‘We are adults and deserve control of our phones’: Examining the risks and opportunities of a right to repair for mobile apps

Konrad Kollnig, Siddhartha Datta, Thomas Şerban von Davier, Max Van Kleek,
Reuben Binns, Ulrik Lyngs, Nigel Shadbolt
{firstname.lastname}@cs.ox.ac.uk
Department of Computer Science, University of Oxford
Oxford, United Kingdom

ABSTRACT

Many mobile apps are designed not just to support end-users’ needs, but also commercial aims. This can result in app designs that compromise end-user privacy, safety, and well-being. Since apps nowadays provide vital digital information and services, users often have no choice but to accept potentially harmful or manipulative app designs. What if, instead, individuals could customise their apps to make them safer and better suit their needs? This exploratory work examines this question through a multi-faceted approach; first, to understand user needs, we conducted a survey ($n = 100$) of changes users wanted in their apps, and of perceptions of risks in app repair. Second, to identify technical challenges, we developed a prototype that enables end-users to change their apps, and realised several modifications suggested by survey participants. Finally, we conduct a set of expert interviews ($n = 8$) to delve into the ethical and legal aspects of such a tool, and synthesise a framework of risks and opportunities of app repair.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in ubiquitous and mobile computing**; • **Security and privacy** → *Software and application security*.

KEYWORDS

mobile apps, dark patterns, digital harms, right to repair

ACM Reference Format:
Konrad Kollnig, Siddhartha Datta, Thomas Şerban von Davier, Max Van Kleek,, Reuben Binns, Ulrik Lyngs, Nigel Shadbolt. 2023. ‘We are adults and deserve control of our phones’: Examining the risks and opportunities of a right to repair for mobile apps. In *Proceedings of ACM Conference on Fairness, Accountability, and Transparency (FAccT ’23).* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Many essential digital technologies are developed and maintained by a handful of powerful tech companies that provide limited means for user participation and negotiation in software design. The practices of many of these platforms and services have, in many sectors, killed off feasible alternatives, exacerbated by an intentional lack of interoperability [20, 28, 37]. As a result, individuals have limited means to mitigate the wide array of harms present in those mobile apps that they rely on for their day-to-day lives. Sometimes, those apps even deliberately exploit their users. Such exploitation ranges from behavioural design (nudging people to take certain actions or become addicted to the product), intentional uses of user interface (UI) *deceptive designs* (also known as dark patterns) [54, 83, 91] (exploiting known psychological biases and weaknesses to benefit the company/service), to features that directly harm the user such as by undermining their privacy either by giving up their personal information directly or by tracking them behind the scenes [17, 76, 77]. Even if apps do not directly harm the majority of users, they often fail to account for the diversity and sensitivities of the most vulnerable groups of users [12, 42, 102].

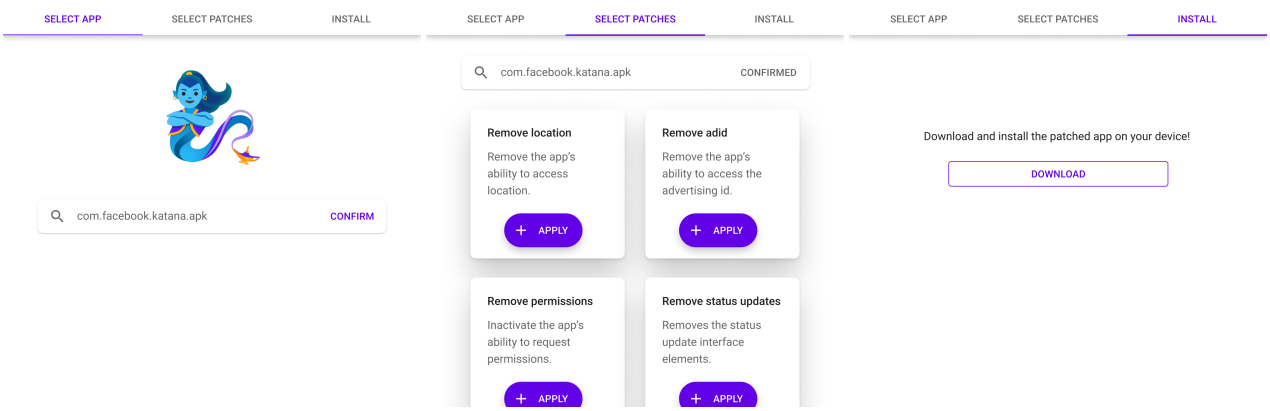
What if, instead of being provided limited choice and having to rely on oneself to know what to ignore, users were given the ability to re-configure and re-shape the essential features of apps – *a right to repair for apps* – to make them more suitable to their needs, including the possibility of removing elements that are hostile, manipulative, or harmful to them? Such capability could empower users both directly, by helping them improve their apps immediately, and, indirectly, by exerting pressure on app developers to address their concerns. Beyond eliminating user-hostile app elements, this ‘superpower’ could be beneficial to improving accessibility and realising more inclusive design. Previous research has found that most users are aware of deceptive designs and their influence on their behaviours, but aware is not enough to mitigate these harms [18]. Yet, like most technology that aims to ‘empower’, what can be used for good can also be used for harm; modifications to apps could bring about any number of unwanted harms to end-users, such as weakening app security, or adding malicious code that perpetuates criminal activity or steals from users.

This work aims to conduct an initial exploration of end-user app repair tools to explore the opportunities and challenges that such tools present. App repair tools allow individuals, like browser extensions on the web, to change – in theory – almost all aspects about an app. We aim to critically evaluate the pros and cons of such an approach, and study the feasibility, ethics, and impact of such a technology. Specifically, we focus on two research questions:

RQ1: *Needs* – Are there common, pressing, or important needs users have that require modification of their apps?
RQ2: *Barriers* – What are current practical, ethical, technical, societal and legal challenges in doing so?

FAccT ’23, June 12–15, 2023, Chicago, US

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Step 1: User selects an app. (b) Step 2: User selects patches. (c) Step 3: User installs patched app.

Figure 1: Our working technical prototype, GreaseDroid, empowers users with average technical skills to remove dark patterns and other harms from their Android apps in three steps. The tool aims to help us study the risks and opportunities of app repair in this paper.

Given the multi-faceted nature of these questions, which broach both user needs and technical challenges, as well as hinging on inherently normative questions, we study these questions using a *mixed-methods approach*. We first conducted a survey-mediated speculative design exercise with 100 participants and applied thematic analysis to understand what app changes individuals want (Section 3). To demonstrate technical feasibility of app repair and explore challenges, we created a prototype tool for Android (see Figure 1) and implemented some app changes that our survey participants wished for (Section 4); we also analyse legal questions raised by this approach in Section 4.3. Finally, we conduct a set of expert interviews ($n = 8$), discuss our results and examine the ethical dimensions and aspects of such a tool in Section 5. This aims to synthesise a theory of the risks in app repair, gather an understanding of the opportunities and challenges, and provide suggestions for the responsible (non-)deployment of this technology.

Importantly, we do not conduct a user study with app repair because a similar technology (browser extensions) has previously been tested in browsers. While we develop a prototype (based on 'app modification', see Section 2.3), the aim of this paper is not to advocate for any single implementation of app repair, but rather understand the long-term viability and implications of such a technology prior to wide deployment. Eventually, device manufacturers may include such a technology into their smartphone operating systems by default, e.g. because of consumer demand or law.

We will make our study materials (including the study design, survey responses, and interview data) as well as the technical prototype available to other researchers as to support future studies into overcoming app-based harms.

2 BACKGROUND

Given the multi-faceted nature of this work, we draw on several different areas of prior research: legal, social, and technical. First, we briefly cover the motivation for and development of the 'right to repair' movement in law, note its potential application to software, and highlight problems with current regulation of software. Then, we briefly summarise research into the harms, as well as deficiencies and inconveniences which might motivate the desire for app repair and changes. Lastly, we cover existing technical and policy measures designed to enable such changes on various platforms including the web, and the barriers they face on apps in particular.

2.1 Legal: Digital rights and right to repair

Throughout early human history, people have had control over the tools that they used, including how they were made [84, 94]. This was because people either crafted their tools, or had someone they trusted to do it for them along the way. Since technology in early human civilisation was relatively simple, it was relatively straightforward to fix if something broke down. Indeed, fixing and tinkering have been arguably the most common manifestation of human ingenuity, leading to innovation and better products in the long run. As products got more complicated over the past century, we have increasingly lost our ability to both make our own tools and fix the tools that we use [108]. For example, automobile owners could normally have their vehicles fixed by a local technician. Car manufacturers even provided detailed repair manuals and the necessary parts to do so. They were, after all, *manufacturers* and not repair shops, and sought to focus on what they were uniquely good at [101]. However, more recently, they have sought to become the sole providers of services necessary to use their products; famously, John Deere locked down some of its tractors, thereby rendering independent repair almost infeasible [4]. The increasing difficulties in repairing physical products sparked the worldwide *right to repair* movement [56, 84, 94, 107]. The right to repair movement encompasses electronic products including smartphones, but could also be extended beyond hardware to include software.

Beyond technical measures, there is a range of legal instruments that are currently used by platforms and developers to expand and enshrine their control over digital systems. This includes copyright law like the Digital Millennium Copyright Act (DCMA) that

is meant to protect copyright and original ideas, but also covers program code. In the past, device manufacturers further tried to use copyright law to go after end-user modifications, but with mixed success. For example, ad blockers have repeatedly been found to be legal [49, 89], along with jailbreaking iOS devices [115]. Some scholars suggest that the current legal regime, including copyright and other laws, disproportionately disadvantages the user over the developers of software [13, 56, 73, 94, 107, 117]. There is increasing pushback against the legal power of tech companies, through increased enforcement of data protection laws [9, 24–27, 30, 43, 69], antitrust and competition law [15, 28, 78, 117], and the enactment of new laws (such as the Digital Services and Markets Act in the EU). There is also a range of recent and ongoing proceedings in courts around *fair use* under copyright law [79, 89, 106, 115], most notably the US Supreme Court ruling on *Google LLC v. Oracle America, Inc.* that underlined the importance of fair use in balancing the interests of rights owners and other stakeholders [106].

