

What Android Auto shows

A static reverse engineering of **com.google.android.projection.gearhead 17.9.664044** as redistributed by GrapheneOS, and what your phone hands to a car it does not control.

82

PERMISSIONS

55

EXPORTED COMPONENTS

3

TRUST DOMAINS

01 Executive summary

03 Anatomy of the app

05 What leaves the phone

07 Where it goes

09 Recommendations

11 Evidence

02 Provenance and method

04 Three trust domains

06 When it happens

08 The GrapheneOS delta

10 Beyond settings

12 Support

The car is a separate party that receives the phone's content.

17.9.664044

Executive summary

Android Auto is a bridge and collects little itself. Its own network footprint is small. It moves the phone's most personal data across a boundary the user does not control, onto a screen owned by a car manufacturer, and keeps a durable registry of every car it has ever trusted.

82 PERMISSIONS DECLARED	55 EXPORTED COMPONENTS	26,427 DECOMPILED CLASSES	65 LITERAL HOSTNAMES
16 DANGEROUS PERMISSIONS	31 NEVER GRANTABLE	26 MB DEX BYTECODE	1 NOTIFICATION LISTENER

The five main findings

- The head unit sees everything the phone renders, including your notifications.** Auto declares an exported `NotificationListenerService`, so once notification access is granted it can read every notification on the device: messages, banking alerts, one-time codes. The car receives what the phone draws. SEEN
- It keeps a persistent registry of cars, with credentials.** The `allowedcars` table stores manufacturer, model, model year, the head unit's vehicle ID, a locally generated random client ID, the Bluetooth address, WiFi SSID/BSSID and the car's WiFi password, plus the head unit's TLS certificate subject and serial. Storing the passphrase is what a wireless reconnect needs, and the database sits in the app's private storage, which other apps cannot read on GrapheneOS. SEEN
- Its own telemetry rides on Play services.** Auto vendors Google's Clearcut client and logs under the `CAR.ANALYTICS` source, but ships no upload endpoint of its own, so events go through the GMS binder. That means the analytics channel survives the sandbox exactly as it does for Play services. SEEN
- What Google gets directly is inherited from bundled SDKs.** Maps SDK telemetry, network location (`clients4.google.com/glm/mmap`), an ads ping (`gen_204?id=gmob-apps`), Assistant voice over gRPC and Phenotype flags. Almost none of it is Auto-specific. It comes with embedding Google's SDKs. SEEN
- The sandbox helps only for permissions you refuse.** 31 of the 82 permissions are signature/privileged and never land. The rest (SMS, contacts, call log, calendar, location, microphone) are runtime prompts you can refuse, at the cost of the features that need them. INFERRED

What can be done about it. Section 09 lists settings that exist today, and the biggest gain comes from running Auto in its own profile. Section 10 goes further: network filtering, disabling components, modifying the APK and patching GrapheneOS. Auto's own network footprint is small, so the filters buy little. Android already lets you filter what Auto's listener reads, by category and by app, so the most useful patch left is a per-app domain allowlist in the OS. Most of the rest was checked by opening the screens on a device, and none of it was tested by changing settings.

Scope. Static analysis of the APK only; no live capture and no head unit was attached. SEEN means the claim resolves to a file and line in the appendix. INFERRED means reasoning over that evidence. What a car manufacturer does with the pixels it receives is outside any APK's reach, so the report does not speculate about it.

02

Provenance and method

The artifact is the official, Google-signed Android Auto from GrapheneOS's own repository. Every byte was checked against the repository's signed metadata before anything was read.

ITEM	VALUE
Repository	https://apps.grapheneos.org/ · metadata.1.0.0.sjson
Package	com.google.android.projection.gearhead · versionCode 179664044 · 17.9.664044-release
Channel / SDK	stable · minSdk 32 · arm64-v8a only
base.apk	35,123,449 bytes · sha256 115228a7af416c4c... (matches metadata)
Signer	CN=Android, O=Google Inc. · sha256 1ca8dcc0bed3cbd8...b65b5295 (matches metadata)
Splits	config.arm64_v8a (8.8 MB, native libs) · config.xxhdpi · config.en
Inside	4 DEX · 26 MB bytecode · 148,023 strings · 543 assets · 8.7 MB native (arm64 split)

Repository note, quoted. "Official release of Android Auto without modifications. After installation, Android Auto has to be set up from *Settings > Apps > Sandboxed Google Play > Android Auto*." That toggle grants the projection profile to a sandboxed app.

What was done

Unpack	apktool for manifest and resources; raw zip for assets and native libraries
Strings	DEX string tables parsed directly (148,023 entries), hosts and URLs extracted from them

Code	jadx 1.5.6 full sweep: 26,427 classes, 176 MB of Java, then targeted reads of projection, messaging, telecom, telephony, telemetry and the car database
Policy	permission protection levels diffed against GrapheneOS <code>frameworks/base</code> branch 17
Scope	base + 3 config splits; GMS and Play Store only as context for the binder channel

What could not be established

- What the head unit records. The car is a black box by definition; this report only covers what is sent to it.
- Exact wire format of the projection link. TLS is used and the car's certificate is pinned, but the payload is a proprietary protocol carried over USB accessory / TCP.
- Whether the car's WiFi password column is encrypted elsewhere before insert. The insert path stores the value directly, but a wrapper could encrypt it (none was found). Under the GrapheneOS sandbox and device storage encryption this is a design necessity, not a finding.
- Any Google-side retention or joining.

