Towards Understanding and Improving Software Professionals' Security and Privacy Practices

Noura Alomar

December 20, 2024

Towards Understanding and Improving Software Professionals' Security and Privacy Practices

By

Noura Nassir Alomar

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Fall 2024

Towards Understanding and Improving Software Professionals' Security and Privacy Practices

Abstract

Towards Understanding and Improving Software Professionals' Security and Privacy Practices

by

Noura Nassir Alomar

Doctor of Philosophy in Computer Science

University of California, Berkeley

Doctor Serge Egelman, Co-chair

Professor David Wagner, Co-chair

Organizations operating in various industries are continuously experiencing the negative consequences of not sufficiently addressing security and privacy issues in their software development processes. Exploitation of such issues is increasingly leading to breaches of sensitive user data and exposing organizations to legal liabilities. Depending on organizations' sizes and the maturity levels of their development processes, effective handling of these issues might fall under the responsibilities of technical and managerial roles that have different specializations. Throughout the process of developing or maintaining software, the practices adopted by these roles can lead to introducing preventable security and privacy issues that have serious implications to their organizations. Despite the prevalence of these issues, we are yet to obtain sufficient understanding of why they persist, and how the characteristics of organizational software development or maintenance processes are influencing the privacy or security practices adopted software professionals who contribute to these processes.

This dissertation presents the results of a series of holistic process-oriented investigations of the factors that are hindering timely detection and remediation of security and privacy issues by organizations. We start by qualitatively examining how process-related factors and interactions with managerial roles were shaping the engineering practices of software developers and testers tasked with handling these issues in organizations operating in the United States and several other countries. We then present the results of in-depth longitudinal measurements of the prevalence of privacy and security issues in thousands of Android apps that were published on the Google Play Store and which were leading their developers to potentially be in violation of applicable privacy regulations. To obtain insights into why these issues existed in the tested apps, we supplement the results of our technical measurements with qualitative data collected through semi-structured interviews and surveys that we targeted to professionals who were involved in the development of the same apps.

Our results identify a range of non-technical factors that hindered informed decision-making on how security and privacy issues should be handled, which include the lack of organizational processes that facilitated information sharing between the various roles whose effective cooperation was needed (e.g., software developers and security engineers). However, the results of our technical and qualitative investigations consistently identified enforcement of regulatory and industry compliance requirements as one of the main factors that was driving organizations' efforts to detect or remediate security or privacy issues. Notably, our large-scale longitudinal measurements of the behaviors of Android apps provide evidence of overall improvement in organizational security and privacy practices, which likely resulted from Google Play's ongoing enforcement efforts of privacy compliance requirements over the past few years (2018 to 2023).

However, we also show that use of third-party code is introducing complexities in software development processes and continuing to lead software developers to introduce privacy issues (e.g., exfiltration of personal data). The discussions we had with developers of these apps showed that most of them lacked awareness of the behaviors of their own apps, did not have sufficient understanding of their privacy compliance obligations, and did not follow systematic approaches to vetting third-party Software Development Kits (SDKs) for inclusion in their apps. These challenges call for the need of reducing the burden of privacy compliance on software developers by providing them with usable guidance that can help them address security and privacy issues early in their software development processes.

We hope that this dissertation will inform regulatory debates about the challenges that are hindering effective handling of security or privacy issues in practice, and the types of interventions that can be incorporated in software development processes to guide software professionals through how to address these issues in a timely fashion.

To my parents, Nassir and Thana, and my brother Mohammed.

Contents

Contents **ii**

List of Figures **v**

List of Tables **vii**

1 Introduction **1**
1.1 Contributions of This Dissertation 3
1.2 Thesis Statement 4
1.3 Roadmap for This Dissertation 4

2 Related Work **7**
2.1 Security and Privacy Practices 7
2.2 Measurements of Security and Privacy Issues 9
2.3 Organizational Processes 11
2.4 Interventions to Help Software Professionals 12

3 Understanding Organizational Vulnerability Discovery and Management Processes **14**
3.1 Introduction 14
3.2 Methodology 16
3.3 Security Teams 17
3.4 The Vulnerability Management Pipeline 19
3.5 Strategic Decision Making 20
3.6 Vulnerability Discovery 27
3.7 Vulnerability Remediation 31
3.8 Discussion 33
3.9 Recommendations 35
3.10 Conclusion 36

4 Understanding Organizational Vulnerability Remediation Processes **38**
4.1 Introduction 38
4.2 Methodology 40

4.3 Results and Analysis 41
4.4 Discussion 62
4.5 Limitations 65
4.6 Conclusion 66

5 Understanding Organizational Privacy Compliance Processes 67
5.1 Introduction 67
5.2 Kids' Privacy Regulations 68
5.3 Methodology 69
5.4 Results and Analysis 74
5.5 Discussion 92
5.6 Limitations 94
5.7 Conclusion 95

6 The Effect of Platform Policies on App Privacy Compliance: A Large-Scale Study of Child-Directed Apps 96
6.1 Introduction 96
6.2 Background 98
6.3 Methodology 100
6.4 Results 106
6.5 Developer Survey 122
6.6 Discussion 125
6.7 Limitations 128
6.8 Ethics 128
6.9 Conclusion 129

7 Designing Actionable Privacy Guidance: A Study of Privacy Checklists 130
7.1 Introduction 130
7.2 Methodology 132
7.3 Results and Analysis 136
7.4 Discussion 152
7.5 Conclusion 155

8 Conclusion 156
8.1 Role of Organizational Processes 156
8.2 Role of Third Parties 159
8.3 Need for Actionable Guidance 161
8.4 Future Work 162
8.5 Concluding Remarks 164

Bibliography 167

A Supplemental Materials for Chapter 3 186

A.1 Interview Guide 186
A.2 Participants' Demographics 192

B Supplemental Materials for Chapter 4 196
B.1 Interview Guide 196
B.2 Survey Questions 197

C Supplemental Materials for Chapter 5 201
C.1 Survey 1: Organizational Survey 201
C.2 Survey 2: Developer Perspectives 203
C.3 Interview Guide 206

D Supplemental Materials for Chapter 6 208
D.1 Developer survey 208
D.2 Traffic Snapshots 211
D.3 Linking of Different Data Types 212
D.4 Example of a Data Safety Label 213
D.5 Timeline of Policy Changes 215

E Supplemental Materials for Chapter 7 216
E.1 Privacy Checklist: Initial Version 216
E.2 Interactive Privacy Checklist: Questionnaire Part 218
E.3 Interactive Privacy Checklist: More Examples of Checklist Items 223
E.4 Interview Guide 224
E.5 Survey Questions 225

List of Figures

5.1 Number of participants who described themselves as experts, knowledgeable, or unaware of the various privacy laws. 80
5.2 Developers' opinions on how challenging privacy compliance was for them. . . . 82
5.3 Developers' opinions on how app markets should be involved in determining apps' compliance with privacy regulations. 86

6.2 Overview of our data collection approach. 103
6.1 Timeline showing when relevant Google Play's policies [177] were introduced. . . 103
6.3 ironSource's [131] server-side COPPA control. 104
6.4 Top 20 data recipients across all app tests. 108
6.5 Inclusion of COPPA and CCPA flags in exchanged communications with six third-party SDK providers. 117

7.1 A checklist item generated by our tool providing guidance on disclosure of data sharing in privacy policies. 134
7.2 A checklist item generated by our tool providing guidance on configuring Chartboost's SDK [58] for privacy compliance in a child-directed app. 134
7.3 A checklist item generated by our tool providing guidance on updating third-party SDKs to comply with Google Play's policies [177]. 134
7.4 A checklist item generated by our tool providing guidance on removing Meta's advertising SDK [160] from apps that are targeting children only. 134

D.1 Collection of AAIDs and app set ID alongside each other despite correct use of AdColony's [8] COPPA configuration. 211
D.2 Collection of AAIDs and app set ID despite correct use of ironSource's [131] COPPA configuration. 211
D.3 Transmitting AAIDs alongside FIDs to app-measurement.com. 212
D.4 Transmitting AAIDs alongside WiFi MAC addresses and SSAIDs to umeng.com. 212
D.5 Number of apps that transmitted AAIDs alongside other types of personal data to app-measurement.com and other domains. 213
D.6 Example of a data safety label. 214
D.7 Example of a data safety label disclosing that no data is collected or shared. . . 214
D.8 Example of an app that did not have a data safety label. 215

E.1 Question on the the countries where apps are available to users. 219
E.2 Question on the types of data shared with third parties. 220
E.3 Question on whether the third-party analytics or advertising SDKs are integrated. 221
E.4 Question on whether a third-party ad mediation SDK is used. 222
E.5 A checklist item generated by our tool providing guidance on using a server-side privacy configuration offered by ironSource's SDK [131] for compliance with US state laws. 223
E.6 A checklist item generated by our tool providing guidance on using a client-side configuration to communicate whether user consent was obtained. 223
E.7 A checklist item generated by our tool providing guidance on disclosures to include in an app's data safety label. 223

List of Tables

5.1 Third-party SDK privacy flags. 73
5.2 Reasons why respondents in the first survey believed that they were exempt from COPPA [61], GDPR [66], or the CCPA [30]. 78
5.3 Wilcoxon Signed Ranks tests results for participants' familiarity with the different privacy regulations. 79

6.1 Breakdown of the apps tested during the two testing periods. 102
6.2 Number of unique apps that transmitted personal data across all the tests. . . . 107
6.3 Number of apps that declared Android permissions. 110
6.4 Adoption of COPPA-related third-party SDK privacy configuration for recent app versions (*flag value** denotes a privacy-friendly configuration). 115
6.5 Adoption of COPPA-related third-party SDK privacy configuration for recent app versions (*flag value** denotes a privacy-friendly configuration) (cont.). 116
6.6 Adoption of CCPA and GDPR third-party SDK privacy configurations, and configurations that can disable collecting personal data for recent app versions. (*flag value** denotes a privacy-friendly configuration) 118
6.7 Adoption of CCPA and GDPR third-party SDK privacy configurations, and configurations that can disable collecting personal data for recent app versions. (*flag value** denotes a privacy-friendly configuration) (cont.) 119
6.8 Disclosure inaccuracies in privacy labels. 122

A.1 Participants' demographics. 193
A.2 Participants' demographics (cont.). 194
A.3 Participants' demographics (cont.). 195

Acknowledgments

First of all, I would like to thank my PhD advisors, Serge Egelman and David Wagner, for their tremendous support and guidance throughout my PhD journey. Without their encouragement, flexibility, trust in my research abilities, and the constructive feedback they provided on my work over the past few years, this dissertation would not have been possible. I cannot thank them enough for giving me the opportunity to explore the technical side of usable security and privacy and work on timely research problems that were aligned with my research interests.

I also would like to thank Chris Hoofnagle and Sarah Chasins for serving on my qualifying exam and dissertation committees. I am grateful to Chris for his invaluable feedback on the regulatory and policy aspects of cybersecurity and privacy that I covered in my doctoral studies. I also thank Sarah for the inspiring discussions we had about the methods used to conduct the studies presented in this dissertation.

Special thanks to my research collaborators who contributed to the research studies presented in this dissertation. I have had the privilege of working with Joel Reardon, Narseo Vallina-Rodriguez, Primal Wijesekera, Frank Li, Yulie Park, Aniketh Girish, and Edward Qiu. These collaborations not only helped sharpen my technical research skills, but also allowed me to form relationships with software professionals working in international organizations who contributed to my doctoral studies. I am very thankful for their support, and I hope to work with them on tackling more interesting research problems in the future!

I extend my sincere thanks to Nathan Malkin, Conor Gilsenan, and Nikita Samarin for being wonderful friends who never hesitated to help. Thank you for the insightful discussions we had about research, PhD life, and everything else! I truly value your friendship and feel fortunate to have had the opportunity of working with you over the past few years. My thanks extends to the other current or past members of the BLUES lab as well, including Irwin Reyes, Alisa Frik, Julia Bernd, Tomasz Kosinski, Catherine Han, and Ruba Abu-Salma.

Due to the interdisciplinary nature of my research, I also benefited from the help of experts in cybersecurity and privacy law in some of my studies. Jennifer Urban, Antonette Igbenoba, Jordan Fischer, Nomi Conway, and Amit Elazari provided feedback on my research and connected us with experienced security, privacy, and legal professionals who helped with my doctoral studies, for which I am very thankful.

I would also like to express my gratitude to the many undergraduate student researchers who helped with the research studies presented in this dissertation. Thank you, Edmund Wang, Hailey Jang, Yulie Park, Edward Qui, Chris Harjadi, Jinan Jiang, Gurkaran Singh, and Carlynda Gao for your contributions to my PhD research.

Much of the analyses and results presented in this dissertation is based on data collected from software professionals who have years of real-world experiences in helping their organizations with their security or privacy processes. This involved conducting more than 120 interviews with these professionals who willingly shared invaluable insights about their work and donated their time to contribute to our studies. The stimulating discussions I had with these professionals, who had years of experience in organizational security or privacy processes, from the first year of my PhD shaped my research interests, guided my subsequent research plans, and helped me understand how things are done in the industry. I am indebted to all the software developers, security engineers, security managers, and privacy lawyers who participated in my studies for their time and support.

This dissertation is dedicated to my father, my role model, and my source of strength, Nassir, who taught me that patience is the key to everything, my mother, Thana, who made a great job at instilling the desire of seeking excellence in me since I was little, and to my brother Mohammed who endured as much as I did while studying with me in Berkeley for many years and without whom I could not have made it through this long journey. I also would like to express my deepest gratitude to my brothers, Ibrahim, Omar, Abdulrahman, Anas, and Sulaiman, my sisters, Nouf and Ashwag, and my uncle Mohammed for all the help and encouragement I received from them over the past few years. I also have to acknowledge the important roles played by my nephew, Nassir, and niece, Aseel, who were born after I started my PhD in helping me recharge whenever I needed to throughout this challenging and rewarding PhD journey. Thank you all for believing in me, for waiting patiently for my safe return to Saudi Arabia all this time, and for your infinite support.

Finally, I am also thankful to King Saud University and UC Berkeley's Center for Long-Term Cybersecurity for funding my doctoral studies, and to the many supporters in the background who helped me complete this dissertation.

Chapter 1

Introduction

The frequent exploitation of preventable software issues detected in widely-used online services continues to show that security and privacy are not given sufficient consideration in modern software development processes. Inadequate handling of these issues by the various roles involved in these processes is not only leading to expanding their organizations' attack surfaces, but is also subjecting users of software systems to serious security and privacy threats. For example, recent measurement studies consistently found evidence of the prevalence of implementation mistakes that enabled inadvertent sharing of sensitive user data by popular online services [102, 103, 136, 164, 185, 189]. Research conducted over the past few years also showed that these issues were not necessarily caused by individual factors (e.g., lack of expertise in cybersecurity) [86, 169], but that software professionals' engineering practices are shaped by organizational factors, which include the extent to which security and privacy best practices are sufficiently supported in their software development processes [29, 141, 181, 209]. Despite our recognition of the role of these factors, we are yet to understand how they are influencing software professionals' engineering practices, and how that is ultimately leading security and privacy issues to persist in software systems.

Despite the availability of advanced tools that can facilitate detecting security and privacy issues (e.g., [23, 217]), organizations are becoming increasingly in need of skilled software professionals who are capable of utilizing these tools to prevent potential threats. Depending on the complexity of their software architectures, the responsibility of handling these issues might be distributed among various roles, whose collective efforts directly reflect their organizations' preparedness to manage security and privacy risks throughout their software development or maintenance processes. These responsibilities could also be outsourced to external stakeholders who might help organizations improve their security posture by performing technical tests of their infrastructures or remediating discovered security vulnerabilities (e.g., security auditing firms and vendors of third-party software). These external stakeholders could also offer consultations that can guide organizations through how to comply with applicable privacy laws or industry standards (e.g., privacy lawyers). Yet, little is known about how security and privacy decisions are taken by the various technical and managerial roles involved in these processes, and how their abilities to address security and privacy issues in a timely fashion are affected by the characteristics of their organizational processes and the quality of their interactions with their external stakeholders.

At the same time, the increased frequency of severe incidents caused by preventable engineering mistakes is leading governments to require online services to comply with regulatory provisions that aim to provide end users with baselines of protection against security and privacy threats. In the United States, for example, online services might be subject to compliance with the provisions of several federal (e.g., the Children's Online Privacy Protection Act (COPPA) [61]) and state laws (e.g., the California Consumer Privacy Act (CCPA) [30]) that are periodically updated to combat emerging threats. In response to the frequent changes in the privacy compliance landscape, major industry players such as app marketplaces (e.g., the Google Play Store [111] and the App Store [22]) are actively defining and enforcing platform policies that are aligned with the provisions of applicable privacy regulations. However, it is unclear how online services are reacting to these regulatory and policy interventions, and whether these interventions are leading software developers to improve their security and privacy practices.

This dissertation presents the results of our comprehensive investigation of the factors that can lead software professionals to introduce or not address security or privacy issues that affect software systems under their control. We explore how various technical and non-technical roles contribute to security and privacy processes followed to handle these issues and how their work is affected by the business priorities of their organizations. Through a series of semi-structured interviews and survey studies with various roles involved in creating or maintaining software, we shed light on the factors that might hinder informed decision-making by software developers, testers, and managers involved in these processes. We pay particular attention to studying how security and privacy responsibilities are distributed among these roles within their development processes and how the effectiveness of their processes is negatively or positively affected by their interactions with external stakeholders (e.g., external security testers and third-party software developers). By doing so, we identify challenges that hindered effective information sharing between these roles, and discuss how other non-technical factors were negatively affecting security and privacy engineering practices.

Additionally, we systematically evaluate the effectiveness of processes followed by online services to comply with applicable privacy regulations and the impact that these processes have had on software professionals' engineering practices. Our investigations are informed by the industry experiences of software development organizations who were subject to compliance with the regulatory provisions of several privacy laws (e.g., COPPA [61]) as well as Google Play's policies [177]. To quantify the impact of regulatory and policy interventions on software development processes, we complement these investigations with the results of large-scale longitudinal measurements of the privacy behaviors of Android apps that were published on the Google Play Store [111]. This allowed us to observe longitudinal trends and identify factors that can lead to long-term improvements in software professionals' engineering practices. To obtain insights into how to help software professionals improve their security and privacy practices, we conclude this dissertation with an exploration of the limitations of the sources of guidance that are being utilized in software development processes, and propose an intervention that we expect to help mobile app developers address security and privacy issues early in their software development processes.

Taken together, the results of our mixed-methods investigations expand our understanding of the technical and non-technical factors that are contributing to improving or hindering software professionals' abilities to address security and privacy issues throughout their software development or maintenance processes. Our findings can inform organizations about how to evaluate the effectiveness of processes followed to address security and privacy issues and how to facilitate informed decision-making by the various roles involved in these processes. We also hope that our findings will inform regulatory debates about how to strengthen enforcement efforts and app market policies that are ultimately leading software developers to improve their security and privacy engineering practices.

1.1 Contributions of This Dissertation

This dissertation contributes to understanding the breadth of factors that are affecting software professionals' abilities to identity and remediate security and privacy issues that affect their software systems. In all the research studies presented in this dissertation, we investigated how organizational processes are shaping software professionals' engineering practices, and focused on identifying the factors that are leading to positive or negative changes in these practices (e.g., enforcement of regulatory and policy requirements). In this dissertation, we provide answers to the following overarching research questions:

1. How might existing organizational processes and related non-technical factors lead software professionals to introduce new security or privacy issues, or to not address existing ones?
2. To what extent are ongoing regulatory and policy efforts leading to improvements in software professionals' security and privacy practices?
3. What types of guidance do software professionals need to help them identify and address security and privacy issues early in their software development or maintenance processes?

Our investigations are partly based on the results of interview and survey studies conducted with more than 120 software developers, testers, and managers involved in discovering or remediating security or privacy issues in organizations of various sizes that were operating in several countries. We complement the results of our qualitative investigations with in-depth technical measurements of these issues in thousands of software systems (Android mobile apps) implemented or maintained by the same roles. The findings presented in this dissertation demonstrate the trade-offs that software professionals have to make when deciding on how to handle security and privacy issues (e.g., potential conflicts with business priorities) and discuss the types of support that might make it easier for them to do so. They also emphasize the importance of utilizing our understanding of the characteristics of organizational processes to inform the design of interventions that can facilitate the work of professionals involved in these processes. We take one step in this direction by using the results of our technical and qualitative investigations to inform the design of an interactive privacy checklist aimed at helping software developers address privacy issues early in their software development processes. Through an observational interview study with software developers (Chapter 7), we show that our tool helped developers identify privacy issues that existed in their apps and expressed their willingness to incorporate it in their development processes.

We expect the results of the investigations presented in this dissertation to inform organizational, regulatory, and policy efforts aimed at encouraging software professionals to address security and privacy issues early in their software development processes. We also hope that our findings will inform the development of actionable guidance that can provide professionals with the support they need to improve their privacy and security practices.

1.2 Thesis Statement

This thesis demonstrates the various non-technical factors that might lead software professionals to not sufficiently address security and privacy issues that affect their software systems. It provides evidence of the essential roles that organizational processes play in shaping these practices. It also evaluates the extent to which ongoing regulatory and policy efforts are leading organizations to improve the security and privacy engineering practices adopted throughout their software development and maintenance processes.

1.3 Roadmap for This Dissertation

This dissertation presents five research studies that employed a mix of technical, qualitative, and quantitative research methods to obtain holistic understanding of the factors that affect software professionals' abilities to identify or address security and privacy issues that exist in their software systems.

In Chapter 2, we start by giving a review of related literature that investigated the extent to which security and privacy are supported in software development processes or measured adoption of specific security or privacy practices. In the following chapters, we then present the five research studies that we conducted with various roles tasked with developing or testing software products or managing processes followed to identify or handle security or privacy issues in their organizations.

In Chapter 3, we present the results of conducting 53 semi-structured interviews with security professionals who helped organizations of various sizes discover security vulnerabilities that were affecting them. The interviewed professionals helped their organizations with their vulnerability discovery efforts while working as penetration testers, blue teamers, red teamers, external security consultants, or security managers. The findings reported in this chapter identified a range of non-technical factors that hindered the work of these professionals and discussed how these factors can lead to shortcomings in modern organizational vulnerability discovery and management processes. This work appeared at the Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020) [18].

The results of our thematic analyses of the 53 interviews showed that organizations struggled with handling discovered security vulnerabilities. This motivated us to conduct a follow-up research study focusing on obtaining improved understanding of the challenges experienced by security professionals after receiving vulnerability reports through the vulnerability discovery channels employed by their organizations (e.g., bug bounty programs and automated vulnerability scanners).

In Chapter 4, we advance this analysis by reporting the results of conducting additional 21 semi-structured interviews and a follow-up survey with security professionals involved in organizational vulnerability remediation processes. We explore how security professionals involved in these processes made various types of remediation decisions and evaluate how such decisions might lead organizations to remediate, not remediate, or delay remediating discovered security vulnerabilities.

Motivated by the ongoing changes in the privacy compliance landscape that are requiring online service providers to ensure the protection of user privacy, we additionally investigate the effects that regulatory or policy requirements have had on software development processes. In Chapter 5, we present the results of conducting two survey studies and 27 semi-structured interviews with developers of child-directed apps, focusing specifically on understanding the changes made to their development processes to comply with the provisions of applicable privacy regulations. We explore whether organizational processes were providing developers with the types of support they needed to identify and address various types of privacy issues. These issues were caused by implementation mistakes, misconfiguration of embedded third-party SDKs, or inaccurate disclosures of data practices in privacy policies. The results provide insights into the reasons why privacy compliance issues were found in child-directed apps and highlight the need for providing usable privacy guidance that can help developers fulfill their privacy compliance obligations. This work appeared at the 22nd Privacy Enhancing Technologies Symposium (PETS 2022) [16] and the Federal Trade Commission's seventh annual PrivacyCon (2022) [182].

To obtain generalizable insights into developers' privacy practices, we conducted a follow-up large-scale measurement study of the privacy practices adopted by developers of child-directed apps. Chapter 6 focuses on presenting the results of this measurement study which uncovered the extent to which app market policies were leading to improvements in overall privacy practices. We considered various types of privacy issues in our measurements, which include exfiltrations of personal data from child users, transmitting personal data over unencrypted communication channels, misconfiguring third-party Software Development Kits (SDKs) to enable transmitting personal data, and not properly disclosing data practices in privacy labels. We supplement our findings with the results of a follow-up survey that we conducted with developers to obtain improved understanding of the reasons why privacy issues were not addressed in their development processes. This work will appear at the 25th

Privacy Enhancing Technologies Symposium (PETS 2025) [4, 17].

The results of the studies presented in Chapters 5 and 6 emphasize the need for interventions that provide mobile app developers with usable privacy guidance that can enable them to take action on privacy issues before releasing their apps to end users on app marketplaces. We show that developers either did not know how to comprehensively identify privacy requirements applicable to their apps or did not understand how to translate such requirements to actionable tasks in their development processes. We also discuss how the burden of compliance is increasing as requirements enacted by regulators or introduced by app stores are increasing.

In Chapter 7, we take one step in this direction by evaluating the feasibility of using privacy checklists in development processes to reduce the burden of compliance on software developers. We implemented an interactive tool that provides developers with privacy checklists customized to their apps and observed developers while they were using it on their apps. We show that privacy checklists allowed developers to think holistically about their apps' privacy practices and provide recommendations for how actionable privacy guidance should be designed.

We conclude this dissertation, in Chapter 8, with a discussion of the implications of the results of our research studies on software professionals' security and privacy practices, software development processes, and ongoing policy efforts.

Our hope is that the findings of our studies will inform ongoing regulatory debates about the factors that are shaping attention to security and privacy within software development processes and advance our understanding of how to strengthen privacy regulations and platform policies that are ultimately shaping software engineering practices.

Chapter 2

Related Work

This chapter reviews the main lines of research that are relevant to the studies presented in this dissertation. These include research efforts conducted to investigate software developers' security or privacy practices, explain such practices from organizational perspectives, or evaluate the effectiveness of specific interventions that can be incorporated into software development processes.

2.1 Security and Privacy Practices

Prior research studies investigated the reasons that drive software developers to not follow security or privacy best practices when writing or maintaining software from different perspectives. In one line of work, researchers focused on understanding the factors that lead developers to introduce specific types of security or privacy issues or enable them to discover their existence in their software during their development processes [42, 96, 159, 168]. Another line of work examined the extent to which development tools and security or privacy Application Programming Interfaces (APIs) were enabling developers to detect and address issues and the types of improvements that can be made to these tools [5, 196]. Other researchers focused on identifying the challenges experienced by the different roles involved in organizational security or privacy processes (e.g., system administrators [141, 209], bug bounty hunters [15], and mobile app developers [88]). In the following, we survey these studies and discuss how they relate to the work presented in the subsequent chapters of this dissertation.

Security Practices

A great deal of research conducted over the past few years examined how specific types of security vulnerabilities get introduced in code and how to help software developers detect these vulnerabilities [41, 42, 67, 158, 168]. These include understanding whether developers are able to prevent injection attacks [42], employ secure encryption mechanisms [5, 157], and follow security best practices to store passwords [67, 158, 159]. One of the main findings of prior studies was that security vulnerabilities were not necessarily left unaddressed due to developers' lack of sufficient technical skills or security expertise [67, 159, 168, 169]. Researchers also showed that, because security is not part of developers' mental models, developers were more likely to implement secure code when they were primed to do so [41, 67, 168]. Researchers also found that developers might believe that their code is secure even when it contained serious vulnerabilities, which could lead them to release vulnerable products to users [5, 67].

Related research also explored how existing software development APIs might be affecting software developers' abilities to write secure code [5, 157]. Acar et al. [5] showed that many cryptographic APIs suffered from various types of usability shortcomings that made it challenging for developers to use them correctly. They identified failures in the designs or documentation of a number of these APIs which could lead to producing insecure code [5]. Based on analyses of code that used cryptography APIs and qualitative data collected from online developer forums, Nadi et al. [157] similarly showed that security issues can be introduced as a result of developers using APIs that are insufficiently documented, hard to use, or do not use secure defaults. For similar reasons, developers of cryptographic APIs were also shown by Jancar et al. [134] to be discouraged from further improving the security of their APIs. Jancar et al. [134] surveyed these developers to investigate why many such APIs were vulnerable to timing attacks and found that they did not have access to usable tools that can help them audit the security of their APIs and make them resistant to such attacks.

Prior work also shed light on factors that impacted developers' abilities to detect security issues as part of their code review practices [41, 86]. Edmundson et al. recruited developers to identify security vulnerabilities that they injected in a software application [86]. Their results showed that all of their participants, regardless of their security expertise and technical skill sets, were unable to discover all of the vulnerabilities that existed in the application [86]. A follow-up qualitative investigation conducted by Braz et al. [41] showed that developers needed tools that can make code review tasks less effort-intensive (e.g., tools that produce visualizations of software architectures). In a related line of work, Smith et al. [196] demonstrated the need for improving the usability of existing security code analysis tools to maximize their effectiveness. The researchers showed that many such tools lacked actionable security guidance that can help developers comprehend information presented about vulnerabilities and understand how to fix them [196]. Gorski et al. [112] focused on how to design warnings presented in developer tools and identified contextualizing security advice as a main requirement that should be considered by tool designers.

Privacy Practices

Motivated by the need to protect user privacy, recent research examined software developers' technical and non-technical privacy practices [88, 146, 154, 189, 203]. This line of research mostly focused on investigating practices that relate to restricting access to personal data [189], obtaining user consent before transmitting such data [164, 165], disclosing data practices [146, 167], and building privacy-preserving software [154]. Similar to the studies presented in Chapters 5 and 6 in this dissertation, many of these research efforts investigated privacy issues from the perspective of privacy compliance requirements, which could be enacted by regulatory bodies or introduced by major industry players (e.g., mobile app platforms). While there are more studies that investigated software developers' security practices, we believe that there are common challenges that affect security and privacy practices, and therefore lessons learned from the security literature can be utilized to understand how to improve privacy practices as well.

Much of the research presented in this dissertation (Chapters 5, 6, and 7) investigates privacy compliance of Android mobile apps that were published by software development organizations on the Google Play Store [111]. In these chapters, we examine privacy issues that could introduce potential violations of privacy compliance requirements enacted by privacy regulations (e.g., COPPA [61]) or the Google Play Store [111]. These include exfiltration of personal data, disclosure mistakes in privacy policies, and not configuring third-party SDKs for privacy compliance. Below, we discuss how our work relates to the literature that investigated these practices.

In 2014, Balebako et al. [32] presented the results of interviewing and surveying mobile app developers about their privacy practices. They showed that privacy is mostly given secondary priority by developers and demonstrated that they lacked understanding of the data practices of third-party SDKs integrated in their apps. More recent qualitative research conducted by Ekambaranathan et al. [88] showed that similar challenges were experienced by developers of child-directed Android apps. They found that these developers trusted the privacy guidance provided by Google Play [111] and did not perceive embedding third-party SDKs to have serious privacy implications on their child users. Qualitative data collected by Ekambaranathan et al. [88] and Mhaidli et al. [154] also shed light on the decision process followed to select third-party SDKs. Their findings show that developers place a large degree of trust in popular third-party SDKs released by well-known players in this space (e.g., Google Admob [107]).

In Chapter 5, we complement these research efforts by comprehensively investigating privacy practices adopted by developers of child-directed apps. Contrary to prior studies that relied on self-reported data about privacy practices, we employ technical and qualitative methods to compare developers' understandings of the privacy practices of their apps with ground truth data about their actual behaviors. We additionally pay particular attention to examining developers' familiarity with their privacy compliance obligations under applicable privacy regulations and evaluating the extent to which they understood how to configure third-party SDKs for compliance with these regulations.

2.2 Measurements of Security and Privacy Issues

There has been a great deal of work that measured the prevalence of various types of security and privacy issues that affected mobile apps available on app marketplaces. Some of which focused specifically on cryptography issues [87, 89], whereas others focused on quantifying privacy violations that relate to transmitting personal data without consent [164, 189] or not sufficiently disclosing such practices to end users [167].

A number of research studies investigated compliance of mobile apps available in Europe with the General Data Protection Regulation (GDPR) [66]. A large-scale dynamic testing of apps conducted by Nguyen et al. [164] in 2021 showed that 29% of them shared personal data with advertising services without obtaining user consent. In a follow-up work they presented in 2022 [165], they specifically examined the extent to which consent mechanisms used by Android apps were aligned with the consent requirements defined by the GDPR [66]. They identified developer mistakes in a large percentage of apps that led these apps to be potentially in violation of the GDPR [165]. For example, their results showed that declining data sharing in consent prompts might not always prevent apps from sharing data with third-party advertising and tracking services. These observations were further supported by Kollnig et al. [136] who put part of the blame on third-party SDK providers whose documentation was not sufficiently guiding developers through how to obtain user consent before enabling data collection. In a related large-scale measurement study, Razaghpanah et al. [185] found that such behaviors not only allowed sharing data with third parties, but that third parties' privacy policies mentioned that received data will be further shared with other data recipients as well. These studies demonstrated that data collection and sharing issues were prevalent in apps that targeted general audiences, rendering them potentially in violation of privacy regulations (e.g., the GDPR [66]).

In a few number of related studies and industry reports, researchers focused their investigations on child-directed apps by evaluating compliance with privacy requirements that are specifically aimed to protect children [64, 65, 90, 189]. To inform improvements to the regulatory provisions of COPPA [61], the FTC [92] released two reports showing the results of technical and quantitative analyses of the behaviors of child-directed apps that were available on app markets in 2012 [64, 65]. These reports showed that many developers potentially violated COPPA [61] by transmitting various types of personal data without obtaining verifiable parental consent [64, 65]. Technical testing performed a few years later by Feal et al. [90] of popular Android parental control apps showed that privacy issues that allowed access to children's data were still present in apps published on the Google Play Store [111]. In 2018, Reyes et al. [189] conducted a large-scale dynamic testing of more than five thousand child-directed apps available to download on Google Play [111]. Their analysis of the network transmissions generated by these apps showed that data sharing was prevalent and was mostly enabled by third-party SDKs embedded in these apps [189]. Since then, Google Play [111] made major improvements to its policies in an effort to protect the privacy of child users. However, it is unclear whether these changes resulted in improvements in overall privacy behaviors of child-directed apps.

In Chapter 6, we build on Reyes et al.'s [189] work to understand the effects of these policies on developers' privacy practices. Our results demonstrate the powerful impact that mobile platforms could have on developers' privacy practices and provide recommendations that could help Google Play [111] further improve its enforcement efforts. Prior qualitative analyses of posts available in online forums found that enforcement of app market policies encouraged developers to seek advice on how to address privacy issues on these forums [144, 193, 204]. However, the investigation presented in Chapter 6 is the first to find technical evidence showing that platform policies can lead to improvements in the privacy behaviors of child-directed Android apps.

2.3 Organizational Processes

Many prior studies that examined privacy or security practices adopted by specific engineering roles recognized the role of organizational processes in shaping these practices (e.g., [29, 32, 141, 209]). While such processes typically define possible interactions between different roles according to specific organizational policies, the literature is yet to paint a comprehensive picture about how these contextual factors can explain software professionals' security and privacy practices. In Chapters 3, 4, and 5, we present the results of a series of qualitative studies that investigated how organizational processes can influence how security and privacy issues are handled by software professionals. Below, we review prior related literature and explain how we build on it in the studies presented in these chapters.

After surveying developers about their secure development practices, Assal and Chiasson [29] found that many struggled to follow security best practices that were not supported by their organizational processes (e.g., not having defined plans for security testing). Based on the results of a similar survey conducted with mobile app developers, Weir et al. [220] found that lack of interaction with security engineers throughout software development processes is one of the factors that can lead software developers to introduce security issues in their code. Palombo et al. [172] situated security testers in an organizational context who were able to observe how decisions related to remediating security issues were made by developers. They showed that such decisions might not necessarily be driven by individual factors (e.g., developers' lack of security expertise), but that social and organizational factors play equally important roles in shaping such decisions [172].

These types of organizational factors were also found to affect the work of professionals responsible for software maintenance. Researchers investigated the challenges faced by system administrators tasked with applying software patches and found that their work was influenced by the extent to which organizational processes were flexible and the quality of their interactions with upper management [141, 209]. The same types of challenges were found by Dietrich et al. [81] to influence security engineers' abilities to use security configurations correctly. While these studies found evidence of the role of organizational processes on shaping security practices adopted throughout the phases of the software development process, we are yet to understand the nature of interactions that take place between various stakeholders involved in security processes (e.g., security managers and engineers) and the reasons that can explain how certain security-related decisions are made at an organizational level.

In Chapters 3 and 4, we present the results of semi-structured interviews and surveys conducted with security professionals who had years of experience in organizational vulnerability discovery or remediation processes. In these investigations, we focused on obtaining *holistic* understanding of these processes, and aimed to uncover how the quality of interactions between different roles might negatively affect the work of security professionals involved in these processes. We also pay particular attention to understanding how security professionals take various strategic decisions related to vulnerability management (e.g., when to create a bug bounty program, how to rate vulnerability severity levels, and when to create an internal red team).

In Chapter 5, we extend our investigation to privacy compliance processes. Privacy engineering practices were shown to be affected by the quality of organizational processes [33], which include the flexibility of these processes and the extent to which privacy compliance requirements are clearly communicated to software developers and testers who are involved in these processes. While prior work found that privacy compliance issues were prevalent in apps released on the Google Play Store [111] (e.g., [189]), we are yet to obtain sufficient understanding of the factors that are leading app developers to neglect addressing these issues. Furthermore, a large percentage of the apps that are published on the Google Play Store [111] are released by individual developers or small organizations that are likely to deal with different types of challenges compared to well-resourced organizations. In Chapter 5, we present the results of a mixed-methods research study that utilized qualitative, quantitative, and technical data to obtain holistic process-oriented understanding of how developers of child-directed apps were adapting their processes to comply with the requirements of applicable privacy regulations. We shed light on how various privacy decisions were taken by these developers, including how they vetted third-party SDKs for inclusion in their apps, how they identified privacy compliance requirements that are relevant to their apps, and how privacy responsibilities were distributed among their internal teams.

2.4 Interventions to Help Software Professionals

Prior work proposed several types of interventions to help software professionals address security or privacy issues during their software development processes. Researchers experimented with introducing features in development tools and APIs that nudge developers to fix specific types of security or privacy issues early in their development processes. These include code injection vulnerabilities [224], not using encryption for communication security [163], using insecure symmetric encryption schemes [113], and triggering network transmissions that contain personal data [142, 145]. While this approach was shown to encourage secure development, its effectiveness is limited to coding mistakes, and might not help developers address issues that require them to have comprehensive understanding of the security and privacy behaviors of software systems under their control. This is especially the case for privacy issues that result from using third-party SDKs, whose usage might trigger issues that are not immediately visible to developers while coding (e.g., transmission of personal data) or might put the responsibility on developers to comprehend information included in other sources (e.g., third-party SDKs' privacy policies).

Since using code snippets available on developer forums was shown to lead to introducing security issues in prior work (e.g., [6]), Geierhaas et al. [101] designed a tool that generates customized and secure code snippets that developers can paste into their code. The results of their experiments are promising, as they demonstrated the feasibility of designing usable interventions for developers. The same design principle (customizing security advice to developers' needs) was similarly shown to be effective by Oltrogge et al. [170] who designed a tool aimed at helping developers with decisions related to use of certificate pinning in their apps. For privacy compliance purposes, researchers also proposed tools that generate disclosures that developers can include in privacy policies or privacy labels based on technical analyses of mobile apps [97, 228]. While these tools might make it easier for developers to prepare accurate disclosures, there are still other privacy compliance requirements that developers are assumed to reason about before publishing their apps, which are not supported in existing tools that provide developers with privacy guidance. These include decisions that require developers to think about how their use of third-party SDKs might lead to introducing privacy issues in their apps and how to ensure accuracy of disclosures included in privacy policies and verify that they are consistent with those included in privacy labels.

In Chapter 7, we take one step in this direction by experimenting with customized privacy checklists as a possible developer intervention. We developed a tool that generates privacy checklists summarizing privacy requirements that developers can take into consideration before releasing their apps. We additionally conducted observational interviews with software developers who published apps on the Google Play Store [111] to evaluate the feasibility of using this intervention in practice.

Chapter 3

Understanding Organizational Vulnerability Discovery and Management Processes

3.1 Introduction

Security vulnerabilities have caused substantial financial and reputational damage to organizations and users over the years and continue to do so at an alarming rate, despite the availability of advanced tools for hunting vulnerabilities down [31, 34]. Manual code inspection, black- and white-box vulnerability scanners, and penetration testing are only a few examples of techniques commonly used for security vulnerability detection before releasing software to users or after putting it into production [40, 86, 148, 174, 188, 208]. We have also witnessed the proliferation of bug bounty and vulnerability disclosure programs that encourage security researchers to assist organizations with vulnerability discovery. Additionally, regulatory bodies have released standards, such as ISO/IEC 29147 and ISO/IEC 30111, that guide organizations on how to design their vulnerability disclosure policies and what to consider after receiving vulnerability reports from external researchers [132, 133]. Despite all these tremendous efforts by the security and regulatory communities, the prevalence of cyber intrusions and leaks suggest that processes involving vulnerability management require urgent improvements.

In this chapter, we shed light on the current state of vulnerability detection and management processes by conducting 53 semi-structured interviews with managerial and technical security practitioners operating in the United States (US) and other countries. The aim of conducting the interviews was to contribute to the understanding of interactions, inter-group issues, and the challenges security teams face working with each other. We recruited security practitioners who were either responsible for security testing as part of internal or external

The work presented in this chapter appeared in the proceedings of the Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020) [18].

teams (e.g., red, blue, purple, and penetration testing teams) or managed the work of these teams at their organizations. Our goal was to reveal insights into how different roles fit into organizational security strategies and how human factors affect vulnerability management processes. For this purpose, we posed the following main research questions:

RQ 1 How do different security testing teams contribute to organizational vulnerability discovery processes?

RQ 2 What factors do decision makers take into consideration before they decide to create a bug bounty program, build an internal security team, or hire a third-party security testing firm?

RQ 3 What are the non-technical factors that impact modern vulnerability discovery processes at organizations?

We interviewed personnel with experience managing and implementing the security posture of their organizations. Our findings are informed by industry experiences of red, blue, purple, and penetration testing teams, working as security practitioners in internal security teams or as third-party contractors, who are involved in vulnerability discovery or remediation on a daily basis. We identified challenges managerial personnel face and the factors they consider when they make decisions on vulnerability discovery and management (e.g., creating bug bounty programs, outsourcing security testing tasks, etc). We also identified the challenges professionals from red, blue, purple, pen testing, and bug bounty teams face in practice and how management decisions affect their work. Our study attempts to piece together the perspectives of different actors involved in vulnerability discovery and management processes and identify challenges that require immediate attention by the security community.

We found that vulnerability discovery and remediation processes are hindered by a host of behavioral and organizational challenges. Our results also show:

- an inherent *trust* problem between organizations and testers or bug bounty hunters.
- a lack of effective communication channels between testers who discover vulnerabilities and those responsible for taking proper remediation actions.
- a lack of clarity around who is responsible for fixing discovered vulnerabilities, which turns vulnerability management into a “blame management” problem.
- a misalignment between security and business priorities, organizations represented by our participants often took reactive rather than proactive approaches to vulnerability management.
- a tendency to adopt *compliance-oriented* approaches to security rather than grounding vulnerability handling processes in an understanding of potential risks.

We believe that our findings can propel further research to better understand the human factors impacting vulnerability discovery and management processes in the wild.

3.2 Methodology

From April to June 2019, we conducted semi-structured interviews with 53 security professionals in different security testing roles (described in Section 3.3). We also recruited current and previous managers who had made important decisions on vulnerability discovery processes in their organizations. We used several channels for recruitment, including advertising on Twitter [211], and two Slack [195] workspaces (*bugbountyworld*, and *bugbountyforum*), which are used by testers and bug bounty hunters to share resources, collaborate, and communicate with bug bounty platform employees. We also sent emails to several pen testing companies, inviting their security testers and managers to participate in the study.

We conducted six pilot interviews to validate our interview process. Participants recruited for the pilot interviews were contacted via Twitter [211] or through professional connections. One pilot participant did not consent to use his interview data; hence, for the remainder of this chapter, we report on the data from 5 pilot interviews and 48 recruited participants. Our study was approved by our Institutional Review Board (IRB). As a token of appreciation, we offered a drawing for five $100 Amazon gift cards.

We asked participants to complete a pre-interview questionnaire, which asked about their security testing experience and their current and prior roles. Next, we conducted a 45-70 minute interview via online video chat software and recorded the audio for transcription purposes (see Appendix A.1). The lead researcher led every interview, and another researcher participated in an observatory capacity. We conducted interviews until the themes for each category reached saturation [57], which was independently determined by two researchers.

We tailored our semi-structured interviews to participants' roles and prior security testing experiences. The interview process for non-managerial participants evolved around:

- the discovery processes they follow,
- tools they use,
- how they communicate with the rest of the teams,
- challenges they face that might hinder their work,
- nature of the vulnerabilities they work on, and
- their expectations.

For those with managerial experience, the interview covered:

- their experience working with different teams,
- reasons for deploying specific strategies in their managerial capacity, and
- their expectations when deploying different security testing strategies.

Our sample included blue, red, and purple teamers, penetration testers working in internal security teams, security consultants, bug bounty hunters, and security managers (e.g., CTOs and CISOs). At the time of our interviews, a sizable portion of engineers and managers that we interviewed were employed by some of the world's largest tech companies, serving billions of end-users. Twenty of the participants had more than ten years of experience doing security testing, eight participants had between 6 to 10 years of experience, 21 participants had 1 to 5 years of experience, and 4 participants had less than one year of experience doing security testing. Many of our participants held multiple previous positions. For example, some had industry experience in red teaming and pen testing at the same time or held managerial positions after spending some time working as security engineers (see Appendix A.2). In particular:

- 38 worked as internal or external penetration testers;
- 28 had experience doing bug bounty hunting;
- 23 had held internal or external red teaming positions;
- 17 reported experience working in blue teams;
- 17 were CISOs, CTOs, or engineering managers;
- 11 had experience working as purple teamers; and
- 5 reported working as quality assurance engineers.

The majority of our participants came from the US, but we also had participants from Canada, Germany, UK, India, Israel, Singapore, Serbia, Brazil, and Bangladesh. Some participants managed their own companies, offering external pen testing services. We asked each participant about the number of employees in their organizations and their security teams. For around 40% of our participants, the number of employees in their organizations were in the thousands; eight worked in organizations with 100 to 5,000 employees, while the rest had 1 to 70 employees in their organizations.

Three external transcribers transcribed the interviews. Next, the lead researcher and two other researchers iteratively built the codebook. We used basic thematic coding, wherein each code mapped to an emerging theme (e.g., *blue-red-conflict*, *bugbounty-trust*). Our codebook has 32 major code categories based on the role of the speaker, and those categories are further divided into 88 sub-codes.

Once the codebook was finalized, the lead researcher and another researcher independently re-coded the interviews and merged the codes to calculate the inter-rater agreement. Disagreements in coding were discussed and resolved after reaching consensus among all the coders. Our recorded inter-rater agreement was 85.36% (Cohens' $\kappa = 0.71$), indicating substantial agreement. We arranged codes into three high-level categories that capture the three phases of what we called the vulnerability management pipeline: strategic decision making, vulnerability discovery, and vulnerability management (see Section 3.4).

3.3 Security Teams

There are a number of security roles that engineers might be tasked with in organizations. Many of our participants seemed to be confused as to how to define their work and how their role contributes to vulnerability discovery. For instance, a director of security who manages several security teams and is responsible for product security at a tech company that serves hundreds of millions of users responded:

"Yeah pen testing, red teaming, those are very confusing terms. What would you say is the meaning of pen testing?" (P52).

A CTO of a security consultancy offering vulnerability discovery services to organizations mentioned that many of his clients think that they should do red teaming or pen testing, when what they need is an incident-response team:

"I have had clients that have suffered a breach in the past and immediately they are like, 'Hey, we want to get a penetration test or a red team to figure out how the attacker broke in'" (P37).

We argue that not being able to distinguish between the roles different teams are expected to play might lead to making sub-optimal decisions and ultimately rendering vulnerability discovery processes ineffective. Below, we present a brief definition for each of the different security testing strategies based on themes found in participant responses and our review of the related literature (e.g., [124]).

Internal Testing. We define this role as part of a quality assurance process in which people with security backgrounds either do testing or work with developers to build secure code:

"We have kind of check points along the development process. We also have kind of programmatic deployment gates that a handful of these tests must be met before things can even be deployed" (P10, security manager tasked with managing internal testing teams at a large organization).

Pen Testing. The goal of penetration testing is to find as many vulnerabilities as possible, focusing on a finished product or a service within a pre-defined period:

"I see penetration testing as audit. So it is assessing for all vulnerabilities, at every single point in time, within a defined scope" (P40, a full-time bug bounty hunter with extensive experience as a pen tester and a red teamer).

Red Teaming. Red teaming is goal-oriented and is about going deep (compared to breadth coverage in pen testing); that is, infiltrating the network in question and exploiting any vulnerabilities in the process to reach a pre-defined goal. It is commonly understood that red teaming emulates real-world hackers in the most realistic way possible:

"In a red team exercise, you are working with no knowledge of the product. It is as if you are an external hacker kind of thing, trying to hack without access or insider knowledge" (P31, security engineer tasked with doing internal red and blue team exercises within a large public organization).

"The red teaming is really more, hey, we are not interested in identifying every stage of vulnerability in the network, we are interested in seeing how far can we get, what type of data can we access, can we actually exploit the things that are most important to the organization" (P30, a security manager who has extensive experience leading red teams).

Blue Teaming. Blue teaming is about being defensive—in contrast to red teaming—and concerns monitoring target systems for abnormal behaviors that might indicate the presence of an adversary in the system or an intrusion attempt. Blue teamers are expected to collaborate with red teamers in order to make sure that their efforts complement each other and that they are improving their detection over time:

"[A] blue teamer is exactly the opposite, is the defending side. This means finding solutions, techniques, procedures to stop hackers, stop red teamers from succeeding in their attacks" (P39, an experienced blue teamer).

Purple Teaming. Purple teams might consist of members from red teams and blue teams. The most significant advantage of a purple team is to have both offensive and defensive teams work together to achieve a common goal:

"The purple team is the one that navigates between the red team and the blue team and sits with them together as a sort of mediator" (P39, an experienced blue teamer).

Bug Bounty Programs. Bug bounty programs can be seen as ways to crowdsource pen testing on publicly-facing systems without any time restrictions. Bug bounty programs invite the public or a group of hackers selected by an organization to test its systems according to a pre-defined scope and policy. HackerOne [114] and BugCrowd [43] are examples of bug bounty platforms that act as intermediaries between organizations and external security testers.

3.4 The Vulnerability Management Pipeline

Based on the themes that emerged from the interviews, we identified three phases of an organization's vulnerability management pipeline: from the moment the organization's decision makers realize that they need security testing ("strategic decision making"), to the methods chosen to perform that testing ("vulnerability discovery"), to when discovered vulnerabilities are fixed ("vulnerability remediation").

Strategic Decision Making. We wanted to understand what decision makers expect from the various security testing strategies and what drives them to deploy a particular one. Our goal was to understand whether management are making correct decisions that serve their organizations' goals, whether those organizations are ready for a chosen testing strategy, and whether there are any cultural and/or organizational barriers.

Vulnerability Discovery. In this phase, the chosen security testing teams begin searching for vulnerabilities in target systems. Different security testing teams have different goals, expectations, and time restrictions. Further, some teams could be better suited for specific categories of companies. We believe that it is important to understand the activities carried out by each team and the expectations for each role.

Vulnerability Remediation. This phase concerns deciding on how to fix the uncovered vulnerabilities in a timely manner to mitigate potential risks. An effective vulnerability management process entails having a systematic approach to prioritizing discovered vulnerabilities according to their potential impact and having clear communication lines between security testing teams and any other stakeholders.

3.5 Strategic Decision Making

The potential success of a vulnerability management pipeline hinges upon utilizing the correct vulnerability discovery strategy at the correct time. In this section, we present findings on participants' stated motives and other factors that go into their decision processes. We focus on understanding:

- when to hire a red, blue, purple, or pen testing team,
- when to create a bug bounty program, and
- what considerations are being made before deciding to outsource security testing.

Deciding Between Different Teams

Below, we present the factors that influence the managerial decision of which security testing strategy to implement.

Red Teaming vs. Pen Testing

We observed uncertainty among our participants about when to do red teaming versus pen testing. The majority of our participants who are considered decision makers did not see the need to do red teaming when their organizations have regular pen testing engagements. Participants expressed different motives for pen testing, a commonly recurring one was to meet various compliance or certification requirements:

"Pen tests are kind of a reproducible formula for getting the engagement done whether it is to get that compliance checkbox checked or they are just kind of going through the motions to make sure that basic security structures are in place"(P10, security manager tasked with vulnerability management operations at a large organization).

Red teaming involves going in depth and emulating a malicious actor. Participants mentioned that another way to look at it is as testing all of the monitoring capabilities installed by the blue team. Thus, to have a fruitful red teaming engagement, an organization should have a reasonably mature security posture with in-depth monitoring:

> *"If my goal is not to find as many vulnerabilities as I can, my goal is to validate my security controls that I have put in place, then I would love to hire a red team"* (P41, a former red teamer and a pen tester who is currently managing application security programs at a large organization).

Regardless of the engagement exercise, participants commonly agreed that neither red teaming nor pen testing should be the first approach to testing in an organization or the first step in their vulnerability management pipeline:

> *"I think most organizations are not mature enough to have true red team exercises performed. I don't think having a red team or having a penetration testing team should come before having an effective vulnerability management program"* (P41, a former red teamer and a pen tester who is currently managing application security programs at a large organization).

We observed that there is a common preference for doing pen tests before putting software products into production and whenever major changes to codebases are made. Red team engagements come at a later stage, when organizations have already tested their products internally and set up monitoring and detection, performed the required pen tests, and are willing to validate the security controls they have in place. However, our results suggest that concerns around exposure to legal liability might drive reluctance around the use of red teams by organizations. Some participants expressed that it would be risky to let external red teaming contractors test their assets without being able to have full visibility into their activities, as they had concerns that their sensitive data might be leaked as a result. This might also create a potential conflict between duty-to-report rules for red teamers and the actual practices of companies. A former red teamer mentioned:

> *"You don't know when someone is in the network, you don't know what he is reading. I know that there are NDAs, but sometimes there is a code of ethic, that says if you find something illegal during a penetration test, you have to report it. And in red teaming, let's be honest not many companies are in regulations with the law"* (P24).

Blue and Purple Teaming

Our participants agreed that organizations should have a blue team that is continuously ready to detect intrusions and respond accordingly. A security director responsible for vulnerability management operations at a leading tech company mentioned:

"The first thing that you need to hire is the blue team. What are we defending? What are the state of our systems? What is our attack surface? That stuff you have to first understand" (P52).

Though, the majority of the participants mentioned that, in most cases, blue teams are not set up until a breach incident happens, which causes upper management to realize the severe consequences of cyber-attacks and the importance of proper security posture. Once an organization properly implements the necessary security controls, red teaming can be utilized to evaluate the extent to which such controls can be exploited by the adversary and help in strengthening blue teams over time. Purple teaming, on the other hand, could be beneficial once a blue team is perceived as not maturing and learning from the results of prior red teaming engagements. However, considering that it would not be realistic for small or medium size organizations to have red, blue, and purple teams, a considerable number of our participants mentioned that they would prefer to start by setting up a purple team, where its members have red team and blue team backgrounds.

Bug Bounty Programs

We found different motives for creating a bug bounty program. Many participants mentioned positive experiences; having many eyes looking into their systems and a large pool of testers with diverse backgrounds can yield interesting vulnerability reports. A security director who started a bug bounty program at a large company, and slowly expanded their scope to cover their vendors as part of their security boundary, pointed out the high costs of pen testing services as the primary motivator for their decision:

"One pen tester for one week is going to cost you a lot of money and you can get much higher coverage with just that money and many more people involved" (P52).

A few participants also had reasons for not creating bug bounty programs. Particularly for financial and governmental organizations, a common theme that emerged in our interviews was that they do not want to risk having unvetted *"people from the Internet"* accidentally gain unauthorized access to sensitive data stored in their systems. They mentioned that they are more comfortable working with established external security firms, where they can keep tabs on people who are allowed to test their systems for security vulnerabilities:

"...trying to convince any government agency to try to offer people a reward to try to break into their financial data, I think we would have a very bad reaction to that, whereas saying, 'it's an audit' would be a much easier sell" (P44, security leader of a consultancy company that works as a government contractor).

Some participants mentioned that it is often difficult to justify the costs of creating a bug bounty program to upper management, as the costs associated with running such programs are not fixed, relative to paying for an outside audit. Some participants also mentioned that they hire external contractors to conduct pen tests to satisfy regulatory requirements and to convince their customers that they are sufficiently secure because they have been tested by an outside entity:

> *"..the quality of that report you get from pen testing companies is much much higher. It can also hold people who did the pen test accountable"* (P32, a security engineer tasked with red teaming and pen testing at a large security consultancy company).

Participants who had created bug bounty programs mentioned that before creating a bug bounty program, organizations should make sure that they have a robust process for handling reports received from bug hunters: assessing them for validity, reproducibility, severity, and impact, and then responding to the bug hunter promptly. Participants said that starting a bug bounty program could easily overwhelm the organization with a lot of noise (false reports). There were also concerns as to whether organizations, especially management, clearly understood the prerequisites:

> *"My CISO likes to be able to tell customers we have a bug bounty program; therefore we are very secure"* (P41, application security manager).

Due to trust-related concerns, several participants preferred to create private bug bounty programs, where the organization gets to pick the bug hunters who are allowed to work on that program after doing proper background checks. One participant also mentioned that his organization decided to do all their bug bounty work on their staging environment, where they do not risk bug hunters getting access to their customers' data, and can make sure that they can reset the testing environment whenever anything goes wrong. Other organizations might prefer to narrow the scope of their public bug bounty program to include a few of their assets and expand the program scope incrementally as they go forward.

Factors affecting decisions

Below, we summarize other factors that might potentially influence decisions on vulnerability management.

Internal Teams vs. External Vendors

We observed common concerns related to the costs and difficulty of building and maintaining internal teams, which were justifications for outsourcing security testing. The majority of our participants agreed that having internal teams is more beneficial in the long term, as testers accumulate institutional knowledge about their organizations' systems and build understanding of their business logic in more depth over time, as well as have conversations with developers and build relationships with other security team members. Having internal teams also helps organizations adopt a proactive approach to vulnerability discovery. An engineer with more than 10 years of experience in red teaming, pen testing, and managing security teams said:

"The internal folks come more or less over time, with a built-in understanding of the business objectives, and the business logic that needs to be expressed. The external folks by definition, don't understand the business as well. And so sometimes their findings are not going to be as relevant to the company" (P17).

"A lot of folks struggle with external bug bounties because folks outside the company will report something and there's no clear implication to the business. And you cannot automatically say this is not a problem. But then you have to decide am I going to spend time even figuring out if this is a problem, versus someone who comes to the table with that work already done for you, then you can immediately start remediating" (P17).

However, participants also mentioned that external testers can help discover vulnerabilities that were previously not noticed by developers or internal testers, since they often have a lot of exposure to different troubleshooting scenarios and clients. Furthermore, many participants highlighted that most security engagements with external testers are time-scoped and hence bear less cost and logistical overhead compared to maintaining internal teams. This was especially the case for small organizations and is the primary reason why organizations may hire external vendors rather than build and maintain internal teams. Said an experienced red teamer who now works as a consultant in blue and red teaming:

"So if I am a company the size of Walmart, I am going to have my own red team. If I'm a small organization, and I don't have the money and IT isn't my focus in the first place, I am not going to try to build a red team and then I would definitely contract with a third party to do it" (P30).

Furthermore, due to trust-related reasons, participants also mentioned that external testers normally do not get to access every resource they need in order to cover significant parts of their clients' attack surfaces.

Selecting and Vetting External Testing Firms

Participants expressed a tendency to rehire the same external firms:

"We, our blue team, prefers a particular pen testing team. We have good relationships with them, their diagnostics and write ups have been just superb" (P9).

