How Much Users Know About ‘Permission’ When They Permit: The Case of the Facebook App

Abdullah Hasan Safir
University of Cambridge
Edited by Lukas Gunschera

ABSTRACT

The Facebook app’s internal infrastructure makes it a giant data extraction medium, often exploiting the users unknowingly. Standing on this premise, this research examines the Manifest file of the app and relevant Application Programming Interfaces (APIs) embedded in its Software Development Kits (SDKs) and undertakes a tracker analysis of the app using the Exodus tool. The article shows that the reviewed version of the Facebook app has 59 permission requests in
the SDKs. Later, it critically analyses the notion of ‘permission’ concerning the network infrastructure and device affordances of the app. The article raises an important question: Can the users (really) ‘permit’? The article suggests that such innovative methods and approaches undertaken in this research can inform more effective policies to regulate platforms like Facebook by safeguarding users’ privacy.

**SCIENCE ⇒ POLICY**

Privacy violation issues are deep-rooted and largely hidden from users in the contemporary time of platform capitalism. These types of violations often involve application infrastructures and designed interactions with the devices on which the apps are installed. To regulate such data extraction by mobile applications, we need to understand to what extent users are informed about the data collection mechanisms and privacy policies of those apps.

**Key Words**

App Infrastructure; Critical Code Studies; Permission; Digital Hermeneutics

**Introduction**

While explaining how the contemporary capitalist big tech firms operate and influence the rest of the economy, Nick Srnicek classifies the characteristics of these platforms in his book ‘Platform Capitalism’ [1, p. 49]. Using these definitions, the social media platform Facebook, developed by its parent company Meta, might be labeled as an ‘advertising platform’ given that the company has a business model mostly dependent on the extraction of user data through this platform.

While the Facebook app is the third most popular app, accounting for 540 million global downloads as of October 2020 [2], the company has a well-established track record of data privacy controversies [3]. The most prominent instance in this regard is the Cambridge Analytica scandal which involved collecting millions of users’ data from the platform without their consent for distorting public opinions and election manipulation by a third-party consulting firm [4]. One article in the previous issue of this journal provokes a discussion around ‘informed consent’ in such contexts which implies asking users to give consent regarding the usage of their personal data by the platforms by agreeing to the privacy policies of the companies [5]. However, the privacy violation issues are far deep-rooted and involve application infrastructures and their designed interactions with the devices on which those apps are installed.

The current article instigates a discussion around the notion of ‘permission’ in relation to the network infrastructure and device affordances of the Facebook app. It critically analyses the Manifest file of the app and relevant Application Programming Interfaces (APIs) embedded in its Software Development Kits (SDKs) and subsequently undertakes a tracker analysis of the app using the Exodus tool. The analysis follows the ‘Critical Code Studies’ approach and frames the Facebook app as a critical object of the hermeneutics of ‘permission’. It presents evidence and allows readers to have an insight into how Facebook’s internal infrastructure makes it a giant data extraction medium, often exploiting the users unknowingly.

**Approach and Methods**

A mobile application is operationalized by an infrastructure consisting of the software itself, the device on which it is running, and the networked interactions between the two. The premise of this article lies in considering such infrastructures as complex material formations and, at the same time, as discursive constructions [6]. Software is mostly a systemic compilation of codes. This
research critically examines some parts of the codes of the Android version of the Facebook app to understand the specific relationships that it maintains with host devices and the data-intensive infrastructure that supports its operations. The analysis takes a ‘Critical Code Studies’ approach (see, [7]) to attain a more comprehensive understanding of the complex cultural situatedness of codes and reading beyond their mathematical or technical interpretations. This type of analysis follows careful consideration of the figurative and literal meanings of codes. In this article, codes related to ‘permission’ will be analysed.

First, the open-source Android Package Kits (.apk) file folder of the Facebook app (version 341.0.0.30.73) was downloaded from APKpure (https://m.apkpure.com/). This ZIP compressed file contains the source code, resources, and Manifest files – the files which make up the app [8]. To decompile the APK file, a particular Python library named ‘Androguard’ was used. The manifest file (manifest.xml), which is mandatorily part of the APK of any app and is, fortunately, both human- and machine-readable, describes essential information regarding the operating system of the app [9]. The Manifest file of the Facebook app is the particular interest of this analysis, as it lists the permissions that the app needs from its host device to do different tasks, such as getting locations, recording voices using the microphone, and taking photos using the camera. The element `<uses-permission>` in the codes of the manifest file defines these permissions.

