io	Not require consent	—	Providing APIs for opt-in and opt-out
10	Vungle	vungle.com	Consent API	6.2.5	
11	Applovin	applovin.com	Consent API	8.0.1	
12	Tapjoy	tapjoy.com	Consent API	11.12.2	GDPR-compliant based on "legitimate interest"
13	ConsoliAds	consoliads.com	Consent API	—	
13	BidMachine	bidmachine.io	Consent API	1.3.0	
14	MoPub	mopub.com	Consent API	5.0.0	
15	Presage	presage.io	—	—	
16	AdinCube	adincube.com	—	—	
17	Ogury	ogury.io	Consent API	4.1.4	
18	Amazon	amazon-adsystem.com	—	—	
19	InMobi	inmobi.com	Consent API	7.1.0	
20	Adbrix	ad-brix.com	Not require consent	—	Providing APIs for opt-in and opt-out
21	Adbrix	adbrix.io	Not require consent	—	Providing APIs for opt-in and opt-out
22-23	Tenjin	tenjin.com, tenjin.io	Not require consent	—	Providing APIs for opt-in and opt-out
24	Mobvista	rayjump.com	—	—	
25	Tenjin	tenjin.io	—	—	
26	Appnext	appnext.com	Consent API	2.3.0	
27	Pollfish	pollfish.com	Not require consent	—	Have to provide disclosure for using this SDK
28	My.com	my.com	—	—	
29	Soomla	soom.la	Consent API	—	Should be a bad practice since default behavior is TRUE
30	Localytics	localytics.com	Not require consent	2.1.0	Providing APIs for opt-in and opt-out

Table A.3: Companies detected as ad-related, for which our analysis of legal documents indicate they act as data controllers.

A.4. GDPR'S CONSENT REQUIREMENTS: EMAIL TEMPLATE

No.	Name	Domain Names	GDPR Solution	Earliest consent support SDK version	Note
31	Tapdaq	tapdaq.com	Consent API	6.2.2	
32	Leanplum	leanplum.com	Not require consent -		Providing APIs for opt-in and opt-out
33	Criteo	criteo.com	CMP	3.7.0	
34	WebEngage	webengage.com	—		
35	Smart AdServer	smartadserver.com	CMP	1.2.0	
36	Umeng	umeng.com	—	—	
37	omtrdc.net	omtrdc.net	—	—	
38	MobiRoller	mobiroller.com	—	—	
39	Kiip	kiip.me	not clear	—	Due to GDPR regulations, NinthDecimal is now blocking all ad requests from the affected EEA regions.
40	Adtrace	adtrace.io	Not require consent -		Have to provide disclosure for using this SDK
41	Airpush	airpush.com	—	—	
42	Inloco	inlococomedia.com	Consent API	4.0.0	
43	PubMatic	pubmatic.com	—	—	
44	Tapstream	tapstream.com	—	—	
45	YovoAds	yovoads.com	—	—	

Table A.4: Companies detected as ad-related, for which our analysis of legal documents indicate they act as data controllers.