2.2 Social: Digital harms and other challenges

Researchers working in HCI, privacy, security and beyond have examined a wide range of potential harms arising from the use of apps. These include the loss of privacy [17, 122], dark patterns [54, 83, 91], addiction and digital distraction [50, 76, 77], discrimination and algorithmic bias [12, 16] and exposure to harmful content, and harassment and bullying [61]. While some of these harms arise from inadequate mitigation or anticipated uses of apps in particular contexts, other harms, such as app addictiveness and privacy harms, arise from deliberate designs devised by app developers and platforms [38, 81]. Some digital harms will yet be unknown to us, or only affect so small communities that they are not talked about by the majority of researchers and the public. To counteract digital harms, users may engage in *critical ignoring*, in which they selectively choose what to pay attention to [71, 114]. Yet, critical ignoring requires training from users to become an effective tool [71].

Another set of problems that have been identified in apps pertain to a lack of accessibility, or other problems relating to access, inclusion, and assumptions relating to users and use contexts. For instance, several studies have discovered widespread lack of accessibility in mobile apps, including inadequate image labelling and poor contrast for the visually impaired, inconsistent use of widgets, among others [6, 39, 92, 113, 120]. Further concerns relate to incorrect assumptions about their users or their needs, such as those documented previously in fertility apps [42], fitness trackers [102], or navigation systems [12]. There are risks around the deepening of a 'digital divide' between those who manage to use digital technologies and those who struggle, such as the elderly [55, 58] or those with disabilities and other conditions [90, 116]. Inappropriate design can have significant negative impacts on affected individuals [12, 14, 16, 66, 72], particularly when algorithmic systems are used.

2.3 Technical: Empowering end-users to modify/fix apps and systems

The world of PC games has had a long storied history of communities creating modifications and enhancements to programs, often called patches or 'cheats' that often become circulated in dedicated communities [8, 63, 68, 82, 87]. Such patches were typically crafted by skilled programmers, who sometimes created and shared tools to simplify the process of reverse-engineering. With respect to the Web, the rise of Web 2.0 coincided with Aaron Boodman's creation of Greasemonkey in 2004, which he created because he wanted to improve the appearance and layout of AllMusic, one of his favourite desktop websites [86]. Greasemonkey enabled those with low-to-moderate web development experience to change the layout and functionality of websites by creating so-called *userscripts* and installing them in their Firefox browser within the Greasemonkey extension. A dedicated community of Greasemonkey users eventually authored and shared thousands of website-modifying scripts on userscripts.org, which included scripts to remove the feed from Facebook or increase the readability of Wikipedia. Greasemonkey, thus, represented the first app repair tool with widespread adoption for non-experts to modify applications created by others at scale. Similar technologies as used in Greasemonkey are now the backbone of browser extensions and ad blockers, which are still widely in use [103].

For mobile devices, relatively few solutions have been created to repair, enhance and modify apps. Methodologically, these solutions 1) modify the operating system [3, 41, 47, 93], 2) use System APIs (e.g. VPN-based ad blockers [2, 70], or overlay-based interventions [31, 32, 46, 51, 52, 52]), or 3) modify apps directly [10, 33, 34, 60, 65, 75, 95, 119]. All of these existing solutions for mobile come with certain limitations. While modifying the operating system can in principle make arbitrary modifications to the behaviour of apps, these usually rely on device vulnerabilities and are subject to the changing practices of device manufacturers. The use of System APIs might often be the most straightforward approach for a non-expert user, but is limited to what is permitted by the smartphone operating system. *App modification* allows for arbitrary modifications of a provided app (only limited by the constraints of the operating system), and can offer ease of use. As such, it combines benefits of system modification and System APIs. Installing custom apps is relatively easy on Android, but not iOS. Despite the potential of this approach, there exist hardly any solutions used in practice using app modification. Exceptions are the app cracking tool Lucky Patcher [75], the privacy tool SRT AppGuard [10], and *apk-mitm* [60] and *objection* [95], both for app security research. Some developers have published modified Android versions of popular apps, including Facebook [44] and YouTube [112] (removing ads and other distracting functions). Unfortunately, such modified apps rely on the continued support of the developers, have uncertain security properties, may break over time (e.g. in case of server-side updates), and exist only for a few select apps.

3 USER NEEDS AND PERCEPTIONS

To explore the space of potential user needs that might be served by an ability to extend and change the functionality of existing mobile apps, we conducted an open-ended survey with 100 participants and a fictional app repair tool. We studied what individuals, if given a chance, would like to change about the apps they often use concerning the user interface and underlying app logic

through a speculative role-playing design prompt, similar to previous work [80]. We chose to conduct a survey to be able to scale across more individuals and elicit a more diverse range of response than we could do with other methods, such as interviews.

Study Design. Our survey had three parts: 1) demographics and app use, 2) speculative role-playing design of app changes, and 3) user reflections. an estimate of their daily phone use. In the speculative design exercise, we introduced the concept of *app changes* through a speculative role-playing design prompt, similar to previous work by Merrill [80]. We asked participants to imagine being granted two 'app-modification superpowers' by a benevolent 'app genie', thereby comparing app changes to magic – almost anything is possible. We then repeatedly asked participants how they would use those two superpowers to change the apps that they often use. Such repeated questioning is a commonly used method in user research to spur creativity, e.g., by Sakichi Toyoda's 'Five Whys' [97]. Firstly, participants were asked to make between four and seven suggestions on how to change their three most used apps (as provided in the first part of the survey). This first part is visualised in Figure 4 in the Appendix. Secondly, participants were asked to make between three and six suggestions on how to change *any* app on their phone. We encouraged participants to take out their phones as part of this process to help the design exercise. In total, participants would make between seven and 13 suggestions for app changes. The full survey can be found in the Appendix.

Recruitment and Ethics Approval. Due to the qualitative nature of our survey, we did not need a large sample size to achieve saturation [59]. Therefore, we recruited 100 participants using Prolific, an online platform for research participant recruitment. We restricted recruitment to current or past Android users (who would be more likely to be able to adopt app repair technology, due to iOS restrictions), aged 18–40 years (to focus on those adults who spend most time with mobile devices for their day-to-day social and professional activities [104]). Participants were compensated for their time and paid the UK Real Living Wage. Our departmental ethics board approved the study (approval reference CS_C1A_021_021). It took participants a median of 19.8 minutes (SD = 11.8 minutes) to complete the survey.

Data Analysis and Sharing. For open-ended responses, we conducted a thematic analysis to look for underlying patterns within our dataset. Initially, we started with an inductive approach, where three authors of this paper independently developed an initial set of codes, from a different set of 100 suggested app changes. The authors then met and discussed the codes to achieve consensus and develop a codebook. Finally, the first author completed the coding using a deductive method, and formed themes based on the previously-identified codes. In total, we identified 29 codes, and 5 themes.

3.1 Results

3.1.1 Participant Demographics. Fifty-nine participants identified as men and 40 as women. One participant did not disclose this information. Fifty-one participants resided in Poland, 21 in South Africa, five in Portugal, five in Greece, four in Spain, four in Italy, and the remaining seven in other European countries (with the exception of one participant from Israel).

3.1.2 Overview of Suggested App Changes. Throughout the survey, the most commonly selected apps to make modifications to were Instagram (selected by 54 participants), Messenger (52), Facebook (49), YouTube (45), and WhatsApp (41). Participants suggested a total of 734 app changes. After a review of all changes, we excluded five responses because we did not understand them or did not include any changes. We also excluded another 166 suggestions that are not directly *actionable or relevant* for app repair, e.g., because they pertained to changes to server-side code or were too abstract.

The set of actionable and relevant app changes represented those that may actually be implemented through app repair and client-side changes. These changes related to three themes namely, i) making apps' business models more user-friendly and respectful, ii) the wish for more attractive, user-friendly, and accessible user interfaces, and iii) rebalancing privacy and security in apps. We describe each of these themes in the following.

Making apps' business models more user-friendly and respectful. In this theme, 60% of participants complained about how apps generate revenue. This was mainly related to advertising (42% of users), being the predominant monetisation model in the app ecosystem. (*[I would wish for] A built-in ad blocker in the browser that blocks intrusive ads (P65, Chrome)*) (*ads, ads, ads, ads, ads, ads, ads, ads, ads, ads (P65, YouTube)*)

Beyond ads, another common monetisation model on Android are in-app purchases. This model can, by its design, conflict with the actions that users take within apps. 4% of participants expressed such annoyance with regards to non-ad app monetisation. (*Less pay to win system and a better leaderboard for ranked (P56, Clash Royale)*) (*Paid options and subscriptions everywhere. It is very annoying when using the app (P66, Tinder)*) A related category comprises patches that provide paid features for free (wished for by 19% of participants). A common concern was monetisation by YouTube, which does not allow playing videos in the background without a subscription. (*I would have free access to premium content because most people cannot afford it (P71, YouTube)*) Since such modifications might constitute fraud, they should not be developed.

The wish for more attractive, user-friendly, and accessible user interfaces. Within this theme, the most common suggestions related to modifications to the user interface (UI) – mentioned by 92% of participants. A common complaint was that a specific app was not 'user-friendly' (P22) or not 'accessible' (P13) enough. A particular case of such UI improvements were cosmetic changes, e.g. to the colour or theme of certain apps. (*The app looks very primitive and uninspired, almost unprofessional. (P23, about Steam)*) (*Change the app to to another colour. I am getting tired of the blue bird (P1, about Twitter)*) Given that 39% of participants expressed such sentiments, there seems to be a rather widespread desire for self-expression and customisation in apps' user interfaces.

As for how to improve the UI, 47% of participants suggested removing certain interface elements or features. Many were unhappy with the prominently placed 'Reels' (P67) and 'Stories' (P95) in the Instagram app, the WhatsApp status page (P82), the promotion of podcasts in the Spotify app (P73), or the new TikTok-like YouTube 'Shorts' (P32). Participants wanted to hide these elements because all of these apps do not currently foresee options to so. (*I hate the "YouTube Shorts" feature, I don't use it at all and I don't find it necessary. I just don't need it. (P32, YouTube)*) (*Have options to hide*

Instagram stories (P95, Instagram)) Some participants complained about the metrification of social media apps and the resulting impact on individuals. Such can be present in many different forms on social media, e.g. in terms of number of friends, post views, and likes. Metrification allows users to judge the virality of social media content but can also put pressure on individuals to create and post more viral content. (*I would hide the amount of like when someone posts. because I feel that most people just post nonsensical thing just to get likes. (P94, Twitter)*) 15% of participants complained about forced actions, a kind of deceptive design [54], that forces individuals to take certain actions. One example is the design of Google Mail, where it is deliberately challenging to sign out only one user at a time. (*I would make it easier to log out and also not to log out all accounts at once. (P2, Google Mail)*) Another aspect that participants complained about was that Instagram recently replaced the 'New Post' button with a new button to access the new 'Reels' (i.e. short videos like on TikTok) – and wanted to *roll back to the previous design* (a class of patches mentioned by 8% of participants).

