Ka-Chow!!! Driving Android Auto

Joshua Hickman Joshua Hickman

Mar 31, 2019

Updated on: Apr 03, 2019

DOI: 10.21428/482a8d7a

Synopsis

Forensic question: What information is recoverable from an Android device that is connected to a vehicle with using the Android Auto app?

OS Version: Android 8.1 (Oreo)
Files: Event Trace Logs (ETL)

Android Auto v3.8.584564 - Installed 01/16/2019 13:15 (EST)
Google v8.91.5.21 - Installed 01/16/2019 13:10 (EST)
Maps v10.7.1 - Installed 01/16/2019 13:01 (EST)

Tools:
WinHex, Version 19.7 (Specialist License)
Cellebrite UFED 4PC, Version 7.12.1.100
DCode, Version 4.02a

Introduction

Recently I purchased a new car. I am talking brand spankin' new. I had been looking for a compact SUV for a while because of a growing family, and I found it: a 2019 Nissan Rogue. I purchased it in 2018, so this would make the car, like, *really* new. I was super excited as this was the first time I had ever purchased a new car.

While signing the novel-sized stack of paperwork that is part of any car purchase, the woman I was working with and I were chatting about all the bells and whistles that were in my newly purchased ride, and she mentioned that she had a term for newer cars: laptops on wheels. She was absolutely correct. My car keeps track of all sorts of things: gas mileage, proximity to objects, tire pressure, what I'm listening to on the radio, external temperature, and other things I probably don't know about. The on-board electronics are crazy compared to my first car (a 1985 Chevy S-10 pickup). Additionally, my new car supports Apple's CarPlay and Google's Android Auto, the two automotive interfaces developed for the two major mobile software platforms

While I have been using CarPlay for some time now, I have never used Android Auto. I was aware of its existence, but that was about it. When I bought the car, I was in the middle of creating a clean Android image for the DFIR community, so I thought it would be great to have this in the image since phone-to-car interfaces will become more and more common. Currently, there are 29 different auto manufacturers that have various models which support Android Auto, and the list continues to grow. Additionally, there are after-market radio manufacturers (e.g. Pioneer, Kenwood, etc.) that are baking Android Auto in to their units, so I feel that this will become more common place as time goes on.

Method

After using Android Auto for just over a week, I created my Android image, and immediately starting poking around to see what I could find. Surprisingly, there is not much in the way of Android Auto itself; it projects certain Android apps on to your car's in-dash receiver which allows you to interact with those apps. My car has a mic (in the roof) that also worked with Android Auto and allows for voice commands/inquiries to Google Assistant without picking up the phone. Now, Android Auto does have the ability to be used on-device, but, for the purposes here, I concentrated on the in-dash flavor.

The data for Android Auto itself is accessible via an Android backup or a file system extraction. However, the good stuff, as is typical with Android, was sitting in the **/data** area, which, as we all know, is not easily accessible these days. Thus, this disclosure: the phone I was using was rooted using TWRP and Magisk.

The tests were conducted using a Nexus 5X (Model H790, Bluetooth

64:BC:0C:F5:D1:C9) running Android 8.1 with a patch level date of December 5, 2018. The image file that contains the data discussed in this article can be downloaded here.

Setup

As I previously mentioned, this research involved the use of my new car. The first thing to do was to download the app itself. Android Auto does not come as a default application with the image I downloaded (bullhead OPM 7), and I am not sure if Android OEMs install it as part of their skinning process. I checked with some co-workers who have Android devices, and Android Auto was not on their devices; those guys have Samsungs, and I do not know anyone who has anything else.

The setup process was slightly different than what I was used to with CarPlay. With CarPlay it is a matter of plugging in my iPhone and driving off. The phone and the car recognize each other, and I did not have to do anything beyond organizing the apps on the screen (like I do on my phone). Android Auto is slightly different. When I plugged the phone in to the car it immediately started Android Auto. From there, it was like any typical Android app that is started for the first time: the app permissions dance. Android Auto requested the following:

- Calendar
- Contacts
- Location
- Microphone
- Phone
- SMS

Additionally, I had to pair it to my car via Bluetooth. Again, this is not something that I had to do with CarPlay. I took the defaults on everything, agreed to the terms of service, and proceeded onward.

