On The Data Privacy Practices Of Android OEMs

Haoyu Liu[1], Paul Patras[1], Douglas J. Leith[2]
[1]University of Edinburgh, UK [2]Trinity College Dublin, Ireland

Abstract

In this paper we present the first in-depth measurement study of the data privacy practices of the proprietary variants of the Android OS produced by Samsung, Xiaomi, Huawei and Realme. We address two questions: how are identifiers used in network connections and what types of data are transmitted. We find that all of the OEMs make undue use of long-lived hardware identifiers such as the hardware serial number, handset IMEI and so fail to follow best privacy practice. Hardware identifiers are also linked to the handset user's real identity when they sign in to an OEM account on the handset. All of the OEMs collect the list of apps installed in a handset. This is a privacy concern since the list of installed apps can be used to profile user traits and preferences. All of the OEMs collect analytics/telemetry data.

1 Introduction

The privacy of mobile apps has been extensively studied, but much less attention has been paid to the privacy pratices of the mobile OS itself. While the Android OS[1] is partly open source[2] it is usually customised by so-called Original Equipment Manufacturers (OEMs) before being deployed on their handset hardware. Most of these OS customisations are proprietary and closed, with little public documentation. In this paper we use measurements of the data transmitted by handsets manufactured by Samsung, Xiaomi, Huawei and Realme to analyse the data privacy practices of these OEMs. Together, these OEMs represent the great majority of the Android handset market in Europe (Samsung currently has by far the largest share of the Android handset market, followed by Xiaomi, Huawei and Oppo the parent company of Realme [1]).

We seek to address two questions. Firstly, how are identifiers used in the network connections made by the Android OEM software. Best practice, e.g. see [2], is to minimise the persistence of the identifiers used so as to enhance user privacy. For example, long-lived hardware identifiers such as the hardware serial number, handset IMEI, sim IMSI should only be used in a strictly limited way. It is also good practice to isolate identifiers used for different purposes e.g. advertising identifiers should be kept separate from identifiers used for other purposes and in particular kept separate from long-lived hardware identifiers.

The second question we address is what types of data are transmitted by Android OEM software. A mobile OS is, of course, expected to communicate with servers to check for software updates, but we find that all of the OEMs studied also transmit advertising and analytics data as well as data for OEM services such as antivirus checkers, caller ID/blocking and remote device configuration. The latter occurs in our

[1]By Android OS we mean the distribution as installed on a handset, not just the kernel. This includes system apps which are privileged pre-installed apps that the OS developer bundles with the OS. They cannot be deleted (they are installed on a protected read-only disk partition) and can be granted enhanced rights/permissions not available to ordinary apps such as those that a user might install.

[2]The hardware drivers and system-on-a-chip firmware are generally closed-source.

experiments despite the user opting out of OEM services when asked, and making no explicit use of these services.

We focus here on defining simple scenarios that can be applied uniformly to the handsets studied and that generate reproducible behaviour, so allowing direct comparisons. We assume a privacy-conscious but busy/non-technical user, who when asked does not select options that share data but otherwise leaves handset settings at their default value.

To the best of our knowledge this work is the first in-depth measurement study of the data privacy practices of popular proprietary variants of the Android OS. The analysis of whether mobile apps disclose sensitive information to their associated back-end servers has been the focus of much research [3, 4, 5, 6, 7], especially with a view to risks such user de-anonymization, location tracking, behaviour profiling, and cross-linking of data by different stakeholders in the device/software supply chain. Probably closest to the present work are recent analyses of the data that web browsers share with their back-end servers [8] and of the data shared by Google Play Services [9, 10]. The latter is motivated in part by the emergence of Covid contact tracing apps based on the Google-Apple Exposure Notification (GAEN) system, which on Android requires that Google Play Services to be enabled. The present study is significantly broader in scope.

2 Methodology

2.1 Hardware and Software Used

Mobile handsets: (i) Samsung Galaxy S9 (model SM-G960F)/Android 10 (build QP1A.190711.020, One UI v2.0), (ii) Xiaomi Redmi Note 9 (model M2003J15SG)/Android 10 (build QP1A.190711.020, MIUI Global 12.0.7 QJOMIXM), (iii) Realme 6 Pro (model RMX2063)/Android 10 (build `RMX2063_11_A.38`, realme UI v1.0), (iv) Huawei P10 Lite (model MAR-LX1B)/Android 9[3] (build 9.1.0.372, EMUI 9.1.0). Rooted using Magisk v20.4 and Magisk Manager v7.5.1.

WiFi access point: Raspberry Pi 4 Model B Rev 1.2/Raspbian GNU Linux 11/Mitmproxy 6.0.2 with iptables firewall configured to redirect HTTP/S traffic to port 8080 (on which mitmproxy listens) and also to block UDP traffic on HTTPS port 443 (so as to force any Google QUIC traffic to fall back to using TCP since we have no tools for decrypting QUIC).

2.2 Device Settings

At the start of each test we removed any SIM card and carried out a hard factory reset of the handset, i.e. we used `TWRP` to manually wipe the data partition, thereby forcibly removing all user data and settings, all user installed apps and resetting any disk encryption. Note that we observed that simply clicking on the "factory reset" option in the UI sometimes did not fully remove user data and settings.

Following this factory reset, the handset reboots to a welcome screen and the user is then typically asked to agree to terms and conditions, and presented with a number of option screens. To mimic a privacy conscious user, we unchecked any of the options that asked to share data and only agreed to mandatory terms and conditions. *Samsung*: we unchecked the *Sending of Diagnostic Data*, *Information Linking*, *Receipt of Marketing Information* components of the terms and conditions, skipped the *Protect Your Phone screen*, did not sign into a Samsung account (which raises a warning that it disables Samsung Cloud, Bixby, Galaxy Themes, Find My Mobile, Samsung Pass, Galaxy Store,

[3] Following US trade sanctions against Huawei, Android 9 is the latest version of Android available on a Huawei handset that we could root.