03

Anatomy of the app

Auto is leaner than Play services, with 196 components instead of 1,310, but the permissions it asks for are the ones that matter. Auto is a very well-connected UI.

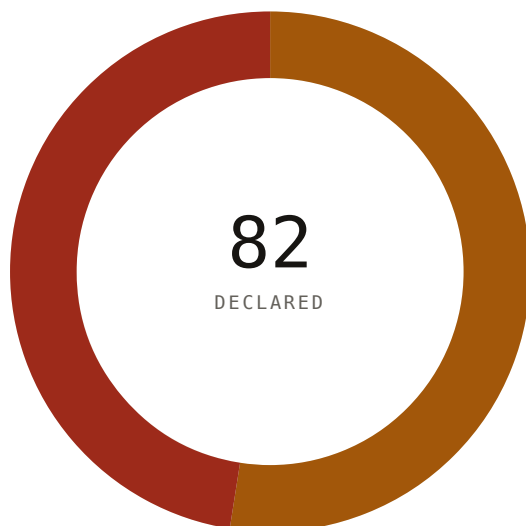


Figure 1: 82 permissions: can they ever be granted?

The permission profile

Only 31 are signature/privileged and 8 are outside AOSP. The bulk, 23 normal and 16 dangerous, is the set a driving UI needs and the set a user can hand over.

Three permissions are absent: `AD_ID`, `READ_SMS` (only receive and send are declared) and background location. The dangerous permissions cover SMS, contacts, call log, calendar, fine location, microphone and phone state.



■ normal · 23 ■ dangerous · 16 ■ special · 4 ■ signature/privileged · 31 ■ not in AOSP · 8

Figure 2: the same 82 permissions by protection level

Surface area

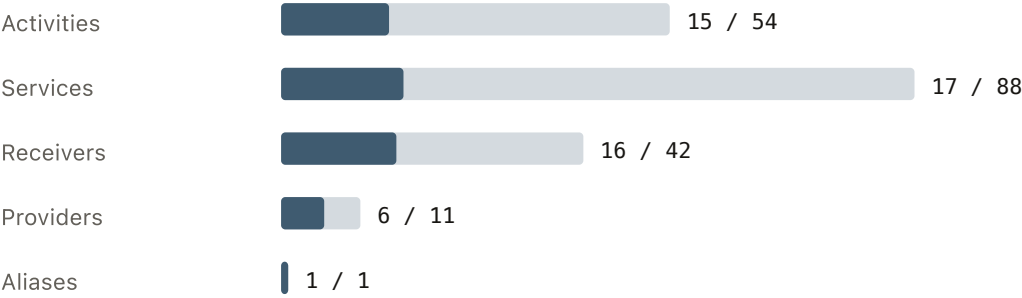


Figure 3: components declared vs. exported (exported / total)

196 components, 55 exported. The listeners that give Auto its reach matter more than the activities: a notification listener, an in-call service, a media browser, a companion-device service and the chimera bridge into Play services.

```
AndroidManifest.xml
com.google.android.gearhead.notifications.SharedNotificationListenerManager$ListenerService
    android:permission="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE"
com.google.android.apps.auto.components.telecom.service.CarProjectionInCallServiceImpl
com.google.android.apps.auto.components.ace.AceInsightRendererService    (Android 16 personal context)
com.google.android.apps.auto.carservice.gmscorecompat.CarChimeraService
    action="com.google.android.gms.car.service.START"
```

Main exposure. A notification listener reads the content of every notification from every app, including apps that deliberately do not expose their content otherwise. Auto projects notifications to the car by design. The privacy question is whether the car is a trustworthy display. [SEEN](#)

Manifest settings

SETTING	WHAT IT MEANS HERE
QUERY_ALL_PACKAGES	Declared. A normal permission, so Auto can list every installed app without a prompt. 16 decompiled files query packages or intents. SEEN
allowBackup="true"	A <code>SettingsBackupAgent</code> is registered. Its specification names the <code>carservice</code> and <code>common_user_settings</code> preference files and launcher keys. The <code>allowedcars</code> database is not in that list, so the stored Wi-Fi passwords do not appear to travel through this agent. Whether any other data is included is not visible here. SEEN · UNKNOWN
cleartextTrafficPermitted="true"	Cleartext HTTP is permitted for all domains. None of the 63 URLs found starts with <code>http://</code> . SEEN
sharedUserId	<code>com.google.android.projection.gearhead</code> . All packages in this family run as one Linux user. SEEN
special access	<code>SYSTEM_ALERT_WINDOW</code> , <code>WRITE_SETTINGS</code> and <code>MANAGE_EXTERNAL_STORAGE</code> are declared. SEEN