To triangulate our findings, another web-based tool named Exodus (https://exodus-privacy.eu.org/en/) was used for examining these device permissions. In addition to giving the list of permissions, Exodus helps exploring how the Facebook app is inter-connected with other third-party trackers. These trackers enable inter-platform capture and transfer of user data. App developers use some specific software development tools namely Software Development Kits (SDKs) from existing open-source databases for enacting additional features and functionalities in their developed applications [10]. These SDKs often have Application Programming Interfaces (APIs), which ensure information flow toward the creators of those SDKs. Exodus deploys an API-based analysis of SDKs in a particular app (for our case, the Facebook app) for data capture and tracking which was not possible through the ‘Androguard’-enabled analysis in the first step.

Results

The Manifest file analysis and Exodus tool provide some critical information in relation to the permissions and trackers present in the Facebook application. Both analyses have confirmed the presence of 59 permission requests in the SDKs of the Facebook application for version 341.0.0.30.73 (see, Appendix for ‘Androguard’ analysis and Figure 1 for Exodus analysis).

Figure 1. Exodus showing 59 permissions and 0 trackers in the Facebook App (version 341.0.0.30.73)

Exodus has identified 13 of these permissions as ‘Dangerous’ or ‘Special’ level following Google’s protection levels advice which denotes that these permissions have higher risks and the application accessing such private data can negatively impact the users. These results show that the Android
operating system gives the Facebook app many permissions including permission to access network and GIS-based location; take pictures and videos; record audio; find accounts on the user’s device; read device status and identity; read and modify contacts; read, add or modify calendar events and details as well as send email to guests without owners’ knowledge; and finally modify or even delete the contents of the user’s SD card. Some of the attributes in the Manifest file confirm that codes for these requests come from different software libraries – including the Android vending library for billing. The codes also indicate that Android permits the app to use features of specific hardware including camera and microphone. However, only a few of these 59 permissions (for example, location, camera, and audio record) are asked explicitly from the users, in the majority cases these are user-blind interactions between the device and the app.

Surprisingly, Exodus has not found the code signature of any tracker known to them in the reviewed version of the Facebook app. In 2020, many versions of the Facebook app did have such trackers; the highest number of trackers was reported in October 2020. That version 293.0.0.43.120 of the Facebook app had nine trackers, including third-party ones like Google Analytics, Mapbox, and other Meta-developed trackers such as Ads, Analytics, Audience, Flipper, Notifications, Place, and Share. The presence of these trackers means that when a user logged into the app, the information automatically went to the third party, for example, Google Ads. This brief historiographic analysis of the app shows that the number of trackers changed significantly in one year (From nine to zero!).

Critical Reflections on Results

However, the qualities of Exodus as a decision-making tool might affect the results of this analysis. Van Es et al. [11, p. 26] point out that “as academics we maintain a critical attitude towards how this affects our work and find ways of making our choices in relation to these tools accountable”.

Exodus has built a Tracker Investigation Platform using the Django framework, where any technical expert who has evidence of a new tracker is able to create a profile for that tracker [12]. By explicitly mentioning this crowdsourced method (information gathered from the public), Exodus warns that its base of trackers may not be comprehensive, and it can thus miss unknown trackers in an application. This implies that although Exodus does not find the presence of any tracker in the reviewed version of the Facebook app, we cannot be sure about this information. There could be multiple trackers, but we do not know about them. It should be noted here that Meta itself has developed five of the top ten most frequent trackers used by all applications across the Google Play Store, including Facebook Login (present in 20% of all apps), Facebook Share, Facebook Analytics, Facebook Ads and Facebook Places [13]. Even other popular social apps use these trackers; for example, TikTok’s version 21.9.4 (November 2021) has ‘Facebook Login’ and ‘Facebook Share’ trackers. Facebook for Developers claims that these trackers provide TikTok and similar app users a ‘convenient’ way to log into and share content using their Facebook credentials [14]. In practice, these are nothing but processes for easier data capture and transfer between the platforms. The important point to note is that these trackers, in whatever number they exist, are never generally visible to the users. In summary, it is not possible for a user to know the details of permissions and trackers that exist in an app unless they undertake complex technical methods such as the ones applied in this research.