Secure Folder). *Xiaomi*: we unchecked the *Location*, *Send Diagnostic Data Automatically*, *Automatic System Updates*, *Personalised Ads, User Experience Programme* options. *Realme*: we unchecked the *User Experience Programme* and *Uploading Device Error Log Data* components of the terms of service, unchecked the *WiFi Assistant* and *Auto-update Overnight* options. *Huawei*: we selected *No Thanks* on the *Enhanced Services* screen, *Later* on the *User Experience Improvement Programme* screen, *Update Manually* on the *Keep Your Software Up To Date* screen. All of the mobile OSe also displayed a Google services screen on first startup. On this we unchecked the *Use Location*, *Allow Scanning*, *Send Usage and Diagnostic Data* options, and we did not log in to a Google user account.

2.3 Test Design

We focus here on defining simple scenarios that can be applied uniformly to the handsets studied and that generate reproducible behaviour, so allowing direct comparisons. For each handset we carry out the following experiments:

1. Start the handset following a factory reset (mimicking a user receiving a new phone), recording the network activity.
2. Insert a SIM, recording the network activity.
3. Following startup, leave the handset untouched for several days (with power cable connected) and record the network activity. This allows us to measure the connections made when the handset is sitting idle. This test is repeated with the user being logged in and logged out, and with location enabled/disabled.
4. Open the settings app and view every option but leave the settings unchanged, recording the network activity. Then close the app.
5. Open the settings app and enable location, then disable. Record the network activity.
6. Make and receive a phone call, send and receive an SMS message. Record the network activity.
7. Create an OEM user account, if possible. Record the network activity.

2.4 Decrypting HTTPS Connections

All of the data we observe is sent over HTTPS connections and so encrypted using TLS/SSL. To decrypt these connections we route handset traffic via a WiFi access point (AP) that we control, configure this AP to use `mitmdump` as a proxy [11] and adjust the firewall settings to redirect all WiFi HTTP/HTTPS traffic to mitmdump so that the proxying is transparent to the handset. When a process running on the handset starts a new network connection, the mitmdump proxy pretends to be the destination server and presents a fake certificate for the target server. This allows mitmdump to decrypt the traffic. It then creates an onward connection to the actual target server and acts as an intermediary, relaying requests and their replies between the app and the target server while logging the traffic. The setup is illustrated schematically in Figure 1.

The mobile handsets typically carry out checks on the authenticity of server certificates received when starting a new connection and abort the connection when these checks fail. Installing the mitmproxy CA cert as a trusted certificate causes these checks to pass, except on the Huawei handset. On the Huawei handset each system app contains embedded server certificate SHA256 hashed and when starting an HTTPS

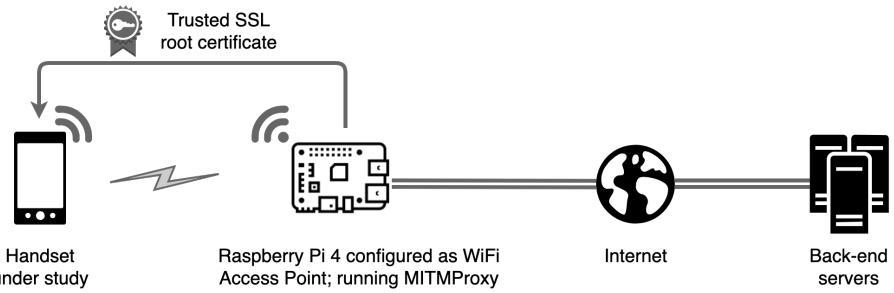


Fig 1. Measurement setup. Mobile handset configured to access the Internet using a WiFi access point hosted on a Raspberry Pi. A system certificate is installed on the phone to be able to decrypt outgoing traffic. The laptop pretends to any process running on the handset to be the destination server, creates a connection to the actual target, and relays requests and their replies between handset and server while logging the traffic.

connection checks that the certificate offered by the server matches one of these hashes. It is thus necessary to bypass these checks on each app individually and we used `Riru/edXposed`[4] for this.

In addition to the use of HTTPS encryption, the data sometimed has one or more additional inner layers of encryption. In more detail:
i) Xiaomi. A number of system apps encrypt data using AES/ECB. To decrypt this data we used `Frida` to intercept the entry points to the various functions used to carry out the AES encryption and record the key as it is passed in, allowing the data to then be decrypted.
ii) Realme. The app `com.heytap.mcs` encrypts data with AES/CBC. The 128-bit AES key and IV are hard-coded in the app and so can be extracted and used to decrypt the data sent. Messages sent by app `com.nearme.romupdate` are AES/CTR encrypted base-64 encoded JSON. A token that helps reconstruct the AES key using a custom encoding scheme is appended to the end of the base-64 message. Using this, the message can be decrypted.
iii) Huawei. We used `Riru/edXposed` to extract plaintext and/or encryption keys from the memory, allowing the data to be decrypted.

2.5 Categorising Identifiers

Identifiers in network connections were extracted by manual inspection. Known values such as the handset IMEI, hardware serial number, Google Advertising Id can often be readily identified in the transmitted data. Otherwise, we manually examined the decompiled app to find the code that writes each value and so establish how the value is generated. This is necessary, for example, to identify values that are hashes of device identifiers.

Following Android's guide on best practices for unique identifiers [2], we assign identifiers to one of the following persistence categories:

1. *Session-only.* The ID value changes every time the user restarts the app.
2. *Install-reset.* The ID value changes every time user uninstalls and reinstalls the app.
3. *FDR-reset.* The ID value changes every time the user factory-resets the device.