External Inspection

The home screen is straight forward. See Figure 1.

The home screen displays various information via cards. In Figure 1 you can see stuff that may be relevant to me. Here you can see the weather and a frequent location. If you scroll down, you can see new text messages that have not yet be read (pictured in Figure 2).

The icons at the bottom of the screen correspond to the different applications that are available through Android Auto. From left to right: Maps (Figure 3), Telephone (Figure 4), the Android home button, various applications that can

Figure 1: ¶ Figure 1

Figure 2: ¶ Figure 2

use Android Auto (Figures 5 and 6), and an exit mechanism to return to my radio's native interface (Figure 7).

Figure 3: ¶ Figure 3

Although not visible from the interface, Google Assistant is available. My particular vehicle has a button on the steering wheel that is able to summon Google Assistant without using the hot word (Ok, Google). I did not test the ability to summon Google Assistant by leveraging the hot word.

Internal Examination

For just over a week, I drove around with Android Auto. I got directions, had it read messages to me (that I sent myself – no one has the number assigned to the test device), and played podcasts. After generating the data, I performed a physical extraction, and loaded it in to WinHex to see if I could find any data left behind by Android Auto that would indicate how the application was used.

Navigating to the **/data/data** directory finds the Android Auto folder. It is listed as **com.google.android.projection.gearhead** (Figure 8).

Inside the folder was a typical layout for an Android application (Figure 9).

The **app_impl** folder contained .apk files that appeared to be relevant to the operating of the application. The **cache** folder acts as a "scratchpad" of sorts

Figure 4: ¶ Figure 4

Figure 5: ¶ Figure 5

Figure 6: ¶ Figure 6

Figure 7: ¶ Figure 7

Figure 8: ¶ Figure 8

Figure 9: ¶ Figure 9

where the applications that can use Android Auto can deposit assets as needed. In my case, there was album art from the podcasts that I listened to along with picture files that were used to display the weather conditions that were displayed. These files had a .0 file extension.

The **code_cache** folder was empty.

The **databases** folder contained four files. The first was *CloudCards.db*, which contained only one table. The table contained one entry that had a URL that referred to a .png file of what was supposed to be a sun for the weather card that was displayed on the home screen. This file had a corresponding journal file that was empty.

The second file in this folder was the *google_analytics_v4 6308.db* file. This file has one table that contains any data: properties. The table contains the Google Analytics Client ID (explained below), the Tracking ID (*tid* - described as an ID that distinguishes to which Google Analytics property to send data), and information about the app version. As with the *CloudCards.db* file, this file also has a corresponding journal file that is empty

The **files** folder contained two files. The first is *gaClientId*, which is the Google Analytics Client ID. The Client ID is a randomly generated UUID (version 4) that is used to anonymously identify a particular user or device.

There was a file named *phenotype.lock*, but the file had no contents.

The **shared_prefs** folder is where the interesting information resides. See Figure 10 for a file listing.

Figure 10: ¶ Figure 10

The first file, *app_state_shared_preferences.xml*, contains the time Android Auto was last run stored in Unix Epoch Time with the tag "pref_lastrun." See Figure 11.

Figure 11: ¶ Figure 11

The next file of interest is the *common_user_settings.xml*. This file contained the Bluetooth MAC address for my car. The file here had one entry. I did not have access to a second car that could run Android Auto, so I do not know if this file could contain multiple listings. See Figure 12.

Figure 12: ¶ Figure 12

The last file in this folder is *location_manager.xml*. See Figure 13.

