Cookies, Identifiers and Other Data That Google Silently Stores on Android Handsets

D.J.Leith
Trinity College Dublin, Ireland
Feb 13th 2025

***Abstract*—We report on the results of a measurement study investigating the cookies, identifiers and other data stored on Android handsets by Google Play Services, the Google Play store and other pre-installed Google apps. We find that multiple cookies and identifiers are sent by Google servers and stored on the handset, even when no Google apps have ever been opened by the user. This includes advertising analytics/tracking cookies, links and device identifiers. No consent is sought for storing any of this data and there is no opt out. To the best of our knowledge, this study is the first to cast light on the cookies etc stored by pre-installed Google apps.**

I. Introduction

This paper presents the results of a measurement study on the cookies, identifiers and other data sent by Google servers and stored on Android handsets by pre-installed Google apps, including the Google Play Services and Google Play store apps. To the best of our knowledge this is the first such study and the first time that the data stored by these apps has been publicly documented.

In summary, we find that:

1) Advertising analytics cookies, links to track clicks/views of adverts, and device identifiers associated with advertising are downloaded and stored on the handset.
2) Tracking cookies are downloaded and stored by the Google Play store app, and then transmitted to Google servers alongside analytics data reporting user interactions with the Play store app (searches, page views etc).
3) The Google Android ID is sent by Google servers and stored to the handset by Google Play Services. Previous studies have shown this identifier is widely used in Google Play Services and Google Play store transmissions to Google servers (see, e.g. [1], [2]) and acts as a persistent device and user identifier.
4) Analytics cookies used for A/B testing of changes to Google apps are downloaded and stored on the handset by Google Play Services, and then transmitted alongside app telemetry data to Google servers.
5) Multiple other cookies and identifiers which can act to uniquely identify the handset and/or user are also downloaded and stored on the handset.

No consent is sought or given for storing any of these cookies and other data, the purposes are not stated and there is no opt out from this data storage. Most of this data is stored even when the device is idle following a factory reset and no Google apps have ever been opened by the user i.e. they are not set in response to services explicitly requested by the user.

The work reported here is inevitably limited in nature. A great many different Google connections are made even when a handset is lying idle, and we have analysed only a subset of them (we focussed on those which tend to be frequently recurring). Nevertheless, our measurements are already enough to raise concerns that Google is violating EU data privacy regulations, which generally require explicit and informed user consent before data can be stored on a handset.

While this study focusses on Android handsets, this work naturally raises questions regarding the data stored by pre-installed Apple apps on iPhones. We leave that to future work, but clearly there is an urgent need for such a study.

A. EU Data Privacy Regulations

We report on a technical study here, not a legal one, and we are not legally qualified. Nevertheless, as already noted, the data storage that we observe by Google raises obvious questions regarding the EU e-Privacy Directive and perhaps also the GDPR data protection regulations (the measurements were all carried out within Ireland using handsets purchased in Ireland and so it is Irish/European data regulations that apply).

1) EU e-Privacy Directive: The e-Privacy Directive was introduced in 2002[1] and amended in 2009[2]. Article 5(3) of the Directive is sometimes referred to as the "cookie law". It "recognises that the devices of users of electronic communications networks and any information stored on their devices are part of their 'private sphere' and that they require protection"[3] and was implemented in Irish law by Statutory Instrument No. 336 of 2011[4]. Its Article 5(3) restricts storage of data on a handset, stating that "A person shall not use an electronic communications network to store information, or to gain access to information already stored in the terminal equipment of a subscriber or user, unless (a) the subscriber or user has given his or her consent to that use, and (b) the subscriber or user has been provided with clear and comprehensive information in accordance with the Data Protection Acts which (i) is both prominently displayed and easily accessible, and (ii) includes, without limitation, the purposes of the processing of

Contact email: doug.leith@tcd.ie

[1]https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:en:HTML

[2]https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:32009L0136

[3]https://www.dataprotection.ie/en/faqs/cookies/what-law-use-cookies, accessed Jan 7th 2025.

[4]https://www.dataprotection.ie/en/who-we-are/data-protection-legislation, accessed Jan 7th 2025.

the information" [3]. Exceptions are allowed when the storage is "for the sole purpose of carrying out the transmission of a communication over an electronic communications network" or "strictly necessary in order to provide an information society service explicitly requested by the subscriber or user".

It is important to note that this regulation applies to *any* information storage on a handset, not just cookies, trackers or personally identifying/sensitive data, and that distribution of tracking links to a handset constitutes storage [4]. European Data Protection Board (EDPB) guidelines [4] also clarify that (amongst other things): "terminal equipment" includes smartphones and "storage" does not depend on the medium and includes both RAM and SSDs nor is there any upper/lower limit on the length of time that information needs to be stored in order for it to count. Studies by regulators have highlighted that the "strictly necessary" exemption should not be leniently interpreted [5], and the EDPB clarifies [6] that for this to apply (i) the user must have taken a positive action to request a service and (ii) the service does not work if the storage is disabled.

2) GDPR: Roughly speaking, there are three main basis under GDPR for data processing[5]: (i) the data is anonymised, i.e. cannot reasonably be linked to an individual person, and so is not personal data, (ii) with consent for a defined purpose and (iii) for the legitimate interests of Google. Data processing includes data storage and personal data includes pseudonymous data if it is easy to identify someone from it[5].

B. Data Storage By Google Apps

Table I summarises some of the data the we observed being stored on a mobile handset by Google apps. Sometimes we can reasonably infer the purpose by analysis of the data stored and of the relevant Google app, in which case we have noted this in the table. The table also gives the app, or apps, which store the data and whether the data is unique to the individual handset and/or user and so acts as a device/user identifer.

1) No Consent: Specific consent is neither sought nor given for the data stored by the Google apps that we observe, and there is no opt out.

2) Purpose Not Stated: In almost all cases there is no Google public documentation stating the purpose of the data storage. The exceptions are the DSID and NID cookies, although the public documentation relates to web browsers not Google Android apps and we found the stated purposes vague and unhelpful.

3) Not Strictly Necessary: Hardly any of the data in Table I is strictly necessary for a service explicitly requested by the handset user. The only exceptions are the Auth/Bearer tokens used to authorize usage of Google services following login to the user's Google account. However, even in this case we observe Auth/Bearer tokens being stored for (i) apps which the user did not request and has never used, including Gmail, Google Docs, Google Videos, Google Search, Google Calendar and Google Chrome and (ii) Google Play Services advertising (`DSID`), telemetry (`Clearcut`) and A/B testing (`Experiments`) subsystems which the user has never requested.

[5]E.g. see https://gdpr.eu/what-is-gdpr/, accessed Jan 7th 2025.

Name	Type	App	Device/User Identifer
DSID	Advertising Analytics Cookie	Google Play Services, Firebase Analytics	✓
AdAttest Token	Advertising Device ID	Google Play Services	✓
Ad URLs	Advert View/Click Tracking	Google Play store	✓
ServerLogs	Analytics Cookie, incl. adverts	Google Play store	✓
Google Android ID	Persistent Device Identifier	Google Play Services, Google Play store	✓
NID	Cookie	Google Play Services, Google Search, Gmail, Google Messages, Google Files	✓
deviceKey	Persistent Device Identifier	Google Play Services	✓
Auth/Bearer Tokens	User & Device Identifier	Google Play Services	✓
auth tokens	Device Identifier	Google Firebase	✓
GCM ID	Device Identifier	Google Play Services	✓
Widevine Key	Device Identifier	Google Play Services, Google Chrome	✓
Experiment Tokens	A/B Testing Analytics Cookie	Google Play Services	?
Server Tokens	A/B Testing Cookie	Google Play Services	?
App settings	A/B Testing	Google Play Services	?

TABLE I: Some of the data stored on handset by Google Apps.

4) Lack of Anonymity: As noted in Table I, much of the data stored can act a device and/or user identifier and therefore likely counts as personal information. GDPR therefore likely applies as well as the e-Privacy Directive.

C. Response From Google

The Google Play Services and Google Play store apps studied here are in active use by hundreds of millions of people. We informed Google of our findings, and delayed publication to allow them to respond. They gave a brief response, stating that they would not comment on the legal aspects (they were not asked to comment on these). They did not point out any errors or mis-statements (which they *were* asked to comment on). They did not respond to our question about whether they planned to make any changes to the cookies etc stored by their software.

D. Mitigations

Users currently have little control over the data that apps store on an Android handset. It is possible to use the Settings app to clear the data stored by an app. This deletes all the data in the app's data folder, and is akin to re-installing the app. There is no ability to selectively delete cookies etc, unlike within a web browser, and no ability to prevent their storage in the first place. The main mitigations are therefore to (i) disable Google Play Services (not all handsets allow this), and (ii) disable the Google Play store app. However, these are not practical options for most users since third-party apps may depend on Google Play Services and the Google Play store is used by most people as the primary way to install third-party apps.

E. Related Work

Since the introduction the EU e-Privacy Directive there has been much interest in the presence/absence, validity and effectiveness of cookie consent notices e.g. see the measurement

studies [7], [8], [9] and investigations by the Irish Data Protection Commissioner [5] and European Data Protection Board [10]. There have also been several large-scale measurement studies documenting the cookies set by web sites, e.g. see [11], [12], [13]. In contrast, there have been few studies on the cookies stored on Android handsets [14], and to the best of our knowledge none at all on the cookies stored by Google apps and the Google Play Services app in particular. Previous measurement studies on Android have, instead, mainly focussed on analysing web traffic to detect transmissions to tracking services or detect transmission of sensitive personal identifiers e.g. see [15], [16], [17], [18], [19], [1], [20], [2], [21].

II. Experimental Setup

A. Hardware and Software Used

Google Pixel 7 running Android 14 (build AP4A.241205.013/12621605, the latest available) with Google Play Services ver. 244738035 and Google Play store app version 44.0.28-31 [0] [PR] 705297086; rooted using Magisk v28.1. Frida server v16.5.7, Mitmdump v11.0.0.dev.

B. Measurement Data & Additional Material

An Additional Material document, the full measurement data and the software tools we developed to analyse it are available anonymously at https://github.com/doug-leith/cookies-etc-stored-by-google-android.

C. Decrypting HTTPS Connections

We route handset traffic via a WiFi access point (AP) that we control, configure this AP to use `mitmdump` as a proxy [22] and adjust the firewall settings to redirect all WiFi HTTP/HTTPS traffic (other traffic is blocked) 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. System processes typically carry out checks on the authenticity of server certificates received when starting a new connection and abort the connection when these checks fail. For Google apps and services, installing the mitmproxy CA cert as a trusted certificate causes these checks to pass. We installed the mitmproxy CA cert as a user cert and used the Magisk trustusercerts module[6] to promote this to a system cert.

D. Analysis Tools Developed

1) Decoding & Tracing Received Data: Many of the responses from Google servers are encoded as protobufs. Protobufs can be decoded without knowledge of the message schema using the Google Protobuf compiler with the `--decode_raw` option. However, this means that the interpretation of values is missing and there is also sometimes ambiguity as to interpretation of the value types. Since there is no public documentation, to determine the meaning of observed values we (i) decompile the Google app that receives the data, (ii) identify the protobuf schema used to decode the network response within the decompiled code (this step is non-trivial since a Google app typically contains thousands of distinct protobufs[7]) and then (iii) trace back within the code to determine how the value of each entry in the protobuf schema is used, whether and where it is stored on the handset disk, and whether it is transmitted in later connections. This analysis often reveals informative error and log message strings which can help understand the data. When the data is named in such message strings, we use that Google name here.

