SOCIALTIC

Digital technology for social change

Technical Report on Tricount, Splitwise and YNAB

Paul Aguilar

Diego Morábito

February, 2022

General remarks

Methodology

We made a static analysis of every app via the Exodus Privacy project web page[1] to find all the trackers embedded in the code and also to find all the permissions also embedded in the code.

We collected web traffic from a Galaxy A5 (2017) phone with Android 8 based in Madrid, Spain connected to WiFi. The phone contained most of the apps that come pre-installed with it[2] and of course all the other apps that are not visible for the common user. This was made to emulate the usage of a common person. To capture the data we set up an Amazon Server located in Frankfurt, Germany with Ubuntu Server 20.04 LTS and OpenVPN 2.5.5 running on top of it. We also installed Tshark 3.6.2.. We connected the phone to the VPN and captured the data packets with Tshark to a .pcap file that later was analysed with Wireshark 3.6.2. It is important to note that we used a phone based in Spain and a server in Germany to trace the geolocation of servers the data was being sent to.

We installed one of the apps analysed (either YNAB, Splitwise or Tricount) with all the other apps closed and began testing the app by logging in with a Gmail account (also based in Spain) and then testing most of the app functions while we were capturing web traffic. The Android phone was also logged in with a Gmail Spain based account (the same we used to log in to the apps). When trying app functions we also noted all permissions the app requests from the user to compare them with the permissions embedded in the code.

Because we were worried there might be too much noise from all the other apps installed in the Android 8 phone, we did a similar test with a Moto G7 Plus with Lineage OS 18.1 installed, so we could have a more controlled environment. No other apps were installed on this phone, we disabled most of the phone services from running, and we only left the needed ones to run the app in question. This phone and the Gmail account we used were based in Mexico, the connection was made via WIFi. We connected it to a private server (not an Amazon one) based in Miami, United States with the same versions (as stated above) of OpenVPN, Ubuntu and Tshark running.

Then we cross-checked the volume of connections and the geolocalization of each server to discard connections not made from the apps we were analysing (for example connections to DNS servers).[3] [4]

We compared the connections made by each app to the owners of the servers to find out if there were trackers that the static analysis could have missed. We checked the publicly available information of every tracker and each company linked to a server to deduce the kind of data that was probably being sent from the app to each server, and to identify CDN's (Content Distribution Networks) and other kinds of servers.

---

[1] https://exodus-privacy.eu.org/en/

[2] WiFi File Transfer, Tiny Scanner, Samsung Pay, LinkedIn, Camera, Radio, Calendar, Clock, Samsung Notes, Calculator, Settings, Contacts, Gallery, Messages, Phone, PlayStore, Galaxy Store, Word, Excel, PowerPoint, OneDrive, Google, Chrome, Gmail, Maps, Drive, Documents, Voice Recorder, Email, My Files, Internet (its an web explorer), Samsung Health, Samsung Members, Safe Folder, Galaxy Themes.

[3] 3. Even though by the time of this publication app versions may be different, we checked and the trackers are still the same, so it is safe to assume that differences might be minimal.

[4] Connections discarded are not shown in the results.

Important Notes

We did not use all of the app functionalities, such as bank account or PayPal integration, therefore that part remains untested.

It is also important to note that we are not active users of these apps. This means that even if we tried almost all of the functionalities and instances of each app, data may change with an active and long-term user.

It is important to make a distinction between the data collected by the actual usage of the apps (and given willingly by the user), i.e. transactions, amount of each transaction, payments made, user name, email, etc., and the one collected by the trackers. Trackers, in general, only collect statistical and anonymous data. The data that the user gives voluntarily is stored within the servers where the app is hosted. Even if this information is hosted on Google, Amazon or another service, that does not mean that the owners of these servers have access to it. What it actually means is that this data could be probably shared with anyone the app developer wants to share it with. One could argue that all of these apps (given the sensitive financial information shared by users) should implement a no-knowledge functionality, where even the app developer has no access to the data being given willingly by users (exactly what WhatsApp claims it is doing with our chats or what some VPN services do.)