This file contains some interesting data. For starters, it has the name of my car, "Nissan." The latitude and longitude geolocates to the location where I was when I last disconnected the test device from the car, and the time was the time I disconnected (in Unix Epoch Time). As far as the value in the accuracy tag, I can only hypothesize; however, I can tell you the latitude and longitude are extremely close to where I was located when I disconnected the test device (I would estimate within three meters of where my car was located when I disconnected).

Other Data

My usage of Android Auto was limited to getting directions, reading messages, and playing podcasts, so I expected to find some references or artifacts to these activities. My examination revealed none of that. My next step was to examine the applications that actually did the work: Google and Maps.

The main method by which I interacted with Android Auto was via voice commands, which is handled by Google Assistant. For the purposes of this paper I

Figure 13: ¶ Figure 13

will not deep-dive in to Google Assistant; I have posted additional research in that area that utilized some of the data here. However, I do want to highlight how Android Auto and Google interact in order to facilitate hands-free use while driving.

The Google app resides in the **/data/data** directory. The folder is **com.google.android.googlequicksearchbox**. See Figure 14.

Figure 14: ¶ Figure 14

The folder has the usual suspect folders along with several others. See Figure 15 for the folder listings.

The folder of interest here is **app_session**. This folder has a great deal of data that is relevant to my usage of Android Auto. It could also house data that is relevant to any number of ways which a user interacts with their device by the use of their voice. The folder contains several .binarypb files. See Figure 16.

Each *.binarypb* file here represents a "session," which I will define as each time Google Assistant was invoked. I will focus on the files that were generated by use of Android Auto; some of the files were generated as a result of a different manner of invocation for other purposes. Based on my notes, I know when I summoned Google Assistant and what I did when I summoned it. By comparing my notes to the MAC times associated with each *.binarypb* file I identified the applicable files. The first time I summoned Google Assistant is represented in the file with the last five digits of *43320.binarypb*. Figure 17 shows the header of the file.

Figure 15: ¶ Figure 15

Figure 16: ¶ Figure 16

Figure 17: ¶ Figure 17

The ASCII "car_assistant" seems to imply this request had been passed to Google Assistant from Android Auto. In each test that I ran, this phrase appeared at the beginning of the file. Additionally, the string in the smaller orange box (0x5951EF) accompanied the phrase at the same byte offset each time. I hesitate to call this a true "file header," though. I think someone with more time in DFIR should make that call.

The data in the red boxes, though, caught my attention.

Here is a fun fact about me: one of the jobs I held in college was that of a disc jockey. During this time, digital audio workstations were really coming in to their own, so I sat in front of Pro Tools quite a bit; I saw *a lot* of audio files in hex/ASCII. I continued to use Pro Tools extensively through the mp3 craze of the late 90's and early 2000's and had worked with mp3 files quite a bit. While it had been some time, my eyeballs immediately saw the "yoda" string. It also helps that I am a Star Wars fan.

The "yoda" string (0xFFF344C4) is a frame sync header for mp3 files. I will not go in to detail, but just know that this string tipped me off that something else, not expected, may be in this file. I decided to scroll to the end of the gobbly-goo to see what I could fine. Needless to say, I was surprised. See Figure 18.

The orange box is important. In it is "LAME3.99.5." LAME is a free, open-source mp3 encoder (codec). This particular version, 3.99.5, was released in February of 2012. The additional yoda strings along with the 0x55 values in between is what happens when mp3 frames need to be padded (i.e. filled in order to round out a frame to fit the bit rate at which the audio is captured). Each mp3 encoder handles padding differently, but the repetitive data in between yodas was indicative. All of this further heightened my suspicion that what I was looking at was mp3 data.

So, I carved from the first yoda (seen in Figure 17), to the last (seen in Figure 18), for a total of 11.1 KB. I then saved the file with no extension and opened it in VLC Player. The following came out of the speakers:

> *"You've got a few choices. Pick the one you want."*

Based on my notes, this was the last phrase Google Assistant spoke to me prior to handing me off to Maps. In this session, I had asked for directions to Starbucks and had not been specific about which one, which caused the returned reply that I had just heard.