[4] `https://github.com/RikkaApps/Riru`, `https://github.com/ElderDrivers/EdXposed`.

4. *FDR-persistent*. The ID value survives factory reset.

and also to one of the following scope categories:

1. *Single-app*. The ID is used by only one app.
2. *Group of apps*. The ID is used by a group of apps.
3. *Device*. The ID is accessible to all apps installed on the device.

The persistence of each identifier is determined by observing whether its value changes after (i) a factory-reset of the handset (which will cause FDR-reset and Session-only identifiers to change value) and (ii) restart of the handset (which stops and restarts any running apps, causing Session-only identifiers to change value.). The *Install-reset* category is not relevant to Android system apps since they cannot be uninstalled. The scope of each identifier is determined by (i) observing the number of apps using the identifier, (ii) inspecting where the identifier is stored so as to evaluate its device-wide accessibility.

Best privacy practice is to use identifiers with the smallest scope and shortest persistence possible. For most purposes that means the use of identifiers with persistence no more than FDR-reset, the exception noted by [2] being the use of the handset IMEI and IMSI for use of carrier-related services (e.g. delivery of SMS messages over IP). Specific guidelines in [2] include: (i) avoid using hardware identifiers, (ii) use an advertising ID for advert targeting, measurement, remarketing, ad fraud detection, (iii) do not link the advertising ID to personally identifiable information, (iv) use a resettable ID for app analytics.

Note that these guidelines may change in future, especially those related to advertising and analytics. For example, in Android 12 there is now the option for users to disable use of the Google advertising identifier in apps (an all-zeros value is used instead).

2.6 Categorizing Network Connections

We assign network connections to one, or more, of the following categories:

1. *Firmware Updates*. Used to query the latest OS over-ther-air (OTA) update version.
2. *App Updates*. Used to query the latest version of individual apps or groups of apps (including system apps).
3. *Analytics*. Logging of device activity, including user interactions with apps (screens viewed etc).
4. *Services*. Provision of OEM system services. Examples: message push services (Xiaomi, Realme), configuration of device features/settings (Samsung, Realme), Caller ID and blocking (Samsung), Anti-virus checking (Huawei, Xiaomi), Swiftkey keyboard (Huawei).
5. *Advertising*. Download of adverts and tracking of impressions and user interactions with ads.

These categories were derived by analysis of the network connections that we measured. We note, however, that there is no widely agreed approach to categorising Android system network connections.

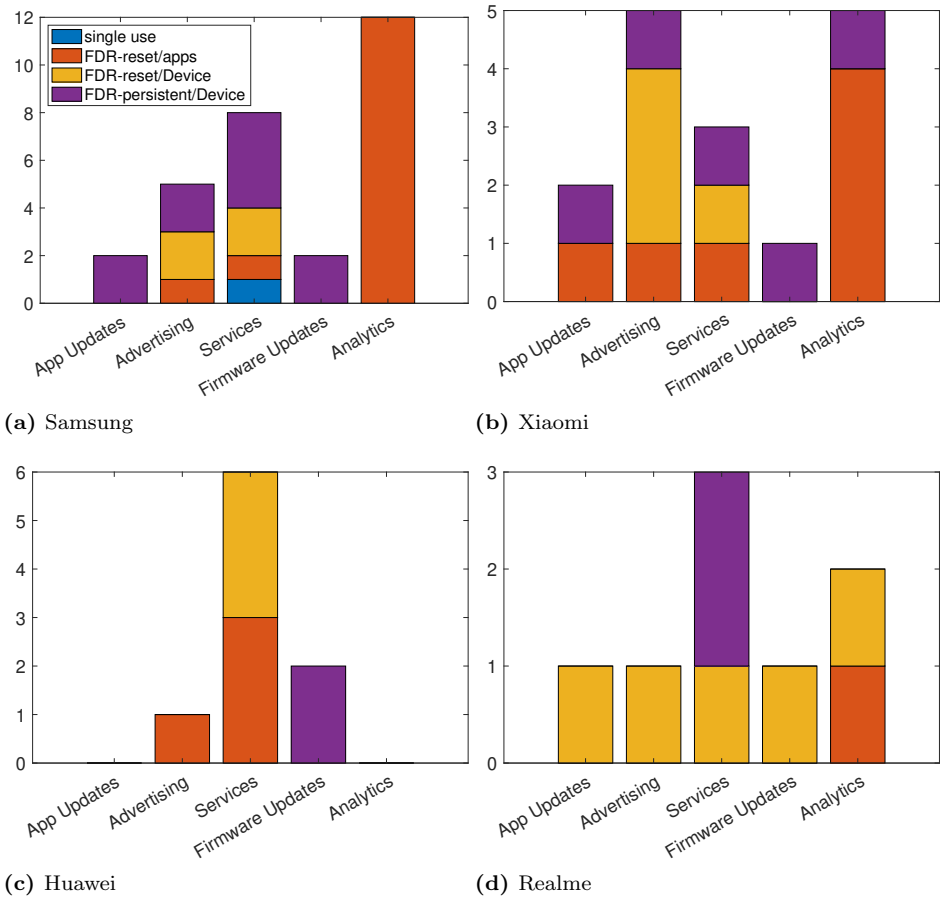
We observed that each URL is used for a single purpose, e.g. on the Realme handset all connections to `adx-f.ads.heytapmobile.com` relate to advertising, and so there is no need to distinguish between individual endpoints sharing the same URL.

2.7 Additional Material: Connection Data

Annotated measurements of network connections are available at `https://github.com/doug-leith/android_OEM_trafficdata`.