This is a fairly laborious process that is difficult to fully automate since the code is obfuscated (classes, methods, fields and variables all have randomised names) and due to (i) the extensive use of multiple nested abstract and interface classes within Google apps, (ii) the wide use of async abstractions (Google's extensions to Java futures) which spread processing amongst multiple classes and break strong typing (data is passed into async callables as generic Java objects, complicating variable tracing), (iii) serialising protobufs to byte strings and arrays and processing them in this raw form, including merging with other protobufs.

The software we have written to decode network requests and responses (and which embodies much of the protobuf analysis work carried out) is available on github https://github.com/doug-leith/android-protobuf-decoding.

2) Exposing Google Auth Token Chain of Authorization: When a user logs in to their Google account, Google apps and subsystems ask Google Play Services to download authorization tokens which are then used, for example, to gain access to user data stored on Google servers. Google servers encypt the tokens using a public key, previously sent to them by Google Play Services. When it receives the tokens, Google Play Services decrypts them using its private key. After decryption the token is a base64 encoded protobuf that contains changing HMAC content, which makes tracking of the linkage back to the authorizing user account sllightly complicated since the base64 token encoding continually changes. We therefore exposed the chain of authorization as follows. The private decryption key is inacessible in the hardware keystore, so we decompiled the Google Play Services, identified the Java method carrying out the decryption and wrote a custom Frida tool to dump out tokens before and after decryption. We then decoded the token protobuf, identified the invariant byte string within it which acts as the actual basis for authorization, and tracked that across network connections. See the Additional Material for much more detail. The tools that we developed are available in the anonymous dropbox folder.

3) Firebase Analytics Test App: To investigate transmission of the DSID cookie we created a Google Firebase Analytics account and an Android app linked to the account based on Google's exemplar analytics app. This app is available in the

[6]https://github.com/lupohan44/TrustUserCertificates

[7]The protobufs themselves are encoded within the app in compact protobuf format, which is undocumented although there are useful comments embedded in the Android source code, see https://cs.android.com/android/platform/superproject/+/master:external/protobuf/java/core/src/main/java/com/google/protobuf/RawMessageInfo.java

anonymous dropbox folder. The Firebase Analytics dashboard settings used were as follows: Google signals data collection enabled, granular location and device data collection enabled, user data collection acknowledged.

E. Test Design

1) Device Settings: At the start of each test we factory reset the handset. Following this, the handset reboots to a welcome screen and the user is then presented with a number of option screens. We collected data for two choice of settings. Firstly, 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. Specifically, we deselected the (i)"Use location" option, (ii) "Allow scanning" option and (iii) the "Send usage and diagnostic data" option. Note that there is no option to deselect automatic updates which must be accepted to continue. Secondly, we left all the options at their default values i.e the (i)"Use location" option, (ii) "Allow scanning" option and (iii) the "Send usage and diagnostic data" option were left enabled. However, we did not observe any great difference in the data stored on the handset for these two choices of settings and mainly report results only for the privacy conscious user. We did not log in to a Google user account during the onboarding process. The handset had a SIM card inserted, with an expired data plan so no cellular data was sent.

Later in the tests we log in to a Google user account using the Google Play store app (which cannot be used without logging in). During onboarding after login, (i) at the "Welcome" screen we click "I agree" (the only option), (ii) at the "Google Services" screen the privacy conscious user sets "Backup device data" off (the default user leaves it on) then clicks "Accept". The Google user account settings were "Web & App Activity" on, "Timeline" and "Youtube History" paused, personalized ads and search personalization set off, location sharing off.

2) Testing Protocol: The e-Privacy Directive allows data storage for services explicitly requested by the handset user. To avoid raising this issue, in our tests we make minimal use of the handset. Folllowing factory reset of the handset the user goes through the onboarding process and then leaves the handset idle i.e makes no explicit service requests at all. After about 24 hours the user logs in to a Google account using the Google Play store app. This makes a request for service by that app, but no more than that. The Google Play store app is of particular interest since it is the primary means by which users install other apps on the handset, and so fundamental to the usefulness of the handset. We also opened the Google Dialer app (used to make/receive phone calls), the Google Messages app (used to send/receive SMS messages), the Google Files app (used to access files on the handset) and the Settings app (used to see Google account settings).

In more detail, we: (i) factory reset the handset, (ii) go through the onboarding process, (iii) enable developer options in the Settings app, use adb to install the Magisk app and reboot, (iv) install the mitmproxy CA cert as a user cert, use the Magisk trustusercerts module to promote this to a system cert and reboot, (v) connect the handset to a WiFi access point running `mitmdump` to log network connections and their content, and

```
GET https://googleads.g.doubleclick.net/pagead/drt/m?is_lat=0
Request Headers:
authorization : Bearer ya29.m.Cqk...GZR // linked to user's Google account
user-agent : com.google.android.gms/244738035 // Google Play Services
Response Headers:
set-cookie: DSID=ACZ...kL4; expires=Fri, 03-Jan-2025 00:00:00 GMT; path=/;
domain=.doubleclick.net; Secure; HttpOnly; SameSite=none
```

Fig. 1: Example showing response to `DSID` connection that sets the DSID cookie.

```
POST https://android.googleapis.com/auth
Request Headers:
user-agent : com.google.android.gms/244738035 // Google Play Services
POST Body:
androidId=3c315701d87ecc5d&lang=en-IE&google_play_services_version=244738035&
sdk_version=35&device_country=ie&it_caveat_types=2&app=com.google.android.gms&
oauth2_foreground=1&Email=dougleith23sep@gmail.com&pkgVersionCode=244738035&
token_request_options=CAA4AVABYAA=&client_sig=389...788&
Token=aas_et/AKp...hrjr0=&consumerVersionCode=244738035&check_email=1&service=
oauth2:https://www.googleapis.com/auth/emeraldsea.mobileapps.doritos.cookie&
system_partition=1&assertion_jwt=eyJ...sPA&callerPkg=com.google.android.gms&
check_tb_upgrade_eligible=1&callerSig=389...788
Response Data:
grantedScopes=https://www.googleapis.com/auth/mobileapps.doritos.cookie
it=AVo...11g
TokenEncrypted=1
<...>
```

Fig. 2: Example showing `Auth` connection fetching authorization/bearer token used in later `DSID` connection.

then leave the handset idle and connected to power for about 24 hours. On the second day we log in to a Google account using the Google Play store app and observe the effect on the network connections made. In more detail, we (i) use adb to run a `Frida` server[8] on the handset and launch a custom tool to dump out `Auth` tokens after they are decrypted within the Google Play Services app, (ii) open the Google Play store app and login, (iii) perform some searches with the Google Play app, (iv) open the Google Dialer app and try to make a phone call, (v) open the Google Messages app and try to send a message, (vi) open the Google Files app and navigate to the Downloads folder, (vii) open the Settings app and use it to view Google account settings, (viii) use adb to install a custom Firebase Analytics test app and open the app.

III. Measurement Results

To aid the expositiion, a brief summary is given at the end of each section below.

A. Cookies & Identifiers Stored By Google Play Services

1) DSID Advertising Tracking Cookie: Shortly after the user logs into their Google account, Google Play Services makes a `DSID` connection to googleads.g.doubleclick.net/pagead/drt/m which sets a DSID cookie, e.g. see Figure 1. The cookie is stored in file shared_prefs/social.account_doritos.xml within the Google Play Services app's data folder /data/users/0/com.google.android.gms.

The `DSID` request which fetches the cookie sends an http `authorisation` header. This header contains a bearer token, which is sent to the handset in response to an earlier `Auth` connection to https://android.googleapis.com/auth, see Figure 2. This `Auth` connection sends the user's Google account email together with an authentication token linked to the account (it is sent to the handset by Google servers during the account login process, see Additional Material for more details). The DSID cookie is therefore directly linked to the

[8] `Frida` https://frida.re/ uses the Android system debugger API to facilitate interception of Java function calls.

```
POST https://region1.app-measurement.com/a
Request Headers:
user-agent : com.google.android.gms/244738035 // Google Play Services
POST Body (decoded):
<...>
event_code: "_vs" // screen_view
event_timestamp: 1734728927324
<...>
event_code: "_e" // user_engagement
event_timestamp: 1734728932076
<...>
package_name: "com.google.firebase.quickstart.analytics_notset"
google_ad_id:"d4a1aab9-f57a-43f0-a138-60fd1af33913"
cookie: "DSID=ACZ...kL4"
```

Fig. 3: Example showing Firebase Analytics connection transmitting the DSID cookie, together with the Google Ad Id.

user's Google account. Inspection of the Google Play Services code also confirms that the DSID cookie is directly linked to the Google user account (e.g. it is retrieved from the handset storage by presenting the user account name).

The DSID cookie is almost certainly the primary mechanism via which Google links analytics and advertising events (e.g. ad views and clicks) to an individual user. To demonstrate its operation we created a Google Firebase Analytics account and an Android app linked to the account based on Google's exemplar analytics app. When the Google Signals option is enabled on the account dashboard, we observe that the DSID cookie is sent in Firebase Analytics connections logging user interactions with the app. See, for example, Figure 3. The DSID cookie allows the event (in this case the user viewing a screen within the app) to be linked to an individual user via their Google account. It can be seen that the Google Advertising Id is also sent alongside the DSID cookie.

Google"s public documentation on the Google Signals option and the DSID cookie is, unfortunately, rather vague and is not as helpful as it might be. A Google cookie policy document[9] states that "Some cookies and similar technologies used for advertising are for users who sign in to use Google services. For example, the 'DSID' cookie is used to identify a signed-in user on non-Google sites so that the user's Ads Personalisation setting is respected accordingly". A Google Analytics 4 "Activate Google signals for Google Analytics properties" help page[10] states that "Google signals are session data from sites and apps that Google associates with users who have signed in to their Google accounts, and who have turned on Ads Personalization. This association of data with these signed-in users is used to enable cross-device remarketing, and cross-device key events export to Google Ads".

No consent is sought from the user before storing the DSID cookie on the handset, and there is no opt out. The DSID cookie is used for marketing/advertising analytics. Since the DSID cookie is personally identifying, GDPR likely applies.

DSID advertising analytics cookie, sent by googleads.g.doubleclick.net and stored by Google Play Services in shared_prefs/social.account_doritos.xml, transmitted in Firebase Analytics connections. Stored after user logs in to Google account.

2) Google Android ID Persistent Device Identifier: Following a factory reset and with the user not logged in, the Google

[9]https://policies.google.com/technologies/cookies, accessed 4th Jan 2025.

[10]support.google.com/analytics/answer/9445345, accessed 4th Jan 2025.

```
POST https://android.googleapis.com/checkin
POST Body (decoded):
imei: "358287589858647" // identities handset sim slot
hardwareSerialNumber: "36141FDH20008V" // identifiers handset
androidId: 0   // iniitally zero, later updated
securityToken: 0  // iniitally zero, later updated
accountCookie: ""  // iniitally empty, later user Google a/c email, NID cookie
<details of device hardware,software and mobile network operator>
Response Data (decoded):
setting {name: "android_id", value: "4337343581573270621"}
<setting name/value pairs>
androidId: 4337343581573270621
securityToken: 209675457024254327
versionInfo: "01tKalJmq7UfQKri541rkWVFl_XjPRk"
deviceDataVersionInfo: "ABF...Nlzc"
```

Fig. 4: Example of response to first Google `Checkin` connection following a factory reset sending Google Android ID.