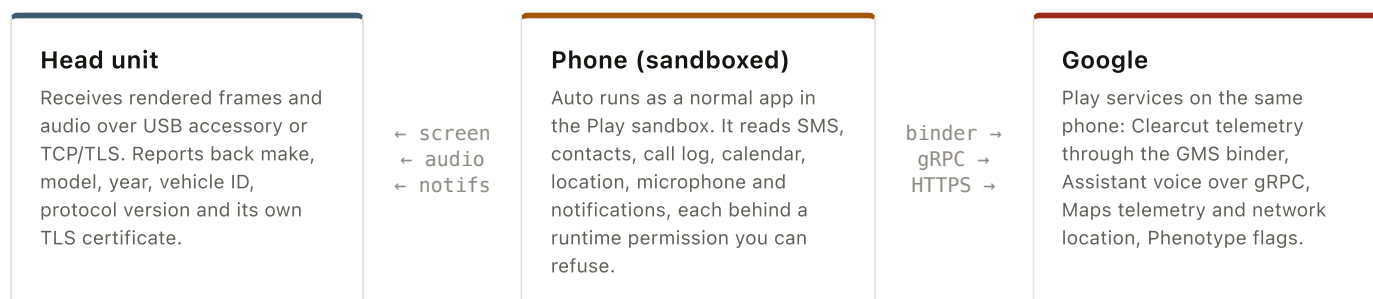
exported providers

6 of 11 are exported. `SharedPreferencesProvider` has no manifest permission but rejects callers in code unless they share Auto's UID or belong to a Google-signed package (`SharedPreferencesProvider.java` , method `b`), and it allows reads and writes of preference values. The microphone provider checks `RECORD_AUDIO` and the calling package. The projection-state, troubleshooter and colour providers are exported without a permission and were not audited in detail. [SEEN](#)

04

Three trust domains

Every privacy claim about Android Auto is a claim about one of three domains. Confusing them is how people end up thinking "it's just a display".



The phone is the only party that talks to both. The car never reaches Google directly; Google never sees the car's screen.

What Auto remembers about a car

The most revealing artifact in the app is a SQLite table called `allowedcars` , written every time a head unit is trusted.

```

defpackage/sqm.java:24 (schema, verbatim)
CREATE TABLE allowedcars (
  id, manufacturer, model, modelyear, vehicleid, vehicleidclient,
  headUnitProtocolMajorVersionNumber, headUnitProtocolMinorVersionNumber,
  bluetoothConnectionAllowed, connectiontime, nickname,
  bluetoothaddress, wifissid, wifibssid, wifipassword, wifisecurity,
  wifiProjectionProtocolOnTcp, wifiProjectionProtocolIpAddress,
  wifiProjectionProtocolTcpPort, headUnitMake, headUnitModel,
  headUnitSoftwareBuild, headUnitSoftwareVersion, vehicleType)
  
```

```

defpackage/jtb.java:34          contentValues.put("wifipassword", ...) // stored as TEXT
com/google/android/gms/carsetup/CarInfoInternal.java:107
... bluetoothaddress=..., wifissid=..., wifibssid=..., sslSubjectName=...,
  sslSerialNumber=..., hashedVehicleId=[REDACTED] ...
  
```

01

Vehicle ID

Reported by the head unit and used to match a car across connections. Typically a VIN-class identifier; the exact source is protocol-defined. [SEEN](#) (field) · [INFERRED](#) (VIN)

02

Client vehicle ID

Generated locally as `Long.toHexString(new SecureRandom().nextLong())` and stored next to the car. This is Auto's own pseudonym for that vehicle, independent of what the car reports. SEEN `jtb.java:128`

03

Head-unit certificate

`sslSubjectName` and `sslSerialNumber`: the TLS identity presented by the car, used to pin trust. It is also a stable fingerprint of that specific head unit. SEEN `CarInfoInternal`

04

Wireless credentials

SSID, BSSID and the hotspot password for wireless projection, plus the TCP endpoint when the car speaks projection-over-TCP. Persistent across sessions until the car is forgotten. SEEN

05

Analytics session

Each projection session gets an ID that travels with Clearcut events, so Google can group telemetry by drive rather than by day. SEEN `AnalyticsSessionInfo`

Rental-car problem. Pairing a rental stores its Bluetooth MAC, WiFi credentials and certificate serial in `allowedcars`, and gives the rental's screen access to your notifications for the duration. Forgetting the car is a privacy control.

05

What leaves the phone

Two tables follow, one for each recipient. The car and Google see different data, and conflating them is the most common mistake in discussions of Android Auto.

To the head unit

CHANNEL	WHAT CROSSES THE BOUNDARY	GATED BY
Screen	Rendered frames of the projected UI: navigation, messages, contacts, media, settings: whatever is on screen at that moment.	always (that is the product)
Notifications	Content of notifications from every app: senders, text, actions. Sensitive notifications included unless the OS redacts them.	notification access, granted by the user
Messages	SMS/MMS threads, sender identity, message bodies; replies typed or dictated in the car.	RECEIVE_SMS, SEND_SMS, READ_CONTACTS
Contacts	Names, numbers, photos, favourites, call history for the dialer.	READ_CONTACTS, READ_CALL_LOG

CHANNEL	WHAT CROSSES THE BOUNDARY	GATED BY
Calls	Call state, caller ID, audio routed through the car's microphone and speakers.	CALL_PHONE, in-call service
Calendar	Events and attendees for the calendar app.	READ_CALENDAR
Media	Track metadata, album art, browse trees, playback control.	media browser
Telephony status	Signal strength and cell info, unless the head unit hides it.	READ_PHONE_STATE
Location	Position used to render navigation; the car receives the map, not raw GPS, unless it asks.	fine location