3 Results

3.1 Scope/Persistence Of Identifiers Used By Android OEMs



(a) Samsung **(b)** Xiaomi

(c) Huawei **(d)** Realme

Fig 2. Summary of measured identifier scope/persistence vs category of network connection and OEM. Network connections without identifiers are not shown.

Figure 2 summarises the identifier scope and persistence for each category of network connection. It can be seen that Huawei and Realme make less use of FDR-persistent identifiers than do Samsung and Xiaomi, as well as tending to have fewer network connections contained identifiers. In more detail, we make a number of observations:

1. With the exception of Xiaomi, checking for firmware updates make use of FDR-persistent identifiers with the greatest persistence and scope e.g. the hardware serial number, handset IMEI. We found this surprising since firmware updates are shared by all devices of the same model. Checking for updates therefore does not require any individual device identifiers, only device model and software version. Similarly, FDR-persistent identifiers can also transmitted by Samsung and Xiaomi when checking for app updates.

OEM	Service	FDR-persistent identifiers
Samsung	Knoxguard	Hashes of IMEI, hardware serial number
Samsung	Device config	Hardware serial number, Samsung Consumer ID
Xiaomi	Xiaomi Cloud	Xiaomi `cloudsp_fid`, security DeviceID
Realme	Warranty	IMEI
Realme	Device config	Heytap DeviceId

Table 1. FDR-persistent identifiers used in network connections associated with OEM system services.

2. With the notable exception of Xiaomi (discussed further below), resettable identifiers are used in analytics connections in accordance with good practice. Xiaomi sends the handset IMEI (which uniquely identifies the handset), sim IMSI (which uniquely identifies the SIM on the mobile network), a hash of the handset Wifi MAC address, and the security device ID to `tracking.intl.miui.com`. These identifiers are all FDR-persistent. This connection is used to record user interactions with the apps installed on the device (app screens viewed etc).

3. Huawei and Realme use resettable identifiers in advertising connections, in accordance with good practice. Samsung and Xiaomi, however, use a mix of resettable and FDR-persistent identifiers. Samsung sends the handset IMEI (an FDR-persistent identifier) in network connections to `sdk.pushmessage.samsung.com` that are associated with the Samsung Push Service which is used for marketing messages. Xiaomi sends a hash of the handset IMEI to `sdkconfig.ad.intl.xiaomi.com` used for advert configuration. Both seem avoidable.

4. All of the handsets use the Google Advertising ID in at least one of their network connections associated with advertising. This is an FDR-resettable identifier, in accordance with good practice, but serves to highlight the central role played by Google in Android advertising.

5. Various mixes of identifier types are used in network connections for OEM system services. With the exception of Huawei, all of the OEMs use FDR-persistent identifiers for at least one service, but resettable identifiers for other services, see Table 1. Recall that in our experiment setup the user opts out of OEM services when asked, and in any case makes no use of OEM services. All of these connections therefore ought to be avoidable.

3.2 Linking Identifiers To User's Real Identity

Our experimental protocol involves the user creating/logging in to an OEM user account on each handset. This is to allow observation of which device identifiers are linked to user personal information durinfg the sign-in process. Note that there was no OEM account login option on the Realme handset studied.

Table 2 reports the FDR-persistent identifiers sent alongside user personal identifiers (email etc) during sign-in. It can be seen that for all three OEMs which have user sign-in personal identifiers are sent alongside one or more FDR-persistent identifiers, thereby trivially linking these together.

Table 3 lists the FDR-persistent identifiers which are sent together in other connections (i.e. not during sign-in) and so trivially linkable. It can be seen from Table 3 that the FDR-persistent identifiers linked to user personal identifiers in Table 2 are also sent alongside most of the other FDR-persistent identifiers on each handset,

OEM	Personal identifiers	FDR-persistent identifiers
Samsung	Email, phone number	IMEI
Xiaomi	Email	Xiaomi `cloudsp_fid`, security DeviceID
Huawei	Email, phone number	Hardware serial number

Table 2. FDR-persistent identifiers linked to personal identifiers during sign-in to OEM account on handset. There was no OEM account login option on the Realme handset studied.

OEM	FDR-persistent identifiers sent together	Google Ad ID sent
Samsung	IMEI, Hardware serial number, Samsung Consumer ID	no
Xiaomi	IMEI, IMSI, hash of WiFi MAC address, Xiaomi `cloudsp_fid`, security DeviceID	yes
Huawei	Hardware serial number, device cert	no
Realme	IMEI	no

Table 3. FDR-persistent identifiers sent together in same network connection (and so trivially linkable), plus whether Google Ad ID is sent in the same connection (so making it trivially linkable to the FDR-persistent identifiers).

	Connection Category				
	Firmware Updates	App Updates	Services	Advertising	Analytics
Samsung	x	x	x	x	-
Xiaomi	x	x	x	x	x
Huawei	x	-	x	x	-
Realme	x	x	x	x	x

Table 4. Connection categories sending device hardware/system configuration data.

	Connection Category		
	App Updates	Services	Analytics
Samsung	-	Device Settings	-
Xiaomi	x	-	x
Huawei	-	-	-
Realme	x	Oppo Push	-

Table 5. Connection categories sending list of installed apps.

thereby potentially linking user personal identifiers to all of these handset FDR-persistent identifiers.

In addition, it can be seen from Table 3 that on the Xiaomi handset the Google Ad ID is sent alongside the FDR-persistent identifiers, and so can be trivially linked to these and therefore also to user personal identifiers.

3.3 What Sort Of Data Is Being Collected?

The data that we observe being sent from handsets can be categorized as: (i) device configuration data, (ii) details of apps installed and (iii) event logging data/telemetry.