Other explanations included:

- they had a positive experience with a particular firm and built relationships and trust with its members;
- they were satisfied with the level of diagnostic investigation and the quality of write-ups they received;
- they did not want to waste a lot of time vetting different external firms;

- they preferred to work with firms that have a good reputation in the industry; or
- they perceived difficulty in obtaining visibility into how they are improving their security over time once they switch vendors, as different vendors have different methodologies and approaches to security testing.

A few participants, however, mentioned that they prefer to rotate through a pool of vendors to:

- fulfill regulatory requirements;
- get specific testing expertise (e.g., testing cryptographic solutions); and
- get different sets of eyes to test the same codebase.

Regarding the latter, an experienced pen tester stated:

"I think it is good to switch from time to time because other people have other techniques and will most likely find other or new issues" (P15).

Staffing and Budget

Participants mentioned the difficulty of justifying security-related staffing or budget decisions. Participants with management responsibilities admitted that staffing is a big problem, and it is not a luxury that many can afford. In response to why they do not have a bug bounty program, the CISO of a large organization responded:

"We already know about more risks than we have the capacity to deal with" (P1).

This might lead to a lack of visibility into unpatched vulnerabilities and a false sense of security until an incident occurs. Participants also expressed the difficulty of convincing upper management to provide security engineers with the required freedom to do their assessments and find vulnerabilities. Others mentioned that there had been situations where their managers decided to build a security team very quickly to react to security incidents without proper planning and forethought:

"...out of the news, all of our [security] costs are basically viewed as not required and extravagant. And it is always a fight to get funding, even though when there is a problem suddenly funding is free and how many people do you need" (P9).

Participants mentioned that, in some organizations, monetary allocations might give upper management false assurances about the organization's security posture. In cases where an organization has an annual budget allocated for security, upper management might be under the impression that they are *secure* when a regular security assessment with an external contractor is only done as a formality to *'tick the boxes'* and have a report that says so. Said a pen tester:

"The CEO does not want to know if you have SQL injection, or XSS. He just wants to see in the report, that we have spent that amount of money on the security, and he always thinks like, 'oh we have spent a lot of money on security. We must be secure!'" (P24).

Many mentioned that they had to frame everything around a dollar value to get attention. That is, some security managers expressed the importance of conveying risks to the business and describing the impact of potential legal liabilities or monetary losses to upper management:

"When you say that this vulnerability has a CVSS score seven, they don't get it. When you can say, like, this risk represents an expected loss of a hundred fifty million dollars, and they're like, 'okay we know what to do with that'" (P17, security engineer).

Other managers also expressed difficulty in building security teams that have combinations of qualified people with different areas of security knowledge; e.g., network security, mobile security, and cryptography. Some security managers also mentioned the difficulty of making sure that discovered vulnerabilities are addressed promptly by their qualified security engineers, as they are worried they might lose their personnel once they are not satisfied with their workload.

Considering the costs of setting up teams of qualified personnel, several different approaches to managing security in small organizations emerged in our interviews. Outsourcing security to external contractors, using open source security testing tools to automate some tasks, leveraging the security teams of well-known cloud providers by putting most of their security services in the cloud, creating private (invite-only) bug bounty programs, and hiring a few security engineers with purple teaming experience are the main approaches that the participants brought to our attention. Particularly for bug bounty programs, decision-makers working in small organizations were concerned that qualified security researchers would not be attracted to work on their programs, given that they would not be able to compete with organizations that have mature bug bounty programs and provide high payouts. This observation was confirmed by several bug bounty hunters:

"I have got two or three programs that I have spent a lot of time on and I keep going back to them because I enjoy working with the team, I know that they pay fairly well and they will turn around my bugs pretty quickly" (P40, bug bounty hunter).

Some participants mentioned that they have a small team of security engineers wearing multiple hats (i.e., doing offensive, defensive, and quality assurance work at the same time).

Scoping Considerations

Our discussions with internal testers, security managers, bug bounty hunters, and external contractors revealed that there is often uncertainty in how to scope security testing. When defining the scope of a bug bounty program, many decision makers preferred to focus on detecting vulnerabilities that can be reached through publicly available services. Some bug bounty hunters also mentioned that they have accidentally stumbled upon critical vulnerabilities that are considered out of scope, and decided against reporting them, because they fear that the organization might take legal action against them. This fear was confirmed by some bug bounty program managers, who expressed that they would not welcome such discoveries, as they do not want to incentivize bounty hunters to go out of scope.

Other program managers expressed that they would welcome such submissions, as it signals that the bug bounty hunter cares about helping them improve their security posture. Other factors that might influence program managers' decisions include the budget for paying bounty hunters and the capacity of their internal triaging team to process incoming reports. A broader scope is likely to incur higher costs, just in terms of coping with more reports.

In contrast to bug bounty programs, scoping red teaming and pen testing engagements is mostly focused on organizations' internal assets. Although no-scope red teaming engagements are perceived to be common for large organizations, some participants mentioned that narrowing the scope of red teaming could be useful in cases where an organization wants to validate certain controls that they recently implemented. We also observed that some organizations prefer to focus on a specific area at a time; e.g., for the first couple of months they will focus on a specific set of systems and then test other systems later. For pen testing, the majority of our participants indicated that no-scope assessments are often expensive, and this might probably lead them to narrow the scope of systems to be tested in order to afford their costs.

Other security managers mentioned that they are not confident with letting external researchers test every system they have, leading to narrow scopes for pen testing or red teaming engagements. For the same reasons, some organizations resorted to creating duplicate environments, where they put all their code in a live environment, but with sensitive data removed to mitigate the impact of any potential data leakage. From the perspectives of pen testers, we noted that narrowing the scopes of pen testing or red teaming assessments could make the vulnerability discovery process very restrictive and completely counter-intuitive to what the vulnerability discovery process should achieve. Said a red teamer:

> *"...but like it gets to the point where they chop everything off to the point that you don't have a bite anymore and, you know, you want to try as hard as you can, just like any bad adversary"* (P42).

3.6 Vulnerability Discovery

Vulnerability discovery can start from the earliest phases of the software development lifecycle. It can be part of each cycle or can be brought in after each major code release. An organization can have a dedicated security team testing all of the ongoing projects or assign a team member(s) to work with each development team(s). Below, we outline the processes security teams follow to discover security vulnerabilities based on a synthesis of common understandings in the industry and our observations from the interviews.

Pen Testing Activities

Pen testers aim for breadth by finding as many vulnerabilities as possible within a predefined scope and time frame. Participants also mentioned that they often get the help of external pen testing contractors to test their products before shipping them to users. Pen testers are not supposed to hide their activities from other members of the organization and are mainly expected to demonstrate the vulnerabilities they discover to prove that they exist, without exploiting them. However, one external pen tester mentioned that clients sometimes ask them to exploit identified vulnerabilities to demonstrate their impact. Furthermore, most pen testers' activities are supposed to be continuously monitored by members of the client organization to limit the consequences of any potential unintended exploitation.

Red Teaming Activities

One of the main goals of red teaming is to help blue teamers improve the defenses they have put in place. Pen testing and red teaming bring different benefits to an organization. Red teaming engagements normally take longer than pen testing engagements. Pen testers are usually given access to internal resources, whereas red teamers start their engagement with minimal to zero information. For instance, some participants mentioned that pen testers usually have internal network access, whereas for red teamers, gaining access to internal networks is a goal they should achieve. Furthermore, pen testing engagements are not stealthy, and many in an organization might be aware of them, whereas red teaming engagements are stealthy by nature to evade any monitoring or access control mechanisms placed in the organization and to effectively simulate the activities of malicious attackers. For instance, some of the tactics red teamers follow to decrease the risk of detection are:

- not installing custom software;
- trying to use system functionality as much as possible, rather than using exploits;
- sticking to regular user working hours, so a blue team would not suddenly start noticing loads of traffic;
- rotating their IP address or hiding them behind different VPN providers; and
- not scanning many ports at once.

Participants explained that there are basic steps that are usually followed by red teamers. At a very high level, the first phase is reconnaissance, which involves gathering as much public information about the target as possible. Based on that, red teamers can draw a plan outlining the potential targets that they can attempt to attack or utilize to exfiltrate data. For instance, they might send phishing emails in the hope that someone will click on a malicious attachment, which will then help them establish a foothold into the network.

They could also examine whether lateral movement is possible and try to plant a backdoor in order to maintain persistence. To effectively simulate real-world adversaries, red teamers usually ask organizations not to inform their employees—including blue teams—that these offensive security engagements are taking place; once blue teamers detect them or they successfully evade a blue team's security controls, they communicate their findings and provide guidance on how to fix the identified vulnerabilities. Red teams need to have broader skill sets, including social engineering, physical security, and network security. Participants also mentioned that pen testing experience would be beneficial for red teaming.

Blue Teaming Activities

Blue teaming is concerned with fixing vulnerabilities found in the discovery phase or implementing proper monitoring capabilities to prevent intrusions. Some examples of blue teaming activities that were brought up in our interviews are installing firewalls and antivirus software, monitoring systems for suspicious activities, responding to cyber incidents, trying to confuse potential adversaries once they are detected in order to gain an improved understanding of their activities and capabilities, and auditing and analyzing logs. Some blue teamers also mentioned that most of their work is concerned with fighting bad security culture, convincing people to install patches, and teaching engineers the best security practices. It is also worth noting that, in contrast to red teams, blue teams usually have access to significant internal resources and are therefore expected to know the specifics of what they are defending. For this reason, asset management is perceived as a challenging task for internal blue teams. That is, they cannot protect systems that they do not know about.

Purple Teaming Activities

We observed common confusion surrounding what purple teams do in practice; some thought the term signals that an engineer has red and blue team experience, while others thought that applying offensive and defensive security practices at the same time can significantly contribute to hardening defenses in organizations. While many of our participants mentioned the difficulty of building a purple team, we noted that small organizations prefer to start with setting up purple teams without having separate blue and red teams due to decreased staffing costs.

In organizations with red and blue teams, purple teams can act as middlemen who facilitate communication between the two teams by making sure that they have regular meetings and regularly exchange information. Other organizations prefer to assign some members of the red team to work closely with blue teamers, to train them to detect the attacks mounted by red teamers or reflect on the regular feedback they get from red teamers. Another way of doing purple teaming exercises is to have blue and red teams work very closely, where all team members are involved in the same exercises. In such settings, once a red teamer manages to break defenses, the red teamer works with blue teamers to address the identified vulnerability. Afterwards, the two teams can resume their work and reflect on their progress.

Bug Bounty Hunters' Activities

Bug bounty hunters consider reconnaissance a critical phase in their vulnerability discovery processes, in which they attempt to collect as much information as possible about the target systems. This phase includes activities such as enumerating all sub-domains, using a service as a regular user to understand its functionality, parsing public datasets, understanding data flow, and identifying dependencies between different features or services. Some examples of the techniques used for vulnerability discovery that we identified in our interviews are disabling certificate pinning, reverse engineering apps, looking for default credentials, discovering old services, and watching for DNS changes.

Inter- and Cross-Team Communication

One of the main themes that emerged in the majority of our interviews with security engineers concerns the communication challenges security teams face with other internal teams. From the perspectives of red teamers and pen testers, they mentioned that their findings are often not welcomed by blue and development teams and that they are generally perceived as threats to other teams; red teamers' findings might make development/blue teams look bad to upper management or potentially increase their workload. Our results therefore suggest that establishing cooperative and collaborative relationships between blue and red teams seems to be a serious challenge.

In most cases, this might turn into a *"blame-management"* problem, in that each team thinks that fixing the discovered vulnerabilities is not part of their job. Several participants mentioned that they had had situations where they decided to report their discoveries to upper management to ensure that the identified vulnerabilities got fixed. From the perspectives of external contractors, they mentioned that they often report their findings to the person who hired them and might not get a chance to interact directly with internal teams who are in charge of fixing discovered vulnerabilities. Participants also noted that red teams are supposed to not disclose their activities or share much information about their testing strategies to make their simulations realistic, and this can be frustrating for blue teamers. From a managerial perspective, how security teams are organized and structured could introduce some communication problems. Said a security manager:

> *"..a lot of organizations diversify teams too quickly so where they have five or six people on a dozen different islands with different names. And they lose a lot of the collaborative potential of the team whenever they're segmented off that way"* (P10).

To address these communication problems, participants mentioned some strategies followed by their organizations:

- situating security teams within engineering teams;
- letting different security engineers test the same product at different points in time over the year and comparing their findings to keep track of what vulnerabilities have been fixed and what still needs to be done;
- setting up a purple team; and
- holding regular security-related meetings with representatives from each team in the organization.

Some also raised the point that having clear objectives and increasing the level of transparency with security engineers could make vulnerability discovery processes more fruitful.

From the perspective of bug bounty hunters, participants have had a mix of positive and negative experiences when reporting their findings to internal teams. Many of the bounty hunters we interviewed mentioned that the bug bounty ecosystem has allowed them to build productive relationships with internal security teams at various companies, allowing them to collaborate on fixing reported vulnerabilities and receiving prompt feedback. Some mentioned that their work as bug bounty hunters increased their chances of getting hired at the companies to which they reported vulnerabilities. Several participants also mentioned that they had situations where internal teams ignored their vulnerability reports or considered the reports as out of scope because the internal teams did not triage the reports correctly.

3.7 Vulnerability Remediation

Below, we summarize our observations that relate to fixing the vulnerabilities uncovered during the discovery phase.

Triaging Vulnerability Reports

Internal teams have to triage vulnerability reports they receive in order to assess the severity level of each reported vulnerability and prioritize fixing the critical ones. Participants described their experiences with bug bounty programs that assigned non-technical people to the task of triaging vulnerability reports, which resulted in closing their reports as insignificant when the triaging team did not fully understand the impact of the findings.

Fixing Vulnerabilities

With regard to addressing vulnerabilities, one main theme emerged: internal teams' inabilities to fix reported vulnerabilities due to the lack of detailed information in the reports that allow reproducing the vulnerabilities and thus make informed decisions on how to fix them. For example, bug bounty hunters mentioned that patches applied by internal security teams could often be bypassed. For this reason, some security managers mentioned that they sometimes offer extra bounties or reputation points for bounty hunters willing to help with remediation. However, participants mentioned that providing extra incentives is not common and that the current bug bounty ecosystem does not incentivize external researchers to go beyond surface-level checks to ensure that reported vulnerabilities have been properly fixed:

"There is one program that I've worked on that specifically says you will receive a high bounty if you provide a remediation advice. So it is not something that I see across the board in having recommended fixes, I tend to think that the teams themselves are in a much better place to know how to fix it, that I'm here to explain the issue in a way that they can understand it" (P40, full-time bug bounty hunter).

Others thought it would be helpful to ask bug bounty hunters to retest the fixes released by companies:

"I think it's good to send it back to you after they fix it and ask you to check it because they might have put a patch that wasn't fully secure. So I think that is a great way to just get confirmation that it is fixed" (P25, bug bounty hunter).

A participant with extensive pen testing experience stated that external security testers usually lack the specifics needed to describe how to fix identified vulnerabilities. On the flip side, from internal teams' perspectives, some participants stated that they often do not have a clear idea of how a vulnerability reported by an external tester was discovered in the first place, which is one of the reasons why they might have no way to validate the fix. Other participants mentioned that they arrange for external pen testers to meet with internal teams to discuss possible remediation strategies. External pen testers might also be asked by the organization to do another pen test after a certain period to check whether the identified vulnerabilities still exist.

From security managers' perspectives, many stressed the importance of providing comprehensive and detailed mitigation strategies by security testers to allow internal teams to find an alternative approach to fixing a vulnerability, once they find it infeasible to fix it in a particular way.

When discussing the extent to which internal teams are capable of addressing vulnerabilities promptly, an experienced blue teamer explained:

" a lot of them don't know how to test whether or not the patch solved the problem. I think that the biggest reason is skill" (P9).

Other barriers hinder the remediation process. For example, internal teams might identify dependencies between different applications that might involve other internal departments or external vendors, or require hiring additional software developers with special expertise. This might decrease the likelihood of applying fixes promptly. Some security managers also mentioned that lack of transparency between all the internal and external parties involved in security processes is often an obstacle towards effective vulnerability remediation:

"We have trained our engineers to not come to us, but we have lost visibility of everything else that is happening, and now we are, back to a super reactive mode, where we want to be in a super proactive mode" (P43).

Compliance vs. Security

Depending on the industry, there are specific compliance requirements that must be met in order to make sure that the organization is doing the bare minimum to secure their users or customers. Such requirements have made organizations to adopt compliance-oriented rather than security-oriented mindsets. Most of the participants who have experience working as external pen testers described their experiences with organizations that ask them to downplay the severity rating of a critical vulnerability in the reports they plan to submit to regulators or to take some vulnerabilities out of a report when they do not have enough time to fix them.

This is made worse when pen testers are pressured by their managers to comply with these client requests to maintain relationships. These external pen tests are sometimes perceived as a way to shift the liability and accountability for security breaches from the client organization to the pen testing contractor. Said two pen testers:

"We've been told we don't want you to actually solve this problem, we just want you to make the check box go away" (P9).

"I think that turns pen tests into commodity and not clients that really want to understand the exposure and ultimately not necessarily interested in actually fixing things and getting a better security" (P38).

3.8 Discussion

In this section, we analyze the challenges that came up in the interviews, which we believe can be addressed by paying more attention to human factors.

Security Is a Reactive Measure

Security testers and engineers mentioned that convincing management that security should be a priority is hard in many organizations. Organizations that struggle to coordinate to place security as a high priority often spend more time reacting to incoming vulnerabilities than proactively searching for vulnerabilities. The reactive approach to vulnerability discovery and remediation may increase the chances of organizations falling victims to security attacks despite advances in vulnerability detection tools and techniques. Staffing and budgeting costs could also push organizations to be conservative in their investments in security, particularly in small organizations.

Insufficient Attention to Vulnerability Remediation

Participants mentioned that management might invest in security for reasons other than securing their systems (e.g., fulfilling compliance requirements). Although the byproduct of these intentions might still lead organizations to invest in vulnerability discovery, the downstream effects of finding a vulnerability may not be effective for fixing the root cause of the vulnerability. Compliance regulations often require organizations to establish and follow their own processes to handle vulnerabilities, with compliance often checked by periodic audits (e.g., every 6 months). Since compliance is checked at discrete intervals instead of continuously, organizations that invest in security for compliance purposes may only address discovered vulnerabilities once an audit is scheduled, so that they may develop a corpus of events for the purpose of passing the audit. This is likely to render vulnerability discovery efforts ineffective since discovered vulnerabilities might not be remediated correctly or promptly.

Trust

When working with external testers, one of the most significant challenges faced by organizations and internal testers is trusting external testers with their systems and data. This can discourage organizations from creating public bug bounty programs or working with external red teaming firms. Trust can also be a factor in deciding what components are in-scope for testing engagements, and this might increase the chances of leaving some vulnerabilities undetected. Our results suggest that many organizations tend to put rules in place that limit what red teamers can realistically do to simulate the adversary. Said the director of a pen testing firm:

> *"..everybody is talking about they want a red team, they want a red team, but at the end of the day, they want to put a bunch of rules around it, just like regular penetration tests"* (P38).

For the same reason, many bug bounty hunters expressed that they often feel hesitant to report vulnerabilities they discovered to responsible entities, as they fear reprisal from organizations. This is likely to discourage external security researchers from reporting their discoveries or collaborating with organizations to improve their security posture.

Communication

Lack of communication had caused issues with the reproducibility of reported vulnerabilities when the person who reported a vulnerability does not provide sufficient detail (e.g., the feasibility of exploitation and whether the reports contain realistic exploits). That is, they lack the knowledge required to put the vulnerabilities in the context of the organization to allow assessing the severity and the impact of reported vulnerabilities correctly. This is a clear indication that the industry lacks a proper standard for communicating discovered vulnerabilities to relevant entities. One standard that does exist for assessing severity levels is the Common Vulnerability Scoring System (CVSS). However, participants stated it is not helpful because it lacks sufficient consideration for business requirements or organizational contexts, which can facilitate communication with testers, internal teams, and management as to whether a reported vulnerability is critical or not. Without an effective vulnerability rating mechanism, vulnerabilities with high impact can go unnoticed or issues that require proper attention might not get it. Another important line of communication is between the security engineers and the management who decide on the budget and staffing. Participants mentioned that they had to put a monetary value on security issues that require immediate attention to garner attention from management. Furthermore, one CISO mentioned that finding a responsible point of contact for each independent system in the event of a breach or incident is especially challenging in large organizations.

Creating a security culture that is driven by a genuine interest in defending technical infrastructure can foster collaboration between developers and testers around resolving security issues. Surprisingly, many of the internal and external testers we interviewed raised the point that they often experience problems concerning receptiveness to their security feedback. One participant mentioned that developers might feel embarrassed or blamed once someone reports a vulnerability in their code. Such attitudes to security are likely to discourage developers from acknowledging the receipt of vulnerability reports and security teams from following up on whether a reported vulnerability has been fixed.

3.9 Recommendations

In light of our findings, we propose a set of recommendations that we expect to help organizations improve their vulnerability management processes.

First, it is essential to identify the technical and business owners of the various systems an organization has early in the process, so that remediation tasks can be assigned smoothly to their corresponding owners, who can take the lead in deciding how to plan subsequent remediation efforts. We would expect there to be guidelines to help identify which stakeholder to notify to improve the level of transparency between the teams involved and reduce the time from discovery to remediation.

Second, organizations should establish a clear set of risk priorities for upper management that communicates the risks an organization is willing to take, in order to guide decisions concerning vulnerability discovery and remediation. This would also allow security teams to communicate the relevance of discovered security vulnerabilities to the organization's priorities and business goals and therefore facilitate discussions that involve convincing upper management to allocate more resources for vulnerability discovery or remediation. Organizations should also have clear procedures for reporting potential risks to upper management and facilitating decisions that concern resource allocation, coordination among teams and stakeholders, and separation of responsibilities between security, development, and legal teams.

Third, we recommend having clear expectations of the maximum period of time that should elapse from the moment a vulnerability is discovered until it is completely remediated by its technical owners. Such expectations should be based on pre-defined criteria for assessing the severity and potential risks of reported vulnerabilities and assigning a timeline for implementing remediation plans that allow addressing critical vulnerabilities in a timely fashion.

Fourth, we recommend that organizations do not rush to create bounty programs, unless they have solid remediation processes in place. One of the prerequisite to creating a bug bounty program is performing internal security assessments (e.g., pen tests) and remediating all discovered vulnerabilities. A proper assessment of the costs of creating such programs, triaging vulnerability reports, assigning discovered vulnerabilities to their technical and business owners, and carrying out subsequent remediation efforts should also be performed before creating these programs.

Fifth, we recommend improving the current incentive structure implemented in bug bounty programs by providing higher bounties for researchers who include suggestions on how to address reported vulnerabilities or help with testing released fixes. We also recommend involving bounty hunters in vulnerability remediation and encouraging internal testers to get in touch with them to explore the available remediation options and get confirmation that a released patch fully remediated the discovered vulnerability. It is also important to define what is in-scope or out-of-scope for bug bounty programs based on a pre-defined threat model, which considers the organization's security objectives and the risks that it aims to mitigate.

Sixth, we recommend customizing the language used in communications between different security teams or other stakeholders to the language that can be clearly understood by the target audience. Communications with upper management should be framed in terms of how a discovered vulnerability might introduce risks to the business to help them make informed decisions. On the other hand, blue teams and development teams might need a mix of technical and business-oriented discussions to guide their decisions on how to fix reported vulnerabilities and how to prioritize addressing them based on their potential impact to the business.

3.10 Conclusion

In this chapter, we presented the results of an investigation of modern vulnerability management processes that focused on exploring the tensions between different security engineering and management roles in small, medium, and large organizations operating in the public and private sectors. We show that while the technical aspects of computer security are imperative for securing organizations' technical infrastructures, these efforts can be hindered by human factors such as a lack of trust, ineffective communication between security teams, and unwillingness to invest in security by upper management. Our findings additionally demonstrate the factors that were taken into consideration by organizations when deciding on how to run their vulnerability management programs. These include decisions related to scoping vulnerability assessments, creating bug bounty programs, building internal security teams, and outsourcing security testing to external researchers.

We show that organizations were not always sufficiently informed about how to make such types of strategic security decisions. This led some of them to create bug bounty programs before establishing internal processes for handling vulnerability reports or consider such programs as substitutes for internal security teams. We therefore leave to future work the task of exploring the extent to which existing security standards are sufficiently guiding organizations through how to structure their vulnerability management programs, how to distribute security testing responsibilities among internal or external security testers, and how to decide on whether to utilize offensive versus defensive security testing strategies (e.g., red teaming versus blue teaming).

Chapter 4

Understanding Organizational Vulnerability Remediation Processes

4.1 Introduction

Maintaining the security of software systems requires software professionals to take timely actions on remediating security vulnerabilities that affect them. In organizational contexts, these could involve software systems that are built in-house or taken from third-party providers (e.g., software libraries). The results of the research study presented in the previous chapter showed that organizational vulnerability management processes are hindered by organizations' lack of abilities to remediate discovered vulnerabilities effectively (see Chapter 3).

External security testers who were either penetration testing contractors or bug bounty hunters discussed their experiences with organizations that failed to address security vulnerabilities long after they were reported to them. It was a common experience for many of them to find the same vulnerabilities at later points in time either because organizations delayed remediating them or released incorrect fixes. Security managers who participated in the same study confirmed this observation by explaining that they frequently found themselves unable to remediate security vulnerabilities in a timely fashion despite their awareness of their existence. In this chapter, we investigate organizational vulnerability remediation processes, focusing specifically on identifying the factors that might lead organizations to leave security vulnerabilities unfixed, delay fixing them, or release incorrect fixes.

To handle discovered security vulnerabilities, organizations are assumed to define systematic procedures that allow them to evaluate vulnerability severity levels, identify all affected systems, devise proper plans for remediation, and validate whether they were effectively remediated. To allow informed decision-making at each of these phases of the vulnerability remediation process, they need to have sufficient contextual information about their environments, affected systems, technical owners of systems, and any other stakeholders whose involvement in the remediation process might be needed (e.g., developers of third-party libraries). While prior work investigated specific technical vulnerability remediation practices (e.g., applying software patches and fixing security configurations [81, 140, 141]), we are yet to understand how organizations' vulnerability remediation decisions are taken at each of the phases of the remediation process. These include decisions related to how vulnerability severity levels are determined, how vulnerability remediation responsibilities are assigned, and how vulnerability fixes are validated. These are only a few examples of strategic remediation decisions that require organizations to consider their business priorities and technical capabilities to avoid potential security consequences. For example, failure to validate vulnerability fixes could lead organizations to believe that they remediated a critical security vulnerability when they actually did not.

To identify the factors that might lead to shortcomings in vulnerability remediation practices, we conducted semi-structured interviews with 21 security professionals who were involved in remediating security vulnerabilities in technical or managerial capacities. During the interviews, we focused our discussions on understanding how security vulnerabilities are normally handled after they are discovered through vulnerability discovery channels, which include those detected by vulnerability scanners or reported by security testers. Our goal was to shed light on challenges experienced with reproducing discovered vulnerabilities, determining their severity levels, remediating them, and validating fixes. We also examined overall remediation workflows by discussing with participants the nature of interactions that take place between those involved in remediation processes (e.g., software developers, testers, and managers). Additionally, we conducted a follow-up survey with 42 security professionals to obtain further insights into vulnerability remediation processes. In this study, we posed the following research questions:

RQ 1 What are the factors that might lead organizations to decide to not remediate or delay remediating a discovered security vulnerability?

RQ 2 How do security professionals handle discovered vulnerabilities that exist in first- or third-party code?

RQ 3 Do vulnerability fixes get validated? If so, what validation approaches are used?

Overall, we found that lack of sufficient resources dedicated to security prevented organizations from remediating all discovered security vulnerabilities and led them to focus on remediating critical and high severity vulnerabilities. However, we also observed that they struggled to confidently determine vulnerability severity levels due to not having sufficient contextual knowledge on internal software assets or not having internal security expertise capable of understanding the nature of vulnerabilities brought to their attention. In the following sections, we present the methodology we followed to conduct this research study and discuss our findings in detail.

4.2 Methodology

Below, we explain the methodology we followed to recruit security professionals to participate in our interviews and follow-up survey, and provide details on the process we followed to qualitatively analyze the collected data.

Semi-Structured Interviews

From late October 2021 to late March 2022, we conducted 21 semi-structured interviews with security professionals who were involved in vulnerability remediation processes in their organizations. The interviews lasted for an average of 56 minutes. For recruitment, we sent invitations by e-mail to security teams whose organizations were receiving vulnerability reports through bug bounty programs hosted on HackerOne [114] or Bugcrowd [43]. We also used social media sites (e.g., LinkedIn [147], Twitter [211], and Reddit [187]) to invite security professionals to participate in our study, and recruited other professionals through personal connections. We offered interview participants $20 Amazon gift cards as a token of appreciation for their participation.

Our interview guide consisted of questions that aimed to help us understand how security vulnerabilities reported by internal or external security testers or detected by automated vulnerability scanners are handled. We particularly focused on identifying the approaches relied upon for triaging security vulnerabilities, understanding how vulnerability remediation responsibilities are assigned, investigating the nature of interactions that take place until a security vulnerability is remediated (e.g., interactions with software development teams and developers of third-party code), and exploring the approaches employed to validate the effectiveness of vulnerability fixes. We additionally focused on obtaining improved understanding of the extent to which organizational factors were impacting vulnerability remediation efforts (e.g., internal security culture and team structures). The interview guide is provided in Appendix B.1.

To analyze the data collected through the semi-structured interviews, we conducted qualitative thematic analysis of 20 interview transcripts. One participant did not consent to audio-recording, and so we instead used notes that we took during their interview in our analysis of the interview data. Two researchers jointly developed a codebook based on an iterative analysis of a subset the interview transcripts. They started by developing an initial codebook based on their iterative independent analysis of four interview transcripts. During this process, they had meetings to discuss their codes, reach agreement on which codes to include in the final codebook, and decide on how to use each of the them. They then followed the same process to code a few additional transcripts, which resulted in adding a few codes to the codebook.

The two researchers then used the final codebook, which consisted of 53 codes, to independently code the remaining transcripts and continued to meet to discuss their coding results, review each other's coding results and resolve disagreements. This resulted in reaching "almost perfect" level of agreement (Cohen's $\kappa = 0.844$). We structured the codebook to include codes that capture details related to how organizations decide on whether to remediate (e.g., *vulnerability-reproducability* and *remediation-criteria*), what workflows are followed in order to remediate or validate a vulnerability fix (e.g., *contact-points-internal* and *remediation-validation*), and how non-technical factors impacted remediation activities (e.g., *vulnerability-reports-quality* and *culture-within-organization*).

Follow-up Survey

We used the same recruitment channels to invite security professionals to participate in a follow-up survey, which we conducted from June to September 2022. To incentivize participation, we offered participants to enter drawings for five $100 Amazon gift cards. The survey was designed based on the preliminary results of the interview study and aimed to help us obtain additional insights into vulnerability remediation processes. We specifically were interested in learning about the factors that impact security professionals' abilities to triage and remediate vulnerabilities effectively as well as understanding the extent to which organizational constraints affected their vulnerability handling efforts.

We also included questions that helped us understand the roles that various stakeholders play in remediation processes (e.g., third-party code developers, testers, and managers). The survey additionally asked respondents about the resources they rely on for technical guidance on how to remediate vulnerabilities, their perceptions of whether their organizations were technically capable of remediating vulnerabilities effectively, and the factors that could lead them to be unable to remediate vulnerabilities. The survey questions are included in Appendix B.2.

The number of security professionals who participated in this survey is 42. Since the survey included open-ended questions, two researchers followed the same process followed to analyze the interview data to develop another codebook that consisted of 16 codes. They then used it to code the survey responses, discussed their coding results, and resolved disagreements. This resulted in reaching "substantial" level of agreement (Cohen's $\kappa = 0.79$). The interview and survey studies were approved by the Institutional Review Board (IRB) at our university.

4.3 Results and Analysis

Our semi-structured interviews were conducted with 21 security professionals, 19 of which were directly involved in remediating security vulnerabilities in a technical or managerial capacity whereas the remaining two were indirectly involved (i.e., they were consulted by security or development teams on how to triage and remediate discovered security vulnerabilities). Similarly, 32 of the 42 security professionals who responded to our follow-up survey were directly involved in remediating vulnerabilities in their current or past roles, 9 were indirectly involved, and 1 understood their organization's remediation process but was not involved in it.

Of the 21 interview participants, 7 were responsible for technical vulnerability remediation tasks only (i.e., software developers or security engineers), another 7 participants held managerial roles only (e.g., managing engineering teams responsible for remediation), and the rest had technical and managerial experiences in remediating security vulnerabilities. Most of security professionals who participated in our follow-up survey were security engineers or software developers (29 of 42), 5 held managerial security roles only, and 8 had technical and managerial experiences in vulnerability remediation.

Additionally, 48% of the 21 interview participants had been involved in remediating security vulnerabilities for more than 10 years, 19% had between 6 and 10 years of experience, and 33% had less than 5 years of experience in vulnerability remediation. Similarly, 47% of the 42 who responded to our follow-up survey had more than 6 years of experience in vulnerability remediation. In the following sections, we present the results of our analysis of the data collected from security professionals through the interviews and our follow-up survey and shed light on organizations' end-to-end vulnerability remediation processes.

Motivations for Remediation

We identified a number of factors that motivated security professionals to take action on remediating discovered security vulnerabilities. In the following, we explain the impact of each of these factors on vulnerability remediation decisions.

Compliance Responsibilities

Our observations suggest that organizations' interest to meet the requirements of specific regulations or industry standards impacted their remediation decisions. These included decisions related to identifying the types of vulnerabilities that should be remediated, defining timelines for remediation, and dedicating resources to remediation work. One participant explained:

"If a vulnerability is compliance-related, it definitely needs to be fixed...it is just not up to discussion that we cannot fix it" ($P10$).

On the other hand, shaping remediation decisions around compliance responsibilities could drive organizations to only remediate vulnerabilities that have compliance-related consequences. This might lead them to not take remediation decisions that are driven by accurate understanding of their organizations' attack surfaces. One participant provided an example of how compliance-driven approaches to remediation can lead organizations to decide not to remediate a security vulnerability:

"Sometimes there are patches available but not approved by the vendor .. and you are simply not allowed to install it, otherwise you are liable." ($P5$)[1].

[1]We use P to refer to interview participants and R to refer to survey respondents.

Previous Security Incidents

Experiencing negative security events is another factor that motivated organizations to improve their remediation processes. Participants discussed that experiencing the consequences of security incidents led their organizations' leaders to dedicate more resources to remediating security vulnerabilities and strive to reduce the time taken to remediate. Two participants explained:

"It wasn't until there was an incident where an attacker breached the environment that anybody cared about what's the right way to do this." (*P*4)

"All of the sudden there was a vulnerability floating around that is available for everybody to know about that we have not fixed, that is now affecting all our clients, so we got that fixed pretty darn quickly." (*P*15)

Potential Impact on Reputation and Revenue

Many of the participants offered services to customers or were relying on services offered by vendors (e.g., third-party tools or libraries). These business relationships affected organizations' willingness to remediate security vulnerabilities because they did not want their reputations or revenue sources to be negatively affected. Participants whose online services were used by customers explained that their customers found vulnerabilities after auditing their services for compliance with certain standards, which they needed to prioritize remediating to maintain trustworthy relationships with these customers. Such decisions were also based on an evaluation of the size of their customers and the amount of revenue they expected to gain from them. One participant explained their experience:

"They have paid for pen testing on our software and so when they found vulnerabilities from their pen testing, there was a lot of pressure on us [from leaders in their organizations] to fix those." (*P*12)

Similarly, participants who discovered vulnerabilities in third-party services noted the positive effects that they experienced after using their contractual agreements with their service providers to pressure them to take timely actions on remediating discovered vulnerabilities. One of them explained:

"Whenever you put pressure monetary-wise on businesses, they are more shaken up and they don't want to lose the contract, they want to fix the thing." (*P*19)

Participants also explained that they considered the potential impact of exploiting vulnerabilities on their reputation in their remediation decision making. This was particularly relevant for participants who stored sensitive customer data in their systems, as they were concerned about the possibility of experiencing data breaches or being subject to legal liability as a result of not remediating certain security vulnerabilities. For instance, one participant explained why they were urging their clients to update an old version of their software that was vulnerable:

> *"This will impact our reputation if they get breached because people will say that a version of our product got breached."* ($P15$)

Organizational Policies

Participants' organizations varied in terms of the extent to which vulnerability remediation work is considered in their organizational work policies. A number of them were part of organizations that defined formal processes for remediating discovered vulnerabilities, which allowed keeping track of the remediation work performed by security professionals. Such tracking of progress put pressure on professionals to remediate discovered vulnerabilities to avoid potential negative consequences (e.g., not allowing them to release new product features). For example, one participant from a large organization noted that delaying remediating a security vulnerability could result in *"restrictions that gets escalated way up the management chain."* ($P10$)

Other participants mentioned that their organizations did not have formal processes for tracking progress on remediating discovered security vulnerabilities. This was mainly due to the lack of communicated expectations from their organizations on who should be responsible for remediating discovered security vulnerabilities. To encourage professionals to remediate vulnerabilities in systems under their control in these types of organizations, one participant explained that they were offered salary increases by their organization who did not consider remediating security vulnerabilities as part of their professionals' formal workloads. Another participant who held a managerial position in a larger organization explained that they addressed this shortcoming in their organizational policies by having regular meetings in which each of their different product teams is given a chance to discuss their progress on remediating vulnerabilities with their other teams and compare their performance with that of the other teams to *"create a sense of competition between systems' owners."* ($P4$)

Support of Organization Leaders

We also found that security professionals' abilities to remediate security vulnerabilities effectively is influenced by the extent to which their seniors are recognizing the importance of doing so and providing needed support. To take timely action on remediating a discovered security vulnerability, security engineers might need to assign tasks to other stakeholders (e.g., software developers), which could lead to delaying work on other tasks that have more visible business impact. Organization leaders therefore play a major role in communicating expectations to all those involved (e.g., software developers) with regards to who should be responsible for remediating discovered vulnerabilities and what types of risks can potentially be accepted. Our observations also suggest that security professionals whose managers were directly involved in remediation processes (e.g., triaging discovered vulnerabilities, discussing remediation plans, and defining remediation timelines) perceived their remediation processes to be more effective. Two participants explained:

"We are very fortunate to have management that are very much behind the work we are doing, otherwise we would just operate in a vacuum." ($P16$)

"Our CEO was very interested and he did a lot of the work, he liked doing that." ($P15$)

On the other hand, another participant noted that lack of support from their organization's leaders introduced communication challenges with teams that had vulnerable systems under their control, leading them to be unable to enforce requirements that allow them to remediate effectively:

"The CIO was not security focused at all, he just had no interest in security." ($P4$)

Vulnerability Triaging

To handle discovered security vulnerabilities, organizations need to review each of them to evaluate its severity level and determine whether it needs to be fixed across all of their affected systems. This not only requires technical understanding of the nature of the discovered vulnerabilities, but also contextual knowledge about affected systems that allows reaching accurate determination of their potential impact. Not having such understanding could lead organizations to waste resources on remediating a vulnerability that does not have actual impact (e.g., a vulnerability in a system that is not used) or not remediate a vulnerability that have serious potential impact. In the following sections, we report our observations in regards to the factors that are taken into consideration by organizations for determining whether a vulnerability needs to be remediated and the associated challenges experienced by security professionals.

Triaging Responsibilities

We observed that organizations followed various approaches to triaging discovered vulnerabilities. Some of the participants who had bug bounty programs reported using triaging services offered by bug bounty platforms such as HackerOne [114]. For other participants, triaging vulnerabilities was handled by internal personnel, such as developers, security engineers, or managers. For participants who were part of small organizations, they mostly followed informal approaches to triaging discovered vulnerabilities and were less likely to experience challenges with identifying code owners who can be assigned remediation tasks. A number of them also mentioned that their managers were involved in triaging discovered vulnerabilities and prioritizing remediation work accordingly. They also did not follow predefined criteria for determining whether to remediate or formal timelines that are determined based on vulnerability severity levels. For example, two of them explained their approaches to triaging:

"It is more or less whoever triages the issue on the security team will make a call." ($P1$)

"It is more of, sort of, a rough outline of 'Oh crap, this could be used to steal somebody's identity, that's serious, we've really got to fix this'." ($P12$)

Participants who were part of larger organizations tended to follow more structured approaches to triage vulnerabilities, which employed specific methods for determining vulnerability severity levels and defining remediation timelines. However, the size and complexity of their codebases made them more likely to experience challenges with locating all parts of their systems that are affected by a discovered vulnerability and finding code owners who would be able to implement vulnerability fixes.

The need for contextualizing discovered vulnerabilities to understand their potential impact necessitates triaging discovered vulnerabilities by personnel who are sufficiently familiar with their organization's technical infrastructure and have cybersecurity expertise. This led most of our participants' organizations to chose to not outsource this task or rely on tools to automate the triaging process. One of them explained that their organization initially outsourced some of their triaging tasks to external security engineers through a bug bounty platform but decided to stop doing so after observing that they frequently *"ended up not flagging something that was actually pretty serious."* ($P1$)

Another participant justified why they were manually triaging all discovered vulnerabilities: *"it will never be 100% perfect and you just cannot risk an important vulnerability to bleed through."* ($P16$)

The results of our follow-up survey supported this observation, showing that most of the 42 respondents dedicated internal personnel to the task of triaging vulnerabilities, 31% outsourced triaging bug bounty reports, and only 29% automated the process of triaging vulnerabilities. Furthermore, 93% and 86% of the 42 respondents indicated that those who triage vulnerabilities at their organizations have reasonable technical and cybersecurity knowledge, respectively.

Vulnerability Severity Levels

Many of the interview participants indicated that they used the Common Vulnerability Scoring System (CVSS) [216] in their evaluation of vulnerability severity levels. They explained that while CVSS was not always sufficient, having a standardized method for evaluating severity levels facilitated discussions with external security testers or third-party code developers about the potential impact of discovered vulnerabilities. It also provided a baseline for their evaluations that they then adjusted based on their understanding of their affected systems. One participant explained how using CVSS made it easier to discuss severity levels with bug bounty hunters:

"You might have a different definition of like 'oh, this is very important' vs somebody else and that's why it is better to default to CVSS." ($P7$)

To determine severity levels based on the context in which a vulnerability exists, other factors were considered by participants' organizations which included the criticality of affected assets and the potential business impact of exploiting vulnerabilities. Other factors that were considered by our participants when deriving severity ratings include whether a vulnerability would affect their compliance with certain industry standards, whether it was reported by a customer, and whether it was being exploited in the wild. For instance, one participant explained how business relationships impacted their their assignments of severity ratings to discovered vulnerabilities:

"If a customer is sending us a vulnerability, that is automatically a high severity vulnerability that we prioritize." ($P6$)

A number of participants who worked for large organizations also mentioned that they used custom vulnerability rating systems that were developed by their organizations for determining severity levels of discovered vulnerabilities. However, our results suggest that having limited information about discovered vulnerabilities during the triaging phase led to introducing a degree of subjectivity to rating vulnerability severity levels, even when an objective rating system is used. For instance, one participant from a large organization said:

"We have an established severity rating model that we adhere to, but how we fill in the specific values is up to the triaging security engineer that is rating that." ($P10$)

Deciding on Whether to Remediate

Our results suggest that organizations' remediation efforts were mostly focused on security vulnerabilities that they deemed to have critical or high severity impact. Remediating medium-severity vulnerabilities received less attention whereas low-severity ones were often ignored. Despite their understanding of the potential negative impact of chaining low-severity vulnerabilities, participants indicated that they were unable to dedicate resources to addressing them and believed that they are unlikely to pose significant risks to their organizations. One of them explained:

"The amount of effort that would go into triaging and documenting every low vulnerability would be immense, .. we cannot cover all the bases all the time." ($P16$)

However, a number of participants who were part of small organizations discussed being in situations where they decided not to remediate due to lack of skills that would allow them to determine severity levels of discovered vulnerabilities or understand how to remediate vulnerabilities that had serious impact. This was also the case for reported vulnerabilities that they could not reproduce due to lack of sufficient reproduction instructions that were provided to them in the initial vulnerability reports. Two of them mentioned:

"One of them we just do not know how to fix, and so we have not done it not because it is not high-priority." ($P12$)

"One of the things that makes that difficult for us is that we often do not know how big of a threat something is." ($P14$)

A few participants from larger organizations discussed their experiences with accepting the risks of vulnerabilities that were discovered in legacy systems under their control. The reliance of their businesses on these systems coupled with them being uncertain about how to remediate such types of vulnerabilities led them to decide not to remediate in order to avoid potential consequences to their businesses. One of them demonstrated their rationale for deciding not to remediate in these situations:

"If you take the system offline, you know there will be a negative impact but if you leave a vulnerable system running, maybe nothing will happen." ($P17$)

This was also the case for security vulnerabilities that were caused by architectural flaws in their systems that were perceived to require significant amount of effort in order to fix, particularly when organizations applied mitigation controls to make it difficult to exploit such types of vulnerabilities.

Those from larger organizations also explained that their processes allowed their managers to accept the risks of certain vulnerabilities when they preferred to dedicate their limited resources on other types of work or did not believe that exploiting such vulnerabilities would have significant business impact. One of them explained how such decisions would lead them to not remediate:

"If somebody who is responsible for the business will make a call and say that 'we are not going to fix this', then they take responsibility for it and I am fine with it." ($P3$)

In other cases, participants indicated that they had decided to leave some of their systems vulnerable due to functionality breakages that they experienced after attempting to apply relevant fixes. For instance, one of the participants justified why they removed a fix from one of their recent software releases:

"We had to do another release to relax our security checks to keep our third party clients working." ($P8$)

Other factors that were considered when deciding on whether to remediate include whether there are exploits available for discovered vulnerabilities, whether there are patches available for systems developed by third parties, and whether organizations would need to take systems that they deemed important offline in order to remediate vulnerabilities. Participants also considered the types of discovered vulnerabilities and the ease of exploiting them when deciding on whether to remediate. For instance, one participant explained why they focused on remediating vulnerabilities that allow remote code execution:

"The ones that are remote code executables, those are the nasty ones and need to be fixed right away." ($P16$)

Challenges with Vulnerability Triaging

One of the main challenges that were experienced by our participants was reaching accurate determination of vulnerability severity levels. This was mainly due to lack of sufficient information available to those responsible for triaging vulnerabilities that:

- allows evaluating the criticality of affected assets to their organizations, and
- facilitates reaching agreement with developers, managers and other stakeholders on whether a vulnerability is of critical, high, medium, or low severity level.

This challenge can ultimately affect organizations' decisions of whether to remediate a discovered vulnerability, as not being able to accurately understand the potential risks of a vulnerability could lead them to determine that a vulnerability is of low severity when it is not. One of the participants demonstrated:

"It will end up in discussions between operations and the security department going back and forth over 'is this something that really needs to be solved?' " ($P3$)

Some of the reasons why organizations struggled with determining vulnerability severity levels include having limited information in vulnerability reports on the nature of reported vulnerabilities and not having information readily available to triagers on affected systems. The effort required for triaging vulnerabilities could also entail having to understand how vulnerable software dependencies, which might have their own dependencies, could introduce risks to organizations. This becomes more challenging when those responsible for triaging do not have sufficient cybersecurity expertise or when they have to determine how to handle novel vulnerabilities. One of the participants explained how lack of expertise can lead to disagreements on severity levels between those responsible for vulnerability triaging and external security testers:

"It becomes a problem when you are talking to the person externally who is actually in security and knows the vulnerability really well, because then you have a dissonance on whether or not they actually understand the vulnerability." ($P7$)

A common challenge experienced by our interview participants was having to deal with vulnerability reports received from internal or external testers that missed important details, such as complete reproduction steps and contextual information on how a vulnerability was discovered. Many of the participants explained that they only have a finite amount of attention that they can spend on triaging vulnerabilities, and struggled to handle vulnerability reports that included inaccurate details or were incomplete. This finding was supported by the results of our follow-up survey which showed that 79% of respondents faced challenges with reproducing vulnerabilities due to not having sufficient details in vulnerability reports. For instance, one participant demonstrated why they struggled with vulnerability reports:

"They [bug bounty hunters] get lost in the technical details but not really describing if it is like an access control vulnerability, they don't describe the different parties that they are working with, the different roles that they are working with and which request is made in which context." (P10)

For small organizations, they also discussed that lack of cybersecurity skills on their teams made it challenging for them to reproduce reported vulnerabilities and determine what actions to take to remediate. Furthermore, organizations whose remediation processes mainly relied on applying patches supplied by third parties found it difficult to handle vulnerabilities that required taking different actions to remediate. Others explained that their teams consisted of software developers with no training in cybersecurity, which made the task of following reproduction instructions that required them to use security tools challenging for them. One of them explained this challenge:

"This is just that sort of like professional division between like security professionals and web devs is that every report is like 'just open up BurbSuite and do a thing' and I am like 'I don't know what that is'. " (P14)

Participants whose organizations had bug bounty programs that incentivized external researchers to report their findings by offering monetary rewards struggled with the volume of vulnerability reports that they had to triage. They mentioned that they often received poor-quality reports which required significant manual effort to review but mostly contained minor discoveries that they considered insignificant. One participant explained this challenge:

"There are a lot of people who run automated tools and then they file lots and lots of reports without even necessarily checking that the particular application that they are filing reports against is really affected." (P1)

Asset Management and Ownership

One of the main reasons why participants' organizations faced challenges with triaging vulnerabilities was that they did not have effective solutions for managing and tracking assets that were under their organizations' control. Many participants, and particularly those who worked for large organizations, explained that information about their organizations' assets was either not available or not kept up to date. Lack of visibility into their technical infrastructures (e.g., third-party dependencies and their versions) limited their abilities to take informed decisions on how to handle discovered vulnerabilities and understand the impact of such vulnerabilities on their organizations' attach surfaces. This also made it more likely that they would not be able to systematically apply vulnerability fixes across all affected systems due to not having up-to-date information on all of their assets prepared in advance. In our follow-up survey, 74% of respondents indicated that they often had to spend at least moderate amount of effort to comprehensively identify assets affected by discovered vulnerabilities. Participants explained:

"It is really one of my biggest headaches with IT that they don't do asset management very well at all." ($P4$)

"It is hard to understand who's got which systems running and yeah it is really not very clear." ($P6$)

"There is always the fear of missing some dark spot where you just did not have eyes on them, and it is still vulnerable, that's what keeps you up at night, fear of breach kind of thing." ($P16$)

A recent example of a critical security vulnerability that required organizations to have clear understanding of all dependencies in their software stacks is the Log4j vulnerability that was discovered in late 2021 in an open source Apache library, which was widely used by organizations [21]. One participant explained why organizations struggled with handling this vulnerability:

"And the other thing that made log4j so difficult is the discovery, where is this thing? What systems actually have this library in it? This is why we are talking about a software bomb." ($P11$)

Finding code owners who should be responsible for remediating discovered vulnerabilities was another challenging task that negatively affected remediation timelines. Participants indicated that not maintaining up to date information on code owners in their organizations and third-party code developers who should be contacted to request patches was one of the reasons why remediating certain vulnerabilities was delayed. The difficulty of taking timely remediation actions due to lack of this information increases as organizations deploy more services that make use of additional third-party dependencies. The results of our follow-up survey supported this result, as 40% of respondents either did not have access to information on who should be responsible for maintaining their systems or indicated that this information was incomplete. One survey respondent explained:

"Most of our time is 'hunting down' the point of contact for the apps and servers." ($R11$)

A related challenge was the lack of clarity around who should be responsible for remediating vulnerabilities discovered in legacy systems or systems whose owners left their organizations. One participant explained why they had to decide not to remediate certain vulnerabilities because of this challenge:

"There are many things which get accepted just because of this reason that we don't know who built that part of it or who is responsible for fixing it, so we cannot really touch it because we don't know what is going to happen if we shut it down." ($P19$)

Vulnerability Remediation Processes

While we observed variations in the remediation processes described by our interview participants, our results suggest that larger organizations tended to follow more formalized approaches to remediating discovered vulnerabilities compared to small organizations. A number of participants who were part of large organizations explained that their organizations had already used vulnerability severity levels as a basis for defining specific timelines for remediating discovered vulnerabilities. Security professionals in these organizations had to follow these timelines when remediating vulnerabilities unless approvals for extensions were obtained from their seniors. In cases where remediating a vulnerability was not feasible, someone from their organizations had also to formally document their acceptance of potential risks. One participant explained how they handled such cases:

"Someone has to sign off on that and agree to be accountable for the consequences of not fixing the problem." ($P17$)

Some of these organizations also employed approaches for tracking and documenting progress on remediating discovered security vulnerabilities. While they had the capability of knowing which vulnerabilities were not remediated on time, lack of consideration of remediation responsibilities in their engineers' normal workloads had negatively impacted their progress on remediation. Challenges related to managing assets and dealing with accumulated technical debt also made their processes less effective and contributed to discouraging them from improving their vulnerability discovery efforts. For example, one participant justified the reason why they were not using more channels for vulnerability discovery:

"Right now I am focused on the ones that we can identify easily. Until we get through that backlog, we don't really pay attention to those, which is probably a bad thing to say." ($P4$)

Interview participants who were part of small organizations either had small security teams or did not have someone on their teams with formal cybersecurity training. These organizations tended to not have formal timelines for remediation that they asked their personnel to follow. Their remediation timelines were based on their resources and the extent to which they were able to understand what they needed to do to remediate discovered vulnerabilities. Two participants explained how they determined remediation timelines:

"If it is something that is an easy patch, then that could be a week but for something longer, honestly it could take months before we have it fixed." ($P6$)

"It is sort of like three states: do it now, put it on the regular queue of work to be done or sort of backlog it and these are the ones that do not get fixed, that are just setting on our backlog, nobody is working on them, and nobody is about to work on them." ($P12$)

Lack of internal cybersecurity expertise was also a main factor that led these organizations to delay remediating discovered vulnerabilities. One of the participant provided an example:

"We did not have a cross-site request forgery protections on our site because none of us knew what that was and again this kind of goes to like the nature of our team." (*P*14)

The results of our follow-up survey showed that 52% of respondents had to follow remediation timelines which are pre-defined by their organizations, whereas the rest indicated that their processes allowed more flexibility in this regard. Additionally, only 40% indicated that not remediating discovered security vulnerabilities required approval from their managers.

There are other process-related factors that impacted progress on remediating discovered security vulnerabilities. These include having formal regular reporting of progress that allows continuous assessment of the performance of involved teams. Those who followed this approach indicated that it facilitated their risk assessment efforts and helped them understand how to improve their processes in the long term (e.g., by reducing the number of accepted vulnerabilities or identifying areas where mitigation measures had to be put in place). One of the participant explained the approach followed by their organization:

"We have monthly reports from the different business teams on how many vulnerabilities were reported to them and how many of them were fixed in time." (*P*20)

Another factor was defining remediation responsibilities in contractual agreements with customers or vendors of third-party software. Some participants explained:

"Some people will pay us a lot more extra money so that so we do the product updates for them." (*P*15)

"If there is a violation in such cases where the vendor is reluctant enough not to fix the vulnerability, then we go back to the contract, we involve the legal, we involve the business." (*P*19)

Thus, formality of remediation processes had the advantage of introducing structure to remediation activities that facilitated:

- coordination between involved stakeholders,
- tracking of progress, and
- clarifying who should be responsible for remediating discovered vulnerabilities early in the process.

Variations in the extent to which organizations formalized their remediation processes by documenting remediation decisions, defining timelines, and tracking progress can therefore explain the extent to which they were able to remediate vulnerabilities efficiently. Shortcomings in these areas could also increase the cost of remediation activities in the long term. However, the dynamic nature of security vulnerabilities would still require organizations' processes to be flexible enough to allow taking timely actions even when their processes do not sufficiently support doing so. For instance, one of the participant explained why it would take them long time to remediate certain types of vulnerabilities:

"That is a struggle there because it is just not part of the same process so it is harder to fit it in and make sure that we are prioritizing it correctly." ($P6$)

Vulnerability Remediation and Third Parties

When asked about how vulnerabilities in third-party code were handled, most participants indicated that they notified third-party developers about vulnerabilities discovered in their code and asked for patches. However, participants described their experiences with third-party developers who either:

- needed long time to release patches, or
- thought that the reported vulnerabilities were of low severity levels and, because of that, did not agree to release patches for them.

For instance, one of the interview participants discussed their experience with one of their software vendors:

"We found this RCE [Remote Code Execution vulnerability] on a third party component in the infrastructure, and their providers said okay we are going to fix it in 6 months." ($P13$)

One survey respondent explained another type of disagreement that they experienced with their third-party developers:

"There have been a few cases where these packages do not consider our situation a bug." ($R11$)

When patches were received from third-party developers, participants explained that they were not always able to remediate discovered vulnerabilities effectively. In certain cases, patches were incomplete or introduced new vulnerabilities. These include scenarios where patches were not addressing the root causes of discovered vulnerabilities or were developed without considering all variants of vulnerabilities that allow bypassing certain types of fixes. In other cases, applying them introduced functionality breakages in other of their software components, which led participants' organizations to either delay applying them or decide not to apply them at all. Three participants explained their experiences:

"They had released a patch for a very critical vulnerability and it was shortly after discovered that the patch provided was not complete, there were workarounds that allowed people to still be vulnerable." ($P9$)

"..SDK had a critical vulnerability that was only solved in the next major package release but that was going to break six of our internal components." ($P14$)

"One version of it [Apache Struts] has a very serious remote code execution vulnerability but you cannot just upgrade your Struts version over night, that might be a six month project." ($P17$)

The results of our follow-up survey showed that 64% indicated that their organizations had pre-defined processes for remediating vulnerabilities discovered in third-party software they used and 37% found such vulnerabilities to be more challenging to remediate. Furthermore, at least 45% of respondents experienced receiving ineffective patches from third-party developers.

From a process-related perspective, maintaining information on all third-party software being used and contact information of third-party developers that can be used to request patches can also make remediation processes more efficient. In our follow-up survey, we found that only 48% of respondents maintained a list that included contact information about their third-party developers, and a number of them believed that their lists were incomplete. Participants also indicated that while there are tools that can be used to keep track of third-party software packages used by their organizations, these tools might not be affordable to small or medium size organizations. They also expressed the need for tools that allow them to understand how their usage of such types of third-party software components is affected by vulnerabilities that existed in them.

Vulnerability Remediation Process Effectiveness

Validating Vulnerability Fixes

We explored how organizations validate that vulnerability fixes are effective (e.g., cannot be bypassed by using variants of originals vulnerabilities). For vulnerabilities discovered by external researchers (e.g., penetration testers or bug bounty hunters), participants indicated that they followed the reproduction instructions provided in vulnerability reports to validate whether vulnerabilities still existed and asked the external researchers to retest fixes whenever possible. One participant explained:

"Once it is marked as remediated we automatically send it to the penetration testers to retest it and see if it is still vulnerable or not." ($P9$)

For vulnerabilities detected by automated vulnerability scanners, investigating whether the same vulnerabilities continued to be detected by the same scanners for validating vulnerability fixes was the primary validation method used by our participants. However, they also did not believe that this validation method always allowed them to be confident that vulnerabilities were remediated effectively. One participant explained the reason why this was not sufficient for validating their vulnerability fixes:

"There are enough false positive reports that it is hard to say whether if you see something that is reported not fixed, does it really mean that it was not fixed? Or is the scanner reporting something incorrectly?" ($P17$)

For patches provided by third parties, participants indicated that they had to trust that they were effective unless they were told otherwise. One of them explained:

"We really put a lot of trust in patch and whoever gave us the patch and the only way we would look at it again is if another CVE came out to say that it is still vulnerable with the patch." ($P6$)

In other cases, participants asked their customers who used their software to check vulnerability fixes provided to them or needed to ask experts to check vulnerability fixes that were addressing novel vulnerabilities. One of the participants explained the approach they followed to validate a fix they implemented for a vulnerability that required special expertise:

"We wanted someone else to review the code before we push it out ... because it would take another expert to find bugs in there probably." ($P1$)

The results of our follow-up survey showed that only 38% of respondents followed formal processes to validate their remediation efforts. Survey respondents also indicated employing the approaches mentioned above for validation (e.g., rechecking with external researchers, following reproduction steps included in vulnerability reports, and checking outputs of vulnerability scanners).