Scrolling further in to the file I found the area shown in Figure 19. The purple box with the arrow next to it was my actual voice inquiry via Android Auto, and the orange box contains the "car_assistant" string. The strings in the blue and green boxes were interesting as I had seen them elsewhere while examining other data in this folder. Also, there is a second phrase there (bottom purple box) that is a variation of what is in the first purple box. Interesting…

The following morning, I headed to Starbucks again. This time I was a little

Figure 18: ¶ Figure 18

Figure 19: ¶ Figure 19

more specific about which one I wanted. I invoked Google Assistant, asked for directions, and navigated to the location. This session is represented *12067.binarypb* file. See Figure 20.

Figure 20: ¶ Figure 20

The same "car_assistant" phrase was there, along with the 0x5951EF string. There is also another yoda. Time to scroll…

There is the LAME codec reference, the padding, and the final yoda in Figure 21. As before, I carved, saved, and played in VLC Player. I heard the following:

"Starbucks is ten minutes from your location by car and light traffic."

Based on my notes, this was, again, the last thing I heard prior to Maps initializing and taking over. I continued to scroll through the file to see if I could find the voice request I had originally given Google Assistant. See Figure 22.

The strings in the blue and green boxes are here, but the string in the red box is different. As before, the purple box with the arrow next to it is what I actually said. As with the previous request, there are multiple versions of what I said below. It could be that the app is generating variations of what I said, assigning some score/weight to them, and then picking which one it *thinks* is what I said. At this point, this is merely speculation on my part. There is definitely more research that can be done in this area.

A couple of days later I had to pick up my child who had been on vacation with the grandparents the previous week. As is normal, my parents and I meet

Figure 21: ¶ Figure 21

Figure 22: ¶ Figure 22

halfway so it isn't too burdensome on either party. This time, I had a couple of text messages that needed to be read prior to starting the navigation. This session is represented by the file *22686.binarypb*. See Figure 23.

Figure 23: ¶ Figure 23

The usual suspects are here, so I repeated my steps with regards to carving out the mp3 file, and got the following audio:

> *"Smithfield Chicken and BBQ is 51 minutes from your location by car and light traffic."*

Now, this is interesting. This audio file is, in fact, the last thing Google Assistant said to me prior to handing me off to Maps. However, this audio file, as with the others, sits at the front of the *.binarypb* file. This particular session started out by reading my text messages. I decided to scroll through to see if I could find that part of the session.

The text in Figure 24 is what started the session. The purple box with the arrow represents what I actually said when I initially invoked Google Assistant. The second purple box, as before, is a variation of what I said. Also, the string in the green box is present, but the string in the blue box is absent. Figure 25 is the message that was read to me. Notice that it added the contact information for the message; I sent it from my personal phone, which has an entry in the contacts app on the test device.

After the message was completed, Google Assistant asked if I wanted to reply. You can see what I said in Figure 26.

The string in the green box is present…again…and a variation of what I said is seen in the bottom purple box. I then proceeded to dictate my message, which is seen in Figure 27.

You can see the string in the green box is present, and what I actually dictated is in the purple box with the arrow next to it. The variations are in the purple boxes below the first. The message was read back to me, and then I was asked if I wanted to send it. See Figure 28.

There's that pesky green box again. This time, apparently, Google Assistant

Figure 24: ¶ Figure 24

Figure 25: ¶ Figure 25

Figure 26: ¶ Figure 26

Figure 27: ¶ Figure 27

Figure 28: ¶ Figure 28

felt that it understood what it was I said. There are no variations of what I said here.

The next thing I asked Google Assistant for was directions. See Figure 29.

Because of the length of my request, I could not fit everything into the screenshot, but I grabbed what I could. The blue box has returned in this request, and the ever-present green box is here. As usual, my actual request is marked by the purple arrow, with variations of what I said in the purple boxes below. The result of this inquiry was the mp3 file I had carved earlier, and I was then handed off to Maps.