3.3.1 Device Configuration Data

Sharing device hardware/system configuration data such as the device model, screen size, operating system version, radio version generally carries little privacy risk since these are common to many devices (e.g. all devices of the same model). Such data is needed when checking for software updates and selecting the right version of an app to install.

Table 4 summarises the collection of device configuration data observed in our measurements. It can be seen that, as expected, all of the OEMs transmit this information when checking for firmware and app updates. This data is also transmitted by one or more services of every OEM (although as noted above in our experiment setup the user opts out of OEM services when asked and makes no use of OEM services). All of the OEMs transmit device configuration data to their advertising endpoints, while Xiaomi and Realme also transmit this data in their analytics connections.

3.3.2 List Of Installed Apps

This list of apps installed on a handset is potentially sensitive information since it can reveal traits of the handset user. For example a muslim prayer app, a period tracking app, a gay dating app, a mental health app can, respectively, reveal the user's religion, gender, sexual preference and mental health status. The newspaper apps installed can reveal pollitical preferences. The set of apps installed is also more likely to be unique to one handset, or a small number of handsets, and so can act as a device fingerprint (especially when combined with device hardware/system configuration data). See, for example, [12, 13, 14, 15, 16] for recent analyses of such privacy risks. We note that in light of such concerns, Google recently categorised the list of apps installed as "personal and sensitive user data" and introduced restrictions on Play Store apps collection of this data[5], but such restrictions do not apply to system apps since these are not installed via the Google Play store.

Table 5 summarises the collection of the list of installed apps observed in our measurements. It can be seen that, with the exception of Huawei, all of the OEMs transmit this data. Xiaomi and Realme transmit this data to app update endpoints and in both cases only FDR-reset identifiers are sent in these connections, which is reasonable practice although it would enhance privacy to use less persistent identifiers e.g. session-based identifiers. Samsung and Realme transmit the list of installed apps to endpoints associated with OEM services. Xiaomi transmit the list of installed apps to an analytics endpoint, along with FDR-persistent identifiers, a poor privacy practice.

[5] https://support.google.com/googleplay/android-developer/answer/10446026

	Telemetry Data Collected
Samsung	SIM insertion, making/receiving phone call, details of user interactions with selected system apps[6]
Xiaomi	Details of user interactions with all apps (not just selected system apps)
Huawei	Events and settings associated with `com.huawei.himovie.overseas` system app, logs app details when a new app is installed. When the keyboard is used within an app, the app name and details of user interactions are logged.
Realme	Events associated with `com.heytap.mcs` system app.

Table 6. Summary of telemetry data sent in network connections.

3.3.3 Event Logging Data/Telemetry

All of the OEMs studied log user interactions with system apps, see Table 6. Note that this occurs despite the user opting out of diagnostics/analytics collection on the handsets during onboarding following factory reset. While some logging of app usage is reasonable, e.g. to allow early detection of app performance issues, ongoing logging of user activity can quickly become intrusive and a serious privacy concern.

Perhaps the most striking logging of user interactions is by Xiaomi, which logs user interactions with every app. Interactions logged include every opening and closing of app windows plus the duration a window is open. FDR-persistent identifiers and the Google Ad ID are sent alongside this data. Xiaomi system apps `com.miui.msa.global`, `com.xiaomi.discover`, `com.android.thememanager` also log user interactions using Google Analytics.

It can be seen from Table 6 that several Samsung system apps use Google Analytics to log user interaction events, including windows/activities viewed plus duration and timestamp. The caller ID/call blocking service logs the making/receiving of phone calls and the Google Ad ID is sent alongside this data.

Huwaei log events (e.g. app start up) associated with system app `com.huawei.himovie.overseas` and when a new app is installed the app details are sent to a scanning service. When the keyboard is used within an app, the app name, number of characters entered and an event timestamp are sent to Microsoft e.g. use of searchbar, contacts and messaging apps is logged. The Google Ad ID is sent alongside this data.

Realme log events (e.g. app start up) related to system app `com.heytap.mcs`.

4 Discussion

4.1 Limitations

The analysis here is based on measurement data. The data itself is not subject to noise or measurement error since it is digital in nature. However, there are two main potential sources of error in the analysis.

The first is that the categorization of network connections may be inaccurate. OEMs provide no public documentation on purpose of the connections made by their handsets and so the categorization is based on our analysis of the content of network connections and of the apps that generated them. As partial validation, we have contacted all of the OEMs studied and presented our measurement data to them so we believe there are no significant errors in the categorization used.

A second potential source of error is that the behaviour of our rooted handsets may differ from that of unrooted handsets. As partial validation we also collected traffic measurements from unrooted handsets and compared these with the measurements on our rooted handsets. On unrooted phones the connections remain encrypted but the SNI can be extracted from the TLS handshake and used to discover the server name associated with each connection. This highlighted that on the Samsung handset connections to `diagmon-serviceapi.samsungdm.com` are present on unrooted phones but not on rooted phone - further investigation established that the DiagmonAgent app relies on trusted storage, which is disabled by Samsung on rooted handsets. However, we did not see connections on the rooted handsets which were not also present on the unrooted handsets, and so the analysis here is likely conservative in the sense that it applies to the subset of connections present on both rooted and unrooted handsets.

We note also that several of the OEMs use third-party apps to provide system services. Namely, Huawei use the Microsoft Swiftkey keyboard and the Avast and 360 Safe antivirus checkers, Samsung uses the Hiya caller ID/call blocking service, Realme partners with Heytapi. Since these services are installed by the OEM we includes the associated network connections in our analysis. Excluding these network connections results in only small changes in the analysis.

All of the handsets studied have Google Play Services pre-installed and transmit data to Google servers. Since our focus here is on OEM practices we do not include these Google connections in the present analysis. For Google-specific measurement studies see e.g. [9, 10].