Perceived Effectiveness of Remediation Processes

We explored the extent to which our study participants believed that their organizations' remediation processes were effective. Some participants thought that their processes were effective while others discussed areas for improvement. Furthermore, 60% of survey respondents stated that they would be likely to recommend their remediation processes to other organizations. For instance, when asked about how their remediation processes can be improved, one participant mentioned:

"I think we are pretty good and we don't need to change anything just yet." ($P8$)

Other participants identified areas for improvement which they believed would require dedicating more resources from their organizations to remediating discovered vulnerabilities. These were mainly related to:

- reducing the number of vulnerabilities that they do not end up remediating,
- investigating the extent to which their use of third-party code is impacting their organizations' attack surfaces,
- following more structured approaches to remediation, and
- hiring more skilled security professionals.

A number of participants explained how lack of resources for remediating security vulnerabilities made their processes less effective:

"There is a lot of vulnerabilities and not enough people to fix them." ($P3$)

"We are not getting paid by our customers to implement security, it is a cost to us, and so with limited funds and limited attention, there is only so much we could put on to that." ($P6$)

"It is really hard to justify a 150 to 200k in an organization as small as we are for a specialist role to the people who in the end cut the checks." ($P14$)

Others believed that their organizations did have skilled professionals who were capable of improving their organizations' remediation processes, but that tasks that had more direct impact on their businesses were often given higher priority. For instance, one survey respondent explained:

"We have a lot of experienced software developers, but they are always very busy and most change companies within 10 years, so it's hard to keep track of good technical contacts." ($R41$)

Shortening software release cycles to allow releasing updates that address discovered vulnerabilities sooner was also mentioned by a few participants as another area for improvement. One of them explained how this was affecting their remediation process:

"So sometimes a thing can be fixed and then sit for as much as a month before it actually makes it into production, and so that is a long cycle." ($P12$)

Other areas for improvement that were mentioned by our participants included:

- obtaining visibility into all vulnerable systems,
- refactoring certain components to allow minimizing functionality breakages that resulted from applying vulnerability fixes, and
- devising systematic approaches for scoping discovered vulnerabilities to allow obtaining accurate understanding of the potential impact of exploiting them.

For instance, one of the survey respondents explained the type of support they needed:

"Implementing a aggregating software to bring all issues into a single-pane of glass." ($R7$)

Communication and Coordination

Taking timely actions on remediating discovered security vulnerabilities often requires efficient communication between different engineering and managerial roles. In this section, we discuss the nature of these communications and explain the impact they might have on vulnerability remediation practices.

Security Engineers and Software Developers

Depending on how teams are structured within an organization, security engineers might need to coordinate with software developers to remediate discovered security vulnerabilities. While security engineers are well-positioned to understand the nature of discovered vulnerabilities, software developers often have more knowledge about the products they contributed to developing. Lack of effective communication between these roles could therefore have negative effects on remediation processes. We explored the dynamics of these communications, focusing on their impact on remediation timelines and the quality of vulnerability fixes.

A number of our participants who worked as security engineers discussed that they had to communicate with software developers in their organizations to request remediating discovered vulnerabilities, and reach agreement on vulnerability severity levels or remediation timelines. They explained that such communications helped them identify cases where they realized that a discovered vulnerability cannot be exploited when developers provided more details about code they had written. The information exchanged in these communications also helped address misunderstandings from both roles, clarify to developers how their code can be exploited, and provide security engineers with more context about affected systems. Participants explained how their communications with developers impacted remediation decisions:

"We will reach out very quickly to those people to try and get a better idea of whether or not it is a real bug and then estimate the effort involved in fixing it." ($P1$)

"They are coming back and telling us like 'Hey, actually the application is designed like this, .. it is not as severe as you thought, .. , this vulnerability instead of a critical is actually a low.' " ($P7$)

However, participants also explained the nature of disagreements that they experienced with software developers, which resulted from:

- lack of defined timelines by their organizations for remediating discovered vulnerabilities,
- lack of resources for remediation, or
- lack of complete understanding of the nature of discovered vulnerabilities.

One participant provided an example of the types of discussions they had with their developers:

"It is kind of like 'no, you kind of misunderstood what the correct fix is, you misunderstood what the vulnerability was and how we could be attacked.' " ($P9$)

In other cases, these disagreements resulted from developers being concerned that remediating certain vulnerabilities would break other parts of their code and therefore have negative business impact. For example, one participant demonstrated why their developers preferred not to remediate certain vulnerabilities:

"If they fix it there could be two other things that get broken or there are other dependencies which are relying on the particular portion of a software that is being used which is probably vulnerable." ($P19$)

They discussed a number of approaches that they used to address these types of disagreements. One of which is escalating to managers to reach agreement on whether potential risks of not remediating discovered vulnerabilities can be accepted. The other approach involved focusing on providing developers with more details on potential security implications of not remediating or more specific instructions on how to remediate. For instance, one participant who worked for a large organization explained that appointing security champions in each of their product teams and providing developers with regular cybersecurity workshops improved the quality of communications between their developers and security engineers. A few others stressed the importance of:

- converting vulnerability reports to actionable remediation tasks before sending them to software developers, and
- minimizing the number of vulnerability remediation requests to only include those that need immediate attention from developers.

One of the participants explained how following this approach helped them improve their communications with their developers:

"They do not ever get bothered by us unless it is important, which means that we have come to save up what I call internally 'our security credit'. We won't provide them with a lot of noise." ($P16$)

Another factor that impacted the quality of communications between developers and security engineers was having intermediaries between these two roles (e.g., project managers), which negatively impacted remediation timelines in our participants' organizations. The ability of security engineers to provide actionable advice to developers was also affected by the extent to which they were familiar with systems affected by discovered vulnerabilities and had sufficient contextual knowledge about past vulnerabilities that affected them. For instance, one participant explained why their organization did not rotate security engineers across product teams:

"It always helps to keep fixed engineers so they know for the last six months, what kind of changes have happened, what kind of vulnerabilities have been fixed and what kind of vulnerabilities have been found." ($P19$)

Relying on ticketing systems only for such communications was also another reason for delaying remediating vulnerabilities, as it was not perceived to be as effective as having direct conversations with developers. This was also perceived as more effective for communicating to developers the level of urgency associated with remediating each discovered vulnerability, one participant explained:

"This person [security engineer] is kind of important because they are kind of the reason for whether or not this issue is going to get fixed and prioritized by these different dev teams." ($P7$)

Security Engineers and Managers

Effective communication between security engineers and their managers can help them define baselines for their remediation efforts (e.g., what potential risks are considered acceptable to their organizations). Our results suggest that the quality of such communications was also affected by the extent to which managers had technical and cybersecurity expertise as well as security engineers' abilities to communicate the importance of remediating vulnerabilities effectively to their managers.

A number of participants explained that their discussions with their managers had direct positive impact on their remediation efforts, leading them to decisions such as prioritizing remediation work over other types of work, removing vulnerable components from their systems, or taking systems temporarily offline to allow remediating vulnerabilities. Two participants discussed how their managers helped facilitate discussions with their developers:

"If they do not understand, I talk to their boss ... Usually, that fixes most cases." ($P13$)

"We rely on our superiors basically, my manager or his manager will go and be like 'fix this!'. The security team is always backed up by the executives to make sure that stuff gets fixed." ($P16$)

However, other participants thought that their managers could play a more active role in facilitating their remediation efforts. Having disagreements on vulnerability severity levels was one of the common challenges mentioned by participants, which was addressed by some of them by explaining potential consequences of exploitation on their businesses to their managers. One participant provided an example:

"Something that expresses the security basics in terms that the other business units think about. So, if you are talking to finance, what is the cost of a breach as opposed to how many systems are unpatched?" ($P4$)

On the other hand, another participant from a small organization explained how challenging it was for them to discuss details about discovered vulnerabilities with their manager who did not have technical expertise:

"That person is going to be like 'what? How?' I'd have to start explaining HTTP because they don't even have a contextual reference of why that is a problem and so it is just hard to do that." ($P14$)

Another participant from a small organization expected their progress in remediating vulnerabilities to improve if their managers showed more interest in improving the security of their infrastructure and dedicated more resources to building an internal security team, they mentioned:

"Honestly, when I go to get my performance review at the end of the year that says how well I am doing, nobody asks me how security of our apps is, they are like 'how productive are we being, are we getting enough features built?' " ($P12$)

The results of our follow-up survey showed that:

- 48% of 42 respondents believed that their managers could help them have better discussions with other teams on remediation work,
- 45% wanted their managers to hire more knowledgeable security professionals, and
- 43% thought that their managers could play a more active role in tracking the progress made by developers in remediating vulnerabilities.

Organizations and External Security Testers

Communications between organizations' development or security teams and external security testers such as bug bounty hunters and penetration testers also influenced remediation processes. When an external security tester reports a vulnerability to an organization, they typically submit reports explaining their findings and might assist organizations with remediation efforts. Many participants explained challenges with vulnerability reports they received which missed details that were required for reproducing reported vulnerabilities or did not include sufficient information on how reported vulnerabilities can be exploited. This was particularly challenging for teams that did not have sufficient cybersecurity expertise. For instance, one participant discussed one of the challenges they experienced while handling vulnerability reports:

"We struggle to reproduce where we are given instructions that are not for us or given instructions that are for another security person." ($P14$)

Another common challenge was reaching agreement with external testers on severity levels of vulnerabilities they reported to organizations. This required more communication effort to further understand the nature of reported vulnerabilities and decide on whether they should be remediated. Two participants provided examples:

"Sometimes it is not a problem at all but there some misunderstandings by the security researcher or misconfigurations that can cause security problems but again it is not a bug, it is with the intended functionality." ($P8$)

"There is some back and forth trying to come to an understanding of like often from our perspective it is not a big problem and trying to explain why that is." ($P17$)

4.4 Discussion

Informed decision making on how to handle vulnerabilities requires having pre-defined processes at an organizational level, which enable curating contextual knowledge about technical environments and system owners as well as communicating expectations on how remediation responsibilities should be distributed. Overall, our results suggest that many of the challenges experienced with remediating discovered security vulnerabilities were caused by lack of timely information available to professionals at the different phases of the remediation process. There were also process-related shortcomings that hindered information sharing between those whose involvement can affect remediation decisions or led professionals to be unable to always take action on remediating vulnerabilities. Below, we discuss how these non-technical factors affected professionals' remediation practices and provide recommendations for how these challenges can be addressed.

Remediation Processes

Remediating vulnerabilities in a timely fashion requires organizations to be prepared to handle vulnerabilities once they are discovered. For them to be able to do so, they would need to have pre-defined processes that can act as frameworks for guiding involved professionals on how to take various decisions that can have direct impact on their businesses. We found that delaying answering questions related to how vulnerability severity levels should be determined, how to decide on whether to remediate, who should remediation responsibilities be assigned to, and what metrics to rely on for evaluating progress over time introduced inefficiencies in remediation processes. Not reflecting remediation responsibilities in organizational requirements was also another factor that contributed to limiting organizations' preparedness to handle discovered vulnerabilities. For instance, security professionals whose organizations had enforced specific remediation timelines found it easier to coordinate remediation tasks with development teams compared to those whose organizations did not communicate similar work expectations.

In order to be able to systematically remediate discovered vulnerabilities, security professionals should be equipped with contextual knowledge about their technical environments that allows them to apply a vulnerability fix across all affected systems. Not having such visibility into technical environments could lead organizations to have false sense of security and not have sufficient understanding of their attack surfaces. This might also make it challenging for professionals responsible for triaging vulnerabilities to accurately determine vulnerability severity levels due to lack of information about the criticality of affected assets to their organizations. The increasing reliance on third-party dependencies makes it even more challenging for organizations to obtain visibility into all the components in their software stacks and take informed remediation decisions. Our results showed that such lack of complete and up to date contextual information about their technical environments is inhibiting security professionals' abilities to take informed decisions at the early phases of the remediation process (e.g., determining vulnerability severity levels) and plan subsequent remediation work accordingly.

Effective remediation of discovered vulnerabilities might require coordination of various stakeholders (e.g., software developers, security engineers, and managers). Not having the responsibilities of these stakeholders clearly defined at an organizational level was a common challenge that participants experienced, which led to communication challenges between involved teams. One participant from a large organization explained that this resulted in considering remediating discovered vulnerabilities as *"a game of shoving the vulnerabilities into someone else's face who needs to fix it."* Maintaining up-to-date information on code owners and developers of third-party code is another type of preparatory work that can enable more effective handling of discovered vulnerabilities. However, our results suggest that security professionals were often not equipped with such types of information at the early phases of the remediation process, which is one of the factors that could lead to delays in the remediation process.

While our results showed that lack of sufficient resourcing for remediation work was a common challenge, performing continuous assessments of remediation processes might help organizations identify reasons that are contributing to increasing the number of discovered vulnerabilities. For instance, the results of such assessments could show that refactoring or removing certain software components would help them eliminate certain types of vulnerabilities in the long term. These assessments could also help them evaluate whether the mean time to remediation in their organizations is helping them reach their security goals and identify process-related shortcomings that can be addressed to improve the efficiency of their remediation processes. For example, they might find that the length of the communication cycle between different teams is introducing communication challenges that are leading to delays in remediation processes. This could also help them understand whether their relationships with certain third parties are negatively impacting their remediation processes (e.g., leading them to use incomplete patches) and whether certain dependencies can be substituted to make their remediation processes more effective.

In order for continuous assessments of remediation processes to be meaningful, we believe that organizations would need to make adjustments to their remediation processes to allow collecting data about their remediation efforts. This could include adopting better documentation practices throughout the remediation process to allow collecting detailed data about remediation decisions (e.g., reasons for why vulnerabilities were assigned certain severity levels). We also expect following systematic approaches to organizations' documentation practices to make their continuous assessments more effective. This could be achieved by providing professionals with templates that allow them to capture specific details about decisions taken to handle each of their discovered vulnerabilities. Such details could include clear descriptions about discovered vulnerabilities, systems affected by the vulnerabilities, assigned severity levels, steps taken to remediate, and stakeholders responsible for remediation.

Another type of improvement that we expect to help organizations optimize their remediation processes is adopting code review practices that allow discovering and remediating vulnerabilities earlier in the software development lifecycle. This would allow reducing the cost of remediating vulnerabilities in the long term and the likelihood of accumulating technical debt as a result of not taking timely action on remediating discovered vulnerabilities.

Vulnerability Reports

Lack of effective communication between the various roles involved in remediation processes was one of the main challenges identified in this research. Internal or external security testers are used to communicating details about discovered vulnerabilities through vulnerability reports. Software professionals who handle these reports might not always get the opportunity to get more details about discovered vulnerabilities from those who prepared the reports. In such cases, the quality of these reports could play a main role in driving remediation decisions, especially when handled by software developers who do not have sufficient cybersecurity expertise.

Our results showed that most participants struggled with handling vulnerability reports that lacked specific details about discovered vulnerabilities (e.g., specific reproduction instructions), and thus left organizations unable to take timely remediation actions. When a vulnerability is discovered by an external researcher, they might not be able to provide remediation instructions due to lack of visibility into organizations' technical environments. And while those handling vulnerability reports are likely to have more information about their organizations' systems, reproducing reported vulnerabilities, remediating them, and validating vulnerability fixes effectively might require supplying them with contextual details on how vulnerabilities were discovered. Vulnerability reports that lack such details could therefore lead those handling them to take remediation decisions based on incorrect assumptions, which could lead them to believe that they remediated a vulnerability effectively when they actually did not.

Well-resourced organizations could partly address this challenge by dedicating professionals to the task of technical writing to help security teams communicate vulnerability information more effectively to developers. However, they would likely still face the same challenges with vulnerability reports received from external researchers. There is therefore a need for interventions that would make it easier for security researchers to prepare more usable vulnerability reports that provide actionable guidance to those handling them. One type of intervention is designing templates that introduce structure to the task of writing vulnerability reports and ensure that relevant details are included. Such details might include vulnerability types, specific reproduction steps, tools that allowed discovering reported vulnerabilities, and instructions on how to identify all affected assets. However, the dynamic nature of security vulnerabilities might require customizing the design of such templates based on certain factors (e.g., vulnerability types), which we expect to be a promising direction for future work.

Actionable Developer Guidance

The complexity of the remediation process offers opportunities for designing actionable guidance that can help professionals triage, remediate, and validate vulnerability fixes more effectively. Many participants, and particularly those who worked for small organizations that cannot afford to build internal security teams, expressed the need for new forms of security guidance that go beyond general security best practices. Our results show that organizations that do not have internal security expertise found the task of handling vulnerability reports received from external researchers overwhelming. While they showed interest in improving the security of their systems, they could not dedicate resources to researching how to handle vulnerabilities reported to them. Given their expertise in software development, they needed guidance on how to reproduce discovered vulnerabilities using tools they were already familiar with and how exploitation of reported vulnerabilities can pose risks to their organizations. They also expressed the need for improving the usability of vulnerability reports such that they can be handled by developers who do not have cybersecurity expertise (e.g., by providing explanations for vulnerability types that are already known to security professionals). For instance, one participant explained:

> *"There is also just an amount of jargon and stuff that is specific to security. I think there is a real lack of developer audience material on how to fix and remediate vulnerabilities."* (P14)

There is also a need for guidance that can help organizations take informed decisions on how to prioritize their remediation efforts. Having this guidance in the triaging phase of the remediation process can help address specific knowledge gaps that organizations might have, such as not knowing whether a vulnerability is being actively exploited and whether there are specific types of systems that are likely to be particularly vulnerable, and allow them to determine vulnerability severity levels more accurately. This can be achieved by building centralized online resources that offer up-to-date guidance in different formats (e.g., decision trees) tailored to the expertise of different stakeholders (e.g., managers, security engineers, and developers). Given the impact that various technical and managerial roles have on remediation decisions, considering the communication needs of these different roles is essential for understanding how to design actionable guidance aimed at assisting organizations with their remediation decisions. Having trustworthy organizations provide such types of actionable guidance could also help reduce organizations' reliance on other sources (e.g., social media sites) that might lead them to follow incomplete or inaccurate guidance on how to remediate vulnerabilities.

4.5 Limitations

The purpose of this research study was to identify the factors that might contribute to limiting the effectiveness of vulnerability remediation processes. While our results revealed differences in the types of challenges experienced by small versus large organizations, our results might not be generalizable since we did not focus on recruiting professionals who work for a specific type of organizations. Given that interview participants were only offered \$20 Amazon gift cards and that many of them worked for organizations that had bug bounty programs, our results are likely relevant to organizations who are more motivated to remediate discovered vulnerabilities than others. While our results are only based on information self-reported by professionals involved in remediation processes, we hope that they will pave the way towards conducting observational investigations of remediation practices in organizational contexts.

4.6 Conclusion

In this chapter, we reported the results of our qualitative investigations of organizational vulnerability remediation processes. We explored the approaches followed by organizations to triage and remediate security vulnerabilities, and shed light on the factors that impacted their decisions of whether to remediate. We found that lack of visibility into organizations' assets negatively impacted security professionals' abilities to triage vulnerabilities effectively. We also discuss how remediation plans are affected by the quality of communications that take place between software developers, security testers, and managers who are involved in remediation processes. We additionally highlight the types of challenges experienced by organizations who do not have internal security expertise when handling discovered vulnerabilities. We hope that our results will spur further discussion on understanding how to help organizations define systematic approaches to remediating discovered vulnerabilities that facilitate information sharing between all the roles that are involved in vulnerability remediation processes.

Chapter 5

Understanding Organizational Privacy Compliance Processes

5.1 Introduction

Since 2000, online services directed at children under 13 in the US have had to comply with the Children's Online Privacy Protection Act (COPPA) [60, 161]. Understanding the successes and failures of differing compliance processes is important, especially as more and more jurisdictions adopt privacy regulations with requirements similar to COPPA's. For example, the General Data Protection Regulation (GDPR) [207] in the EU and the California Consumer Privacy Act (CCPA) [30], like COPPA, require verifiable parental consent when the data subject is below a certain age (Section 5.2).

US regulators regularly pursue enforcement actions against those violating COPPA, including developers of mobile apps and third-party SDKs. For example, HyberBeard, a developer of child-directed apps available on the Google Play Store, was alleged to have shared users' persistent identifiers with third-party SDKs for tracking purposes [69]. Both InMobi and OpenX, providers of advertising SDKs, reached settlements with the U.S. Federal Trade Commission (FTC) over separate allegations that they had collected location data to track child users without parental consent [14, 156]. Despite ongoing public enforcement actions over COPPA's nearly 25 years of existence, prior research on COPPA compliance found potential violations in a majority of child-directed Android apps [136, 167, 189]. While many of these issues were attributed to third-party SDKs that developers had embedded within their apps [186, 189], little is known about *why* these issues persist, the effectiveness of various compliance processes, or how processes can be improved.

In this chapter, we present the results of our investigation of the privacy compliance processes followed by developers of child-directed apps. We distributed two surveys to developers of apps available to children on Google Play, followed by semi-structured interviews

The work presented in this chapter resulted in a research paper which was published in the proceedings of the 22nd Privacy Enhancing Technologies Symposium (PETS 2022) [16].

to gain additional insights. We paid particular attention to developers' perspectives on the requirements of COPPA, GDPR, and CCPA, their experiences with app market policies, their data collection and consent handling procedures, and the processes followed to select and configure third-party SDKs. We also performed technical analyses of some of the apps published by our participants, so that we could discuss the results during the interviews. Our goal was to answer the following research questions:

RQ 1 How do privacy compliance considerations fit in the software development processes followed by developers[1] of child-directed apps?

RQ 2 How do Google Play Store policies and third-party SDK providers influence developers' privacy compliance processes?

RQ 3 To what extent do developers of child-directed apps follow formal processes to vet third-party SDKs for inclusion in their apps and configure them for privacy compliance?

RQ 4 How do app developers' beliefs about their apps' privacy practices compare with their apps' actual practices?

Our sample included software engineers, CEOs, and product managers who work for app development companies of varying sizes, as well as independent developers. We found that most developers place a large degree of trust in the policy enforcement performed by app markets and consider their apps compliant with privacy laws once accepted for inclusion on the stores (Section 5.4). We also found that many developers are unaware of relevant SDK privacy settings, which are necessary to configure SDKs for use in compliance with applicable privacy laws. Our findings highlight the need for usable tools that allow developers to identify potential privacy violations and guide them towards improving their apps' privacy compliance. We discuss the challenges that developers of child-directed apps face when trying to comply with privacy regulations and present recommendations to make it easier for developers to identify gaps in their privacy-compliance processes (Section 5.5).

5.2 Kids' Privacy Regulations

The Children's Online Privacy Protection Act (COPPA) [62] is a federal law that protects the privacy of children residing in the US from inappropriate collection of their personal data by online services, which includes websites and mobile apps [60]. Prohibited practices include collecting contact information (including location data), as well as collecting identifiable data for profiling or behavioral advertising, without verifiable parental consent. It applies to services whose target audiences *include* children—defined as those under 13—even if children are not their *"primary audience."* In the case of mixed audiences that include children, if users' data may be used for COPPA-prohibited purposes, child users must be identified (e.g., using age gates) so that their data receives COPPA-compliant treatment.

[1]We use "developer" to refer to an app development organization. Our recruitment targeted anyone involved in developing child-directed apps, including technical and managerial roles.

The EU's General Data Protection Regulation (GDPR) [207] is a privacy law that applies to online services available to individuals in Europe [207]. Under Article 6 [27], online services are required to have a legal basis for processing their users' data, and obtaining informed consent is one of the legal bases recognized in the law. The GDPR expressly recognizes that *"[c]hildren merit specific protection"* [100]. Thus, when consent is the legal basis for data processing, that consent must be provided or approved by a parent, when the user is a child. The child provisions of the GDPR apply to users under the age of 16, though any EU member state is given the flexibility to lower it to 13 [28].

The California Consumer Privacy Act (CCPA) [30] is a state privacy law that is intended to protect the privacy rights of California residents [30]. Not all online businesses serving Californians are required to comply, and this determination can be made based on the size of the business, the number of users in California, and the amount of revenue produced as a result of data collection [30]. The CCPA requires obtaining verifiable consent from parents before selling data collected from children under the age of 13. However, children who are between 13 and 16 years of age may consent [210]. That is, under the CCPA, those over 16 must explicitly opt out of data sales, those 13–16 must explicitly opt in to data sales, and the parents of those under 13 must choose to opt their children in to data sales.

When COPPA, GDPR or the CCPA apply to a developer of child-directed app(s), and if consent is relied upon as a basis for lawful data processing, the developer would be required to be transparent with regards to the types of data they collect from their users, the kinds of third parties that receive users' data, and the purposes of their data processing activities [30, 99, 210]. Data subjects are provided with other rights, including the ability to withdraw their consent, request that the developer destroy the data they have about them, inquire about how their children's data was used or processed, and stop any potential sales of their data (e.g., to advertising networks) [30, 207].

5.3 Methodology

Our primary goal was to understand the organizational privacy practices of developers of children's apps. To that end, we designed an initial survey and recruited respondents by identifying the official contact e-mails for developers of children's apps published on the Google Play store (Section 5.3). The results inspired us to identify questions for individual developers, allowing us to better contextualize different privacy compliance processes and developers' roles. We therefore targeted our second survey at personnel within organizations and used it to learn about their specific roles (Section 5.3). To capture additional process-related details, we subsequently recruited app developers for semi-structured interviews (Section 5.3).

We collected responses for the first survey[2] from April–August 2021, whereas the responses for the second survey were collected from August–October 2021, and the interviews were conducted from September–December 2021. As an incentive to participate, we offered

[2]Our Institutional Review Board (IRB) did not consider the first survey on organizational processes to be human subjects research; they approved our second survey and interviews.

drawings for five $200 Amazon gift cards at each of the three stages of the study. We promised study participants anonymity: participants wishing to participate in the drawings provided e-mail addresses, which were stored separately from—and unlinkable to—the study data. We collected no other personally-identifiable information to preserve participants' privacy. In the two surveys, we used identifiers coded to organizations, allowing us to link survey responses to apps available in the Google Play store [111], which we separately analyzed.

Survey 1: Organizational Processes

Our initial survey had 37 questions across six sections focusing on understanding developers' awareness of the data transmission and collection behaviors of their apps, privacy compliance processes, and familiarity with relevant children's privacy laws (Appendix C.1):

- A section on understanding whether developers are fully aware of their apps' behaviors, including:
    1. the types of data collected,
    2. the purposes of data collection,
    3. whether parental consent is obtained before any data collection or transmission, and
    4. the means by which consent is obtained.
- A section on understanding:
    1. whether developers have formal processes for addressing privacy issues,
    2. whether they are familiar with COPPA, GDPR, and CCPA, and believe that these regulations apply to their apps, and
    3. the processes followed to vet third-party code for inclusion in their apps.
- Three more sections for each privacy regulation (COPPA, GDPR, and CCPA) that probed organizational processes to address each.

For recruitment, we scraped Google Play [111] to collect e-mail addresses of developers who had apps listed in the Designed for Families (DFF) program [105], the section specifically directed at children. To find additional child-directed apps, we also performed keyword searches, such as "toddler," "kid," "preschool," and "children" to identify developers of apps that target child users. We identified 1,903 developers of child-directed Android apps, and received 50 responses from developers whose apps were available to users in California and the EU, therefore subjecting them to COPPA, GDPR's children's provisions, and potentially the CCPA.

We reviewed the responses to the 23 open-ended questions and then we built an initial codebook, which was used to code the data. It consisted of 18 codes, selected based on the requirements of privacy laws and Google Play store policies [177] (e.g., *parental-consent,* *children-data, opt-out, age-gates,* and *privacy-policies*). Since the responses across all open-ended questions frequently overlapped, we elected to use all 18 codes to code all the questions, rather than choosing distinct codes for each. Two coders independently coded all open-ended responses and then met to reconcile differences (Cohen's $\kappa = 0.74$, "substantial" agreement [139]), such that the resulting coded data represented unanimity.

Survey 2: Developer Perspectives

This survey consisted of 33 questions (Appendix C.2) across four sections to understand respondents':

- roles in the development of child-directed apps;
- personal experiences with the Designed for Families (DFF) program and Google Play Store policies [177];
- familiarity with COPPA, GDPR, and CCPA;
- resources for compliance guidance, and their confidence in their abilities to build compliant apps; and
- efforts to select and properly configure the privacy settings of third-party SDKs.

We also asked respondents about their challenges and the type of support they think they need from their organizations and the Google Play Store [111] to make it easier for them to comply with applicable privacy regulations. We recruited participants through social media, the Google Play Store [111], and additionally used online search tools to find publicly-available email addresses of personnel working at organizations that develop child-directed apps (e.g., searching developers' websites to look for information on how to reach their technical and privacy contacts).

We sent 3,202 invitation e-mails for the second survey, and received 77 responses from individuals involved in developing child-directed apps, corresponding to 72 different organizations. At the end of each survey, we asked whether we could contact participants in the future for follow-up questions; we conducted follow-up semi-structured interviews with those who agreed. For this survey, we built a codebook using the same process as in the first survey which used additional codes (e.g., *SDK-process, privacy-laws-resources, compliance-challenges,* and *Google-Play-compliance*). It was then used to code 16 open-ended questions. As before, two coders independently coded the responses, and then met to reconcile disagreements (Cohen's $\kappa = 0.88$, "almost perfect" agreement [139]).

Semi-Structured Interviews

A subset of survey respondents returned for follow-up interviews, during which we shared the results of our app analyses with them (Section 5.3). (Our second survey contained a question at the end asking respondents if we could follow up with them for interviews.) Our interview guide (Appendix C.3) consisted of questions about the processes participants' organizations follow to comply with privacy regulations, select and configure third-party SDKs, comply with the Google Play Store policies [177], and their experiences publishing apps.

We conducted 27 semi-structured interviews with individuals from 24 organizations, each lasting 60–90 minutes. For participants who consented to audio recording (all except two),[3] we transcribed the recordings and then used inductive coding to build a new codebook. It was then used by two coders to independently code a subset of the data, and then they met to discuss their initial results and made any necessary updates to it. The interview codebook consisted of 40 codes that covered many themes, including privacy processes, third-party SDKs, and experiences with the Google Play Store [111]. As before, the coders independently coded the responses and then met to reconcile differences (Cohen's $\kappa = 0.70$, "substantial" agreement [139]).

App Analysis

Prior to each interview session, we analyzed child-directed mobile apps published by participants' organizations to detect potential privacy issues (i.e., non-compliance with privacy regulations). We used their apps as a child would and then used dynamic analysis tools validated by prior work to examine apps' transmissions [189]. To ensure that we understand how apps behave when child users use them, we proceeded through age gates as a child would, and did not unlock any functionality accessible only to adult users.

We were interested in not just the exfiltration of users' personal data, but also whether any exfiltrations were accompanied by use limitations. As noted in prior work [189], many apps bundle third-party SDKs that provide configuration options that allow them to be deployed in compliance with various privacy laws. Thus, for each third-party SDK identified in participants' apps, we researched how to configure it to comply with COPPA, CCPA, or GDPR. This included integrating these SDKs in test apps we developed, and then analyzing the network traffic to identify whether apps in our dataset were correctly using those configurations.

For example, Google's Admob SDK requires developers of child-directed apps to use a client-side function that sets a flag, `tag_for_child_directed_treatment`, in outbound traffic to indicate that the corresponding data should be treated in accordance with COPPA [110]. For CCPA, Admob also has another flag, `rdp`, which app developers can set to 1 to restrict the processing of data collected from their users (e.g., after an adult user has made a "do not sell request" or before a parent has consented) [109].

Other third-party SDKs, such as Unity Ads [189, 213], have options on their online dashboards that ask app developers whether their apps target children under the age of 13 and then send flags in inbound traffic—to the app—indicating that the app must comply with COPPA (e.g., Unity Ads [213] sets a flag called `coppa` to `true`). Thus, we built a list of the privacy configuration options provided by each of the SDKs in question (Table 5.1).

[3] We interviewed the two participants who did not consent to audio-recording and the notes we took during their interviews were subsequently used in our data analyses.

Our analysis included Google Admob [109, 110], Adcolony [8], Vungle [218], Chartboost [82] and a few other popular advertising SDKs. Finally, we read each app's privacy policy and determined whether they accurately described the observed data flows. When there were inconsistencies between apps' observed behaviors and either their privacy policies or applicable laws, we presented participants with our findings during the interviews.

Table 5.1: Third-party SDK privacy flags.

SDK	**Flags and privacy compliance settings**
Unity Ads [213]	`coppa` (true or false), `appLevelCoppa` (true or false), `calculatedCoppa` (true or false)
Google Admob [109, 110]	`tag_for_child_directed_treatment` (0 or 1), `rdp` (0 or 1)
Adcolony [8]	`gdpr_required` (true or false), `coppa_required` (true or false), `ccpa_required` (true or false)
Vungle [218]	`user_gdpr_consent_status` (opted_in or opted_out), `user_ccpa_status` (opted_in or opted_out)
Chartboost [82]	`pidatauseconsent` (-1, 0, 1), `privacy_us_privacy` (1NY-, 1NN-)
Flurry [95]	`fl.ccpa.optout` (true, false) Note: Flurry does not allow integrating its SDK in apps before developers "certify" that their apps are not directed at children.
Mobfox [155]	`coppa` (0, 1), `gdpr` (0, 1), `gdpr_consent` (0, 1)
Appodeal [26]	`coppa` (true, false) [outbound traffic], `for_kids` (true, false) [inbound traffic] , `consent` (true, false) [outbound traffic] (0, 1)
Appsflyer [214]	SDK documentation expects developers to not initialize the SDK before obtaining user consent [214].
Amplitude [19]	The flags `city`, `ip_address` and `lat_lng` are set to false after using a method offered by the SDK which is called `enableCoppaControl`.
Yandex [225]	`user_consent` (0, 1)
Facebook [151]	`COPPA` (true, false)
Supersonic ads/IronSource [131]	`is_coppa` (true, false), `consent` (0, 1), `gdprConsentStatus` (true, false), `do_not_sell` (true, false)
Tapjoy [205]	`cgdpr` (0, 1), `gdpr` (0, 1), `below_consent_age` (0, 1)

5.4 Results and Analysis

In our organizational survey, we asked organizations about their compliance processes and received responses from 50 organizations who collectively published more than 900 apps on Google Play [111] (72% published < 20 apps, 14% published 20–39 apps, and 14% published > 40 apps). We received 77 responses to the second survey from individuals working in 72 organizations and who varied in terms of their roles and the number of apps they were involved in developing.

Individual respondents in the second survey had roles in developing >10 apps (36%), 5–10 apps (17%), or 1–5 apps (43%), whereas the remaining 4% had no technical app development roles. As for their job functions, 78% were either software developers, test engineers, team leads, or product managers; 11% had leadership roles (e.g., CEOs); and the remaining 11% were either business/marketing employees, user-experience engineers, or customer service representatives. According to the DFF badging on Google Play [111], 56% of our interview participants and 72% of our survey respondents developed apps targeting kids who are up to 12 years of age. The rest developed apps that were not enrolled in DFF, but featured descriptions indicating the apps were directed at children.

Data Collection Procedures

In our organizational survey, we asked about the types of user data their apps collect and found that 38% of the 50 responding organizations reported none. Of the 31 organizations who reported collecting user data, they reported collecting: behavioral data about user interactions (42%), advertising IDs (35%), location data (32%), email addresses (16%), names of users (6%), IMEIs (6%), WiFi MAC addresses (3%), and other persistent identifiers (10%). Only 26% (of the 50) reported transmitting data to third parties, whereas 72% were certain that their apps did not (one was unsure). However, our app testing showed that 25% were incorrect about their apps not transmitting identifiers to third parties. The main purposes for data collection (open-ended) mentioned by the 31 respondents were:

1. Advertising and analytics:
   - serving personalized ads,
   - understanding user acquisition,
   - analyzing app usability,
   - collecting crash reports, and
   - getting insights about how to improve user engagement
2. User account creation/recovery or providing users with prior app usage data
3. Fulfilling developers' contractual obligations
4. Providing support for internal operations

Of those reporting collecting personal information, only 52% were "extremely" or "somewhat" confident that their apps always used encryption to do so (e.g., COPPA requires data be transmitted securely). In response to whether or not their privacy policies fully disclosed their apps' data collection and sharing behaviors, 92% answered affirmatively; dynamic analysis indicated that 17% of these respondents published child-directed apps that transmitted persistent identifiers to third-party domains, despite posting privacy policies that stated they do not collect children's identifiers.

We also found that certain interview participants' privacy policies either did not list all third parties receiving data from their apps, or said that an app is not directed to kids when the Play Store description indicated otherwise. We discussed these issues with participants during the interviews to understand why their privacy policies were not fully transparent. We identified a few explanations:

- one developer indicated that they offer apps directed to adults, as well, and they used the same privacy policies for their child-directed apps without modification,
- another developer mentioned that their privacy policies were copied from other developers, and
- two developers mentioned that the privacy policies were prepared by separate app publishers and they did not have visibility into their specifics.

One participant explained: *"We took it from other apps that have several million downloads and just changed the company name"* ($C4$).[4] We also observed that participants incorrectly thought that certain SDKs were removed from all their apps in previous releases.

Therefore, the takeaway from this section is that developers might not always have complete understandings of their apps' behaviors, which could affect their ability to make accurate disclosures in their privacy policies.

Consent Processes

Sixty percent of the 50 responses to the initial organizational survey indicated that their organizations believe they need to obtain parental consent before collecting data from child users. However, only 36% (of 50) reported doing so across *all* of their organizations' child-directed apps, prior to data collection (two reported doing so in *some* of their apps). In terms of *how* parental consent is obtained,

- 16% mentioned they use age gates to separate child users from adult users,
- 10% indicated they present users with knowledge-based questions that are not easy for kids to answer,
- 6% have certain features that are not available for children users,
- two ask for payment details that only a parent would be able to provide,
- one indicated that they ask all users to agree to their terms of service and/or privacy policy, and

[4] A: a respondent from the first survey, B: a respondent from the second survey, and C: interview participant.

- another indicated that they ask a parent to sign a consent form before letting their children use their apps.

We tested each organizations' apps from IP addresses in California to examine in-app consent screens, though did not observe any mechanisms for obtaining verifiable parental consent, including apps that we found sharing users' identifiers with third-party advertisers. We found that 26% of app developers hid certain features behind simple knowledge-based questions (e.g., basic arithmetic) that could likely be answered by children, whereas 6% displayed age gates that could be trivially bypassed by providing a birth year. None of these methods appear to meet the FTC's requirements for "verifiable parental consent" [62].

Despite the absence of consent screens in the apps that we examined, many developers seemed to understand the importance of obtaining parental consent:

*"...can't go into a legal agreement with a child." (A*12)

*"Because Google Play requires parental consent before collecting data from children." (A*9)

*"Most children could not understand what it means to give/deny consent to collect data. As such, if one were to collect data, parental consent would be the minimal ethical responsibility." (A*14)

Respondents who said that their organizations do not need parental consent explained:

*"Our app is not dangerous when used by children so there is no need for parental approval first." (A*34)

*"It's listed in the app that it collects data and its listed in the privacy policy it's doing so." (A*32)

*"Data collection is done by analytics SDK and it collects general data only, like location of devices." (A*23)

*"All apps in the Google Play store must be evaluated and rated before they are published." (A*22)

Only 28% of the 50 responding organizations indicated that they allow users to opt out of data collection, and 48% explicitly stated they do not. With regards to the specific opt-out mechanisms used,

- 14% indicated that they rely on the system-wide "Opt out of Ads Personalization" [106] setting,
- 10% indicated that child users are opted out by default (even though we observed that all of them had at least one app that transmitted persistent identifiers to third parties without parental consent and without having any obvious opt-out mechanism),

- 8% said they have opt-out settings within their apps,
- 4% said they use opt-out configurations offered by the SDKs integrated in their apps (we confirmed this for one developer by identifying the corresponding SDK configuration flags in their apps' traffic; however, the other developer had an app that sent advertising IDs to AppsFlyer [214] without opting child users out), and
- two said they allow users to contact them to opt out.

When running the apps corresponding to our interview participants, only a few had verifiable consent methods. When we explained what "verifiable consent" means, many participants mentioned that they were not aware of that legal requirement:

"I did not know about the thing where it makes sure that it is the parent and not the kid." ($C15$)

"actually this is very profitable for developers that it is not verified because children are clicking on ads all the time, .. and kids like this and they can click on purchase and you have a sale." ($C14$)

"...cannot even imagine how I would do it like give your consent that I can show you ads and then users immediately remove the app." ($C2$)

Some participants stated that it is the responsibility of parents to review their apps before letting their kids use them. Others thought that the app stores should be responsible for explaining to developers how to obtain verifiable parental consent:

"I don't know how the child deals with that but I think that will be the responsibility of the parent." ($C10$)

"I think that is too big of a problem for our little studio to sort it out in a meaningful way. I think the logical step should be for the app stores to be responsible for developing sure ways to do this." ($C12$)

The takeaway from this section is that many developers are not relying on verifiable methods for obtaining parental consent, and it appears that there is a general lack of awareness of this legal requirement.

Specific Privacy Laws

We report on developers' awareness of their legal obligations. We believe those who participated in our study were subject to COPPA [61] and GDPR [66], based on their apps' availability in the EU and US. Since their apps were available in California, we asked them about CCPA [30], as well, but cannot be certain that they were subject to it (e.g., we do not know their revenues).

Table 5.2: Reasons why respondents in the first survey believed that they were exempt from COPPA [61], GDPR [66], or the CCPA [30].

Reason	**COPPA**	**GDPR**	**CCPA**
SDK companies take responsibility for protecting the received data	4%	8%	8%
Users can opt out using the system-wide "Limit Ad Tracking" setting	4%	6%	12%
We don't collect data from our users	10%	4%	12%
We are covered under the internal operations exemption	4%	none	4%
Our apps are not targeted at kids	6%	6%	10%
Our apps don't target users in US, EU/UK or California (one region was displayed for each regulation)	none	4%	2%

Awareness

Only 52% of the 50 organizations in the first survey stated that they have formal processes for addressing privacy issues during the development process, whereas 42% stated that they did not (6% preferred not to answer). Regarding familiarity with specific laws, 80%, 76%, and 52% stated their organizations were familiar with COPPA, GDPR, and CCPA, respectively. This is consistent with the 82%, 78%, and 62% of the organizational survey respondents who indicated they believed their apps are required to comply with these laws.

Those claiming exemptions used a multiple-choice question to provide justifications (Table 5.2). For those who indicated that their apps did not collect user data, we tested their apps and found that all but three were correct; all three published at least one app that transmitted identifiers to advertising networks. Another 10% indicated that kids were not among their target audiences, though we found that they either had apps that were listed in the "families" section of Google Play, at least one of their apps was described as child-directed, or they responded to another question in the survey that their app(s) were designed for children.

Two respondents who claimed that data collection to support their organizations' internal operations is exempt from CCPA—an exemption that only exists for COPPA—also indicated in their responses to another question that they are not familiar with the CCPA. Three respondents also mentioned that they were not required to comply with the GDPR or CCPA, because their apps are not directed at users in the EU or California. However, in all cases, we were able to download and use their apps from IP addresses in both places.

In contrast to the organizational survey, which asked whether organizations have processes for GDPR/COPPA/CCPA compliance, the second survey asked about individual developers' awareness of these laws in their day-to-day work, as reported on a 5-point Likert

Table 5.3: Wilcoxon Signed Ranks tests results for participants' familiarity with the different privacy regulations.

Hypothesis	**(means, SDs), Z, effect size**
More familiar with COPPA than GDPR ($p = 0.189$)	COPPA(3.26, 1.081), GDPR(3.40, 1.029), $Z = 1.31$, $r = 0.10$
More familiar with COPPA than CCPA ($p < 0.001$)	COPPA(3.26, 1.081), CCPA(2.12, 1.203), $Z = 5.91$, $r = 0.47$
More familiar with COPPA than CPRA ($p < 0.001$)	COPPA(3.26, 1.081), CPRA(1.74, 1.018), $Z = 6.74$, $r = 0.54$
More familiar with GDPR than CCPA ($p < 0.001$)	GDPR(3.40, 1.029), CCPA(2.12, 1.203), $Z = 6.16$, $r = 0.49$
More familiar with GDPR than CPRA ($p < 0.001$)	GDPR(3.40, 1.029), CPRA(1.74, 1.018) $Z = 6.79$, $r = 0.54$
More familiar with CCPA than CPRA ($p < 0.001$)	CCPA(2.12, 1.203), CPRA(1.74, 1.018), $Z = 3.67$,$r = 0.29$

scale (from "expert" to "unaware"). We additionally asked about respondents' awareness of the California Privacy Rights Act (CPRA), which will replace the CCPA. To be able to evaluate respondents' confidence in their knowledge of the laws in question, we added a fictitious privacy law, "FLIRPPA," as a quality control. Among the 77 respondents, familiarity with COPPA and GDPR was higher than familiarity with CCPA and CPRA (see Figure 5.1). As expected, familiarity with FLIRPPA was lowest; however, 18 respondents claimed familiarity. Having 23% claiming familiarity with a fictitious law could demonstrate the level of uncertainty of developers in their compliance obligations, given the increasing number of laws that they are asked to comply with and that developers might not be based in the US or the EU. A Wilcoxon Signed Ranks test showed that differing levels of familiarity with the different laws were statistically significant, except when comparing COPPA to GDPR (see Table 5.3).

In the second survey, we also asked respondents to use an open-ended response to explain how they determine the privacy laws that are applicable to them:

- 36% primarily use information from Google Play [111];
- 21% said they either relied on their own online research or asked friends with more experience;
- 12% mentioned that they ignore privacy laws either because they assume that their apps are compliant or because they are small developers;
- 9% have consulted lawyers;
- 8% believed that they are required to comply with the laws applicable to the countries where their apps are available;

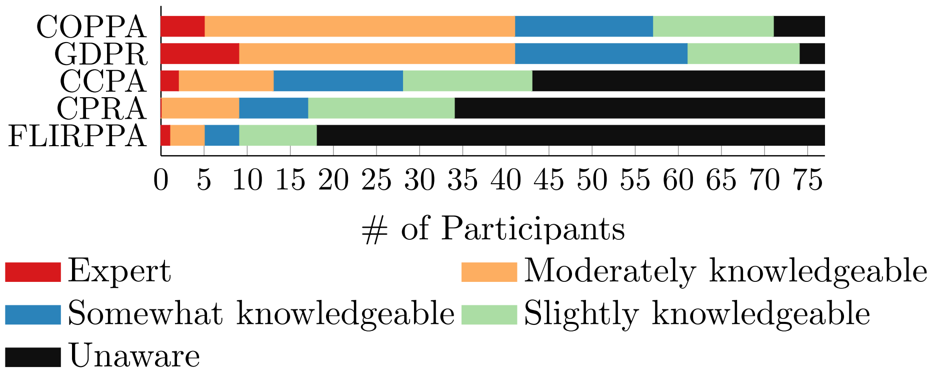


Figure 5.1: Number of participants who described themselves as experts, knowledgeable, or unaware of the various privacy laws.

- 8% stated they follow specifications from app publishers or managers in their organizations; and
- around 3% mentioned working with third-party compliance services and said they trust their advice.

We used a follow-up multiple-choice question to quantify how frequently developers consult various information sources and found that:

- 52% (of 77) rely on Google Play,
- 26% rely on other online sources,
- 16% have in-house counsel, and
- 9% consult external lawyers.

Understanding Obligations

We found that 80%, 70%, and 60% of the 50 responding organizations believed that all their apps were compliant with COPPA, GDPR, and CCPA, respectively. For each of these laws, we asked participants about their familiarity with their obligations and their organizations' compliance processes for addressing them.

When asked about COPPA requirements, open-ended responses included: not collecting personal data (42%), using parental consent screens or age gates (14%), avoiding user targeting or tracking (10%), and posting privacy policies (4%). At least 12% did not know. On a subsequent page, we explained that COPPA applies to apps available to kids under the age of 13 in the US and that developers are required to obtain verifiable parental consent before collecting certain data through these apps. After receiving this prompt, 26% of developers stated that this information changed their beliefs about their obligations. Respondents also explained that they learn about their privacy obligations via enforcement performed by app markets:

"Google routinely verify the respect of COPPA. They send notifications in case of violations, so we can fix them." (*A*15)

"Our app was pulled from the store, it was easier to change the target audience to 13 and above." (*A*22)

As for respondents' self-described GDPR obligations, the main themes were not collecting data from users (34%), obtaining consent before collecting data (8%), posting privacy policies (8%), and limiting tracking or advertising (8%). At least 16% either did not know what GDPR is or were not sure about what it required of them. As before, a subsequent page informed respondents that the children's provisions of GDPR (Article 8) apply to apps directed at kids in the EU under the age of 16. Subsequently, 30% said this information changed their beliefs about GDPR.

Finally, when we asked about CCPA, 28% said that it requires that they do not collect data, only 6% provided explanations related to the *"right-to-know"* requirements, and 4% indicated their apps should have age gates or age-appropriate ads. At least 34% either could not state any CCPA requirements or stated that they were not sure. As before, after subsequently explaining CCPA's requirements, over a third (36%) stated this information changed their understanding.

Compliance Processes

Forty-four percent (of 50 organizational survey respondents) stated that their approach to privacy compliance is to collect no data from users. Though, when we tested their apps, 27% transmitted advertising IDs to third parties, such as Unity Ads [213], Facebook [160], and Chartboost [58], and open-ended responses raised questions about developers' understandings of data collection generally:

"We never collect any data from users and we always use Admob for ads." (*A*16)

"We do not collect personal data. We disclose the use of ads SDKs, and follow their guidelines." (*A*15)

"Not sure which ones [data types] exactly collected by ad networks we have used. I hope it is only Ad identifier." (*A*2)

Consistent with prior findings, 18% rely on Google Play for guidance, with several indicating that their compliance processes consist of waiting to see if their apps are rejected:

"...when the submission time comes we fill in those questionnaires. If it turns out we violate some regulation, we go back and fix it." (*A*45)

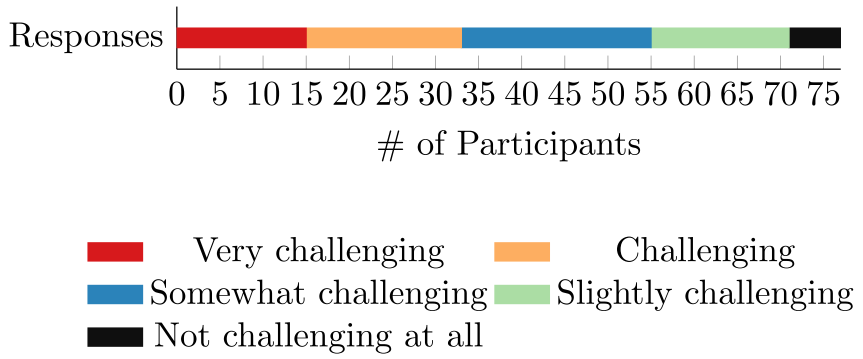


Figure 5.2: Developers' opinions on how challenging privacy compliance was for them.

"Once we have a reject from Google Play, we fix it." (A4)

As for access to legal resources, only 10% of the 50 organizations indicated that they get help from legal experts, whereas two said they relied on information from privacy-related news, conferences, or forums.

Only 8% of the 77 responses to our followup survey indicated that privacy compliance was not a challenge for them (see Figure 5.2). However, 68% of respondents stated that it is unlikely that they would be investigated by regulators if privacy issues are found in their apps, and 44% felt extremely or moderately concerned by the possibility of having their apps removed from the Play Store due to policy violations. We ran a Wilcoxon Signed Ranks test to examine whether participants were more concerned about the prospect of removal from Google Play than being investigated by privacy regulators and found the result to be statistically significant ($Z = 3.914$, $p < 0.001$, $r = 0.315$), which further highlights the major role of app stores in the privacy compliance ecosystem. As for whether they are concerned about the possibility of losing users due to publicized privacy issues or losing revenue if they avoid collecting personal data from their users, 38% and 31% were extremely or moderately concerned, respectively.

During the followup interviews, most correctly understood that they are required to comply with privacy regulations applicable to any region where their apps are available. However, they indicated that they use the store policies as their main source for guidance and trust that they will be notified if their apps have potential violations of relevant regulations. This was particularly prevalent among developers who work independently:

"I don't think that we'd think a lot about COPPA, it was just assumed that the Google store would approve and then we were okay with everything." (C15)

"...if my app is not complying with COPPA, GDPR or the Google Play policies, then my apps should not be on the store. I am not liable because Google is checking and so the responsibility is on Google." (C9)

"...I can confidently say that all my apps are compliant with all the standards because it was verified by Google before they published it." (C26)

A participant claimed their company does not have to comply with COPPA, CCPA, or GDPR because they are not based in the US or EU—even though their apps are available in both—explained, *"...we deal with Google and so Google is responsible for all the data"* (C10).

Participants who worked with publishers or within large companies said that their superiors instruct them on what privacy features to implement or SDKs to integrate, and that even though their role is to ensure that apps do not have inappropriate content, they are not involved in privacy-related discussions:

"We have our law team for that and have tons of professionals that know specific details about privacy, but I as a developer am not sure about what this process exactly is" (C18).

"To be completely honest with you, there aren't really any [process followed by the developer], it is totally handled by them [the publisher]." (C3)

Employees of large organizations said they rely on third parties, such as PRIVO [183] and kidSAFE [135], to audit their apps (e.g., recommending SDKs to remove and providing feedback on how to design consent screens and age gates):

"Having a compliance agency is important. I don't think that you'd ever have all the answers you need internally." (C23)

"...kidSAFE would do that for us. We have a yearly check with them that audits everything." (C13)

Since our sample did not include a large number of developers whose apps were approved by these safe harbor programs, however, we leave investigating the efficacy of their auditing processes to future work.

Some mentioned contacts at major app markets who they consult whenever they need advice on how to comply, one explained:

"we have a manager for our accounts at the Google store, they notify us of every policy change and give us an update on what we should do" (C17).

Others mentioned that their technical teams are separated from their privacy and legal teams, and that, if they were to be involved in privacy compliance decisions, they expect that to help them improve the privacy of their apps. One explained:

"we have around 50 people and there is only one legal guy. I think that if everyone has knowledge, then it would help" (C23).

Participants who had legal teams highlighted that these teams are mostly responsible for writing their privacy policies, and helping engineers understand how to comply with the policies of the stores. One mentioned:

"We have a legal team. Their responsibility is to comply as much as possible with current Google Play, and AppStore policies" ($C17$).

Therefore, the general takeaway from this section is that even though the majority of developers have awareness that this is a regulated area, they lack complete understanding of what their compliance obligations are and rely on external feedback to determine whether they are in compliance.

Platform Policies

Apps that are directed to children are required to participate in Google's Designed for Families (DFF) program [105]. Eighty-three percent of the 77 who responded to the second survey indicated that they were aware of the DFF program and around 57% indicated that they developed apps that participate. Several respondents said they would prefer not to participate in DFF, if given a choice, for the following reasons:

- not being able to use certain analytics or advertising services;
- having to wait for a long time before their apps get approved for inclusion;
- incurring additional costs to make apps compliant with DFF requirements;
- not having enough clarity on how to make apps compliant with DFF; and
- not being able to obtain clear enough explanations on why an app was removed from the store.

For example, some respondents mentioned:

"With DFF, the adverts don't work well (50% of our revenues lost)." ($B42$)

"Hard to follow all DFF rules. Google Play can disable app without clean explanation on what exactly was wrong." ($B62$)

"[The] main problem is the review time, Google take too long to review the app, could reach 5 to 7 days. Also due to legal requirements we can't use attribution solutions to analyze our marketing campaigns." ($B2$)

"Enrolling in the program incurs additional costs for our development process, and adds additional burden in terms of factors we have to consider, yet potentially questionable whether it has any positive impacts for us." ($B35$)

In contrast, others liked having their apps participate in DFF because they felt it provides credibility for their apps and gives them more visibility:

"The requirements are pretty simple when we already have a more strict policy." ($B30$)

"I want the store to be able to confidently share my app with kids, if participating in the DFF helps with this, then I am on board." ($B34$)

"Kids safety is very important, despite the revenue for us is dramatically low. This program should give some advantage to companies that really are taking care of the content." ($B17$)

Several app developers who are parents themselves perceived the importance of DFF:

"As a parent I think the DFF is a great idea. I wish it would be handled by an independent org." ($B45$)

"As a parent, and developer myself, I think it's great to have a separate set of rules for software companies that target under 13 users." ($B27$)

"Because as a dad of 4 year kid, I don't prefer that my child watch/play anything that is disturbing for him. I think Google policy is good for developers who want to make healthy apps for kids." ($B65$)

Our results suggest that for programs like DFF to become widely used and successful, app stores should enforce participation, while at the same time providing more useful guidance to developers.

Non-Compliance Warnings

As noted previously, developers rely on warnings from Google to inform them of privacy issues within their apps. Sixty-eight percent of the 77 respondents to our followup survey reported previously receiving these notifications, covering topics such as the following:

- age-inappropriate content,
- collection or sharing of personal data,
- selecting an inappropriate target audience when submitting apps to the store, or
- broken links to privacy policies or privacy policies lacking sufficient detail.

At the same time, 55% stated that app markets should take a greater role in evaluating mobile apps for compliance (only 5% said they should perform a smaller role) (see Figure 5.3). In our follow-up interviews, most of our participants said that they have received these notifications, and indicated challenges addressing them because they lacked detail, are received too often, include false positives that waste developers' time, and developers cannot readily obtain support from Google. One explained, *"they usually send us very standard texts like "you are not complying" and usually we have no idea about what the problem is"* ($C27$).

Two engineers working at companies that did not have dedicated teams to advise them explained:

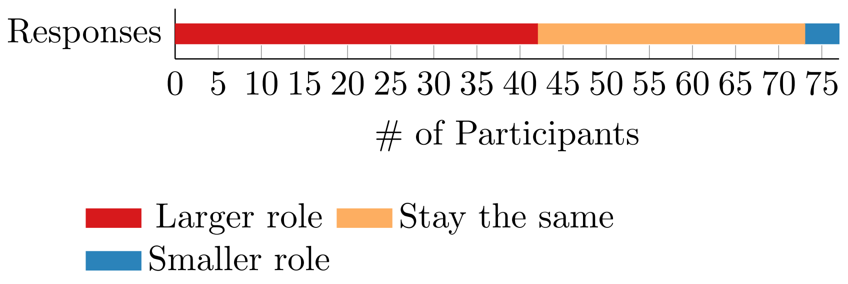


Figure 5.3: Developers' opinions on how app markets should be involved in determining apps' compliance with privacy regulations.

"I got an app rejected because the message said there was an SDK that wasn't complying, but it did not say which one, I thought let me get rid of this one and then reapply and it got rejected again, and I would remove another one and reapply and it got rejected again, and it was tiring." (C15)

"...Apps gets rejected for reasons that do not make sense actually." (C20)

Other participants complained that app stores force them to seek outside legal advice:

"If we talk about anything that has a potential COPPA implication, they say we cannot answer your question, go to your compliance agency and that's just [a] pain when you are working closely on releases" (C23).

This therefore shows that while getting app stores to notify developers about issues that exist in their apps can be a powerful tool to influence compliance rates, developers have to be provided with proper advice on what the issues are and how to fix them.

Developers' Self-Efficacy

We asked developers in our second survey to rate their confidence in their abilities to:[5]

1. identify all types of data collected by their app(s) ($\mu = 3.64$, $\sigma = 1.327$);
2. identify all third parties receiving data from their app(s) ($\mu = 3.51$, $\sigma = 1.420$);
3. comply with applicable privacy regulations ($\mu = 3.82$, $\sigma = 0.996$);
4. configure third-party SDKs to be compliant ($\mu = 3.58$, $\sigma = 1.080$);
5. identify all privacy issues their apps have ($\mu = 3.55$, $\sigma = 0.967$);
6. fix privacy issues reported to them by external parties ($\mu = 4.03$, $\sigma = 0.959$);

[5]Respondents used a 5-point Likert scale ranging from "Very Confident" (5) to "Not Confident at All" (1).

7. envision how a privacy vulnerability that exists in their app(s) can be exploited *($\mu = 3.60$, $\sigma = 1.042$)*;
8. control the kinds of data accessed or transmitted by third-party SDKs *($\mu = 3.19$, $\sigma = 1.257$)*; and
9. limit the sharing of data by SDKs with other parties *($\mu = 3.17$, $\sigma = 1.250$)*.