Usage of trackers

Trackers	Tricount	Splitwise	Revolut	YNAB	Fintonic
Google Firebase Analytics	x	x	x	x	x
Google Crashlytics	x	x	x		x
Google AdMob	x				x
Google Analytic	x				
Google Tag Manager	x				
Branch	x		x		
Huawei Mobile Services (HMS) Core	x		x		
Appsflyer					x
LeanPlum					x
Braze				x	
Bugsnag				x	
mParticle				x	
Pusher				x	
Amazon Advertisement	x				
Amazon Analytics		x			
Facebook Login	x				
Facebook Share	x				
Smart	x				

Apps

Tricount 5.3.0

Data gathering profile

These **Google** trackers are present in the app:

- Google Analytics[5]
- Google Crashlytics[6]
- Google Firebase Analytics[7]
- Google Tag Manager[8]
- Admob[9]

Derived from public information available on these **Google** trackers (all now contained within the Firebase SDK framework), we can suppose certain things.

Google Crashlytics records what the user was doing when the app crashed. Probably they also record an ID that has certain system specifications like OS version, app version and Smartphone brand, so as to link specific user behaviour to specific software and hardware.

Google Analytics, along with Firebase Analytics and Google Tag Manager record over 500 types of custom events that the developer decides to implement and to link this information to a user ID. These events may be: time spent using the app, time spent on each instance of the app, clicks made, purchases made, etc..

We do not know exactly what type of information is gathered or how much. We can also suppose that the tracker Tag Manager is used to tag all these events and share them with the Branch.io tracker. Probably many of these events create a profile of the users to better target them with the **AdMob tracker**.

Amazon (via Amazon adtracker)[10] is used to let Amazon clients have their products advertised. In that sense, it probably collects certain data to identify the user, the hardware used, the operating system (OS), and maybe some other specific details such as country, language, time, and app version, to create a really specific user ID that can be correctly ad-targeted.

Then we have the **Branch** tracker. One of its primary functions is to link data between different channels and devices. In this case, we can infer the web page of Tricount also has the Branch.io tracker installed.

It is supposed to help developers maximise user engagement and performance. Based on public documentation we can infer it tracks a user by setting a user ID and linking it to certain events (the documentation mentions 5 kind of events), the more important ones for our project are:

---

[5] https://firebase.google.com/docs/ads
[6] https://firebase.google.com/docs/crashlytics
[7] https://firebase.google.com/docs/analytics
[8] https://developers.google.com/tag-platform/tag-manager/android/v5?hl=en
[9] https://firebase.google.com/docs/admob
[10] https://advertising.amazon.com/API/docs/en-us/info/api-overview

- When a user buys something
- When a user interacts with any instance of the app
- When the user progresses in the usage of the app (creates a profile, links an email, etc.)
- Custom events defined by the developer[11]

Also, if you have content (like frequently asked questions, for example) it can tell you how many times this content has been viewed.

Branch.io might be connected to Facebook if the developers want to, so it could potentially check if your phone has a Facebook app installed in order to make a link between your Facebook profile and this tracker.

It also has the possibility to record events recorded by Google Tag Manager and Google Firebase. Therefore, we can safely assume that Branch has the possibility to access most of the information that Google trackers get from the user.

The **Facebook** tracker (part of the Facebook SDK) works in a similar way as the other trackers we've been analysing[12]. You connect this tracker to the developer's Facebook Ad Account and then it collects:

- The apps you have installed on your device
- How many times your app has been launched
- Records of your in-app purchases
- All sorts of events linked to certain parameters the app developer can decide to use[13]

The **Smart** tracker[14] is a service that targets ads to the user - it collects the data necessary to identify a user and then serves specific ads to them.

The **Huawei** tracker is embedded in the Huawei SDK. Its first and most important function is that it allows the app to run on Huawei devices. Since this framework offers quite a lot of possibilities, we are not sure exactly which data, if any, it is gathering[15].