We study the European models of handsets from Samsung, Xiaomi, Huawei and Realme and use the handsets within Europe. The data collection behaviour on models targeted at other regions may differ.

4.2 Privacy Scoring

Our analysis can be used as the basis for ranking the privacy of the handsets studied. Various metrics might be used for generating a score to use for ranking, but one approach is to rank handsets based on how many changes are needed to bring the use of identifiers into line with current best practice. Use of long-lived hardware identifiers is not warranted for the data transmitted in our experiments and so we score 1 for each connection using such identifiers. Device-wide identifiers are also not warranted, and should be replaced by single-app or group-of-apps identifiers. We score 0.5 for for each connection using device-wide FDR-reset identifiers. Using this approach we arrive at the following scores: Samsung $8 + 4 \times 0.5 = 10$, Xiaomi $5 + 4 \times 0.5 = 7$, Huawei $2 + 3 \times 0.5 = 4.5$, Realme $2 + 5 \times 0.5 = 4.5$ and a ranking with Huawei/Realme most private, followed by Xiaomi and then Samsung.

This scoring system is certainly imperfect, e.g. it ignores the intrusive telemetry collection by Xiaomi (logging user interactions with all apps) and the failure by Xiaomi to keep the Google Ad Id separate from long-lived hardware identifiers, but it does highlight the clear difference between the use of identifiers by Huawei/Realme and Samsung/Xiaomi. It also serves to make the point that the data generated by the analysis here has the potential for use in privacy scoring, although we leave the development of more refined scoring approaches to future work.

5 Conclusions

We present the first in-depth measurement study of the data privacy practices of the proprietary variants of the Android OS produced by Samsung, Xiaomi, Huawei and

Realme and address two questions: how are identifiers used in network connections and what types of data are transmitted. We find that all of the OEMs make undue use of long-lived hardware identifiers such as the hardware serial number, handset IMEI and so fail to follow best privacy practice. Hardware identifiers are also linked to the handset user's real identity when they sign in to an OEM account on the handset. All of the OEMs collect the list of apps installed in a handset. This is a privacy concern since the list of installed apps can be used to profile user traits and preferences. All of the OEMs collect analytics/telemetry data.

Acknowledgements

This work was supported by Science Foundation Ireland (SFI) under grant 16/IA/4610.

References

[1] IDC. Worldwide Quarterly Mobile Phone Tracker; 2021.

[2] Best practices for unique identifiers, Android Developers Guide; Accessed 20th July 2022. Available from: `https://developer.android.com/training/articles/user-data-ids`.

[3] Jin H, Liu M, Dodhia K, Li Y, Srivastava G, Fredrikson M, et al. Why Are They Collecting My Data? Inferring the Purposes of Network Traffic in Mobile Apps. Proc ACM Interact Mob Wearable Ubiquitous Technol. 2018;2(4).

[4] Ren J, Lindorfer M, Dubois DJ, Rao A, Choffnes D, Vallina-Rodriguez N. Bug Fixes, Improvements,... and Privacy Leaks. In: Network and Distributed System Security Symposium (NDSS); 2018.

[5] Razaghpanah A, Nithyanand R, Vallina-Rodriguez N, Sundaresan S. Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking Ecosystem. In: Network and Distributed System Security Symposium (NDSS); 2018.

[6] Reardon J, Feal Á, Wijesekera P, On AEB, Vallina-Rodriguez N, Egelman S. 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System. In: 28th USENIX Security Symposium (USENIX Security 19). Santa Clara, CA: USENIX Association; 2019. p. 603–620. Available from: `https://www.usenix.org/conference/usenixsecurity19/presentation/reardon`.

[7] Wang Z, Li Z, Xue M, Tyson G. Exploring the Eastern Frontier: A First Look at Mobile App Tracking in China. In: Sperotto A, Dainotti A, Stiller B, editors. Passive and Active Measurement; 2020.

[8] Leith DJ. Web Browser Privacy: What Do Browsers Say When They Phone Home? IEEE Access. 2021;doi:https://doi.org/10.1109/ACCESS.2021.3065243.

[9] Leith DJ, Farrell S. Contact Tracing App Privacy: What Data Is Shared By Europe's GAEN Contact Tracing Apps. In: Proc IEEE INFOCOM; 2021.

[10] Leith DJ. Mobile Handset Privacy: Measuring The Data iOS and Android Send to Apple And Google. In: Proc Securecomm; 2021.

[11] Cortesi A, Hils M, Kriechbaumer T, contributors. mitmproxy: A free and open source interactive HTTPS proxy (v5.01); 2020. Available from: https://mitmproxy.org/.

[12] Seneviratne S, Seneviratne A, Mohapatra P, Mahanti A. Your Installed Apps Reveal Your Gender and More! SIGMOBILE Mob Comput Commun Rev. 2015;18(3):55–61. doi:10.1145/2721896.2721908.

[13] Zhao S, Pan G, Zhao Y, Tao J, Chen J, Li S, et al. Mining User Attributes Using Large-Scale APP Lists of Smartphones. IEEE Systems Journal. 2017;11(1):315–323. doi:10.1109/JSYST.2015.2431323.

[14] Frey RM, Xu R, Ilic A. Mobile app adoption in different life stages: An empirical analysis. Pervasive and Mobile Computing. 2017;40:512–527. doi:https://doi.org/10.1016/j.pmcj.2017.01.006.

[15] Pham A, Dacosta I, Losiouk E, Stephan J, Huguenin K, Hubaux JP. HideMyApp: Hiding the Presence of Sensitive Apps on Android. In: 28th USENIX Security Symposium (USENIX Security 19). Santa Clara, CA: USENIX Association; 2019. p. 711–728. Available from: https://www.usenix.org/conference/usenixsecurity19/presentation/pham.