The mean scores for the nine items in our scale show that most survey respondents were "Somewhat Confident" or "Confident" in their abilities to handle privacy issues. However, our interviews showed that many developers were not fully aware of the privacy practices of their own apps and relied mainly on the feedback received from the stores for determining whether their apps are sufficiently compliant. Because the nine items exhibited high internal consistency (Cronbach's α=0.869), we used them as a scale ($\mu = 32.08$, $\sigma = 7.271$). Using a Mann-Whitney U test, we examined whether respondents whose apps did not transmit personal data to third-party domains scored higher on our scale than those whose apps did, but we did not find a statistically significant difference ($Z = 0.552$, $p = 0.581$, $r = 0.045$). Similarly, those participating in DFF did not score significantly higher than those who did not ($Z = 0.319$, $p = 0.750$, $r = 0.026$).

Processes for Third-Party SDKs

This section summarizes our findings on how third-party SDKs are selected for inclusion, and the extent to which developers are familiar with SDK privacy settings.

Third-Party SDK Selection

In terms of the processes used to vet third-party SDKs, 22% of the 50 who responded to our organizational survey claimed to not have SDKs in their apps (our testing showed that they were correct except for two developers who integrated the AdColony [9] and Adjust [10] SDKs, which collected users' advertising IDs), 20% said that they check the documentation about data collected by SDKs before integrating them, 10% stated that they only choose SDKs that are certified by Google [123], and 10% said they only work with trusted SDK companies:

"We only work with big name SDK data collectors e.g. Facebook, Apps Flyer - and believe they comply with the law." (*A*32)

"We don't incorporate 3rd party code because it is too difficult to be assured of their practices. Reading their privacy policies is rarely sufficient." (*A*35)

"We allow only Google SDK on our apps, which we believe they follow all regulations. But honestly, we cannot know what they do with the data." (*A*6)

Absent from most responses were indications that they understood that both Google-approved SDKs and Google's own SDKs are not necessarily compliant by default and require developers to still configure them for use in child-directed apps [104]. Only 6% mentioned using technical testing to identify the kinds of data collected by SDKs or to configure them for compliance (one of them published an app that transmitted advertising IDs to Apps-Flyer [214]). Furthermore, in our second survey, only 31% of 77 had processes to vet third-party SDKs, whereas 48% did not. When asked to explain how they decide whether to use an SDK or not in an open-ended response, several themes were identified:

- independent developers doing their own research, asking friends and deciding based on the information obtained from available resources;
- developers who used SDKs made available by big players they trust to be compliant;
- small companies making decisions based on internal discussions, delegating these decisions to their CEOs or just relying on their developers to make their best judgements; and
- medium-to-large companies delegating these tasks to a dedicated team.

They explained:

"We stick with well known players in the space." ($B4$)

"Developers suggest the SDKs. They need approval from CEO to include them." ($B51$)

"As we're a small team, we discuss things and decide together based on the information collected by developers." ($B33$)

"I decide which SDK to include, my org lacks the technical experience and has a lot of juniors." ($B3$)

In our interviews, we confirmed that developers, and particularly those who did not have dedicated teams to help, trusted that popular SDKs are legal to use. Additional factors in deciding whether to use an SDK included:

- the functionality provided;
- cost vs. expected revenue from using it;
- ease of integration; and
- availability of technical support.

Participants who had multiple teams said that such decisions are made by the app development team first, with the option to contact their legal teams later. Others said that engineers can propose an SDK, which then gets reviewed by other dedicated teams in order to approve whether to integrate it. In cases where there are different teams for different apps, each would have their own tech lead who handles SDK-related decisions. This shows that such decisions can be handled in a decentralized way, which may explain why a single organization might have apps with varying states of compliance. In our interviews, we also confirmed that well-resourced developers rely on third-party services for advice on SDK-related decisions:

"We will run our proposed implementation by PRIVO just to make sure that we are sending data in a COPPA-compliant way, and we are not sharing identifiers of kids" (C23).

Contract developers mentioned that SDKs often get chosen by their clients: *"some clients told me they want Facebook Audience and so I must integrate it"* (C10).

Those working with publishers mentioned that they are asked to integrate their publishers' SDKs, and that they need to consult publishers before integrating additional SDKs (Section 5.4). As noted earlier, many participants appreciated having strict rules set by app markets, which they believe help them identify safe SDKs for child users:

"...The stores force you to be compliant, which is great, because now you cannot mess up. They won't publish unless you are compliant" (C8).

We also investigated how frequently third-party SDKs get updated by developers and found that 88% of the 77 respondents frequently (43%) or occasionally (45%) check for SDK updates after their initial integrations. Using a Spearman's correlation test, we examined whether developers who are more confident of their abilities to control the types of data accessed by SDKs (Section 5.4) are more likely to make sure that SDKs integrated in their apps are up-to-date and found that there is a small positive correlation ($p = 0.041$, $\rho = 0.233$). In terms of releasing updates to fix privacy issues, 18% (of 77 respondents) claimed that they do so often or very often, whereas 52% rarely or never released privacy updates. We used a Mann-Whitney U test to examine whether those who claimed to release privacy updates are less likely to have sharing of personal data issues in their app(s) compared to respondents whose app(s) did not have issues; the difference was not statistically significant ($Z = 0.556$, $p = 0.578$, $r = 0.453$).

SDK Documentation

A majority of the 77 who responded to the second survey were confident or very confident that someone from their company has looked at all of their third-party SDKs' privacy policies, terms of service, quickstart guides, or configuration documentation before integrating them. At least 58% of the 77 respondents also indicated that their organizations do check that the SDKs they use comply with applicable privacy regulations and their organizations' privacy policies. Specifically, 18% mentioned that they look for information about the kinds of data collected by SDKs, 5% mentioned that they look for opt-out options and privacy flags that can be set to limit tracking children (one of them had an app that transmitted advertising IDs to Facebook without setting the corresponding privacy flag correctly), and a few others said that they use these to try to understand whether SDKs are compatible with their development frameworks. However, in our follow-up interviews, many of our participants mentioned that reading SDK documentation is not part of their process, except for a few who worked for large organizations. One explained:

"I don't read them because I rely on the fact that these SDKs are widely spread" (C4).

This is likely why many of our participants did not know that they needed to configure SDKs for compliance (Section 5.4). When we asked a participant why she did not use them, she said: *"I was expecting that they do this automatically"* (C2). Other participants mentioned that they only read parts of SDK documentation that are relevant to the app stores to make sure that their apps are not rejected. One participant justified why she does not proactively read SDK documentation:

"If I have a problem, Google asks me to read them to fix the problem and in that case, I read them" (C10).

Others mentioned that they look specifically for licensing and copyright information only, and some said that they tried reading SDK documentation, but found it difficult to understand.

SDK Misconfigurations

In terms of testing whether SDKs are configured correctly, only 5% of the 77 individual survey respondents indicated that they look at their apps' network traffic to check for configuration issues. Most respondents indicated that they do functional or user experience tests or that they do not test SDKs after their initial integrations. Few interview participants mentioned that they consider looking at the types of data collected by third-party SDKs before deciding to integrate them; all who do work at large corporations. One participant explained how they modified an SDK:

"we have access to the source code, we manually removed the IDFA-related code, and then we were able to use it" (C24).

Another explained one of their test cases: *"In order to trigger CCPA, we use a VPN to go to LA, and I see the logs from the device that are sent in real time"* (C11).

Most of the 27 developers we interviewed had at least one third-party SDK privacy misconfiguration (Section 5.3). We discussed our findings with the participants and identified a number of reasons that might have led to these misconfigurations. The majority of them did not seem to know about the SDK privacy settings related to children generally or COPPA, GDPR or CCPA in particular. Although we had participants who claimed that they knew about some, but not all, of the SDK settings they were supposed to configure, they expressed that they expected the stores to reject their apps if they had any SDK-related issues. They assumed that their apps were compliant because the stores accepted them:

"It sounds to me that this could be detected automatically by Google, like: 'Hey, you have marked your app for kids but you did not configure your SDK properly.' " (C2)

"...we didn't use these functions before because they did not force us to use them." (C10)

"When the app being uploaded to their servers is using AppsFlyer, they could easily check if there is a consent form." (C3)

Additionally, many participants indicated that the apps that had SDK configuration errors had not been updated for a long time, and that they lacked documentation to help other developers work on updating them, one mentioned:

"...they are legacy, it is hard for us to know when this code exists for I don't know how many years" (C24).

Furthermore, when we told some of our participants that we found transmissions from their apps to certain SDKs' servers, they indicated that they had stopped using these SDKs and thought that they were removed from their apps accordingly, which made them realize that they still needed to remove them. Other participants either: knew how to use some, but not all of the configurations that they were supposed to use; or used certain configurations correctly in some, but not all, of their apps. After raising the issue with them, they indicated that their teams do not have the time or resources that would allow them to keep checking whether SDK privacy options are used correctly:

"...We are a small team, we try to develop an app and then we are busy with another one, and we end up in a circle running around." (C6)

"...We try our best but sometimes apps can stay on the market for a long time and not get updated and not get deleted and not comply with the policies." (C17)

Furthermore, some participants indicated that their companies have different teams that are responsible for different apps, and that is another reason why some of their apps used SDK privacy options correctly, while they also have other apps that failed to use the same options. Others mentioned that they follow SDK integration instructions they receive from their superiors, and that privacy configurations were not part of what they received:

"The publisher has very specific instructions on how to configure SDKs and I followed them to the letter. They didn't mention this to me" (C1).

The general takeaway from this section is that vetting third-party SDKs for inclusion in child-directed apps and correctly configuring them for compliance are examples of tasks where developers could use better guidance, and that many look to app stores.

Work Models

Developers' work models have implications for how privacy-related responsibilities are handled. Some developers were fully responsible for their apps' development, release, and maintenance. Others worked with publishers who asked them to ensure that their apps are functioning correctly, by first releasing them on Google Play under their own accounts for testing purposes. Once the publishers are satisfied with how the apps perform, they acquire the apps, add consent screens, privacy policies, and any SDKs they need before releasing the apps under the publishers' accounts. Thus, apps are initially made available to users without sufficient measures to protect their privacy. One explained:

> *"the apps in our account, they are just tests. When the game gets to a stage where privacy becomes relevant, we would hand over the game to our publisher"* ($C3$).

Participants believed that this helps them become compliant, since publishers handle privacy compliance to maintain their reputations. Medium-size companies could be either startups, who have small development teams and no access to lawyers, or companies that are owned by larger companies that offer them access to their legal teams. Participants who followed this model mentioned that they have apps published under their own accounts, and they also work on a work-for-hire basis, where they develop apps and publish them under their clients' accounts. In these cases, privacy considerations could be handled by the publisher or the developer, depending on the terms of their contracts:

> *"If there are technical tasks to implement privacy, we take care of it, like permissions. Other times, we don't have any say in what they do"* ($C21$).

5.5 Discussion

Through this research we show that most developers lack a good understanding of the data collection behaviors of their own apps, and as a result, often end up violating privacy laws. Many of these issues stem from the use of third-party SDKs. While code reuse is a good and accepted practice, it adds a layer of complexity that often leads to "supply chain issues." Given developers' lack of awareness of the behaviors of various third-party SDKs, it is clear that they need access to more usable compliance-checking and auditing tools. The lack of proper guidance led developers to search for workarounds to satisfy app store requirements, such as using copied privacy policies, removing apps from DFF or delegating compliance decisions to app publishers who lacked sufficient understanding of compliance requirements. Indeed, a common theme amongst small-to-medium size organizations was *the burden of compliance* and being ill-equipped to meet it:

> *"It is overwhelming for small businesses to check all the regulations. It is not difficult to comply if you know that you ticked all the boxes, but you need to understand which boxes you need to tick."* ($C27$)

"Nobody wants not to be compliant with this stuff, but everything requires technical development which can take time and effort." (*B*6)

"It is a lot of work for small companies as we are limited in time & money for lawyers." (*A*38)

We also observed that app stores play a substantial role in how developers both learn about relevant privacy laws and assess whether their apps are in compliance. For example, the fact that Google Play currently does not check whether child-directed apps obtain "verifiable" parental consent before collecting personal data for prohibited purposes could explain why developers were not aware of that legal requirement. The lack of a rigorous auditing process by the store left developers with the belief that their apps were in compliance when:

- their methods to obtain parental consent could be bypassed by children, or
- the third-party SDKs embedded in their apps were collecting identifiers without appropriate privacy controls.

Nonetheless, developers believed that they were in compliance because they had not been told otherwise. Because participating in the DFF program was not actively enforced for child-directed apps, many developers also simply ceased participating in the program when they faced notices of noncompliance, rather than fixing the underlying issues.

Participants who also developed iOS apps found Apple's review process to be more thorough and recommended that Google adopt a similar approach. Prior work found that child-directed apps available on the iOS App Store shared data with fewer advertisers compared to Android apps [137], which could be due to differences in app review processes. App review should include a mandatory checklist that helps developers understand how to comply and provides pointers to SDK compliance options that need to be configured. While Google Play now provides a list of acceptable child-directed advertising SDKs, it does not enforce configuring them for compliance with relevant privacy laws, nor does it provide information about how to correctly do so. In the absence of this, developers appear to believe—incorrectly—that by choosing SDKs from this list, compliance is a given.

SDK providers should coordinate with app stores to enable providing timely and up-to-date advice to developers on how to exclude data collected from child users from being used for prohibited activities. This would relieve developers from having to search SDK documentation for compliance advice, which was shown to be challenging [203]. Since developers were found to keep using default settings of third-party SDKs [154], modifying these settings to be privacy-preserving (e.g., serving only contextual ads) would also help reduce the burden of compliance.

Developers require actionable guidance, but for that guidance to be productive, it has to not introduce more burden on developers by making sure that it is accurate, timely, and easy to understand. Notifying developers about issues that turn out to be false positives not only negatively impacts revenues and reputations, but might also lead developers to not act on future recommendations. Several developers stated that they wanted to be provided with guidance soon after they upload their apps to the stores, since they expressed difficulties with resolving privacy issues after having moved on to working on other apps or when some engineers are no longer available to maintain existing apps. They also preferred that app stores be more transparent about the criteria they use to audit apps so that they can adjust their processes accordingly:

"we're forced to submit our app for review, get rejected several times, and figure out what's wrong on our own" (*B*70).

App store messaging at the various stages of the app submission process should therefore be leveraged to educate developers about their compliance responsibilities. While preparing apps for submission, developers could be presented with information about their compliance status with applicable laws and educated about the potential consequences of not complying. The stores could also reward developers by including badges that show their apps' compliance status to users. Effective implementation of compliance indicators could motivate developers to work harder on improving the privacy of their apps in order to improve their reputations on the store and be assured that they are indeed in compliance.

Finally, while app developers are ultimately responsible for their apps' compliance, far greater levels of compliance could be achieved if the central app stores provided both better guidance and enforcement, as they are best positioned to do so. While we offer recommendations about how centralized app stores could offer better guidance and centralized compliance checking, future research is needed to iteratively examine *how* best to offer this guidance.

5.6 Limitations

This work was not without limitations. The most obvious of which is our reliance on self-reported data (as is the case with all surveys and interviews). While the response rate to our unsolicited emails was relatively low, it was consistent with response rates to other unsolicited surveys. Even though our sample included individual developers as well as developer organizations of various sizes, it is possible that those who chose to respond were not representative of all developers of child-directed Android apps.

To compare our sample to the broader developer population, we selected a random sample of developers of child-directed apps who did not participate in our study, and then tested one app per developer, which we selected at random. We compared these results with one app selected at random from the developers who participated. We found that 14.6% of the apps included in the random sample contained potential issues (95% CI=8.9%–22.1%), whereas we identified potential issues with 26% of study participants' apps (95% CI=18.5%–34.7%). Despite the appearance of more potential privacy issues amongst our participants' apps, the developers who participated were amongst the more popular developers of child-directed apps. This could suggest that popular apps contain more privacy issues, but a measurement study that includes a much larger sample of apps is needed to validate this relationship, which we leave to future work.

Our results should not be interpreted as suggesting prevalence of specific developer privacy misconceptions or behaviors. We believe that our sample is representative of the broader developer population, but much more insight would have been uncovered had we recruited more individuals within each of the developer organizations that participated. This was not possible, however, given how challenging it is to recruit all those who are involved in compliance processes. We promised to analyze collected data anonymously, but given that privacy compliance is considered a sensitive topic, it is likely that many developers preferred not to participate. For the same reason, it is also possible that some of those who participated could have provided inaccurate information that cannot be validated by analyzing their apps or reading their privacy policies. Our observations can therefore be used as likely explanations of how privacy compliance is being handled by developers of child-directed apps, though it remains unclear how representative these explanations are of the industry as a whole.

5.7 Conclusion

In this chapter, we presented the results of our investigation of the privacy compliance processes followed by development organizations who published child-directed apps on the Google Play Store [111]. We distributed two research surveys to these developers, conducted semi-structured interviews with 27 individuals who contributed to developing their child-directed apps, and analyzed the actual privacy behaviors of the child-directed apps they made available on the store. Overall, the results of our mixed-methods research study show that many developers were not fully aware of the data practices of their apps and trusted that Google Play [111] would notify them about privacy issues that might lead them to be in violation of applicable privacy regulations. We also found that improper use of third-party SDKs is enabling transmission of children's personal data and leading to disclosure issues in privacy policies. Despite developers' familiarity with the privacy regulations that were applicable to their apps, they did not demonstrate sufficient understanding of their privacy compliance obligations and struggled with addressing privacy issues that were reported to them by the store. These findings emphasize the need for usable tools that can reduce the burden of privacy compliance on developers by helping them take informed decisions on how to identify and address privacy compliance issues early in their development processes.

Chapter 6

The Effect of Platform Policies on App Privacy Compliance: A Large-Scale Study of Child-Directed Apps

6.1 Introduction

Regulations in both the United States [30, 61] and Europe [66] require that child-directed online services provide heightened privacy protections for their users. Investigating the effects these regulations have had on the behaviors of child-directed online services has been the subject of several prior studies (e.g., [36, 64, 65, 90, 91, 137, 150, 185, 229]). For example, in 2018, Reyes et al. [189] conducted a large-scale analysis of the network traffic generated from nearly six thousand child-directed mobile apps and showed that the data collection and sharing behaviors of 57% of them rendered them potentially in violation of the Children's Online Privacy Protection Act (COPPA), a US law that has existed for over 25 years. The identified violations were due to incorrect data transmission, consent handling, disclosure, and software configuration practices of the general developer population [44, 68, 89, 165, 215, 223, 226]. These findings have motivated regulators, such as the Federal Trade Commission (FTC) [92], to take action; this included new rulemaking efforts [63], as well as enforcement actions (e.g., [1, 2, 3]).

Industry has also taken notice: both major mobile platforms adopted new policies to help app developers comply with COPPA and other applicable privacy laws [222]. In this chapter, we examine how Google has changed its policies to improve privacy amongst child-directed Android apps [49, 50], and whether this resulted in greater levels of compliance with applicable privacy regulations. For example, all developers are now required to prepare

This work resulted in a research paper that will appear in the proceedings of the 25nd Privacy Enhancing Technologies Symposium (PETS 2025) [4, 17].

privacy labels—“data safety labels” in Google’s parlance [51]—that disclose their apps’ data handling practices, developers of child-directed apps can now only embed third-party advertising Software Development Kits (SDKs) approved by Google [123], and cannot collect the Android Advertising ID (AAID) from children [49]. Research has shown that developers’ understanding of their compliance obligations under applicable privacy regulations is influenced by the stringency of platform policy enforcement efforts [16]. While Google has made various changes to its developer policies over the past five years [50], to our knowledge, the real-world impact of these changes on the privacy-related behaviors of child-directed apps has not previously been studied. Understanding the effects of policy changes on the privacy practices of child-directed apps is essential for shaping future policy efforts aimed at regulating the practices of other types of apps.

We conducted a systematic large-scale analysis of the extent to which child-directed Android apps are potentially in violation of the Google Play Store policies [49]. We specifically focused on measuring the prevalence of privacy issues that would render developers in violation of these policies, and which can also be considered potential violations of privacy regulations (e.g., COPPA). We measured the prevalence of privacy issues related to:

- collecting or sharing personal data from children,
- making inaccurate disclosures in apps’ privacy labels [51], and
- using privacy configurations of third-party SDKs embedded in child-directed apps in potentially-violative ways (e.g., that result in user profiling and/or behavioral advertising).

We also supplement our technical analyses with the results of a follow-up developer survey to gather additional explanatory data. In this chapter, we answer the following research questions:

RQ 1 Have the updates made to the Google Play Store policies resulted in improved compliance rates among child-directed apps?
RQ 2 How prevalent are third-party SDK privacy misconfigurations among child-directed apps?
RQ 3 How prevalent are data safety label disclosure mistakes among child-directed apps?
RQ 4 Are developers giving sufficient consideration to recently-introduced privacy controls (e.g., third-party SDK privacy configurations and Google Play’s data safety labels) in their development processes?

Through our analysis of 7,377 child-directed apps tested from early February to early July of 2023 and a comparison of our results with those of Reyes et al. [189], we observe an overall improvement in the privacy practices of child-directed apps. For example, we show that the number of apps that transmitted AAIDs decreased from 59% to 8.8% and those that collected geolocation coordinates decreased from 3% to 0.1% [189].

However, we also find that developers are still struggling with understanding the behaviors of third-party SDKs, which led many of them to be unable to prepare accurate privacy labels for their apps. While the adoption of some of the third-party SDK privacy configurations has increased, there were child-directed apps that were still using third-party advertising SDKs which are not allowed by Google Play's families policies [119, 123], as well as incorrectly configuring those that are (i.e., included on Google's list of self-certified SDKs [123]).

The results of our survey support these findings: 60% did not use technical testing tools to identify the data types that need to be disclosed in their privacy labels; while 74% of respondents were familiar with third-party SDK privacy configurations, 64% stated that using them was challenging. While new platform policies have largely been effective at increasing compliance amongst child-directed mobile apps, more work needs to be done to educate developers about the responsible (and legal) use of third-party SDKs *vis-à-vis* relevant privacy regulations.

6.2 Background

This section describes the Google Play policies [177] that are applicable to child-directed apps and briefly summarizes the kids provisions of COPPA [61], GDPR [66], and the CCPA [30]. The latter is included to demonstrate how the privacy issues that we investigated would be considered potential violations of platform policies as well as privacy regulations.

COPPA, GDPR, and CCPA

The Children's Online Privacy Protection Act (COPPA) [61] regulates the practices of online services used by users under the age of 13 in the US. It applies to services that have actual knowledge of their use by child users or use any type of content that can attract them [62]. Depending on the purpose of collecting personal data, COPPA might require obtaining verifiable parental consent prior to doing so (e.g., opting in to behavioral advertising) [61, 62].

Services operating in the US targeting child users in California are additionally required to comply with the kids provisions of the California Consumer Privacy Act (CCPA) [30]. The sale of personal information collected from those under 13 is considered a violation of these provisions if parental consent is not obtained [30].

Similarly, the General Data Protection Regulation (GDPR) [66] has provisions that aim to protect the privacy of users below 16 years of age in the EU. According to Article 6 of the GDPR, online services collecting personal data from users in the EU are required to have one of the six legal bases defined by the GDPR for data processing, which include fulfilling legal obligations, serving legitimate interests, and having user consent [66]. For services that are targeted at child users, the GDPR has additional provisions in Article 8 that state that user consent cannot be relied upon as a legal basis unless services are able to verify that they obtained it from guardians [66].

These three privacy regulations have common disclosure and consent requirements that apply to child-directed apps, which are also reflected in Google Play's families policies [49]. For instance, obtaining affirmative parental consent before transmitting personal data to third parties for certain purposes is required under COPPA [61], can be used as a basis for data processing under the GDPR [66], and can be used to justify selling children's data to third parties under the CCPA [30].

Google Play Store Policies

Publishing apps on Google Play is subject to security and privacy policies [50, 177], which have been in alignment with the requirements of privacy regulations. While research has shown that they are insufficiently enforced [16, 189], developers cannot submit their apps to the store without:

- certifying that they are compliant,
- indicating their apps' target audiences,
- providing links to their privacy policies, and
- disclosing whether they use advertisements [120, 121, 176].

Google Play extended its policies in 2022 to also require making certain disclosures about apps' data practices in the form of privacy labels [51] (Section 6.3). While it is unclear whether and how Google Play checks the accuracy of these disclosures, Google Play still considers making inaccurate disclosures a "deceptive" practice [47, 53].

Google Play also requires developers to disclose their apps' data collection and sharing behaviors in their privacy policies, ensure that users' explicit consent is obtained before transmitting their personal data (for certain uses), and aviod insecure communications [46, 118]. The policies also regulate the use of persistent and resettable identifiers that allow tracking of users (e.g., AAIDs, BSSIDs and IMEIs) [117, 119]. For advertising, developers can only use the AAID, which can be reset using system settings [117]. For it to preserve this property, Google Play prohibits "linking" it to non-resettable identifiers and requires obtaining consent before transmitting it with other types of personal data [45, 46, 54].

Google Play's Designed for Families (DFF) program introduced additional policies that apply specifically to child-directed apps [49, 119, 120]. They prohibit collecting certain types of personal data (e.g., AAIDs and geolocation data) and protect access to them through Android permissions (e.g., the `AD_ID` and `ACCESS_FINE_LOCATION` permissions) [49, 119]. Alternatively, developers can use the "app set ID" (Section 6.4) for permitted purposes (e.g., analytics), which has better privacy properties compared to existing identifiers (e.g., AAIDs or IMEIs) [117, 119]. Accordingly, the families policies prohibit transmitting app set IDs alongside AAIDs (or other persistent identifiers) or using them to enable behavioral targeting [54].

The policies also restricted the use of third-party advertising SDKs in child-directed apps [48, 52, 119]. Apps whose primary audience is children are only allowed to embed specific SDKs and versions listed in Google Play's list of self-certified providers (e.g., not older than `4.0.1` for Unity Ads [85]) [123]. The SDKs included in this list ostensibly either do not perform behavioral advertising or user profiling, or include configurations that allow disabling that functionality for children. Accordingly, developers are required to correctly configure these SDKs for child-directed treatment [48, 52, 119, 123] (Section 6.3). For example, in apps that target Android versions 12 and below, where the `AD_ID` permission does not exist, use of these configurations is necessary to prevent collecting AAIDs by these third parties [35, 74].

6.3 Methodology

This section explains how we identified and tested a large corpus of child-directed apps, as well as the heuristics we used to quantify the prevalence of potential policy violations in these apps.

Identification of Child-Directed Apps

We scraped the Google Play Store to identify apps that:

- were advertised as compliant with the Google Play Store Families program or Google Play's Teacher Approved program (i.e., their app listings included the statement "Committed to follow the Play Families Policy" [49][1] or the "Teacher Approved" badge [175]); or
- had titles or descriptions that included keywords that signified that children are among their target audiences.

To do so, we identified a set of keywords that we expected to help us find child-directed apps on the store and then searched for them in apps' titles and descriptions. These are: "kid," "baby," "babies," "preschool," "school," "ABC," "kindergarten," "first grade," "second grade," "third grade," "fourth grade," "fifth grade," "coloring," "learn," "spelling," "child," "children," "toddler," "alphabet," and "math." We chose them based on an initial analysis of the titles and descriptions of a subset of child-directed apps that we identified through manual searches.

Using the previously-described process, we identified 11,490 apps that appeared to be child directed. Since our keyword searches could have resulted in false positives, as a sanity check, we manually examined the results and excluded apps that did not appear to be child directed. This review allowed further confirming that the apps are child-directed based on information communicated in their titles, descriptions, Google Play badges, or screenshots that featured characters that were likely to be attractive to children. We then crawled the Google Play Store over two different periods of time in 2023 to collect two complementary

[1]We considered these apps as enrolled in Google Play's Designed for Families (DFF) program.

datasets containing the identified child-directed apps that we queued for testing and their associated metadata (Section 6.3).

Testing of Child-Directed Apps

We used dynamic app testing methods established in the literature to capture network traffic data and decode transmissions exchanged between the tested apps and remote end-points [16, 102, 103, 115, 149, 189, 190]. Each app was run for 10 minutes on a custom version of Android and fed auto-generated input by Android's Application Exerciser Monkey [80]. To obtain visibility into encrypted traffic, we instrumented the functions that the tested apps used to read or write to TLS sockets, which allowed observing data before encryption and after decryption.

We then searched for encodings and permutations of personal information (e.g., AAIDs, Android IDs/SSAIDs, app set IDs, Firebase installation identifiers [FIDs], and geolocation coordinates) in the network traffic that we were able to observe in plaintext. We additionally used regular expressions to find privacy signals in communications exchanged with third-party SDK servers (Section 6.3). To perform static analysis, we relied on Android tools [75], such as the Android Debug Bridge (ADB) [72] and the Android Asset Packaging Tool (AAPT) [71]), to identify the Android versions targeted by the apps and the permissions they requested (Section 6.4).

As a pilot, we queued the apps identified in Section 6.3 for dynamic testing from early February to early March of 2023 using Google Pixel 3a phones running Android 12 in California. We then worked on improving our instrumentation to fix some issues that we detected in the pilot study, and added support to detect the app set ID [76] and additional types of personal data. We then used that improved instrumentation for a second round of testing. Of the apps that we initially identified as being child directed, many apps could not be installed during testing. We determined that this was due to a combination of factors: some apps were not available in our region, some were paid, some had been removed from the store during the period between identifying child-directed apps by scraping the Play Store website and queueing them for testing a few months later, and others were not compiled for our particular hardware configuration.[2]

From mid-May to early July of 2023, we re-queued all of the previous apps for testing and added 302 additional apps that we subsequently identified using the same methodology. However, in addition to many apps not being available to download for the reasons previously described, we also found that not all of the apps that we initially tested were still available.

[2]While Android 13 was the most recent version of Android available at the time of testing, we wanted to examine a prior version that did not regulate access to the AAID via the permissions system, allowing us to examine whether SDK privacy settings were being correctly configured by app developers. This still allowed us to examine whether apps targeting Android 13 were using the new `AD_ID` permission by performing static analysis on the `AndroidManifest.xml` file. More importantly, at the time of our testing, 85% of Android users were using a version older than 13 [201], and therefore testing Android 13 would not necessarily be representative.

Table 6.1: Breakdown of the apps tested during the two testing periods.

	First dataset	**Second dataset**	**Union of two datasets**
# of tested child-directed apps	6,797	4,975	7,377
# of DFF-badged apps [49]	3,095 (45.5%)	3,085 (62.0%)	3,684 (49.9%)
# of teacher-approved apps [175]	1,334 (19.6%)	983 (19.8%)	1,590 (21.6%)
# of apps whose titles indicated that they are targeting kids	2,520 (37.1%)	1,940 (39.0%)	2,776 (37.6%)
# of apps whose descriptions indicated that they are targeting kids	6,044 (89.0%)	4,372 (87.9%)	6,521 (88.4%)

This resulted in reducing the number of apps that we were able to successfully test in the later round of testing. For this reason, we decided to also report on the pilot results, when appropriate. Table 6.1 provides more details on the number of apps successfully tested in both rounds of testing.

We tested a total of 7,377 unique apps released by 3,390 unique developers, cumulatively installed by 11.5 billion devices at the time of testing. Sixty-eight percent of the 7,377 apps were categorized as educational apps by the store. After comparing the two datasets, we found that 4,395 apps were tested in both testing rounds, 2,402 were tested only in the initial pilot, and 580 apps were only tested in the second testing round. We revisited the apps that we were not able to install in our second round of testing and found that this could have resulted from Google enforcing a new policy that prevented apps targeting Android versions older than 11 (API levels below 30) from being installed on devices running more recent Android versions [38, 55, 79, 122].

The difference between the number of apps successfully tested in the two runs could also be explained by the existence of apps that were not available on the store at the time of testing, or apps that their developers converted them from free to paid apps or made them no longer available for use from California. Figure 6.1 provides a timeline showing when our datasets where collected and when relevant policies were introduced (more details are included in Appendix D.5).

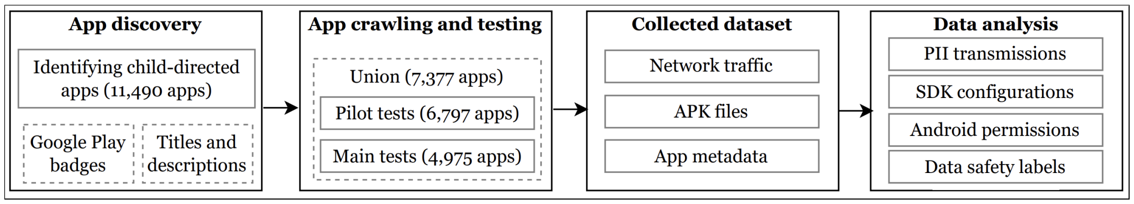


Figure 6.2: Overview of our data collection approach.

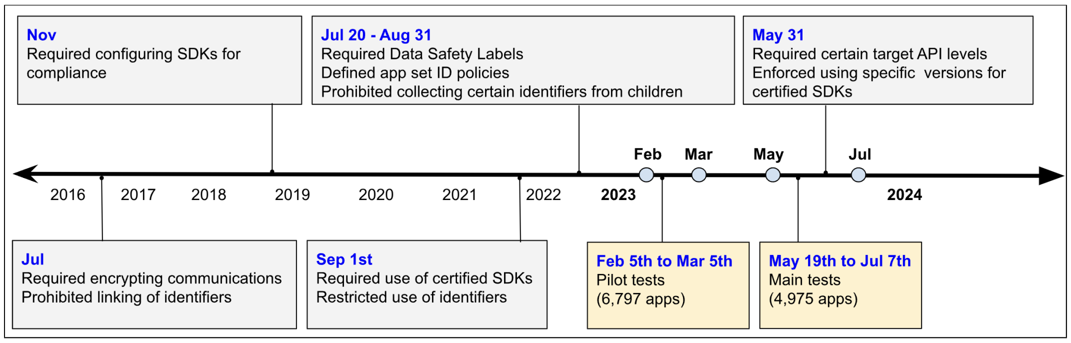

Nov
Required configuring SDKs for compliance
Jul 20 - Aug 31
Required Data Safety Labels
Defined app set ID policies
Prohibited collecting certain identifiers from children
May 31
Required certain target API levels
Enforced using specific versions for certified SDKs
Feb
Mar
May
Jul
2016
2017
2018
2019
2020
2021
2022
2023
2024
Jul
Required encrypting communications
Prohibited linking of identifiers
Sep 1st
Required use of certified SDKs
Restricted use of identifiers
Feb 5th to Mar 5th
Pilot tests
(6,797 apps)
May 19th to Jul 7th
Main tests
(4,975 apps)

Figure 6.1: Timeline showing when relevant Google Play's policies [177] were introduced.

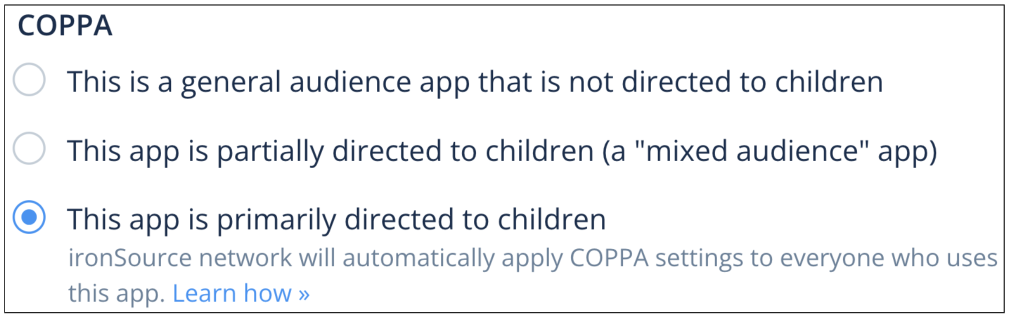
In Section 6.4, we report on all 7,377 unique apps tested, while considering the most recent versions of apps that we tested in both testing rounds (when two different versions of the same app were tested). We also indicate when certain analyses apply to only one dataset (e.g., developers' adoption of the app set ID, which was only measured during the second round). Whenever possible, we additionally compare our results to those reported by Reyes et al. [189] to understand the how the Google Play policies that were introduced since 2018 impacted the privacy practices of child-directed apps. We chose Reyes et al. [189] as a basis for our temporal comparisons since we targeted the same population of apps and employed similar app testing methods. Figure 6.2 summarizes our data collection approach.

Testing of Privacy Configurations

Google Play requires developers of child-directed apps to configure third-party SDKs for compliance with applicable privacy regulations [48, 123]. Privacy configurations offered by third-party SDKs can help developers signal to data recipients that:

- personal data was collected from a child user,
- it was collected from a region where the provisions of a specific privacy regulation apply, or that
- a user has not consented to using their data for certain purposes.

Chartboost [82], for example, provides a client-side method, `addDataUseConsent`, which can help developers comply with COPPA, GDPR, or the CCPA by setting three corresponding privacy flags (`coppa`, `pidatauseconsent` and `us_privacy`) in outbound transmissions.[3] Figure 6.3 provides an example of a server-side configuration provided by ironSource [131].


COPPA

( ) This is a general audience app that is not directed to children

( ) This app is partially directed to children (a "mixed audience" app)

(●) This app is primarily directed to children
ironSource network will automatically apply COPPA settings to everyone who uses this app. Learn how »

Figure 6.3: ironSource's [131] server-side COPPA control.

We tested the privacy configurations offered by a number of third-party SDKs that were either on Google's list of self-certified advertising SDKs [123] or were found to be used in child-directed apps in prior work (e.g., Meta's analytics and advertising SDKs [160]) [189]. We examined whether their terms of service allowed integrating them in child-directed apps, and for those that did, we identified the privacy controls that their documentation ask app developers to use. We then integrated these SDKs in prototype Android apps we created and used each of the identified privacy configurations to observe how their usage manifested in network traffic. This allowed us to:

- understand how SDK providers expect their SDKs to be configured in child-directed apps (including mixed-audience apps),
- extract the privacy flags that get included in inbound[4] or outbound transmissions as a result of using privacy configurations,
- build an understanding of how each possible flag is set,
- identify the correct values that developers are supposed to use, and
- build a corpus of API endpoints corresponding to all of the tested SDKs.

We also investigated the default values of each identified flag, the interactions between different privacy flags (e.g., setting `coppa` to `true` resulted in changing `consent` to `false` in some of our tests), and the specific configuration usages that would prevent collecting certain types of data (e.g., AAIDs).

We then searched for the identified privacy flags within the captured network traffic of our tested child-directed apps. We complemented our traffic searches with Frida-based [184]

[3]According to Chartboost's developer documentation [82], developers can disable targeted advertising by setting the values of `coppa` to "`true`," `pidatauseconsent` to "`0`," or `us_privacy` to "`1NY-`."

[4]In some cases, inbound transmissions reveal server-side privacy configurations.

instrumentation to monitor the invocation of client-side SDK privacy methods and their parameters by apps. For example, developers of apps using ironSource's advertising SDK [131] can use the `setMetaData` method to set a number of privacy flags (e.g., correctly set `is_child_directed` to `true`). Our approach enabled detecting when privacy configurations were used by different third-party SDKs integrated in the apps we tested and not necessarily called by first-party code (e.g., in case a developer is using a number of ad networks through ad mediation).

We also explored whether the design of privacy configurations impacted developers' level of adoption and enabled transmitting personal data from children (Section 6.4). For example, we found third-party SDKs that transmitted AAIDs to their servers, even when privacy configurations were correctly used by app developers. While the personal data included in these transmissions were flagged as collected from children, which signals to recipients that they should not be used for behavioral advertising, the mere collection of unused personal data may be at odds with the data minimization requirements of privacy regulations, such as GDPR [166], as well as Google Play's policies [119].

Analysis of Data Safety Labels

Since late July 2022, the Google Play Store has required disclosing apps' privacy practices in standardized privacy labels [51]. To do so, developers fill out a questionnaire that asks them to provide details about their apps' data transmission and handling practices before they submit their apps to the store [51, 176]. Using the questionnaire, they can choose from a set of pre-defined data types (e.g., "Device IDs or other IDs," "Location," and "Personal Info") and purposes of data collection or sharing (e.g., "App functionality" and "Advertising or marketing"), which then get disclosed as part of their app listings [51]. As part of this process, they are also expected to follow a specific criteria defined by Google Play to indicate whether each disclosed data type is *collected* or *shared*,[5] and whether encryption is used for communication security [51]. Examples are provided in Appendix D.4.

We evaluated whether developers of child-directed apps are able to prepare accurate privacy labels. To do so, the labels for the apps in our corpus were scraped at the time of testing. We then inspected them to understand whether the personal data that we observed being collected or shared were accurately disclosed. For that, we identified the domain names that we observed receiving personal information from the apps that we had tested to identify the recipient, and then reasoned about the data collection purpose by examining their website, API documentation, and contents of the transmission. For apps that did not use TLS in their transmissions that contained personal data, we also examined whether doing so was accurately disclosed in their privacy labels.

We followed three heuristics to obtain a lower bound for the number of apps that had disclosure mistakes in their data safety labels. First, we looked for whether each transmitted

[5]Google Play's specification for data safety labels details the circumstances under which a data transmission should disclosed as *collection* or *sharing* (e.g., data types transmitted to analytics services should be disclosed as *collected* data types since such services are considered service providers) [51].

data type was disclosed in apps' privacy labels, regardless of whether the data types were disclosed as *collected* or *shared*. Second, we considered "Analytics", "Personalization" and "Advertising" as similar purposes because a number of third-party recipients' policies state they will use received data for more than one of these purposes (e.g., Unity Ads [84, 206], ironSource [127, 130], Start.io [200] and Meta [152, 160]). Third, we looked for whether a purpose of data collection and/or sharing was disclosed at all, regardless of whether it was disclosed for the specific data types that we observed being transmitted.

6.4 Results

In this section, we measure the prevalence of various privacy issues across child-directed apps. Overall, we observe a decrease in the number of apps that transmitted personal data to first- or third-party recipients or did not use TLS. However, we also show that the use of third-party SDKs is still contributing to other transmission and disclosure issues.

Access and Collection of Personal Data

We observed that 828 (11.2%) of the 7,377 apps transmitted AAIDs, geolocation data, WiFi MAC addresses, router MAC addresses, router SSIDs, names, or email addresses to first- or third-party recipients. Of these apps transmitting personal data, 44% (363 apps) were DFF-badged and 8% were teacher-approved. Across the two testing periods, 172 apps (21% of 828 observed transmitting personal data) released new versions which we observed no longer transmitting certain types of personal data during the second testing round. Of the 657 apps that continued to transmit personal data during the second testing round, 39% were DFF-badged and 8.5% were teacher-approved.

The transmissions that contained personal data were either initiated by the tested apps or system processes, such as Google Mobile Services (GMS) [20] and Google Services Framework (GSF), which can be called by apps to request certain functionality. We also observed GMS and GFS transmit AAIDs, SSAIDs, and FIDs [77] to `app-measurement.com` alongside app package names,[6] which allows Google to gather analytics data at scale. Additionally, 368 apps transmitted FIDs alongside AAIDs to `app-measurement.com`, which allows linking these two identifiers (Figure D.3). Of these 368 apps, 234 (63.6%) were advertised as compliant with DFF; for 284 of these apps (77%), Google was the only recipient of personal data.

Table 6.2 provides details on the number of apps that transmitted each type of personal data captured by our instrumentation. It also illustrates how these numbers change after excluding the apps that ceased transmitting the same types of personal data in their subsequent versions. Figure 6.4 shows the top 20 recipients of personal data (excluding

[6]Each Android app is uniquely identified by its package name, which also allows it to be located in the Google Play Store.

Table 6.2: Number of unique apps that transmitted personal data across all the tests.

	Data type	**# of apps [all app versions] (N=7,377)**	**# of apps [most recent app versions only] (N=7,377)**	**Reyes et al. [189] (N=5,855)**
Apps	Android Advertising IDs (AAIDs)	824 apps (11.2%)	652 apps (8.8%)	3454 apps (59%)
	Android IDs (SSAIDs)	720 apps (9.8%)	611 apps (8.3%)	NA
	Geolocation data (excluding IP location)	7 apps (0.09%)	7 apps (0.09%)	184 apps (3%)
	WiFi MAC	2 apps (0.03%)	1 app (0.01%)	NA
	Router BSSIDs	2 apps (0.03%)	2 apps (0.03%)	101 apps (2%)
	Router SSIDs	2 apps (0.03%)	2 apps (0.03%)	148 apps (2.5%)
	Names	1 app (0.01%)	1 app (0.01%)	NA
	E-mail addresses	1 app (0.01%)	1 app (0.01%)	107 apps (1.8%)
	Phone numbers	none	none	10 apps (0.17%)
	App set IDs (N=4,975)	562 apps (11.3%)	562 apps (11.3%)	NA
	Firebase installation ID (FIDs)	2115 apps (29%)	2080 (28.2%)	NA
GMS/GSF	AAIDs	702 apps (9.5%)	666 apps (9%)	NA
	SSAIDs	8 apps (0.11%)	7 apps (0.09%)	NA
	FIDs	579 apps (8%)	556 apps (7.5%)	NA

SSAIDs, app set IDs, and FIDs)[7] across the 7,377 apps. Other less common recipients include Umeng [212] (8 apps), OneSignal [171] (7 apps), Singular [194] (7 apps), Applifier (6 apps), and Kochava [192] (4 apps).

In the following sections, we summarize our results for each type of personal data and compare our findings with those of Reyes et al. [189], to directly compare how COPPA compliance has likely changed over the intervening years.

Identifiers

Google Play lists a number of identifiers that child-directed apps are not allowed to collect [119]. Our results show that many child-directed apps collected AAIDs and two collected WiFi MAC addresses, both of which are prohibited because they allow long-term tracking of child users (see Table 6.2).

AAIDs. More than 11% of the 7,377 apps transmitted AAIDs, 172 of which released updated versions that stopped transmitting AAIDs in the second round of testing. Compared to the analysis conducted on 5,855 DFF apps by Reyes et al. [189], our results show a noticeable decrease in apps that transmitted AAIDs. While 59% of the apps analyzed by Reyes et al. [189] did so, our results show that this percentage decreased to 8.8% (652 apps).

[7]We excluded these identifiers because they are unique to each app installation, and therefore do not directly allow for users to be tracked across apps and services.

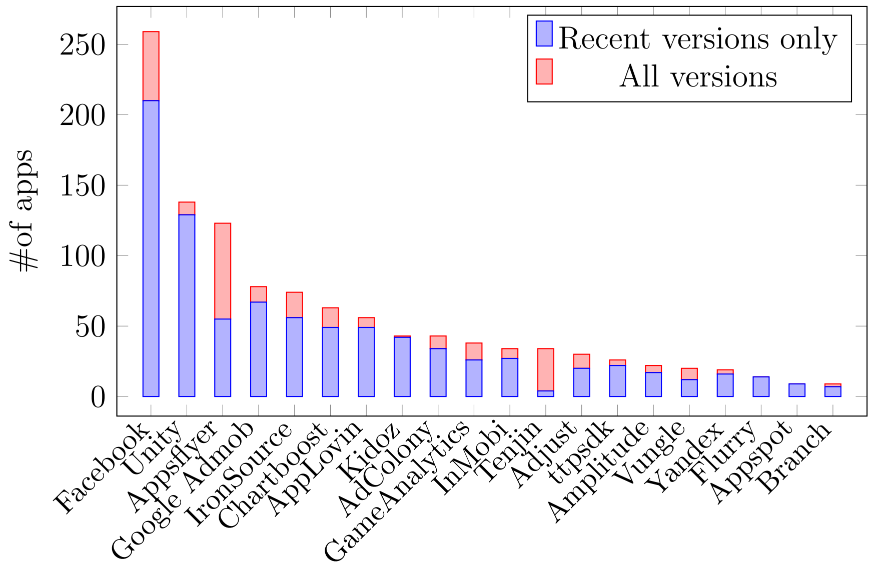


Figure 6.4: Top 20 data recipients across all app tests.

In recent versions of Android (13 or later), AAIDs cannot be accessed unless the `AD_ID` permission is declared [117, 191]. To examine whether apps are able to use AAIDs, we statically analyzed apps' `APK` files to identify apps that either targeted older Android versions (i.e., `targetSdkVersion` below 33) or declared the `AD_ID` permission when targeting recent Android versions. Of the 7,377 apps, 1,460 (20%) targeted Android 13 or 14, whereas the rest targeted older versions that allow accessing AAIDs by default. Furthermore, 578 of the 1,460 apps did not target Android 13 or 14 in our pilot tests, but updated their apps to do so in our recent tests. Similarly, 341 of the apps that targeted Android versions below 33 updated to Android 12 (i.e., `targetSdkVersion` 31 or 32) and 16 apps downgraded from Android 13 to a lower version, therefore they were still able to access AAIDs without declaring the permission.

Of the 1,460 apps, 617 (42%) declared the `AD_ID` permission. However, the permission was also declared in 1,284 of the apps that targeted Android versions older than 13 (i.e., the permission is not needed for accessing AAIDs in these versions). This could be explained by having:

- third-party SDKs that declare the permission embedded in these apps, or
- developers mistakenly declaring the permission in their apps when following third-party SDK integration instructions targeted at those whose apps target Android 13.

We also found 257 apps that added the `AD_ID` permission in their recent versions and 124 apps that updated their apps to remove it. Of the 652 apps that transmitted AAIDs (see Table 6.2), only 84 apps (13%) needed to use the `AD_ID` permission because they targeted

Android 13, whereas the rest (568 apps) targeted older Android versions. Of the 1901 apps that declared the `AD_ID` permission, 70% were DFF-badged at the time of testing.

WiFi MAC addresses. Only 2 apps transmitted WiFi MAC addresses, one of which did not do so in its updated version. Both of them were not DFF-badged. One of them shared AAIDs, SSAIDs and WiFi MAC addresses with Umeng [212], enabling tracking of children by linking different AAIDs that belong to the same device or SSAIDs associated with different app developers (Figure D.4).

SSAIDs. Android IDs were transmitted by 720 apps, 109 of which stopped doing so in their recent versions. While SSAIDs cannot be relied upon for tracking users across apps released by different developers since the release of Android 8 in 2017, they can still allow tracking, as their values do not get reset after apps get uninstalled or updated but rather only due to a factory reset of the device [70]. Additionally, transmitting SSAIDs alongside certain types of personal data to the same recipients is considered a violation of Google's policies [45]. Across all the apps we tested, SSAIDs were transmitted alongside AAIDs (168 apps), FIDs (26 apps), WiFi MAC addresses (2 apps), and router SSIDs (1 app) to first- and third-party recipients such as Unity3d [206], Chartboost [58], and Yandex [225]. In total, 191 apps included at least one type of personal data with SSAIDs in the same transmissions to domains other than `app-measurement.com`.

AAIDs were transmitted alongside FIDs (13 apps), fine location coordinates (3 apps), WiFi MAC addresses (1 app), router BSSID (1 app), or IP location (1 app) to the same recipients. These included Amplitude [19], AppsFlyer [214], Yandex [225], and Umeng [212]. Thus, the total number of apps that transmitted AAIDs alongside location data (including router MAC addresses), FIDs, SSAIDs, or WiFi MAC addresses to domains other than `app-measurement.com` is 179 (2.4% of 7,377). Sharing these data types alongside AAIDs allows converting resettable AAIDs to persistent identifiers that can be used for long-term tracking, as it allows recipients to link AAIDs that belong to the same device after they have been reset. However, Reyes et al. [189] showed that more than 66% of the 5,855 apps they tested shared a persistent identifier (e.g., IMEIs or SSAIDs in Android versions that are older than 8) alongside AAIDs. This shows a drastic decrease in the number of apps that are attempting to link AAIDs to other types of personal data (RQ1), particularly those that might persist over time (e.g., router MAC addresses or fine location coordinates).

Location Data

Seven apps transmitted geolocation coordinates, three of which transmitted approximate geolocation coordinates only whereas the rest transmitted exact geolocation coordinates. Additionally, we identified another 34 apps that transmitted or received IP location (e.g., using IP geolocation services such as ipwhois.io [126]), but we did not consider them as cases of data leakage since relying on IP addresses for obtaining user locations does not often lead to finding accurate user locations. However, 194 apps declared at least one of

Android's location permissions (see Table 6.3). Thus, even though only 7 apps transmitted fine or coarse location coordinates, the number of apps that had the capability of accessing children's location data exceeded that number. We also found 160 apps that potentially violated Google Play's policies by declaring the `ACCESS_FINE_LOCATION` permission, which allows accessing child users' exact locations [49].

However, our results also suggest a considerable decrease in the number of apps that have the capability of accessing location data. We observed that only 2.6% of the apps we tested declared `ACCESS_FINE_LOCATION` or `ACCESS_COARSE_LOCATION` whereas 12% of the apps tested by Reyes et al. [189] declared one of these permissions. Additionally, only 9 apps collected coarse or fine geolocation coordinates or seemed to have the capability of knowing users' approximate or exact locations using router MAC addresses or router SSIDs, whereas Reyes et al. [189] identified 256 apps that collected these types of data (see Table 6.2).

Table 6.3: Number of apps that declared Android permissions.

Permission	**# of apps (N=7377)**
`READ_EXTERNAL_STORAGE`	2793 apps (38%)
`AD_ID`	1901 apps (26%)
`CAMERA`	688 apps (9.3%)
`READ_PHONE_STATE`	605 apps (8.2%)
`ACCESS_FINE_LOCATION`	160 apps (2.2%)
`ACCESS_COARSE_LOCATION`	152 apps (2.06%)
`READ_MEDIA_IMAGES`	122 apps (1.7%)
`GET_ACCOUNTS`	116 apps (1.6%)
`READ_MEDIA_VIDEO`	103 apps (1.4%)
`READ_MEDIA_AUDIO`	94 apps (1.3%)
`READ_CONTACTS`	34 apps (0.46%)
`ACCESS_MEDIA_LOCATION`	4 apps (0.05%)
`ACCESS_BACKGROUND_LOCATION`	2 apps (0.03%)
`BODY_SENSORS`	2 apps (0.03%)
`READ_PHONE_NUMBERS`	2 apps (0.03%)
`READ_SMS`	0 apps (0%)
`BODY_SENSORS_BACKGROUND`	0 apps (0%)

Contact Information

Only one app collected names and email addresses. The app, which was DFF-badged, transmitted these two data types to (`www.googleapis.com`). None of the apps in our corpus transmitted phone numbers. We investigated the number of apps that declared `GET_ACCOUNTS` or `READ_PHONE_STATE`, which are the permissions that regulate access to names, email addresses and phone numbers. We found 605 and 116 apps that declared `READ_PHONE_STATE` and

`GET_ACCOUNTS`, respectively. This shows the actual number of apps that had the capability of accessing these data types, and also confirms the decrease in the number of apps that could access children's contact information when compared to the results of Reyes's et al. [189] investigation in 2018. While only 8% of the apps we tested declared the `READ_PHONE_STATE` permission, 30% of the apps tested by Reyes et al. [189] did so. Similarly, Reyes et al. [189] found the `GET_ACCOUNTS` permission declared in 13% of the apps they tested whereas our results show that only 1.6% of those in our corpus did so (see Table 6.3).

Thus, the main takeaway from this analysis is that while we found evidence of the decrease in the number of apps that transmitted or were able to access personal data, third parties were still able to track children (RQ1). This was likely caused by the use of third-party SDKs that triggered the transmission of device identifiers.

TLS Usage

We also investigated the extent to which the apps secure their inbound or outbound communications using the Transport Layer Security (TLS) protocol. Only 51 of the 7,377 apps contained personal data in their inbound or outbound communications that were not secured using TLS (SSAIDs: 18 apps, location data: 16 apps, AAIDs: 12 apps, FIDs: 10 apps, and WiFi MAC: 1 app). For 22 of them, personal data was transmitted to first-party recipients only, whereas the recipients of personal data for the 29 remaining apps were third parties such as InMobi [125] and Umeng [212] (two of these apps contained personal data in their unencrypted communications to first-party servers as well). For location data, only one of the 16 apps included fine GPS coordinates, whereas the remaining 15 apps only contained IP location in their unencrypted communications. Twenty-one of the 51 apps were DFF-badged whereas 6 of them were teacher-approved.

While these percentages show that Google Play still publishes apps that are not fully utilizing TLS, they also suggest that the majority of apps are doing so. Thus, finding that less than 1% of apps did not fully adopt TLS shows an increase in the adoption of TLS in the past few years (RQ1), as Reyes's et al. [189] showed that 40% of child-directed apps suffered from communication insecurity due to not using TLS.

Effects of Platform Policies

In this section, we analyze how Google Play's recent policies impacted developers' use of third-party SDKs in child-directed apps and app set IDs as an alternative to other device identifiers. We also evaluate the extent to which DFF-badged apps are compliant.

Use of Third-party SDKs

We observed third-party advertising SDKs that are not on Google's list of self-certified SDKs [123] being used in child-directed apps, including SDKs provided by Meta [160], Yandex [225], and Start.io [199]. Other apps shared personal data with providers of analytics SDKs which do not allow the use of their SDKs in child-directed apps (e.g., Flurry [94, 95]).

In the following, we analyze developers' correct and incorrect adoption of third-party SDK privacy configurations in child-directed apps (RQ2). These include server- or client-side configurations aimed at communicating to third-party SDK servers whether: (1) children are among apps' target audiences, (2) COPPA, GDPR, or CCPA provisions are applicable, (3) users provided consent to data sharing or sale of personal information, or (4) developers instructed the SDK to stop collecting certain types of personal data (e.g., AAIDs). Detailed measurements are provided in Tables 6.4, 6.5, 6.6 and 6.7[8].

Configurations for child-directed treatment. We investigated the use of COPPA-related privacy configurations in all apps that communicated or shared personal data with third-party servers contacted by the SDKs we tested (Section 6.3). Given that Google's policies prohibit collecting AAIDs from children [49], we focused on understanding whether developers correctly used privacy configurations for third-party SDKs embedded in their apps and whether correct usage of such configurations affected sharing of AAIDs collected from children with third-party servers (Section 6.3).

We found Meta's advertising SDK [160] being used in 24 apps, 20 of which transmitted AAIDs while incorrectly setting Meta's `COPPA` flag [151] to `false` and none of them set it to `true`. However, we observed higher levels of adoption of COPPA-related privacy configurations offered by a number of other third-party advertising SDKs, including ironSource [128], Unity [85], Google Admob [11], and AdColony [9]. Of the 299 apps that communicated with ironSource [128], 94% correctly set `is_coppa` or `is_child_directed` to `true`, which are based on one mandatory server-side configuration and an optional client-side configuration, respectively [131]. Of the 56 apps that transmitted AAIDs to ironSource [128] servers, 73% used at least one of these configurations correctly, 16% used them incorrectly, and 11% did not use them at all. Developers who use ironSource [128, 129] as an ad mediation platform are also offered a set of optional privacy signals that can be propagated to their selected ad provider (e.g., `Applovin_AgeRestrictedUser` for developers of child-directed apps who instruct ironSource [128, 129] to request ads from AppLovin [25]). We identified 191 apps that correctly used at least one of these configurations and none that incorrectly did so (see Table 6.4, and 6.5).

Of the 741 apps that communicated with Unity [85], 61% signaled that their apps were used by children through Unity's server-side configuration. Only 20 of these correctly-configured apps transmitted AAIDs to Unity, whereas the remaining apps that transmitted AAIDs to Unity either signaled that they were not child-directed (36 apps) or did not set a signal in their transmissions (73 apps). Sixty-five percent of apps that communicated

[8]In the last column of these tables, we demonstrate the number of apps that shared AAIDs when privacy configurations were used correctly, incorrectly or not used by using this notation [correctly configured, incorrectly configured, configuration not used].

with InMobi [56, 125] similarly signaled that COPPA applies to their apps, 15% of which transmitted AAIDs to InMobi [125].

For Google Admob [11], we monitored the use of `setTagForChildDirectedTreatment` and the related `setTagForUnderAgeOfConsent`, which are client-side configurations. The values of `tag_for_child_directed_treatment` and `tag_for_under_age_of_consent` are set based on these configurations, respectively. We found that at least one of these configurations was used correctly by 37.5% of the 3,147 apps that communicated with `doubleclick.net` or another of Google's domains, which is `fundingchoicemessages.google.com`.[9] Of the 64 apps that shared AAID with one of these endpoints, 32 apps set one or both of these signals to their correct values, 29 apps signaled that they were not child directed, and 3 apps did not include any of these signals in their communications.