Rebalancing privacy and security in apps. The last theme related to changing apps' privacy and security practices (mentioned by 55% of participants). Users' desire for more privacy and security in apps is well documented in the research literature, as is the relative lack of such in practice [98, 111]. Recent initiatives like the General Data Protection Regulation (GDPR) in the EU and new privacy protections measures on iOS and Android have arguably further spurred this desire. This theme included the wish for protection from other users of the service (25%), as illustrated here: (*Must not allow people to take screenshots or save what others have posted on their status (P37, WhatsApp)*)

There was also a wish for greater knowledge of other users' activities (11%). (*Be able to save people's videos or pictures from the app without the need of installing downloading app (P37, Instagram)*) Some participants even wanted greater knowledge of others and better protection from others. 13% of participants wished for better protection of their location, and 12% of participants wanted less tracking of their activities by companies and better protection from them. (*Ensure no data is being sent to Facebook, I dislike knowing that Whatsapp is owned by Facebook (P84, WhatsApp)*) (*Browse without cookies and no product tracking. (P1, Chrome)*)

3.1.3 Participant Reflections. In the last survey part, we asked participants to reflect on their previous app experiences. Asked what they did in the past when they were unhappy with an app, most said that they did nothing (54%). Many researched how to fix the problem themselves (40%) or complained on social media (30%). Some reached out to app developers either in private (17%) or public (7%). This suggests that many end-users are resigned to the current design of their apps. 85 participants (85%) said that the ability to make changes to their apps would be 'rather' useful for *them*. This is in line with previous research which suggests that users want more control over online services they use, especially as a protection from deceptive designs [18, 53]. Among the remaining 15 participants (answering 'rather no'), 7 expressed that they would not want to make changes that they find sufficiently important (*'I don't really care that much as long as it works'*, P46), while another 7 said that this would not be possible in practice or that they lacked the time and skills (*'I don't have the knowledge for that'*, P12). Sixty-nine participants said they would 'rather' like *everyone* to have the ability to make such changes. Among those that did not want this (answering 'rather no'), the reasons were varied and included concerns around feasibility (5%, *'it would be too hard to programme that'*, P10), security (4%, *'devices are designed to function in certain way'*, P42), and the acts of others (4%, *'they should have the independent right to change the setting on their OWN phone but it should limit to that'*, P65).

4 TECHNICAL FEASIBILITY AND LEGAL ANALYSIS

Our survey underlined that there exist many aspects of apps that individuals potentially would like to change, and that many of these changes might even reduce existing digital harms. Previous work has also identified the importance of tools and methods which would enhance user autonomy to protect users from deceptive design and dark patterns [18, 53]. We saw the responses and needs of our participants as an opportunity to investigate the technical capabilities of user-initiated app repair, as well as associated challenges. Therefore, this section implements a technical prototype – called GreaseDroid – and tries to address a handful of the changes requested by our survey participants. This prototype relies on app *modification* (see Section 2.3), as a means to implement app *repair*.

We study if patching of apps is possible without breaking them, how difficult it is to create patches and the nature of those patches (particularly how many lines of code is needed), and also how patches directly impact the user experience in using patched apps. We also analyse the legality of our approach for research purposes in Section 4.3. Subsequently, Section 5 will outline the remaining risks that need to be addressed concerning ethics, fairness, and security of GreaseDroid patches. We will share the code of our functioning prototype on GitHub.

4.1 Prototype: An App Repair Framework based on App Modification

To implement app *repair*, our tool builds on the concept of app *modification*, which was identified as a fruitful technology from the review of previous work in Section 2.3. Our tool aims to give lay users the ability to change harmful aspects about their apps, and make this process as straightforward as possible similar to Greasemonkey (see Section 2.3). At the core of GreaseDroid, we use the existing `apktool` [29], which decodes Android `apk` files (the standard format for apps on Android) into `smali` code (i.e. low-level code instructions) and other resources (e.g., `xml` files that describe the layout). Our implementation is visualised in Figure 1. We discuss specific examples of patches in Section 4.2, implementing some of the suggestions from the survey. We focus on Android because this platform tends to be more permissive than iOS. To make modifications to apps, users go through three main phases (Figures 1 and 2):

(1) App selection. In the first step, the user selects the app they want to patch by GreaseDroid.

(2) Patch selection. In the second phase, the user selects a set of patches to apply to the chosen app. These patches are developed

by expert community users that we call *patch developers*. This developer community-driven platform approach is inspired by that taken by the Greasemonkey tool, and is also currently a model for browser extensions (see Section 2.3). Once the user has chosen a set of patches, the app gets decoded with `apktool` into `smali` code and other resources, and the patches then get applied to these decoded resources. In our implementation, patches are implemented as bash scripts that describe what modifications to make to a decoded app. Two examples of such scripts are shown in Figure 3 in the Appendix and discussed in our case studies in Section 4.2.

(3) Re-deployment. After successful app patching, we need to reassemble the modified app with `apktool` and re-sign in order to install the modified app on the user's device. Lastly, the user installs the patched app.

4.2 Implementation of Survey Suggestions

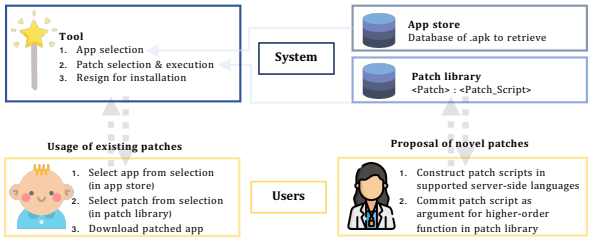
Having explored the design space for app interventions in Section 3 with participants, we proceed to evaluating the development of patches with our prototype tool. We selected these case studies from the survey responses. This selection was guided by our RQ1, which sought 'common, pressing, or important' app changes. We explore how difficult and feasible the patch development is, and also discuss how the patch would change and directly impact the app experience. The code behind some of our patches is shown in Figure 3 in the Appendix.

4.2.1 Distraction and Deceptive Designs in Social Media Apps. Many respondents wished to remove specific interface elements and forced action deceptive designs (41% and 15% of participants, respectively). As highlighted in Section 3, some of these undesired interface elements manifest as deceptive designs as they introduce obstructions or redirection to the original user flow [54], in particular, the 'Stories' functionality of Facebook, Instagram, Snapchat and Twitter.

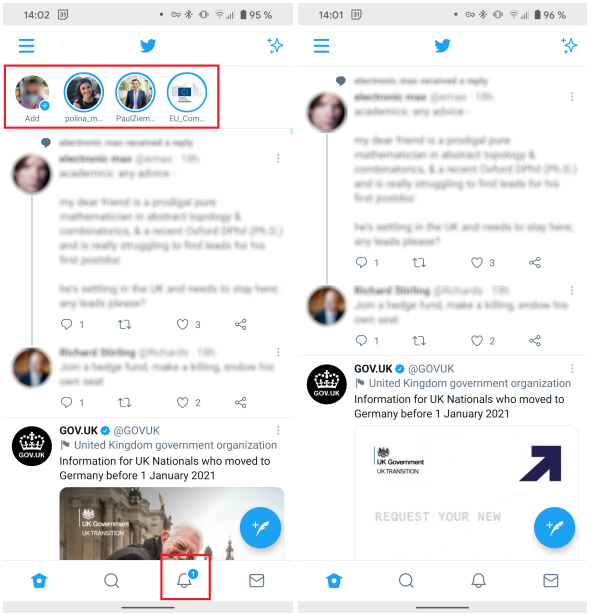
Development of patch: We were able to remove the stories functionality from the Twitter app, and to hide the notification counter which can introduce additional distraction (see Figure 2). We did so by removing the relevant sections from the `xml` layout files from the decoded Twitter app (i.e. from those files that describe the UI on Android). Identifying these sections was straightforward with the run-time app layout inspection tools from the Play Store.

We tried to develop a similar patch for the Facebook app, but found that `apktool` could not operate on the corresponding `xml` files. This is due to Facebook's use of sophisticated obfuscation techniques, and might get addressed in upcoming versions of `apktool` [7]. This observation underlines the potentially tedious cat-and-mouse game if app repair is not officially supported by the smartphone OS. We managed, however, to hide the Facebook feed through `smali` code modifications, thereby allowing to replicate work on digital distraction by Lyngs et al. on mobile [77].

Impact on UX: The proposed changes to the Twitter app might well make the app less distracting. They might also make it less valuable for end-users, being a black-and-white choice (i.e. removal of *all* Stories from the Twitter app).



(a) Overview of the GreaseDroid paradigm.



(b) Original app. (c) Patched app.

Figure 2: GreaseDroid enables the removal of dark patterns (highlighted in red) and other harms in Android apps. Compared to the default Twitter app (middle), stories and notifications have been disabled to reduce distractions in the patched app (right). We discuss the implementation of our prototype and a range of different patches in Section 4 ('Case Studies').

4.2.2 Network Traffic Analysis. Quite a few participants (12%) expressed concerns around data sharing between their apps and third-party companies. To analyse apps' data flows to companies and hold those companies accountable, independent researchers need to install a self-signed certificate on the Android device [88, 99, 109]. Unfortunately, as of Android 7 from 2016, Google has banned this practice, thereby significantly inhibiting app privacy research. This is why we wrote a GreaseDroid patch that allows to apply `apk-mitm` [60], a tool to remove these limitations from `*.apk` files, with one click (rather than having to set up a `Node.js` development environment and running terminal commands).

Development of patch: The development of the patch was straightforward because `apk-mitm` already existed, and also relied on `apktool`.

All we needed to do is run `apk-mitm` on the decoded app resources during the patching process.

Impact on UX: `apk-mitm` makes deep modifications to apps As a result, patched apps sometimes refuse to connect to the internet after patching. However, a simple way to apply `apk-mitm` (rather than having to use a command line) might still have positive benefits for less tech-savvy app privacy researchers.

4.2.3 In-app Advertising. Forty-two percent of respondents wished to limit in-app advertisements. For many apps, there currently exist no means to disable ads, or doing so is expensive (as was commonly mentioned regarding the YouTube Premium subscription). We therefore explored ad blocking with GreaseDroid.