To Google

CHANNEL	ENDPOINT (AS SEEN)	WHAT LEAVES
Clearcut via GMS binder	(no local uploader)	Events under the <code>CAR.ANALYTICS</code> log source: connection lifecycle, protocol errors, audio diagnostics, Bluetooth FSM states, permission denials, feature flags. Carries the analytics session ID and, through GMS, the Android ID and account when present.
Assistant	assistant-maps.google.com gRPC via GSA	Voice audio for every "Hey Google" in the car, plus the transcript context needed to answer. This is the largest content stream, and it is opt-in by use.
Maps / navigation	mobilemaps-pa.googleapis.com mapsmobilesdfs- pa.googleapis.com autopush-navigationsdkusage...	Map usage, navigation sessions, search queries, Places lookups, SDK performance metrics.
Network location	clients4.google.com/glm/mmap	Cell/WiFi fingerprints sent for geolocation when GPS is unavailable. This is the classic Google location API.
Ads ping	pagead2.googlesyndication. com/pagead/gen_204?id=gmob- apps	A lightweight ad-SDK beacon from the bundled mobile ads library. Auto declares no AD_ID permission, so the advertising ID is not available to it.
Config	phenotype (via GMS)	Flag fetches and experiment tokens; the same channel described in the Play services report.
Attribution	InstallReferrerReceiver	Install referrer on first run, showing how the install was acquired.
Feedback	feedback-pa.googleapis.com	User-initiated bug reports and ratings, including screenshots if attached.

Asymmetry. Google gets telemetry and voice; the car gets content. The car sees your messages, and Google sees how the app behaved while showing them. The two carry different risks and have different remedies.

Does Android Auto use Play Integrity itself?

No client call was found. The decompiled code has no import of a DroidGuard or Play Integrity client and no call to an integrity token API. The word `DR0IDGUARD` appears only as a name: an entry in a request-header list (`qda.java:20` using `vji.DR0IDGUARD`), three flag names (`qgw.java:102-104`) and an error label (`qyv.java:67`), all inside bundled library code. SEEN My reading is that this is the vendored account and request-signing library carrying the field, with the actual call made by Play services if at all. INFERRED Android Auto still depends on the Play services car module, and Play services do consume attestation for their own features, as the Play services report describes.

06

When it happens

Auto is event-driven. Almost nothing happens on a timer; almost everything happens when a cable is plugged in or a Bluetooth link comes up.

USB attach	Accessory handshake. AOA negotiation (<code>USB_ACCESSORY_ATTACHED</code> , <code>ACCESSORY_HANDSHAKE</code>), protocol version exchange, then projection. SEEN
BT connect	Wireless trigger. ACL/headset connection events wake the wireless setup path; the saved car is matched by Bluetooth address and certificate. SEEN
projection start	Session opens. An analytics session ID is minted; the car record is created or updated in <code>allowedcars</code> . SEEN
while driving	Streams. Video frames, audio, notifications, telephony state and location flow continuously. Clearcut events queue as they occur. SEEN
boot / update	Re-arm. Component init on <code>BOOT_COMPLETED</code> and <code>MY_PACKAGE_REPLACED</code> ; FRX recovery receiver; Phenotype flag update. SEEN
first install	Attribution. Install referrer is consumed, then the app waits for setup. SEEN
on demand	Voice. Microphone opens only for an Assistant session, then closes. SEEN

No idle beacon. Unlike Play services, Auto has no 12-hour check-in of its own. Its background footprint is close to zero until a car appears, so the "when" question for Auto comes down to what was plugged in.

Where it goes

The whole APK has only 65 literal hostnames, an order of magnitude fewer than Play services, which fits an app that delegates its cloud work.

HOST	ROLE
<code>www.googleapis.com</code>	generic Google API front door used by bundled libraries
<code>clients4.google.com/glm/mmap</code>	network location (cell/WiFi to coordinates)
<code>mobilemaps-pa.googleapis.com</code>	Maps telemetry channel
<code>mapsmobilesdks-pa.googleapis.com</code>	Maps SDK usage metrics
<code>autopush-navigationsdkusage.sandbox.googleapis.com</code>	Navigation SDK usage telemetry
<code>assistant-maps.google.com</code>	Assistant integration for maps/navigation
<code>pagead2.google syndication.com</code>	ads SDK beacon (<code>gen_204</code>)
<code>www.google-analytics.com</code> · <code>ssl.google-analytics.com</code>	classic Google Analytics endpoints
<code>feedback-pa.googleapis.com</code>	user feedback and bug reports
<code>play.google.com</code>	store links, update prompts
<code>support.google.com/androidauto</code>	help articles opened in WebView
<code>developers.google.com</code> · <code>console.developers.google.com</code>	Maps API key setup links shown to users
<code>goo.gl</code>	short links in help text

`play.googleapis.com` does not appear, which indicates there is no Clearcut uploader in Auto. The `PlayLoggerContext` class is vendored, but the transport is the GMS binder, as the `CarTelemetryLogger` signature and the `com.google.android.gms.car.service.START` bridge show.

Reading the list. A hostname in the APK does not mean traffic. Several of these are reachable only from bundled SDKs when a feature is used (navigation, voice, feedback). The list is an upper bound on capability and does not describe steady-state traffic.

The GrapheneOS delta

On GrapheneOS, Auto runs inside the Play sandbox. This changes content exposure less than expected and changes privileges more than expected.

CAPABILITY	STOCK AOSP	GRAPHENEOS SANDBOX
Read SMS / contacts / call log	via runtime prompts	same, but each is independently revocable and audited
Notification access	user toggle	same; nothing grants it silently
Capture secure video / add trusted display	system app	denied (signature/privileged)
Manage USB / tethering	privileged	denied for the privileged paths; accessory mode still works
Bluetooth privileged operations	yes	denied; normal BLUETOOTH_CONNECT/SCAN apply
Read privileged phone state	yes	denied; READ_PHONE_STATE prompt only
Network access	always	revocable for most apps, but locked on for Auto ("Device requires this permission to operate")
Local network (wireless projection)	normal	prompted : ACCESS_LOCAL_NETWORK is dangerous

What the sandbox removes

- 31 signature/privileged permissions: secure video capture, trusted displays, virtual devices, privileged Bluetooth, privileged phone state.
- Silent background privileges; Auto competes with other apps.
- The ability to make itself a system component of the car experience.