---

[11] Here is a detailed explanation of the events. https://help.branch.io/developers-hub/docs/tracking-commerce-content-lifecycle-and-custom-events#section-track-commerce-events. And here is an even more detailed explanation of data that potentially could be gathered: https://help.branch.io/developers-hub/docs/branch-event-ontology.
https://help.branch.io/developers-hub/docs/logging-branch-events-using-google-tag-manager

[12] https://developers.facebook.com/docs/app-events/getting-started-app-events-android

[13] This is a short list of how events may be triggered. https://developers.facebook.com/docs/app-events/reference#standard-event-parameters

[14] https://documentation.smartadserver.com/displaySDK/index.html

[15] https://developer.huawei.com/consumer/en/doc/start/sdkindex-0000001053768544

Endpoints resume

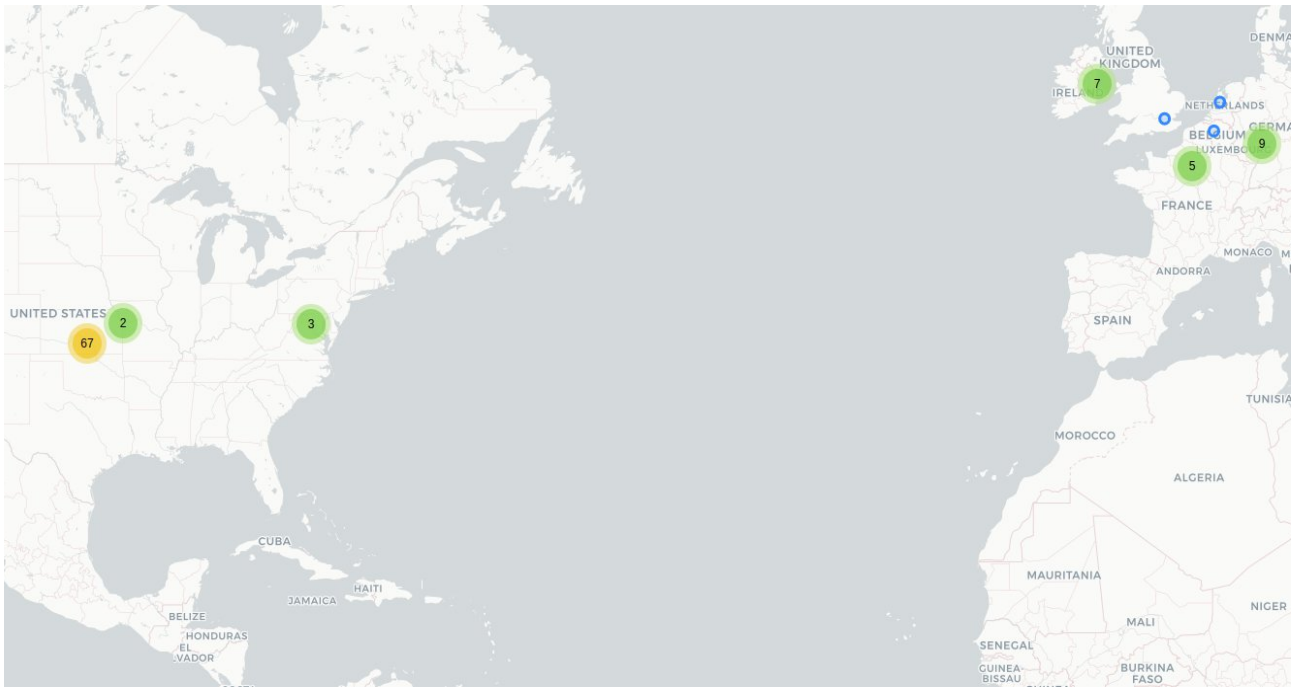
- Connections from a Mexico-based device with exit in a Miami server
- Connections from a device based in Spain with exit in Frankfurt, Germany