Development of patch: Reducing unwanted ads requires that participants have a good understanding of Android `smali` code, which is an advanced programming skill. There emerged three main strategies to reduce ads within apps: 1) interacting with the built-in methods of ad libraries to manage data collection (e.g. to pass 'no' consent to the Vungle ad library), 2) preventing calls to the methods of ad libraries (e.g. calls to the `loadAd` method of Google Ads), and 3) modifying the API endpoints inside ad libraries, so that network requests fail (e.g. by changing `doubleclick.com` to `localhost`). We successfully tried these strategies on the top 20 Android tracking libraries [45].

Impact on UX: GreaseDroid makes the removal of many types of ads from apps easy for end-users. The negative impacts of these changes on the app experience are likely minimal because apps must already foresee fallback options in case they do not have a stable internet connection and cannot load ads. In rare cases, current apps might refuse to work because ads and app functionality are so tightly intertwined. Courts hold that removing ads per se is legal [49, 89].

4.3 Legality and Right to Repair for Apps?

Using GreaseDroid raises various legal issues and questions. This is why we have consulted various legal experts to help with the legally compliant design of our prototype. While we are confident that the prototyping activities we have undertaken as part of this research project are legitimate, the use of these techniques in non-research settings might have legal repercussions. In particular, copyright laws like the DMCA in the US, the Computer Programs Directive in the EU, and the Copyright, Designs and Patents Act 1988 in the UK. A preliminary analysis of these areas of law raise at least two main issues: 1) the distribution of patched apps and 2) the decompilation and disassembly of apps. Copyright laws typically grant exclusive rights over copying and distribution to the owner; clearly, this impacts the ability for a patch maker to legally distribute a patched app, and similarly in so far as decompilation and disassembly requires copying. However, the decentralised approach of GreaseDroid might help to at least partly overcome such legal challenges, by enabling users themselves to apply patches to their own legitimately obtained copies of apps. While developers have previously shared patched apps online, the model of GreaseDroid described above separates the distribution of patches from the patching of apps. Patches are applied at install-time and on the user's device. There would therefore be no need to distribute patched apps. Such private modification of apps might be covered by the 'fair use' and 'right to repair' principles, because GreaseDroid allows users to remove deficiencies from apps, especially in research settings.

While copyright law worldwide protects right holders' creative works (including programs), it does not protect ideas. UK, EU, and US laws explicitly allow the observation of the ideas behind programs (interoperability clause). Contractual obligations cannot usually override these rights. In this view, researchers who want to study strategies to make apps less addictive are allowed to do so under copyright law, regardless of the contractual obligations imposed by app publishers.

Our implementation of patching in GreaseDroid does not rely on *decompilation* of the program code, but rather on *disassembly*. This could reduce issues with legislation that bans decompilation. Even if decompilation were required, the law in some cases may allow it; under the EU Computer Programs Directive decompilation is permitted when *'necessary for the use of the computer program by the lawful acquirer in accordance with its intended purpose, including for error correction'*. This opens up important, but complex questions about what constitutes the 'intended purpose' of an app and therefore to what ends decompilation (and ultimately app modification) might be permitted.

The legality of app modification is less clear in a non-research setting and requires further analysis. The same holds for an app repair method that does not build upon app modification (as GreaseDroid does), such as an app repair method shipped with the smartphone operating system (e.g. as required by future law).

5 EXPERT INTERVIEWS AND DISCUSSION

These results include the survey-based speculative design exercise of app modifications (in Section 3), our prototype and implementation of modifications from the survey (in Section 4), as well as a legal analysis of our prototype (in Section 4.3). We will first discuss risks and challenges identified from our research in Section 5.2, then benefits and opportunities of the approach in Section 5.1, and lastly the implications of our work for future research and the potential deployment of app repair in practice in Section 5.3.

We additionally complemented the overview of risks and challenges in app repair through semi-structured interviews of about 30 minutes with eight experts in the domains of law, software development, HCI digital distraction and privacy. In these interviews, we asked the experts for their views on the risks and challenges in app repair. Our departmental ethics board approved the study (approval reference CS_C1A_021_021). Table 1 shows the demographics of the experts. The number of interviewees is common for expert studies [11, 19, 100]. The interview script can be found in the Appendix.

5.1 Benefits and opportunities

Increasing participants and benefits for app developers. An ecosystem of patch developers and users is not isolated from the original app developers. Indeed, it may bring benefits for them. The development of patches can serve as a feedback loop for the original developer. With an active network of patch developers and adopters, GreaseDroid might speed up the app developer's development cycle and reduce costs through crowdsourced software

Expert	Field	Industry	Years of Experience
E1	Tech Law	Law	25+
E2	Tech Law	Law	5+
E3	Privacy Law, Dark Patterns	Academia	10+
E4	HCI, Digital Distraction	Academia	5+
E5	UX Design	Freelancing	5+
E6	HCI, Digital Distraction	Academia	5+
E7	Frontend Engineering, Privacy	Academia, Freelancing	5+
E8	Backend Engineering	Freelancing	5+

Table 1: Overview of experts that we consulted for our ethical and legal analysis.

engineering efforts. Indeed, the ability to create patches through GreaseDroid might create new financial incentives, wherein app developers reward patch developers, similar to existing bug bounty programs.

Even if it is not possible to actually implement certain modifications through client-side app repair technology, changing the thinking and expectations around app design can still deliver positive benefits. Many of us currently accept the status quo because we resign or struggle to imagine that different implementations might be possible. These feelings of resignations were also highlighted in our survey, in which a notable number of participants expressed that app repair would not be technically feasible and that they had never approached app developers over concerns around apps. However, our case studies from Section 4.2 showed that, at least in some instances, the suggested changes can actually be implemented and deployed.

Enabling legitimate research into app harms. Research can help overcome app-based harms, by studying their effects. Unfortunately, research in digital distraction has long struggled with studying distraction in mobile apps, and instead tended to focus on desktop [77]. At the same time, users nowadays spend significantly more time with their mobile devices; indeed, many households do not possess a computer anymore. Our case study of removing distracting elements in the Twitter and Facebook apps (see Section 4.2.1) shows how app repair can help such research studies into user autonomy. Other research in smartphone privacy struggles with analysing apps' network traffic ever since Google imposed a de-facto ban on self-signed root certificates in Android in 2016. As we demonstrated in our case studies, GreaseDroid can help researchers overcome such current limitations to conducting research.

Overcoming uniformity in app design. Many participants perceived a lack of customisation options for apps. For example, 39% of participants wanted to change the cosmetics of their apps. This underlines that software nowadays does not contain the wealth of customisation options anymore as it had in the past. Instead, there is usually only one design for all users. If end-users are unhappy with the app design, or might even struggle with it (e.g. due to reading difficulties or other conditions), this can be challenging or even harmful.

Simplicity of patch development. The development of patches that targeted the user interface was relatively straightforward. The relevant `xml` files can be easily understood and modified by those with experience in Android app development. Yet, the modification of `xml` files was currently not possible for those apps that implemented certain obfuscation techniques, e.g. the Facebook app. This might be addressed in future versions of `apktool`, upon which GreaseDroid builds. Meanwhile, making modifications to apps' control flow was significantly more challenging, since this involved working with low-level `smali` code. Reading such low-level code is especially challenging when obfuscation techniques are used. This, too, might be addressed in future work, as app repair functionality becomes more mature.

Limited impact on paid Android apps. Rather than just disabling ads, some might want to use GreaseDroid to unlock paid-for functionality in apps. Yet, getting a paid app for free on Android has long been as easy as seeking a cracked app version on Google, or using LuckyPatcher [75]. This is arguably even in the interest of Google whose whole business model is centred around ads. In 2020 alone, the parent company of Google, Alphabet, generated an estimated $147bn (80%) of its revenue from advertising [5], with more than half coming from mobile devices [40]. As a result, there already exist very few paid apps on Android [1]. An app repair functionality and a wide deployment of GreaseDroid would thus have limited impact on these already existing incentives.

5.2 Risks and challenges

Risk of malicious and unfair patches. A critical challenge for a responsible app repair framework is the threat of malicious patches. Greasemonkey's script market suffered from a significant number of malicious scripts disguised as benign scripts with abusable vulnerabilities [85, 110]. Though there have been recent developments in automated malware detection [21, 62, 67, 74, 105, 118, 121], automated approaches might not be sufficiently reliable to mitigate the threat of malicious patches [22, 23, 35, 57]. To ensure app repair has a positive impact, it would be important to implement robust measures, both to protect users of app repair from malicious acts of others, but also to protect others from the malicious acts of app repair users.

Long-term impact on ad revenue. In Section 4, we discussed how app repair can be used to develop better ad blockers for mobile. We explored such interventions because nearly half our survey participants wished for it. There is, however, a risk that this would cut important revenue streams for app developers. Paid models are often not a feasible alternative, especially on Android [38, 81]. App repair might thus disproportionately affect app developers. This can ultimately have negative effects for end-users, who might end

up with fewer and worse apps as a result of app changes that are unfair and disproportionate towards app developers. Conversely, app repair might also incentivise the development of less distractive and intrusive ads, which survey participants complained about especially, and ultimately more responsible monetisation models. This would be similar to the 'acceptable ads' programme that emerged on desktop browsers in response to ad blockers.

Potential arms race and countermeasures. If app repair was deployed widely, app developers may decide to implement countermeasures. This is what many developers already do in response to ad blockers on the web. Ultimately, this might spark an arms race that will diminish the potential benefits of app repair, and might render this technology useless. This could be a loss for research and other legitimate uses of app repair technology. As we saw in Section 4.2.1, the Facebook app already implements such countermeasures. There have also been reports that WhatsApp has previously banned users of modified app versions in the past [48]. As we saw in Section 4.3, such bans might actually be illegal, given the explicit rights that researchers and other individuals have in studying apps. As we saw from our survey with participants, the majority of participants want to have some sort of control over their apps, especially to reduce deceptive designs, dark patterns and potential harms apps can have on users.

Inadequate usability or accessibility. In our survey, most participants wished for an ability to make modifications to their apps. Some, however, were critical if this would be technically feasible or of how they would be able to accomplish such by themselves. This points to potential problems around usability and accessibility. If an app repair approach cannot be used by those individuals that would benefit from it most, this brings in question the whole concept of app repair and whether it justifies the potential risks.