What it does not remove

- The three-domain flow: content still goes to the car, telemetry still goes to Google.
- Clearcut through the GMS binder.
- Assistant voice, Maps telemetry, network location.
- Anything you grant at runtime: SMS, contacts, notifications, location, microphone.

Limits of the sandbox. Sandboxing Auto turns a privileged car component into an app with the same rights as any other app. For a product whose job is to hand your data to a third-party screen, that reduces the blast radius, but it does not make the exposure private. The data still goes to the car.

Three user profiles

PROFILE	WHAT IS EXPOSED	RISK
Wired only, messaging denied	Navigation and media to the car; Maps telemetry and Assistant (if used) to Google. No messages, contacts or notifications cross.	LOW content stays on the phone

PROFILE	WHAT IS EXPOSED	RISK
Wireless + messaging + notifications	Full projection: messages, contacts, notifications, calls. Car credentials stored. Telemetry per drive.	MEDIUM the car becomes a data recipient
Rental / shared car, everything granted	All of the above, on hardware you do not own or trust, with its credentials saved on your phone afterwards.	HIGH third-party screen with your notifications

09

Recommendations

Ordered by what each step buys you per unit of inconvenience. None of them requires giving up navigation, and every item here is a setting or feature that exists today. Tags show what a claim rests on: **SEEN** in the APK, on a GrapheneOS device or in the GrapheneOS usage guide, **INFERRED** from what was seen, **UNTESTED** not tried on a live device.

Where to click

Menu names below were checked on a GrapheneOS device (build 2026102101, Android 17) by opening the screens, not by changing them. The hub is **Settings > Apps > Sandboxed Google Play > Android Auto**. It holds *Android Auto settings*, a *Permissions* group with four switches, *Android Auto app info*, *Notification access settings* and a list of optional dependencies (Google Maps, Speech Recognition & Synthesis, Google app). **SEEN**

GOAL	PATH	SET IT TO
Choose which permission bundles Auto gets	Android Auto > Permissions: <i>Allow permissions for wired Android Auto, for wireless Android Auto, Allow to reroute audio from device to the car, Allow permissions for handling phone calls</i>	Switch off the bundles you do not use (for example wireless, if you only plug in). I did not flip them, so what each one revokes is UNTESTED .
Limit which notifications the car can read	Android Auto > Notification access settings > <i>Allow notification access</i> , then the categories <i>Real-time, Conversations, Notifications, Silent</i> , then <i>See all apps</i> (screen title "Apps shown on device")	Untick the categories and every app you do not want on a car screen. One checkbox per app.
Keep notifications away from Google	Android Auto settings > Messaging > <i>Notifications with Assistant</i> ("Allow Android Auto to send your notifications to Google for features like playing your messages")	Off, unless you want Assistant to read messages aloud
Less message text on the screen	Android Auto settings > Messaging > <i>Show message notifications, Show group conversations, Show first line of messages</i>	Off for the ones you do not need
No phone features on a locked phone	Android Auto settings > Startup > <i>Start Android Auto while locked</i> ("Access to your phone features will be available on the vehicle screen while locked")	Off
Cars Auto remembers	Android Auto settings > Connection > <i>Vehicles</i> (lists <i>Accepted vehicles</i> and <i>Rejected vehicles</i>)	Open an entry to remove cars you no longer use. I did not open an entry, so the exact wording of the remove action is UNTESTED .

GOAL	PATH	SET IT TO
No setup prompt for unknown cars	Same screen, <i>Add new vehicles to Android Auto</i>	Off while you are not adding a car (rentals)
Contacts without giving them	Android Auto > Android Auto app info > Permissions > <i>Contacts and accounts</i> > <i>Contact Scopes</i>	Enable it instead of <i>Allow</i>
Microphone	Android Auto app info > Permissions > Microphone	<i>Don't allow</i> , if you never use voice
Auto's own network access	Android Auto app info > Permissions > Network	Cannot be changed: <i>Allow</i> and <i>Don't allow</i> are greyed out with "Device requires this permission to operate". Use Play services' Network switch instead (Play services report, "Where to click").
Separate profile	Settings > Security & privacy > Privacy > Private space	Set Auto up there

1. Put Auto in its own profile

The GrapheneOS guide says Android Auto can be set up in a secondary profile or in Private Space, from *Settings* > *Apps* > *Sandboxed Google Play* > *Android Auto* SEEN. A notification listener only receives the notifications of the user it runs in, and a package query only sees that user's packages INFERRED. Keep only the navigation and media apps in that profile and the car cannot see anything from your main profile. The cost: a messaging app you want in the car has to live in the same profile.