	Smartphone with Lineage OS (controlled environment)		Smartphone with Android in Spain (normal user environment)	
Server Company	**Number of servers**	**Location**	**Number of servers**	**Location**
SMARTADSERVER	5	Canada		
Smartadserver	1	France		
Smartadserver			3	France
OVH-SAS	2	France; US	1	France
Amazon	4	Germany	2	Germany
Amazon	35	US	17	US
Amazon			6	Ireland
Akamai	4	US		
Akamai			3	Germany
Google	46	US	42	US
Google			1	Germany
Freewheel	1	US		
Alibaba	1	US		
ASN APPnex	1	US		
Yahoo	1	US		
Yahoo			1	UK
Yahoo			1	Ireland
Highwinds	1	US	1	US
Adsafe	1	US		
Facebook	2	US		
Facebook			1	France
Facebook			3	Germany
Fastly			4	US
LLNW			1	US
Microsoft			1	US
Microsoft			1	Netherlands
Adjust Gmbh	2	US	1	Belgium
SPOTX-IAD	1	US		
SPOTX-DEN	0			
Uninet	0			

Connections resume

Connections from device with Lineage OS (from Mexico with exit in Miami)

Connections Android in Spain with exit in Frankfurt, Germany



Permissions and information

Permission by static analysis	Description
ACCESS_NETWORK_STATE	View network connections
ACCESS_WIFI_STATE	View WiFi Connections
INTERNET	Full Network Access
READ_CONTACTS	Read Your contacts
READ_EXTERNAL_STORAGE	Read the contents of SD card
RECEIVE_BOOT_COMPLETED	Run at startup
VIBRATE	Control Vibration
WAKE_LOCK	Prevent phone from sleeping
WRITE_EXTERNAL_STORAGE	Modify or delete contents of your SD card
BILLING	Access Google billing
RECEIVE	Allows apps to accept cloud to device messages sent by the app's service
BIND_GET_INSTALL_REFERRER_SERVICE	Google Play Store Installation Information
AD_ID	ID for personalized advertisement. As of 2021 can be disabled in smartphone configuration

Permissions and information asked in-app usage	Type (Required, Optional, but app loses functionality, Optional)
CAMERA	Optional, but the app loses functionality
READ_CONTACTS	Optional, but the app loses functionality
Access To Email Or Google Account	Optional, but app loses functionality
Currency	Required
Name	Required (when not logging in with Google account)

Splitwise 5.1.9/585

Data gathering profile

This app has only three trackers. **Amazon Analytics, Google Firebase Analytics and Google Crashlytics**. Of the three apps analysed so far, it is the one with the fewest trackers. Google trackers are explained in the previous app, so we will only analyse Amazon Analytics.

Amazon Analytics, as stated in their public documentation[16], is used as a tracker that analyses installs, returning app users and custom events decided by the developer. This probably includes, as we have seen, clicks, time usage, sessions, etc.. They probably collect this data linked with a specific ID of the user.

Endpoints resume

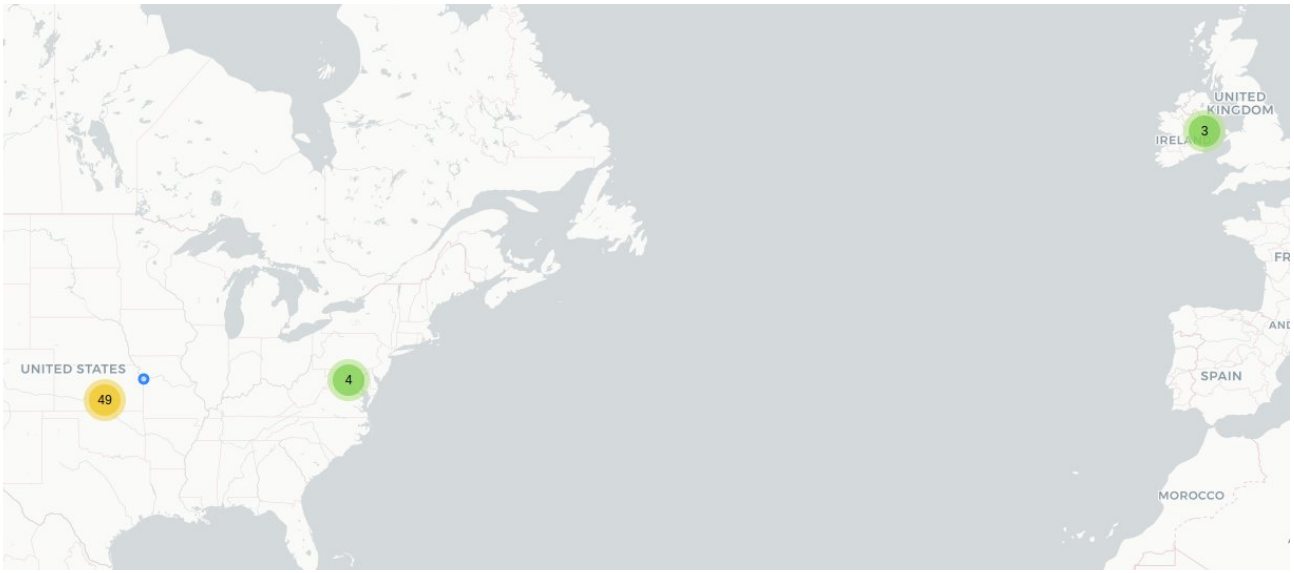
	Smartphone with Lineage OS (controlled environment)		Smartphone with Android in Spain (normal user environment)	
Server Company	**Number of servers**	**Location**	**Number of servers**	**Location**
Amazon	26	US	13	US
Amazon			3	Ireland
Google	18	US	42	US