Android ID (aka Google Services Framework Android ID or GSFID[11]) assigned to a handset is sent in the response to the first `Checkin` connection made by Google Play Services to android.googleapis.com/checkin, see for example Figure 4. This value is widely used within Google Play Services and is stored in multiple places on the handset, in particular: in shared_prefs/Checkin.xml, files/checkin_id_token, databases/gservices.db, shared_prefs/constellation_prefs.xml within the Google Play Services app's data folder /data/users/0/com.google.android.gms.

The Google Android ID is a persistent device identifier that can only be changed by factory resetting the handset (which wipes all data). It is sent in many connections to Google servers made by Google Play Services and the Google Play store app, see Additional Material for a list of connections observed in our tests, which is likely not exhaustive. The Google Android ID is stored and transmitted even when the user is not logged in to their Google account. Once the user logs in then the Google Android ID becomes linked to their Google account[12], and so often to their real identity[13]. Logging out from the Google account does not remove this link. Google themselves know that the Google Android ID is personally identifying information (PII), stating in a code comment[14] ""Gservices" android ID. Considered PII.".

While the Google Android ID is the primary device identifier sent by the `Checkin` response, the securityToken and deviceDataVersionInfo also can act as device identifiers. In connections to android.apis.google.com/c2dm/register3 made by Google Play Services, the securityToken value sent in the `Checkin` response is combined with the Google Android ID and sent in http header "authorization: AidLogin 4337343581573270621:209675457024254327". These

[11]https://support.google.com/android/answer/9021432?hl=en

[12]We confirmed the linkage between the Google Android ID and the user's Google account by making a Google Takeout request using https://takeout.google.com/, which shows the Google Android ID under the Device Configuration Service data.

[13]When creating a Google account a user is encouraged to supply their phone number. Use of Google services such as buying a paid app on the Google Play store or using Google Pay further links a person's Google account to their credit card/bank details. A user's Google account can therefore commonly be expected to be linked to the person's real identity. When multiple Google accounts are used on a handset then the same Google Android ID is linked to each of these accounts. However, even when a handset contains multiple Google accounts these are most likely used either by the same person, e.g. for home and business purposes, or by a very small number of people, e.g. a family sharing a tablet.

[14]https://android.googlesource.com/platform/packages/experimental/+/13c69c6/PixelPerfect/imported_protos/src/wireless/android/play/playlog/proto/clientanalytics.proto, accessed 4th Jan 2025.

/register3 connections seem to be associated with Google's Cloud Messaging service, used for push notifications. The securityToken is stored in file files/checkin_id_token within the Google Play Services app's data folder. The deviceDataVersionInfo value is sent in `Experiments` and `Clearcut` connections, in Google Play store app connections to play-fe.googleapis.com (in the `X-DFE-Device-Checkin-Consistency-Token` header), and to https://accounts.google.com during Google user account login. It is stored in file shared_prefs/Checkin.xml.

The purpose of the Google Android ID is not clear, no consent is sought from the user before storing the Google Android Id on the handset and there is no opt out. Since the Google Android ID is typically personally identifying, GDPR likely applies.

Google Android ID persistent device identifier, sent by android.googleapis.com and stored by Google Play Services in shared_prefs/Checkin.xml and also elsewhere, transmitted in many Google Play Services and Google Play store connections. Stored shortly after handset startup, before user logs in to Google account.

3) NID Cookies: The responses to several Google connections set an NID cookie, the value of which is then transmitted in subsequent connections. From inspection of the decompiled Google Play Services code this cookie is also referred to by Google as a Zweiback cookie and a pseudonymous Id token. The purpose of these cookies is not clear, no consent is sought from users before storing these cookies on the handset and there is no opt out.

Google Play Services. Google Play Services `Experiments`, `Checkin` and `Clearcut` connections share the same NID cookie value, as do connections made by the Google Play store app to play.googleapis.com/play/log. The cookie value is set by the responses to `Experiments` and `Clearcut` connections.

In more detail, following a factory reset, the protobuf response to the first `Experiments` connection to www.googleapis.com/experimentsandconfigs made by Google Play Services sends a NID cookie value, see for example Figure 5. This value is then sent in later `Experiments` connections and also in `Checkin` connections to android.googleapis.com/checkin, `Clearcut` connections to play.googleapis.com/log/batch and in Play store app connections to play.googleapis.com/play/log. Its value is then periodically updated via either a set-cookie header in a `Clearcut` response or by an `Experiments` protobuf response.

The NID cookie value is stored in the file shared_prefs/PseudonymousIdIntentService.xml within the Google Play Services app's data folder, as well as within the log data queued for transmission in folder files/clearcut.

Google provides no public documentation on the NID cookie on Android, and its purpose is not clear. It's slowly changing value suggests that it is used to link together connections into sessions. However, since the cookie is sent alongside persistent device and user identifiers such as the Google Android ID and

```
POST https://www.googleapis.com/experimentsandconfigs/v1/
getExperimentsAndConfigs?r=12&c=1
Request Headers:
user-agent : com.google.android.gms/243333045 // Google Play Services
POST Body (decoded):
<details of device hardware, mobile network operator and installed software>
Response Data (decoded):
<setting name/value pairs>
experimentToken {value:"" 1: 3}
serverToken: "CAMSAA=="
NIDcookie: "520=dzR...pV"
```

Fig. 5: Example showing response to `Experiments` connection setting NID cookie value.

user Google account email, it's changing values can be trivially linked to the device and user. That is, it can be used as a user and device tracking cookie.

We also observe the same NID cookie value being sent in other connections: by the Google Messages and Google Files apps to notifications-pa.googleapis.com/v1/multiloginupdate, by Google Play Services to feedback-pa.googleapis.com/google.internal.feedback.v1.SurveyService/GetSurveyStartupConfig, scone-pa.googleapis.com/scone.v1.SurveyService/TriggerAnonymous and geller-pa.googleapis.com/geller.oneplatform.GellerService/BatchSync.

NID cookie, sent by www.googleapis.com and play.googleapis.com, stored by Google Play Services in shared_prefs/PseudonymousIdIntentService.xml, transmitted in several Google Play Services connections. Stored before and after user logs in to Google account.

Google Search App. The Google Search app com.google.android.googlequicksearchbox (the search bar shown on the phone home screen) sets and sends an NID cookie in connections to www.google.com/gen_204, proactivebackend-pa.googleapis.com/assistant.proactive.v1.ProactiveBackend/Sync, discover-pa.googleapis.com/google.internal.discover.discofeed.feedrenderer.v1.DiscoverFeedRenderer/QueryBackgroundFeed, discover-pa.googleapis.com/google.internal.discover.discofeed.streamingfeedrenderer.v1.DiscoverStreamingFeedRenderer/QueryStreamingFeed. The same cookie value is shared across this set of connections, and is different from that sent in other connections.

We also observe the same NID cookie value being sent to https://geller-pa.googleapis.com/geller.oneplatform.GellerService/ReconciliationSync and geller-pa.googleapis.com/geller.oneplatform.GellerService/BatchSync. These connections appear to be associated with the user's web history[15].

A Google policy document 9 states that "the 'NID' cookie enables personalised autocomplete features in Search as you type search terms", although that document is mainly focussed on web pages.

Gmail app. The Google Gmail app makes connections to inbox.google.com/sync/st/s, inbox.google.com/sync/i/s, inbox.google.com/sync/i/fd, mail.google.com/mail/adsfe/main that set and send an NID cookie (also a COMPASS cookie). The

[15] `Auth` scope https://www.googleapis.com/auth/webhistory, see Table III in Additional Material.

```
POST https://android.googleapis.com/auth/devicekey
Request Headers:
user-agent : com.google.android.gms/243333045 // Google Play Services
device : 3c315701d87ecc5d // Google Android ID as hex
POST Body (decoded):
droidguardResultRequestString: "Cgb...aaA" // likely acts as a device
fingerprint
androidID: 4337343581573270621
droidguardSessionID: 18102890410522577986
Response data (decoded):
droidguardSessionID: 18102890410522577986
androidID: 4337343581573270621
4: " \3269\331\352...\340E\211P"
deviceKeyToken: "\000^j\200...\200v\356"
```

Fig. 6: Example showing response sending the Google device key value.

```
x-goog-spatula : CjY...87g==
Decoded x-goog-spatula header:
appCertificate {
  packageName: "com.google.android.gms" // Google Play Services
  hashPackageCert: "OJGKRT0HGZNU+LGa8F7GViztV4g="
}
hashAppCertificate: "J\216Xr...\003\003("
androidID: 4337343581573270621
droidguardSessionID: 18102890410522577986
deviceKeyToken: "\000^j\200...\200v\356
```

Fig. 7: Example of an x-goog-spatula header, and its decoding.

same cookie value is shared across this set of connections, and is different from that sent in other connections.

Google Account login. During login to the user's Google account on the Google Play store app NID cookies are set and sent, along with a __Host-GAPS cookie. The connections are mainly to endpoints on server accounts.google.com, although the same cookie value is also sent to telemetry logging server play.google.com/log.

Once-off NID Cookie Use. A number of connections set an NID cookie but never resend it i.e. the cookie is likely scrubbed after the connection. These include connections by the Google Play store app to https://www.googleapis.com/experimentsandconfigs/v1/getExperimentsAndConfigs, safebrowsing.google.com/safebrowsing/clientreport/download-multi, play.googleapis.com/play/log/timestamp. Also connections by the Google Messages app to messages.google.com/web/timesource and Google advert click analytics sent to www.google.com/aclk.

4) Google Device Key Token: Google Play Services periodically makes connections to android.googleapis.com/auth/devicekey. The response is a protobuf which is stored in file files/device_key within the Google Play Services app's data folder, e.g. see Figure 6. The deviceKeyToken value within this protobuf is subsequently sent within an `x-goog-spatula` header in several connections to Google servers. See Additional Material for a list of connections observed in our tests, which is likely not exhaustive.

The `x-goog-spatula` header value is a base64 encoded protobuf, which contains, amongst other things, the Google Android ID and the deviceKeyToken, see Figure 7. Inspection of the decompiled Google Play Services code indicates that the `x-goog-spatula` header is (i) used by the gRPC clients within Google Play Services and (ii) disabled when setting auth_enable_auth_proxy for Google Play Services subsystem com.google.android.gms.auth_account is set false (it defaults to true). As Figure 6 shows, the Google Android ID is sent in the devicekey request, as well as opaque droidguard data that likely acts as a device fingerprint, see Section III-F. Putting these observations together, it seems possible that the `x-goog-spatula` header is used for device-based authentication, with the deviceKeyToken acting as a device fingerprint additional to the Google Android ID. However, this is speculation and Google's public documentation is silent as to the purpose of the deviceKey token.

It is worth noting that some of the connections sending an `x-goog-spatula` header contain sensitive personal data. For example, connections to android-context-data.googleapis.com/google.internal.android.location.kollektomat.v1.KollektomatService send the user location to Google (e.g. GPS co-ordinates plus the signal strength and MAC addresses of nearby WiFi access points). The `x-goog-spatula` header acts to link this sensitive data to the individual handset and user via the Google Android ID and perhaps also via the deviceKey token.

The purpose of the deviceKey token is not clear, no consent is sought from users before storing this token on the handset and there is no opt out.

deviceKey, sent by android.googleapis.com and stored by Google Play Services in files/device_key, transmitted within the `x-goog-spatula` header of several Google Play Services connections. Stored both before and after user logs in to Google account.