2. Narrow what the notification listener can read

The listener turns the car into a reader of your phone's most sensitive feed, one-time codes and banking alerts included. Android already lets you narrow it without a patch: the *Notification access settings* screen of Auto has four category checkboxes and a per-app checklist. SEEN (on a device) Keep the list to the few apps you want on the dash. Do not rely on this alone for banking or authenticator apps: the safest setting for them is a profile Auto cannot see (item 1). INFERRED

3. Stop notifications going to Google

The *Notifications with Assistant* switch lets Auto send your notifications to Google for features such as reading messages aloud. Turn it off unless you use that feature. SEEN (on a device, wording quoted above)

4. Grant SMS and contacts only if you message from the car

These unlock the dialer and the messaging UI. Without them Auto still navigates, plays media and takes voice commands. The GrapheneOS Contact Scopes feature gives an app the impression that it has the Contacts permission while returning nothing, and blocks writes SEEN (usage guide). Calling by name from the car will not work. INFERRED

5. Forget cars you no longer use, and do not accept new ones by default

The `allowedcars` table keeps the vehicle ID, Bluetooth address, Wi-Fi credentials and certificate serial. Before you sell the phone or hand back a rental, remove the car in the vehicle list. Unplugging does not remove the record. SEEN Turn off *Add new vehicles to Android Auto* when you are not setting up a car, so an unknown head unit cannot start the setup flow. INFERRED

6. Prefer wired projection, and switch off the wireless bundle

Wired projection stores no hotspot credentials. The manifest declares `ACCESS_LOCAL_NETWORK`, `NEARBY_WIFI_DEVICES`, `CHANGE_WIFI_STATE` and Bluetooth scan and connect for the wireless path. SEEN On GrapheneOS these permissions are granted as system-fixed for Auto and appear as bundles in the four switches above, not as individual prompts. SEEN (on a device) Expect wireless projection to stop working if you switch the wireless bundle off. INFERRED

7. The network switch works on Play services, not on Auto

Auto's own Network permission is locked on. Turning off the network for Play services stops Assistant, Maps traffic and telemetry that go through it, and it also breaks navigation that needs live data. Use it as a per-device policy rather than a general privacy switch. SEEN (locked permission) · INFERRED (effect)

8. Refuse the special access Auto asks for but does not need

The manifest declares `SYSTEM_ALERT_WINDOW`, `WRITE_SETTINGS` and `MANAGE_EXTERNAL_STORAGE`. Projection works without them, so if a support page asks you to grant them, question it. SEEN (declarations) · INFERRED (not needed)

Wired still exposes the screen. The cable carries the same rendered screen as the wireless link. Changing the transport changes which credentials are stored. The same screen reaches the car either way.

10

Beyond settings: blocking, patching

Settings stop where Google chose to expose a control. This section goes further: filtering the network, disabling components, modifying the APK, and patching GrapheneOS. They are engineering proposals built on the host and component inventories in this report. **None of them was tried on a device.** Android Auto is a small app that leans on Play services for its telemetry, so this section also points to the matching steps in the Play services report.

10.1 Filter the network from outside

No patch, fully reversible, and the APK stays Google-signed. Use Private DNS with a filtering resolver, or a firewall VPN installed only in the profile that holds Auto (Android VPNs are per user INFERRED). A DNS or SNI filter sees hostnames, never paths.

HOST	ROLE	BLOCK?	CONSEQUENCE
pagead2.googlesyndication.com	ads SDK beacon SEEN	YES	Nothing visible. Auto has no ad surface.
www.google-analytics.com, ssl.google-analytics.com	classic analytics SEEN	YES	Nothing visible.

HOST	ROLE	BLOCK?	CONSEQUENCE
autopush-navigationsdkusage.sandbox.googleapis.com	navigation SDK usage SEEN	YES	A staging endpoint. Nothing visible.
feedback-pa.googleapis.com	feedback and bug reports	YES	Feedback is not sent.
clients4.google.com	network location (glm/mmap) SEEN	OPTIONAL	GrapheneOS reroutes Play location requests to its own service by default (usage guide), so Auto may not need it. UNTESTED
mobilemaps-pa.googleapis.com, mapsmobilesdfs-pa.googleapis.com	Maps telemetry and SDK usage	RISKY	May be the same hosts navigation uses. Test before relying on it.
assistant-maps.google.com	Assistant for maps	IF NO VOICE	Voice commands fail.
www.googleapis.com	shared API host	NO	Used by bundled libraries and by Play services.

Cleartext is allowed, no use found. The network security config sets `cleartextTrafficPermitted="true"` for every domain, yet none of the 63 URLs found in the APK starts with `http://`. So the capability is there and no use of it was found. **SEEN** The telemetry that matters goes through Play services, so the filter in section 10.1 of the Play services report applies to it as well.

10.2 Disable components over adb, no patch

With USB debugging on, the shell can switch individual components off per user, for example `adb shell pm disable --user <id> com.google.android.projection.gearhead/<component>`. **UNTESTED**

COMPONENT (ALL SEEN IN THE MANIFEST)	WHAT IT STOPS	LIKELY COST
gearhead.telemetry.InstallReferrerReceiver	install attribution	None
gearhead.feedback. CrashReporterReceiver\$CollectFeedbackService, components.feedback.*	crash and feedback collection	None
auto.components.ace.AceInsightRendererService	personal-context insights (Android 16)	That feature
libraries.assistant.auto.aaptransport.callbacks. CarAssistantCallbacksService	Assistant callbacks	Voice in the car
gearhead.notifications. SharedNotificationListenerManager\$ListenerService	notification projection	Notifications on the dash. Revoking notification access does the same job more simply.

Telemetry. Auto's events leave through the Play services binder, so the components that matter for it are in Play services (see section 10.2 of that report). The same warning applies: Play services can toggle its own components back.