Connections resume

Connections from device with Lineage OS (from Mexico with exit in Miami)

[16] https://docs.aws.amazon.com/mobile/sdkforxamarin/developerguide/getting-started-analytics.html

Connections Android in Spain with exit in Frankfurt, Germany



Permissions and information

Permission by static analysis	Description
ACCESS_NETWORK_STATE	View network connections
AUTHENTICATE_ACCOUNTS	Allows the app to use the account authenticator capabilities of the Account Manager, including creating accounts and getting and setting their passwords.
CAMERA	Take pictures and videos
FOREGROUND_SERVICE	Run foreground service
GET_ACCOUNTS	Find accounts on the device
INTERNET	Full Network Access
READ_CONTACTS	Read Your contacts
READ_EXTERNAL_STORAGE	Read the contents of SD card
READ_SYNC_SETTINGS	Read sync settings
RECEIVE_BOOT_COMPLETED	Run at startup
USE_BIOMETRIC	Use biometric hardware
USE_CREDENTIALS	Allows the app to request authentication tokens
USE_FINGERPRINT	Use fingerprint hardware
VIBRATE	Control Vibration
WAKE_LOCK	Prevent phone from sleeping
WRITE_EXTERNAL_STORAGE	Modify or delete contents of your SD card
WRITE_SYNC_SETTINGS	Toggles sync on and off

BILLING	Access Google billing
RECEIVE	Allows apps to accept cloud to device messages sent by the app's service
READ_GSERVICES	Allows this app to read Google service configuration data

Permissions and information asked in-app usage	Type (Required, Optional, but app loses functionality, Optional)
CAMERA	Optional, but the app loses functionality
READ_CONTACTS	Optional, but the app loses functionality
Access To Email Or Google Account	Required
Name	Required (When not using a Google account)
Telephone Number	Optional, but the app loses functionality
Currency	Required
Profile Photo	Optional

You Need A Budget / YNAB 7.6.2

Data gathering profile

Trackers embedded in the app are **Bugsnag, Braze, Mparticle** and **Pusher**.

Bugsnag is a tracker for reporting app crashes, similar to Google Crashlytics. From their documentation[17] we know the following information is gathered:

- App running time
- Running time in foreground and active screen

Build information of the app, which includes:

- Name
- Version
- Release stage (beta, for example)
- Manufacturer of the smartphone
- Model
- OS version
- Screen size and density and total memory

It also collects system state, that includes:

- Screen orientation
- Free memory
- Available disk space
- Battery level
- Network connectivity.