5) Authorization/Bearer Tokens Allowing User Tracking: Shortly after a user logs into their Google account several Google apps and subsystems make `Auth` connections to android.googleapis.com/auth. The Google server responses to these connections send authorization tokens that are stored on the handset. See the Additional Material for a detailed analysis of the content of these tokens. Here we make the following observations:

1) The tokens are stored in file files/authaccount/shared/IntermediateTokenStore.pb within the Google Play Services app's data folder
2) The downloaded tokens are sent in `authorization` headers by many Google connections, including `Experiments` and `Clearcut` connections. See the Additional Material for details.
3) The authority for the downloaded authorization tokens derives from the user's Google account (an access token for that account is used when requesting the tokens), see the Additional Material for details. A Google app possessing an authorization token is therefore effectively logged in to the user's Google account.
4) In our tests the user opens the Google Play store app and logs in using that app. The user therefore expects to be logged in to the Google Play store app. However, Google also silently logs the user into a number of other apps, including Gmail, Google Docs, Google Videos, Google Search, Google Calendar, Google Messages and Google Chrome.
5) Google also silently logs various subsystems of Google Play Services in to the user's Google account. In particular: (i) the `DSID` subsystem which then uses this authorization to download a DSID cookie linked to the user's Google account, (ii) the `Clearcut` logging system, which then links transmitted handset telemetry and usage data to Google to the user's Google account.

No consent is sought from the user before these tokens are stored on the handset. Some are for advertising and many are for apps which the user has never used. Since the tokens are personally identifying, GDPR likely applies.

Auth/Bearer tokens, sent by android.googleapis.com and stored by Google Play Services in files/authaccount/shared/IntermediateTokenStore.pb, transmitted in many Google app connections. Stored after user logs in to Google account.

6) Advertising-Related Device Identifiers: A Google Play Services connection to www.googleapis.com/androidcheck/v1/attestations/adAttest downloads a protobuf containing a binary token which likely acts as a device identifier. This token is stored on the handset in table "ad_attestation" within sqlite database databases/gass.db within the Google Play Services app's data folder. Based on inspection of the decompiled Google Play Services app, this token is likely sent within the encrypted `ms` query parameter used in advert-related network connections, e.g. see Figure 9. Similarly, a Google Play Services to deviceintegritytokens-pa.googleapis.com/v1/getAdEventToken also downloads a protobuf which likely acts as a device identifier. It is stored in file shared_prefs/event_attestation_settings.xml as entry "event_attestation_integrity_token". It is accesible via the EventAttestationService API.

A Google Play Services connection to deviceintegritytokens-pa.googleapis.com/v1/getPoIntegrityToken downloads a protobuf that is stored in file files/potokens/shared/potoken.pb within the Google Play Services app's data folder. It is accesible via the IPoTokensService API, and is used as a device identifier by Google's Youtube app[16]

Advertising-related device identifiers (AdAttest token, AdEvent token, POIntegrity token), are sent by www.googleapis.com and deviceintegritytokens-pa.googleapis.com, and stored by Google Play Services (in databases/gass.db, shared_prefs/event_attestation_settings.xml, files/potokens/shared/potoken.pb respectively).

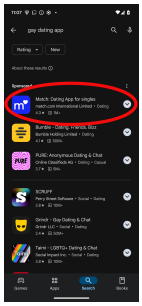
B. Cookies & Ad Tracking Links Stored On Handset By Google Play Store App

1) Play Store Ad Tracking Links: When performing a search within the Google Play store app, "sponsored" results are shown i.e. adverts. When sending the search results to the handset, Google servers also send advert tracking links for each app. When the users clicks on an app in the search results, the corresponding tracking link is used to inform Google of this event.

This behaviour is illustrated in Figures 8-9. In the Google Play store app the user searches for "gay dating apps". Figure 8(a) shows the Play store app fetching the search results. It can be seen that the Match app is contained in the search results, and the corresponding ad tracking link is highlighted in red which contains an `ai` ad identifier query parameter. The ad tracking url is stored in file cache/streamdatastore/streamdatastore.db within the Play store app's data folder on the handset. Figure 8(b) shows a screenshot of the results displayed on the handset,

[16]https://github.com/microg/GmsCore/issues/2253, accessed Jan 6th 2025.

```
GET https://play-fe.googleapis.com/fdfe/search?q=gay+
dating+app&sb=5&ksm=1&ps=1&nocache_pwr=true
Request Headers:
authorization : Bearer ya29.m.CoY...NkE
x-ps-rh : H4s...wAA
Decoded x-ps-rh header:
  androidID: 4337343581573270621
  advertisingID: "d4a1aab9-f57a-43f0-a138-60fd1af33913"
  <...>
POST Body (decoded):
searchResponse: originalQuery "gay dating app"
<...>
serverLogsCookie: "\010\022\232...*\001\001"
<info on net.ilius.android.match app (brief desciption,
category, number of reviews, link to app icon etc>
<...>
1: "net.ilius.android.match"
1: "/pagead/adview?ai=
DChcSEwik5tSsmraKAxV8RPYIHepfHbYYABAAIAE&sigh=NpDWUqio8Ss&
cid=CAQ...YAQ&gvr=1"
1: "https://www.google.com/aclk?sa=L&ai=
DChcSEwik5tSsmraKAxV8RPYIHepfHbYYABAAIAE&num=1&cid=CAQ...
YAQ&sig=AOD...3tg&adurl="
<...>
```



(a) (b)

Fig. 8: (a) Google Play store connection fetching results for search "gay dating app", showing download of ad tracking links for the Match app. (b) screenshot of results shown, with the Match app highlighted.

```
GET
https://www.google.com/aclk?sa=L&ai=DChcSEwik5tSsmraKAxV8RPYIHepfHbYYABAAIAE&
num=1&cid=CAQ...YAQ&sig=AOD...3tg&ms=CoA...fOA&adurl=&nb=22&dim=315x62
Request Headers:
user-agent : Mozilla/5.0 (Linux; Android 15; Pixel%207 Build/AP4A.241205.013)
AppleWebKit/537.36 Finsky/44.0.28-31%20%5B0%5D%20%5BPR%5D%207052970 86 Mobile
Response Headers:
server : adclick_server
set-cookie : NID=520=KoT...7Rc; expires=Sat, 21-Jun-2025 11:07:22 GMT; path=/;
 domain=.google.com; HttpOnly
```

Fig. 9: Ad tracking connection when click on app in search results.

with the Match app highlighted. Clicking on this entry in the search results generates the request shown in Figure 9. The url of this request matches the downloaded ad tracking link in Figure 8(a). The Play store app extends the url with additional `ms`, `nb` and `dim` query parameters. From inspection of the decompiled Play store and Google Play Services apps, the `ms` parameter appears to be a base64 encoded encrypted protobuf that may contain device identifiers. The `dim` parameter appears to give the size of the displayed advert.

The Play store app also reports the ad click event to by sending details to Google server https://play.googleapis.com/play/log. In particular, the events `AD_CLICK_302_RESPONSE` and `CLICK_TRACKING_SUCCESS_SEARCH_AD` are logged.

No consent is sought from users before storing these advertising tracker links on the handset and there is no opt out.

Ad tracking links, sent by play-fe.googleapis.com and stored by Google Play store in cache/streamdatastore/streamdatastore.db, transmitted when click on ad. Stored after user logs in to Google account.

2) Google Play Store ServerLogs Cookies: The Google Play store app downloads content from server play-fe.googleapis.com/fdfe, e.g. see Figure 8(a). The data is sent as a protobuf, but the some of the schema used is publicly available[17]. Each content item is tagged with a binary token, which Google refers to within the schema as a server_logs_cookie, and as a ServerLogsCookie in many places within the Google Play

[17]See https://android.googlesource.com/platform/packages/experimental/+/13c69c6/PixelPerfect/imported_protos/src/wireless/android/play/playlog/proto/clientanalytics.proto and https://github.com/mmcloughlin/finsky/tree/master/protobuf, accessed 4th Jan 2025.

```
POST https://play.googleapis.com/play/log?format=raw&proto_v2=true
Request Headers:
Authorization : Bearer ya29.m.CoY...NkE
POST Body (decoded):
android_id: 4337343581573270621
google_ad_id: d4a1aab9-f57a-43f0-a138-60fd1af33913
<...>
backgroundEvent: SEARCH_TRIGGERED "gay dating app"
serverLogsCookie: "\010\022\232...*\001\001"
<...>
backgroundEvent: SEARCH_SUGGESTIONS_OFFERED
serverLogsCookie: "\010\022\232...*\001\001"
<...>
backgroundEvent: SEARCH_RESULTS_RESPONSE
serverLogsCookie: "\010\022\232...*\001\001"
<...>
backgroundEvent: AD_VIEWABLE_FOR_VTC
serverLogsCookie: "\010\022\232...*\001\001"
<...>
backgroundEvent: CLICK_TRACKING_SUCCESS_SEARCH_AD
document: "net.ilius.android.match"
serverLogsCookie: "\010\022\232...*\001\001"
<...>
```

Fig. 10: Example showing tracking of user interactions while using the Play store app. Interactions are tagged with a ServerLogs cookie downloaded from play-fe.googleapis.com/fdfe, shown in Figure 8(a).

store app. For example, Figure 8(a) shows the ServerLogs Cookie associated with the Match app search result entry.

When the user interacts with content in the Google Play Store app, the app sends details of the interaction plus the relevant ServerLogs cookie to play.googleapis.com/play/log. For example, Figure 10 shows some of the user interaction tracking data sent when the user searches for "gay dating apps" and clicks on the Match app advert in the search response. All of these interactions are tagged with a ServerLogs cookie downloaded from play-fe.googleapis.com/fdfe. Figure 10 also shows that the data sent istagged with the Google Android ID and the Google Advertising ID and so directly linked to the user's Google account.

Each binary ServerLogs cookie decodes as a protobuf. The protobuf consists of a header plus additional information describing the content tagged by the token. The header part is stored in files/finsky/shared/server_logs_cookie_valuestore.pb within the Google Play store app's data folder and likely acts as a user identifier since the user account email is stored alongside it.

No consent is sought from users before storing these tracking cookies on the handset and there is no opt out. Since an authorization/bearer header and the Google Android ID are used when downloading the ServerLogs cookies, they are probably personally identifying in which case GDPR likely applies. Note also that tracking occurs even for GDPR special category data (in the example above, data concerning sexual orientation) requiring explicit consent.

ServerLogs analytics cookies, sent by play-fe.googleapis.com to the Google Play store app, transmitted by Google Play store app to play.googleapis.com. Stored after user logs in to Google account.