10.3 Modify the APK

Is it possible at all? Technically yes. It is an ordinary signed APK and nothing in this analysis stops you from unpacking, editing and signing it again. I did not test an install, and I did not look for a self-integrity check inside the app. The practical barriers, from this analysis:

BARRIER	WHY IT MATTERS
Install route	The GrapheneOS guide says Android Auto cannot be installed through the Play Store or other sources, only through the GrapheneOS App Store, and that it depends on sandboxed Google Play. SEEN (usage guide) A re-signed build is no longer an App Store package and no longer updates there. INFERRED
Shared UID	The manifest declares <code>sharedUserId="com.google.android.projection.gearhead"</code> , so every package in that family has to be signed with the same key. SEEN
Pinned certificates	The Play services code that pins package and certificate pairs lists the Play Store, the Google app, Fitness and Wear, but not <code>com.google.android.projection.gearhead</code> . SEEN That removes one barrier that applies to Play services itself. The car-related parts of Play services may check callers some other way. UNKNOWN
Compatibility layer	GrapheneOS runs Android Auto through an extension of its Play compatibility layer with reduced privileges. I did not check how that layer recognises Auto. If it keys on the signer, an OS patch is needed as well. UNTESTED
The car	The phone pins the head unit's certificate. Whether the head unit checks anything about the phone side's signature is not visible in the APK. A re-signed build might be refused by some cars. UNKNOWN
Obfuscation churn	Class names are renamed per build (<code>jtb</code> , <code>plo</code> , ...), so a smali patch has to be found again on every release. Public component names are stable. INFERRED

The only patch that survives version churn is a manifest-only one: set `android:enabled="false"` on the components in 10.2 with a binary manifest editor, leave the DEX untouched, zipalign, and sign the base and the three splits with your own key. UNTESTED Anything that changes behaviour inside the code, such as wrapping the stored Wi-Fi password with a Keystore key, means a smali patch per release.

Does Google's server judge a modified Auto?

Less than it does for Play services. Android Auto itself makes no attestation call: no DroidGuard or Play Integrity client is imported anywhere in the decompiled code (section 05). Its server contacts are Maps, Assistant and configuration traffic, and its telemetry goes through Play services. A re-signed Auto would not be checked by Auto itself. What it would meet is the Play services car module that it talks to, and whether that module accepts an unfamiliar signer is unknown. UNKNOWN The attestation risk for the phone as a whole is described in the Play services report, section 10.3.

10.4 Patch GrapheneOS

The OS is where the signature stays intact and the change survives updates. You would need to build and sign your own GrapheneOS, which means your own verified-boot key and no official over-the-air updates. I did not check which of these ideas already exist upstream. Check that and raise an issue before building anything.

PROPOSAL	WHERE	EFFECT	EFFORT / RISK
A. Notification filter	Notification access settings of Auto	Already there: four category checkboxes and a per-app list ("Apps shown on device"). Not a patch, so it moved to section 09. A finer version, for example filtering by conversation, would be a patch. SEEN (on a device)	None for the existing screen.
B. Package scopes	Package visibility filter	<code>QUERY_ALL_PACKAGES</code> is declared and 16 decompiled files query packages or intents SEEN . Treat it as not held, as proposed for Play services.	Medium. Shared with the Play services proposal.
C. Per-app domain allowlist	Network stack, keyed on UID	Allow Maps and Assistant hosts, drop analytics and ads. Same mechanism as for Play services.	High. Hostnames only.
D. Car credential purge	Profile and backup lifecycle	Clear Auto's car records when a profile is wiped or a user logs out, so Wi-Fi credentials do not outlive the profile. INFERRED	Low to medium. Needs a hook, not a patch to Auto.

My order: C, then B, then D. A already exists. C needs no cooperation from Google and no change to Auto's code.

10.5 Which one to pick

OPTION	PRIVACY GAIN	BREAKAGE	EFFORT	UPKEEP
Settings and own profile (09)	High for notifications	Low	Minutes	None
Network toggle	Full, offline	Navigation, voice	Seconds	None
DNS / VPN filter (10.1)	Low to medium	Low	Hours	List upkeep
adb disable (10.2)	Low	Low	Hours	After updates
Manifest-patched APK (10.3)	Low	Medium, unknown for cars	Days	Every release
OS patch (10.4)	High	Low	Weeks	Every OS release

Recommended path. Start with a dedicated profile, the notification filter set to the few apps you want on the dash, the Assistant switch off, and cars forgotten after use. The filters and component switches buy little for Auto because its own network footprint is small. If you are prepared to maintain a fork, build the per-app domain allowlist (10.4 C). Skip repackaging Auto: the gain is small and the cars are an untested variable.

11

Evidence

Every SEEN claim resolves here. Paths are as jadx 1.5.6 names them for the APK in the artifact index, unless marked as manifest or strings.

Not published. The decompiled sources, the APKs are Google's proprietary software, and I do not redistribute them. To check a claim, download the same APK from the GrapheneOS repository (hash in the artifact index), decompile it yourself with jadx 1.5.6 and open the path given.

Cross-checks

The report's core claims were checked directly in the decompiled code.

- The `allowedcars` schema (`sqm.java:24`), the Wi-Fi password insert (`jtb.java:34`), the network-location URL (`qbf.java:12`), the ads beacon (`qtf.java:16`), the 82 permissions with no `AD_ID` and no `READ_SMS` , and the absence of `play.googleapis.com` from the strings. SEEN
- The random client vehicle ID is generated at `jtb.java:147` and `jtb.java:454` .
- The Wi-Fi password is not only stored: `plo.java:511` logs it, but only when a debug system property (`debug.gh.print_wifi_info`) and two flags are set. Normal builds do not reach that line. SEEN · INFERRED (normal builds)
- The password reaches `setWpa2Passphrase` / `setWpa3Passphrase` (`plo.java:579–593`) as stored. No Keystore or encrypted-preferences code was found anywhere in the app. SEEN The app sandbox and device encryption protect that file on GrapheneOS, so this is context, not a defect.