For AdColony [9], we monitored the use of two signals that are set based on client-side configurations (`coppa_required` and `is_child_directed`), and a `coppa` signal which is set based on a setting on AdColony's [9] dashboard. Of the 158 apps that communicated with AdColony's endpoints, 93% correctly used at least one of these configurations. Of the 34 apps that transmitted AAIDs to AdColony, 26 apps set `coppa_required` or `coppa` to `true`, 8 apps incorrectly set one or both of these signals to `false`, and none set `is_child_directed` to `true`, which is a configuration provided by AdColony to prevent transmitting AAIDs [8]. The same configuration was correctly used by another 109 apps to disable the collection of AAIDs from children.

Similarly, we searched the communications exchanged with Vungle [219] by 203 apps to measure adoption of its `is_coppa` signal which is based on an optional client-side configuration [218], and found it to be correctly set to `true` by 82% of these apps. None of the apps that correctly used this configuration transmitted AAIDs to Vungle, whereas 5 of the apps that incorrectly used it did so. Using Chartboost's `addDataUseConsent` function [58, 82] to set a `coppa` signal to `true` in outbound transmissions had a similar effect, as none of the 39 apps that did so transmitted AAIDs to Chartboost [58]. Additionally, none of the apps that communicated with Yandex [225] or Appodeal [26] used their COPPA-related configurations correctly, even though neither of them are on Google's list of self-certified SDKs [123]. Furthermore, although AppLovin [25] is no longer included on Google's list of self-certified SDKs [123] and that integrating it in apps that are primarily child-directed is against its terms of service [24], 49 apps shared AAIDs with its servers, only 6 of which signaled that they are child-directed through a client-side configuration.

After comparing the number of apps that included COPPA-related signals set to correct values in their transmissions with those that used the signals to communicate that that they are not child-directed, we identified 72 apps that set different signals to conflicting values. They either set COPPA-related signals that belong to the same SDK to conflicting values or did so for signals that belong to different SDKs embedded in their apps. We additionally

[9] In certain cases, a signal was sent to one of these endpoints (e.g., `doubleclick.net`), but AAIDs were sent to the other (e.g., `fundingchoicemessages.google.com`). Since both of them belong to Google and Google Admob's User Messaging Platform SDK [13] communicates with `fundingchoicemessages.google.com`, we assumed that they are related.

found developers who correctly used COPPA-related signals in some of their apps, and at the same time incorrectly used the same signals in other of their child-directed apps. These cases demonstrate the difficultly developers are experiencing with configuring all SDKs correctly for child-appropriate treatment.

In 2018, Reyes et al. [189] quantified the use of COPPA-related configurations provided by Meta [160] and Unity [85]. While they found correct usage of Unity's COPPA configuration [85] in only 16.5% of the apps that used Unity, we show that this percentage increased to 61% (RQ1). Although our corpus included only 24 apps that embedded Meta's advertising SDK [160], none of these apps set its `COPPA` flag to `true`. However, Meta prohibits use of its SDK in apps that are primarily child directed; the `COPPA` flag is provided to identify child users of mixed-audience apps [151]. Therefore, this suggests an improvement in apps' privacy behaviors (i.e., a decrease in the number of apps that used Meta's advertising SDK).

Configurations for CCPA-compliance. We observed varying levels of adoption of CCPA-related configurations that instruct third parties not to sell users' personal information to other parties. Of the apps that communicated with AdColony [8], Vungle [218], and ironSource [131], 72%, 73% and 68.5% opted out children residing in California from the sale of their personal data, respectively. Related CCPA configurations offered by Unity [83, 85], Chartboost [58, 82], InMobi [56], Google Admob [11, 12], AppLovin [24], and Flurry [95] received lower adoption (see Table 6.6 and 6.7). However, some of these SDKs offered server-side configurations which did not result in transmitting discernible signals in inbound traffic, and thus our results can be considered a lower bound for the adoption of these configurations. Furthermore, it is unclear whether all of the apps that failed to correctly use these configurations were subject to compliance with the CCPA [30]. Answering this question might require estimating the sizes of the organizations that developed these apps, their revenues or the number of California residents who used their apps, which we leave to future work. Figure 6.5 shows the percentages of apps that included COPPA or CCPA flags in their communications with six of the third parties whose SDKs are included in Google's list of certified SDKs [123].

Our results also suggest the presence of signals that communicated that users consented to data sharing when they did not. However, since most of these signals were offered for GDPR compliance purposes by SDK providers and our tests were conducted from California, we leave verifying this observation to future work aimed at testing child-directed apps from the EU.

Other configurations. We investigated the adoption of other configurations that result in requesting non-personalized advertisements or preventing the transmission of personal data. Unity [85] offers a server-side option that allows disabling personalized advertising when indicating that an app is not child-directed through its server-side COPPA control. The `user.nonBehavioral` is another signal, which serves a similar purpose but can be set using a client-side configuration [85]. None of the apps that incorrectly used Unity's

Table 6.4: Adoption of COPPA-related third-party SDK privacy configuration for recent app versions (*flag value** denotes a privacy-friendly configuration).

Privacy flag or function	**Third-party SDK [Receiving domain(s)]**	**# of apps that contacted the domain(s)**	**# apps per each flag value**	**# of apps that shared AAIDs**
`COPPA`	Meta Audience Network [160] `[facebook.com]`	339	`TRUE*: 0, FALSE: 24`	[0, 20, 0]
`is_coppa`			`TRUE*: 281, FALSE: 9`	[41, 9, 6]
`is_child_directed`			`TRUE*: 227, FALSE: 4`	[6, 4, 46]
`google_family_self_certified_sdks`			`TRUE*: 0, FALSE: 0`	[0, 0, 56]
`AdColony_COPPA`			`TRUE*: 144, FALSE: 0`	[4, 0, 52]
`AdColony_APP_Child_Directed`			`TRUE*: 139, FALSE: 0`	[0, 0, 56]
`AdMob_TFCD`			`TRUE*: 176, FALSE: 0`	[4, 0, 52]
`AdMob_TFUA`	ironSource [128] `[supersonicads.com]`	299	`TRUE*: 116, FALSE: 0`	[4, 0, 52]
`Chartboost_Coppa`			`TRUE*: 20, FALSE: 0`	[0, 0, 56]
`Pangle_COPPA`			`TRUE or 1*: 70, FALSE: 0`	[0, 0, 56]
`Vungle_coppa`			`TRUE or 1*: 178, FALSE: 0`	[0, 0, 56]
`InMobi_AgeRestricted`			`TRUE*: 171, FALSE: 0`	[0, 0, 56]
`AppLovin_AgeRestrictedUser`			`TRUE*: 61, FALSE: 0`	[4, 0, 52]
`META_Mixed_Audience`			`TRUE*: 55 , FALSE: 0`	[0, 0, 56]
`coppa`			`TRUE*: 430, FALSE: 35`	[9, 33, 87]
`coppaCompliant`			`TRUE*: 453, FALSE: 38`	[20, 36, 73]
`appLevelCoppa`			`TRUE*: 455, FALSE: 37`	[20, 36, 73]
`calculatedCoppa`	Unity [85] `[unity3d.com]`	741	`TRUE*: 454, FALSE: 38`	[20, 36, 73]
`user.nonBehavioral`			`TRUE*: 13, FALSE: 14`	[0, 14, 115]
`contextualOnly`			`TRUE*: 0, FALSE: 492`	[0, 56, 73]
`coppa`	Appodeal [26] `[appbaqend.com]`	4	`TRUE*: 0, FALSE: 4`	[0, 1, 0]
`for_kids`			`TRUE*: 0, FALSE: 4`	[0, 1, 0]
`is_coppa`	Vungle [219] `[vungle.com]`	203	`TRUE*: 167, FALSE: 8`	[0, 5, 7]

Table 6.5: Adoption of COPPA-related third-party SDK privacy configuration for recent app versions (*flag value** denotes a privacy-friendly configuration) (cont.).

Privacy flag or function	**Third-party SDK [Receiving domain(s)]**	**# of apps that contacted the domain(s)**	**# apps per each flag value**	**# of apps that shared AAIDs**
`age_restricted_user`	Yandex [225] [`yandex.net`, `yandex.ru`]	40	`0: 0, 1*: 0`	[0, 0, 16]
`coppa`	InMobi [125] [`inmobi.com`]	169	`TRUE or 1*: 110, FALSE: 44`	[16, 6, 5]
`applyGdprAgeOfConsent`			`TRUE*: 108 FALSE: 45`	[16, 6, 5]
`u-age-restricted`			`1*: 112, 0: 7`	[1, 6, 20]
`coppa`	Chartboost [58] [`chartboost.com`]	91	`TRUE*: 39, FALSE: 0`	[0, 0, 49]
`coppa_required`	AdColony [9] [`adcolony.com`]	158	`TRUE*: 115, FALSE: 7`	[5, 5, 24]
`is_child_directed`			`TRUE*: 109, FALSE: 0`	[0, 0, 34]
`coppa`			`TRUE or 1*: 145, FALSE or 0: 10`	[25, 8, 1]
`tag_for_child_directed_treatment` or `tfcd`	Google Admob [11] [`doubleclick.net`, `fundingchoicesmessages.google.com`]	3147	`NaN: 44, 1*: 1159, 0: 43`	[19, 2, 43]
`tag_for_under_age_of_consent` or `tfua`			`TRUE or 1*: 390, FALSE or 0: 41, NaN: 125`	[32, 28, 4]
`is_age_restricted_user`	AppLovin [25] [`applovin.com`]	163	`TRUE*: 32, FALSE: 4`	[6, 3, 40]

COPPA server-side configuration disabled personalized advertising using any of these two configurations.

Meta's analytics SDK [152] offers a client-side configuration that prevents transmitting AAIDs, and which results in setting a privacy signal called `advertiser_id_collection_enabled` in exchanged communications. This configuration was correctly used by only 22% of the apps that embedded this SDK. Similarly, 75% of apps that communicated with ironSource [131] and 87% of those that communicated with Vungle [218] disabled collecting AAIDs from children. We additionally found Google Admob's `npa` flag correctly used by 342 apps to disable personalized advertising, 71% of which correctly used a COPPA-related signal (e.g., `tfcd`) and only 6% are of those that signaled that they are not child-directed.

The general takeaway from our analysis of third-party SDK configurations is that developers were clearly not using all the privacy configurations that were offered to them by

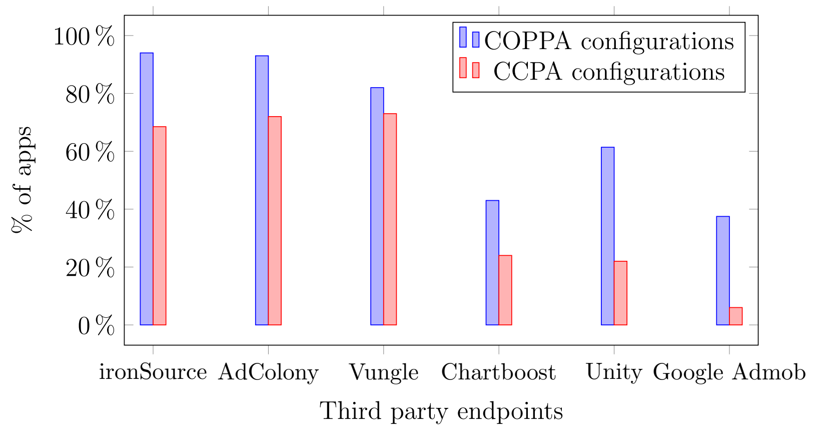


Figure 6.5: Inclusion of COPPA and CCPA flags in exchanged communications with six third-party SDK providers.

third-party SDKs, and also did not consistently use these configurations in all of their apps.

SDK versions. In addition to specifying the third-party advertising SDKs that are allowed to be embedded in child-directed apps, Google Play also specifies the minimum versions of these SDKs that meet their requirements [123]. For six of the SDKs that were included on Google's list [123], we investigated whether allowed versions were used. Our results suggest that 31% of the apps that integrated Chartboost [58], 28% of those that integrated Unity Ads [85], and 10% of those that integrated InMobi [125] used a version below the minimum version allowed [123]. The same applies to 8%, 5% and 4% of apps that integrated AdColony [9], Vungle [219], and ironSource [128], respectively.

App Set IDs

The app set ID is an identifier that Google recently introduced to allow obtaining analytics about app usage while preventing cross-device tracking or long-term tracking across apps released by different developers [37, 76, 77, 78]. Unlike AAIDs, which are the same for all apps installed on the same device until they get reset by users using system settings, app set IDs can only uniquely identify transmissions from apps released by the same developer on Google Play that are installed on the same device [76, 77, 78]. Unlike SSAIDs, which can similarly identify apps associated with a specific developer on the same device, the values of app set IDs change to new values once users uninstall all apps that share a common app set ID value, whereas changing SSAIDs requires users to factory reset their devices [73, 78]. While Google's policies allow using the app set ID for analytics purposes, they prohibit using it to target ads to child users and collecting it alongside certain other types of personal data [46, 119].

Table 6.6: Adoption of CCPA and GDPR third-party SDK privacy configurations, and configurations that can disable collecting personal data for recent app versions. (*flag value** denotes a privacy-friendly configuration)

Privacy flag or function	**Third-party SDK [Receiving domain(s)]**	**# of apps that contacted the domain(s)**	**# of apps per each flag value**	**# of apps that shared AAIDs**
`advertiser_id_collection_enabled`	Meta's event tracking SDK [152] `[facebook.com]`	339	`TRUE: 96, FALSE*: 66`	[0, 87, 113]
`DATA_PROCESSING_OPTIONS`	Meta Audience Network [160] `[facebook.com]`	339	`null: 21, LDU*: 0`	[0, 0, 20]
`DATA_PROCESSING_OPTIONS_COUNTRY`			`null: 21, 1: 0`	[0, 0, 20]
`DATA_PROCESSING_OPTIONS_COUNTRY_STATE`			`null: 21, 1000: 0`	[0, 0, 20]
`do_not_sell`	ironSource [128] `[supersonicads.com]`	299	`TRUE*: 205, FALSE: 4`	[7, 4, 45]
`metadata_consent`			`FALSE or 0: 102, TRUE or 1: 6`	[6, 4, 46]
`is_deviceid_optout`			`TRUE*: 224, FALSE: 0`	[4, 0, 52]
`gdpr.consent`	Unity [85] `[unity3d.com]`	741	`TRUE: 9, FALSE: 86`	[4, 3, 122]
`privacy.consent`			`TRUE: 16, FALSE*: 163`	[1, 8, 120]
`disable_ad_id`	Vungle [219] `[vungle.com]`	203	`TRUE*: 177, FALSE: 11`	[0, 10, 2]
`user.ccpa.status`			`opted_in: 49, opted_out*: 149`	[0, 12, 0]
`user.gdpr.consent_status`			`opted_in: 4, opted_out: 57, unknown: 142`	[1, 3, 8]
`user_consent`	Yandex [225] `[yandex.net, yandex.ru]`	40	`0: none, 1: 1`	[0, 1, 15]

Table 6.7: Adoption of CCPA and GDPR third-party SDK privacy configurations, and configurations that can disable collecting personal data for recent app versions. (*flag value** denotes a privacy-friendly configuration) (cont.)

Privacy flag or function	**Third-party SDK [Receiving domain(s)]**	**# of apps that contacted the domain(s)**	**# of apps per each flag value**	**# of apps that shared AAIDs**
`fl.ccpa.optout`	Flurry [95] [`flurry.com`]	34	`TRUE*: 0 , FALSE: 10`	[0, 0, 14]
`fl.report.location.enabled`			`TRUE: 7, FALSE*: 5`	[NA, NA, NA]
`gdpr_consent_available`	InMobi [125] [`inmobi.com`]	169	`TRUE: 2, FALSE: 43`	[4, 2, 21]
`do_not_sell`			`1: 57, 0: 0`	[0, 0, 27]
`pidatauseconsent`	Chartboost [58] [`chartboost.com`]	91	`0: 4, -1: 74 , 1: 4`	[2, 4, 43]
`us_privacy`			`1YY-*: 21 , 1NY-*: 1 , 1NN-: 3, 1YN-: 2 , NULL: 26`	[1, 3, 45]
`gdpr_required`	AdColony [9] [`adcolony.com`]	158	`TRUE: 28, FALSE: 58`	[3, 6, 25]
`gdpr_consent_string`			`1: 5 , 0: 25`	[3, 3, 28]
`ccpa_required`			`TRUE*: 117, FALSE: 2`	[8, 0, 26]
`ccpa_consent_string`			`1: 6 , 0: 113`	[4, 4, 26]
`npa`	Google Admob [11] [`doubleclick.net`, `fundingchoicesmessages.google.com`]	3147	`1*: 342, 0: 27`	[6, 2, 56]
`rdp`			`1*: 188, 0: 13`	[0, 0, 64]
`has_user_consent`	AppLovin [25] [`applovin.com`]	163	`TRUE: 11, FALSE: 13`	[4, 3, 42]
`is_do_not_sell`			`TRUE*: 26, FALSE: 2`	[3, 0, 46]

Of the 4,975 child-directed apps that we tested in the second app runs, 11.3% of them (562 apps) transmitted app set IDs. However, 30% of these 562 apps generated outbound transmissions that shared app set IDs and AAIDs (107 apps) (see Figures D.1 and D.2), SSAIDs (67 apps), or FIDs (1 app) with the same data recipients. The top data recipients in these cases were third-party SDK providers, such as SupersonicAds [128] (81 apps), AdColony [9] (30 apps), InMobi [125] (26 apps), AppLovin [25] (23 apps), and Chartboost [58] (16 apps). While 69% of the 562 apps (389 apps) did not include additional identifiers in transmissions that contained app set IDs, 12% of these 389 apps (46 apps) did not consistently use app set IDs as replacement to other identifiers in all their transmissions since they included AAIDs or SSAIDs in other transmissions.

These findings show that while apps were starting to use app set IDs as an alternative to other device identifiers, not all apps were utilizing its privacy properties (i.e., many apps gave third parties the ability to link app set IDs with other personal data).

Designed for Families Badge

While complying with Google's families policies is mandatory for all apps that target children, adding the "Committed to follow the Play Families Policy" badge to their privacy labels is optional [49, 51]. We examined whether having the badge displayed in apps' privacy labels implied developers' compliance with a number of Google Play's families policies. Of the 3,684 apps that used the badge, we found:

- 698 apps (19%) transmitted AAIDs, 361 (10%) of which did so to domains other than `app-measurement.com`;
- 449 apps (12.2%) targeted Android 13 and declared the `AD_ID` permission;
- 340 apps (9.2%) transmitted AAIDs alongside app set IDs, FIDs or SSAIDs to the same recipients, 106 (2.9%) of which did so to domains other than `app-measurement.com`;
- 163 apps (4.4%) transmitted AAIDs to third-party advertising or analytics services without using any of their COPPA or CCPA privacy configurations correctly[10];
- 145 apps (4%) prepared data safety labels that included at least one disclosure mistake about the personal data they transmitted;
- 97 apps (2.6%) declared the `ACCESS_FINE_LOCATION` permission;
- 89 apps (2.4%) transmitted AAIDs, SSAIDs or app set IDs to providers of third-party advertising SDKs which are not on Google's list of self-certified advertising SDKs [123] (AppLovin [25] or Yandex [225]);
- 71 apps (1.9%) embedded a version of a self-certified advertising SDK (Unity Ads [85], AdColony [9], ironSource [128], Chartboost [58], InMobi [125], or Vungle [219]) that Google Play prohibits embedding in child-directed apps [123]; and
- 21 apps (0.6%) did not use TLS in all their communications that contained personal data.

These percentages show that DFF-badged apps were not necessarily compliant with Google Play's families policies [49].

Data Safety Labels

We analyzed the data safety labels of the 828 apps that transmitted AAIDs, fine or coarse geolocation coordinates, WiFi MAC addresses, router SSIDs, router BSSIDs, names, or email addresses (see Table 6.2). Of these 828 apps, 491 apps (59.5%) had disclosure issues in their labels. Of the 491 apps, 393 either did not have a label (284 apps) (Figure D.8) or had a label that explicitly stated that no personal data is collected or shared by their apps (109 apps) (Figure D.7). Of the 207 apps that posted inaccurate labels, 200 apps did not disclose

[10]In our analysis, we assumed that a client- or server-side privacy configuration was not used when we did not find the corresponding privacy signal in inbound or outbound transmissions.

at least one of the data types we observed being transmitted and 121 apps did not have all the correct purposes for the data types they collected and/or shared listed in their labels. Additionally, 70% of the 207 apps had the DFF badge in their inaccurate data safety labels. After excluding the 172 apps that we did not observe continuing to transmit certain data types in their later versions, the number of apps that had disclosure mistakes decreased from 491 to 399 (61% of 657). Table 6.8 summarizes how this improved over time for the apps that we observed transmitting AAIDs, location data, MAC addresses, router SSIDs or BSSIDs, names, and email addresses.

We also investigated whether developers' use of "Data is encrypted in transit" versus "Data isn't encrypted" in their labels was accurate. Of the 51 apps that did not use TLS in transmissions that contained personal data (Section 6.4), only 20 apps disclosed that encryption was not used in their labels, whereas the rest either incorrectly stated that encryption was used when it was not (9 apps), or did not make any disclosures about whether data was encrypted in transit (22 apps). Five of the 20 apps initially disclosed that encryption was used when it was not, but updated their labels to state that it was not used instead of securing their unencrypted communications.

We also investigated whether the 2,578 apps that transmitted SSAIDs, app set IDs, and FIDs to domains that do not include `app-measurement.com` disclosed doing so in their labels and found that 57% (1472 apps) of them did not, 53 of them updated their labels to disclose transmitting "Device IDs" whereas 42 removed this data type from their labels after our initial tests. Of the 2,831 apps that transmitted any of the data types listed in Table 6.2 to domains other than `app-measurement.com`, 1,673 (59%) either did not prepare a data safety label or posted one that did not disclose all the data types transmitted by their apps (RQ3). Fifty-seven of the 1,673 apps fixed their labels after the first testing round to add the relevant disclosures and 41 removed device identifiers from their labels despite continuing to transmit them.

The improvement in apps' privacy practices over time was also corroborated by observing that the number of apps without data safety labels decreased from 2,189 (32%) in the first round to 499 (10%) in the second round. Of the apps tested in both rounds, 177 of the apps that initially did not have a label updated their app listing to publish one. Thus, the total number of apps that did not have a data safety label in any of the two testing rounds was 2,071.

The improvements observed between the two testing rounds could have resulted from developers reacting to Google Play's notifications that informed them that their labels *"will be invalidated"* unless they update them to include accurate disclosures about their apps' practices by specific enforcement deadlines [198].

Our results also suggest that developers are not always able to correctly follow Google Play's classification of data types or purposes, or understand Google Play's distinction between *collected* versus *shared* data types [51] when preparing their labels. For example, we found developers who transmitted AAIDs to advertising services who considered doing so as data collection and not sharing, or used "App functionality" as a purpose for doing so instead of using "Advertising or marketing." Additionally, 113 of the apps that transmitted device

Table 6.8: Disclosure inaccuracies in privacy labels.

	# of apps (N=828) [all app versions]	**# of apps (N=657) [recent versions only]**
Total number of apps that have DSL mistakes	491	399
# of apps that did not have DSLs	284	259
# of apps that did not disclose a data type	200	134
# of apps that did not disclose a purpose	121	96

identifiers used “User IDs” instead of “Device IDs” in their labels, which could suggest that their developers were unable to distinguish between these two data types.

6.5 Developer Survey

We recruited developers of child-directed apps to participate in a survey approved by our Institutional Review Board (IRB). It focused on developers’ experiences with configuring third-party SDKs for compliance with privacy regulations and Google’s policies, and preparing their apps’ privacy labels. It also asked about the privacy guidance that they rely on (if any) or need to satisfy their compliance obligations (see Appendix D.1). It also helped us shed light on the phases of the development process where privacy guidance is most likely to be used and the specific privacy guidance formats that developers would like to have (e.g., privacy checklists, app templates, and interactions with experts).

We used emails found in apps’ Play Store listings to invite 2,378 developers to participate in the survey and sent invitations to developers through their publicly-available accounts on social media. The contacted developers were part of app development organizations whose child-directed apps were available on Google Play. To incentivize participation, we offered respondents the option to enter a drawing for one of five $200 Amazon gift cards. We received 53 complete responses to our survey. This response rate is consistent with the rates observed in related studies that sent surveys to the same population (e.g., [16, 88]).

Results

This section summarizes the results of our survey. For open-ended responses, two researchers qualitatively coded them using a codebook that they jointly developed after analyzing a subset. During this process, they frequently met to discuss coding results and resolve disagreements until they reached a high level of agreement (Cohen’s κ score of 0.90).

Working with Third-Party SDKs

Most of the respondents indicated that their apps embedded SDKs that shared personal data with some of the third-party SDKs listed in Tables 6.4, 6.5, 6.6, and 6.7 for advertising or analytics purposes. Of the 53 respondents, 49%, 20%, 15%, and 6% used SDKs in their apps that shared data with Google Admob [11], Unity Ads [85], Meta [152, 153], and ironSource [128], respectively. While 74% of respondents indicated familiarity with third-party SDK privacy configurations, only 55% used them in some or all of their apps (RQ4). Those who did not use privacy configurations despite their familiarity with them provided a number of justifications. They indicated that Google Play accepted their apps regardless, that there is no need to configure SDKs included on Google Play's of certified SDKs [123] for compliance, that the SDKs were compliance by default or that they did not understand how to do so.

Furthermore, 64% and 69% of respondents found configuring third-party SDKs for compliance or keeping them up-to-date to be challenging, respectively. They provided a number of reasons that can explain why not all of the apps we tested used privacy configurations correctly (Tables 6.4, 6.5, 6.6, and 6.7). These include shortcomings in SDK developer documentation that led developers to be unable to identify or correctly use the specific configurations applicable to their apps and the operational overhead associated with keeping up with the frequent updates made by SDK providers to their privacy configurations. One respondent explained:

"You have to read all the manuals and find some information in the bottom spots of some instructions which make creating these environments troublesome" (R46).

We additionally examined whether they understood the consequences of not using privacy configurations and found that 40% said that their apps could get removed from the store as a result. This finding further demonstrates the powerful impact that platform policies could have on developers' privacy practices. It shows that Google Play's ongoing enforcement efforts are shaping developers' perceptions about the consequences of not addressing privacy issues, which might subsequently lead them to improve their privacy practices. However, 45% of respondents trusted that the data collection practices of third-party SDKs would not introduce privacy compliance issues. This could therefore explain why privacy issues that result from developers' use of third-party SDKs might not get addressed until developers get notified by the store about them.

Preparing Privacy Labels

Thirty-percent of respondents found the task of preparing data safety labels for their apps to be difficult. While 57% of respondents' organizations dedicated parts of their development processes to this task (RQ4), 60% did not employ traffic analysis tools to identify data types transmitted by their apps before submitting their labels to the store. Furthermore, 45% tasked development teams with filling out Google Play's data safety questionnaire, whereas the rest assigned this task to non-technical teams. This could explain the observed discrepancies between the actual behaviors of the tested apps and their labels (Section 6.4). Two main approaches were relied on for guidance on what to include in data safety labels, which were third-party SDK documentation and notifications received from Google Play. One explained:

"Sometimes we know the answer, sometimes we choose at random, .. and then we wait for a decision from Google Play. If something is wrong, fix it and re-upload" (R31).

Only 21% of respondents believed that Google Play does not check the accuracy of disclosures included in data safety labels. Fifty-three percent of respondents, however, believed that these labels should be automatically prepared by the store upon submitting their apps. Respondents expressed challenges with mapping their apps' data practices to Google Play's classification of data types [51], frequently maintaining their labels to reflect changes in their apps' practices, and identifying the practices of third-party that need to be reflected in their labels. This is supported by the analysis presented in Section 6.4, as most of the observed discrepancies were caused by sharing of personal data with third parties. One participant explained:

"Even Google SDKs like Admob don't provide specific answers about how to fill out the Google data safety sheets. It is difficult to find out how to do it correctly" (R18).

Access to Privacy Guidance

Only 42% of respondents indicated that their organizations provided them with some form of guidance on how to fulfill their privacy compliance obligations. This was mostly presented in checklists, templates for privacy policies, or descriptions of privacy compliance requirements. Additionally, 81% of respondents expressed their willingness to add better support for privacy compliance in their processes once they are provided with proper guidance that can help them do so.

In terms of the type of guidance they wanted to have, it included having privacy checklists, reference apps demonstrating how to avoid common privacy issues, and the ability to interact with experts who can provide specific technical or legal advice. Many also needed guidance on how to avoid compliance issues that result from embedding third-party SDKs. Specifically, they needed to be guided through how to use third-party SDK privacy configurations, how to understand whether collected data would be used for prohibited purposes, and how to identify the types of personal data collected by third-party SDKs. This provides further support for the results of our technical analyses, which showed that most of the detected issues were due to improper use of third-party SDKs. For example, one respondent explained that they wanted a:

"list of identifiers used by the SDK and reference to relevant regulations for their use" (R38).

As for when in their development processes they would be most likely to use this guidance, most respondents indicated that they would use it during the development (58%) or testing (36%) phases of their processes. However, 57% of respondents also preferred to have Google Play present this guidance as part of the app submission process. This provides further evidence of the powerful impact that Google play could have on developers' privacy compliance processes.

The results of this survey therefore show that many developers added support for configuring third-party SDKs for compliance in their development processes and were adapting to Google Play's requirement that asked them to prepare data safety labels for their apps. However, they were also facing challenges with these tasks and needed to be guided through how to correctly use these privacy controls.

6.6 Discussion

The results of this study show a drastic decrease in the number of apps that transmitted children's personal data or did not encrypt their communications. However, we also show that use of third-party SDKs is still causing many types of privacy issues.

Third-party SDK developers

Misuse of third-party SDKs is still leading developers to not accurately disclose their apps' practices and enabling sharing of children's personal data (Section 6.4). While Google Play made major updates to its policies in the past few years [180], these policy efforts are still not eliminating privacy issues caused by improper use of third-party SDKs. This is also limiting the efficacy of Google Play's recently-introduced data safety labels [51] that aim to provide users with higher levels of transparency. Developers' lack of understanding of the behaviors of third-party SDKs embedded in their apps not only allowed sharing of personal data, but also enabled linking of identifiers, overprivileging apps, and publishing inaccurate data safety labels (Sections 6.4).

Currently, developers of third-party SDKs are not aligning their practices with the requirements of Google Play's policies [177]. While the policies prohibit collecting AAIDs from children, correct use of third-party SDK privacy configurations did not necessarily prevent collecting AAIDs from children or linking it with other identifiers (see Figures D.1 and D.2, and Tables 6.4, 6.5, 6.6, and 6.7). Use of third-party SDKs also led to sharing app set IDs with advertising services even though the policies do not allow using this identifier for advertising purposes [46, 119]. Such shortcomings in the designs of third-party SDKs can lead to introducing privacy compliance issues, even if SDK privacy configurations are used correctly.

There are improvements that can be made to the design of third-party SDK privacy configurations to reduce the operational overhead associated with using them. These include reducing the number of configurations that developers would need to use for each SDK and also standardizing the design of such configurations across the industry. As demonstrated in Tables 6.4, 6.5, 6.6, and 6.7, third-party SDKs currently have different configurations that can be used for different purposes, which increases the likelihood of error (i.e., misconfiguring SDKs for compliance). Before using ironSource's SDK [128], for example, developers need to decide on which of its two server-side configurations and five client-side configurations to use to comply with COPPA, GDPR and the CCPA [131].

To reduce the number of configurations, third-party SDK providers could utilize publicly-available metadata about apps (e.g., whether they are DFF-badged or teacher-approved) to determine whether they are primarily child-directed and disable targeted advertising accordingly. This feature is currently partly supported by Google Admob [11], but not by other SDK providers. Many privacy issues could also be eliminated once third-party SDKs use privacy-preserving defaults that disable data sharing. While we showed an overall improvement in apps' privacy practices, more progress can therefore be achieved once third-party SDK providers do their part in helping reduce the burden of compliance on developers.

The Google Play Store

The observed overall improvement in apps' privacy practices is likely the result of developers adapting to changes made to Google Play's policies over the past few years [177, 180]. This shows that Google Play is in a powerful position to influence organizational privacy compliance practices worldwide. Given the various privacy issues that are still present in child-directed apps, there is a need for further strengthening Google Play's enforcement efforts to protect children's privacy. In order for these policy efforts to be effective, Google Play might also need to utilize its powerful position to educate developers about how privacy compliance issues can be identified and addressed early in their software development processes. To do so, the various points of interaction between Google Play and developers could be leveraged to raise developers' awareness about how to build privacy-preserving apps. These are mainly Google Play's developer console [176], e-mail notifications, and policy documentation [177].

Despite the drastic reduction in the number of child-directed apps that transmitted personal data, improper use of third-party SDKs is still the leading root cause of privacy compliance issues. While Google Play is continuously improving its developer policies, it appears to not be comprehensively auditing the practices of third-party SDKs before including them on its list of certified SDKs [48, 123]. Instead of requiring SDK developers to *self-certify* their compliance with Google Play's policies [48], the certification process should involve performing technical investigations of third-party SDKs' data collection practices from multiple perspectives. There are many heuristics that can be employed in this process, which include examining the effects of using SDK privacy configurations on data collection, understanding the extent to which SDKs are able to link different data types, and detecting whether SDKs are using side channels to access personal data [186]. Employing a comprehensive auditing process of SDKs' privacy practices is therefore likely to improve the effectiveness of Google Play's enforcement efforts.

Anecdotal evidence suggests that Google Play has been auditing apps' data transmission practices to verify the accuracy of data safety labels [197, 198]. As part of Google Play's ongoing enforcement efforts, developers have been receiving notifications requiring them to update their labels once discrepancies are detected. While this approach is likely leading many developers to fix disclosure issues detected in their labels, following a proactive approach to performing such audits by Google Play might be more effective. Google Play's developer console [176] can incorporate features that provide developers with real-time feedback about their apps' privacy practices as part of the app submission process. This would not only prevent developers from publishing apps that have privacy issues, but also allow them to understand their apps' data practices and resolve privacy issues early in their development processes. We showed that 60% of survey respondents did not have access to traffic analysis tools that would help them do so (Section 6.5), which emphasizes the need for providing developers with feedback about their apps' privacy behaviors.

Google Play's developer console [176] could also utilize data entered during the app submission process to provide customized privacy guidance. While Google Play has access to data about apps' target audiences and the countries where apps are available to users, this data could be utilized to raise developers' awareness about their compliance obligations (e.g., how disable sale of personal data). Google Play's enforcement efforts should also prevent developers from making inaccurate representations to users through its badging system. For example, while the DFF badge allowed developers to communicate their *commitment* to complying with Google's policies [49, 51], we showed that even DFF-badged apps suffered from various types of privacy issues (Section 6.4).

Regulators and Policy Makers

Most of the data sharing issues we identified resulted from failures to apply data minimization principles by third-party SDK providers. Sharing children's personal data even when privacy configurations were used correctly and including different identifiers in the same transmissions are two prominent examples. The prevalence of these issues calls for the need of adding more restrictions to existing legal provisions that allow collecting device IDs for certain purposes. For instance, restricting the number of identifiers that can be collected by third parties who are relying on COPPA's internal operations exception could lead to further improvements in overall privacy practices [61].

Our results showed that Google Play's policy efforts can shape privacy practices of development organizations publishing apps from various countries of the world. Measuring adoption of different types of SDK privacy configurations also demonstrated how developers are responding to requirements of federal (i.e., COPPA) versus state regulations (i.e., CCPA) in the United States. However, the effects of Google Play's efforts on worldwide developer privacy practices raise the question of whether enacting more privacy regulations in various regions of the world is introducing unnecessary complexity to development processes. Given the differences in the regulatory requirements of privacy regulations, unifying privacy requirements of different privacy regulations enacted in one country (e.g., federal and state regulations in the US) or across countries is likely to reduce the burden of compliance on developers.

While our investigations applied specifically to Android child-directed apps, our results still show that app platforms are in a powerful position to make drastic improvements in developers' privacy practices. Future work could therefore explore the feasibility of aligning platform policies with the requirements of applicable privacy regulations and the specific roles that app platforms can play in enforcing the requirements of privacy regulations on all app populations.

6.7 Limitations

We employed technical and qualitative methods to understand how changes in platform policies impacted development practices. However, a few factors could have introduced a degree of uncertainty to our results. First, while we also considered apps that are not DFF-badged, all developers of child-directed apps are required to comply with the families policies [49], just as they are required to comply with COPPA. For this reason, and because we wanted to identify as many child-directed apps as possible, comparing our results to those of Reyes et al. [189] is slightly confounded because Reyes et al. only examined apps that are enrolled in DFF (i.e., a subset of the apps that are likely subject to COPPA). Second, traffic obfuscation could have affected our analysis of traffic data. However, we employed advanced de-obfuscation techniques that were verified in the literature, and therefore our results can be considered a lower bound for the types of privacy issues that exist in general population of child-directed apps. Third, there could have been privacy configurations that we did not consider due to: (1) developers using old SDK versions that offered different configurations, or (2) SDKs employing server-side configurations that do not trigger sending privacy flags. To address this limitation, we repeatedly monitored third-party SDK documentation to test the updates made to their privacy configurations over the past two years and tried to identify other indicators of the use of server-side configurations (e.g., not transmitting AAIDs when used correctly). Fourth, although Google's list of self-certified SDKs [123] was updated after our pilot tests to specify the minimum allowed SDK versions and remove AppLovin [25], we included these in our measurements to understand developers' privacy practices at that point in time.

6.8 Ethics

The survey we distributed to app developers was IRB-approved by our university. To preserve respondents' privacy, we did not collect identifiable data from them other than an e-mail address that we stored separately from the survey data. We collected e-mail addresses from respondents who wanted to enter our drawing for five $200 Amazon gift cards, but we also allowed developers to participate without providing their e-mail addresses. Participation in our survey was voluntary and informed consent was obtained from all the respondents. For recruitment, we used the publicly available e-mail addresses posted by developers on their app listing on the Google Play Store. When we sent our e-mail invitations, we included an option that allowed developers to opt out from any future communications with us. We additionally sent developers of child-directed apps invitations to participate in our survey through the publicly-available profiles of their app development organizations that we found on social media.

While our large-scale crawling of the Google Play Store to identify and test child-directed apps could have put some pressure on the store, our crawling approach is consistent with those followed in prior related studies (e.g., [189]). Furthermore, since apps were downloaded over a long period of time (i.e., several weeks) using a few number of Android devices, it is unlikely that our approach had negative effects on the store.

6.9 Conclusion

In this chapter, we presented the results of a comprehensive investigation of the privacy practices of child-directed apps that were available on the Google Play Store in 2023. Based on comparisons of our results with those observed by a related study conducted in 2018, we provide evidence of continuous improvements in the behaviors of the population of Android child-directed apps. This is likely due to the major changes that were made to Google Play's policies over the past few years and developers becoming increasingly aware of their privacy compliance responsibilities. However, we also show that the community is yet to address one of the leading causes of privacy compliance issues, which is developers' inabilities to use third-party SDKs in ways that do not negatively affect their compliance with platform policies and applicable privacy regulations.

This is contributing to the persistence of the same types of privacy issues that were observed a few years ago (e.g., sharing of device identifiers with third parties) [189]. It is also negatively affecting the effectiveness of app set IDs and leading to the prevalence of disclosure issues in data safety labels, both of which are recently-introduced interventions aimed at providing users with higher levels of privacy. Furthermore, while adoption of third-party SDK privacy configurations is increasing, developers were still struggling to correctly and consistently use configurations applicable them across all their apps. The results of a follow-up developer survey showed that while developers made changes to their development processes in response to recently-introduced platform policies, use of third-party SDKs was still introducing challenges that are leading to disclosure and configuration issues. Taken together, our findings demonstrate the need for providing developers with actionable privacy guidance and shed light on how Google Play can further strengthen its policy enforcement efforts.

Chapter 7

Designing Actionable Privacy Guidance: A Study of Privacy Checklists

7.1 Introduction

The results of the technical and qualitative analyses presented in Chapters 5 and 6 demonstrated the nature of privacy compliance issues being introduced in app development processes and explained the reasons why such issues were introduced by developers. They also showed the real-world complexities developers are experiencing while trying to build apps that are compliant with applicable privacy regulations and app market policies. To ensure compliance with the requirements of applicable regulations and policies, developers are assumed to identify applicable privacy requirements, understand how they apply to their apps, and make necessary adjustments to their development processes accordingly. Based on these requirements, their responsibilities might involve implementing proper consent mechanisms in their apps, limiting the transmission of certain types of personal data from users, adding relevant disclosures to privacy policies or privacy labels, or configuring third-party SDKs for privacy compliance.

However, we also showed that developers were unable to translate privacy requirements into actionable tasks that can then be given sufficient consideration in their development processes. The results of the survey and interview studies discussed in Chapters 5 and 6 provided a number of explanations, which include developers' inabilities to:

- stay up-to-date with applicable privacy compliance requirements, and
- understand whether their use of third-party SDKs is introducing privacy compliance issues.

These challenges led many developers, and particularly those who cannot afford to hire legal experts, to consider their apps compliant with all applicable privacy regulations as long as the privacy issues that existed in their apps were not flagged as policy violations by the Google Play Store [111].

Our qualitative investigations also demonstrated developers' willingness to address privacy compliance issues that exist in their apps once they are provided with usable privacy guidance that can help them understand how to do so. Currently, developers who are interested in understanding their privacy compliance obligations are referred to lengthy documentation provided by the Google Play Store [177], SDK providers, and government websites. Thus, translating such privacy guidance to actionable tasks not only requires developers to dedicate time and resources to this challenging task, but also have special expertise in privacy compliance, which makes privacy guidance inaccessible to many developers. While the Google Play Store [177] is providing developers with some privacy guidance when they notify them about the existence of certain types of privacy issues in their apps, the results presented in Chapters 5 and 6 showed that developers did not always understand the types of privacy issues reported to them or how to fix them. These findings emphasize the importance of understanding the limitations of privacy guidance that developers are currently relying on and designing alternative forms of privacy guidance that address these limitations.

To that end, we followed an iterative process to design an interactive tool that generates privacy checklists for mobile apps, which can then be used by developers in their app development processes. We also conducted 20 interviews with developers who published apps on the Google Play Store [111] to obtain feedback on how to further improve our tool to make it more suitable for their development workflows. We chose to focus on privacy checklists for the following reasons:

- they were recommended as a possible intervention by many developers who participated in our previous survey and interview studies (see Chapters 5 and 6),
- they were proven to be effective at reducing error rates in other domains (e.g., medicine [98, 116, 138]), and
- they are being offered in online sources, including law databases such as Bloomberg Law [39] and WestLaw [221], yet it is unclear whether using them would be effective at helping app developers identify and address common privacy compliance issues.

During the interviews, we also asked participants about the sources of guidance that they used for understanding their privacy compliance obligations and the specific forms of privacy guidance that they need. To further understand the extent to which existing forms of privacy guidance are effective, we also recruited lawyers to participate in an anonymous survey that inquired about the types of privacy advice they provide to software developers (e.g., whether they use privacy checklists available in law databases). Our goal was to answer the following research questions:

RQ 1 What are the main sources of privacy guidance that are being used by developers of Android apps?

RQ 2 Would app developers be willing to use privacy checklists within their existing development processes?

RQ 3 If so, what are the design requirements for privacy compliance checklists targeted at Android app developers?

In this chapter, we explain the process we followed to design an initial version of a privacy checklist, which we then used as a basis for implementing a tool that generates privacy checklists for mobile apps. We then discuss our findings in regards to the design requirements that can be considered when designing this type of privacy guidance.

7.2 Methodology

We designed an initial version of a privacy checklist aimed at helping app developers address privacy issues before releasing their apps on the Google Play Store [111]. The checklist was designed based on our review of the basic privacy compliance requirements of a number of privacy regulations and the Google Play Store policies [177], and our observation of the types of privacy issues that were commonly introduced in development processes in our prior studies (see Chapters 5 and 6). It included items that aimed to remind developers to implement proper consent mechanisms, disclose collection or sharing of personal data in privacy policies and Google Play's data safety labels [51], configure third-party SDKs for privacy compliance, and handle users' personal data in accordance with applicable regulations and policies. It also explicitly mentioned that it is designed for research purposes only, and therefore not meant to provide developers with legal advice. This checklist is included in Appendix E.1.

We recruited five developers of child-directed apps to participate in semi-structured interviews, which took from 60 to 90 minutes (an average of 80 minutes). We compensated participants $30 Amazon gift cards for their participation in this round of interviews. During the interview sessions, we shared with participants our privacy checklist and asked them to provide feedback on each of its items and suggestions for improvement. Our aim was to evaluate the extent to which the content of our privacy checklist is understandable to app developers and understand whether developers would be willing to follow the provided guidance. We also asked about the types of privacy guidance they use in their daily work and discussed with them the extent to which these types of guidance were helping them understand how to fulfill their privacy compliance obligations. The interview questions are included in Appendix E.4.

Interactive Privacy Checklist

The results of our thematic qualitative analyses of the data collected through the five interviews helped us identify a preliminary set of design requirements for privacy checklists, which we then used to design and implement a tool that generates privacy checklists customized to mobile apps. Our tool asks developers a number of questions about their apps (examples are provided in Appendix E.2), and then generates privacy checklists customized to the apps in question. The questionnaire part of the tool mainly asks app developers about:

- The countries where their apps are available to download by users.
- The audiences targeted by their apps.
- The types of data transmitted by their apps to first or third-party recipients.
- The Android permissions declared in their apps.
- The third-party advertising and analytics SDKs integrated in their apps.

Based on the answers provided by developers to the questionnaire, the tool would then generate a privacy checklist containing a list of checklist items that are relevant to their app(s). This allows shortening the generated checklist items to only present privacy guidance that is relevant to the app in question, and also alleviates developers from the burden of identifying applicable privacy guidance.

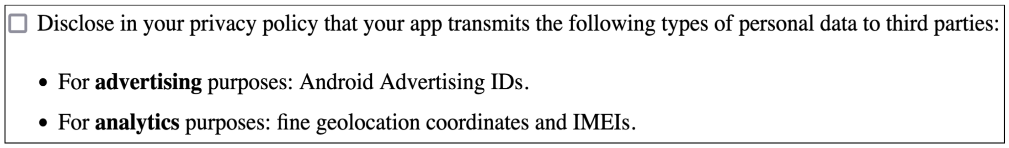
Since the results of our prior qualitative investigations demonstrated the difficulty developers are experiencing with navigating third-party SDK documentation, we also implemented the tool such that it generates privacy guidance relevant to embedded third-party SDKs. For this purpose, we selected a set of popular third-party advertising and analytics SDKs, and then read their developer documentation to identify the types of data they collect by default and also the privacy configurations offered to developers to control data collection and/or sharing. Then, we implemented the tool such that it generates checklist items that are based on what we found in third-party SDK documentation. This covered guidance related to disclosing the sharing of certain types of personal data in privacy policies and data safety labels, using third-party SDK privacy configurations to control data sharing, and obtaining user consent before allowing the transmission of personal data.

Therefore, we implemented the tool such that it generates privacy checklists that mainly provide developers information on the following:

- Privacy regulations that might apply to their apps.
- In-app privacy features that they might need to consider implementing in their apps (e.g., consent screens or age gates).
- Data types that should be disclosed in their apps' privacy policies or data safety labels.
- Collection of third-party SDK privacy configurations that are relevant to their apps.
- Dangerous Android permissions which they might need to remove from their apps.

Figures 7.1, 7.2, 7.3, and 7.4 provide examples of checklist items generated by our tool (more examples are provided in Appendix E.3). After implementing the tool, we recruited 15 app developers for a second round of interviews to further understand how to design actionable privacy guidance (two of them participated in the first round of interviews as well). In this round of interviews, we started by asking developers a set of questions about the types of privacy guidance they used and then asked them to use our tool to generate a privacy checklist for one of their apps. This allowed us to observe developers while they used our tool to generate a privacy checklist for one of their apps during the interview sessions.

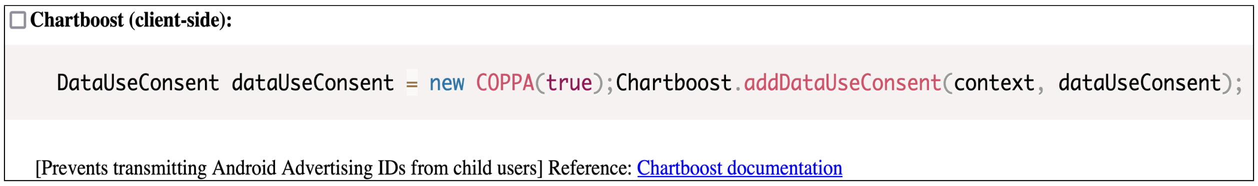
Prior to conducting the interview sessions, we asked developers to share with us the apps that they wanted to generate a checklist for during the interview sessions. This allowed us



☐ Disclose in your privacy policy that your app transmits the following types of personal data to third parties:

- For **advertising** purposes: Android Advertising IDs.
- For **analytics** purposes: fine geolocation coordinates and IMEIs.

Figure 7.1: A checklist item generated by our tool providing guidance on disclosure of data sharing in privacy policies.

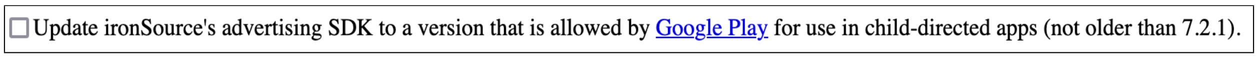


☐ **Chartboost (client-side):**

```
DataUseConsent dataUseConsent = new COPPA(true);Chartboost.addDataUseConsent(context, dataUseConsent);
```

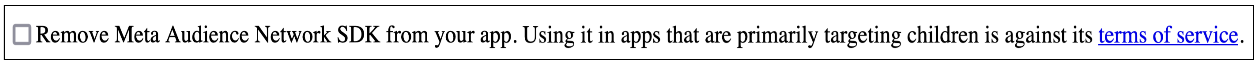
[Prevents transmitting Android Advertising IDs from child users] Reference: Chartboost documentation

Figure 7.2: A checklist item generated by our tool providing guidance on configuring Chartboost's SDK [58] for privacy compliance in a child-directed app.



☐ Update ironSource's advertising SDK to a version that is allowed by Google Play for use in child-directed apps (not older than 7.2.1).

Figure 7.3: A checklist item generated by our tool providing guidance on updating third-party SDKs to comply with Google Play's policies [177].



☐ Remove Meta Audience Network SDK from your app. Using it in apps that are primarily targeting children is against its terms of service.

Figure 7.4: A checklist item generated by our tool providing guidance on removing Meta's advertising SDK [160] from apps that are targeting children only.

to test participants' apps beforehand to understand their privacy practices and use that as a basis for our discussions with participants during the interview sessions. By doing so, we were able to observe how they would use our tool, whether they would be able to answer all the questions included in the questionnaire, and whether they would find the content of the generated checklists helpful to them.

Given that we designed the tool to generate checklists that include privacy-related technical and legal information, we also aimed to understand the extent to which developers believed that the included content is relevant to their responsibilities. Similar to what we did in the first round of interviews, we also asked them to:

- evaluate the types of privacy guidance currently available to them,
- discuss with us the types of guidance they need, and
- provide feedback on the content of the checklists they generated using our tool during the interview sessions.

These interviews took from 60 to 120 minutes (an average of 93 minutes). Interview participants were recruited through the e-mails published as part of app listings on the Google Play Store [111] and social media. They were offered $80 Amazon gift cards for their participation in this round of interviews. Our studies were IRB-approved by the ethics committee at our university.

After conducting the first round of interviews, two researchers followed an iterative process to build a codebook that was then used to perform thematic analysis of the transcribed interviews. The codebook consisted of 40 codes, which covered codes such as *consulted-online-resources, privacy-checklist-actions-to-take, privacy-guidance-limitations,* and *guidance-sdlc-phase*.

We followed the same process to analyze the remaining 15 interviews, which resulted in adding 42 codes to the codebook. The additional codes allowed capturing details related to whether developers were able to identify privacy issues that existed in their apps after using our tool (e.g., *privacy-checklist-positive-effect*), whether their organizations already used privacy checklists (e.g., *existing-checklists-usage-scenarios*), and whether they would be willing to incorporate checklists in their development processes (e.g., *privacy-checklist-willingness-to-use*).

Two researchers used the final codebook, which consisted of 82 codes, to independently code all the interview transcripts. During this process, they frequently met to discuss their coding results for each of the interview transcripts and resolve disagreements. In Section 7.3, we present the results of our qualitative analysis of the 20 interviews, conducted in the two rounds with the 18 developers.

Survey of Lawyers

We also designed an anonymous survey that we targeted to privacy lawyers involved in advising mobile app developers on how to comply with privacy regulations. Our goal was to understand the nature of interactions that take place between developers and their legal advisors, and identify the types of privacy guidance that were reaching developers from their advisors. Our survey included questions about lawyers' experiences with advising mobile app developers on how to comply with privacy regulations (e.g., COPPA [61], the GDPR [66], and the CCPA [30]). We asked about how lawyers provided privacy guidance to app developers (e.g., whether they used documented guidance or communicated their guidance in discussions with developers) and how developers used the provided guidance to fulfill their compliance obligations.

For those who indicated that they provided their clients with documented forms of privacy guidance, we additionally asked about the format of that guidance (e.g., privacy checklists versus templates for privacy policies) and the type of content included in the provided guidance. We also included questions about the challenges experienced by lawyers when trying to communicate privacy advice to developers and their recommendations for how to improve the quality of such communications. The survey questions are included in Appendix E.5.

We distributed our survey on social media sites and sent e-mail invitations via mailing lists of lawyers (e.g., members of bar association privacy interest groups). We received responses from 14 lawyers to our survey, which we discuss in the following section. Our survey was approved by our IRB.

7.3 Results and Analysis

Of the 18 developers who participated in our interviews, 14 were involved in developing child-directed apps whereas the rest developed apps that targeted general audiences. Seven of the participants worked as independent developers, seven were part of small or medium size organizations (between 10 and 50 employees), and four worked for organizations that have more than 50 employees. Below, we present the results of our thematic analysis of the collected data, focusing specifically on:

- identifying the challenges developers are experiencing when using existing sources of privacy guidance,
- understanding the extent to which privacy checklists could address these challenges, and
- discussing how privacy guidance targeted to developers should be designed.

Existing Sources of Privacy Guidance

In the following, we list the main sources of privacy guidance used by the developers who participated in our interviews.

Professional Legal Advisors

None of the independent app developers who participated in the study consulted lawyers. Of the 11 developers who were not working independently, two participants had dedicated legal teams and another two participants consulted external lawyers. However, three of the four who used professional legal advice from internal or external counsel explained that their advisors were not actively involved in all decisions that relate to privacy compliance. They mainly helped with drafting their privacy policies or provided advice on how to comply with the Google Play Store policies [177]. Our results also suggest that independent developers do not have the resources that allow them to seek professional legal advice. Three independent developers explained why they never consulted privacy lawyers:

"I will come to this point at the stage when my apps start earning money." ($P11$)

"Maybe if I am in the top 10 in the app store developing apps with millions of revenue, maybe I will consult lawyers." ($P2$)

"It is maybe a 100 to 200 dollars a month startup cost, and one of the ways to keep cost down is not to use lawyers." (*P*9)

Online Resources

Most participants mentioned relying on the developer documentation provided by Google Play [177] as their main source for privacy guidance. Many also referred to online forums and social media to learn about how other developers addressed certain privacy compliance issues. However, developers mainly used these resources to understand how to comply with the Google Play Store policies [177], and did not often specifically seek guidance on how to comply with each of the privacy regulations applicable to their apps. Some also trusted that following guidance provided in Google Play's documentation [177] is sufficient for understanding how to comply with applicable privacy regulations. For example, even though Google Play's documentation explicitly states that developers are responsible for ensuring that their apps comply with applicable privacy regulations [49], one participant explained:

"We refer to Google developers' website where Google provides guidelines on how to follow COPPA, how to follow California privacy guidelines and the European privacy guidelines." (*P*8)

The guidance provided by Google Play [177] also seems to influence the extent to which development organizations that are not based in the United States or Europe are familiar with privacy laws that are applicable to their apps. Our results suggest that the emphasis placed on COPPA [61] in Google Play's guidance [177] led developers to be more familiar with COPPA [61] compared to other applicable privacy laws. However, this also led other developers to incorrectly believe that their apps are compliant with COPPA [61] once they are accepted for inclusion on the Google Play Store [111]. Two participants explained how they learned about COPPA [61]:

"I do not think that our company follows CCPA or GDPR because I never heard of these terms in our company. I have heard of COPPA, it is from Google and our company relies on that only." (*P*15)

"Because of the news and also the guides of Google Play and Admob, we know this COPPA law." (*P*8)

Developers also mentioned using online services that helped them generate privacy policies for their apps. They appreciated having such tools that allow them to generate privacy policies customized to their apps without having to dedicate time to writing their privacy policies. One of the developers who worked independently explained how they used such services:

"I was extremely glad when I found a service that asked me 'is your app for kids?' Yes, 'is your app showing blood?' No. So I was creating my privacy policy with this service and to be honest I did not read the content at the end." ($P11$)

Another online resource that we found to be used for guidance on privacy compliance is ChatGPT [59]. Five participants explained that they used ChatGPT [59] for various purposes, which include generating their privacy policies, understanding what disclosures to include in their apps' data safety labels, or resolving technical issues that were detected in their apps. It was additionally used when developers were seeking further explanation on privacy guidance found on government websites (e.g., FTC's guidance on COPPA compliance [61]), third-party SDKs' privacy policies, and Google Play's policies [177]. Participants provided examples demonstrating how they used ChatGPT [59]:

"So we are just asking ChatGPT about the particular things like saying why are we getting these rejections from the play store." ($P13$)

"I have a privacy policy page, which was generated by ChatGPT ... I am not sure if it is compliant with the regulations and policies." ($P6$)

"I'll go to the COPPA page by the government that has all the regulations, I'll take all that content, give it to ChatGPT and then I will have it review whatever I need it to review." ($P9$)

Privacy Guidance Developed Internally

Google Play's ongoing enforcement efforts also led organizations to develop resources for privacy guidance that they required their engineers to follow so that their apps do not get rejected by the store. These include templates for privacy policies, checklists to be followed before uploading apps to the store, and sample code for consent screens. Developers explained that they used lessons learned from repeated rejections of their apps by the Google Play Store [111] to build these resources to avoid facing the same rejections in the future. However, most of them also indicated that these internally-developed resources were not reviewed by legal advisors, as they merely served to help them comply with the Google Play Store policies [177]. Participants explained the types of content included in these internally-developed resources:

"These are just the failures we received after submitting some of our apps to Google Play, we kept adding these failures, and that became our checklist." ($P17$)

"We have a checklist, .. and whatever we learn from Google, we add to that checklist." ($P15$)

"We already had a template for our privacy policy, and we just had to replace the name, and it used to work for all of our apps." ($P12$)

In total, five participants who were not working as independent developers indicated that checklists were used in their development processes. For four of them, checklists were developed by someone in their organizations based on their understanding of Google Play's policies [177], and one of them used a checklist for GDPR [66] compliance that they received from a law firm. Three of the four participants whose checklists were developed by their organizations explained that their checklists were maintained by their seniors, who were responsible for updating their checklists' content to reflect changes in privacy requirements (e.g., updates made to Google Play's policies [177]). The fourth participant indicated that developers in their organization create customized checklists during the planning phases of each of their apps, which they then follow during subsequent phases to help them comply with Google Play's policies [177]).

Participants also explained that checklists helped them ensure that privacy requirements are consistently followed across all their projects and teams, and facilitated privacy-related discussions with their managers. One participant provided an example of how they used their checklist:

"Before coming out with a new version release, we go through the list and see whether we are adhering to whatever we are supposed to adhere to and if Yes then only we submit the build for release." ($P3$)

Our results suggest that Google Play's ongoing enforcement efforts not only led organizations to develop resources that their personnel can use for privacy guidance, but also improved developers' familiarity with their compliance obligations. During the interviews, developers provided details about some privacy issues that led their apps to get rejected by the Google Play Store [111] (e.g., sharing of advertising IDs with third parties and missing disclosures in their apps' data safety labels). Developers' efforts to resolve these issues by reading Google Play's policies [177], referring to the links included in the messages received from Google Play, and submitting required details through the Google Play console [176] seemed to have improved their understanding of privacy compliance requirements. For example, one participant explained why they decided to implement an age gate in their mixed-audience app:

"Google has this guideline saying that if your app attracts both children and older audiences, then you should implement an age gate and you should show ads based upon the age of the user." ($P8$)

However, this also led other developers to not fix privacy issues that they knew existed in their apps until they received notifications from the store telling them to do so. For example, when we asked one participant about the reason why they did not disclose in their app's data safety label that their app shared personal data with Google Admob [107], they explained:

"We already know that we are collecting data. So this is wrong but they do not stop me, so I put no data collected and go on." ($P2$)

Other developers mentioned that Google Play's messaging improved their familiarity with privacy requirements, but that they were still struggling with understanding how to apply these requirements to their apps (e.g., needing guidance on how to obtain user consent). This shows that Google Play's ongoing enforcement of privacy requirements is shaping attention to privacy compliance in app development processes. Developers might therefore be more likely to dedicate resources to complying with requirements of applicable privacy regulations that are enforced by Google Play, compared to those that are not.