C. Google Apps Use of Firebase & Analytics

Several Google apps register with Google Firebase. For example, Figure 11(a) shows the Google Play store app registering with Firebase. The response from the Google server sends two authorization tokens (a refreshToken and an authToken). These tokens are stored in file files/PersistedInstallation.W0R.

```
POST https://firebaseinstallations.googleapis.com/v1/projects/android.com:
api-project-221571841318/installations
Request Headers:
X-Android-Package : com.android.vending // Google Play store app
POST Body:
{"fid":"dT55cNetSmiJkAd8tGpXEz","appId":"1:221571841318:android:9
c547b5ed466b580","authVersion":"FIS_v2","sdkVersion":"a:17.0.2_lp"}
Response Data:
{ "name": "projects/221571841318/installations/dT55cNetSmiJkAd8tGpXEz",
  "fid": "dT55cNetSmiJkAd8tGpXEz",
 "refreshToken": "3_AS...Egw",
 "authToken":  "token": "eyJ...TfM", "expiresIn": "604800s"
```

(a)

```
POST https://android.apis.google.com/c2dm/register3
Request Headers:
authorization : AidLogin 433734358157327062l:209675457024254327
app : com.android.vending // Google Play store app
POST Body:
X-subtype=932144863878&sender=932144863878&X-app_ver=84301830&X-osv=35&X-cliv=
fiid-21.1.1&X-gmsv=243333045&X-appid=dT55cNetSmiJkAd8tGpXEz&X-scope=*&
X-Goog-Firebase-Installations-Auth=eyJ...TfM&X-gmp_app_id=1:221571841318:
android:9c547b5ed466b580&X-firebase-app-name-hash=R1d...xhI&X-app_ver_name
=43.0.18-31 [0] [PR] 679685942&app=com.android.vending&device
=4337343581573270621&app_ver=84301830&info=01tKa1Jmq7UfQKri541rkWVF1_XjPRk&
gcm_ver=243333045&plat=0&cert=389...788&target_ver=35
Response Data:
token=dT5...PPU
```

(b)

Fig. 11: Example showing Google Play store app (a) downloading Firebase tokens and (b) later uploading them.

```
POST https://region1.app-measurement.com/a
POST Body (decoded):
event_code: "_f" // first_open
event_timestamp: 1734600130510
package_name: "com.android.vending" // Google Play store app
<...>
google_ad_id: "d4a1aab9-f57a-43f0-a138-60fd1af33913"
firebase_instance_id: "dT55cNetSmiJkAd8tGpXEz"
```

Fig. 12: Example showing Google Play store app sending user interaction data to Google's Firebase Analytics server region1.app-measurement.com.

..iNTgw.json within the Google Play store app's data folder /data/0/com.android.vending. Figure 11(b) shows the authToken (highlighted in red) later being transmitted to android.apis.google.com/c2dm/register3. See Additional Material for a list of other apps observed making a similar pair of connections.

Google Firebase provides several services, not just Firebase Analytics, and the full set of services used by these apps is not clear. However, we observe the following Google apps making Firebase Analytics connections to transmit user interaction tracking data: android.vending, google.android.apps.nbu.files, google.android.dialer. See, for example, Figure 12.

The connection in Figure 11(b) registers the app with the Google Cloud Messaging service, which is used to send push notifications. The response is a token which is stored on the handset in file shared_prefs/finsky.xml within the Google Play store app's data folder, where it is labelled `gcm-registration-id-on-server`. The `authorization` header sent in the connection is an AidLogin <Google Android ID>:<securityToken> pair, uniquely identifying the handset, and the token also likely uniquely identifies the handset so as to enable push notifications to be appropriately targetted.

No consent is sought from users before storing any of these tokens on the handset. Some of the tokens are associated with marketing analytics, and many are for apps that the user has never used. There is no opt out from storage of these tokens. Since the tokens are probably personally identifying, GDPR likely applies.

```
POST https://www.googleapis.com/experimentsandconfigs/v1/
getExperimentsAndConfigs?r=12&c=1
Request Headers:
user-agent : com.google.android.gms/243333045 // Google Play Services
POST Body (decoded):
androidID: 4337343581573270621
pseudonymousIdToken: "520=oWF...coA"
<details of device hardware, mobile network operator and installed software>
Response Data (decoded):
packageName: "com.google.android.platform.adservices", "com.google.android.gms
"
bytesTag: "v\231\252I\034\313\307\212"
<setting name/value pairs>
experimentToken_value: "\025\210\006...\347\006\004" 1: 3
serverToken: "CAM...GBA=="
<settings for other packages>
```

Fig. 13: Example of an `Experiments` connection showing the response sending Google app settings tagged with an experimentToken and a serverToken.

Auth/refresh tokens, sent by firebaseinstallations.googleapis.com and stored by Google apps, transmitted by Google Firebase; GCM Id, sent by android.apis.google.com and stored by Google Play Services. Stored both before and after user logs in to Google account.

D. A/B Testing of Google App Updates

The `Experiments` subsystem of Google Play Services downloads a protobuf which contains settings for Google apps and subsystems, see for example Figure 13. An experimentToken and a serverToken are sent alongside the settings for each app/subsystem. This data is stored in subfolder files/phenotype/shared/ within the Google Play Services app's data folder, with a separate file for each app/subsystem.

For the default device settings, the experimentTokens are later transmitted by `Clearcut` to play.googleapis.com/log/batch alongside telemetry data. From discussions with Google [1], what is happening here is that Google trials updated app settings via `Experiments` and monitors the result via `Clearcut`, using the experimentTokens to link the telemetry with the choice of settings. This is used, for example, when rolling out app updates: the update is initially rolled out to a small subset of users, if all goes well (as indicated by `Clearcut` telemetry) then the update is gradually rolled out to more users. That is, users are silently used as testers of Google app changes, with no opt out.

The experimentTokens are also transmitted by `Experiments` to www.googleapis.com/experimentsandconfigs (alongside details of the installed software), regardless of the device settings.

The purpose of the serverTokens value is unclear. Several are transmitted in `Clearcut` connections, and many related to the Google Search app are sent to geller-pa.googleapis.com/geller.oneplatform.GellerService/BatchSync.

The Google Play store app also makes its own connections to www.googleapis.com/experimentsandconfigs. These send device details (similar to those sent in `Checkin` connections) and the response is Play store app settings plus experimentTokens and serverTokens.

The settings stored on the handset by the `Experiments` subsystem can potentially contain unique identifers that could be used to track the individual handset and user. While we have not seen any evidence of this in our measurements, a great many setting values are downloaded to the handset and we have only been able to properly inspect a small fraction of them.

No consent is sought from users before storing these settings and tokens on the handset and there is no opt out.

App settings/experiment tokens/server tokens, sent by www.googleapis.com and stored in folder files/phenotype/shared/, experiment and server tokens transmitted to play.googleapis.com. Stored both before and after user logs in to Google account.

E. Cryptographic Keys

1) Google Attestation Of Hardware Keystore Keys: Android handsets contain a hardware trusted execution environment[18]. To ensure that private keys never have to leave the trusted environment, an API is exposed to apps running on the handset and encryption and decryption is carried out by code running in the trusted environment. The public key associated with a private key is also exposed. The public key is signed by an intermediate certificate which in turn may be signed by other intermediate certificates until a root certificate is reached. This root certificate is owned by Google. An app can verify the integrity of this certificate chain (no intermediate certificate have been revoked etc), and so the integrity of the key used by the app.

When the trusted environment, generates new private/public key pairs for use by apps, the process for asking Google to sign these keys is outlined in a Google blog post[19]. In the factory a private/public key pair are generated in the handset trusted environment and the public key is extracted and sent to Google and added to a list of known factory public keys. On first use, following a factory reset, or when running low on keys, the trusted environment generates additional private/public key pairs for use by apps. Android then sends certificate signing requests for these new keys to Google servers. These requests are self-signed by the factory private key. The Google server checks that the public key corresponding to the private key used to sign the request is on Google's list of known factory public keys. If it is on the list, Google signs the new keys and sends them back to the handset.

It is worth noting that this key signing setup potentially allows Google to track the individual handset associated with every Android keystore public certificate. To mitigate this privacy concern, Google state in their blog post that the server which verifies the factory public key is kept isolated and never sees the keys which have been sent for signing.

Apart from the Google blog post there appears to be no public documentation on the data sent and received during the key signing process. However, the functionality of part of the

[18]https://source.android.com/docs/security/features/trusty, accessed Jan 6th 2025

[19]https://android-developers.googleblog.com/2022/03/upgrading-android-attestation-remote.html, accessed 5th Jan 2025.

```
POST https://remoteprovisioning.googleapis.com/v1/:fetchEekChain
POST Body:
{'id': 82726, 'fingerprint': 'google/panther/panther:15/AP4A
.241205.013/12621605:user/release-keys'}
Response Data (decoded):
<pair of ECDH keys and cert chains, used in later cert signing request>
Challenge: AAABk94/XPABILStY05LFxLe7GlYapF6oEyMc4g=
DeviceConfig: {'bad_cert_end': 1719532800000, 'bad_cert_start': 1719187200000,
 'time_to_refresh_hours': 72, 'num_extra_attestation_keys': 12}
```

(a) Fetch endpoint encryption key (eek)

```
POST https://remoteprovisioning.googleapis.com/v1/:signCertificates?challenge=
AAABk94_XPABILStY05LFxLe7GlYapF6oEyMc4g%3D&request_id=7
d29cbe2-4873-4e9e-8927-8c615dadf624
POST Body:
<device details, list of public keys, other binary content>
Response Data (decoded):
<Google certificate chain>
<public keys signed by Google>
```

(b) Key signing request.

Fig. 14: Example of the pair of connections made when downloading Google signed public keys.

```
https://www.googleapis.com/certificateprovisioning/v1/devicecertificates/
create?key=AIzaSyB-5OLKTx2iU5mko18DfdwK5611JIjbUhE&preProvisioning=true&
signedRequest=<base64 encoded binary data>
Request Headers:
User-Agent : AndroidRemoteProvisioner/google/Pixel 7/user/12621605/AP4A
.241205.013
Response Data:
{"header": {},
  "signedResponse": <base64 encoded binary data>,
  "kind": "certificateprovisioning#certificateProvisioningResponse"}
```

Fig. 15: Example showing a Widevine provisioning request.

Android Open Source Project (AOSP) and so at least we can inspect the code[20].

Figure 14 shows examples of the pair of connections we observe during the key signing process. The first connection fetches a pair of encryption keys. One of these keys is then used to encrypt part of the data sent in the second connection, which also sends a list of public keys to be signed. The response to the second connection is a certificate chain and a sequence of signed public keys.

While this key signing process does not seem especially worriesome, no consent is sought from users before storing the key data on the handset and there is no opt out.

2) Widevine DRM Provisioning: Widevine is Google's proprietary Digital Rights Management system[21]. There is little public documentation on Widevine, but there has been some analysis by security researchers [23]. Video and audio media content is encrypted and sent to the handset, it is then decrypted inside the trusted environment on the handset. This process requires the exchange of encryption keys with the handset. Key exchanges are protected by a private device key which is sent to the handset during initial "provisioning". Scheduling of the initial provisioning connections is handled by AOSP[22].

Figure 15 shows an example of a Widevine provisioning request. The `signedRequest` query parameter is a base64 encoded protobuf. When decoded the first entry is "widevine.com", and the rest of the content appears to be encrypted (it is generated within the handset trusted environment so we cannot easily access the plain text). However, it is likely that the data sent contains unique device identifiers [23]. The response is a base64 encoded protobuf, and again the content appears to be encrypted.

We observe two sets of connections similar to that in Figure 15 following handset factory reset. The first appears to be generated by the Android system (with user-agent header "AndroidRemoteProvisioner") and the second by Google Chrome (with user-agent header "Widevine CDM v1.0")[23] even though the Google Chrome app has never been used.

No consent is sought from users before storing the Widevine provisioning data on the handset and there is no opt out. The data is stored despite the fact that no media content is downloaded or viewed in our tests.

Widevine key, sent by www.googleapis.com and stored by Android system and Google Chrome in trusted environment. Stored before user logs in to Google account.

F. Droidguard

The `Droidguard` subsystem of Google Play Services underpins Google's Play Integrity service (previously called SafetyNet)[24] that is made available to third-party apps. There is almost no Google public documentation on `Droidguard`, but its role in SafetyNet was recently analysed by security researchers [24]. This work indicates that `Droidguard` generates device fingerprints (likely uniquely identifying). Our measurements suggest that `Droidguard` is also used by other Google Play Services subsystems e.g. `Checkin` and those handling reporting of advert events such as views and clicks.