Also checked: the manifest findings in section 03 (`QUERY_ALL_PACKAGES` , backup agent, cleartext, shared UID, special access, exported providers) and the Play Integrity answer in section 05. The notification listener class itself is a 7-line subclass of `qrk` , whose `onNotificationPosted` hands the posting to `mir.a` (`qrk.java:90–93`); what happens after that was not traced. SEEN

Code locations

EVIDENCE	WHAT IT ESTABLISHES
AndroidManifest.xml	82 permissions (no AD_ID), exported notification listener, GMS car binder bridge
defpackage/sqm.java:24,125	allowedcars schema, wifipassword column

EVIDENCE	WHAT IT ESTABLISHES
defpackage/jtb.java:34	WiFi password written to the table
defpackage/jtb.java:147,454	random client vehicle ID (SecureRandom)
defpackage/plo.java:509-511,579-593	Wi-Fi password logged only under a debug property; passed to WPA2/WPA3 passphrase setters as stored
defpackage/qrk.java:90-93	notification listener base class, <code>onNotificationPosted</code>
defpackage/qda.java:20, qgw.java:102-104, qxv.java:67	DroidGuard names only, no client call
.../config/preferences/contentprovider/ SharedPreferencesProvider.java	exported provider with in-code caller check
res/xml/network_security_config.xml	cleartext permitted for all domains
com/google/android/gms/carsetup/CarInfoInternal. java:107	car record incl. SSL subject/serial, hashed VIN redacted
com/google/android/gms/clearcut/ · PlayLoggerContext. java	vendor'd clearcut client; no upload URL anywhere in the APK
com/google/android/gearhead/telemetry/ InstallReferrerReceiver.java	install attribution
defpackage/qbf.java:12	network location endpoint (glm/mmap)
defpackage/qtf.java:16	ads SDK gen_204 beacon
com/google/android/gms/car/AnalyticsSessionInfo.java	per-session analytics ID
com/google/android/projection/gearhead/messaging/ SmsMessagingService.java	SMS projection service
strings · CAR_API_* / AUDIO_DIAGNOSTICS_*	telemetry event families

Artifact index

base.apk	35,123,449 bytes · sha256 115228a7af416c4c... · signer 1ca8dcc0...
splits	config.arm64_v8a 8.8 MB · config.xxhdpi · config.en
metadata	metadata.1.0.0.sjson · versionCode 179664044
decompiled	jadx 1.5.6 output on my machine · 176 MB · 26,427 classes (not published)
strings	148,023 entries extracted from the APK (not published)
hosts	65 hosts, 63 URLs extracted from the strings (not published)

Sources

REFERENCE	LOCATION
App repository	https://apps.grapheneos.org/ · <code>metadata.1.0.0.sjson</code>
Artifact	<code>packages/com.google.android.projection.gearhead/179664044/base.apk</code>
Tooling	<code>jadx 1.5.6</code> · <code>apktool</code> · <code>aapt2</code> · <code>apksigner 36.0.0</code>
GrapheneOS docs	https://grapheneos.org/usage · <code>frameworks_base</code> branch 17

Limits of the measurement

This report measures an APK and gives no verdict on a product. It shows what Auto is wired to do and what it stores. It cannot show what a specific car records, or what Google retains. Read the tags literally: the line between **SEEN** and **INFERRED** separates what the APK shows from what is reasoned from it.

Reproducing this

```
curl -O https://apps.grapheneos.org/metadata.1.0.0.sjson
curl -O https://apps.grapheneos.org/packages/com.google.android.projection.gearhead/179664044/base.apk

apksigner verify --print-certs base.apk          # expect 1ca8dcc0bed3cbd8...b65b5295
apktool d -f --no-src -o res base.apk           # manifest + resources
jadx -d out --no-res --show-bad-code base.apk   # 26,427 classes

# permission classification: join the APK's 82 uses-permission entries against
# protectionLevel from GrapheneOS frameworks_base branch 17 (core/res/AndroidManifest.xml)
```

Hashes in this report were verified against the repository's signify-signed metadata before any analysis. If the metadata hash and the APK hash disagree, stop: you are not looking at the artifact this report describes.

Support

If the teardown helped you and you want to support it, you can send any amount. Pick whatever suits you.



Lightning (BOLT 12)

<lno1pg98xatswphhyapqg48pvvggz7yjsxhrlfqjju2xf4qnvqd9fjk8acdy4cf7e7kep33vspyw8kplcs>

[Pay in wallet](#)



Monero

<89jGcmA65CReki73oLFvxHUzyR45wMqTN8G93ZYRizwVF2XUQgYKbPLGQYgxj2YjX4fxAUPQ7tLr4ATP7fpR4dSeHH3r9iH>

[Pay in wallet](#)



Zcash (unified address)

u1defv75w2x9p6u82qu8je0rrz5g98gnhvs0ervckctfrjvsscvcx00e4qgfk63vm0uar587exxhtp3ps7gwd54anwm32a3sg2432htr53nv22d23agngdpdkdt75lzd8pp0e3f6s2c6n52nk9ecmnjwvdxggfcl7ndsh5uq3wqptk60z

Other ways to support: juraj.bednar.io/en/support-me/