Referring to Popular Apps

Reviewing the privacy practices of apps that were published in similar categories on the Google Play Store [111] or were highly popular on the store was another source of privacy guidance used by app developers. Participants followed this approach when they needed to prepare their privacy policies, understand how to design age gates, and decide on how to obtain user consent. For example, a developer of child-directed apps mentioned referring to apps published by PBS Kids [173], which is a developer of popular child-directed apps on the store, for guidance on how to verify whether children or parents were using their app. We also observed that this practice was not only adopted by independent developers, but also others who were part of app development organizations whose development processes dedicated resources to researching the practices of similar apps that were published on the store. For example, one participant explained why their app's privacy policy missed disclosures about its data sharing practices:

"I just take a similar app which has lots of downloads. I trust that they did it well and I just copy things." ($P11$)

Limitations of Existing Privacy Guidance

Our thematic analysis of the interview data allowed us to identify a number of factors that made it challenging for app developers to follow privacy guidance offered by Google Play [111], official documentation of third-party SDKs, or government websites. Most developers indicated that they were unable to dedicate time to finding privacy guidance that is relevant to them in these resources, particularly due to the amount of content that is included in them. They also mentioned that much of the guidance provided in these resources included information that they could not understand, either because it used legal terms that they were not familiar with or targeted developers with special technical expertise. Participants explained some of the challenges they experienced when they tried to use these resources:

"I ignore policies.., because they are very long. They are good for courts, for lawyers, for these people but not for me." ($P6$)

"It felt like I was reading a file that is written for lawyers." ($P16$)

"There are some words which we are not commonly using, so we do not understand the meaning of these words." (*P*17)

Developers who attempted to follow privacy guidance provided in existing sources explained a number of reasons that made it difficult for them to understand how to take action to improve their apps' privacy practices. One of which was the lack of sufficient technical guidance on privacy compliance that can help developers understand how to resolve privacy issues that existed in their apps. Developers found existing technical guidance to be either incomplete, outdated, or not optimized to reduce implementation effort. They also indicated that they often were only able to find generic privacy guidance that was scattered across multiple sources, leading them to be unable to synthesize this guidance and understand how to apply it to their apps. Another reason was having privacy guidance presented in lengthy text only while not using other formats that could make it easier for developers to consume privacy guidance. These reasons discouraged developers from following existing privacy guidance, two explained:

"We check but sometimes we are fast in our development, and whenever anything is not clear to us, we do not use it." (*P*15)

"Maybe I did not find it helpful that they did not make it easy for me explaining what they want in the policy but on the other hand they rarely kicked out my apps. So I feel good about that." (*P*1)

Developers reported experiencing similar difficulties when they tried to address some of the privacy issues reported to them by Google Play [111]. They explained being uncertain about why a privacy issue reported by Google Play [111] existed in their apps, what action to take to fix it, and how to confirm that they successfully fixed it. These challenges were particularly relevant to privacy issues that were introduced by third-party SDKs that developers embedded in their apps without sufficiently understanding their data collection and sharing practices. This led developers to think that some of the reported issues did not exist in their apps and that they received the corresponding notifications from Google Play [111] by mistake. Two participants discussed their experiences:

"For data safety things, we received a couple of rejections from the play store and could not get to understand what the problem was." (*P*7)

"They claim that I use AD_ID which I do not, my manifest file absolutely does not use that." (*P*9)

Participants expressed the need for interactive forms of privacy guidance that allow them to ask specific questions without having to search through content that might not be relevant to them. These include chatting with experts in privacy compliance or using ticketing systems that are specifically designed to support developers who cannot afford to hire legal advisors.

Having such alternative forms of privacy guidance would also allow developers to address privacy issues during their early phases of development, which they deemed to be easier than having to remediate privacy issues upon receiving a notification from Google Play [111] long after their apps have been released. Two explained the types of privacy guidance they wanted to have:

"It would be great to have some systematic external tool that is regularly updated and which we know we fall into to make sure that we are not missing anything." ($P5$)

"If there is a tool that can help us learn the basic points that we have to take care of for privacy policies, that would be helpful." ($P7$)

Interactive Privacy Checklists: Feedback and Reactions

In the following, we summarize the results obtained after observing developers while they were using our interactive tool to generate privacy checklists for their app(s) and evaluate the extent to which this developer intervention would be effective if it were to be used in practice.

Privacy Questionnaire

For developers to be able to utilize an interactive privacy checklist, they would need to provide accurate answers to questions that have direct relevance to their privacy compliance obligations. During the interview sessions, we asked developers to think-aloud while they were answering the questions included in our privacy questionnaire and indicate whether they were able to accurately answer them. The questionnaire asked developers about the countries where their apps were available to download by users, types of data transmitted to first or third-party servers, Android permissions declared in their apps, and the third-party advertising or analytics SDKs embedded in their apps (Section 7.2). It also asked them to indicate the versions of each of the third-party advertising or analytics SDKs used in their apps, and provide additional details on whether they targeted children, whether they prepared privacy policies for their apps, and whether they used an ad mediation SDK [7] in their apps.

Most developers were able to answer all the questions included in the privacy questionnaire except the questions that relate to the types of data shared with third parties. Participants found it challenging to list all the types of personal data collected by third-party SDKs, and specifically persistent and non-persistent identifiers (Figure E.2). A number of participants also either could not recognize specific identifiers (e.g., app set IDs) or did not understand the difference between different types of identifiers. When answering the question related to shared data types (Figure E.2), some of them did so based on the permissions declared in their apps (e.g., `ACCESS_FINE_LOCATION` and `AD_ID`), or tried finding "Device IDs" in the answer options, which is the term used in Google Play's specification for data safety labels [51]. Participants expressed being uncertain about their apps' data sharing practices while they were filling our privacy questionnaire:

"I do not know like what Google Analytics would collect from an app but probably coarse location." ($P9$)

"I hope I am not sending WiFi MAC addresses, but in my app, I have a permission for accessing WiFi.." ($P11$)

Independent developers found the questions that asked them to list the Android permissions declared in their apps and the third-party SDKs integrated in their apps particularly easy to answer, since they just had to refer to their apps' Android manifest files to find declared permissions or look at their apps' dependencies. Two of those who were part of development teams could not list all the permissions declared in their apps and mentioned that they would need to consult others in their organizations to get these code details about their apps. However, they indicated that if the tool were to be provided to them, they would use it as a team (e.g., lead developer and their managers together). Furthermore, one participant who worked for a development organization in India could not accurately specify the audiences targeted by their app or the countries where their app is available. They explained that such details were not shared with them and that they were separately handled by their deployment team, who was responsible for making such decisions before uploading their apps to the store.

Willingness to Use Privacy Checklists

Most participants who tried our interactive checklist tool indicated that they would be willing to use such a tool if it were to be provided to them by the Google Play Store [111] or an independent third party. They appreciated the idea of having relevant privacy guidance provided to them in one place without needing to consult various sources in order to identify applicable privacy compliance requirements. They found the privacy guidance related to third-party SDKs integrated in their apps particularly helpful, which was presented as checklist items that listed disclosures that should be added to their privacy policies and relevant privacy configurations. Participants mentioned:

"It will be really helpful if there is a tool like a checklist to help me tick like I did this, this, this, it will be really nice." ($P2$)

"We are good as long as the checklist is not very lengthy, but if it is a very lengthy checklist, then it will be the same as the other policies we have, no benefit." ($P6$)

"I think this could save people a lot of time, effort, money, headache and possibly even legal problems. It is a global legal landscape to navigate, it is an impossible task for us." ($P9$)

However, we also observed that some developers were only interested in following privacy guidance related to compliance requirements that were enforced by Google [177]. For example, one participant explained why they believed that they did not need to follow all the guidance that was presented to them by the privacy checklist they generated during the interview session:

"If whatever you are telling me is not shown to me here [the Google Play console page], I am not going to take any action." ($P1$)

Our results also suggest that developers prefer privacy guidance that allows them to take immediate action without having to do additional work or think about what to do next. For example, a number of participants suggested incorporating a feature for generating text for relevant disclosures that they can use in their privacy policies. Others indicated that they would not use privacy checklists that ask them to consult lengthy documentation (e.g., third-party SDKs' privacy policies).

A few participants also suggested incorporating features that allow reducing the number of checklist items either by prioritizing them according to importance or excluding items for which they do not need to take action. An interactive privacy checklist tool could, for example, incorporate functionality that allows identifying third-party SDK configurations that were already used correctly by apps and use that as a basis for excluding corresponding checklist items accordingly. They expected such usability improvement to also reduce the effort required to revisit privacy checklists before releasing updates for their apps on the store.

Observed Positive Effect

We asked the developers who used our tool to generate a privacy checklist for one of their apps to discuss with us what they learned from the privacy checklists they generated and whether they found them helpful. Most participants indicated that they were not familiar with all the privacy laws that might be applicable to their apps, and particularly US state laws. Since our tool also presented privacy guidance that is customized to a number of third-party advertising and analytics SDKs (Figured E.3 and E.4), many participants also realized during the interviews that they either needed to include additional disclosures in their privacy policies or data safety labels, obtain consent for sharing certain types of personal data with third parties, or use at least one additional third-party SDK privacy configuration. This was a consequence of developers not sufficiently exploring documentation of third-party SDKs to understand their data collection practices before submitting their apps to the store. A few participants explained:

"I got to know this today only, I thought they only collected advertising IDs, there is much more." ($P12$)

"A couple of information is new to me like you mentioned in this Google Admob coding portion that I need to use to restrict sharing of personal information." ($P7$)

"I need to add things about data sharing in my privacy policy. After knowing this, I would also answer the data safety label part differently." ($P2$)

However, a number of participants recommended adding details on how to follow the provided privacy guidance. This was particularly the case for checklist items that asked participants to consider obtaining user consent before collecting or sharing certain types of personal data. A number of participants also thought that they were obtaining user consent when they did not, either because they:

- considered Android permissions as consent prompts,
- assumed that users who downloaded their apps had already implicitly consented to all their practices which were disclosed in their app's data safety labels on the store, or
- believed that they are not responsible for the practices of third-party SDKs embedded in their apps.

Participants reacted to the guidance that asked them to obtain user consent before collecting or sharing users' personal data by saying:

"But they already gave me their consent because my data safety section mentions everything which is going to be shared before they installed it." ($P1$)

"If I indicated that my app asks for AD_ID permission, then why is it asking me to obtain user consent?" ($P7$)

"If you show me a dummy app with a consent screen that would allow me to understand simply what should be implemented." ($P11$)

Relevance of Checklist Content to App Developers

Developers found the technical content included in the generated checklists relevant to their work responsibilities (e.g., privacy guidance on third-party SDK configurations and consent screens). However, we observed that not all developers who were part of organizations (i.e., not working as independent developers) considered information related to applicable privacy laws, privacy policies and data safety labels relevant to their responsibilities. Despite that, some of them indicated that having these details in their checklists would still allow them to have better discussions about privacy with other teams that are in charge of these tasks (e.g., legal or marketing teams). A few others preferred to exclude content not relevant to their responsibilities from privacy checklists. However, some of the independent developers preferred to use a comprehensive privacy checklist that can be used as a one-stop solution that concisely presents all privacy guidance relevant to their apps. Participants explained which type of privacy guidance would be relevant to their responsibilities:

"I think that privacy policy and data safety stuff are mostly relevant to the marketing team." ($P3$)

"They would review it and then let us know if we are missing anything or if we are not describing things accurately. So developers are involved in reviewing our privacy policy and data safety label." (*P*5)

"I would even look for more information. To me, this information could be extended with even more details and more suggestions." (*P*11)

Privacy Guidance in Development Processes

We asked participants about the phases of their development processes where they would be most likely to make use of privacy guidance. We found that most participants would either do so once they are ready to make new releases of their app(s) or upon being instructed by the Google Play Store [111] to address a privacy issue detected in their apps. Participants mentioned:

"You want to create your app, spend like 90% of your time to find technically how you can implement your stuff, and this is like 1% you left to the end, you want to go through it as quickly as possible." (*P*2)

"We have to first follow requirements provided by customers. Privacy is given secondary priority." (*P*7)

"That is going to be my approach that okay if Google complains that this thing is missing in the privacy policy, I am going to put it then." (*P*1)

Shortcomings of Privacy Checklists

Based on our observations, we identified a number of shortcomings of privacy checklists as a developer intervention. First, developers might make certain assumptions about how their apps would be used by users, and if such assumptions are not considered in the design of the interactive privacy checklist they are using, this could lead to presenting them with privacy guidance that might not be applicable to their apps. For example, presenting privacy guidance on implementing an age gate might not be applicable to a developer of a mixed-audience app that treats all users as children. Second, our interactive privacy checklist tool would not help developers who cannot provide answers to the privacy questionnaire. For instance, one participant mentioned that they bought some of their apps off-the-shelf, and do not have complete understanding of their data practices. Third, organizations tend to have different team structures and development processes, which would make it challenging to design a privacy checklist generation tool that can help all types of organizations.

Design Considerations for Privacy Guidance

The feedback we obtained from app developers on the two versions of our privacy checklist allowed us to obtain improved understanding of how to design usable privacy guidance for developers. In the following, we discuss a number of design considerations that we expect to help researchers design more effective privacy guidance.

Affordable

It is evident that a large percentage of developers are either working independently or as part of organizations that cannot afford to hire legal advisors. Such developers are also incentivised to release their apps in many countries to grow their businesses, which would increase the number of privacy regulations applicable to them. Given their lack of resources and legal expertise, many of them are currently relying on Google Play's enforcement of policy requirements only, which might not be sufficient for fulfilling their compliance obligations under applicable privacy laws.

Our results showed the need for affordable tools that provide comprehensive privacy guidance without requiring developers to consult many sources to identify applicable privacy requirements or understand how to address privacy issues that affect their apps. Such tools could also incorporate functionality that make it easier for developers to keep track of how changes in the privacy compliance landscape affect their privacy compliance obligations. For example, one participant explained why they would be willing to use a tool like ours:

"I would definitely use it because it streamlines all the checks, so it is a central place to check everything." (*P*8)

Concise

For privacy guidance to be usable, it has to allow developers to take proper actions without having to dedicate significant amount of time to understand how to do so. As we showed in the design of our interactive privacy checklist, one approach to reduce the length of documented privacy guidance is to present developers with guidance that is customized to their apps. For example, our results showed that developers appreciated that our tool listed the types of personal data collected by third-party SDKs embedded in their apps and did not just ask them to consult relevant SDK documentation.

Organizations could also further reduce the length of privacy checklists by customizing them to each of their teams to exclude content that is not relevant to their responsibilities. Another approach is to include additional features in the tool that allow identifying privacy compliance requirements that developers have already satisfied so that developers can only be presented with those that they need to take action on. This would allow further reducing the length of privacy checklists and make it easier for developers to regularly revisit them. One participant explained why they did not use existing sources of privacy guidance:

"We do not have much time to dedicate to privacy and data safety, even from the organization itself." (*P*7)

Easy to Understand

Our results highlighted the need for making privacy compliance requirements easy to understand for developers. Many app developers are not only unable to dedicate time to researching how to comply with applicable privacy regulations, but also have various educational levels and might not be fluent in English. There is therefore a need for simplifying existing privacy guidance to make it possible for developers who come from different backgrounds to understand how to comply (e.g., versions of privacy guidance that provide explanations of legal terms or use simpler language structures). A number of participants also recommended using visuals for presenting privacy guidance and experimenting with how to structure privacy guidance in a matter that would make it easier for developers to comprehend and follow. We also observed that developers were more familiar with privacy terminology that is used in Google Play's policies [177], which could suggest that they would be more likely to understand privacy guidance that uses the same terminology. One participant explained why they thought that privacy guidance should use Google Play's [111] terminology:

> *"I think it is important to either have one to one mapping, like have it the same in the console, so that I do not spend time to translate it from documents to the console or that it is easy to read."* ($P11$)

Actionable

In order for developers to follow privacy guidance, they should be provided with sufficient detail that allows them to understand why a privacy requirement is applicable to them and how to take correct action. Not doing so could lead them to take incorrect actions or believe that they are compliant when they are not. For example, one of the main findings of our interviews was that developers had misconceptions about how to obtain user consent, which led some of them to believe that the disclosures they had in their privacy policies or data safety labels are considered valid mechanisms for obtaining user consent under applicable privacy regulations.

The language used to frame privacy guidance could also lead developers to not understand what they are supposed to do to take action. The goal of designing usable privacy guidance is to allow developers to identify specific privacy tasks and act on them in a timely manner. We showed that using generic languages for framing privacy guidance led many developers to struggle with understanding their compliance obligations. For example, designers of privacy checklists should avoid including guidance that asks developers to check what could be considered personal data under applicable privacy laws, as many of them would find it challenging to do so. Actionable privacy guidance should instead answer such types of questions for developers and at the same time allow developers to identify the specific changes that they need to make to their apps (e.g., removing certain third-party SDKs or adding specific disclosures to their privacy policies). Whenever possible, it should also provide developers with a number of possible approaches that can be followed to address privacy issues to increase the likelihood of addressing privacy issues in case developers find themselves unable to follow a specific recommended approach.

Privacy guidance might not be considered actionable for developers if it does not provide sufficient detail that allows them to understand how important it is to follow certain privacy compliance requirements and prioritize privacy tasks accordingly in their development processes. Having such guidance available to developers early in their development processes is also important for allowing developers to take correct actions on detecting and addressing privacy issues that exist in their apps. Our results suggest that developers struggled to address privacy issues that were detected by Google Play [111] long after they released their apps. This was due to various reasons, which include forgetting details about their apps or having to dedicate their time to developing new apps instead. These are usability-related factors that we believe should be considered when designing privacy guidance aimed to help developers better understand their compliance obligations and make it easier for them to address privacy issues before making their apps available on app stores.

Describes Expected Outcome

After following privacy guidance, developers should have sufficient understanding of the extent to which they fulfilled their compliance obligations. This is particularly relevant to documented privacy guidance that allows developers to holistically think about their apps' practices, such as privacy checklists. Developers who need such types of guidance would likely not be able to hire experts who can audit their apps for compliance. We therefore expect providing them with a mechanism to evaluate their progress on their privacy compliance obligations to be helpful. For example, one participant explained the outcome they would like to have after using an interactive privacy checklist tool like ours:

> *"After using your checklist, I should be able to say to myself 'I am good, I am safe,' so my app can bring me money and I do not need to do anything."* ($P11$)

Privacy Guidance and Organizational Processes

Lack of proper support for privacy in organizational processes would likely discourage developers from following privacy guidance. This was one of the reasons that led developers who participated in our study to believe that they did not need to comply with applicable privacy regulations and not dedicate resources to understand how their work can affect the extent to which they are compliant. For example, one participant explained why they believed that they did not need to use a privacy checklist:

> *"These laws do matter, but I am working on a small-scale company where they might even not know what these are."* ($P12$)

In other cases, developers explained that lack of information sharing between relevant teams impacted their understanding of privacy compliance requirements. While their coding responsibilities were of direct relevance to privacy compliance (e.g., embedding third-party SDKs that collect personal data), they did not have privacy-focused discussions with the teams that were responsible for preparing their privacy policies and uploading their apps to the store. Lack of privacy-focused discussions with teams whose work is relevant to privacy compliance is likely to lead to privacy issues, such as including inaccurate disclosures of data practices in privacy policies.

On the other hand, participants who worked for organizations that provided them with privacy checklists demonstrated higher levels of awareness of their compliance responsibilities. This also made it easier for developers to have privacy-focused discussions with their managers and mutual understanding of how to follow privacy compliance requirements. For example, one participant explained why they had regular meetings with other developers to discuss the content of a privacy checklist they were using:

"They [app developers] might not understand the importance of it, they might misinterpret things that are written in there, so at least from our experience it has been better to have a discussion in addition to the list." ($P5$)

For privacy guidance to be effectively used in organizational contexts, organizations might also need to define how such guidance should be used by developers (e.g., defining the phases in their development processes where privacy checklists should be used and deciding on who should be responsible for checking whether they were followed correctly). They might also need to assign some of their personnel the task of customizing privacy guidance provided to them to suit the needs of their development processes.

Results of Lawyers Survey

Of the 14 lawyers who responded to our survey, 57% worked at law firms that consulted with organizations that develop mobile apps and 29% were part of legal teams within organizations that develop mobile apps. Thirty-six of respondents specialized in providing security or privacy-focused consultations and 50% of them had experience providing advice on privacy or security matters, but it was not their main specialization. Furthermore, 93% and 57% of them advised app developers who operate in the United States and Europe, respectively. This included advising them on how to comply with GDPR (86%), CCPA (79%), and COPPA (64%).

Eleven respondents indicated that they provided their clients with documents that contained privacy guidance customized to their needs. They indicated providing privacy guidance in the following formats:

- Checklists (57%)
- Templates for privacy policies or consent screens (57%)
- Descriptions of the requirements of applicable privacy laws (50%)
- Pointers to online resources (43%)

- Videos explaining privacy compliance (21%)

For those who used checklists, we asked them whether they used the checklists available in law database, such as those provided by Bloomberg [39] Law or Westlaw [221]. None of the respondents used these checklists as they are and five respondents indicated that their checklists were based on the checklists provided in these databases. Additionally, two respondents developed their own checklists, and one mentioned that they used checklists available in law databases before but that they are not using them anymore. When asked about the reasons why they did not rely on the checklists available in law databases, respondents indicated that they found them to be generic and needed to prepare checklists that are customized to their business needs. One respondent explained:

"The Bloomberg Law and Westlaw ones weren't always accurate, updated, or were too generic for the specific issue at hand." ($R1$)

As for when their clients used the provided privacy guidance, 10 of the 14 respondents indicated that their clients used it right before release or during the testing phases of their development processes. In terms of the content of the provided guidance, only 8 respondents provided guidance that was specifically on protecting children's privacy. Our results also suggest that lawyers focused on certain topics in their privacy guidance, such as explaining what could be considered personal data under applicable privacy regulations (79%) and how to opt users out from data collection (71%).

However, we also find that the privacy guidance that was provided by many respondents did not sufficiently address compliance responsibilities that are specifically related to use of third-party SDKs. For example, only 4 and 3 of the 14 respondents helped developers with vetting third-party SDKs for inclusion in their apps or configuring SDKs for privacy compliance, respectively. When asked about the guidance that was specifically on how to design consent screens, we found that 86% of respondents prepared the content of consent screens for their clients. However, we also find that only 6 respondents covered how consent screens in child-directed apps should be designed or when to use age gates in their documented privacy guidance.

The lawyers who participated in our survey also focused primarily on helping their clients with preparing their privacy policies. Other tasks were considered part of their clients' responsibilities (e.g., configuring third-party SDKs for compliance and preparing data safety labels for their apps).

As for the types of follow-ups that happen after lawyers provided privacy guidance, our results suggest that most of them did not follow a formal process in their follow ups (e.g., they only check by email if their clients have questions). To check whether clients followed provided guidance correctly, 43% respondents indicated that they check privacy policies afterwards, 36% indicated that they reviewed their clients' apps, and only 14% used tools to audit their clients' software as part of that checking.

Respondents explained a number of challenges that they experienced in their interactions with app developers. One of which was that developers were not always able to understand the guidance provided to them by their legal advisors. This becomes particularly the case when developers are busy with other development tasks that might be considered of higher priority. For instance, one respondent explained:

"There is often more than the team [the development team] can digest in a reasonable amount of time." ($R3$)

Another challenge was developers' lack of understanding of the behaviors of their own apps (e.g., types of personal data included in outbound transmissions). This could hinder effective communication between developers and their lawyers (e.g., discussions about disclosures that need to be included in privacy policies) and could ultimately lead to the persistence of privacy issues in apps.

Respondents also indicated that developers did not follow the provided guidance when they were pressured by their business people (or other teams) to focus on other tasks that they deemed to be of higher importance. One respondent explained such types of tensions:

"At larger clients there is tension between legal and marketing, and developers are often caught in the middle. But marketing pays the development salaries, not legal." ($R6$)

Respondents expected increasing the frequency of meetings with developers to improve the effectiveness of their interactions with them and recommended providing developers with privacy guidance containing detailed information on how to identify and address privacy issues. One respondent explained:

"I would make it more like an instruction manual. That's what most developers ultimately want. " ($R14$)

7.4 Discussion

The variety of software engineering tasks that have privacy implications motivated researchers to propose various types of solutions that allow incorporating privacy guidance in software development processes. These include IDE plugins that point developers to privacy issues while coding [142, 162], privacy policy generators [227, 228] and tools for generating privacy labels [143]. While many of these interventions were shown to be effective at providing better support for privacy in development processes, they were mostly focused on helping developers address specific types of privacy issues, and might not be sufficient for allowing developers to reason about their privacy compliance obligations holistically. In this research study, we focused on studying how to design usable privacy checklists in an effort to take one step in this direction to understand how to help developers think holistically about their apps' practices. We showed that developers expressed willingness to use privacy checklists that satisfy certain usability requirements and demonstrated the need for such types of interventions.

Lessons learned from the usable security literature can be used to inform the design of usable tools that provide developers with actionable privacy guidance that can be used for privacy compliance purposes. Given that many developers would only be able to dedicate limited resources to privacy compliance in their development processes, one of the main design requirements of such tools is to help developers understand where to focus their limited attention when trying to address privacy issues during development. One step towards that end is to reduce the effort required to identify privacy issues that need to be addressed by developers. This could be achieved by providing developers with usable privacy checklists that can be customized according to their needs.

In this research study, we presented a prototype implementation of a tool that generated customized checklists based on a set of parameters (e.g,, embedded third-party SDKs and targeted countries). We believe that there are promising areas for improvement, which include adding features for app testing and generating text for specific disclosures that developers can include in their privacy policies. Interactive privacy checklists could also be composed of sub checklists that can further customize privacy guidance based on specific parameters. These could include sub checklists for helping developers understand:

- how to accurately determine audiences targeted by an app,
- how to vet third-party SDKs before integrating them,
- how to determine whether a US state law, such as the CCPA, is applicable, and
- how to comply with app market policies.

In order for tools that provide developers with usable privacy guidance to serve their purpose, they should also be built by experts in privacy compliance who are capable of translating privacy requirements into actionable tasks for developers. This would require those experts to have technical and legal expertise that allows them to build resources targeted to helping developers who have certain needs (e.g., independent developers who publish apps on the Google Play Store [111]). Their roles would involve:

- studying the requirements of privacy regulations and app market policies,
- researching and auditing the practices of third-party SDKs, and then
- building a centralized resource that allows providing developers with actionable and up-to-date privacy guidance accordingly.

As we showed in our results, many developers were only able to find generic privacy guidance that they could not translate into actionable tasks and were also unable to consult legal advisors. This further emphasizes the importance of providing developers with more effective types of privacy guidance. Thus, we hope that building a centralized resource that developers can consult would at least help developers avoid common privacy issues that we identified in our prior work (see Chapters 5 and 6).

We expect having such a resource to also allow developers who are able to consult legal advisors to have better discussions with them about their privacy compliance obligations. The results of our survey show that developers' lack of understanding of the behaviors of their own apps hinder the quality of their communications with their legal advisors. This finding further highlights the need for providing developers with tools that allow them to better understand their apps' data practices. Utilizing such tools in development processes could help bridge the communication gap that currently exists between developers and their legal advisors, which would ultimately address many types of privacy issues early in development processes (e.g., disclosure mistakes in privacy policies that are caused by lack of understanding of the behaviors of embedded third-party SDKs).

The increasing number of privacy regulations that are being enacted coupled with the frequent updates made to app market policies and third-party SDKs would continue to further increase the burden of compliance on developers. As our results show, many developers currently are either only considering privacy requirements that are enforced by Google Play [111] or relying on privacy guidance that is not comprehensive. They are even building internal resources for privacy guidance based on their past experiences with Google Play's enforcement actions only, which is leading them to rely on incomplete privacy guidance that does not consider all the requirements of applicable privacy regulations.

Many developers are also referring to online forums for privacy guidance [193, 204], which is also unlikely to lead them to have sufficient understanding of their privacy compliance obligations. For instance, when we asked a participant to explain why their app's data safety label disclosed sharing device identifiers with third parties, they explained that they followed advice provided on an online forum to resolve a privacy issue that was detected by Google Play without understanding that their app actually embedded a third-party SDK that collected device identifiers. We also showed how existing sources of privacy guidance led developers to incorrectly believe that they were compliant with certain privacy requirements when they actually were not (e.g., considering Android permissions as mechanisms that satisfy consent requirements of applicable privacy regulations). Such knowledge gaps and misconceptions that developers have about their privacy compliance obligations can be addressed by improving the quality of privacy guidance that is reaching them, which could therefore be achieved by building resources dedicated to providing developers with usable privacy guidance.

While there is clearly a need for building a centralized resource that offers actionable privacy guidance, it is unclear whether it should be built by an independent third-party, app markets such as the Google Play Store [111], or regulators. Given that our research consistently showed that developers are improving their practices based on Google Play's enforcement efforts, app markets seem to be best-positioned to provide actionable privacy guidance to developers. They could provide it through their developer consoles or centralized resources that are dedicated to helping developers with their privacy compliance responsibilities. In the process of building such a resource, cooperation of third-party SDK providers might be needed to understand how their practices might introduce privacy compliance issues and inform the design of privacy guidance accordingly. As we showed in Chapter 6, embedding third-party SDKs which were not sufficiently documented was one of the main challenges that led to introducing various types of privacy issues (e.g., inaccurate disclosures in privacy labels). For designers of privacy guidance to be able to help developers address this challenge, they might therefore need to thoroughly research the practices of third-party SDKs that are commonly used by app developers, which could require performing technical audits of these SDKs and cooperating with their providers to further understand their practices.

App markets and providers of third-party SDKs could also play a major role in reducing the operational overhead associated with privacy compliance requirements. Currently, developers who release their apps on different app markets have to learn about how to comply with their different platform policies, which increases the burden of compliance on developers. Similarly, developers are expected to correctly use different privacy configurations for each of the third-party SDKs embedded in their apps, which increases the likelihood of introducing privacy misconfigurations. Standardizing the design of third-party SDK privacy configurations and ensuring consistency of policies introduced by different app platforms could therefore contribute to reducing the burden of compliance on developers.

7.5 Conclusion

In this chapter, we shed light on the types of privacy guidance that are currently being relied upon by software development organizations and the reasons why they fall short at helping developers address privacy issues before releasing their apps on app stores. Our findings demonstrate the need for usable forms of privacy guidance that can help developers identify and address privacy issues before publishing their apps on app stores. The results of an anonymous survey that we targeted to lawyers show that providing this guidance could additionally help developers have more productive discussions with their legal advisors on how to fulfill their privacy compliance obligations.

In order for the provided privacy guidance to serve its purpose, it has to not introduce more burden on development organizations and allow informed decision-making on how to fulfill applicable privacy compliance requirements from the early phases of the development process. This can be achieved by providing it in a central place that allows developers to tailor it to their development processes and workflows. We experimented with designing a privacy checklist aimed at helping developers understand how to comply with privacy regulations and platform policies applicable to their apps. The process of designing this checklist allowed us to identify a number of design requirements that we hope will pave the towards understanding how to present privacy guidance to software developers.

Chapter 8

Conclusion

Software professionals play major roles in helping their organizations address various types of security and privacy issues throughout their software development or maintenance processes. Within organizational contexts, their security and privacy practices are shaped by pre-defined organizational processes and their interactions with various technical and non-technical stakeholders (e.g., software developers and organization leaders). This dissertation showed that shortcomings in organizational processes negatively impacted security and privacy engineering practices. It also demonstrated the extent to which regulatory and policy efforts can lead to changes in the security and privacy practices of professionals involved in these processes.

8.1 Role of Organizational Processes

Investigating organizational processes followed to handle security vulnerabilities and fulfill privacy compliance requirements allowed us to identify common challenges that are experienced by security and privacy professionals involved in these processes. A common characteristic between these processes is that they task professionals with identifying or addressing issues that affect software under their control. Contrary to other types of software flaws, security and privacy issues might emerge at any point of the software development process and handling them might require the involvement of various internal and external stakeholders (e.g., third-party developers). In order for organizations to be prepared to handle any security or privacy issue that might affect them, they would have to have processes that define the responsibilities of involved stakeholders and allow them to communicate effectively. While it is important for such processes to have pre-defined workflows that streamline coordination between involved professionals, they should also provide professionals with the flexibility they need to address issues that would require following different workflows (e.g., new types of security vulnerabilities or privacy compliance requirements).

Shortcomings in Organizational Processes

Software professionals struggled to handle privacy or security issues when their organizations did not have structured workflows for such tasks (Chapters 3, 4, 5, and 7). In Chapters 3 and 4, we show how lack of pre-defined processes for handling security vulnerabilities was one of the reasons that explained why vulnerabilities remained unfixed. For instance, lack of such workflows led security professionals to be uncertain about how to handle vulnerabilities detected in third-party code or validate the effectiveness of vulnerability fixes.

Another common challenge was having to follow processes that require professionals to decide on how to address security or privacy issues based on limited information. For instance, the results of our investigation of vulnerability remediation processes in Chapter 4 showed that security professionals did not always have up-to-date information about their technical infrastructures, which made it challenging for them to understand how discovered vulnerabilities affected their organizations and decide on whether to remediate them. Additionally, through the investigations presented in Chapters 4 and 5, we found that lack of sufficient detail about security or privacy issues that were reported by external software testers or the Google Play Store [111] led software professionals to neglect addressing them.

Organizational processes could also negatively impact software professionals' abilities to handle security or privacy issues if they do not facilitate coordination and information sharing between the various roles who are involved in these processes. This could be experienced when organizational processes do not support direct communication between those involved in handling security or privacy issues (e.g., security engineers and software developers). Throughout our studies, ineffective communication was a common challenge experienced by technical and non-technical stakeholders. This was caused by shortcomings in organizational processes (e.g., lengthy communication cycles), differences in risk perceptions (e.g., disagreements on severity levels of discovered vulnerabilities) or interactions with stakeholders who do not have the same skill sets or backgrounds (e.g., security engineers and business leaders). In Chapter 4, we show that lack of direct communication between security testers and software developers led developers to release incorrect fixes or introduced delays in the remediation process. This observation is also supported by the results of the studies presented in Chapter 7 which showed that developers whose organizational processes did not allow them to communicate with their legal teams struggled with translating privacy compliance requirements to actionable tasks in their development processes.

Furthermore, the findings of the studies presented in Chapters 3, 4, and 5 show that most organizations adopted reactive rather proactive approaches to addressing security and privacy issues in software development processes. In Chapter 5, for example, we found that developers only focused on addressing privacy compliance issues that were flagged as policy violations by the Google Play Store [111], long after their apps were published. However, our findings also identify a number of factors that led to expediting progress in addressing security and privacy issues. In Chapter 4, we found that organizations prioritized remediating security vulnerabilities when they believed that not doing so would affect their compliance with applicable industry standards or lead them to lose relationships with customers who were using their software. Similarly, the results of the studies presented in Chapters 5 and 7 show that developers focused on addressing privacy issues that might lead them to be in violation of Google Play's policies [177] or lose business relationships.

Through the qualitative investigations presented in Chapters 3, 4, 5, and 7, we also found evidence of organizations' reliance on feedback received from external parties (e.g., the Google Play Store [111] and external security testers) for validating whether they addressed security and privacy issues effectively. The large degree of trust that organizations placed in the feedback received from these parties could suggest that they lacked internal processes for validating the effectiveness of their efforts or understanding of how to do so.

Opportunities for Improvement

One promising direction for future work is to obtain improved understanding of the communication needs of the various stakeholders involved in security and privacy processes and develop interventions that allow them to communicate more effectively. For privacy compliance processes, such interventions could make it easier for software developers and their legal advisors to make informed decisions on issues that require technical and legal expertise at the same time. Examples of such decisions include identifying the set of third-party SDK configurations that developers need to use to comply with all applicable privacy regulations and understanding how to disclose the practices of embedded third-party SDKs in privacy policies. For organizations to support communications between stakeholders involved in security and privacy decision making, they might also need to make organizational process improvements that facilitate information sharing between involved stakeholders and allow them to have actionable discussions on how to address privacy and security issues. This could involve keeping development teams updated on their organizations' privacy compliance obligations and clearly communicating to them the types of privacy issues that should be avoided during software development.

There are other types of organizational process improvements that we expect to make it easier for software professionals to follow structured approaches to handling security and privacy issues. These include adopting better software documentation practices, improving reporting structures between engineering and managerial roles, and defining approaches for tracking progress on how security and privacy issues were handled. Adopting these practices could also reveal opportunities for defining indicators that allow organizations to evaluate the efficiency of processes followed by their professionals to identify and address security and privacy issues (e.g., time taken to remediate security vulnerabilities). This could also allow identifying factors that are limiting the effectiveness of organizational security and privacy processes (e.g., not receiving patches from certain third-party software vendors in a timely fashion and having to deal with vulnerable dependencies that are part of organizations' software supply chains).

8.2 Role of Third Parties

This dissertation demonstrated how various types of third parties can influence organizational security and privacy practices. These include third-party SDK providers, app stores, and external security or privacy consulting firms. Given that we paid particular attention to investigating compliance of Android apps with privacy regulations in Chapters 5, 6, and 7, the following sections discuss how the Google Play Store [111] and third-party SDK providers can contribute to making it easier for organizations to improve their privacy practices.

Third-party SDK providers

Use of third-party SDKs is leading software developers to introduce various types of privacy issues that are rendering them in violation of applicable privacy laws and platform policies (Chapters 5, 6, and 7). The technical analyses reported in Chapters 5 and 6 show that improper use of these SDKs led to disclosure mistakes, enabled sharing of children's personal data, and gave third parties the ability to bridge AAID resets. However, we also found evidence that developers are currently struggling to understand how their usage of third-party SDKs is affecting their compliance with applicable privacy regulations and policies (Chapters 5, 6, and 7). Lack of awareness of the data practices of third-party SDKs is also leading developers to incorrectly believe that their apps are compliant with applicable privacy compliance requirements (Chapters 5). In Chapter 6, we also observe that improper use of third-party SDKs is hindering effectiveness of recent industry efforts that aim to protect user privacy (e.g., app set IDs [76], Google Play's data safety labels [51], and the `AD_ID` permission [35]). This is caused by third-party SDK providers not being completely transparent with developers about their data practices or requiring developers to read documentation that are not kept up-to-date or not usable [136, 203].

To comply with applicable privacy regulations and platform policies, app developers are expected to vet third-party SDKs for inclusion in their apps, configure them for privacy compliance, and have sufficient understanding of their data practices. Third-party SDK providers can play a major role in simplifying these tasks for app developers. Addressing the limitations of third-party SDK documentation could require reducing their length or using better structures for presenting information related to SDK data practices (e.g., types of personal data collected, third parties who might receive collected data, and details on how collected data is handled).

There are also opportunities for making design improvements to third-party SDK privacy configurations to reduce the effort required to use them correctly. Our investigation of these configurations in Chapters 5 and 6 showed that third-party SDKs varied in terms of how their privacy configurations were designed, which made it challenging for developers to choose the set of privacy configurations that were applicable to their apps and configure them correctly. Standardizing the design of these configurations is therefore likely to reduce the operational overhead associated with using them and help reduce the burden of compliance on developers. Another opportunity for improving the design of these configurations is for SDK providers to allow propagating privacy flags across advertising networks for apps that use ad mediation technologies. This functionality is currently not supported by all third-party SDK providers (e.g., Google Admob [108]), which is contributing to increasing the effort required to configure SDKs for privacy compliance. The design of privacy configurations could also be improved by using privacy-preserving defaults and following data minimization principles while also giving developers the option to enable data collection when needed. While there is currently no economic incentives for them to do so, recent research showed that such types of design changes are likely to help developers improve their privacy practices [202].

The Google Play Store

The results presented in Chapters 5, 6, and 7 demonstrated the powerful impact that Google Play's policies [177] had on developers' privacy practices. In Chapters 5 and 7, we also found evidence that development organizations focused on addressing privacy issues that were reported to them by the store and designed privacy checklists based on their experiences with Google Play's ongoing enforcement efforts. However, in Chapter 5, we also show that developers were not always able to understand the content of Google Play's notifications, which negatively affected their progress on addressing privacy issues.

There are opportunities for Google Play [177] to further strengthen its enforcement efforts and better align them with the requirements of applicable privacy regulations. Google Play [177] could also supplement their enforcement efforts with tools that provide developers with usable privacy guidance that they can utilize throughout their development processes. This can be accomplished by simplifying existing developer policies, improving the content of e-mail notifications sent to developers upon detecting privacy issues in their apps, and providing developers with tools that allow them to audit their apps before submitting them to the store.

Given that developers are continuously found to struggle with understanding the behaviors of third-party SDKs embedded in their apps, Google Play [177] could also improve its app submission process to provide developers with a summary about the transmissions generated by their apps and the types of personal data included in these transmissions. This would give developers the opportunity to include missing disclosures in their privacy policies and/or data safety labels before making their apps available to users. Currently, developers are notified by Google Play [177] about privacy issues detected in their apps long after releasing them, which is introducing challenges in their development processes (Chapters 5). We therefore expect providing privacy guidance at an early point in app development processes to make it easier for developers to incorporate it in their processes and improve their apps' privacy practices accordingly.

Google Play [177] could also provide developers with real-time guidance that allows them to identify the third-party SDK configurations that they need to use in their apps. This could be accomplished by having third-party SDK providers coordinate their efforts with Google Play [177] to allow notifying developers about privacy configurations that they need to use in their apps, for example. Such types of integration would allow providing developers with actionable privacy guidance that can also help them understand the extent to which their use of third-party SDKs is affecting their compliance with platform policies and applicable privacy regulations.

8.3 Need for Actionable Guidance

Our qualitative investigations of the approaches followed by software professionals to handle security vulnerabilities and privacy compliance issues consistently demonstrated the need for actionable guidance that goes beyond known best practices. As discussed in Chapters 3 and 4, lack of actionable security guidance led to shortcomings in software professionals' abilities to understand how their organizations were affected by discovered vulnerabilities and remediate them in a timely fashion. This presents two opportunities for helping professionals with their remediation decisions, which include incorporating guidance in the outputs generated by vulnerability scanners and vulnerability reports received from internal or external security testers.

There is also a need for researching the factors that might lead such guidance to not be considered actionable by software professionals. For instance, development teams would probably be more likely to take action on vulnerabilities when they are provided with details on the various remediation plans that could be considered in case following a specific one is not technically feasible for them. In Chapters 3 and 4, we showed that this was one of the reasons that led organizations to delay remediating vulnerabilities reported by security auditing firms or bug bounty hunters. The findings presented in this dissertation also emphasize the importance of considering the capabilities of software professionals in the design of guidance aimed at helping professionals remediate security vulnerabilities effectively. Those who are responsible for remediation might not have sufficient expertise in cybersecurity (e.g., software developers who are part of small organizations) or have different specializations (e.g., offensive versus defensive security for red and blue teams, respectively).

Several factors are also contributing to the lack of actionable privacy guidance that can help developers fulfill their privacy compliance obligations (Chapters 5, 6, and 7). Our results suggest that many of these factors can be addressed by improving the usability of privacy guidance provided in government websites (e.g., the FTC [92]), mobile platforms (e.g., Google Play [111]), and third-party SDK providers. As we showed in Chapter 7, app developers were not willfully disregarding privacy compliance requirements but found navigating available sources of guidance to be challenging. Many of these sources included lengthy privacy guidance that was not tailored to the expertise of software developers (e.g., required special legal expertise).

Given that privacy compliance is likely to not be given high priority in development processes (particularly in small organizations), there is clearly a need for reducing the effort required for developers to understand their privacy compliance obligations and address privacy issues early in their development processes. This might include simplifying privacy guidance provided by existing resources to make it easier for developers to follow, providing developers with code examples that they can use as references in their development processes, and building centralized resources for presenting guidance on privacy compliance. Such types of improvements might contribute to reducing the number of decisions that software developers need to make when trying to fulfill their privacy compliance obligations, and thus reduce the burden of compliance on developers.

The findings presented in Chapters 4, 6, and 7 show that many of the software professionals who participated in our studies lacked understanding of how to test the security and privacy behaviors of their software. They also did not have access to software testing tools that allow them to do so (e.g., network traffic inspection tools). This led them to not address security or privacy issues that affected their software in a timely fashion. The observations reported in Chapters 7, for example, show that most software developers relied on Google Play's auditing system for identifying privacy issues that affected their apps and did not know how to do so early in their development processes. These findings highlight the need for providing developers with guidance and affordable software testing tools that can help them understand how to identify these issues early in their development processes and address them accordingly.

This dissertation also highlights the need for designing guidance that can help small organizations understand how to provide sufficient consideration for privacy and security in their development processes. These organizations have special needs that should be considered when designing actionable security and privacy guidance. In Chapters 4 and 7, we show that these organizations cannot afford to hire experts who can guide them through how to comply with privacy regulations or build internal security teams who can help them handle security vulnerabilities. They are also often only able to dedicate limited resources to security and privacy in their development processes. There is therefore a need for understanding the types of guidance that can make handling security and privacy issues less burdensome for these organizations.

8.4 Future Work

The results of our technical and qualitative investigations demonstrated the diversity of technical, organizational, legal, and social factors that are shaping software professionals' security and privacy practices. In the following, we discuss how future work can address some of the challenges identified in our studies and list a number of open questions that might pave the way towards further explorations of software professionals' security and privacy practices.

Third-party code

Use of third-party code can introduce many types of security and privacy issues. These types of issues are particularly difficult to handle by software professionals. Our research showed that not obtaining complete understanding of the behaviors of third-party SDKs is leading developers to inaccurately disclose their apps' practices in their privacy policies. In Chapter 6, we showed that use of third-party SDKs also negatively impacted the effectiveness of Google Play's specification for data safety labels [51], which was recently introduced to encourage more transparency in the mobile ecosystem. The complexity of handling third party code increases when developers of third-party SDKs are intentionally not being transparent about their security or privacy practices.

To help organizations that use third-party code address these challenges, there are three open questions that are worth investigating in future work:

- Would auditing third-party SDKs by independent trusted organizations lead to improvements in security and privacy practices?
- What types of interventions can be designed to help organizations vet third-party code?
- What approaches can organizations follow to identify security and privacy issues introduced by all third-party code included in their supply chains?

The Google Play Store

In Chapter 6, we showed the powerful impact that policy efforts by the Google Play Store have had on developers' privacy practices. App markets define their own policies, many of which are likely in alignment with the requirements of privacy regulations. Given the role of app markets in shaping organizational privacy practices, we believe that future work should investigate the following research questions:

- To what extent should they be involved in enforcing the requirements of applicable privacy regulations?
- How can their messaging systems be improved to provide usable privacy guidance to app developers?
- What types of collaborations between third-party SDK providers and app markets would be effective at reducing the burden of privacy compliance on app developers?

Economics

Software developers, app markets, and third-party SDK providers might not always have incentives to comply with applicable privacy regulations, particularly if privacy compliance requirements are not sufficiently enforced. The research presented in Chapters 5 and 6 particularly focused on studying privacy compliance processes followed by developers of child-directed apps, who are subject to more stringent privacy compliance requirements compared to other developer populations. The reliance of many of these developers on third-party advertising services to monetize their apps on app markets is in conflict with privacy requirements that prohibit sharing children's personal data with third parties without verifiable parental consent. Future work should therefore answer the following research questions:

- What types of interventions would be effective at incentivizing these developers to comply with applicable privacy regulations?
- Are there alternative monetization models that can be relied upon by developers of child-directed apps?

Third-party advertising and tracking services experience a similar challenge. In Chapters 5 and 6, we showed that embedding SDKs provided by these services enabled sharing of children's personal data in ways that might render app developers in violation of applicable privacy regulations. While providers of these SDKs offered privacy configurations that allow limiting data sharing, many of these configurations were not made usable to developers and sufficient developer documentation were not always available. Additionally, the default behaviors of many of the SDKs we tested were not privacy-preserving (i.e., data sharing was triggered upon SDK initialization). While SDK providers can improve the design of their configurations to be privacy-preserving by defaults, there is no economic incentive for them to do so. How can third-party SDK providers be encouraged to play an active role in reducing the burden of privacy compliance on developers?

Organizations are also utilizing open source libraries which can introduce series security vulnerabilities (e.g., the Log4j vulnerability [21]). While many of these types of libraries are widely used by organizations, they might not be sufficiently maintained by specific parties. What types of economic incentives would be effective at encouraging skilled security professionals to audit widely-used open source libraries to prevent potential security risks?

Communication

The survey and interview-based studies presented in this dissertation with software developers, testers, and managers showed that lack of effective communication between various roles is contributing to shortcomings in organizational security and privacy processes. Improving the quality of interactions between different engineering and managerial roles can help expedite progress on addressing security and privacy issues throughout the phases of the software development process. Given that such interactions might require the involvement of stakeholders who have different backgrounds and skill sets (e.g., might not have technical cybersecurity or privacy expertise), how can we use lessons learned from the usable security and privacy literature to bridge the communication gaps that exist between the various technical and managerial roles involved in these processes?

8.5 Concluding Remarks

Software professionals are tasked with various types of engineering and managerial responsibilities that directly impact their organizations' business objectives. Their responsibilities might involve discovering or addressing security or privacy issues as part of their organizational software development or maintenance processes. Shortcomings in their security or privacy practices could have serious implications, such as expanding their organizations' attack surfaces or exposing their organizations to legal liabilities. The increasing recognition of the societal implications of security and privacy issues will further increase the need for skilled professionals who are capable of helping organizations avoid potential threats.

The results of semi-structured interviews conducted with more than 120 software professionals, several research surveys conducted with the same populations, and technical tests of the practices of thousands of Android apps consistently identified non-technical factors that shaped organizational security and privacy practices. Two of the leading factors were the enforcement of policy requirements by app markets and the extent to which security and privacy practices are supported by organizational processes. We demonstrated differences in the types of challenges experienced by professionals who worked for small versus large organizations and evaluated the robustness of processes followed to address security and privacy issues in these organizations. We also showed that organizational vulnerability management and privacy compliance processes are people-centric, and thus lack of effective communication between the various roles involved in such processes can hinder informed security and privacy decision making by organizations.

One of the main takeaways of the studies presented in this dissertation is that actionable guidance on how to address security and privacy issues is lacking. This is particularly needed by small organizations, who lack internal security or privacy expertise and do not have the resources to hire external consultants. The identified challenges highlight the need for interventions that can guide professionals through how to make various types of decisions in their vulnerability management and privacy compliance processes. These interventions should also allow decision makers to think holistically about how the practices of their software can lead to introducing security or privacy issues. To audit Android apps for privacy compliance, for example, professionals would need to think about the nature of data transmissions generated by first-party code and embedded third-party SDKs, and at the same time understand how to properly disclose such practices in privacy policies and data safety labels. Interventions might also need to be customized by organizations to suit their processes and the types of interactions that take place between the various teams involved in these processes.

The industry is continuously proposing controls that aim to provide users with more control over their security and privacy. Third-party SDK configurations and Google Play's data safety labels are examples of such controls. While developer organizations are continuously expected to make adaptation to their development processes to add support for such controls in their development processes, they are not offered guidance on how to properly do so (e.g., accurately disclose their practices in their apps' data safety labels). The industry also seems to not be addressing the root causes of privacy and security issues before requiring developers to add support for more controls in their development processes. For instance, while privacy policies were shown to suffer from disclosure issues caused by developers' lack of understanding of the behaviors of embedded third-party SDKs, the industry failed to support developers in this task before finding that developers repeated the same types of mistakes in their apps' privacy labels. Addressing the root causes of developer mistakes could therefore help us make more meaningful progress in improving software professionals' security and privacy practices.

Bibliography

[1] Federal Trade Commission (FTC). *Hyberbeard FTC case.* `https://www.ftc.gov/legal-library/browse/cases-proceedings/192-3109-hyperbeard-inc`. [Online; accessed: 11-November-2024].

[2] Federal Trade Commission (FTC). *InMobi FTC case.* `https://www.ftc.gov/legal-library/browse/cases-proceedings/152-3203-inmobi-pte-ltd`. [Online; accessed: 11-November-2024].

[3] Federal Trade Commission (FTC). *Kuuhuub FTC case.* `https://www.ftc.gov/legal-library/browse/cases-proceedings/182-3184-kuuhuub-inc-et-al-us-v-recolor-oy`. [Online; accessed: 11-November-2024].

[4] *25th Privacy Enhancing Technologies Symposium (PETS 2025).* `https://petsymposium.org/`. [Online; accessed: 1-December-2024].

[5] Yasemin Acar, Michael Backes, Sascha Fahl, Simson Garfinkel, Doowon Kim, Michelle L Mazurek, and Christian Stransky. "Comparing the Usability of Cryptographic APIs". In: *2017 IEEE Symposium on Security and Privacy (SP).* IEEE. 2017, pp. 154–171.

[6] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. "You Get Where You're Looking For: The Impact of Information Sources on Code Security". In: *2016 IEEE symposium on security and privacy (SP).* IEEE. 2016, pp. 289–305.

[7] *Ad mediation.* `https://www.is.com/glossary/ad-mediation/`. [Online; accessed: 26-September-2024].

[8] AdColony. *AdColony's Privacy Settings.* `https://github.com/AdColony/AdColony-Android-SDK/wiki/Privacy-Laws`. [Online; accessed: 3-January-2024].

[9] AdColony. *AdColony's SDK (deprecated).* `https://support.adcolony.com/`. [Online; accessed: 17-November-2023].

[10] Adjust. *Adjust Services.* `https://www.adjust.com/`. [Online; accessed: 11-November-2024].

[11] Google Admob. *Targeting Configurations.* `https://developers.google.com/admob/android/targeting`. [Online; accessed: 11-November-2024].

[12] Google Admob. *U.S. states privacy laws compliance.* `https://developers.google.com/admob/android/privacy/us-states`. [Online; accessed: 11-November-2024].

[13] Google Admob. *User Messaging Platform SDK.* `https://developers.google.com/admob/android/privacy`. [Online; accessed: 11-November-2024].

[14] *Advertising Platform OpenX Will Pay $2 Million for Collecting Personal Information from Children in Violation of Children's Privacy Law.* `https://www.ftc.gov/news-events/press-releases/2021/12/advertising-platform-openx-will-pay-2-million-collecting-personal`. [Online; accessed: 24-January-2022].

[15] Omer Akgul, Taha Eghtesad, Amit Elazari, Omprakash Gnawali, Jens Grossklags, Michelle L Mazurek, Daniel Votipka, and Aron Laszka. "Bug Hunters' Perspectives on the Challenges and Benefits of the Bug Bounty Ecosystem". In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 2275–2291.

[16] Noura Alomar and Serge Egelman. "Developers Say the Darnedest Things: Privacy Compliance Processes Followed by Developers of Child-Directed Apps". In: *Proceedings on Privacy Enhancing Technologies* (2022).

[17] Noura Alomar, Joel Reardon, Aniketh Girish, Narseo Vallina-Rodriguez, and Serge Egelman. "The Effect of Platform Policies on App Privacy Compliance: A Study of Child-Directed Apps". In: *Proceedings on Privacy Enhancing Technologies.* [Accepted but not published yet]. 2025.

[18] Noura Alomar, Primal Wijesekera, Edward Qiu, and Serge Egelman. ""You've Got Your Nice List of Bugs, Now What?" Vulnerability Discovery and Management Processes in the Wild". In: *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. 2020, pp. 319–339.

[19] *Amplitude - Android SDK.* `https://www.docs.developers.amplitude.com/data/sdks/android/`. [Online; accessed: 05-July-2022].

[20] Android. *Google Mobile Services (GMS).* `https://www.android.com/gms/`. [Online; accessed: 11-November-2024].

[21] *Apache Log4j Security Vulnerabilities.* `https://logging.apache.org/log4j/2.x/security.html`. [Online; accessed: 26-October-2024].

[22] *App Store.* `https://www.apple.com/app-store/`. [Online; accessed: 4-November-2024].

[23] *AppCensus.* `https://appcensus.io/`. [Online; accessed: 24-October-2023].

[24] AppLovin. *Consent, Other Applicable Flags, and Data APIs.* `https://dash.applovin.com/documentation/mediation/android/getting-started/privacy`. [Online; accessed: 11-November-2024].

[25] AppLovin. *Support Center.* `https://dash.applovin.com/documentation/mediation`. [Online; accessed: 11-November-2024]. 2024.

[26] Appodeal. *Appodeal Android SDK.* `https://docs.appodeal.com/android/get-started`. [Online; accessed: 11-November-2024].

[27] *Article 6 of GDPR.* `https://gdpr-info.eu/art-6-gdpr/`. [Online; accessed: 02-January-2022].

[28] *Article 8 of GDPR.* `https://gdpr-info.eu/art-8-gdpr/`. [Online; accessed: 27-February-2022].

[29] Hala Assal and Sonia Chiasson. "'Think secure from the beginning' A Survey with Software Developers". In: *Proceedings of the 2019 CHI conference on human factors in computing systems*. 2019, pp. 1–13.

[30] Office of the Attorney General. *California Consumer Privacy Act (CCPA).* `https://oag.ca.gov/privacy/ccpa`. [Online; accessed: 11-November-2024].

[31] Andrew Austin and Laurie Williams. "One Technique is Not Enough: A Comparison of Vulnerability Discovery Techniques". In: *International Symposium on Empirical Software Engineering and Measurement*. IEEE. 2011, pp. 97–106.

[32] Rebecca Balebako, Abigail Marsh, Jialiu Lin, Jason Hong, and Lorrie Faith Cranor. "The Privacy and Security Behaviors of Smartphone App Developers". In: *Workshop on Usable Security*. Citeseer. 2014, pp. 1–10.

[33] Kenneth A Bamberger and Deirdre K Mulligan. *Privacy on the ground: driving corporate behavior in the United States and Europe*. MIT Press, 2015.

[34] Jason Bau, Elie Bursztein, Divij Gupta, and John Mitchell. "State of the Art: Automated Black-Box Web Application Vulnerability Testing". In: *IEEE Symposium on Security and Privacy*. 2010, pp. 332–345.

[35] *Behavior changes: Apps targeting Android 13 or higher.* `https://developer.android.com/about/versions/13/behavior-changes-13`. [Online; accessed: 11-November-2024].

[36] Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt. "Third Party Tracking in the Mobile Ecosystem". In: *Proceedings of the 10th ACM Conference on Web Science*. 2018, pp. 23–31.

[37] Android Developers Blog. *Announcing Policy Updates To Bolster Privacy and Security.* `https://android-developers.googleblog.com/2021/07/announcing-policy-updates-to-bolster.html`. [Online; accessed: 11-November-2024].

[38] Android Developers Blog. *Expanding Play's Target Level API Requirements to Strengthen User Security.* `https://android-developers.googleblog.com/2022/04/expanding-plays-target-level-api-requirements-to-strengthen-user-security.html`. [Online; accessed: 11-November-2024].

[39] *Bloomberg Law database.* `https://pro.bloomberglaw.com/`. [Online; accessed: 19-September-2024].

[40] Mehran Bozorgi, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. "Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2010, pp. 105–114.

[41] Larissa Braz and Alberto Bacchelli. "Software Security during Modern Code Review: The Developer's Perspective". In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022, pp. 810–821.

[42] Larissa Braz, Enrico Fregnan, Gül Çalikli, and Alberto Bacchelli. "Why Don't Developers Detect Improper Input Validation?'; DROP TABLE Papers;–". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 499–511.

[43] BugCrowd. `https://www.bugcrowd.com/`. [Online; accessed: 11-October-2024].

[44] Duc Bui, Yuan Yao, Kang G Shin, Jong-Min Choi, and Junbum Shin. "Consistency Analysis of Data-Usage Purposes in Mobile Apps". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 2824–2843.