We can safely assume that Google Crashlytics also records this data, since the two apps have similar functions. And probably, Bugsnag, like most of the other trackers we've seen, generates a unique ID of the user device and also tracks:

- Activity Lifecycle callbacks
- Network connectivity changes
- Bluetooth connectivity changes
- Battery state changes

- Device rotation
- Media Scanner events
- Telephony events

---

[17] https://docs.bugsnag.com/platforms/android/
https://docs.bugsnag.com/platforms/android/automatically-captured-data/#breadcrumbs

The **Braze** tracker is used mainly for marketing purposes. From the public documentation we know it collects[18]:

- First Used App (Time)
- Last Used App (Time)
- Total Session Count (Number)
- Number of Feedback Items (Number)
- Number of Sessions in the Last Y Days (Number and Time)
- Email Available (Boolean)
- News Feed View Count (Number)
- Location Available (Boolean)
- Most Recent Location (if location permission is granted to your app)
- Push Enabled (Boolean)
- Device Locale
- Language (taken from Device Locale)
- Country (first taken from IP Address. If this is not available, taken from Device Locale)
- Most Recent App Version
- Device Model
- Device OS Version
- Device Resolution
- Device Wireless Carrier
- Device Time Zone
- Device Identifier
- Uninstalled (Time and Boolean)

It also records some custom events defined by the developer and it also has this function: "*The User API allows you to track information on your users by logging data about your users that comes from outside your mobile app.*[19]"

Mparticle states[20] on their webpage:

"mParticle is a customer data platform that simplifies how you collect and connect your user data to hundreds of vendors without needing to manage multiple integrations. We simplify the entire process for you, so you can do more with your data without the hassle of complex integrations."

Services they provide include:

- IDsync[21] - this service allows the developer to give a user a unique ID that can be tracked when the user is using an app but has not been logged in and then they continue to use it when they are logged in; can track a user on multiple devices and platforms; can track a user in different apps; it also has "*the ability to provide evidence that demonstrates that your organization is in regulatory compliance is important to every Chief Privacy Officer and corporate information security executive. GDPR and CCPA*

---

[18] https://www.braze.com/docs/developer_guide/platform_wide/analytics_overview/#automatically-collected-data
[19] https://www.braze.com/docs/api/endpoints/user_data/
[20] https://docs.mparticle.com/
[21] https://docs.mparticle.com/guides/idsync/use-cases/

data privacy controls and traceability are core to mParticle's user profile data policies. In addition, the IDSync Search capability can verify that a matching User Profile exists. It can also be used after a GDPR or CCPA User Profile Delete Request has been processed, to validate that the process has completed successfully and thereby validate compliance." It can generate a user ID linked between social media, email, IP address or session ID. *"Behind the scenes, mParticle maintains a User Profile for each user. You can think of a User Profile as a big folder of data: events, user attributes, identities, attribution info, device info. User Profiles are used to drive the Audience Builder and to enrich incoming data with all relevant information about a user before forwarding it to an Output service*."

- Another service is Audiences[22] - this service is used for two purposes in general. The first one is to make users more engaged with the app. By forwarding data gathered from analysing user habits to services like Facebook or Mailchimp, they can automatically start push notifications, mails and ads to make the users use the app more. The second goal of this service is to track and build a profile of the users that actually use an app frequently, and based on this information, create targeted ad campaigns on Facebook or other social media platforms to attract more similar customers.

All this is done and really well explained in this link: https://docs.mparticle.com/guides/platform-guide/activity/. We also suggest you read the information in this other link about privacy: https://docs.mparticle.com/guides/data-subject-requests/forwarding/

The **Pusher** tracker, from publicly available information, is a piece of software that collects data in order to implement certain app functions such as:

- Real time chats
- Notifications
- Location tracking
- In-app chat.

In this specific case, this tracker is probably present for the in-app chat function.

In the section *Help* of the app, there are texts (blog), podcasts and videos hosted on YouTube. When the user accesses this section there are many connections to Facebook, YouTube, Spotify, Snapchat, LinkedIn, Twitter and Pinterest. These connections transfer user data to these services, but are minimal. They are likely necessary to make this service work properly. In the tests we ran on the Spain-based device, these connections do not appear because a YouTube video was never played or the blog accessed.