The (Digital) Hermeneutics of ‘Permission’

The Manifest file and Exodus use the word ‘permission’ and this particular word needs to be investigated beyond the technical analysis explained in the previous sections. Following the ‘Critical Code Studies’ approach, the current research reflects on this terminology in app development as a
cultural discourse: What does this ‘permission’ actually mean?

Wilson and Sperber emphasise the role of the addresser to break the cycle of the hearer’s inaction and disinterest as a requirement to describe a situation as ‘permission’ [15]. For example, if person A asks permission from person B to open a window, it is desirable to person A for the window to be opened, but they doubt its potentiality as person B can certainly refuse to let them open that window. By referring to them, Portner claimed that the connotative meanings of both ‘requirement’ and ‘permission imperatives’ can often be similar [16]. In this way, an application must inform the users about its required permissions from the Android before installation [17]. The users, on the opposite side, must have the agency to respond. They must have the right to accept or refuse to permit.

Practically, users are often unaware of this list of permissions and unknowingly click ‘accept’ for the few permissions they are explicitly presented (the discussions of ‘informed consent’ become relevant in those cases [5]). Even more important are the instances when apps assume the user’s consent with the installation of the app and automatically start receiving permissions from the device.

Davis and Chouinard provide useful analytical terminologies to explain this relationship between the users and the app [18]. The ‘permission’ feature can be attributed to the ‘request mechanism’ of affordances of the SDKs as theoretically these recommended actions can be accepted or refused by the users. However, practically, there exist no potential alternatives for users to install the app ignoring those permissions but only cancelling the installation process. Users cannot install the app with an agency to refuse these permission mechanisms as they are embedded within the app’s infrastructure. The permission features in the app are, in actual sense, interactions between the Android operating system and the applications (as explained in the sections 2 and 3), often acting independently by removing the users from the process. They create a situation in which these artifacts (the android operating system and the applications) can have their agency to ‘permit’ data or decision flow.

In these circumstances, a request for permissions works as a demand of an app for the users who are unaware of the whole process but eager to use the app. This raises a critical question: Can the users (really) ‘permit'? Our analysis shows that the answer is: no, at least in most cases. Essentially, users cannot permit if they are unaware of all that they are permitting to (in the case of the current app – all 59 of them). Moreover, if not permitting means not using the app at all then the word ‘permission’ does not make sense. This is particularly truer in this age of platform monopoly where there exist no viable alternatives of large social media platforms like Facebook [19] and when its parent company Meta has owned and merged other platforms such as Instagram, Messenger, and WhatsApp with it [20]. Therefore, users cannot help but accept a platform that has such large-scale commercial surveillance mechanisms considering the opportunity costs of the ease of communications it provides. When the users click to ‘accept’ for some permissions of such apps (let’s assume they have read the whole long texts that came up with these permission requests - which is often not the case, see, [21]), they can only get a glimpse of the real scenario of data capture mechanism. As explained in section 3, the major portion of the app infrastructure remains out of view and is difficult to perceive for users, such that they hardly participate in the overall process (to know more about this invisible nature of infrastructure see, [22]).

Policy Implications

The impact of these unnoticed permissions and trackers in a popular app like Facebook is deceitful for users. External vendors use Facebook and its technical infrastructures for service provision, user analysis, customer feedback, and even for payment facilitation.
Meta also shares user information with external companies for marketing their company and products, measures the effectiveness of their marketing campaigns, and performs advertising research [23]. Often, individuals or companies unknowingly share user data with Facebook by using its business tools. Facebook and other apps have abundant, often undue liberty to contravene the privacy and security of the users by intruding straight way to the delicate resources they share in the digital space and increasing user vulnerabilities [24]. In this way, companies like Meta translate human experience into behavioural raw material to produce marketable products to sell and maximise profit by undermining human autonomy [25].

This article, therefore, to better understand the Facebook app, looks at its Manifest file and analyses it as a critical object of the hermeneutics of permission. This Critical Code Studies approach and innovative methods provide novel insights around Facebook’s use of invisible internal infrastructure to exploit users and unknowingly extract massive data. It shows that app ‘permissions’ are not necessarily about the users as it sounds like; rather, users remain vulnerable in this whole process of data collection. Existing informed consent mechanisms (in fact, which is often not effective, see [26]) only deal with permissions that are visible to the users, but in practice, most of the data collection happens by keeping the users ignorant about the process - as this article shows. Therefore, the policies around digital privacy need to be informed by detailed, concrete knowledge of how the platforms violate privacy through intricate mechanisms such as APIs and trackers. The more we know about the internal infrastructures of the data extractors, the more we will be able to formulate effective policies to regulate malpractices around privacy breach.