[16] Scoccia GL, Kanj I, Malavolta I, Razavi K. Leave My Apps Alone! A Study on How Android Developers Access Installed Apps on User's Device. MOBILESoft '20; 2020. p. 38–49. Available from: https://doi.org/10.1145/3387905.3388594.

S1 Summary Of Measurement Data

Identifier	Persistence	Scope
hardware serial number	FDR-persistent	device
IMEI	FDR-persistent	device
Samsung Consumer ID	FDR-persistent	device
secure settings android_id	FDR-reset	device
Firebase ID	FDR-reset	group of apps
Firebase tokens	FDR-reset	group of apps
Google Ad ID	FDR-reset	device
*-odc.samsungapps.com transactionId	single use	single app
*-odc.samsungapps.com sessionId	session only	single app
*-odc.samsungapps.com logId (or stduk)	FDR-persistent	group of apps
*-odc.samsungapps.com cookie JSESSIONID	session only	single app
gos-api.gos-gsp.io uuid	FDR-reset	single app
capi.samsungcloud.com device_id	FDR-reset	device
capi.samsungcloud.com access_token	FDR-reset	single app
us-api.mcsvc.samsung.com x-smcs-did	FDR-reset	device
us-rd.mcsvc.samsung.com sid	FDR-reset	single app
us-rd.mcsvc.samsung.com ri	FDR-reset	single app
samsung-directory.edge.hiyaapi.com X-Hiya-Request-Id	single use	single app
samsung-directory.edge.hiyaapi.com x-seq_id	single use	single app
samsung-directory.edge.hiyaapi.com X-Hiya-Installation-User-ID (also called hin)	FDR-reset	single app
samsung-directory.edge.hiyaapi.com X-Hiya-Device-User-ID	FDR-reset	single app
samsung-directory.edge.hiyaapi.com Authorization bearer	FDR-reset	single app
oath nonce/token	session only	single app
x-samsung-trace-id	single use	single app

Endpoint	Category	Identifiers
hub-odc.samsungapps.com	App Updates	transactionId, sessionId, logId
ie-odc.samsungapps.com	App Updates	IMEI, transactionId, sessionId, logId, cookie JSESSIONID
vas.samsungapps.com	App Updates	
gos-api.gos-gsp.io	Services/App Config	uuid, x-samsung-trace-id
api.omc.samsungdm.com	Services/Device Settings	hardware serial number, Samsung Consumer ID, Firebase ID/token (for com.samsung.android.app.omcagent)
dir-apis.samsungdm.com	Advertising	IMEI, hardware serial number, Samsung Consumer ID
api.gras.samsungdm.com	Services/Device Settings	hardware serial number, Firebase ID/token (for com.samsung.android.sdm.config)
fota-apis.samsungdm.com	Services/Device Settings	oath nonce
capi.samsungcloud.com	Services/Device Settings	device_id, access_token
gslb.secb2b.com	Services/Knoxguard	hash of hardware serial number
eu-kaf.samsungknox.com	Services/Knoxguard	hashes of IMEIs, hash of hardware serial number
eu-segd-api.secb2b.com	Services/Knoxguard	hashes of IMEIs, hash of hardware serial number
www.ospserver.net	Firmware Updates	IMEI, Samsung Consumer ID, hardware serial number, Firebase ID/token (for app wssyncmldm - an update monitor app), oath nonce
fota-cloud-dn.ospserver.net	Firmware Updates	-
dms.ospserver.net	Firmware Updates	IMEI, sid
sspapi-prd.samsungrs.com	Advertising	Google AID
sdk.pushmessage.samsung.com	Advertising	IMEI, p_deviceId/smpid value returned by server after request sending IMEI, Firebase ID/token (for app com.sec.spp.push "Samsung Push Service")
us-api.mcsvc.samsung.com	Advertising	x-smcs-did
us-cdn-gpp.mcsvc.samsung.com	Advertising	-
us-rd.mcsvc.samsung.com	Advertising	sid, ri, plus others: ii, pg, pi, li, cp downloaded with ad and all are FDR-persistent, also ri, sid downloaded with ad and these are FDR-reset.
samsung-directory.edge.hiyaapi.com	Services/CallerIdBlocking	Google Ad ID (X-Hiya-Advertising-ID), X-Hiya-Request-Id, x-seq id, X-Hiya-Installation-User-ID, X-Hiya-Device-User-ID, Authorization bearer, eventId
Firebase	Analytics	GAID, `com.sec.android.app.samsungapps`, `com.samsung.android.app.simplesharing`, `com.samsung.android.authfw`, `com.samsung.android.bixby.agent`, `com.samsung.android.kgclient`, `com.samsung.android.mobileservice`, `com.samsung.android.rubin.app`, , `com.samsung.android.themestore`, `com.sec.android.app.billing`, `com.samsung.android.game.gamehome` Firebase ids

Table 7. Samsung

Identifier	Persistence	Scope
IMEI	FDR-persistent	device
IMSI	FDR-persistent	device
Security DeviceID	FDR-persistent	device
Xiaomi VAID	FDR-reset	device
Google Ad Id/GAID	FDR-reset	device
android_id	FDR-reset	device
Xiaomi AAID	FDR-reset	device
Xiaomi devID	FDR-reset	device
cloudsp_fid	FDR-persistent	device
Wifi MAC address	FDR-persistent	device
data.mistat.intl.xiaomi.com sid	session only	single app
data.mistat.intl.xiaomi.com AES key	session only	single app
mcc.intl.inf.miui.com uid	FDR-reset	single app
global.market.xiaomi.com guid	FDR-reset	single app
Firebase ID	FDR-reset	group of apps
Firebase tokens	FDR-reset	single app
au.ff.avast.sec.miui.com uuid	FDR-reset	single app