This app is probably hosted either on Google or Amazon servers. So many connections to these servers are needed for the app to work. There are also many connections with servers owned by Fastly. Fastly is a CDN (we do not know by which tracker or app functions it is used).

---

[22] https://docs.mparticle.com/guides/platform-guide/audiences/

Endpoints resume

	Smartphone with Lineage OS (controlled environment)		Smartphone with Android in Spain (normal user environment)	
Server Company	**Number of servers**	**Location**	**Number of servers**	**Location**
Amazon	16	US	11	US
Amazon			2	Ireland
Akamai	4	US		
Google	39	US	18	US
Fastly	15	US	6	Fastly
Facebook	2	US		
Microsoft	1	US		
LinkedIn	1	US		
Uninet	0			
Twitter	2	US		

Connections resume

Connections from device with Lineage OS (from Mexico with exit in Miami)

Connections Android in Spain with exit in Frankfurt, Germany

Permissions and information

Permissions by static analysis	Description
ACCESS_FINE_LOCATION	access precise location (GPS and network-based)
ACCESS_NETWORK_STATE	view network connections
FOREGROUND_SERVICE	run foreground service
INTERNET	have full network access
READ_EXTERNAL_STORAGE	read the contents of your SD card
RECEIVE_BOOT_COMPLETED	run at startup
WAKE_LOCK	prevent phone from sleeping
WRITE_EXTERNAL_STORAGE	modify or delete the contents of your SD card
BILLING	Access Google billing
BIND_GET_INSTALL_REFERRER_SERVICE	Google Play Store Installation Information
RECEIVE	Allows apps to accept cloud to device messages sent by the app's service

Permissions and information asked in-app usage	Type (Required, Optional, but app loses functionality, Optional)
Email, Google account or Apple account	Required
Location access	Optional
Currency	Required
Add Bank Accounts	Optional, but app loses functionality
Paypal Account	Optional, but app loses functionality

Conclusions

Because data is encrypted, without breaking the security provided by SSL Pinning and HTTPS, it is impossible to know which actual data is being collected. We can make educated guesses based on the public information gathered about each tracker, but in the end, almost every tracker has multiple functions that may or may not be implemented by the app developers.

We can say for sure that every tracker generates a user specific ID based probably on country, language, OS, app version, local time, hardware vendor and the carrier. It is an educated guess that far more details (all of them anonymised) are likely gathered from each user, so as to make the user ID more specific. Then, this user ID serves the purpose of the tracker, whether it be a targeted advertisement, usage analytics or bug reports. Then, usage analytics can be used, as stated in the mParticle tracker, to modify user behaviour or define specific targeted ad campaigns.

mParticle, at least because of the transparency in its documentation, seems to be a kind of “super tracker” in a way that it helps the developer gather specific data from a user, confront that data with other data gathered by other means (social networks and web surfing) from the same user and then use that data with other services. Then, it must also be the case that Facebook, Amazon and Google do the same thing, but since they are so big the “other services” they share data with are their own.

Most of the data sent by these apps is being gathered in US based servers. Because of GDPR (General Data Protection Regulation), it remains to see if anonymised data must stay in Europe according to law or whether it may be stored in the US. In any case, it is important to differentiate between data that is not anonymous (not statistical data) and data that is. Data freely given by the user must be put in a different category from the one that is being gathered by trackers, and laws must take that into consideration. In this case it is also important to note that, when analysing web traffic, we cannot know for sure where these two types of data are being sent.

As security for users is scaling (which is a good thing), the possibility to verify what big data companies are gathering from users becomes harder (and illegal), so we end up in a game where more security also means more opacity in what big corporations do.

Another big subject is about software development kits (SDK’s). They help developers save time and make better apps, but at the same time a great percentage of the internet is being built on frameworks owned by corporations such as Google, Facebook, Microsoft and Amazon. That means that, as a developer, the data you need gathered for your apps to work is also being shared with whatever SDK owner you chose to use. This, obviously, centralizes the internet infrastructure and gives these corporations a lot of power. The problem, then, is not only users’ ignorance or laziness, it is also that developers are choosing or are pushed to be part of this data extraction situation.