[45] Google Play Console Center. *Ads*. `https://support.google.com/googleplay/android-developer/answer/9857753`. [Online; accessed: 11-November-2024]. 2023.

[46] Google Play Console Center. *User Data*. `https://support.google.com/googleplay/android-developer/answer/10144311`. [Online; accessed: 11-November-2024]. 2023.

[47] Google Play Policy Center. *Deceptive Behavior*. `https://support.google.com/googleplay/android-developer/answer/9888077`. [Online; accessed: 11-November-2024]. 2024.

[48] Google Play Policy Center. *Families Self-Certified Ads SDK Program*. `https://support.google.com/googleplay/android-developer/answer/9900633`. [Online; accessed: 11-November-2024]. 2024.

[49] Google Play Policy Center. *Google Play Families Policies*. `https://support.google.com/googleplay/android-developer/answer/9893335`. [Online; accessed: 10-July-2023]. 2023.

[50] Google Play Policy Center. *Policy Deadlines*. `https://support.google.com/googleplay/android-developer/table/12921780`. [Online; accessed: 11-November-2024]. 2024.

[51] Google Play Policy Center. *Provide information for Google Play's Data safety section*. `https://support.google.com/googleplay/android-developer/answer/10787469`. [Online; accessed: 11-November-2024]. 2023.

[52] Google Play Policy Center. *SDK Requirements.* https://support.google.com/googleplay/android-developer/answer/13323374. [Online; accessed: 11-November-2024]. 2024.

[53] Google Play Policy Center. *Store Listing and Promotion.* https://support.google.com/googleplay/android-developer/topic/9877064. [Online; accessed: 11-November-2024]. 2024.

[54] Google Play Policy Center. *User Data.* https://support.google.com/googleplay/android-developer/answer/13316080. [Online; accessed: 11-November-2024]. 2023.

[55] Google Play Store Policy Center. *Google Play's Target API Level Policy.* https://support.google.com/googleplay/android-developer/answer/11917020. [Online; accessed: 11-November-2024]. 2024.

[56] InMobi Support Center. *InMobi - Android SDK Documentation.* https://support.inmobi.com/monetize/sdk-documentation/android-guidelines/overview-android-guidelines. [Online; accessed: 11-November-2024]. 2024.

[57] Kathy Charmaz. *Constructing grounded theory: A practical guide through qualitative analysis.* Sage, 2006.

[58] Inc Chartboost. *Chartboost Monetization and Advertising.* https://www.chartboost.com/. [Online; accessed: 11-November-2024]. 2024.

[59] *ChatGPT.* https://chatgpt.com/. [Online; accessed: 24-September-2024].

[60] *Children's Privacy.* https://www.ftc.gov/tips-advice/business-center/privacy-and-security/children%27s-privacy. [Online; accessed: 31-October-2021].

[61] Federal Trade Commission. *Children's Online Privacy Protection Rule (COPPA).* https://www.ftc.gov/legal-library/browse/rules/childrens-online-privacy-protection-rule-coppa. [Online; accessed: 11-November-2024]. 2013.

[62] Federal Trade Commission. *Complying with COPPA: Frequently Asked Questions.* https://www.ftc.gov/business-guidance/resources/complying-coppa-frequently-asked-questions. [Online; accessed: 11-November-2024]. 2020.

[63] Federal Trade Commission. *FTC Seeks Comments on Children's Online Privacy Protection Act Rule.* https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-seeks-comments-childrens-online-privacy-protection-act-rule. [Online; accessed: 20-January-2024]. 2019.

[64] Federal Trade Commission. *Mobile Apps for Kids: Current Privacy Disclosures are Disappointing.* https://www.ftc.gov/sites/default/files/documents/reports/mobile-apps-kids-current-privacy-disclosures-are-disappointing/120216mobile_apps_kids.pdf. [Online; accessed: 21-January-2024]. 2012.

[65] Federal Trade Commission. *Mobile Apps for Kids: Disclosures Still Not Making the Grade.* `https://www.ftc.gov/sites/default/files/documents/reports/mobile-apps-kids-disclosures-still-not-making-grade/121210mobilekidsappreport.pdf`. [Online; accessed: 21-January-2024]. 2012.

[66] intersoft consulting. *General Data Protection Regulation (GDPR).* `https://gdpr-info.eu/`. [Online; accessed: 11-November-2024]. 2016.

[67] Anastasia Danilova, Alena Naiakshina, Anna Rasgauski, and Matthew Smith. "Code Reviewing as Methodology for Online Security Studies with Developers - A Case Study with Freelancers on Password Storage". In: *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021).* 2021, pp. 397–416.

[68] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes. "Keep me Updated: An Empirical Study of Third-party Library Updatability on Android". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.* 2017, pp. 2187–2200.

[69] *Developer of Apps Popular with Children Agrees to Settle FTC Allegations It Illegally Collected Kids' Data without Parental Consent.* `https://www.ftc.gov/news-events/press-releases/2020/06/developer-apps-popular-children-agrees-settle-ftc-allegations-it`. [Online; accessed: 17-November-2021].

[70] Android Developers. *Android 8.0 Behavior Changes.* `https://developer.android.com/about/versions/oreo/android-8.0-changes`. [Online; accessed: 11-November-2024]. 2023.

[71] Android Developers. *Android Asset Packaging Tool (AAPT).* `https://developer.android.com/tools/aapt2`. [Online; accessed: 11-November-2024]. 2023.

[72] Android Developers. *Android Debug Bridge (ADB).* `https://developer.android.com/tools/adb`. [Online; accessed: 11-November-2024]. 2024.

[73] Android Developers. *Android ID.* `https://developer.android.com/reference/android/provider/Settings.Secure#ANDROID_ID`. [Online; accessed: 11-November-2024]. 2024.

[74] Android Developers. *Android Releases.* `https://developer.android.com/about/versions"`. [Online; accessed: 11-November-2024]. 2024.

[75] Android Developers. *Android SDK: Command line tools.* `https://developer.android.com/tools`. [Online; accessed: 11-November-2024]. 2024.

[76] Android Developers. *AppSetId.* `https://developer.android.com/design-for-safety/privacy-sandbox/reference/adservices/appsetid/AppSetId`. [Online; accessed: 11-November-2024]. 2023.

[77] Android Developers. *Best practices for unique identifiers.* `https://developer.android.com/training/articles/user-data-ids`. [Online; accessed: 11-November-2024]. 2024.

[78] Android Developers. *Identify developer-owned apps*. https://developer.android.com/training/articles/app-set-id. [Online; accessed: 11-November-2024]. 2024.

[79] Android Developers. *Meet Google Play's target API level requirement*. https://developer.android.com/google/play/requirements/target-sdk. [Online; accessed: 11-November-2024]. 2024.

[80] Android Developers. *UI/Application Exerciser Monkey*. https://developer.android.com/studio/test/other-testing-tools/monkey. [Online; accessed: 11-November-2024]. 2023.

[81] Constanze Dietrich, Katharina Krombholz, Kevin Borgolte, and Tobias Fiebig. "Investigating System Operators' Perspective on Security Misconfigurations". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1272–1289.

[82] Chartboost Documentation. *SDK Privacy Methods*. https://docs.chartboost.com/en/monetization/integrate/android/sdk-privacy-methods/. [Online; accessed: 11-November-2024]. 2024.

[83] Unity Documentation. *Consumer privacy act compliance*. https://docs.unity.com/ads/en-us/manual/CCPACompliance. [Online; accessed: 11-November-2024]. 2023.

[84] Unity Documentation. *Google Play data safety section for Unity Ads*. https://docs.unity.com/ads/en-us/manual/GoogleDataSafety. [Online; accessed: 11-November-2024]. 2023.

[85] Unity Documentation. *Installing the Unity Ads SDK for Android*. https://docs.unity.com/ads/en-us/manual/InstallingTheAndroidSDK. [Online; accessed: 11-November-2024]. 2024.

[86] Anne Edmundson, Brian Holtkamp, Emanuel Rivera, Matthew Finifter, Adrian Mettler, and David Wagner. "An Empirical Study on the Effectiveness of Security Code Review". In: *International Symposium on Engineering Secure Software and Systems*. Springer. 2013, pp. 197–212.

[87] Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel. "An Empirical Study of Cryptographic Misuse in Android Applications". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 73–84.

[88] Anirudh Ekambaranathan, Jun Zhao, and Max Van Kleek. "“Money makes the world go around”: Identifying Barriers to Better Privacy in Children's Apps From Developers' Perspectives". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–15.

[89] Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben, and Matthew Smith. "Why Eve and Mallory love Android: An Analysis of Android SSL (in) Security". In: *Proceedings of the 2012 ACM conference on Computer and communications security.* 2012, pp. 50–61.

[90] Álvaro Feal, Paolo Calciati, Narseo Vallina-Rodriguez, Carmela Troncoso, and Alessandra Gorla. "Angel or Devil? A Privacy Study of Mobile Parental Control Apps." In: *Proc. Priv. Enhancing Technol.* 2020.2 (2020), pp. 314–335.

[91] Álvaro Feal, Julien Gamba, Juan Tapiador, Primal Wijesekera, Joel Reardon, Serge Egelman, and Narseo Vallina-Rodriguez. "Don't Accept Candy from Strangers: An Analysis of Third-Party Mobile SDKs". In: *Data Protection and Privacy, Volume 13: Data Protection and Artificial Intelligence* 13 (2021), p. 1.

[92] *Federal Trade Commission (FTC).* `https://www.ftc.gov/`. [Online; accessed: 11-November-2024]. 2024.

[93] Google Firebase. `https://firebase.google.com/`. [Online; accessed: 16-November-2024]. 2024.

[94] Flurry. *Flurry Analytics Terms of Service.* `https://developer.yahoo.com/flurry/legal-privacy/terms-service/flurry-analytics-terms-service.html`. [Online; accessed: 06-January-2024]. 2023.

[95] *Flurry - Android SDK.* `https://developer.yahoo.com/flurry/docs/integrateflurry/android/`. [Online; accessed: 05-July-2022].

[96] Kelsey R Fulton, Daniel Votipka, Desiree Abrokwa, Michelle L Mazurek, Michael Hicks, and James Parker. "Understanding the How and the Why: Exploring Secure Development Practices through a Course Competition". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* 2022, pp. 1141–1155.

[97] Jack Gardner, Yuanyuan Feng, Kayla Reiman, Zhi Lin, Akshath Jain, and Norman Sadeh. "Helping Mobile Application Developers Create Accurate Privacy Labels". In: *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW).* IEEE. 2022, pp. 212–230.

[98] Atul Gawande. *The Checklist Manifesto.* Metropolitan Books, 2009.

[99] *GDPR - Consent.* `https://gdpr-info.eu/issues/consent/`. [Online; accessed: 01-November-2021].

[100] *GDPR - Recital 38.* `https://gdpr-info.eu/recitals/no-38/`. [Online; accessed: 01-November-2021].

[101] Lisa Geierhaas, Anna-Marie Ortloff, Matthew Smith, and Alena Naiakshina. "Let's Hash: Helping Developers with Password Security". In: *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022).* 2022, pp. 503–522.

[102] Conor Gilsenan, Fuzail Shakir, Noura Alomar, and Serge Egelman. "Security and Privacy Failures in Popular 2FA Apps". In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023.

[103] Aniketh Girish, Tianrui Hu, Vijay Prakash, Daniel J Dubois, Srdjan Matic, Danny Yuxing Huang, Serge Egelman, Joel Reardon, Juan Tapiador, David Choffnes, et al. "In the Room Where It Happens: Characterizing Local Communication and Threats in Smart Homes". In: *Proceedings of the 2023 ACM on Internet Measurement Conference*. 2023, pp. 437–456.

[104] Google. *Comply with Google Play's Families Policy using AdMob*. `https://support.google.com/admob/answer/6223431?hl=en`. [Online; accessed: 17-December-2024]. 2022.

[105] Google. *Participate in Designed for Families*. `https://support.google.com/googleplay/android-developer/answer/9859555?hl=en`. [Online; accessed: 17-August-2022]. 2022.

[106] *Google - Advertising ID*. `https://support.google.com/googleplay/android-developer/answer/6048248#zippy=%2Ctargeting-devices-without-an-advertising-id%2Chow-to-opt-out-of-personalized-ads`. [Online; accessed: 31-January-2022].

[107] *Google Admob*. `https://admob.google.com/home/`. [Online; accessed: 9-November-2024].

[108] *Google Admob Mediation - AppLovin*. `https://developers.google.com/admob/android/mediation/applovin`. [Online; accessed: 12-November-2024].

[109] *Google Admob: CCPA Preparation*. `https://developers.google.com/admob/android/ccpa`. [Online; accessed: 19-November-2021].

[110] *Google Admob: Child-Directed SDK Settings*. `https://developers.google.com/admob/android/targeting#child-directed_setting`. [Online; accessed: 19-November-2021].

[111] *Google Play Store*. `https://play.google.com/store/apps`. [Online; accessed: 4-November-2024].

[112] Peter Leo Gorski, Yasemin Acar, Luigi Lo Iacono, and Sascha Fahl. "Listen to Developers! A Participatory Design Study on Security Warnings for Cryptographic APIs". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–13.

[113] Peter Leo Gorski, Luigi Lo Iacono, Dominik Wermke, Christian Stransky, Sebastian Möller, Yasemin Acar, and Sascha Fahl. "Developers Deserve Security Warnings, Too: On the Effect of Integrated Security Advice on Cryptographic API Misuse". In: *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 2018, pp. 265–281.

[114] HackerOne. `https://www.hackerone.com/`. [Online; accessed: 11-October-2024].

[115] Catherine Han, Irwin Reyes, Álvaro Feal, Joel Reardon, Primal Wijesekera, Narseo Vallina-Rodriguez, Amit Elazari, Kenneth A Bamberger, and Serge Egelman. “The Price is (Not) Right: Comparing Privacy in Free and Paid Apps”. In: *Proceedings on Privacy Enhancing Technologies*. 2020.

[116] Alex B Haynes, Thomas G Weiser, William R Berry, Stuart R Lipsitz, Abdel-Hadi S Breizat, E Patchen Dellinger, Teodoro Herbosa, Sudhir Joseph, Pascience L Kibatala, Marie Carmela M Lapitan, et al. “A Surgical Safety Checklist to Reduce Morbidity and Mortality in a Global Population”. In: *New England journal of medicine* 360.5 (2009), pp. 491–499.

[117] Google Play Console Help. *Advertising ID*. `https://support.google.com/googleplay/android-developer/answer/6048248`. [Online; accessed: 11-November-2024]. 2023.

[118] Google Play Console Help. *Best practices for prominent disclosure and consent*. `https://support.google.com/googleplay/android-developer/answer/11150561`. [Online; accessed: 11-November-2024]. 2023.

[119] Google Play Console Help. *Google Play: Data practices in Families apps*. `https://support.google.com/googleplay/android-developer/answer/11043825`. [Online; accessed: 11-November-2024]. 2023.

[120] Google Play Console Help. *Manage target audience and app content settings*. `https://support.google.com/googleplay/android-developer/answer/9867159`. [Online; accessed: 11-November-2024]. 2024.

[121] Google Play Console Help. *Prepare your app for review*. `https://support.google.com/googleplay/android-developer/answer/9859455`. [Online; accessed: 11-November-2024]. 2023.

[122] Google Play Console Help. *Target API level requirements for Google Play apps*. `https://support.google.com/googleplay/android-developer/answer/11926878`. [Online; accessed: 11-November-2024]. 2024.

[123] Google Play Policy Help. *Participate in the Families Self-Certified Ads SDK Program*. `https://support.google.com/googleplay/android-developer/answer/12955712`. [Online; accessed: 11-November-2024]. 2024.

[124] Steven J Hutchison. *Cybersecurity: Defending the new battlefield*. Tech. rep. Defense Acquisition University, Fort Belvoir, VA, 2013.

[125] InMobi. *InMobi Services*. [Online; accessed: 11-November-2024]. 2024. URL: `https://www.inmobi.com/`.

[126] *IPWHOIS.IO*. `https://ipwhois.io/`. [Online; accessed: 11-November-2024]. 2024.

[127] ironSource. *Google’s Data Safety Questionnaire*. `https://developers.is.com/ironsource-mobile/general/googles-data-safety-questionnaire-full/`. [Online; accessed: 30-November-2023]. 2023.

[128] ironSource. *IronSource Services.* https://www.is.com/. [Online; accessed: 11-November-2024]. 2024.

[129] ironSource. *Mediation Networks for Android.* https://developers.is.com/ironsource-mobile/android/mediation-networks-android/. [Online; accessed: 3-January-2024]. 2023.

[130] ironSource. *Privacy Policy.* https://developers.ironsrc.com/ironsource-mobile/air/ironsource-mobile-privacy-policy. [Online; accessed: 30-November-2023]. 2023.

[131] ironSource. *Regulation Advanced Settings.* https://developers.is.com/ironsource-mobile/android/regulation-advanced-settings/#step-1. [Online; accessed: 26-September-2023]. 2023.

[132] *ISO/IEC 29147: Information Technology – Security Techniques – Vulnerability Disclosure.* https://www.iso.org/standard/45170.html. [Online; accessed: 27-June-2019]. 2019.

[133] *ISO/IEC 30111: Information Technology – Security Techniques – Vulnerability Handling Processes.* https://www.iso.org/obp/ui/#iso:std:iso-iec:30111:ed-1:v1:en. [Online; accessed: 27-June-2019]. 2019.

[134] Jan Jancar, Marcel Fourné, Daniel De Almeida Braga, Mohamed Sabt, Peter Schwabe, Gilles Barthe, Pierre-Alain Fouque, and Yasemin Acar. "“They're not that hard to mitigate”: What cryptographic library developers think about timing attacks". In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2022, pp. 632–649.

[135] *kidSAFE.* https://www.kidsafeseal.com/aboutourprogram.html. [Online; accessed: 21-February-2022].

[136] Konrad Kollnig, Pierre Dewitte, Max Van Kleek, Ge Wang, Daniel Omeiza, Helena Webb, and Nigel Shadbolt. "A Fait Accompli? An Empirical Study into the Absence of Consent to Third-Party Tracking in Android Apps". In: *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. 2021, pp. 181–196.

[137] Konrad Kollnig, Anastasia Shuba, Reuben Binns, Max Van Kleek, and Nigel Shadbolt. "Are iPhones Really Better for Privacy? A Comparative Study of iOS and Android Apps". In: *Proceedings on Privacy Enhancing Technologies* 2022.2 (2022).

[138] Pei-Yi Kuo, Rajiv Saran, Marissa Argentina, Michael Heung, Jennifer L Bragg-Gresham, Dinesh Chatoth, Brenda Gillespie, Sarah Krein, Rebecca Wingard, Kai Zheng, et al. "Development of a Checklist for the Prevention of Intradialytic Hypotension in Hemodialysis Care: Design Considerations Based on Activity Theory". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–14.

[139] J. Richard Landis and Gary G. Koch. "The Measurement of Observer Agreement for Categorical Data". In: *Biometrics* 33.1 (1977), pp. 159–174. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/2529310.

[140] Frank Li and Vern Paxson. “A Large-Scale Empirical Study of Security Patches”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 2201–2215.

[141] Frank Li, Lisa Rogers, Arunesh Mathur, Nathan Malkin, and Marshini Chetty. “Keepers of the Machines: Examining How System Administrators Manage Software Updates”. In: *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. 2019, pp. 273–288.

[142] Tianshi Li, Yuvraj Agarwal, and Jason I Hong. “Coconut: An IDE Plugin for Developing Privacy-Friendly Apps”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.4 (2018), pp. 1–35.

[143] Tianshi Li, Lorrie Faith Cranor, Yuvraj Agarwal, and Jason I Hong. “Matcha: An IDE Plugin for Creating Accurate Privacy Nutrition Labels”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8.1 (2024), pp. 1–38.

[144] Tianshi Li, Elizabeth Louie, Laura Dabbish, and Jason I Hong. “How Developers Talk About Personal Data and What It Means for User Privacy: A Case Study of a Developer Forum on Reddit”. In: *Proceedings of the ACM on Human-Computer Interaction* 4.CSCW3 (2021), pp. 1–28.

[145] Tianshi Li, Elijah B Neundorfer, Yuvraj Agarwal, and Jason I Hong. “Honeysuckle: Annotation-Guided Code Generation of In-App Privacy Notices”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5.3 (2021), pp. 1–27.

[146] Tianshi Li, Kayla Reiman, Yuvraj Agarwal, Lorrie Faith Cranor, and Jason I Hong. “Understanding challenges for developers to create accurate privacy nutrition labels”. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–24.

[147] *LinkedIn*. `https://www.linkedin.com/`. [Online; accessed: 26-November-2024]. 2024.

[148] V Benjamin Livshits and Monica S Lam. “Finding Security Vulnerabilities in Java Applications with Static Analysis”. In: *USENIX Security Symposium*. Vol. 14. 2005.

[149] Allan Lyons, Julien Gamba, Austin Shawaga, Joel Reardon, Juan Tapiador, Serge Egelman, Narseo Vallina-Rodriguez, et al. “Log: It’s Big, It’s Heavy, It’s Filled with Personal Data! Measuring the Logging of Sensitive Information in the Android Ecosystem”. In: *Usenix Security Symposium*. 2023.

[150] Tinhinane Medjkoune, Oana Goga, and Juliette Senechal. “Marketing to Children Through Online Targeted Advertising: Targeting Mechanisms and Legal Aspects”. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2023, pp. 180–194.

[151] Meta. *Information for Child-Directed Apps and Services*. `https://developers.facebook.com/docs/audience-network/optimization/best-practices/coppa/`. [Online; accessed: 11-November-2024]. 2024.

[152] Meta. *Meta App Events SDK*. `https://developers.facebook.com/docs/app-events/getting-started-app-events-android`. [Online; accessed: 11-November-2024]. 2024.

[153] Meta. *Meta Audience Network SDK*. `https://developers.facebook.com/docs/audience-network/setting-up/ad-setup/android/`. [Online; accessed: 11-November-2024]. 2024.

[154] Abraham H Mhaidli, Yixin Zou, and Florian Schaub. “”We Can’t Live Without Them!” App Developers’ Adoption of Ad Networks and Their Considerations of Consumer Risks”. In: *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. 2019, pp. 225–244.

[155] *Mobfox - Android SDK*. `https://mobfox.atlassian.net/wiki/spaces/PUMD/pages/1205239832/Integrate+via+Android+SDK`. [Online; accessed: 05-July-2022].

[156] *Mobile Advertising Network InMobi Settles FTC Charges It Tracked Hundreds of Millions of Consumers’ Locations Without Permission*. `https://www.ftc.gov/news-events/press-releases/2016/06/mobile-advertising-network-inmobi-settles-ftc-charges-it-tracked`. [Online; accessed: 17-November-2021].

[157] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. “Jumping Through Hoops: Why do Java Developers Struggle with Cryptography APIs?” In: *Proceedings of the 38th International Conference on Software Engineering*. 2016, pp. 935–946.

[158] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel Von Zezschwitz, and Matthew Smith. “” If you want, I can store the encrypted password” A Password-Storage Field Study with Freelance Developers”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–12.

[159] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. “Why Do Developers Get Password Storage Wrong? A Qualitative Usability Study”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 311–328.

[160] Meta Audience Network. *Add the Audience Network SDK to your Android App*. `https://developers.facebook.com/docs/audience-network/setting-up/platform-setup/android/add-sdk`. [Online; accessed: 25-September-2023]. 2023.

[161] *New Rule to Protect Children’s Online Privacy Takes Effect April 21, 2000*. `https://www.ftc.gov/news-events/press-releases/2000/04/new-rule-protect-childrens-online-privacy-takes-effect-april-21`. [Online; accessed: 26-January-2022].

[162] Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. “Up2dep: Android Tool Support to Fix Insecure Code Dependencies”. In: *Proceedings of the 36th Annual Computer Security Applications Conference*. 2020, pp. 263–276.

[163] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. “A Stitch in Time: Supporting Android Developers in Writing Secure Code”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1065–1077.

[164] Trung Tin Nguyen, Michael Backes, Ninja Marnau, and Ben Stock. “Share First, Ask Later (or Never?) Studying Violations of GDPR’s Explicit Consent in Android Apps”. In: *USENIX Security Symposium*. 2021.

[165] Trung Tin Nguyen, Michael Backes, and Ben Stock. “Freely Given Consent? Studying Consent Notice of Third-Party Tracking and Its Violations of GDPR in Android Apps”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2022, pp. 2369–2383.

[166] Information Commissioner’s Office. *GDPR - Data minimisation*. 2016. URL: `https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/childrens-information/childrens-code-guidance-and-resources/age-appropriate-design-a-code-of-practice-for-online-services/8-data-minimisation/`.

[167] Ehimare Okoyomon, Nikita Samarin, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, Irwin Reyes, Álvaro Feal, Serge Egelman, et al. “On The Ridiculousness of Notice and Consent: Contradictions in App Privacy Policies”. In: *Workshop on Technology and Consumer Protection (ConPro 2019), in conjunction with the 39th IEEE Symposium on Security and Privacy*. 2019.

[168] Daniela Oliveira, Marissa Rosenthal, Nicole Morin, Kuo-Chuan Yeh, Justin Cappos, and Yanyan Zhuang. “It’s the Psychology Stupid: How Heuristics Explain Software Vulnerabilities and How Priming Can Illuminate Developer’s Blind Spots”. In: *Proceedings of the 30th Annual Computer Security Applications Conference*. 2014, pp. 296–305.

[169] Daniela Seabra Oliveira, Tian Lin, Muhammad Sajidur Rahman, Rad Akefirad, Donovan Ellis, Eliany Perez, Rahul Bobhate, Lois A DeLong, Justin Cappos, and Yuriy Brun. “API Blindspots: Why Experienced Developers Write Vulnerable Code”. In: *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 2018, pp. 315–328.

[170] Marten Oltrogge, Yasemin Acar, Sergej Dechand, Matthew Smith, and Sascha Fahl. “To Pin or Not to Pin—Helping App Developers Bullet Proof Their TLS Connections”. In: *24th USENIX Security Symposium (USENIX Security 15)*. 2015, pp. 239–254.

[171] OneSignal. *OneSignal Services*. `https://onesignal.com/`. [Online; accessed: 11-November-2024]. 2024.

[172] Hernan Palombo, Armin Ziaie Tabari, Daniel Lende, Jay Ligatti, and Xinming Ou. "An Ethnographic Understanding of Software In Security and a Co-Creation Model to Improve Secure Software Development". In: *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020).* 2020, pp. 205–220.

[173] *PBS Kids.* `https://play.google.com/store/apps/dev?id=8332149447945516079&hl=en_US`. [Online; accessed: 24-September-2024].

[174] Charles P Pfleeger, Shari Lawrence Pfleeger, and Mary Frances Theofanos. "A methodology for penetration testing". In: *Computers & Security* 8.7 (1989), pp. 613–620.

[175] Google Play. *Build Teacher Approved apps.* `https://play.google.com/console/about/programs/teacherapproved/`. [Online; accessed: 10-July-2023]. 2023.

[176] Google Play. *Developer Console.* `https://play.google.com/console/about/`. [Online; accessed: 11-November-2024]. 2024.

[177] Google Play. *Developer Policy Center.* `https://play.google.com/about/developer-content-policy/`. [Online; accessed: 11-November-2024]. 2024.

[178] Google Play. *Policy announcement: April 5, 2023.* `https://support.google.com/googleplay/android-developer/answer/13411745?sjid=14219206286122674930-NC`. [Online; accessed: 14-November-2024]. 2024.

[179] Google Play. *Policy announcement: November 16, 2022.* `https://support.google.com/googleplay/android-developer/answer/14550832?sjid=14219206286122674930-NC`. [Online; accessed: 14-November-2024]. 2024.

[180] Google Play. *Policy Archive.* `https://support.google.com/googleplay/android-developer/answer/13386702?sjid=14219206286122674930-NC`. [Online; accessed: 14-November-2024]. 2024.

[181] Andreas Poller, Laura Kocksch, Sven Türpe, Felix Anand Epp, and Katharina Kinder-Kurlanda. "Can Security Become a Routine? A Study of Organizational Change in an Agile Software Development Group". In: *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing.* 2017, pp. 2489–2503.

[182] *PrivacyCon 2022.* `https://www.ftc.gov/news-events/events/2022/11/privacycon-2022`. [Online; accessed: 23-November-2024].

[183] *PRIVO.* `https://www.privo.com/welcome`. [Online; accessed: 21-February-2022].

[184] Ole André V. Ravnås. *Frida.* `https://frida.re/docs/android/`. [Online; accessed: 11-November-2024]. 2023.

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

[186] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. "50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System". In: *28th USENIX Security Symposium (USENIX Security 19)*, pp. 603–620.

[187] *Reddit*. `https://www.reddit.com/`. [Online; accessed: 26-November-2024]. 2024.

[188] Eric Rescorla. "Is Finding Security Holes a Good Idea?" In: *IEEE Security & Privacy* 3.1 (2005), pp. 14–19.

[189] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, Serge Egelman, et al. ""Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale". In: *The 18th Privacy Enhancing Technologies Symposium (PETS 2018)*.

[190] Nikita Samarin, Shayna Kothari, Zaina Siyed, Oscar Bjorkman, Reena Yuan, Primal Wijesekera, Noura Alomar, Jordan Fischer, Chris Hoofnagle, and Serge Egelman. "Lessons in VCR Repair: Compliance of Android App Developers with the California Consumer Privacy Act (CCPA)". In: *Proceedings on Privacy Enhancing Technologies* (2023).

[191] Google Play services. *AdvertisingIdClient.Info*. `https://developers.google.com/android/reference/com/google/android/gms/ads/identifier/AdvertisingIdClient.Info`. [Online; accessed: 11-November-2024]. 2024.

[192] Kochava Services. *Kochava*. `https://www.kochava.com/`. [Online; accessed: 11-November-2024]. 2024.

[193] Katie Shilton and Daniel Greene. "Linking Platforms, Practices, and Developer Ethics: Levers for Privacy Discourse in Mobile Application Development". In: *Journal of Business Ethics* 155.1 (2019), pp. 131–146.

[194] Singular. *Singular Services*. `https://www.singular.net/`. [Online; accessed: 11-November-2024]. 2024.

[195] *Slack*. `https://slack.com/`. [Online; accessed: 26-November-2024]. 2024.

[196] Justin Smith, Lisa Nguyen Do, and Emerson Murphy-Hill. "Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security". In: *Proceedings of the Sixteenth Symposium on Usable Privacy and Security*. 2020.

[197] StackOverflow. *Issue found: Invalid Data safety section. How to fix this issue? [closed]*. `https://stackoverflow.com/questions/71217909/issue-found-invalid-data-safety-section-how-to-fix-this-issue`. [Online; accessed: 16-November-2024]. 2022.

[198] StackOverflow. *Your app's Data safety section will be invalidated and state No information available*. `https://stackoverflow.com/questions/73849730/your-apps-data-safety-section-will-be-invalidated-and-state-no-information-ava`. [Online; accessed: 16-November-2024]. 2022.

[199] Start.io. *Android SDK*. https://support.start.io/hc/en-us/articles/360014774799-Integration-via-Maven#addingsdktoyourproject. [Online; accessed: 11-November-2024]. 2024.

[200] Start.io. *Google Play Safety Section - Data Disclosure*. https://support.start.io/hc/en-us/articles/4413793394962-Google-Play-Safety-Section-Data-Disclosure. [Online; accessed: 11-November-2024]. 2022.

[201] statcounter. *Android Version Market Share Worldwide*. https://gs.statcounter.com/os-version-market-share/android. [Online; accessed: 21-January-2024]. 2024.

[202] Mohammad Tahaei, Alisa Frik, and Kami Vaniea. "Deciding on Personalized Ads: Nudging Developers About User Privacy". In: *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. 2021, pp. 573–596.

[203] Mohammad Tahaei, Kopo M Ramokapane, Tianshi Li, Jason I Hong, and Awais Rashid. "Charting App Developers' Journey Through Privacy Regulation Features in Ad Networks". In: *Proceedings on Privacy Enhancing Technologies* 1 (2022), p. 24.

[204] Mohammad Tahaei, Kami Vaniea, and Naomi Saphra. "Understanding Privacy-Related Questions on Stack Overflow". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–14.

[205] *Tapjoy - Android SDK*. https://dev.tapjoy.com/en/android-sdk/Quickstart. [Online; accessed: 05-July-2022].

[206] Unity Technologies. *Unity Ads: Mobile Game Ad Network Platform Analytics*. https://unity.com/products/unity-ads. [Online; accessed: 11-November-2024]. 2024.

[207] *The General Data Protection Regulation (GDPR)*. https://gdpr-info.eu/. [Online; accessed: 01-November-2021].

[208] Herbert H Thompson. "Application penetration testing". In: *IEEE Security & Privacy* 3.1 (2005), pp. 66–69.

[209] Christian Tiefenau, Maximilian Häring, Katharina Krombholz, and Emanuel von Zezschwitz. "Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators". In: *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. 2020, pp. 239–258.

[210] *TITLE 1.81.5. California Consumer Privacy Act of 2018*. https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5. [Online; accessed: 02-November-2021].

[211] *Twitter*. https://x.com/. [Online; accessed: 26-November-2024]. 2024.

[212] Umeng. *Umeng Services*. https://www.umeng.com/. [Online; accessed: 11-November-2024]. 2024.

[213] *Unity Ads - COPPA Compliance*. https://docs.unity3d.com/Manual/UnityAnalyticsCOPPA.html. [Online; accessed: 19-November-2021].

[214] *User opt-in/opt-out in the AppsFlyer SDK*. `https://support.appsflyer.com/hc/en-us/articles/360001422989-User-opt-in-opt-out-in-the-AppsFlyer-SDK`. [Online; accessed: 27-January-2022].

[215] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. "(Un) Informed Consent: Studying GDPR Consent Notices in the Field". In: *Proceedings of the 2019 AC SIGSAC Conference on Computer and Communications Security*. 2019, pp. 973–990.

[216] *Vulnerability Metrics*. `https://nvd.nist.gov/vuln-metrics/cvss`. [Online; accessed: 23-October-2024]. 2024.

[217] *Vulnerability Scanning Tools*. `https://owasp.org/www-community/Vulnerability_Scanning_Tools`. [Online; accessed: 8-December-2024].

[218] Vungle. *Advanced Settings*. `https://support.vungle.com/hc/en-us/articles/360047780372-Advanced-Settings`. [Online; accessed: 3-January-2024]. 2023.

[219] Vungle. *Vungle Services*. `https://vungle.com`. [Online; accessed: 11-November-2024]. 2024.

[220] Charles Weir, Ben Hermann, and Sascha Fahl. "From Needs to Actions to Secure Apps? The Effect of Requirements and Developer Practices on App Security". In: *29th USENIX Security Symposium (USENIX Security 20)*. 2020, pp. 289–305.

[221] *WestLaw database*. `https://www.westlaw.com`. [Online; accessed: 19-September-2024].

[222] Zack Whittaker. *Apple restricts ads and third-party trackers in iPhone apps for kids*. `https://techcrunch.com/2019/06/03/apple-kid-apps-trackers/"`. [Online; accessed: 11-November-2024]. 2019.

[223] Anhao Xiang, Weiping Pei, and Chuan Yue. "PolicyChecker: Analyzing the GDPR Completeness of Mobile Apps' Privacy Policies". In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2023, pp. 3373–3387.

[224] Jing Xie, Heather Lipford, and Bei-Tseng Chu. "Evaluating Interactive Support for Secure Programming". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012, pp. 2707–2716.

[225] Yandex. *Yandex Mobile Ads SDK for Android*. [Online; accessed: 11-November-2024]. 2023. URL: `https://yandex.ru/support2/mobile-ads/en/dev/android`.

[226] Yuqing Yang, Mohamed Elsabagh, Chaoshun Zuo, Ryan Johnson, Angelos Stavrou, and Zhiqiang Lin. "Detecting and Measuring Misconfigured Manifests in Android Apps". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2022, pp. 3063–3077.

[227] Le Yu, Tao Zhang, Xiapu Luo, and Lei Xue. "Autoppg: Towards Automatic Generation of Privacy Policy for Android Applications". In: *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*. 2015, pp. 39–50.

[228] Sebastian Zimmeck, Rafael Goldstein, and David Baraka. "PrivacyFlash Pro: Automating Privacy Policy Generation for Mobile Apps." In: *NDSS*. Vol. 2. 2021, p. 4.

[229] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman Sadeh, Steven Bellovin, and Joel Reidenberg. "Automated Analysis of Privacy Requirements for Mobile Apps". In: *2016 AAAI Fall Symposium Series*. 2016.

Appendix A

Supplemental Materials for Chapter 3

A.1 Interview Guide

The following subsections include the questions we used in our pre-interview questionnaire and the ones we asked the participants in our semi-structured interviews. Some of the questions were asked in all the interviews, whereas the remaining questions were chosen based on the current role/job title of the participant.

Pre-Interview Questionnaire

1. How long have you been working as a hacker or a tester?
2. How did you learn hacking/vulnerability discovery skills?
3. What is your job title (employment status)?
4. What other roles have you had in the past?
5. Briefly describe your career path (e.g., bug bounty hunter, pen tester and then red teamer).
6. Do you participate in bug bounty programs? (If yes, how many times have you been awarded a bug bounty? and how many years have you been working as a bug hunter?)
7. Have you received formal security education in computer security or software engineering? If yes, please list the degrees and security certifications you have.
8. How many people work at your company?
9. How many security professionals does your company have?
10. What is your company's main line of business?

Interview questions for all groups

1. How would you structure your testing process if you were explaining it to a university student?

2. Why do you think your role is important in testing/vulnerability discovery? (follow-up question for managerial positions: How would you decide to use a given testing team and at what point? What sort of factors go into that decision?)
3. Based on your experience, what is the contribution of each team?, how would each of them fit into the testing pipeline?, how do you decide what software gets tested, how it gets tested and when it gets tested?, and how does the timeframe affect your approach to testing?
4. Are there any common mistakes or incorrect practices that you often notice people do when it comes to testing the target software?
5. Can you list a set of metrics that you find useful for quantifying your success/failure with regards to what you found in the software you are testing?
6. How do you define security-related test cases and based on what?
7. What happens after you submit the reports to other teams? (follow-up question: In order to make the process more efficient, what are the changes that you want/expect to see after you report a vulnerability? (i.e., what are some ways the flow of the vulnerability reporting would be improved throughout an organization?))
8. How does the testing process work for projects that follow agile methodologies such as Scrum?
9. Briefly describe your recon methodology. (follow-up questions: what is the type of information you find useful for the type of testing you do? and what are the main sources of internal and external information you normally rely on to complete your testing work including training and knowledge flows?)
10. What sorts of tools are you using?, and what tools do you find useful for testing?
11. What are the different testing phases that you use tools for?
12. How do you decide what tools to get to your teams? What factors can help you decide on that? Please describe your tool selection process and are there any restrictions in terms of what tools could your engineers use? (this question is for managerial positions such as CISOs)
13. Describe the role of ‘automation’ in terms of the types of tasks you do.
14. How would you define the scope for software/web application testing?
15. Do you think that scoping is limiting you in terms of what types of testing you can do? If so, why?
16. Are there any other organizational barriers/workplace policies or/and contractual limitations that could restrict your team from carrying out certain types of testing. If yes, please explain.
17. What are the vulnerability types you normally look for?
18. What are the vulnerability types that are commonly and consistently found in different projects?
19. Do you interact with other (red, blue, purple, pen testers, etc) teams on your day to day job?

20. How often do you communicate with other teams as part of your job? Please explain.
21. Have you ever had the chance to discuss the security vulnerabilities you found with software developers? If yes, please comment on the impact of such discussion on improving the security of the software you were testing.
22. Explain how did you acquire your current skill set?
23. How do you stay up-to-date with the latest discovered vulnerabilities and attacks?, and how do you reflect this knowledge in your testing work?

Questions for red teams

1. What are the factors that drive companies to decide to create or contract with a red team?
2. How are red teams structured?
3. What is the frequency in which you do red teaming assessment in a year?, and what factors affect the number of assessments done in a year?
4. Describe the workflow of red teaming.
5. What is the most common ways you establish an initial foothold on an organization?
6. What are the tactics you employ to remain undetected by blue teams?
7. What are the rules of engagement that you usually make sure are followed?
8. What are the common mistakes that you observed blue teamers or other testing teams do? and what are your recommendations on how to make their job better?
9. Do you talk/teach/explain blue teams about your findings? If so, how regular are these meetings? and what sorts of information do red teams/blue teams share with each other?
10. The basic idea behind red teaming is to challenge assumptions and identify blind spots, do you have any special standard techniques/checklists? And how do you normally operate? Where do you start?

Questions for blue teams

1. What are the factors that drive companies to decide to create a blue team?
2. How are blue teams structured?
3. Describe the workflow of blue teaming.
4. What information can help you differentiate between attacks mounted by red teams or real attackers?
5. How is your job different from red teamers' job? Can you describe red teamers' methodology to testing?
6. What are the common mistakes that you observed red teamers or other testing teams do? and what are your recommendations on how to make their job better?
7. Do you communicate with red teams? If so, how regular these meetings are? and, what sorts of information do red teams/blue teams share with each other?

8. If you get an alert, how do you respond? Do you have a process for that? And what bottlenecks do you normally experience throughout the process?

Questions for purple teams

1. What are the factors that drive companies to decide to create a purple team?
2. What is unique about your purple team from the blue team and the red team? and what is your team's role in the organization?
3. How are purple teams structured?
4. Describe the workflow of purple teaming.
5. How to improve communication and workflow between red and blue teams?
6. What are the skillsets you share with blue teamers? and what are the skillsets you share with red teamers?
7. Are there differences in the sets of tools used by blue teams and red teams?
8. Do you see areas where improvements can be made in terms of elevating blue and red teamers' chances of finding security vulnerabilities more efficiently and improving the effectiveness of their testing work?

Questions for penetration testers

1. To what degree do you rely on automated vulnerability detection tools as opposed to manually learn about the systems of interest?
2. In your opinion, how to improve testers' vulnerability discovery capabilities?, how differently companies could use pen testers efficiently than the current practices? and what are the obstacles that could hinder your work?
3. Do you follow a systematic approach to finding vulnerabilities? Please explain.
4. Do you target your recon based on knowledge of an organization's business interests? If so, how?
5. How do you draw your penetration testing plans? is it based on the contract? and does the penetration testing team have the freedom to come up with its own testing plan?
6. How and when do you start to collect information about a target?
7. In your opinion, what are the reasons that drive companies to choose to outsource security testing to an external pen-testing company rather than doing the tests internally by their internal testing teams?
8. What are the types of sensitive details that companies normally disclose to third-party pen-testing services?
9. Can you comment on the nature of non-disclosure agreements that contractors must sign before starting the pen-testing work? and can you comment on whether these agreements are affecting the efficiency or effectiveness of penetration testing?

10. What mechanisms are used to assure accountability of actions taken by pen-testing teams? e.g., do companies keep logs that allow them to inspect what pen testers have done and investigate whether there is some degree of damage that resulted from pen testers actions?
11. Do companies archive information collected during pen testing? And how do they make sure that this information is not accessed by unauthorized people?

Questions for managerial positions such as CISOs

1. What are the factors that drive companies to decide to create an internal red/blue/purple team or involve an external red team in their pen testing processes?
2. In your opinion, when should an organization consider creating a bug bounty program?
3. Describe the dynamics of interaction that occur between different teams.
4. Do you have an internal red team, pen testing or application testing team? What is the main function of each team?, and how do they integrate into the rest of the security ecosystem?
5. In your opinion, how can we maximize the effectiveness of all the testing teams in an organization and are there specific areas where you think that significant improvements can be made?
6. In your opinion, what are the challenges to unifying the efforts of all different teams?
7. Can you explain some lessons learned from previous pentesting works?
8. In your opinion, where does the role of bug bounty programs fit in the whole security testing process?
9. What is the organization of your blue/red teams? (i.e., what other types of teams/subteams do you have in your organization?)
10. What is the workflow that your blue team would follow once an incident of a malicious attack has occurred?
11. Do you recall any bad experiences that relate to red/blue/purple teaming?
12. If red/blue/purple teaming is done correctly, how do you make sure that the vulnerabilities reported are actually fixed?, and what procedures do you follow to ensure that your security posture has actually improved as a result of the testing you've done?
13. Are people normally receptive/welcoming to the type of feedback they receive from red teamers? how about other teams? If no, how do you address this problem?
14. If you were to involve a red team, would you do it in-house or outsource the red teaming process to another company? Please explain the reasons behind your answer.
15. How do you perceive each team's contribution as a whole? If you were to structure these different teams in tiers, how would you do it?
16. In your opinion, how can an organization make the feedback loop between the different testing teams faster?
17. How do bug bounty programs integrate with whatever other testing teams an organization has in place (e.g., red/blue/purple and pentesting teams)?
18. Are there redundancies between what bug hunters do and what other testing teams do?
19. Do you think that some testing teams are biased towards discovering certain vulnerability types? If yes, how so?
20. Have you ever noticed that there is misconnect between different teams? If yes, please explain why such misconnect exists.
21. Do have a bug bounty program?, and what motivated you to create a bug bounty

program in the first place?

Questions for bug bounty hunters

1. How do you view your role in software/application vulnerability discovery in general?
2. In your opinion, when should an organization consider creating a bug bounty program?
3. How do bug bounty programs integrate with whatever other testing teams an organization has in place (e.g., red/blue/purple and pen testing teams)?
4. Describe some ways in which VRPs/bug bounty programs can be improved.
5. What would push you away from a bug bounty program?

Questions for internal security engineers

1. What information do you know that is unique to your team that other teams find valuable?
2. What information do you feel like would be valuable to how well you can do your job that other teams have?
3. How does your team deal with security vulnerabilities that are found in production after the software has went through your testing pipeline?
4. How does your team deal with bugs found by other teams?
5. Where do you feel your team is weak in terms of finding security vulnerabilities?
6. What do you feel like is your team's strength in terms of improving the security of the organization as a whole?
7. How are internal testing teams structured?
8. Are there any tools that would help you identify security vulnerabilities during the testing pipeline that you have heard are great or wish to have?

A.2 Participants' Demographics

Tables A.1 - A.3 include detailed demographic information about our participants and whether they held engineering roles, managerial roles, or both. We mark a participant as a decider if he/she held a managerial position and a do-er if he/she held a security engineering position.

Table A.1: Participants' demographics.

P	**Job title**	**Other roles held in the past**	**Years of experience doing security testing**	**Organization size**	**Decider**	**Do-er**
P1	CISO	Programmer, manager, CISO	Less than 1 year	Large	✓	
P2	Bug bounty hunter and part-time security engineer	External pen tester, internal red teamer, bug bounty hunter	1 to 5 years	Small		✓
P3	Security engineer	Bug bounty hunter, pen tester	1 to 5 years	Medium		✓
P4	Lead infrastructure security engineer	Internal pen tester, internal red teamer, blue teamer, bug bounty hunter	1 to 5 years	Medium		✓
P5	Bug bounty hunter	Bug bounty hunter, pen tester	1 to 5 years	Small		✓
P6	Team lead penetration tester	Internal pen tester, external pen tester, internal red teamer, external red teamer, bug bounty hunter	1 to 5 years	Small		✓
P7	Full time developer and runs a pen testing firm	Internal pen tester, external pen tester, bug bounty hunter, developer, security officer	1 to 5 years	Medium	✓	✓
P8	Full time red teamer	Internal pen tester, external pen tester, internal red teamer, external red teamer, bug bounty hunter, vulnerability researcher	6 to 10 years	Small		✓
P9	Security engineer	Pen tester,blue teamer, quality assurance engineer, purple teamer	6 to 10 years	Large		✓
P10	Security manager	Blue teamer, purple teamer, bug bounty hunter, security manager	1 to 5 years	Large	✓	✓
P11	Security engineer	Quality assurance engineer, bug bounty hunter	1 to 5 years	Small		✓
P12	CISO	CISO	More than 10 years	Small	✓	
P13	Security engineer	Internal pen tester, external pen tester, internal red teamer, external red teamer, blue teamer, bug bounty hunter	More than 10 years	Large		✓
P14	Bug bounty hunter	Bug bounty hunter	1 to 5 years	NA		✓
P15	Pen tester	External pen tester, bug bounty hunter	More than 10 years	Small		✓
P16	Pen tester	Internal pen tester, bug bounty hunter	1 to 5 years	Small		✓
P17	Security engineer	Internal pen tester, external pen tester, internal/external red teamer, blue teamer, CISO	More than 10 years	Large	✓	✓
P18	Bug bounty hunter	External pen tester, internal red teamer, bug bounty hunter	1 to 5 years	NA		✓

Table A.2: Participants' demographics (cont.).

P	**Job title**	**Other roles held in the past**	**Years of experience doing security testing**	**Organization size**	**Decider**	**Do-er**
P19	Security engineer	Internal pen tester, bug bounty hunter	6 to 10 years	Large		✓
P20	Pen tester	Quality assurance engineer, external pen tester, external red teamer	1 to 5 years	Small		✓
P21	Bug bounty hunter	Bug bounty hunter, external pen tester	1 to 5 years	NA		✓
P22	Security engineer	Security engineer, blue teamer	1 to 5 years	Medium		✓
P23	Security engineer	Internal pen tester, external pen tester, internal red teamer, bug bounty hunter	1 to 5 years	Large		✓
P24	Application security analyst	Security engineer, red teamer, pen tester, bug bounty hunter	6 to 10 years	Small		✓
P25	Security engineer	Security engineer, bug bounty hunter	Less than 1 year	Large		✓
P26	Red teamer	Internal pen tester, red teamer	6 to 10 years	Large		✓
P27	Bug bounty hunter	Internal pen tester, bug bounty hunter	1 to 5 years	Small		✓
P28	Bug bounty hunter	Bug bounty hunter	1 to 5 years	NA		✓
P29	CTO	CTO, external pen tester	1 to 5 years	Small	✓	✓
P30	Security manager	Internal pen tester, external pen tester, internal red teamer, external red teamer, blue teamer, quality assurance engineer, security manager	More than 10 years	Large	✓	✓
P31	Security engineer	Internal pen tester, internal red teamer, blue teamer, purple teamer	1 to 5 years	Large		✓
P32	Security consultant	Pen tester, red teamer, blue teamer, purple teamer	1 to 5 years	Large		✓
P33	Security engineer	Internal pen tester	More than 10 years	Small		✓
P34	Security manager	Internal pen tester, external pen tester, external red teamer, purple teamer, CISO	More than 10 years	Small	✓	✓
P35	Security engineer	Bug bounty hunter, security engineer	1 to 5 years	Large		✓

Table A.3: Participants' demographics (cont.).

P	Job title	Other roles held in the past	Years of experience doing security testing	Organization size	Decider	Do-er
P36	Pen tester	Internal pen tester, external pen tester, external red teamer,bug bounty hunter	Less than 1 year	Small		✓
P37	CTO	External pen tester, external red teamer, purple teamer, bug bounty hunter	More than 10 years	Small	✓	✓
P38	Director of pen testing firm	Internal pen tester, external pen tester, bug bounty hunter	More than 10 years	Small	✓	✓
P39	Blue teamer	Blue teamer	More than 10 years	Large		✓
P40	Bug bounty hunter	External pen tester, internal red teamer, external red teamer, bug bounty hunter	More than 10 years	NA		✓
P41	Application security lead	Internal pen tester, external pen tester, internal red teamer, external red teamer, purple teamer, bug bounty hunter	More than 10 years	Large		✓
P42	Technical Cybersecurity Architect	Red teamer, blue teamer, purple teamer	More than 10 years	Large		✓
P43	Security manager	Pen tester, blue teamer	More than 10 years	Large	✓	✓
P44	Security manager	Internal pen tester, blue teamer, quality assurance engineer	More than 10 years	Small	✓	✓
P45	CTO	Internal pen tester, external pen tester, internal red teamer, external red teamer, purple teamer	6 to 10 years	Small	✓	✓
P46	Blue teamer	Internal pen. tester, external pen tester, internal red teamer, blue teamer, bug bounty hunter	More than 10 years	Medium		✓
P47	Pen tester	Internal pen tester, external pen tester, internal red teamer, external red teamer	6 to 10 years	Small		✓
P48	CISO	Internal pen tester, external pen tester, blue teamer, purple teamer, CISO	More than 10 years	Medium	✓	✓
P49	CISO	CISO	Less than 1 year	Large	✓	
P50	CISO	Internal pen tester, External pen tester, Internal red teamer, external red teamer, blue teamer, purple teamer	More than 10 years	Small	✓	✓
P51	Secrity manager	Security engineer, security manager	More than 10 years	Large	✓	✓
P52	Security manager	Security manager, blue teamer	6 to 10 years	Large	✓	✓
P53	Security manager	Security manager	More than 10 years	Small	✓	

Appendix B

Supplemental Materials for Chapter 4

B.1 Interview Guide

1. How do you normally learn that there is a security vulnerability that may affect you?
2. How do you decide whether to remediate or not remediate a security vulnerability?
3. How do you triage discovered security vulnerabilities?
4. What criteria do you use for prioritizing security vulnerabilities and determining their severity levels?
5. What percentage of security vulnerabilities do you actually end up remediating?
6. Can you describe the process that is normally followed to remediate security vulnerabilities at your organization?
7. What type of tools do you use in your vulnerability remediation processes?
8. How do you remediate security vulnerabilities that exist in third-party code?
9. How does your organization determine the due dates for remediating security vulnerabilities?
10. Which of your teams is responsible for remediating security vulnerabilities?
11. How do you decide who does what after a security vulnerability is discovered?
12. Who would normally need to collaborate with whom in order to remediate a security vulnerability at your organization?
13. What type of information do you think would help you make better decisions in terms of how to remediate security vulnerabilities effectively?
14. What metrics do you use to evaluate your progress in remediating security vulnerabilities?
15. What role does your managers play in your vulnerability remediation processes?
16. Are there organizational constraints that hinder your vulnerability remediation efforts?
17. How do you validate whether your vulnerability fixes actually addressed discovered vulnerabilities? And is that type of checking normally considered in your vulnerability remediation processes?

18. Are there non-technical factors that are hindering your vulnerability remediation efforts? Please explain.
19. Do you think that improving asset management at your organization would make your vulnerability remediation processes more effective? Please explain.
20. What would help you reduce the time between when a security vulnerability is discovered to when it is remediated?

B.2 Survey Questions

General Questions

1. How would you describe your level of involvement in remediating security vulnerabilities at your organization?
2. How would you describe your role in vulnerability remediation?
3. How many years of experience do you have in vulnerability remediation?
4. Are you involved in implementing vulnerability fixes (i.e., writing code for or testing security patches)?
5. How would you describe the size of the organization you currently work for?
6. How would you describe the main line of business of your organization?
7. If you are involved in implementing fixes for security vulnerabilities, what are your main sources for technical vulnerability remediation advice?
8. Do you believe that your organization has the financial resources it needs to remediate security vulnerabilities effectively (e.g., to hire more employees or to buy tools)?
9. Do you believe that your organization is providing you with enough resources that enable you to remediate security vulnerabilities effectively?
10. Do you believe that your organization has employees who have the technical cybersecurity expertise needed to fix security vulnerabilities effectively?
11. Do you receive vulnerability reports through a bug bounty program?

Triaging Vulnerability Reports

1. How does your organization triage the vulnerabilities received from your various vulnerability discovery channels?
2. How would you describe the overall technical expertise/background of the person(s) who triages discovered vulnerabilities at your organization?
3. How would you describe the overall cybersecurity expertise/background of the person(s) who triages discovered vulnerabilities at your organization?
4. Which of the following challenges do you often face when trying to reproduce vulnerabilities reported to you?

5. Generally speaking, how much effort does it take you to reproduce a vulnerability reported by a bug bounty hunter?
6. Generally speaking, how much effort does it take you to reproduce a vulnerability reported by internal engineers or developers?
7. Generally speaking, how much effort does it take you to reproduce a vulnerability reported by paid penetration testing services?
8. At your organization, how much effort does it take you to identify the person responsible for fixing a security vulnerability once it is discovered?
9. Does your organization keep track internally of who developed (or is responsible for) which software that can help you identify whom to contact once a vulnerability needs to be fixed?

Remediating Discovered Vulnerabilities

1. How likely would you be to recommend your organization's vulnerability remediation process to other organizations?
2. Which types of vulnerabilities are you generally least worried about fixing?
3. At your organization, how do you decide whether to remediate or not remediate a discovered security vulnerability?
4. Out of all the critical and high severity vulnerabilities discovered in your systems which can be remediated and using a scale from 0 to 100, what percentage of them do you normally end up remediating? [note: assume that the vulnerabilities have already been triaged and assigned a "high" or "critical" severity rating, and they are not false positives]
5. Out of all the medium severity vulnerabilities discovered in your systems which can be remediated and using a scale from 0 to 100, what percentage of them do you normally end up remediating? [note: assume that the vulnerabilities have already been triaged and assigned a "medium" severity rating, and they are not false positives]
6. Out of all the low severity vulnerabilities discovered in your systems which can be remediated and using a scale from 0 to 100, what percentage of them do you normally end up remediating? [note: assume that the vulnerabilities have already been triaged and assigned a "low" severity rating, and they are not false positives]
7. Once the person responsible for fixing a security vulnerability is identified, how much effort does it take you to communicate the vulnerability to them and articulate the importance of getting it fixed?
8. At your organization, how much effort does it take you to identify all the pieces of code (or software components) affected by a vulnerability once it is discovered?
9. Once you have identified all the areas/components affected by a vulnerability, how much effort does it take you to apply the fix across all the systems affected by it?
10. At your organization, how do you track all the assets that your organization has (i.e., asset management)?

11. At your organization, do you have a formal process to validate whether patches are solving the vulnerabilities they are supposed to fix?
12. If yes, how do you validate whether your vulnerability fixes are working as intended?

Vulnerability Remediation and Third-Party Code

1. Does your organization have a formal process for remediating vulnerabilities that exist in third-party code?
2. How easy is it to remediate vulnerabilities that exist in code developed by someone at your organization relative to vulnerabilities that exist in third-party code?
3. How does your organization handle vulnerabilities that exist in third-party code (e.g., libraries, tools or external dependencies)?
4. Do you have a list showing the points of contact for all the third parties you are working with and who you can contact once you learn that their code has a vulnerability?
5. Have you been in situations where you received ineffective patches from your third party vendors? [note: by ineffective, we mean patches are not fully fixing the corresponding vulnerability]
6. If yes, please explain what the issue(s) was with the patch that you received and how you resolved it.
7. If yes, how did your organization deal with this problem?
8. Does your organization develop products for third-party organizations (e.g., as a contractor for government agencies or other organizations/companies)?
9. If yes, who takes care of remediating vulnerabilities that get discovered in the products you developed for your customers?
10. If yes and if your organization takes care of vulnerability remediation, does your decision of whether to remediate or not depend on which client/customer you are remediating vulnerabilities for?

Vulnerability Management

1. At your organization, who determines how much time your teams have to fix a security vulnerability?
2. What kind of role(s) would you like your managers (e.g., CISOs) to have in your organization's vulnerability remediation decisions?
3. At your organization, who has the authority to approve the decision of not remediating a security vulnerability?

Challenges and Recommendations

1. If you were to make changes to your organization's vulnerability remediation process, how would you improve it? (e.g., in terms of the effectiveness of the vulnerability fixes

and the time it takes you to fix vulnerabilities)

2. What kind of challenges does your organization face when trying to remediate vulnerabilities effectively and efficiently?
3. What kind of help does your organization need in order to improve your vulnerability remediation process?

Appendix C

Supplemental Materials for Chapter 5

C.1 Survey 1: Organizational Survey

This section lists all the questions we included in the survey that we sent to developer organizations.

Section 1: Data Collection and Transmission

1. Approximately how many mobile apps does your organization develop or distribute that are available through Google Play?
2. Are any of these apps designed for children?
3. Do any of your apps transfer data to third parties (e.g., ad networks or other domains)?
4. If Yes, please list the third parties that receive data from your apps.
5. What types of data do you collect from the users of your apps?
6. If your app collects data from users, for what purposes do you use the data that you collect? (If your app(s) do not collect data from users, please indicate that in your answer)
7. When your app transmits data over the Internet, how confident are you that it always uses encryption (i.e., SSL, TLS)?
8. Do your apps have privacy policies?
9. Do your app(s)' privacy policies mention all the data collection and sharing behaviors of your app(s)?
10. Do your app(s) obtain parental consent before collecting certain types of data from child users?
11. If Yes, how do your apps obtain parental consent before collecting users' data?
12. Does your organization think that parental consent needs to be obtained before collecting data from children users?
13. Do any of your apps use age gates to separate child users from adult users?