Technical limitations of app repair. For many of the suggested app changes, it is unlikely that app repair will allow them to be implemented. We explored some suggested modifications in our technical study and found that GreaseDroid currently embraces a black-and-white approach: either removing all ads or none, either removing all Stories from the Twitter app or none. Some of this might be fixed through better patches and immense engineering efforts. Creating more user-friendly ads and social media recommendations will, however, mostly lie with the companies behind these technologies. GreaseDroid, however, might help in communicating users' wishes and setting incentives, and thereby increase pressure against harmful practices within apps. Another challenge relates to patch compatibility with app updates [36], which get released regularly and might break some patches.

Legality for non-research purposes. As we showed in Section 4.3, the use of app repair for research purposes is likely unproblematic. Other uses, however, remain in a legal grey area and need further study. Interestingly, if app repair functionality was foreseen by the operating system, then our legal analysis would likely be different. Developers would then be forced to foresee extensibility. Such would arguably then be within the *intended purpose* (in the sense of the EU Computer Programs Directive) of apps and app repair be on a more sound legal foundation.

5.3 Future work: Towards responsible app repair

From our findings, it is clear that any large-scale practical deployment of app repair technology would need to address challenges in three core areas: social, legal, and technical. In the following, we introduce a set of guiding principles and measures for each of these areas. These measures may not be sufficient to deploy an app repair technology in a responsible manner. Based upon our research, we do think, however, that they are a necessary minimum set for such responsible deployment.

Social. As highlighted by this and previous work, the development and deployment of app repair technology is not without risks. This is why it is important to have a thorough screening process in place, as is widely adopted in software marketplaces, such as the Apple App Store or the Chrome browser extensions store. This screening process of the specific examples of app repair could be inspired by the open-source approach of F-Droid. This app store for Android has in the past proved incredibly secure. There are no known reports of large-scale security problems in the past, to the best of our knowledge. At the heart of this app store lie principles of openness, transparency and community. The same principles should be considered in deploying app repair responsibly. Motivated by this insight, we developed an example *code of conduct* and ethical declaration for patch developers, based on our research, in Figure 7 in the Appendix.

Legal. As discussed in Section 4.3, our implementation of app repair, relying on app modification, is likely fine for research purposes, but is in a more uncertain legal area in other settings. The reason for this is not that app repair is illegal per se, but rather that there exists limited legal precedent. Different technical implementations may not face similar problems, especially if app repair was directly integrated into the smartphone operating system – similar to how extensions are part of most modern browsers. A code of conduct, as suggested in the previous section, could provide further legal certainty by making patch developers give basic guarantees as to the safety and fairness of their patches.

Technical. The technical design highly depends on the assessment of requirements relating to the previous two aspects, legal and social. Our prototype from Section 4 could serve as a sample implementation. However, patching should be ideally done directly on a user's smartphone and not rely on an external server. Due to our Linux-based approach, it might be possible to port our system in the future to Android, which itself runs the Linux kernel. The current system also relies on `apktool`, which is compute-intensive and should be addressed. At the same time, a certain level of friction in the system might be positive to prevent harm. A future system should further foresee an option to turn patches on and off easily. To foster transparency around patches (as highlighted in 'social'), the technical implementation should foresee measures to allow end-users to study the code of patches, and to flag concerns with patches. As highlighted by our set of case studies in Section 4, many promising patches actually only require a few lines of code. Those should be easy to inspect by the user's of patches. If unusually many lines of code are required, more thorough screening methods would need to be implemented.

6 LIMITATIONS

There are several limitations to this work. We had a limited sample size and only considered a non-representative population aged 18–40, participating through Prolific Academic. We had a limited sample size due to the qualitative nature of our survey [59], and note that participants from online crowdsourcing studies tend to be younger than the general population [64, 96]. The case studies do not cover all possible app modifications, only a set of commonly mentioned and important modifications.

7 CONCLUSIONS

This work explored the risks and opportunities of an ability to repair apps for end-users. After a review of the relevant background in Section 2, we first surveyed 100 participants to investigate what aspects of their regularly used apps they would like to change in Section 3. Eighty-five percent of participants said that the ability to make changes to their apps would be helpful for them, but they were suspicious about malicious use of such functionality by others. Using a prototype for an app repair framework (based on app modification) and a set of case studies derived from the survey, we then (in Section 4) illustrated the patch development process, demonstrated the feasibility of this method, and discussed the potential direct impact of such a technology on the app experience. We also analysed legal challenges around app repair in Section 4.3. Finally, in Section 5, we conducted eight expert interviews, synthesised a framework of risks and opportunities of app repair, and put forward recommendations for the responsible deployment of app repair, including a code of conduct and ethical declaration for patch developers (see Figure 7 in the Appendix).

Our work underlines that app repair is no silver bullet against app-based harms. Many app changes, that individuals wish for, cannot be implemented through solely client-side changes. Some changes might even pose harm to others. There are, however, also valuable promises, including reduced harms and increased user participation in the design of mobile apps – besides enabling a whole wealth of new research into mobile apps. In other words, the current non-deployment of app repair technology, too, causes harm and involves many unethical aspects. Given the currently significant imbalances between users and platforms/developers, we conclude, based upon our analysis, that a responsibly deployed app repair framework would pose relatively small harm and would deliver tangible benefits to individuals. We anticipate that app repair should rather be implemented by the manufacturers behind smartphones, rather than retrofitting existing systems through approaches like app modification (as done in this work), for example through an explicit legal right to repair apps.

ACKNOWLEDGMENTS

We thank our study participants, both those in the survey as well as those in our expert interviews. We also thank Martin J Kraemer, Lin Kyi, Ge Wang, Helena Webb, and Claudine Tinsman. Konrad Kollnig was funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/R513295/1. Ulrik Lyngs was supported by a Carlsberg Foundation Oxford Visiting Fellowship under grant number CF20-0678. Konrad Kollnig, Reuben Binns, Max Van Kleek, and Nigel Shadbolt have been supported by the Oxford Martin School EWADA Programme.

REFERENCES

[1] Statista & 42matters. 2022. Distribution of free and paid apps in the Apple App Store and Google Play. https://www.statista.com/statistics/263797/number-of-applications-for-mobile-phones/
[2] AdGuard. [n.d.]. AdGuard. https://adguard.com/
[3] Yuvraj Agarwal and Malcolm Hall. 2013. ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '13*. ACM Press, Taipei, Taiwan, 97. https://doi.org/10.1145/2462456.2464460
[4] Agriland. 2022. John Deere to face lawsuits over right to repair. https://www.agriland.co.uk/farming-news/john-deere-to-face-lawsuits-over-right-to-repair/
[5] Alphabet. 2020. FORM 10-K. https://abc.xyz/investor/static/pdf/20210203_alphabet_10K.pdf?cache=b44182d
[6] Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. 2020. Accessibility Issues in Android Apps: State of Affairs, Sentiments, and Ways Forward. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 1323–1334. https://doi.org/10.1145/3377811.3380392
[7] apktool: GitHub Issue Tracker. [n.d.]. decompiling latest Facebook APK. https://github.com/iBotPeaches/Apktool/issues/1719
[8] Jai Asundi and Rajiv Jayant. 2007. Patch Review Processes in Open Source Software Development Communities: A Comparative Case Study. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. 166c–166c. https://doi.org/10.1109/HICSS.2007.426
[9] Austrian Data Protection Authority. 2021. Partial Decision D155.027, 2021-0.586.257. https://noyb.eu/sites/default/files/2022-01/E-DSB%20-%20Google%20Analytics_EN_bk.pdf
[10] Michael Backes, Sebastian Gerling, Christian Hammer, Matteo Maffei, and Philipp von Styp-Rekowsky. 2014. AppGuard – Fine-Grained Policy Enforcement for Untrusted Android Applications. In *Data Privacy Management and Autonomous Spontaneous Security*, Joaquin Garcia-Alfaro, Georgios Lioudakis, Nora Cuppens-Boulahia, Simon Foley, and William M. Fitzgerald (Eds.). Lecture Notes in Computer Science, Vol. 8247. Springer Berlin Heidelberg, Berlin, Heidelberg, 213–231. https://doi.org/10.1007/978-3-642-54568-9_14
[11] Kathrin Bednar, Sarah Spiekermann, and Marc Langheinrich. 2019. Engineering Privacy by Design: Are engineers ready to live up to the challenge? *Information Society* 35, 3 (5 2019), 122–142. https://doi.org/10.1080/01972243.2019.1583296
[12] Ruha Benjamin. 2019. *Race after Technology: Abolitionist Tools for the New Jim Code*. Polity, Cambridge, United Kingdom.
[13] Yochai Benkler. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.
[14] Jill L. Bezyak, Scott A. Sabella, and Robert H. Gattis. 2017. Public Transportation: An Investigation of Barriers for People With Disabilities. *Journal of Disability Policy Studies* 28, 1 (2017), 52–60. https://doi.org/10.1177/1044207317702070 arXiv:https://doi.org/10.1177/1044207317702070
[15] Reuben Binns and Elettra Bietti. 2020. Dissolving privacy, one merger at a time: Competition, data and third party tracking. *Computer Law & Security Review* 36 (2020), 105369. https://doi.org/10.1016/j.clsr.2019.105369
[16] Reuben Binns and Reuben Kirkham. 2021. How Could Equality and Data Protection Law Shape AI Fairness for People with Disabilities? (2021), 37.
[17] Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt. 2018. Third Party Tracking in the Mobile Ecosystem. In *Proceedings of the 10th ACM Conference on Web Science - WebSci '18* (Amsterdam, Netherlands). ACM Press, New York, NY, United States, 23–31. https://doi.org/10.1145/3201064.3201089
[18] Kerstin Bongard-Blanchy, Arianna Rossi, Salvador Rivas, Sophie Doublet, Vincent Koenig, and Gabriele Lenzini. 2021. " I am Definitely Manipulated, Even When I am Aware of it. It's Ridiculous!"-Dark Patterns from the End-User Perspective. In *Designing Interactive Systems Conference 2021*. 763–776.
[19] Rym Boulkedid, Hendy Abdoul, Marine Loustau, Olivier Sibony, and Corinne Alberti. 2011. Using and Reporting the Delphi Method for Selecting Healthcare Quality Indicators: A Systematic Review. *PloS One* 6, 6 (2011), e20476. https://doi.org/10.1371/journal.pone.0020476
[20] Bundeskartellamt. 2019. B6-22/16 (Facebook v Bundeskartellamt).
[21] Lingwei Chen, Shifu Hou, and Yanfang Ye. 2017. SecureDroid: Enhancing Security of Machine Learning-Based Detection against Adversarial Android Malware Attacks. In *Proceedings of the 33rd Annual Computer Security Applications Conference* (Orlando, FL, USA) *(ACSAC 2017)*. Association for Computing Machinery, New York, NY, USA, 362–372. https://doi.org/10.1145/3134600.3134636
[22] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. 2018. Automated poisoning attacks and defenses in malware detection

systems: An adversarial machine learning approach. *Computers & Security* 73 (2018), 326 – 344. https://doi.org/10.1016/j.cose.2017.11.007