Endpoint	Category	Identifiers
tracking.intl.miui.com	Analytics	IMEIs, IMSI, VAID, GAID,cloudsp_fid, hash of Wifi MAC address, instance id, CPUID, Security DeviceID
api.sec.intl.miui.com	App Updates	hash of android id in secure settings, hash of Security DeviceID
api.ad.intl.xiaomi.com	Advertising	GAID
sdkconfig.ad.intl.xiaomi.com	Advertising	hash of IMEI
privacy.api.intl.miui.com	Advertising	GAID
update.intl.miui.com	Firmware Updates	hash of IMEI, hash of Security DeviceID
fr.register.xmpush.global.xiaomi.com	Services/Xiaomi Push	GAID, VAID, AAID, devId
find.api.micloud.xiaomi.net	Services/Xiaomi Cloud	cloudsp_fid, Security DeviceID
data.mistat.intl.xiaomi.com	Advertising	sid, AES key, android_id
mcc.intl.inf.miui.com uid	Advertising	uid
global.market.xiaomi.com	App Updates	guid, instance_id (Firebase ID of app com.xiaomi.discover)
au.ff.avast.sec.miui.com	Services/Antivirus	uuid
Firebase	Analytics	GAID, com.google.android.apps.messages, com.miui.msa.global, com.xiaomi.discover, com.mi.android.globalFileexplorer Firebase ids

Table 8. Xiaomi

Identifier	Persistence	Scope
hardware serial number	FDR-persistent	device
device cert	FDR-persistent	device
huawei udid	-	
servicesupport.hicloud.com userId	FDR-reset	single app
360safe.com udid	FDR-reset	single app
android id	FDR-reset	device
avast.com ABUID	FDR-reset	single app
avast.com X-AVAST-KeyId	FDR-reset	single app
Google Ad ID /GAID	FDR-reset	device
pebed.dmevent.net instance_uuid	FDR-reset	single app
pebed.dmevent.net id/cookie	FDR-reset	single app
installid	FDR-reset	single app

Endpoint	Category	Identifiers
query.hicloud.com	Firmware Updates	hardware serial number, device cert, udid
configserver-dre.platform.hicloud.com	Firmware Updates	hardware serial number
servicesupport.hicloud.com	Services/UI Theme	userId
shepherd.sb.avast.com	Services/Antivirus	hash of android id, ABUID
apkrep.ff.avast.com	Services/Antivirus	hash of android id, ABUID, X-AVAST-KeyId
mvconf.cloud.360safe.com	Services/Antivirus	uid
mclean.cloud.360safe.com	Services/Antivirus	uid
pebed.dmevent.net	Advertising	GAID, instance_uuid, id/cookie
telemetry.api.swiftkey.com	Services/Keyboard	installid, GAID
in.appcenter.ms	Services/Keyboard	installid
Firebase	Analytics	GAID, com.google.android.apps.messages Firebase id

Table 9. Huawei

Identifier	Persistence	Scope
IMEI	FDR-persistent	device
Google Ad ID	FDR-reset	device
OUID	FDR-reset	device
DUID/VAID	FDR-reset	device
registrationId	FDR-reset	device
shorteuex.push.heytapmobile.com device_id	FDR-reset	single app
httpdns-euex-push.heytapmobile.com deviceID	FDR-persistent	device
guid	FDR-reset	device

Endpoint	Category	Identifiers
ifrus-eu.coloros.com	Firmware Updates	guid, registrationId
ifota-eu.realmemobile.com	Firmware Updates	guid, registrationId
icosa-eu.coloros.com	AppUpdates	guid
esa-reg-eup.myoppo.com	Services/Warranty	IMEI, guid
httpdns-euex-push.heytapmobile.com	Services/Config	deviceId, DUID/VAID
adx-f.ads.heytapmobile.com	Advertising	GAID, OUID
shorteuex.push.heytapmobile.com	Services/Oppo Push	DUID/VAID, OUID, device_id, Firebase id and token (for app com.heytap.mcs)
dceuex.push.heytapmobile.com	Analytics	DUID/VAID
Firebase	Analytics	GAID, com.heytap.mcs Firebase id

Table 10. Realme

Endpoint	Device Config	List of Apps	Telemetry
Samsung			
gos-api.gos-gsp.io		x	
www.ospserver.net	x		
dms.ospserver.net	x		
sdk.pushmessage.samsung.com	x		
sspapi-prd.samsungrs.com	x		
us-api.mcsvc.samsung.com	x		
samsung-directory.edge.hiyaapi.com		x	x
api.omc.samsungdm.com		x	
Xiaomi			
tracking.intl.miui.com	x	x	x
global.market.xiaomi.com	x	x	
api.sec.intl.miui.com	x		
api.ad.intl.xiaomi.com	x		
fr.register.xmpush.global.xiaomi.com	x		
sdkconfig.ad.intl.xiaomi.com	x		
mcc.intl.inf.miui.com	x		
Huawei			
query.hicloud.com	x		
configserver-dre.platform.hicloud.com	x		
servicesupport.hicloud.com	x		
shepherd.sb.avast.com	x		
apkrep.ff.avast.com	x		x
mvconf.cloud.360safe.com/safeupdate	x		
telemetry.api.swiftkey.com	x		x
in.appcenter.ms	x		x
bibo.api.swiftkey.com	x		
pebed.dm-event.net	x		x
Realme			
ifrus-eu.coloros.com	x		
ifota-eu.realmemobile.com	x		
icosa-eu.coloros.com		x	
adx-f.ads.heytapmobile.com	x		
shorteuex.push.heytapmobile.com	x	x	
dceuex.push.heytapmobile.com	x		x

Table 11. Type of data observed transmitted to each endpoint.