14. If Yes, how is data from child users treated differently, after a child user is identified using your age gate?
15. Does your app obtain users' consent prior to sending personal information to third parties?
16. If Yes, what mechanism do you use to obtain user consent?
17. Does your app allow users to opt out of the collection of their personal information?
18. If Yes, what mechanism(s) does your app use to allow users to opt out of data collection?

Section 2: Privacy Compliance Processes

1. Does your organization have a formal process for addressing privacy issues during the software development process?
2. Is your organization familiar with any of the following laws? (Choices: COPPA, CCPA, GDPR, None)
3. Which of the following laws does your organization believe your apps are subject to compliance with? (Choices: COPPA, CCPA, GDPR, None)
4. If your organization believes that your app is exempt from COPPA/GDPR/CCPA, which of the following does your organization think apply to your app(s)? (Choices: Our app(s) is not directed to kids, The users can just opt out by enabling the Opt out of Ads Personalization on their phones, Our app does not target users in the US, We are allowed under the law to collect data for our internal operations [please explain], We believe that SDK companies are responsible for protecting data they receive from our app, Other [please explain])
5. Please describe the processes your organization has in place for complying with various privacy regulations.
6. When your organization integrates third-party code into its apps (e.g., SDKs), what processes does it follow to ensure that those third-party components do not violate your organization's privacy obligations?
7. Do you offer both free and paid versions of your app(s)?
8. If Yes, does your organization treat paid apps differently than free apps in terms of the types of data you collect from users or the types of privacy features offered in your apps?

Section 3: Privacy Laws - COPPA

1. Does your organization believe that your apps are compliant with COPPA?
2. What obligations does your organization believe it has under COPPA?
3. COPPA applies to apps that are directed to kids (under the age of 13) in the US. Therefore, your apps may be subject to COPPA if you make them available to kids in the US through Google's Play store. Under COPPA, developers are required to obtain verifiable parental consent before collecting certain types of data from kids. Does this

change your organization's beliefs/understanding about your organization's obligations under COPPA?

Section 4: Privacy Laws - GDPR

1. Does your organization believe that your apps are compliant with GDPR?
2. What does your organization believe their obligations are under GDPR?
3. GDPR applies to all users in Europe, and apps or websites that target users under the age of 16 are subject to the kids provisions of GDPR. Therefore, if you offer your app(s) to users under the age of 16 in any European country through Google's Play store, then your app(s) may be subject to compliance with GDPR. Does that change your organization's beliefs about your organization's obligations under GDPR?

Section 5: Privacy Laws - CCPA

1. Does your organization believe that your apps are compliant with CCPA?
2. What does your organization believe their obligations are under CCPA?
3. CCPA applies to all users in California, and apps or websites that target users under the age of 16 may be subject to the kids provisions of CCPA. Therefore, if your app is targeting children and is available to users in California through Google's Play store, then it may be subject to compliance with CCPA. Under CCPA, verifiable parental consent must be obtained in order to opt-in children users (whose ages are under 13) to the sale of their personal information. Does that change your organization's beliefs about your organization's obligations under CCPA?

Section 6: Final Reflections

1. Based on your organization's experience, what would make it easier for app developers to comply with privacy regulations?
2. Is there anything else that you would like to add?

C.2 Survey 2: Developer Perspectives

This section lists the questions we sent to personnel within developer organizations.

Section 1: General Questions

1. Which of the following roles best describes your job function?
2. How many mobile apps have you had a role in developing?
3. Do you individually develop or have you ever worked at an organization that develops mobile apps directed at kids under the age of 13?

4. Are you aware of the Designed for Families (DFF) program in the Google Play store?
5. Have you contributed to the development of any mobile apps that participate in the DFF program, now or previously?
6. If Yes: if given the choice, would you prefer to not participate in the DFF program?

Section 2: Privacy Regulations

1. Have you (or someone else at your organization) ever been notified by Google that one of your apps is not compliant with the Play Store's policies?
2. If Yes, please explain the reasons why Google thought that your app was not compliant with Play Store policies, including what policy they believed your app violated.
3. If you were asked to list all the third parties that get contacted by your apps, how confident are you that you would be able to identify all of them?
4. If you were asked to list all the types of personal data that your app(s) collect from users (including persistent identifiers), how confident are you that you will be able to identify all of them?
5. How knowledgeable are you about each of the following regulations: COPPA, GDPR, CCPA, CPRA and FILRPPA
6. How do you determine what laws/regulations are applicable to the mobile apps that you have a role in developing?
7. Which of the following resources do you rely on for guidance on how to comply with applicable privacy regulations? (please select all that apply) (Choices: In-house legal counsel, Online sources (e.g., websites and social media), Regular advice from external lawyers, The Google Play store, None of the above, Other [please explain])
8. How confident are you of your ability to comply with privacy regulations?
9. How confident are you that all third-party code embedded in your app(s) is correctly configured to comply with applicable privacy regulations?
10. How confident are you of being able to identify all the potential privacy issues that exist in your app(s)?
11. How confident are you of your ability to fix a privacy issue found in your app by an external tester?
12. How confident are you of your ability to envision the risks your users might experience should they exploit a privacy vulnerability that exists in your app (e.g., an attacker using location data collected by your app)?
13. How challenging is it to make your app(s) compliant with privacy regulations?
14. Please explain your answer to the previous question.
15. What role should central app markets, like the Google Play store or Apple App Store, take in determining whether mobile apps comply with applicable privacy regulations? (Choices: Smaller role, Stay the same, Larger role)
16. What, if anything, should central app markets, like the Google Play store or Apple App

Store, do differently to help developers comply with applicable privacy regulations?

17. How likely do you think you or your company are to be investigated by privacy regulators as a result of violating privacy regulations?
18. How frequently do you check for whether SDKs integrated in your app(s) have released new versions after your initial integrations?
19. How often do you release updates aimed at fixing a privacy issue found in your app?
20. How concerned are you about the possibility of your app(s) being removed from the Play Store as a result of a discovered privacy issue?
21. How concerned are you about the possibility of experiencing a decline in the number of users who use your app(s) once they learn about a privacy issue (e.g., unexpectedly sharing their data with third parties)?
22. How concerned are you about the possibility of experiencing a decline in revenue if you must limit your apps' data collection practices?

Section 3: SDKs

1. Does your organization have a formal process that it uses to vet potential SDKs for inclusion in your app(s)?
2. Please explain the process your organization uses to decide whether or not to include a given SDK. For example, is this a decision left to individual developers? Is there a designated person or department that vets new SDKs?
3. Once a decision has been made to include a new SDK, is there anything that you or your organization does to ensure that the SDK has been configured correctly?
4. When integrating an SDK in your app(s), how confident are you that you or someone within your organization has read that SDK's: (a) Privacy policy?, (b) Terms of service?, (c) Quickstart guide? and (d) Configuration documentation?
5. What kind of information do you look for in SDK documentation when trying to make your app(s) compliant with privacy regulations?
6. Is there someone within your organization who is responsible for ensuring that the use of third-party SDKs complies with your organization's privacy policy? Please explain your answer.
7. Is there someone within your organization who is responsible for ensuring that the use of third-party SDKs complies with applicable laws? Please explain your answer.
8. How confident are you of your ability to control the types of data accessed and/or transmitted by SDKs integrated in your app(s)?
9. How confident are you of your ability to limit the transmission of personal data collected by an SDK integrated in your app with other third parties?

Section 4: Challenges and Recommendations

1. Briefly summarize the challenges you experience when trying to make changes in your app(s) to comply with privacy regulations.
2. What kind of support from the Google Play store do you think would make it easier for you to make your app(s) compliant with privacy regulations?
3. What kind of support from your organization's leaders do you think would make it easier for you to develop apps that are compliant with privacy regulations?
4. What kind of support do you need from other people in your organization to make it easier for you to develop apps that are compliant with privacy regulations?

C.3 Interview Guide

Below, we list the questions we used for our semi-structured interviews.

1. Are you aware of the following privacy regulations: COPPA, GDPR, CCPA?
2. If yes, do you believe that your child apps are compliant with COPPA, CCPA, and/or GDPR? Please explain.
3. Can you sketch the processes you and your team follow to comply with COPPA, CCPA, and/or GDPR?
4. Do these processes change when designing apps specifically targeted at children? If yes, how?
5. Would you change the design or add/remove features when designing apps specifically targeted at children? Please explain.
6. Have you ever been told by Google that your app is not compliant with the Play Store rules?
7. What tools do you use to test and/or monitor your compliance with COPPA, CCPA, and/or GDPR? How effective are these tools in helping you comply with these privacy regulations?
8. Do you think that you have the resources and/or (technical and legal) expertise that could help you become compliant with COPPA, CCPA, and/or GDPR?
9. Please explain what you think you would need to improve the compliance of your apps with these privacy regulations.
10. Do you believe it is difficult to comply with privacy regulations such as COPPA, CCPA and/or GDPR? What are the challenges that you experience when trying to comply with them?
11. What kind of third-party code (SDKs) do you integrate into the child apps you develop?
12. Can you explain the reasons why you need to integrate SDKs in your apps?
13. How do you decide whether it is okay to integrate an SDK in your apps or not? Can you explain your thought/decision process?

14. Do you read the privacy policies of SDKs before deciding to integrate them in your apps? If yes, what kind of information do you look for in such privacy policies? And do you find it helpful to do so?
15. Once you read SDK documentation, do you do any kind of testing to determine what kind of data is collected by SDKs?
16. Do you know the types of data collected by each SDK you integrate in your apps?
17. What challenges does your organization face when you integrate SDKs in your apps?
18. Are there any other third-party services that you use in your apps? And if so, how do you make sure that they are COPPA, CCPA, and/or GDPR compliant before you start using them?
19. What kind of questions do you consider asking third-party providers before you use their services?
20. Please provide some examples of privacy concerns that were raised to you by end-users (if any).
21. What are the reasons that may have caused the privacy concerns or issues we have just discussed?
22. What kind of support do you think developers need in order to comply with privacy regulations (COPPA, CCPA, GDPR)?
23. What do you think the implications are of not complying with COPPA, CCPA, and GDPR?
24. Are you concerned about violating the policies of Google Play?
25. Did you know that your app collects this kind of data from end users? Please explain the purpose of this data collection.
26. Did you know that your app communicates with these domains? Please explain the purpose of such communication.
27. Were you aware that these SDKs (refer to specific SDKs we detected in their app) collect this data?
28. Do any of these behaviors contradict what is mentioned in the privacy policy of your app?
29. Did you inform end users about these kinds of behaviors (collection and transmission of user data) in your privacy policy or any other means?
30. Do you have any plans for limiting data collection and data sharing with third parties?
31. Do you have any plans for removing specific SDKs from your app? If yes, please explain.

Appendix D

Supplemental Materials for Chapter 6

D.1 Developer survey

This section lists the questions included in our developer survey which helped us evaluate the extent to which configuring third-party SDKs for child-appropriate treatment and preparing accurate privacy labels are supported within existing app development processes. The survey also asked developers about the types of privacy guidance that they relied on and their perspectives on how such guidance can be improved.

Part 1: Third-party Software Development Kits (SDKs)

1. Which of the following third-party SDK providers receive personal data from the child-directed apps that you published on the Google Play Store?
2. For what purposes does your organization use third-party SDKs in its mobile apps?
3. Many third-party SDKs offer privacy configurations that can be used to signal to an SDK that children are among an app's target audiences, that a user has provided consent (or not) or to limit the collection of certain data types (e.g., advertising IDs and location data). Are you familiar with such types of third-party SDK privacy configurations?
4. To what extent does your organization make use of third-party SDK privacy-compliance configurations/settings to limit data sharing with third parties?
5. What are the potential consequences to your organization for not configuring third-party SDK privacy-compliance settings correctly?
6. To what extent does your organization find that it is challenging to correctly use third-party SDK privacy compliance configurations to comply with applicable privacy regulations?
7. What kind of challenges did your team experience in their effort to configure third-party SDKs for compliance with privacy regulations?

8. Is there someone within your organization responsible for ensuring that third-party SDKs bundled within your organization's mobile apps are kept up-to-date?
9. To what extent does your organization trust that personal data collected by third-party SDKs through your app(s) will not be used for purposes that are prohibited (e.g., behavioral advertising) under applicable privacy laws or the Google Play Store policies?
10. What kind of improvements/changes would your organization like third-party SDK providers to make in order to make it easier for your organization to use third-party SDK privacy compliance configurations correctly?

Part 2: Data Safety labels

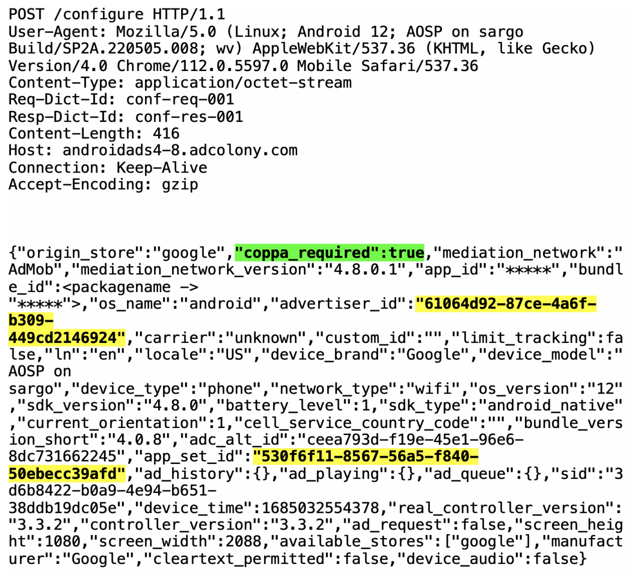
1. Who is responsible for filling out the details for your apps' data safety labels?
2. How would you rate the difficulty of accurately filling out the details required for data safety labels on the Google Play store?
3. Does your development process dedicate time and/or resources to ensuring the accuracy of information included in your apps' data safety labels?
4. If Yes, what kind of process did your team(s) follow to ensure the accuracy of the information included in your apps' data safety labels?
5. How does your team(s) identify the types of data collected by third-party SDKs before filling out your apps' data safety labels?
6. Do you use any dynamic or static testing tools that allow you to identify the types of data collected by your app(s) before filling out the details needed for the Data Safety labels (e.g., tools for capturing network traffic)?
7. If Yes, could you provide more details on the tools that you use for testing your app(s) to identify the types of data your app(s) collect and/or share?
8. What kind of challenges did your team(s) face while filling out the details required for your apps' data safety labels?
9. If you were to make changes to the Google Play developer console to make the process of filling out data safety labels easier for app developers, what would you do?
10. Should filling out the details of the data safety label be automatically done by Google Play instead of leaving this task to app developers?
11. Do you believe that Google checks your app(s)' Data Safety label(s) for accuracy?
12. In your opinion, what are the potential consequences for having inaccuracies in the disclosures made in apps' data safety labels?

Part 3: Privacy Guidance for App Developers

1. What kind of resources do you have access to through your organization for guidance on how to improve the privacy of your app(s) or comply with applicable privacy regulations?

2. Does your developer organization currently provide you with specific guidance or resources to help you improve the privacy of your app(s) or comply with applicable privacy regulations?
3. If Yes, what kind of privacy guidance do you currently have access to?
4. If you were to be offered privacy guidance, would you use it in your development tasks?
5. If Yes, what type of privacy guidance would you most want to have?
6. Where, in your development process, do you think that developer privacy guidance would be most likely to be considered?
7. If you were to be offered a specific type of privacy developer guidance, in what format would you like to receive this guidance?
8. Where would you like to see the privacy developer guidance presented to you?
9. What kind of content in third-party SDK documentation do you find challenging to understand or find?
10. What kind of help do you need to understand the data collection behaviors of third-party SDKs integrated in your app?
11. If you were to be offered a specific type of documented privacy developer guidance, what type of technical content would you like this guidance to have?
12. If you were to be offered a specific type of documented privacy developer guidance, what type of legal content would you like this guidance to have?
13. Assuming that we provided you with documented guidance to help you improve the privacy of your app(s), what else would you need?

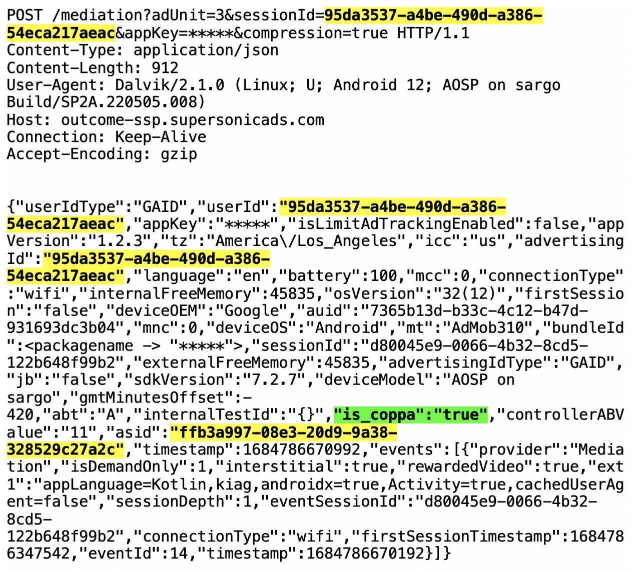
D.2 Traffic Snapshots



```
POST /configure HTTP/1.1
User-Agent: Mozilla/5.0 (Linux; Android 12; AOSP on sargo
Build/SP2A.220505.008; wv) AppleWebKit/537.36 (KHTML, like Gecko)
Version/4.0 Chrome/112.0.5597.0 Mobile Safari/537.36
Content-Type: application/octet-stream
Req-Dict-Id: conf-req-001
Resp-Dict-Id: conf-res-001
Content-Length: 416
Host: androidads4-8.adcolony.com
Connection: Keep-Alive
Accept-Encoding: gzip


{"origin_store":"google","coppa_required":true,"mediation_network":"
AdMob","mediation_network_version":"4.8.0.1","app_id":"*****","bundl
e_id":<packagename ->
"*****">,"os_name":"android","advertiser_id":"61064d92-87ce-4a6f-
b309-
449cd2146924","carrier":"unknown","custom_id":"","limit_tracking":fa
lse,"ln":"en","locale":"US","device_brand":"Google","device_model":"
AOSP on
sargo","device_type":"phone","network_type":"wifi","os_version":"12"
,"sdk_version":"4.8.0","battery_level":1,"sdk_type":"android_native"
,"current_orientation":1,"cell_service_country_code":"","bundle_vers
ion_short":"4.0.8","adc_alt_id":"ceea793d-f19e-45e1-96e6-
8dc731662245","app_set_id":"530f6f11-8567-56a5-f840-
50ebecc39afd","ad_history":{},"ad_playing":{},"ad_queue":{},"sid":"3
d6b8422-b0a9-4e94-b651-
38ddb19dc05e","device_time":1685032554378,"real_controller_version":
"3.3.2","controller_version":"3.3.2","ad_request":false,"screen_heig
ht":1080,"screen_width":2088,"available_stores":["google"],"manufact
urer":"Google","cleartext_permitted":false,"device_audio":false}
```

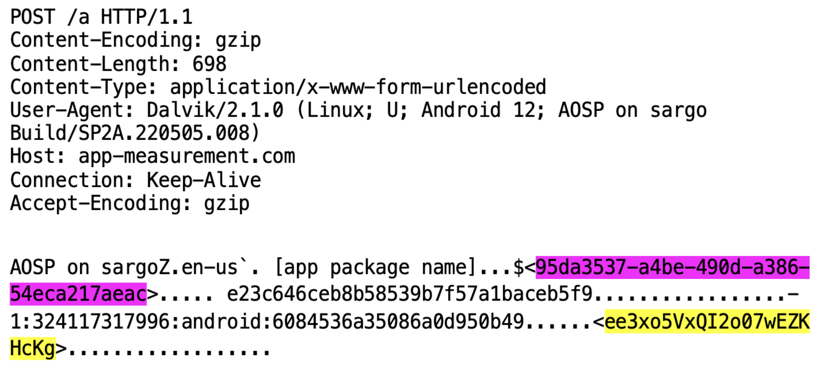
Figure D.1: Collection of AAIDs and app set ID alongside each other despite correct use of AdColony's [8] COPPA configuration.



```
POST /mediation?adUnit=3&sessionId=95da3537-a4be-490d-a386-
54eca217aeac&appKey=*****&compression=true HTTP/1.1
Content-Type: application/json
Content-Length: 912
User-Agent: Dalvik/2.1.0 (Linux; U; Android 12; AOSP on sargo
Build/SP2A.220505.008)
Host: outcome-ssp.supersonicads.com
Connection: Keep-Alive
Accept-Encoding: gzip


{"userIdType":"GAID","userId":"95da3537-a4be-490d-a386-
54eca217aeac","appKey":"*****","isLimitAdTrackingEnabled":false,"app
Version":"1.2.3","tz":"America\/Los_Angeles","icc":"us","advertising
Id":"95da3537-a4be-490d-a386-
54eca217aeac","language":"en","battery":100,"mcc":0,"connectionType"
:"wifi","internalFreeMemory":45835,"osVersion":"32(12)","firstSessio
n":"false","deviceOEM":"Google","auid":"7365b13d-b33c-4c12-b47d-
931693dc3b04","mnc":0,"deviceOS":"Android","mt":"AdMob310","bundleId
":<packagename -> "*****">,"sessionId":"d80045e9-0066-4b32-8cd5-
122b648f99b2","externalFreeMemory":45835,"advertisingIdType":"GAID",
"jb":"false","sdkVersion":"7.2.7","deviceModel":"AOSP on
sargo","gmtMinutesOffset":-
420,"abt":"A","internalTestId":"{}","is_coppa":"true","controllerABV
alue":"11","asid":"ffb3a997-08e3-20d9-9a38-
328529c27a2c","timestamp":1684786670992,"events":[{"provider":"Media
tion","isDemandOnly":1,"interstitial":true,"rewardedVideo":true,"ext
1":"appLanguage=Kotlin,kiag,androidx=true,Activity=true,cachedUserAg
ent=false","sessionDepth":1,"eventSessionId":"d80045e9-0066-4b32-
8cd5-
122b648f99b2","connectionType":"wifi","firstSessionTimestamp":168478
6347542,"eventId":14,"timestamp":1684786670192}]}
```

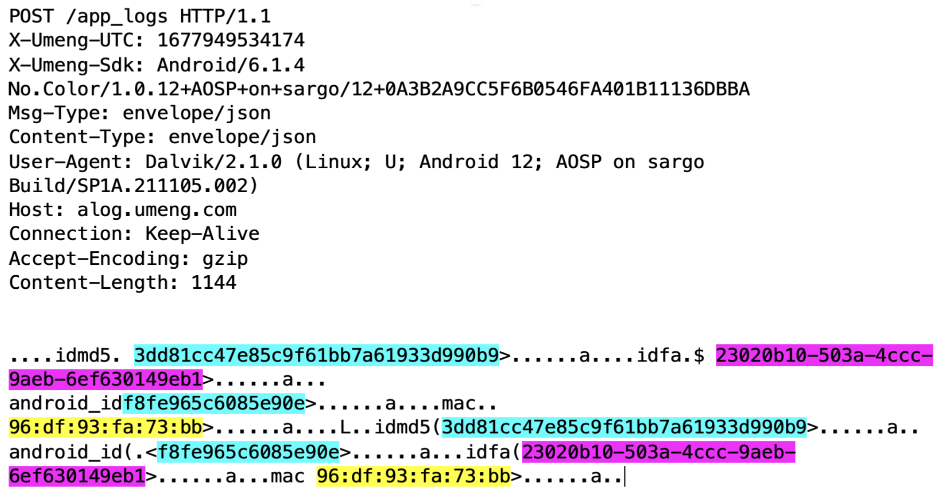
Figure D.2: Collection of AAIDs and app set ID despite correct use of ironSource's [131] COPPA configuration.



```
POST /a HTTP/1.1
Content-Encoding: gzip
Content-Length: 698
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 12; AOSP on sargo
Build/SP2A.220505.008)
Host: app-measurement.com
Connection: Keep-Alive
Accept-Encoding: gzip

AOSP on sargoZ.en-us`. [app package name]...$<95da3537-a4be-490d-a386-
54eca217aeac>..... e23c646ceb8b58539b7f57a1baceb5f9................-
1:324117317996:android:6084536a35086a0d950b49......<ee3xo5VxQI2o07wEZK
HcKg>.................
```

Figure D.3: Transmitting AAIDs alongside FIDs to app-measurement.com.



```
POST /app_logs HTTP/1.1
X-Umeng-UTC: 1677949534174
X-Umeng-Sdk: Android/6.1.4
No.Color/1.0.12+AOSP+on+sargo/12+0A3B2A9CC5F6B0546FA401B11136DBBA
Msg-Type: envelope/json
Content-Type: envelope/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 12; AOSP on sargo
Build/SP1A.211105.002)
Host: alog.umeng.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 1144

....idmd5. 3dd81cc47e85c9f61bb7a61933d990b9>......a....idfa.$ 23020b10-503a-4ccc-
9aeb-6ef630149eb1>......a...
android_idf8fe965c6085e90e>......a....mac..
96:df:93:fa:73:bb>......a....L..idmd5(3dd81cc47e85c9f61bb7a61933d990b9>......a..
android_id(.<f8fe965c6085e90e>......a...idfa(23020b10-503a-4ccc-9aeb-
6ef630149eb1>......a...mac 96:df:93:fa:73:bb>......a..
```

Figure D.4: Transmitting AAIDs alongside WiFi MAC addresses and SSAIDs to umeng.com.

D.3 Linking of Different Data Types

Figure D.5 shows the total number of apps that sent AAIDs alongside other types of personal data, which allows bridging AAIDs after they have been reset by users. Most of the linking was due to transmitting FIDs alongside AAIDs to app-measurement.com, which is owned by Firebase [93].

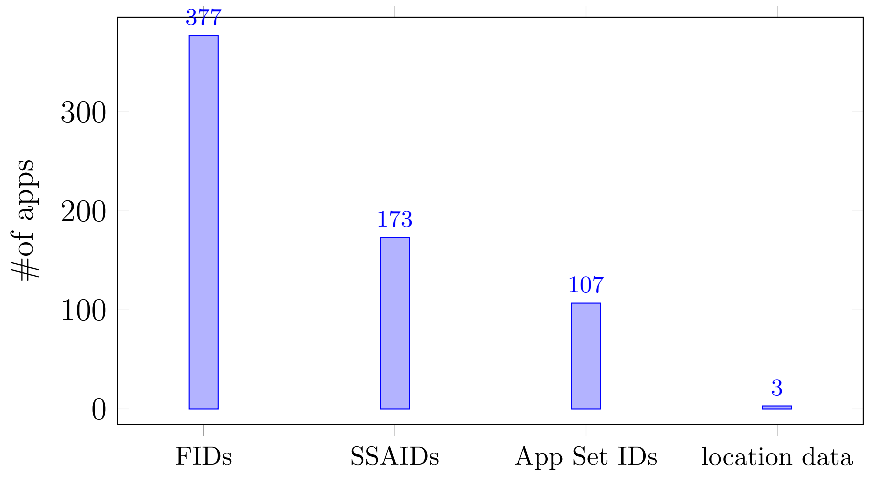
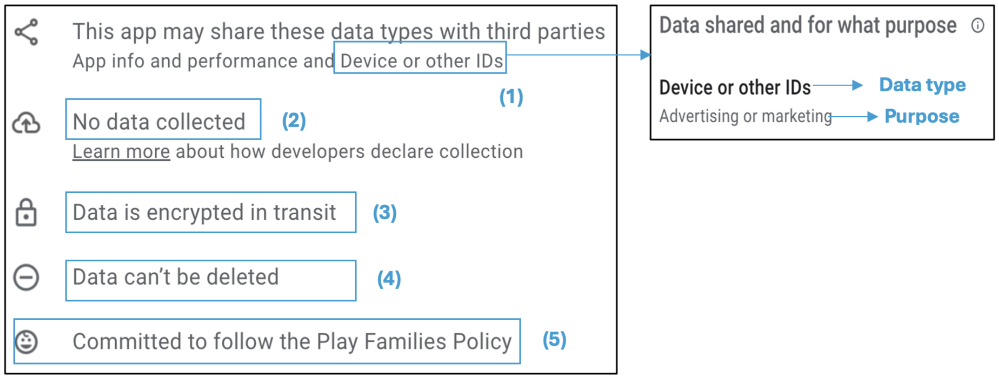


Figure D.5: Number of apps that transmitted AAIDs alongside other types of personal data to app-measurement.com and other domains.

D.4 Example of a Data Safety Label

Figure D.6 shows an example of a data safety label prepared by a developer for one of their DFF-badged apps (5). In this example, the developer disclosed that their apps shares two of the data types defined in Google Play's specification for data safety labels [51], which are "Device IDs" and "App info and performance". The developer also indicated that their app does not collect personal data (2), that encryption is used in their communications that include the disclosed data types (3) and that users cannot request the deletion of their data (4). In our analyses, we examined the accuracy of these disclosures by comparing them to apps' actual data collection and sharing behaviors. For example, if we observed the app whose label is included in Figure D.6 transmitted AAIDs to *live.chartboost.com*, we considered that an accurate disclosure.



This app may share these data types with third parties
App info and performance and Device or other IDs (1)

No data collected (2)
Learn more about how developers declare collection

Data is encrypted in transit (3)

Data can't be deleted (4)

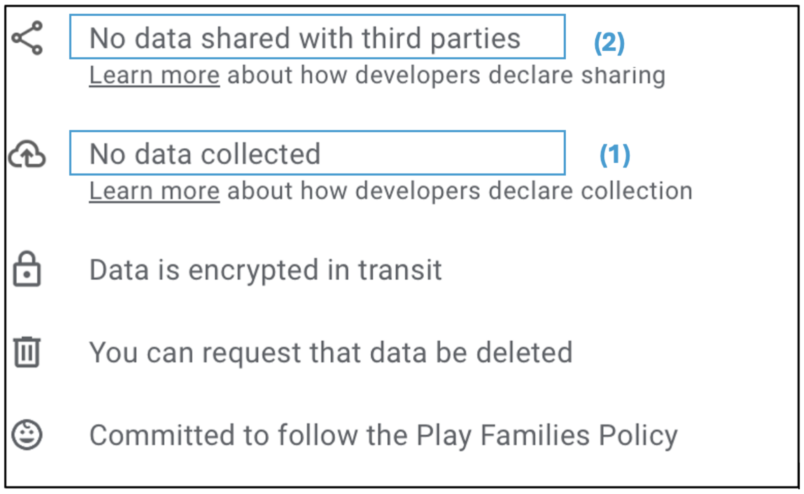
Committed to follow the Play Families Policy (5)

Data shared and for what purpose

Device or other IDs → Data type
Advertising or marketing → Purpose

Figure D.6: Example of a data safety label.

Figure D.7 shows an example of data safety label whose developer disclosed that their app does not collect (1) or share (2) any of the data types considered in Google Play's specification for data safety labels [51]. In this example, if we observed that the app whose label is included in Figure D.7 transmitted AAIDs to *graph.facebook.com*, we considered that an inaccurate disclosure of the transmitted data type and the purpose of such transmission.



No data shared with third parties (2)
Learn more about how developers declare sharing

No data collected (1)
Learn more about how developers declare collection

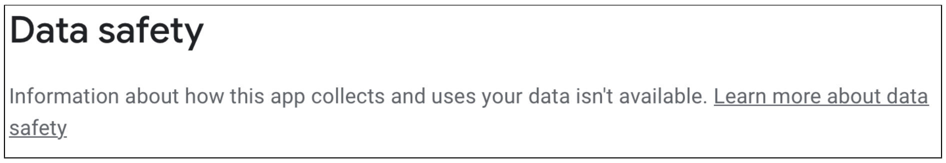
Data is encrypted in transit

You can request that data be deleted

Committed to follow the Play Families Policy

Figure D.7: Example of a data safety label disclosing that no data is collected or shared.

Figure D.8 shows an example of an app that did not have a data safety label available on the store at the time of testing. This could resulted from developers not preparing a label for their apps or preparing one that the store removed after finding inaccurate disclosures [198].



Data safety

Information about how this app collects and uses your data isn't available. Learn more about data safety

Figure D.8: Example of an app that did not have a data safety label.

D.5 Timeline of Policy Changes

Figure 6.1 provides an timeline that demonstrates when relevant Google Play policies were introduced or became effective. This timeline is based on the details available on Google Play's policy archive [180]. The dates shown in Figure 6.1 might not necessarily demonstrate whether these policies were enforced by Google Play at the time. However, anecdotal evidence from developer forums could provide evidence about the dates on which Google Play started enforcing its policies. For example, developers posted questions in late 2022 on StackOverflow (e.g., [197, 198]) that requested guidance on how to fix their data safety labels to react to notifications they received from Google Play.

To our knowledge, all the policies relevant to our experiments were effective before we started our experiments (early February 2023) except for two of them. One of which required developers to update the API levels targeted by their apps and the other one required using specific versions for self-certified SDKs in child-directed apps. The ongoing enforcement of Google Play's target API level policy that was happening during or after our pilot tests could explain why we were only able to test 4,975 apps in the second app runs (Section 6.3). While developers of child-directed apps were required to comply with the SDK versions policy starting from May 31 2023 [178], the policy was announced in late November 2022 which could have led developers to update SDKs to one of their more recent versions [179].

Appendix E

Supplemental Materials for Chapter 7

E.1 Privacy Checklist: Initial Version

In the initial version of the privacy checklist, we included five sections which ask developers to consider checking for the existence of a number of privacy issues, which mainly relate to transmission of personal data, disclosure of data practices and usage of third-party SDKs. The content of this checklist is included in the following sections.

Section 1: App Target Audiences

1. Are children among your app's target audiences? If No, skip Step 1-a.
    a) If Yes, did you indicate on your app's listing on the store that your app is child-directed?
2. Will your app(s) be used by child users in the United States where COPPA applies?
3. Will your app(s) be used by users in California where CCPA applies?
4. Will your app(s) be used by users in Europe where GDPR applies?

Section 2: Types of Data Your App Collects

1. Have you identified the types of personal data your app(s) collects from users as a first party?
2. For each data type you identified in Step 2.1, have you checked whether you need to obtain user consent (or verifiable parental consent for child users) before collecting the data type from users? (hint: you can make this determination by checking what each of the laws identified in Steps 1.2, 1.3 and 1.4 requires from app developers.)
3. For each data type you identified in Step 2.1, have you disclosed its collection in your privacy policy and your app's Data Safety label on the Google Play Store?
4. For each data type you identified in Step 2.1, have you disclosed the purpose of its collection in your privacy policy and app's Data Safety label on the Google Play Store?

Section 3: Third-Party SDKs Embedded in Your App

1. Have you enumerated all the third-party SDKs integrated in your app?
2. If you answered Yes to Step 1.1, have you checked whether the terms of service of each third-party SDK you identified in Step 3.1 allows its integration in apps that target children?
3. For each third-party SDK you identified in Step 3.1, have you checked its documentation to understand what data it will collect from your users?
4. For each data type you identified in Step 3.3, have you checked whether you need to obtain user consent (or verifiable parental consent for child users) before collecting the data type through the use of the SDK that collects it? (hint: you can make this determination by checking what each of the laws identified in Steps 1.2, 1.3 and 1.4 requires from app developers.)
5. For each data type you identified in Step 3.3, have you disclosed its collection in your privacy policy and your app's Data Safety label on the Google Play Store?
6. For each data type you identified in Step 3.3, have you disclosed the purpose of its collection in your privacy policy and app's Data Safety label on the Google Play Store?
7. For each third-party SDK you identified in Step 3.1, have you checked whether the SDK asks developers to include specific disclosures/clauses in their apps' privacy policies?
8. For each third-party SDK you identified in Step 3.1, have you checked that it is up-to-date?
9. For each third-party SDK you identified in Step 3.1, have you identified the privacy settings it offers and followed the steps provided in the SDK documentation to configure each of these settings properly?
   a) If you answered Yes to Step 1.1, have you checked whether each of the third-party SDKs you identified in Step 3.1 has client or server-side privacy settings that allow signaling to the SDK that children are among your app's target audiences and used the settings accordingly?
   b) If you answered Yes to Steps 1.2, 1.3 and 1.4, have you checked whether each of the third-party SDKs you identified in Step 3.1 has privacy settings that allow signaling whether a user has consented and used the settings accordingly?
   c) If you answered Yes to Steps 1.1 and 1.3, have you checked whether each of the third-party SDKs you identified in Step 3.1 has privacy settings that allow opting out child users from the sale of their personal information and used the settings accordingly?
   d) If you answered Yes to Steps 1.1, 1.2 or 1.3, have you checked whether each of the third-party SDKs you identified in Step 3.1 has privacy settings that allow limiting the collection of personal data from users until they provide their consent (or verifiable parental consent for child users)?

Section 4: Other Disclosures

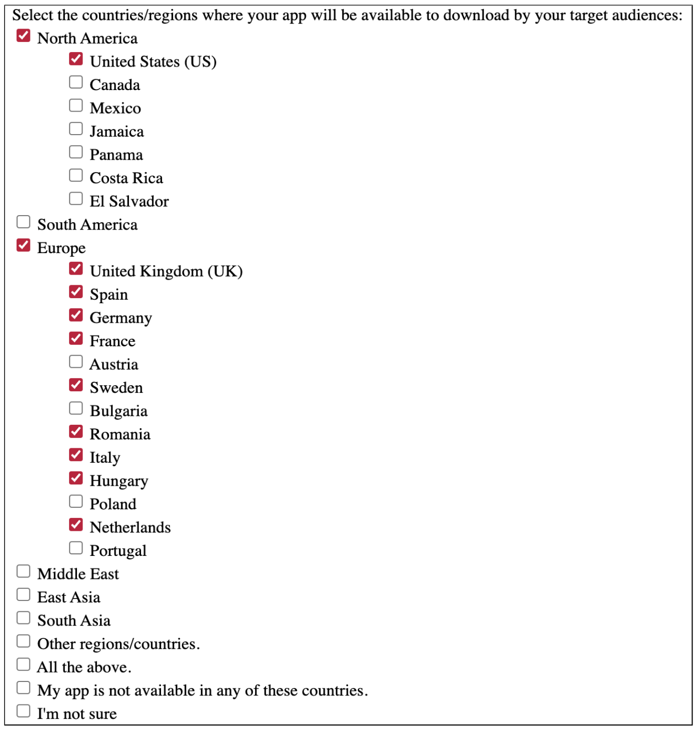
1. In your app's privacy policy, have you explained how users can submit requests to: enquire about the data your app collected about them, request the deletion of their data, and opt them out of the collection of personal data?
2. In your app's privacy policy, have you listed all the third parties (or categories of third-party businesses) that will receive user personal data collected by your app?
3. In your app's Data Safety label, have you listed the types of access requests that users can send to you (i.e., data deletion requests, data access requests, and reqeusts to opt out from data collection and/or sharing)?

Section 5: Resource Access and Data Security

1. Have you removed any unnecessary and/or dangerous permissions your app has before releasing it to users on the Google Play store?
2. Have you used TLS to encrypt all the data transmissions generated from your app?
3. Have you ensured that any user personal data stored in your systems is encrypted at rest?

E.2 Interactive Privacy Checklist: Questionnaire Part

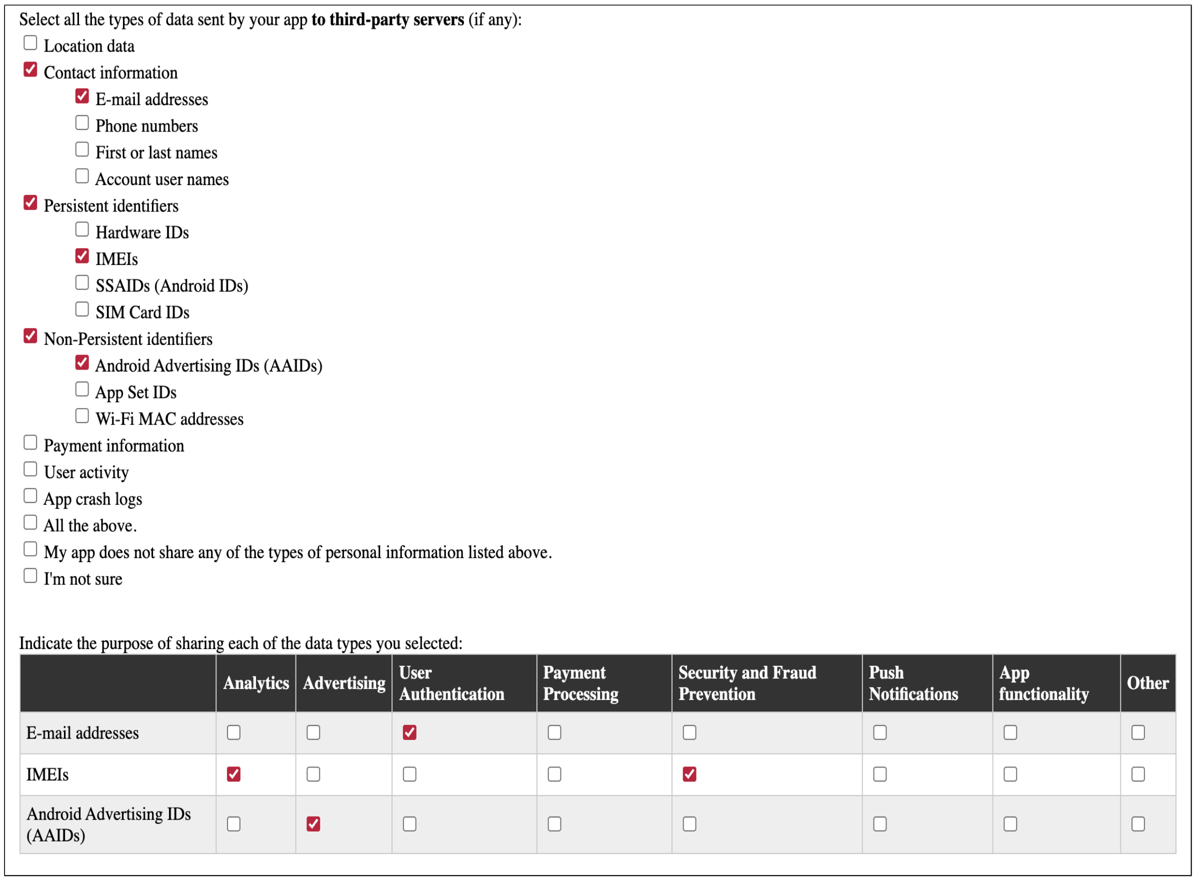
Below, we list examples of the questions that we included in the questionnaire part of our interactive privacy checklist tool.



Select the countries/regions where your app will be available to download by your target audiences:

- [✓] North America
  - [✓] United States (US)
  - [ ] Canada
  - [ ] Mexico
  - [ ] Jamaica
  - [ ] Panama
  - [ ] Costa Rica
  - [ ] El Salvador
- [ ] South America
- [✓] Europe
  - [✓] United Kingdom (UK)
  - [✓] Spain
  - [✓] Germany
  - [✓] France
  - [ ] Austria
  - [✓] Sweden
  - [ ] Bulgaria
  - [✓] Romania
  - [✓] Italy
  - [✓] Hungary
  - [ ] Poland
  - [✓] Netherlands
  - [ ] Portugal
- [ ] Middle East
- [ ] East Asia
- [ ] South Asia
- [ ] Other regions/countries.
- [ ] All the above.
- [ ] My app is not available in any of these countries.
- [ ] I'm not sure

Figure E.1: Question on the the countries where apps are available to users.



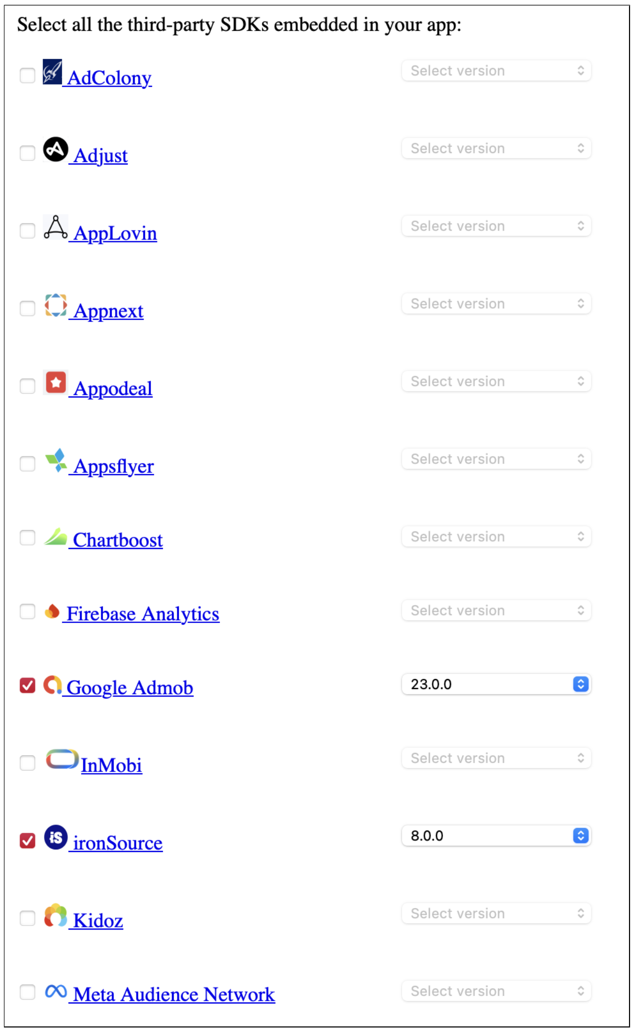
Select all the types of data sent by your app **to third-party servers** (if any):

- ☐ Location data
- [✓ Contact information]
  - [✓ E-mail addresses]
  - ☐ Phone numbers
  - ☐ First or last names
  - ☐ Account user names
- [✓ Persistent identifiers]
  - ☐ Hardware IDs
  - [✓ IMEIs]
  - ☐ SSAIDs (Android IDs)
  - ☐ SIM Card IDs
- [✓ Non-Persistent identifiers]
  - [✓ Android Advertising IDs (AAIDs)]
  - ☐ App Set IDs
  - ☐ Wi-Fi MAC addresses
- ☐ Payment information
- ☐ User activity
- ☐ App crash logs
- ☐ All the above.
- ☐ My app does not share any of the types of personal information listed above.
- ☐ I'm not sure

Indicate the purpose of sharing each of the data types you selected:

	Analytics	Advertising	User Authentication	Payment Processing	Security and Fraud Prevention	Push Notifications	App functionality	Other
E-mail addresses	☐	☐	✓	☐	☐	☐	☐	☐
IMEIs	✓	☐	☐	☐	✓	☐	☐	☐
Android Advertising IDs (AAIDs)	☐	✓	☐	☐	☐	☐	☐	☐

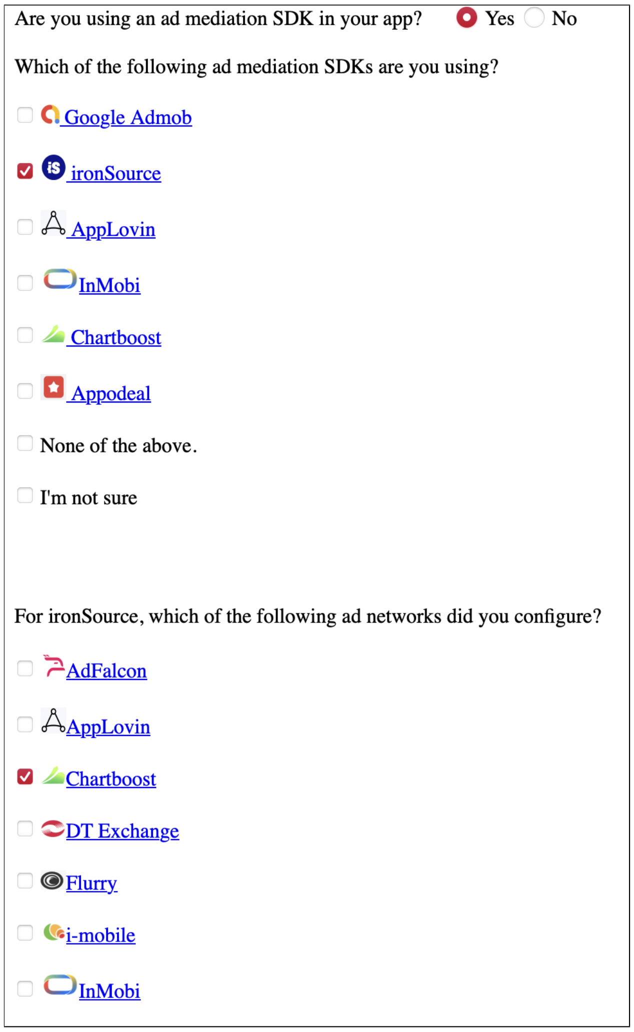
Figure E.2: Question on the types of data shared with third parties.



Select all the third-party SDKs embedded in your app:

☐	AdColony	Select version
☐	Adjust	Select version
☐	AppLovin	Select version
☐	Appnext	Select version
☐	Appodeal	Select version
☐	Appsflyer	Select version
☐	Chartboost	Select version
☐	Firebase Analytics	Select version
☑	Google Admob	23.0.0
☐	InMobi	Select version
☑	ironSource	8.0.0
☐	Kidoz	Select version
☐	Meta Audience Network	Select version

Figure E.3: Question on whether the third-party analytics or advertising SDKs are integrated.



Are you using an ad mediation SDK in your app? [✓ Yes] No

Which of the following ad mediation SDKs are you using?

- ☐ Google Admob
- ☑ ironSource
- ☐ AppLovin
- ☐ InMobi
- ☐ Chartboost
- ☐ Appodeal
- ☐ None of the above.
- ☐ I'm not sure

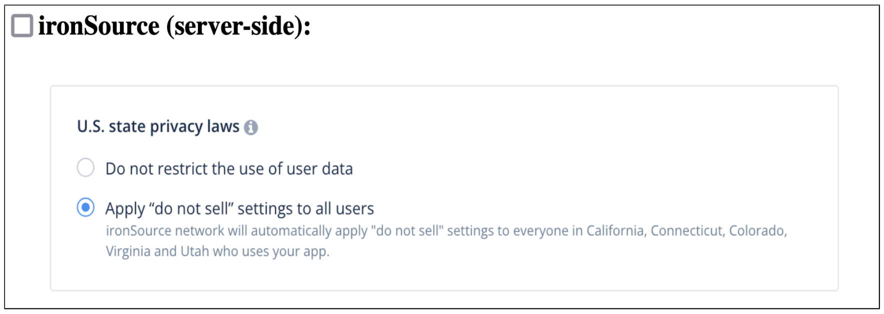
For ironSource, which of the following ad networks did you configure?

- ☐ AdFalcon
- ☐ AppLovin
- ☑ Chartboost
- ☐ DT Exchange
- ☐ Flurry
- ☐ i-mobile
- ☐ InMobi

Figure E.4: Question on whether a third-party ad mediation SDK is used.

E.3 Interactive Privacy Checklist: More Examples of Checklist Items

Below, we provide more examples of checklist items that are dynamically generated by our interactive privacy checklist.



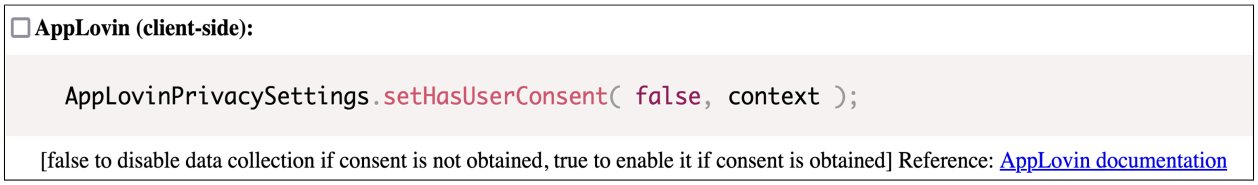
☐ **ironSource (server-side):**

U.S. state privacy laws

Do not restrict the use of user data [✓ Apply "do not sell" settings to all users]

ironSource network will automatically apply "do not sell" settings to everyone in California, Connecticut, Colorado, Virginia and Utah who uses your app.

Figure E.5: A checklist item generated by our tool providing guidance on using a server-side privacy configuration offered by ironSource's SDK [131] for compliance with US state laws.

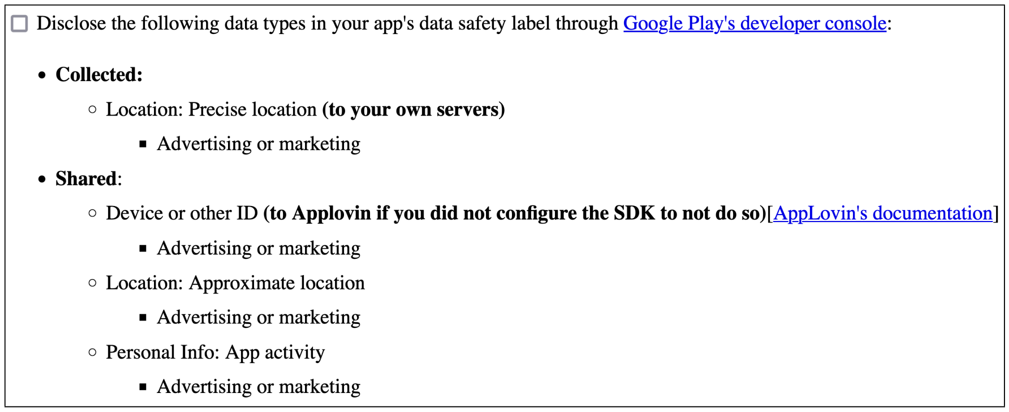


☐ **AppLovin (client-side):**

```
AppLovinPrivacySettings.setHasUserConsent( false, context );
```

[false to disable data collection if consent is not obtained, true to enable it if consent is obtained] Reference: AppLovin documentation

Figure E.6: A checklist item generated by our tool providing guidance on using a client-side configuration to communicate whether user consent was obtained.



☐ Disclose the following data types in your app's data safety label through Google Play's developer console:

- **Collected:**
  - Location: Precise location **(to your own servers)**
    - Advertising or marketing
- **Shared**:
  - Device or other ID **(to Applovin if you did not configure the SDK to not do so)**[AppLovin's documentation]
    - Advertising or marketing
  - Location: Approximate location
    - Advertising or marketing
  - Personal Info: App activity
    - Advertising or marketing

Figure E.7: A checklist item generated by our tool providing guidance on disclosures to include in an app's data safety label.

E.4 Interview Guide

Below, we list the questions that we asked participants during the interview sessions we conducted to obtain improved understanding of how developers use available sources of privacy guidance and evaluate the feasibility of using privacy checklists in their development processes.

Section 1: Privacy Guidance Available to Developers

1. What resources does your organization rely on for understanding how to comply with Google Play policies and applicable privacy regulations?
2. How did you find using the privacy guidance provided by Google Play (e.g., through their email messages, their web pages and/or Google Play console)?
3. What kind of uncertainties do you normally have with regards to how to check whether your apps have privacy issues before releasing them to users on the Google Play Store?
4. What types of privacy guidance do you currently use or have used before?
5. Have you ever used privacy checklists in your software development processes?
6. Would you be willing to use privacy checklists to help you identify and fix privacy issues before releasing your apps to users?
7. To what extent do you feel that you need privacy guidance to help you understand how to comply with Google Play policies and applicable privacy regulations? Please explain and specify the kinds of help and guidance that you need.

Section 2: Feedback and Reactions on Our Privacy Checklist

1. After reading each of the items included in the generated checklist, we asked participants:
    a) How easy would it be for you to take action?
    b) Is someone else in your organization better positioned to take action on them and/or answer the questions generated by the tool?
    c) Would you always be able to take action? Please explain.
2. Is any of the information included in the generated checklist unclear to you? And are there any terminologies that should be clarified? Please explain.
3. Is the amount of content included in the checklist reasonable? (i.e., should we add more detail or shorten it?)
4. If you were to be presented with the questions generated by the tool for all your apps, would you be able to answer them? Please explain.
5. Do you need to consult anyone in your organization before answering any of the questions generated by the tool? If so, who is that person? And how can we redesign this tool to make it easier for you to answer these questions?

6. Would you be willing to incorporate this tool into your development process? Please explain.
7. Which of your teams/personnel would be most likely to benefit from it? Please explain.
8. How can we make this tool better suit the needs of your development process?
9. After participants reviewed the content of the checklists they generated for their apps, we asked:
   a) Did the tool help you identify any privacy issues that exist in your app(s)?
   b) Did the tool help you understand how to fix these issues?
   c) Did the tool help you better understand the behaviors of the third-party SDKs embedded in your app(s)?
   d) Did the tool help you identify the disclosures that need to be included in your apps' privacy policy and/or data safety label?
   e) Did any of the checklist items help you develop a better understanding of your privacy compliance obligations? Please explain.
10. Do you already use privacy checklists in your development processes? If so, we asked:
    a) Where did you get your checklist from?
    b) Who developed it?
    c) What type of content does your privacy checklist contain?
    d) What process was followed to develop it?
    e) Who are audiences targeted by your privacy checklist?
    f) What are the reasons that motivated you to use a privacy checklist?
    g) How do you use it in your organization? And at which point in your development lifecycle do you use it?
    h) How does it guide your software development and test plans?
    i) Could you explain how using a privacy checklist impacted your privacy practices?
11. If you were to further customize this checklist to your development process and/or app(s), what kind of changes would you make to it?

E.5 Survey Questions

Below, we list the questions that we included in the survey that we distributed to privacy lawyers.

Section 1: General Questions

1. Do you work at a law firm or are you part of a legal team in a company/organization that develops or distributes mobile apps?
2. Does your firm (or legal team) have a group of lawyers that focuses on advising about privacy and/or security matters?

3. I have advised mobile app developers who operate in these countries .. [Options: United States, United Kingdom, Canada, Germany, Brazil, France, Spain, China, Japan, I prefer not to answer, Other]
4. I have advised clients to comply with ... [Options: The Children Online Privacy Protection Act (COPPA), The General Data Protection Regulation (GDPR), The California Consumer Privacy Act (CCPA), Brazilian General Data Protection Law (LGPD), The Personal Information Protection Law of the People's Republic of China (PIPL), The Health Insurance Portability and Accountability Act (HIPAA), The Gramm-Leach-Bliley Act, I prefer not to answer, Other]

Section 2: Privacy Guidance

1. How do you communicate your privacy compliance advice to mobile app developers?
2. Who is typically the target audience for your privacy guidance documents?
3. How is privacy guidance presented in the documents you provide to mobile app developers?
4. Have you ever used any of the privacy checklists available in law databases, such as Bloomberg Law or Westlaw, when you advise mobile app developers on how to comply with privacy regulations?
5. When do your clients (i.e., mobile app developers) typically implement the advice provided in your privacy guidance documents?

Section 3: The Content of the Provided Privacy Guidance

1. Is children's privacy covered in your oral or documented privacy guidance you provide to mobile app developers?
2. Generally speaking, which of the following topics is covered in the oral or documented privacy guidance you provide to mobile app developers? [a list of answer choices was provided]
3. In terms of designing consent screens, which of the following is addressed by the oral or documented privacy guidance documents you provide to mobile app developers? [a list of answer choices was provided]
4. In terms of working with third-party tools and code libraries, which of the following is addressed by the oral or documented privacy guidance you provide to mobile app developers? [a list of answer choices was provided]
5. In terms of preparing privacy policies and privacy labels (i.e., for display in app stores) for mobile apps, your oral or documented privacy guidance provide mobile app developers with information that allows them to understand how to ... [a list of answer choices was provided]
6. Which of the following is considered your responsibility (as a lawyer) when working with mobile app developers? [a list of answer choices was provided]

7. Which of the following is considered the responsibility of your clients (i.e., mobile app developers) and not yours (as a lawyer)? [a list of answer choices was provided]

Section 4: Follow-ups with Clients

1. After providing your privacy guidance to mobile app developers, what kind of follow-up do you normally have with them to check whether they have followed your guidance correctly?
2. How do you check whether your clients have followed your privacy guidance correctly?
3. Generally speaking, what kind of challenges do you face when providing privacy compliance advice/feedback to mobile app developers?
4. If you were to improve the oral or documented privacy guidance you provide to mobile app developers, what kind of changes would you make to this guidance to make it more effective for mobile app developers?