[23] Xiao Chen, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren. 2020. Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection. *IEEE Transactions on Information Forensics and Security* 15 (2020), 987–1001. https://doi.org/10.1109/TIFS.2019.2932228

[24] Commission Nationale de l'Informatique et des Libertés. 2018. Décision n° MED 2018-042 du 30 octobre 2018 mettant en demeure la société X. https://www.legifrance.gouv.fr/cnil/id/CNILTEXT000037594451/

[25] Commission Nationale de l'Informatique et des Libertés. 2019. Délibération SAN-2019-001 du 21 janvier 2019. https://www.legifrance.gouv.fr/cnil/id/CNILTEXT000038032552/

[26] Commission Nationale de l'Informatique et des Libertés. 2020. Délibération SAN-2020-012 du 7 décembre 2020. https://www.legifrance.gouv.fr/cnil/id/CNILTEXT000042635706

[27] Commission Nationale de l'Informatique et des Libertés. 2022. Use of Google Analytics and data transfers to the United States: the CNIL orders a website manager/operator to comply. https://www.cnil.fr/en/use-google-analytics-and-data-transfers-united-states-cnil-orders-website-manageroperator-comply

[28] Competition and Markets Authority. 2020. Online Platforms and Digital Advertising.

[29] Connor Tumbleson. [n.d.]. Apktool. https://ibotpeaches.github.io/Apktool/

[30] Court of Justice of the European Union. 2020. Data Protection Commissioner v Facebook Ireland and Maximillian Schrems. https://curia.europa.eu/juris/documents.jsf?num=C-311/18

[31] Siddhartha Datta, Konrad Kollnig, and Nigel Shadbolt. 2021. Mind-proofing Your Phone: Navigating the Digital Minefield with GreaseTerminator. *CoRR* abs/2112.10699 (2021), 22 pages. arXiv:2112.10699 https://arxiv.org/abs/2112.10699

[32] Siddhartha Datta, Konrad Kollnig, and Nigel Shadbolt. 2022. GreaseVision: Rewriting the Rules of the Interface. https://doi.org/10.48550/ARXIV.2204.03731

[33] Benjamin Davis and Hao Chen. 2013. RetroSkeleton: Retrofitting Android Apps. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '13*. ACM Press, Taipei, Taiwan, 181. https://doi.org/10.1145/2462456.2464462

[34] Benjamin Davis, Ben S, Armen Khodaverdian, and Hao Chen. 2012. I-ARM-Droid: A rewriting framework for in-app reference monitors for android applications. In *In Proceedings of the Mobile Security Technologies 2012, MOST '12*. IEEE, New York, NY, United States, 1–9.

[35] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. 2019. Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection. *IEEE Trans. Dependable Secur. Comput.* 16, 4 (July 2019), 711–724. https://doi.org/10.1109/TDSC.2017.2700270

[36] Oscar Díaz, Cristóbal Arellano, and Jon Iturrioz. 2010. Interfaces for Scripting: Making Greasemonkey Scripts Resilient to Website Upgrades. In *Web Engineering*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Boualem Benatallah, Fabio Casati, Gerti Kappel, and Gustavo Rossi (Eds.). Vol. 6189. Springer Berlin Heidelberg, Berlin, Heidelberg, 233–247. https://doi.org/10.1007/978-3-642-13911-6_16

[37] Cory Doctorow. 2021. Competitive compatibility: let's fix the internet, not the tech giants. *Commun. ACM* 64, 10 (2021), 26–29.

[38] Anirudh Ekambaranathan, Jun Zhao, and Max Van Kleek. 2021. "Money makes the world go around": Identifying Barriers to Better Privacy in Children's Apps From Developers' Perspectives. In *Conference on Human Factors in Computing Systems (CHI '21)* (Yokohama, Japan, 2021). ACM Press, 1–24. https://doi.org/10.1145/3411764.3445599

[39] Marcelo Medeiros Eler, Jose Miguel Rojas, Yan Ge, and Gordon Fraser. 2018. Automated Accessibility Testing of Mobile Apps. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. 116–126. https://doi.org/10.1109/ICST.2018.00021

[40] eMarketer. 2016. Mobile Moves to Majority Share of Google's Worldwide Ad Revenues. https://www.emarketer.com/Article/Mobile-Moves-Majority-Share-of-Googles-Worldwide-Ad-Revenues/1014633

[41] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. 2010. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10)*. USENIX Association, Berkeley, CA, United States, 393–407.

[42] Daniel A. Epstein, Nicole B. Lee, Jennifer H. Kang, Elena Agapie, Jessica Schroeder, Laura R. Pina, James Fogarty, Julie A. Kientz, and Sean Munson. 2017. Examining Menstrual Tracking to Inform the Design of Personal Informatics Tools. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 6876–6888. https://doi.org/10.1145/3025453.3025635

[43] European Data Protection Supervisor. 2022. Decision of the European Data Protection Supervisor in complaint case 2020-1013 submitted by Members of the Parliament against the European Parliament. https://noyb.eu/sites/default/files/2022-01/Case%202020-1013%20-%20EDPS%20Decision_bk.pdf

[44] evilwombat. [n.d.]. A de-bullshified version of Facebook (less ads, less clutter, less crap). https://forum.xda-developers.com/t/a-de-bullshified-version-of-facebook-less-ads-less-clutter-less-crap.3586318/

[45] Exodus. [n.d.]. Reports. https://reports.exodus-privacy.eu.org/en/reports/

[46] flxapps. [n.d.]. DetoxDroid. https://github.com/flxapps/DetoxDroid

[47] Jay Freeman. [n.d.]. Cydia Substrate. http://www.cydiasubstrate.com/

[48] Gadgets Now. 2022. 10 things you should not do to avoid getting banned on WhatsApp. https://www.gadgetsnow.com/slideshows/10-things-that-may-land-you-in-trouble-on-whatsapp/photolist/89733017.cms

[49] German Federal Court of Justice. 2018. Ruling in case I ZR 154/16. https://juris.bundesgerichtshof.de/cgi-bin/rechtsprechung/document.py?Gericht=bgh&Art=en&Datum=Aktuell&anz=1&pos=0&nr=82856&linked=pm&Blank=1.

[50] Kovacs Geza. 2019. HabitLab: In-The-Wild Behavior Change Experiments at Scale. *Stanford Department of Computer Science* (2019). https://stacks.stanford.edu/file/druid:qq438qv1791/Thesis-augmented.pdf

[51] Vegard IT GmbH. [n.d.]. Gray-Switch. https://play.google.com/store/apps/details?id=com.vegardit.grayswitch

[52] Google. [n.d.]. Android Accessibility Suite. https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback

[53] Colin M Gray, Jingle Chen, Shruthi Sai Chivukula, and Liyang Qu. 2021. End user accounts of dark patterns as felt manipulation. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–25.

[54] Colin M. Gray, Yubo Kou, Bryan Battles, Joseph Hoggatt, and Austin L. Toombs. 2018. The Dark (Patterns) Side of UX Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3174108

[55] Peter Gregor, Alan F. Newell, and Mary Zajicek. 2002. Designing for Dynamic Diversity: Interfaces for Older People. In *Proceedings of the Fifth International ACM Conference on Assistive Technologies* (Edinburgh, Scotland) *(Assets '02)*. Association for Computing Machinery, New York, NY, USA, 151–156. https://doi.org/10.1145/638249.638277

[56] James Grimmelmann. 2022. Spyware vs. Spyware: Software Conflicts and User Autonomy. *Ohio State Technology Law Journal* 16 (2022), 43.

[57] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *Computer Security – ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Proceedings (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics))*, Simon N. Foley, Dieter Gollmann, and Einar Snekkenes (Eds.). Springer Verlag, Germany, 62–79. https://doi.org/10.1007/978-3-319-66399-9_4

[58] Vicki L. Hanson. 2010. Influencing technology adoption by older adults. *Interacting with Computers* 22, 6 (09 2010), 502–509. https://doi.org/10.1016/j.intcom.2010.09.001 arXiv:https://academic.oup.com/iwc/article-pdf/22/6/502/2241774/iwc22-0502.pdf

[59] Monique Hennink and Bonnie N Kaiser. 2021. Sample sizes for saturation in qualitative research: A systematic review of empirical tests. *Social Science & Medicine* (2021), 114523.

[60] Niklas Higi. [n.d.]. apk-mitm. https://github.com/shroudedcode/apk-mitm/

[61] Government HM. 2019. Online Harms White Paper. *Government Report on Transparency Reporting* (2019). https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/793360/Online_Harms_White_Paper.pdf

[62] Shifu Hou, Aaron Saas, Lifei Chen, and Yanfang Ye. 2016. Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*. IEEE, Omaha, NE, USA, 104–111. https://doi.org/10.1109/WIW.2016.040

[63] Jack Hughes, Ben Collier, and Alice Hutchings. 2019. From Playing Games to Committing Crimes: A Multi-Technique Approach to Predicting Key Actors on an Online Gaming Forum. In *2019 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, Pittsburgh, PA, USA, 1–12. https://doi.org/10.1109/eCrime47957.2019.9037586

[64] Panagiotis G Ipeirotis. 2010. Demographics of Mechanical Turk. (2010).

[65] Jinseong Jeon, Kristopher K. Micinski, Jeffrey A. Vaughan, Ari Fogel, Nikhilesh Reddy, Jeffrey S. Foster, and Todd Millstein. 2012. Dr. Android and Mr. Hide: Fine-Grained Permissions in Android Applications. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM '12*. ACM Press, Raleigh, North Carolina, USA, 3. https://doi.org/10.1145/2381934.2381938

[66] P. John Clarkson and Roger Coleman. 2015. History of Inclusive Design in the UK. *Applied Ergonomics* 46 (2015), 235–247. https://doi.org/10.1016/j.apergo.2013.03.002 Special Issue: Inclusive Design.