References


About the Author

Abdullah Hasan Safir
Abdullah Hasan Safir is a Research Assistant at Leverhulme Centre for the Future of Intelligence (LCFI) at the University of Cambridge. He was a member of the scientific and organizing committee of ‘Many Worlds of AI’ conference at Cambridge. Safir is currently working on delivering academic and policy-oriented outputs on intercultural perspectives on global ethical AI policies and governance practices based on this event. Safir finished his Master’s from the Centre for Interdisciplinary Methodologies at the University of Warwick with Commonwealth Scholarship, achieved distinctions and was awarded for his academic excellence. His previous publications engage with design and development issues at the intersection of digital technologies and their implications on Rohingya refugees and Internally Displaced Populations in Bangladesh. Safir’s current research interest lies in critically analysing AI, codes, data infrastructures and digital artifacts, and reimagining them - particularly from the Global South/ Majority World perspectives.

This research is an original work and was part of the author’s academic endeavours undertaken at the Centre for Interdisciplinary Methodologies at the University of Warwick as part of his master’s programme.

ORCID: 0009-0003-2328-3840

Corresponding address:
sa2168@cam.ac.uk

Conflict of interest: The author declares no conflict of interest.
Appendix: Manifest File of Facebook App

[Note: The file has been shortened due to its length, only permission features have been copied here from the original manifest file.]

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="308212468"
android:versionName="341.0.0.30.73" android:installLocation="0"
android:compileSdkVersion="30"
android:compileSdkVersionCodename="11" package="com.facebook.katana"
platformBuildVersionCode="30"
platformBuildVersionName="11">
  <uses-permission-sdk-23 android:packageName="com.android.vending.BILLING"/>
  <compatible-screens>
    <screen android:screenSize="200" android:screenDensity="560"/>
    <screen android:screenSize="300" android:screenDensity="560"/>
    <screen android:screenSize="400" android:screenDensity="560"/>
    <screen android:screenSize="500" android:screenDensity="560"/>
    <screen android:screenSize="200" android:screenDensity="640"/>
    <screen android:screenSize="300" android:screenDensity="640"/>
    <screen android:screenSize="400" android:screenDensity="640"/>
    <screen android:screenSize="500" android:screenDensity="640"/>
  </compatible-screens>
  <uses-sdk android:minSdkVersion="28" android:targetSdkVersion="30"/>
  <uses-permission-sdk-23 android:name="android.permission.BLUETOOTH"/>
  <uses-permission-sdk-23 android:name="android.permission.BLUETOOTH_ADMIN"/>
  <uses-permission-sdk-23 android:name="android.permission.CALL_PHONE"/>
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
  <uses-permission android:name="android.permission.WAKE_LOCK"/>
  <uses-permission android:name="android.permission.VIBRATE"/>
  <uses-permission android:name="android.permission.READ_CONTACTS"/>
  <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
  <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
  <uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
  <uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS"/>
  <uses-permission android:name="android.permission.READ_SYNC_SETTINGS"/>
  <uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS"/>
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
  <uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
  <uses-permission android:name="android.permission.READ_PHONE_NUMBERS"/>
  <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
  <uses-permission android:name="android.permission.READ_CALENDAR"/>
  <uses-permission android:name="android.permission.WRITE_CALENDAR"/>
  <uses-feature android:name="android.hardware.camera" android:required="false"/>
  <uses-feature android:name="android.hardware.microphone" android:required="false"/>
  <uses-feature android:name="android.hardware.location" android:required="false"/>
  <uses-feature android:name="android.hardware.location.network" android:required="false"/>
  <uses-feature android:name="android.hardware.location.gps" android:required="false"/>
  <uses-permission-sdk-23 android:name="android.permission.READ_PHONE_NUMBERS"/>
  <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
  <uses-permission android:name="android.permission.READ_CALENDAR"/>
  <uses-permission android:name="android.permission.WRITE_CALENDAR"/>
</manifest>