Just an observation here…the hex string in the green box showed up each time Google Assistant listened for input from me, and the hex string in the blue box showed up each time I asked for directions to a location. Is the green box a header? Maybe. I'll defer to someone who has more experience in this area.

The next place I looked was Maps. Maps was a little thin in the way of unique data related to Android Auto. The biggest thing I found were audio files that contained the set of navigation directions that were used when I asked for directions to Smithfield Chicken & BBQ. Those files can be found in **/data/data/com.google.android.apps.maps/apps_tts-temp** folder. These file names are numbers starting with 0 and are in chronological order. There are no file extensions, but the file headers are "RIFF."

Conclusions

In the way of artifacts, Android Auto leaves very little behind with regards to user data, which is not surprising. The app merely functions as an interface to other apps and services in Android, and those apps keep the data to themselves; this makes sense as they are the ones doing all the work. It does, however, reveal some basic data about its use and (at least) one vehicle to which it has been connected.

As vehicles get more and more complex, vehicle manufacturers will continue to add Android Auto as a stock option to their various models. Additionally, Android Auto can be used on the device itself, without the need of an infotainment center. While this ability was not tested specifically, it stands to reason that the artifacts left behind are similar.

The information found in the Google Assistant area of the phone is interesting and could be a great research project. Hopefully, someone will take it up.

Below you will find a chart that contains a quick reference about the artifacts/data discussed in this article.

Figure 29: ¶ Figure 29

File Name	Location	Artifact	Relevance
google_analytics_v4.db	com.google.android.projection.gearhead/	Google Analytics Client ID	Google Analytics Client ID (gaClientID) - Randomly generated UUID that is used to anonymously identify a particular device or user.
app_state_shared_preferences.xml	com.google.android.projection.gearhead/shared_prefs	Last time Android Auto was run	
cache folder	com.google.android.projection.gearhead/	N/A	Applications that can use Android Auto will drop assets/resources in this folder
common_user_settings.xml	com.google.android.projection.gearhead/shared_prefs	Bluetooth MAC Address	The MAC address for the paired vehicle
common_user_settings.xml	com.google.android.projection.gearhead/shared_prefs	Car Name & Last Location	The name of the car paired, and the last location Android Auto was located
app_session folder	com.android.googlequicksearchbox/files/hotword_interaction	N/A	The binarypb files in this folder contain interaction with Google Assistant via Android Auto. The file header will indicate when Google Assistant received a request via Android Auto (”car_assistant”)

DFIR Review

The author presents data he obtained from an Android device coupled to his car's entertainment console using Android Auto. The research is done rigorously, the author describes his very detailed note taking during the one week period for which he analyzed the data. The description goes into details regarding the possibilities of the interaction, giving valuable insight for examiners that might not have access to a live system.

The data analysed was obtained from the phone utilizing physical extraction. This was possible because the used phone was rooted, something that may often not be possible when analyzing a phone in a criminal investigation context. Especially for car crashes, where the information may frequently be of interest, it is unlikely that rooting of the phone is considered proportional, even if rooting is available.

The found data is of interest for all cases where interaction with an Android phone using the car as an intermediate takes place. Contrary to other voice controlled devices like Alexa, voice recordings of the driver do not seem to be stored. Only the interpreted data is stored, in multiple possible versions as it seems. It is therefore not possible to gain any information on the individual that issued the command.

The post provides an interesting insight into the traces that can be found in a similar case and is a solid foundation for the interpretation of traces found in a case. The limitation of the experiments are well indicated and provide a handhold limiting the risk of overstretching statements based on the research at hand. Care should be applied when applying the findings on ulterior versions of the application.

Future Work

Further research could delve into the information that can be found on the car's side of the interactions for cases where the device is not available for analysis.

Reviewers

- Addisu Afework Birhanu (Methodology Review)
- Timothy Bollé (Methodology Review)
- Ali Hadi (Methodology Review)
- Francesco Servida (Methodology Review)
- Hannes Spichiger (Methodology Review)