[67] ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, and Djedjiga Mouheb. 2018. MalDozer: Automatic framework for android malware detection using deep learning. *Digital Investigation* 24 (2018), S48 – S59. https://doi.org/10.1016/j.diin.2018.01.007

[68] Panicos Karkallis, Jorge Blasco, Guillermo Suarez-Tangil, and Sergio Pastrana. 2021. Detecting Video-Game Injectors Exchanged in Game Cheating Communities. In *Computer Security – ESORICS 2021*, Elisa Bertino, Haya Shulman, and Michael Waidner (Eds.). Vol. 12972. Springer International Publishing, Cham, 305–324. https://doi.org/10.1007/978-3-030-88418-5_15

[69] Konrad Kollnig, Reuben Binns, Max Van Kleek, Ulrik Lyngs, Jun Zhao, Claudine Tinsman, and Nigel Shadbolt. 2021. Before and after GDPR: Tracking in Mobile Apps. 10, 4 (2021). https://doi.org/10.14763/2021.4.1611

[70] Konrad Kollnig and Nigel Shadbolt. 2022. TrackerControl: Transparency and Choice around App Tracking. *Journal of Open Source Software* 7, 75 (2022), 4270. https://doi.org/10.21105/joss.04270

[71] Anastasia Kozyreva, Sam Wineburg, Stephan Lewandowsky, and Ralph Hertwig. 0. Critical Ignoring as a Core Competence for Digital Citizens. *Current Directions in Psychological Science* 0, 0 (0), 09637214221121570. https://doi.org/10.1177/09637214221121570 arXiv:https://doi.org/10.1177/09637214221121570

[72] Priya Lalvani. 2015. Disability, Stigma and Otherness: Perspectives of Parents and Teachers. *International Journal of Disability, Development and Education* 62, 4 (2015), 379–393. https://doi.org/10.1080/1034912X.2015.1029877 arXiv:https://doi.org/10.1080/1034912X.2015.1029877

[73] Lawence Lessig. 2006. *Code 2.0* (1 ed.). Basic Books.

[74] Hongliang Liang, Yan Song, and Da Xiao. 2017. An end-to-end model for Android malware detection. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, Beijing, 140–142. https://doi.org/10.1109/ISI.2017.8004891

[75] LuckyPatcher. [n.d.]. Lucky Patcher. https://www.luckypatchers.com/

[76] Ulrik Lyngs, Kai Lukoff, Petr Slovak, Reuben Binns, Adam Slack, Michael Inzlicht, Max Van Kleek, and Nigel Shadbolt. 2019. Self-Control in Cyberspace: Applying Dual Systems Theory to a Review of Digital Self-Control Tools. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, United States, 1–18. https://doi.org/10.1145/3290605.3300361

[77] Ulrik Lyngs, Kai Lukoff, Petr Slovak, William Seymour, Helena Webb, Marina Jirotka, Jun Zhao, Max Van Kleek, and Nigel Shadbolt. 2020. 'I Just Want to Hack Myself to Not Get Distracted': Evaluating Design Interventions for Self-Control on Facebook. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–15. https://doi.org/10.1145/3313831.3376672

[78] Orla Lynskey. 2019. Grappling with "Data Power": Normative Nudges from Data Protection and Privacy. *Theoretical Inquiries in Law* 20, 1 (2019), 189–220. https://doi.org/10.1515/til-2019-0007

[79] MacRumours. [n.d.]. Apple Fails in Bid to Dismiss Cydia Creator's Amended Antitrust Lawsuit. https://www.macrumors.com/2022/05/30/judge-allows-cydia-lawsuit-against-apple/

[80] Nick Merrill. 2020. Security Fictions: Bridging Speculative Design and Computer Security. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*. ACM, Eindhoven Netherlands, 1727–1735. https://doi.org/10.1145/3357236.3395451

[81] Abraham H Mhaidli, Yixin Zou, and Florian Schaub. 2019. "We Can't Live Without Them!" App Developers' Adoption of Ad Networks and Their Considerations of Consumer Risks. *Proceedings of the Fifteenth Symposium on Usable Privacy and Security* (2019), 21.

[82] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. 2002. Evolution Patterns of Open-Source Software Systems and Communities. In *Proceedings of the International Workshop on Principles of Software Evolution* (Orlando, Florida) *(IWPSE '02)*. Association for Computing Machinery, New York, NY, USA, 76–85. https://doi.org/10.1145/512035.512055

[83] Midas Nouwens, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal. 2020. Dark Patterns after the GDPR: Scraping Consent Pop-ups and Demonstrating their Influence. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020). https://doi.org/10.1145/3313831.3376321 arXiv:2001.02479

[84] Aaron Perzanowski. 2022. *The Right to Repair: Reclaiming Control over the Things We Own*. Cambridge University Press, Cambridge, United Kingdom; New York, NY.

[85] Pablo Picazo-Sanchez, Lara Ortiz-Martin, Gerardo Schneider, and Andrei Sabelfeld. 2022. Are Chrome Extensions Compliant with the Spirit of Least Privilege? *International Journal of Information Security* (Sept. 2022). https://doi.org/10.1007/s10207-022-00610-w

[86] Mark Pilgrim. 2005. *Greasemonkey Hacks*. O'Reilly, Sebastopol, Calif.

[87] Santiago Pontiroli. 2019. The Cake Is a Lie! Uncovering the Secret World of Malware-like Cheats in Video Games. (2019), 10.

[88] Abbas Razaghpanah, Arian Akhavan Niaki, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Johanna Amann, and Phillipa Gill. 2017. Studying TLS Usage in Android Apps. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies* (Incheon, Republic of Korea) *(CoNEXT '17)*. Association for Computing Machinery, New York, NY, USA, 350–362. https://doi.org/10.1145/3143361.3143400

[89] Regional Court of Hamburg. 2022. Ruling in case 308 O 130/19. https://dejure.org/dienste/vernetzung/rechtsprechung?Gericht=LG%20Hamburg&Datum=14.01.2022&Aktenzeichen=308%20O%20130%2F19.

[90] Sebastian Rieger and Caroline Sinders. 2020. Dark Patterns: Regulating Digital Design. https://www.stiftung-nv.de/sites/default/files/dark.patterns.english.pdf

[91] Yvonne Rogers, Paul Dourish, Patrick Olivier, Margot Brereton, and Jodi Forlizzi. 2020. The Dark Side of Interaction Design. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3334480.3381070

[92] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2020. An Epidemiology-Inspired Large-Scale Analysis of Android App Accessibility. *ACM Trans. Access. Comput.* 13, 1, Article 4 (apr 2020), 36 pages. https://doi.org/10.1145/3348797

[93] rovo89. [n.d.]. Xposed Framework. https://xposed.info/

[94] Pamela Samuelson. 2016. Freedom to Tinker. *Theoretical Inquiries in Law* 17, 2 (July 2016).

[95] SensePost. [n.d.]. objection. https://github.com/sensepost/objection

[96] Eunjin Seong and Seungjun Kim. 2020. Designing a crowdsourcing system for the elderly: A gamified approach to speech collection. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–9.

[97] Olivier Serrat. 2017. *The Five Whys Technique*. Springer Singapore, Singapore, 307–310. https://doi.org/10.1007/978-981-10-0983-9_32

[98] Irina Shklovski, Scott D. Mainwaring, Halla Hrund Skúladóttir, and Höskuldur Borgthorsson. 2014. Leakiness and Creepiness in App Space: Perceptions of Privacy and Mobile App Use. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI '14* (Toronto, Ontario, Canada). ACM Press, New York, NY, United States, 2347–2356. https://doi.org/10.1145/2556288.2557421

[99] Anastasia Shuba and Athina Markopoulou. 2020. NoMoATS: Towards Automatic Detection of Mobile Tracking. *Proceedings on Privacy Enhancing Technologies* 2020, 2 (2020), 45–66. https://doi.org/10.2478/popets-2020-0017

[100] Sean Sirur, Jason R.C. Nurse, and Helena Webb. 2018. Are we there yet? Understanding the challenges faced in complying with the General Data Protection Regulation (GDPR). *Proceedings of the ACM Conference on Computer and Communications Security* (10 2018), 88–95. https://doi.org/10.1145/3267357.3267368

[101] Adam Smith. 1776. *An Inquiry into the Nature and Causes of the Wealth of Nations*.

[102] Katta Spiel, Fares Kayali, Louise Horvath, Michael Penkler, Sabine Harrer, Miguel Sicart, and Jessica Hammer. 2018. Fitter, Happier, More Productive? The Normative Ontology of Fitness Trackers. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI EA '18)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3170427.3188401

[103] Statista. 2021. Ad blocking user penetration rate in the United States from 2014 to 2021. https://www.statista.com/statistics/804008/ad-blocking-reach-usage-us/

[104] Statista / Vorhaus Advisors. 2022. Time spent using smartphone. https://www.statista.com/statistics/1310218/time-spent-using-smartphone-age-us/

[105] Xin Su, Dafang Zhang, Wenjia Li, and Kai Zhao. 2016. A Deep Learning Approach to Android Malware Feature Learning and Detection. In *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, Tianjin, China, 244–251. https://doi.org/10.1109/TrustCom.2016.0070

[106] Supreme Court of the United States. 2020. Google LLC v. Oracle America, Inc. https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf

[107] Aurelia Tamò-Larrieux, Zaira Zihlmann, Kimberly Garcia, and Simon Mayer. 2021. The Right to Customization: Conceptualizing the Right to Repair for Informational Privacy. In *Privacy Technologies and Policy*, Nils Gruschka, Luís Filipe Coelho Antunes, Kai Rannenberg, and Prokopios Drogkaris (Eds.). Vol. 12703. Springer International Publishing, Cham, 3–22. https://doi.org/10.1007/978-3-030-76663-4_1

[108] Thomas Thwaites. 2011. *The toaster project: Or a heroic attempt to build a simple electric appliance from scratch*. Chronicle Books.