`Droidguard` makes periodic connections to https://www.googleapis.com/androidantiabuse/v1/x/create, downloading large opaque protobufs. These appear to be stored in folder app_dg_cache within the Google Play Services app's data folder.

No consent is sought from users before storing the Droidguard data on the handset and there is no opt out. Data is stored both before and after user logs in to Google account.

IV. Summary and Conclusions

We report on the results of a measurement study investigating the cookies, identifiers and other data stored on Android handsets by Google Play Services, the Google Play store and other pre-installed Google apps. We find that multiple cookies and identifiers are sent by Google servers and stored on the handset, even when no Google apps have ever been opened by

[20]Useful entry points are https://cs.android.com/android/platform/superproject/main/+/main:packages/modules/RemoteKeyProvisioning/app/src/com/android/rkpdapp/interfaces/ServerInterface.java, https://cs.android.com/android/platform/superproject/main/+/main:system/keymaster/android_keymaster/android_keymaster.cpp, accessed 5th Jan 2025

[21]https://developers.google.com/widevine/drm/overview, accessed Jan 6th 2025

[22]https://cs.android.com/android/platform/superproject/main/+/main:packages/modules/RemoteKeyProvisioning/app/src/com/android/rkpdapp/provisioner/WidevineProvisioner.java, accessed Jan 6th 2025.

[23]https://github.com/chromium/chromium/blob/133.0.6909.1/content/browser/media/url_provision_fetcher.cc, https://github.com/chromium/chromium/blob/133.0.6909.1/content/browser/media/url_provision_fetcher.cc, accessed Jan 6th 2025.

[24]https://developer.android.com/google/play/integrity, accessed Jan 6th 2025.

the user. This includes advertising analytics/tracking cookies, links and device identifiers. No consent is sought for storing any of this data and there is no opt out. To the best of our knowledge, this study is the first describing the cookies etc stored by pre-installed Google apps.

REFERENCES

[1] D. J. Leith, "Mobile Handset Privacy: Measuring The Data iOS and Android Send to Apple And Google," in *Proc Securecomm*, 2021.

[2] ——, "What Data Do The Google Dialer and Messages Apps On Android Send to Google?" in *Proc Securecomm*, 2022.

[3] "Report by the Data Protection Commission on the use of cookies and other tracking technologies," April 2020. [Online]. Available: https://www.irishstatutebook.ie/eli/2011/si/336/

[4] "Guidelines 2/2023 on Technical Scope of Art. 5(3) of ePrivacy Directive, EDPB," Oct 2024. [Online]. Available: https://www.edpb.europa.eu/system/files/2024-10/edpb_guidelines_202302_technical_scope_art_53_eprivacydirective_v2_en_0.pdf

[5] "S.I. No. 336/2011," 2011. [Online]. Available: https://www.dataprotection.ie/sites/default/files/uploads/2020-04/Data%20Protection%20Commission%20cookies%20sweep%20REVISED%2015%20April%202020%20v.01.pdf

[6] "Opinion 04/2012 on Cookie Consent Exemption, European Data Protection Board," June 2012. [Online]. Available: https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2012/wp194_en.pdf

[7] M. Trevisan, S. Traverso, E. Bassi, and M. Mellia, "4 Years of EU Cookie Law: Results and Lessons Learned," in *Proc Privacy Enhancing Technologies Symposium*, no. 2, 2019, pp. 126–145.

[8] T. T. Nguyen, M. Backes, and B. Stock, "Freely Given Consent? Studying Consent Notice of Third-Party Tracking and Its Violations of GDPR in Android Apps," in *Proc ACM SIGSAC*, 2022.

[9] M. Degeling, C. Utz, C. Lentzsch, H. Hosseini, F. Schaub, and T. Holz, "We Value Your Privacy ... Now Take Some Cookies: Measuring the GDPR's Impact on Web Privacy," in *Proc NDSS*, 2019.

[10] "Report of the work undertaken by the Cookie Banner Taskforce, European Data Protection Board," Jan 2023. [Online]. Available: https://www.edpb.europa.eu/system/files/2023-01/edpb_20230118_report_cookie_banner_taskforce_en.pdf

[11] A. Cahn, S. Alfeld, P. Barford, and S. Muthukrishnan, "An Empirical Study of Web Cookies," in *Proc WWW*, 2016.

[12] R. Gonzalez, L. Jiang, M. Ahmed, M. Marciel, R. Cuevas, H. Metwalley, and S. Niccolini, "The cookie recipe: Untangling the use of cookies in the wild," in *Proc TMA*, 2017, pp. 1–9.

[13] M. Gotze, S. Matic, C. Iordanou, G. Smaragdakis, and N. Laoutaris, "Measuring Web Cookies in Governmental Websites," in *Proc 4th ACM Web Science Conference*, 2022.

[14] K. Kanungo, R. Khatoliya, V. Arora, A. Bari, A. Bhattacharya, M. Maity, and S. Chakravarty, "How Many Hands in the Cookie Jar? Examining Privacy Implications of Popular Apps in India," in *Proc EuroS&P*, 2024.

[15] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, and S. Sundaresan, "Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking Ecosystem," in *Proc NDSS*, 2018.

[16] H. Jin, M. Liu, K. Dodhia, Y. Li, G. Srivastava, M. Fredrikson, Y. Agarwal, and J. I. Hong, "Why are they collecting my data? inferring the purposes of network traffic in mobile apps," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 4, Dec. 2018.

[17] J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez, "Bug Fixes, Improvements,... and Privacy Leaks," in *Proc NDSS*, 2018.

[18] Q. Jia, L. Zhou, H. Li, R. Yang, S. Du, and H. Zhu, "Who Leaks My Privacy: Towards Automatic and Association Detection with GDPR Compliance," in *Proc Wireless Algorithms, Systems, and Applications*, 2019.

[19] D. J. Leith and S. Farrell, "Contact Tracing App Privacy: What Data Is Shared By Europe's GAEN Contact Tracing Apps," in *Proc IEEE INFOCOM*, 2021.

[20] H. Liu, P. Patras, and D. J. Leith, "Android Mobile OS Snooping By Samsung,Xiaomi, Huawei and Realme Handsets," in *SCSS Tech Report, Oct 2021*. [Online]. Available: https://www.scss.tcd.ie/doug.leith/Android_privacy_report.pdf

[21] H. Liu, D. J. Leith, and P. Patras, "Android OS Privacy Under the Loupe – A Tale from the East," in *Proc WiSec*, 2023.

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

[23] G. Patat, M. Sabt, and P.-A. Fouque, "Exploring Widevine for Fun and Profit," in *IEEE Security and Privacy Workshops (SPW)*, 2022, pp. 277–288.

[24] Romain Thomas, "DroidGuard: A Deep Dive into SafetyNet," in *Proc BlackHat Asia*, 2022.

ADDITIONAL MATERIAL

A. Connections Which Send Google Android ID

```
https://accounts.google.com/v3/signin/_/AccountsSignInUi/data/batchexecute
https://accountsettingsmobile-pa.googleapis.com/google.internal.identity.
accountsettings.mobile.v1.AccountSettingsMobile/GetResource
https://afwprovisioning-pa.googleapis.com/v1/get_device_provisioning_record
https://android-safebrowsing.google.com/safebrowsing/clientreport/download
https://android-safebrowsing.google.com/safebrowsing/clientreport/
download-multi
https://android.apis.google.com/c2dm/register3
https://android.clients.google.com/fdfe/api/userProfile
https://android.googleapis.com/auth
https://android.googleapis.com/auth/devicekey
https://android.googleapis.com/auth/enterprise/fetch_managing_app_for_user
https://android.googleapis.com/auth/lookup/account_state
https://android.googleapis.com/checkin
https://auditrecording-pa.googleapis.com/google.internal.api.auditrecording.v1
.AuditRecordingMobileService/CreateAuditRecord
https://chromesyncpasswords-pa.googleapis.com/google.internal.chrome.sync.
passwords.v1.Passwords/GetMetadata
https://chromesyncpasswords-pa.googleapis.com/google.internal.chrome.sync.
passwords.v1.Passwords/ListIncomingPasswordSharingInvitations
https://chromesyncpasswords-pa.googleapis.com/google.internal.chrome.sync.
passwords.v1.Passwords/ListPasswords
https://chromesyncpasswords-pa.googleapis.com/google.internal.chrome.sync.
passwords.v1.Passwords/ListPriorityPreferences
https://chromesyncpasswords-pa.googleapis.com/google.internal.chrome.sync.
passwords.v1.Passwords/ListWebauthnCredentials
https://cryptauthdevicesync.googleapis.com/google.cryptauth.devicesync.v1.
DeviceSync/BatchGetFeatureStatuses
https://cryptauthdevicesync.googleapis.com/google.cryptauth.devicesync.v1.
DeviceSync/ShareGroupPrivateKey
https://cryptauthdevicesync.googleapis.com/google.cryptauth.devicesync.v1.
DeviceSync/SyncMetadata
https://cryptauthenrollment.googleapis.com/google.cryptauth.enrollment.v1.
Enrollment/EnrollKeys
https://cryptauthenrollment.googleapis.com/google.cryptauth.enrollment.v1.
Enrollment/SyncKeys
https://discover-pa.googleapis.com/google.internal.discover.discofeed.
feedrenderer.v1.DiscoverFeedRenderer/QueryBackgroundFeed
https://findmydevice-pa.googleapis.com/google.internal.fmd.FmdApiService/
FetchSpotAndroidDeviceRegistration
https://flexagon-pa.googleapis.com/google.internal.googlegrowth.flexagon.v1.
FlexagonApi/GetCurrentDropsCampaignStatus
https://gameswhitelisted.googleapis.com/google.play.games.whitelisted.
v1whitelisted.PlayersFirstParty/GetPlayerFirstParty
https://gmscompliance-pa.googleapis.com/google.internal.gmscore.gmscompliance.
v1.GmsCompliance/GetEnforcement
https://instantmessaging-pa.googleapis.com/google.internal.communications.
instantmessaging.v1.Registration/SignInGaia
https://locationhistory-pa.googleapis.com/google.internal.locationhistory.v1.
LocationHistoryService/GetSettings
https://locationhistory-pa.googleapis.com/google.internal.locationhistory.v1.
LocationHistoryService/ListTombstones
https://locationhistory-pa.googleapis.com/google.internal.locationhistory.v1.
LocationHistoryService/UpdateDeviceMetadata
https://mobileconfiguration-pa.googleapis.com/google.internal.communications.
mobileconfiguration.v1.MobileConfigurationService/GetConfiguration
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListMyDevices
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListPublicCertificates
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListReachablePhoneNumbers
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListSenderCertificates
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/UpdateDevice
https://notifications-pa.googleapis.com/google.internal.notifications.v1.
NotificationsApiService/MultiLoginUpdate
https://notifications-pa.googleapis.com/v1/multiloginupdate
https://notifications-pa.googleapis.com/v1/storetarget
https://notifications-pa.googleapis.com/v1/updateandfetchthreads
https://people-pa.googleapis.com/google.internal.people.v2.
InternalPeopleService/ListContactGroups
https://people-pa.googleapis.com/google.internal.people.v2.
InternalPeopleService/SyncPeople
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/GetConsent
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/Proceed
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/SetConsent
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/Sync
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneNumber/GetVerifiedPhoneNumbers
https://pixelonboarding-pa.googleapis.com/google.internal.android.pixel.
onboarding.v1.TipsContentService/GetTipsContent
https://play-fe.googleapis.com/fdfe/delivery
https://play-fe.googleapis.com/fdfe/earlyUpdate
https://play-fe.googleapis.com/fdfe/moduleDelivery
https://play-fe.googleapis.com/fdfe/selfUpdate
https://play-fe.googleapis.com/fdfe/sync
https://play.googleapis.com/log/batch
https://play.googleapis.com/play/log
https://playatoms-pa.googleapis.com/google.internal.play.atoms.api.v1.
WhApiService/GetDomainFilter
https://playatoms-pa.googleapis.com/google.internal.play.atoms.api.v1.
WhApiService/UpdateUserPrefs
https://playatoms-pa.googleapis.com/v1/userPrefs
https://playgateway-pa.googleapis.com/play.gateway.adapter.gmscore.v1.
PlayGatewayGMSCoreService/GetSearchCompletions
https://reminders-pa.googleapis.com/caribou.tasks.service.
CustomizeSnoozePresetsService/GetSnoozePresets
https://reminders-pa.googleapis.com/caribou.tasks.service.TasksApiService/
ListTasks
https://restrictedapps-pa.googleapis.com/google.internal.identity.checkpoint.
restrictedapps.v1.RestrictedAppsService/SyncRestrictedAndroidApps
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/GetSecurityDomain
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/GetSecurityDomainMember
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/JoinSecurityDomain
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/ListSecurityDomainMembers
https://semanticlocation-pa.googleapis.com/userlocation.
SemanticLocationService/GetUserEditInfo
https://ulp1p-pa.googleapis.com/il8n.language_profile.mobile.MobileUlpService/
GetUserLanguageProfile
https://ulp1p-pa.googleapis.com/il8n.language_profile.mobile.MobileUlpService/
SetLanguageSettings
https://userlocation.googleapis.com/userlocation.UserLocationReportingService/
SetApiSettings
https://wallet.google.com/payments/apis-secure/androidsetupwizardservice/
initialize
https://www.googleapis.com/experimentsandconfigs/v1/getExperimentsAndConfigs
```