[109] Marcos Tileria and Jorge Blasco. 2022. Watch Over Your TV: A Security and Privacy Analysis of the Android TV Ecosystem. *Proceedings on Privacy Enhancing Technologies* 2022, 3 (July 2022), 692–710. https://doi.org/10.56553/popets-2022-0092

[110] Steven Van Acker, Nick Nikiforakis, Lieven Desmet, Frank Piessens, and Wouter Joosen. 2014. Monkey-in-the-Browser: Malware and Vulnerabilities in Augmented Browsing Script Markets. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security* (Kyoto, Japan) *(ASIA CCS '14)*. Association for Computing Machinery, New York, NY, USA, 525–530. https://doi.org/10.1145/2590296.2590311

[111] Max Van Kleek, Ilaria Liccardi, Reuben Binns, Jun Zhao, Daniel J. Weitzner, and Nigel Shadbolt. 2017. Better the Devil You Know: Exposing the Data Sharing

Practices of Smartphone Apps. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17 (CHI '17)*. ACM Press, Denver, Colorado, USA, 5208–5220. https://doi.org/10.1145/3025453.3025556

[112] Vanced. [n.d.]. Vanced. https://vancedapp.com/

[113] Christopher Vendome, Diana Solano, Santiago Liñán, and Mario Linares-Vásquez. 2019. Can Everyone use my app? An Empirical Study on Accessibility in Android Apps. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 41–52. https://doi.org/10.1109/ICSME.2019.00014

[114] S Wineburg. 2021. To navigate the dangers of the web, you need critical thinking–But also critical ignoring. *The Conversation* (2021).

[115] Michael H Wolk. 2010. The iPhone Jailbreaking Exemption and the Issue of Openness. *Cornell Journal of Law and Public Policy:* 19, 3 (2010), 35.

[116] Shaomei Wu, Lindsay Reynolds, Xian Li, and Francisco Guzmán. 2019. Design and Evaluation of a Social Media Writing Support Tool for People with Dyslexia. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–14. https://doi.org/10.1145/3290605.3300746

[117] Tim Wu. 2018. *The Curse of Bigness: Antitrust in the New Gilded Age*. Columbia Global Reports. OCLC: on1029205194.

[118] K. Xu, Yingjiu Li, R. Deng, and K. Chen. 2018. DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks. *2018 IEEE European Symposium on Security and Privacy (EuroS&P)* 1, 1 (2018), 473–487. https://doi.org/10.1109/EuroSP.2018.00040

[119] Rubin Xu, Hassen Saidi, and Ross Anderson. 2012. Aurasium: Practical Policy Enforcement for Android Applications. In *21st USENIX Security Symposium (USENIX Security 12)*. USENIX Association, Bellevue, WA, 539–552. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/xu_rubin

[120] Shunguo Yan and P. G. Ramachandran. 2019. The Current Status of Accessibility in Mobile Apps. *ACM Trans. Access. Comput.* 12, 1, Article 3 (feb 2019), 31 pages. https://doi.org/10.1145/3300176

[121] Wei Yang, Deguang Kong, Tao Xie, and Carl A. Gunter. 2017. Malware Detection in Adversarial Settings: Exploiting Feature Evolutions and Confusions in Android Apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference* (Orlando, FL, USA) *(ACSAC 2017)*. Association for Computing Machinery, New York, NY, USA, 288–302. https://doi.org/10.1145/3134600.3134642

[122] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel Reidenberg, N Cameron Russell, and Norman Sadeh. 2019. MAPS: Scaling Privacy Compliance Analysis to a Million Apps. *Privacy Enhancing Technologies Symposium 2019* 72, 3 (6 2019), 66–86. https://doi.org/10.2478/popets-2019-0037

patch: Remove location

properties: app-agnostic, control flow

```
perl -i -p0e 's/
    [^\r\n]+Landroid\/location\/Location;
    >(getLongitude|getLatitude)\(\)D[\r\n]+
    move-result-wide v([0-9]+)/"
    const-wide\/16 v$2, 0x0"/seg' $f
```

patch: Remove stories bar

properties: app-specific, interface

```
-<FrameLayout android:background="?
    coreColorAppBackground" android: layout_width="
    fill_parent" android:layout_height="wrap_content"
+<FrameLayout android:background="?
    coreColorAppBackground" android:visibility="gone"
    android:layout_width="fill_parent" android:
    layout_height="wrap_content"
```

Figure 3: In our case studies, we explored app-agnostic patches (i.e. those that apply across apps) and app-specific patches (i.e. those that apply to one app only). Left: An *app-agnostic* patch, a Perl script to prevent apps from accessing the longitude and latitude of the current physical user location. Right: An *app-specific* patch, a diff script to remove the *Stories* bar from the Twitter app.

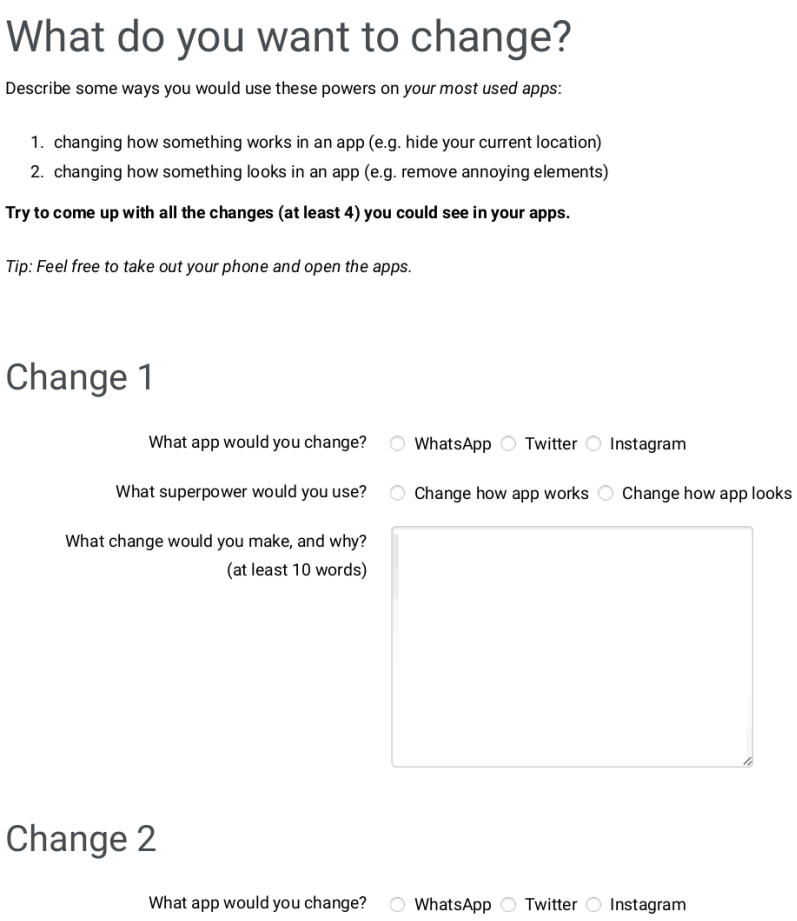


Figure 4: Screenshot from survey: We repeatedly asked participants to suggest changes to those apps that they regularly use. This repetition aimed to stimulate participants' creativity.

Figure 5: A survey to ask participants about aspects that they would like to change about their apps.

(1) Demographics:
  (a) What is your gender? (Woman / Man / Non-binary / Prefer not to answer / Prefer to self-describe)
  (b) What type of smartphone do you currently use? (Android / iPhone / No smartphone)
  (c) In a typical week, what three apps do you use most on your phone?
  (d) In the past week, on average, how much time PER DAY have you spent using your smartphone? (make your best guess) (Less than 10 minutes per day / 10–30 minutes per day / 31–60 minutes per day / 1–2 hours per day / 2–3 hours per day / 3–4 hours per day / 4–5 hours per day / 5–6 hours per day / 6—7 hours per day / 7–8 hours per day / 8–9 hours per day / 9–10 hours per day / More than 10 hours per day)
  (e) Do you have any long-term health conditions or illnesses that make using your smartphone more difficult? (Yes / No / Prefer not to answer)
(2) (Introduction of the participants to the concept of 'app modifications')
(3) App Changes (repeatedly, as outlined in Section 3):
  (a) What app would you change?
  (b) What superpower would you use? (Change how app looks / Change how app works)
  (c) What change would you make, and why? (at least 10 words)
(4) User reflections:
  (a) In the past, if you wanted to change something about an app, what did you do? (tick all that apply) (Tell my friends about it on social media / Tell the developer privately (e.g. via email or direct messages) / Tell the developer publicly (e.g. on a tweet) / Research ways to change it and tried to do it myself / Do nothing / Other)
  (b) Which of the following activities are you comfortable doing on your smartphone? (check all that apply) (Taking photos / Reading emails / Installing apps from outside the official app store / Installing a custom operating system / None of this)
  (c) You may have other thoughts about changing apps. If so, please share them with us here. (optional)
  (d) Do you agree: "Changing things about apps on my own phone could be useful for me." (Rather yes / Rather no)
  (e) Why? Please explain.
  (f) Do you agree: "Everyone should be able to change apps on their own phone." (Rather yes / Rather no)
  (g) Why? Please explain.

Figure 6: Script for short semi-structured expert interviews about risks and opportunities of app repair.

(1) Intro:
  (a) How would you describe your main area of expertise? How many years of experience?
  (b) (Show a short demo of our GreaseDroid prototype to introduce the concept of app modification.)
(2) Main questions:
  (a) What might be common / pressing / important user needs? How might app repair help (or nor help) end-users? What groups of individuals might most benefit from such an approach and how?
  (b) How would it (positively / negatively) impact developers?
  (c) What do you see as key challenges around app repair? Practical? Ethical? Technical? Societal? Legal? How would you address them?
  (d) Should users be able to make changes to their apps? For what aims are patches ethical?
(3) Outro:
  (a) Is there anything that you'd like to add?

Code of Conduct: Ethical Declaration for Patch Developers

On submitting a patch for distribution in an app repair system, patch developers could vow that

(1) The submitted patch does no harm and is fair to all involved stakeholders.
(2) The submitted patch has been thoroughly tested across multiple devices, and app and Android versions.
(3) The patch is clearly legible for those who have basic programming skills.
(4) The submitted patch is free, libre and open-source software (FLOSS), for example through a GPL or Apache license. All necessary material is available to patch users.
(5) The authors will respond to requests relating to the patch in a timely and adequate manner, especially when these requests concern patch safety.

Figure 7: Inspired by our research findings and the famous Dogme 95 (i.e. a set of ten rules to ensure that films were still made in a traditional manner in a time with increasing reliance on digital technologies in films), we developed a sample code of conduct for those who develop patches. We developed this code of conduct by iterating over it over many months in our research group, which includes HCI experts, ethicists, software developers and legal experts, and also included the results from our survey and expert interviews. In developing the declaration, we took inspiration from the requirements of the F-Droid store. These rules, however, are only an initial set and will be further refined over time and in future work.