B. Connections Which Send X-goog-spatula Header

```
https://accountsettingsmobile-pa.googleapis.com/google.internal.identity.
accountsettings.mobile.v1.AccountSettingsMobile/GetResource
https://android-context-data.googleapis.com/google.internal.android.location.
kollektomat.v1.KollektomatService/Offer
https://androidplatformbackuprestore-pa.googleapis.com/google.internal.apps.
backup.androidplatformbackuprestore.v1.AndroidPlatformBackupRestoreService/
GetStorageQuotaInfo
https://auditrecording-pa.googleapis.com/google.internal.api.auditrecording.v1
.AuditRecordingMobileService/CreateAuditRecord
https://chromesyncentities-pa.googleapis.com/google.internal.chrome.sync.
entities.v1.SyncEntities/CreateDeviceInfo
https://chromesyncpasswords-pa.googleapis.com/google.internal.chrome.sync.
passwords.v1.Passwords/GetMetadata
https://cryptauthdevicesync.googleapis.com/google.cryptauth.devicesync.v1.
DeviceSync/BatchGetFeatureStatuses
https://cryptauthdevicesync.googleapis.com/google.cryptauth.devicesync.v1.
DeviceSync/SyncMetadata
https://cryptauthenrollment.googleapis.com/google.cryptauth.enrollment.v1.
Enrollment/EnrollKeys
https://cryptauthenrollment.googleapis.com/google.cryptauth.enrollment.v1.
Enrollment/SyncKeys
https://devicelockcheckin-pa.googleapis.com/google.internal.android.devicelock
.checkin.v1.DeviceLockCheckinService/GetDeviceCheckinStatus
https://dialercallinfolookup-pa.googleapis.com/google.internal.dialer.v1.
DialerCallInfoLookupService/GetCallInfo
https://discover-pa.googleapis.com/google.internal.discover.discofeed.
feedrenderer.v1.DiscoverFeedRenderer/QueryBackgroundFeed
https://flexagon-pa.googleapis.com/google.internal.googlegrowth.flexagon.v1.
FlexagonApi/GetCurrentDropsCampaignStatus
https://gameswhitelisted.googleapis.com/google.play.games.whitelisted.
v1whitelisted.PlayersFirstParty/GetPlayerFirstParty
https://geomobileservices-pa.googleapis.com/google.internal.maps.
geomobileservices.geocoding.v3mobile.GeocodingService/ReverseGeocode
https://googleonebackup-pa.googleapis.com/google.subscriptions.backup.v1.
GoogleOneBackupService/UpdateBackupConfig
https://instantmessaging-pa.googleapis.com/google.internal.communications.
instantmessaging.v1.Registration/SignInGaia
https://locationhistory-pa.googleapis.com/google.internal.locationhistory.v1.
LocationHistoryService/GetSettings
https://locationhistory-pa.googleapis.com/google.internal.locationhistory.v1.
LocationHistoryService/ListTombstones
https://locationhistory-pa.googleapis.com/google.internal.locationhistory.v1.
LocationHistoryService/UpdateDeviceMetadata
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListMyDevices
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListPublicCertificates
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListReachablePhoneNumbers
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/ListSenderCertificates
https://nearbysharing-pa.googleapis.com/location.nearby.sharing.v1.
NearbySharingService/UpdateDevice
https://notifications-pa.googleapis.com/google.internal.notifications.v1.
NotificationsApiService/MultiLoginUpdate
https://people-pa.googleapis.com/google.internal.people.v2.
InternalPeopleService/FetchBackUpDeviceContactInfo
https://people-pa.googleapis.com/google.internal.people.v2.
InternalPeopleService/ListContactGroups
https://people-pa.googleapis.com/google.internal.people.v2.
InternalPeopleService/SyncPeople
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/GetConsent
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/Proceed
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/SetConsent
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneDeviceVerification/Sync
https://phonedeviceverification-pa.googleapis.com/google.internal.
communications.phonedeviceverification.v1.PhoneNumber/GetVerifiedPhoneNumbers
https://pixelonboarding-pa.googleapis.com/google.internal.android.pixel.
onboarding.v1.TipsContentService/GetTipsContent
https://playatoms-pa.googleapis.com/google.internal.play.atoms.api.v1.
WhApiService/GetDomainFilter
https://playatoms-pa.googleapis.com/google.internal.play.atoms.api.v1.
WhApiService/UpdateUserPrefs
https://quake-pa.googleapis.com/google.internal.android.location.quake.v1.
QuakeApiService/NodeOnline
https://restrictedapps-pa.googleapis.com/google.internal.identity.checkpoint.
restrictedapps.v1.RestrictedAppsService/SyncRestrictedAndroidApps
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/GetSecurityDomain
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/GetSecurityDomainMember
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/JoinSecurityDomain
https://securitydomain-pa.googleapis.com/google.internal.identity.
securitydomain.v1.SecurityDomainService/ListSecurityDomainMembers
https://semanticlocation-pa.googleapis.com/userlocation.
SemanticLocationService/GetUserEditInfo
https://userlocation.googleapis.com/userlocation.UserLocationReportingService/
SetApiSettings
https://voilatile-pa.googleapis.com/google.internal.android.location.voilatile
.v1.VoilaTileService/FindTiles
```

```
POST https://play.googleapis.com/log/batch
Request Headers:
authorization : Bearer ya29.m.CoY...WUE
Decoded authorization Bearer header:
  tokenBytes: "\001\'\376...\301\000\304"
  allowedScopeIds {isUpdated: 1, scopeIds: 16, 3660, elapsedTimeMillis: 0}
  hmacSHA256: "T4\373\317...?\324\020"
  4 { 1: 1}
  5: "aCgYKAZoSARISFQHGX2Miqee7B4Q0u3fTg_lZdzkGMw"
  <...>
```

Fig. 16: Example showing decoded `it` token sent in authorzation/Bearer header of a `Clearcut` logger connections

```
POST https://android.googleapis.com/auth
Request Headers:
user-agent : com.google.android.gms/244738035 // Google Play Services
app : com.google.android.gms // app requesting auth token
POST Body:
androidId=3c315701d87ecc5d&lang=en-IE&google_play_services_version=244738035&
sdk_version=35&device_country=ie&Email=dougleith23sep@gmail.com&pkgVersionCode
=244738035&build_product=panther&build_brand=google&Token=oauth2_4/0Aan...vbw&
registration_jwt=eyJ...EVw&build_fingerprint=google/panther/panther:15/AP4A
.241205.013/12621605:user/release-keys&build_device=panther&service=ac2dm&
get_accountid=1&ACCESS_TOKEN=1&callerPkg=com.google.android.gms&add_account=1&
droidguard_results=Cgbl...788
Response Data:
Token=aas_et/AKp...jr0=
TokenBound=1
services=mail,cl,android,youtube,glphonebackup,hist
Email=dougleith23sep@gmail.com
accountId=117028603991747926668
firstName=Doug
lastName=Leith
capabilities.canHaveUsername=1
capabilities.canHavePassword=1
capabilities=<base64 encoded protobuf>
<...>
```

Fig. 17: Example showing first `Auth` connection after the user has logged into their Google account.

C. Operation Of Google Authorization/Bearer Headers

After a user logs into their Google account, many Google connections transmit an `authorization` header with a bearer token of the form `ya29.m.`<base64 encoded protobuf>. For example, Figure 16 shows a decoded bearer token sent in the authorzation header of a `Clearcut` connection. The protobuf contains tokenBytes, a binary token which is invariant and acts as the access token, and an HMAC hash which changes over time. Due to the HMAC hash changing, the protobuf changes between connections, and so its base64 encoding. Hence, the bearer token observed being sent changes from connection to connection, yet when the bearer token is decoded the binary token within it stays the same.

The invariant binary token is linked to the user's Google account via the following chain of authorization connections.

During Google account login, the response to a connection to accounts.google.com sends an `oauth2_4` token. This `oauth2_4` is then sent in an `Auth` to android.googleapis.com/auth to obtain an `aas_et` access token, see Figure 17. The `registration_jwt` value sent in this `Auth` connection is a JWT token with a public key that is used by Google servers to encrypt the `it` tokens sent in later `Auth` connections (the private key needed for decryption stays on the handset).

The `aas_et` token is unencrypted and is sent in subsequent `Auth` connections made by various Google apps and their subsystems. For example, Figure 18 shows an `Auth` connection made by the `Clearcut` subsystem of Google Play Services. The response is an `it` token that it encrypted (as indicated by the "TokenEncrypted=1" entry in the response). The private key needed to decrypt the `it` token is not easily accessed (likely it is kept in the secure hardware keystore). We

```
POST https://android.googleapis.com/auth
Request Headers:
device : 3c315701d87ecc5d // Google Android ID as hex value
user-agent : com.google.android.gms/244738035 // Google Play Services
app : com.google.android.gms // app requesting auth token
POST Body:
androidId=3c315701d87ecc5d&lang=en-IE&google_play_services_version=244738035&
sdk_version=35&device_country=ie&it_caveat_types=2&app=com.google.android.gms&
oauth2_foreground=0&Email=dougleith23sep@gmail.com&pkgVersionCode=244738035&
token_request_options=CAA4AVABYAA=&client_sig=389...788&
Token=aas_et/AKp...jr0=&consumerVersionCode=244738035&check_email=1&service=
oauth2:https://www.googleapis.com/auth/cclog&system_partition=1&assertion_jwt=
eyJ...IOw&callerPkg=com.google.android.gms&check_tb_upgrade_eligible=1&
callerSig=389...788
Response Data:
issueAdvice=auto
Expiry=1734751659
ExpiresInDurationSec=58920
grantedScopes=https://www.googleapis.com/auth/cclog
itMetadata=GgMKAQIgzBw
it=AS_kt-0...6Gk // encrypted token
TokenEncrypted=1
<...>
```

Fig. 18: Example showing first `Auth` connection for Clearcut logger

```
Decoded decrypted it token:
tokenBytes: "\001\'\376...\301\000\304"
hmacSHA256: "\330,\263...2\365\375C"
4 {1: 1}
5: "aCgYKAZoSARISFQHGX2Miqee7B4Q0u3fTg_lZdzkGMw"
<...>
```

Fig. 19: The decoded decrypted `it` token sent in Figure 18.

therefore analysed the Google Play Services app to identify the (obfuscated) function that carries out the decryption and wrote a `Frida` tool to dump out the encrypted input to this function (i.e. the encrypted `it` token) and the decrypted output (i.e. the decrypted `it` token). The decrypted tokens are of the form `ya29.m.`<base64 encoded protobuf> i.e. the same as the `authorization` bearer tokens. Figure 19 shows the decrypted `it` token sent in Figure 18. The binary tokenBytes entry matches that in Figure 16 although the HMAC hash is different.

The decrypted `it` tokens are stored in file files/authaccount/shared/IntermediateTokenStore.pb within the Google Play Services app's data folder /data/users/0/com.google.android.gms.

Table II lists the scopes observed being granted to Google apps by `Auth`. By extracting the invariant binary token from the `authorization` bearer tokens we can link together connections using the same authorization, and by then linking these to the `Auth` connection that originally sent the binary token we can find the app and scopes associated with the connections. Table III summarises the results of this analysis. Sometimes several scopes are given, this is because the same token is granted multiple scopes and we cnanot distiunguish which one of these is associated with a particular connection.

D. Apps Observed Making Firebase Connections

We observe the following Google apps making connections to https://firebaseinstallations.googleapis.com and android.apis.google.com/c2dm/register3:

com.android.vending (Google Play store), google.android.apps.messaging (Google Messages), google.android.apps.nbu.files (Google Files), google.android.apps.photos (Google Photos), google.android.apps.safetyhub (Google Personal Safety), google.android.apps.work.clouddpc (Google Enterprise Android Device Policy), google.android.calendar (Google Calendar), google.android.dialer (Google Phone/Dialer), google.

App	Granted auth scopes
Google Messages	https://www.googleapis.com/auth/android-messages,https://www.googleapis.com/auth/carrier-message-store
com.google.android.apps.pixel.dcservice	https://www.googleapis.com/auth/notifications
Google Play Services	https://www.googleapis.com/auth/accounts.reauth,https://www.googleapis.com/auth/numberer,https://www.googleapis.com/auth/cryptauth,https://www.googleapis.com/auth/account_settings_mobile,https://www.googleapis.com/auth/android_checkin,https://www.googleapis.com/auth/userlocation.reporting,https://www.googleapis.com/auth/sierra,https://www.googleapis.com/auth/android_platform_backup_restore,https://www.googleapis.com/auth/tachyon,https://www.googleapis.com/auth/subscriptions,https://www.googleapis.com/auth/notifications,https://www.googleapis.com/auth/games.firstparty,https://www.googleapis.com/auth/login_manager,https://www.googleapis.com/auth/locationhistory,https://www.googleapis.com/auth/peopleapi.legacy.readwrite,https://www.googleapis.com/auth/userinfo.email,https://www.googleapis.com/auth/nearbysharing-pa,https://www.googleapis.com/auth/lc.anonymous,https://www.googleapis.com/auth/spot,https://www.googleapis.com/auth/cclog,https://www.googleapis.com/auth/googleplay,https://www.googleapis.com/auth/account.service_flags,https://www.googleapis.com/auth/firebase.messaging,https://www.googleapis.com/auth/mobile_user_preferences,https://www.googleapis.com/auth/account.capabilities,https://www.googleapis.com/auth/peopleapi.readwrite,https://www.googleapis.com/auth/playatoms,emailhttps://www.googleapis.com/auth/gcm,https://www.googleapis.com/auth/semanticlocation.readonly,https://www.googleapis.com/auth/mobileapps.doritos.cookie,https://www.googleapis.com/auth/android_device_manager,https://www.googleapis.com/auth/googleplay.enterprise,https://www.googleapis.com/auth/peopleapi.readonly,https://www.googleapis.com/auth/auditrecording-pa,https://www.googleapis.com/auth/account.restricted_apps,https://www.googleapis.com/auth/webhistory,https://www.googleapis.com/auth/experimentsandconfigs,openid
Google Search app com.google.android.googlequicksearchbox	https://www.googleapis.com/auth/assistant,https://www.googleapis.com/auth/calendar,https://www.googleapis.com/auth/webhistory,https://www.google.com/accounts/OAuthLogin,https://www.googleapis.com/auth/drive,https://www.googleapis.com/auth/notifications,https://www.googleapis.com/auth/googlenow
Google Docs	https://www.googleapis.com/auth/discussions,https://www.googleapis.com/auth/drive.activity.readonly,https://www.googleapis.com/auth/drive.file,https://www.googleapis.com/auth/cloud_search.query,https://www.googleapis.com/auth/drive.labels.readonly,https://www.googleapis.com/auth/memento,https://www.googleapis.com/auth/drive,https://www.googleapis.com/auth/subscriptions,https://www.googleapis.com/auth/vouchers,https://www.googleapis.com/auth/activity,https://www.googleapis.com/auth/spreadsheets,https://www.googleapis.com/auth/peopleapi.readonly,https://www.googleapis.com/auth/drive.apps,https://www.googleapis.com/auth/drive.readonly,https://www.googleapis.com/auth/drive.metadata.readonly,https://www.googleapis.com/auth/docs,https://www.googleapis.com/auth/cloudprint,https://www.googleapis.com/auth/notifications,https://www.googleapis.com/auth/peopleapi.readwrite,https://www.googleapis.com/auth/gmail.readonly
Google Play store	https://www.googleapis.com/auth/experimentsandconfigs,https://www.googleapis.com/auth/voledevice,https://www.googleapis.com/auth/playatoms,https://www.googleapis.com/auth/googleplay
Gmail	https://www.googleapis.com/auth/peopleapi.readonly,https://mail.google.com/,https://www.googleapis.com/auth/espresso,https://www.googleapis.com/auth/gmail.readonly,https://www.googleapis.com/auth/notifications,https://www.googleapis.com/auth/drive,https://www.googleapis.com/auth/gmail.full_access,https://www.googleapis.com/auth/gmail.locker.read,https://www.googleapis.com/auth/calendar.readonly,https://www.googleapis.com/auth/gmail.ads,https://www.googleapis.com/auth/reminders,https://www.googleapis.com/auth/subscriptions,https://www.googleapis.com/auth/taskassist.readonly,https://www.googleapis.com/auth/chat,https://www.googleapis.com/auth/gmail.publisher_first_party
Google Calendar	https://www.googleapis.com/auth/calendar,https://www.googleapis.com/auth/tasks,https://www.googleapis.com/auth/notifications
Google Play store	profile,openid,https://www.googleapis.com/auth/userinfo.profile,https://www.googleapis.com/auth/userinfo.email,email
Google Videos	https://www.googleapis.com/auth/android_video,https://www.googleapis.com/auth/google_tv,https://www.googleapis.com/auth/userinfo.email,email,openid

TABLE II: Scopes observed granted to various Google apps by `Auth`

App	Scopes	Urls
Google Play Services	https://www.googleapis.com/auth/mobileapps.doritos.cookie	https://googleads.g.doubleclick.net/pagead/drt/m
Google Play Services	https://www.googleapis.com/auth/cryptauth	https://securitydomain-pa.googleapis.com,https://cryptauthenrollment.googleapis.com
Google Play Services	https://www.googleapis.com/auth/account.restricted_apps	https://restrictedapps-pa.googleapis.com
Google Play Services	https://www.googleapis.com/auth/googleplay.enterprise	https://android.googleapis.com/auth/enterpriser
Google Play Services	https://www.googleapis.com/auth/login_manager	https://chromesyncpasswords-pa.googleapis.com
Google Play Services	https://www.googleapis.com/auth/peopleapi.legacy.readwrite	https://people-pa.googleapis.com
Google Play Services	https://www.googleapis.com/auth/sierra	https://wallet.google.com
Google Play Services	https://www.googleapis.com/auth/account.capabilities,openidhttps://www.googleapis.com/auth/userinfo.email,email,https://www.googleapis.com/auth/account.service_flags	https://android.googleapis.com/auth/lookup/account_state
Google Play Services	https://www.googleapis.com/auth/accounts.reauth	https://android.googleapis.com/auth/reauthsettings
Google Play Services	https://www.googleapis.com/auth/spot,https://www.googleapis.com/auth/android_device_manager	https://spot-pa.googleapis.com
Google Play Services	https://www.googleapis.com/auth/semanticlocation.readonly	https://semanticlocation-pa.googleapis.com
Google Play Services	https://www.googleapis.com/auth/cclog	https://play.googleapis.com/log/batch
Google Play Services	https://www.googleapis.com/auth/lc.anonymous	https://quake-pa.googleapis.com
Google Play Services	https://www.googleapis.com/auth/userlocation.reporting	https://userlocation.googleapis.com
Gmail	gmail scopes	https://inbox.google.com,https://taskassist-pa.googleapis.com,https://mail.google.com,https://inbox.google.com
Google Play Services	https://www.googleapis.com/auth/tachyon	https://instantmessaging-pa.googleapis.com
Google Play Services	https://www.googleapis.com/auth/nearbysharing-pa	https://nearbysharing-pa.googleapis.com
Google Play store	https://www.googleapis.com/auth/experimentsandconfigs	https://www.googleapis.com/experimentsandconfigs
Google Play store	https://www.googleapis.com/auth/playatoms	https://playatoms-pa.googleapis.com
Google Play store	https://www.googleapis.com/auth/googleplay	https://play-fe.googleapis.com, https://play.googleapis.com/play/log, https://android.clients.google.com/fdfe/api/userProfile
Google Search app com.google.android.google quicksearchbox	https://www.googleapis.com/auth/assistant	https://proactivebackend-pa.googleapis.com
Google Search app com.google.android.google quicksearchbox	https://www.googleapis.com/auth/webhistory	https://geller-pa.googleapis.com
Google Search app com.google.android.google quicksearchbox	https://www.googleapis.com/auth/drive,https://www.googleapis.com/auth/assistant,https://www.googleapis.com/auth/calendars	https://proactivebackend-pa.googleapis.com
Google Calendar	https://www.googleapis.com/auth/tasks	https://tasks-pa.googleapis.com
Google Docs	https://www.googleapis.com/auth/notifications	https://notifications-pa.googleapis.com

TABLE III: For each binary access token, the app, scopes and urls observed sending the token (as an authorization/bearer header).

android.gm (Gmail), google.android.gms (Google Play Services), google.android.googlequicksearchbox (Google Search app), google.android.videos (Google Videos), google.android.wfcactivation, google.android.apps.pixel.dcservice, google.android.apps.work.oobconfig, google.android.carrier, google.android.devicelockcontroller, google.android.euicc, google.android.apps.scone