DFIR Review

Parsing Google's Now Playing History on Pixel Devices

Kevin Pagano

Published on: Aug 31, 2020

DOI: 10.21428/b0ac9c28.8b3978af

License: [Creative Commons Attribution 4.0 International License \(CC-BY 4.0\)](https://creativecommons.org/licenses/by/4.0/)

Synopsis

Forensic question: What information is recoverable from the use of the Now Playing feature on Google Pixel phones?
OS: Android 10
Tools: Magnet Acquire 2.25.0.20236 DB Browser for SQLite 3.11.2 protoc 3.11.4 ALEAPP	

Google implemented a baked in app/feature called Now Playing as part of the Pixel 2 and Pixel 2XL launch in 2017 and has been included in every Pixel phone release since. It gives you the option to allow Google to try and recognize song information of music that is playing around you. Per Google:

When music plays nearby, your phone compares a few seconds of music to its on-device library to try to recognize the song. This processing happens on your phone and is private to you. - [Google Answers](#)

So since this history information is stored locally, we get a nice little database full of information. To view what you can from the UI, you can go to:

Settings > Sound > Now Playing

You will see a screen similar to this:

There isn't many switches to play with here other than turning on song info on your lock screen or receiving notifications. From here we can click Now Playing History to view a list of song history from as far back as the beginning of when the app was initiated.

If you click on a single song, you get a list of apps that the song can be streamed from:

Now that we know what can be accessed from the UI, let's take a look what we have from a forensic perspective via an ADB backup. The file path where this app information can be located is:

apps/com.google.intelligence.sense/db/history_db

The file is a sqlite database with only a few columns. Opening this in DB Browser we see two tables, only one of real significance, *recognition_history*:

Tables (2)

android_metadata	
locale	TEXT
recognition_history	
timestamp	LONG
history_entry	BLOB
track_id	STRING
shards_region	STRING
downloaded_shards_version	INTEGER
core_shard_version	INTEGER

	timestamp	history_entry	track_id
	Filter	Filter	Filter
1	1539056375198	BLOB	Tb57vjsem2h45yzip6c5akfv5m
2	1539056421656	BLOB	Tb57vjsem2h45yzip6c5akfv5m
3	1539056493219	BLOB	Tb57vjsem2h45yzip6c5akfv5m
4	1539056586764	BLOB	Tb57vjsem2h45yzip6c5akfv5m
5	1539057284328	BLOB	Tsmixeygppa5mg3hiva3kydn12m
6	1539057344888	BLOB	Tsmixeygppa5mg3hiva3kydn12m
7	1539058623835	BLOB	Tu6itdswgreieivjmxayk6jbky
8	1539058684172	BLOB	Tu6itdswgreieivjmxayk6jbky
9	1539058754503	BLOB	Tu6itdswgreieivjmxayk6jbky
10	1539058844008	BLOB	T4ivtimyyywb1tcgteont57pgpi

Timestamp was easily parsed, it was in Unix Epoch format ([Alexis Brignoni made a post on this](#)).

History_entry is a Protobuf blob, which we can see some useful information just by looking at the hex:

```

0000 3a 10 41 6d 65 72 69 63 61 2f 4e 65 77 5f 59 6f  :.America/New Yo
0010 72 6b 4a 94 06 0a 1b 54 32 32 7a 34 63 35 77 6a  rkJ ...T22z4c5wj
0020 69 71 70 76 37 69 74 6e 77 6c 75 66 6d 6a 74 35  iqpv7itnwlufmjt5
0030 7a 69 12 1b 42 6a 65 6b 6f 7a 6f 6a 6f 79 79 6d  z1..Bjekozojoyym
0040 61 37 32 33 65 67 32 68 67 75 6b 7a 65 61 61 1a  a723eq2hgukzreaa.
0050 13 46 75 6e 6e 79 20 74 68 65 20 57 61 79 20 49  .Funny the Way I
0060 74 20 49 73 22 12 44 61 76 65 20 4d 61 74 74 68  t Is".Dave Matth
0070 65 77 73 20 42 61 6e 64 31 00 00 00 00 00 ac 70  ews Bandl.....p
0080 40 38 b1 02 38 03 38 b2 02 38 b3 02 42 0d 2f 67  @8±.8°.8°.B./g
0090 2f 31 31 62 76 30 67 30 38 71 39 4a 98 01 12 88  /1lbv0g08q9J ..
00a0 01 61 6e 64 72 6f 69 64 2d 61 70 70 3a 2f 2f 63  .android-app://c
00b0 6f 6d 2e 61 70 70 6c 65 2e 61 6e 64 72 6f 69 64  om.apple.android
00c0 2e 6d 75 73 69 63 2f 68 74 74 70 73 2f 69 74 75  .music/https/itu
00d0 6e 65 73 2e 61 70 70 6c 65 2e 63 6f 6d 2f 75 73  nes.apple.com/us
00e0 2f 61 6c 62 75 6d 2f 62 69 67 2d 77 68 69 73 6b  /album/big-whisk
00f0 65 79 2d 61 6e 64 2d 74 68 65 2d 67 72 6f 6f 67  ey-and-the-groog
0100 72 75 78 2d 6b 69 6e 67 2f 33 31 32 36 30 39 31  rux-king/3126091
0110 35 35 3f 69 3d 33 31 32 36 30 39 33 36 33 26 69  55?i=312609363&i
0120 67 6e 2d 67 61 63 74 3d 32 1a 0b 2f 6d 2f 30 31  an-gact=2.../m/01

```

But for further analysis we can export the blob out of DB Browser and parse using [protoc.exe](#) (thanks for the hint [Joshua Hickman](#)) and the following command:

```
protoc.exe --decode_raw < samplesong1
```

Here is the output we get:

```

1: "America/New_York"
2: "12224c5c10p77tm1urft5s1" ----- Track ID
3: "520202077m721agp4c2eaa" ----- Album ID
4: "Funny the way it is" ----- Song Title
5: "Dave Matthews Band" ----- Artist
6: 644878ac00000000 ----- Song Duration
7: 365
8: 1
9: 1
10: 1
11: 1
12: 1
13: 1
14: 1
15: 1
16: 1
17: 1
18: 1
19: 1
20: 1
21: 1
22: 1
23: 1
24: 1
25: 1
26: 1
27: 1
28: 1
29: 1
30: 1
31: 1
32: 1
33: 1
34: 1
35: 1
36: 1
37: 1
38: 1
39: 1
40: 1
41: 1
42: 1
43: 1
44: 1
45: 1
46: 1
47: 1
48: 1
49: 1
50: 1
51: 1
52: 1
53: 1
54: 1
55: 1
56: 1
57: 1
58: 1
59: 1
60: 1
61: 1
62: 1
63: 1
64: 1
65: 1
66: 1
67: 1
68: 1
69: 1
70: 1
71: 1
72: 1
73: 1
74: 1
75: 1
76: 1
77: 1
78: 1
79: 1
80: 1
81: 1
82: 1
83: 1
84: 1
85: 1
86: 1
87: 1
88: 1
89: 1
90: 1
91: 1
92: 1
93: 1
94: 1
95: 1
96: 1
97: 1
98: 1
99: 1
100: 1
101: 1
102: 1
103: 1
104: 1
105: 1
106: 1
107: 1
108: 1
109: 1
110: 1
111: 1
112: 1
113: 1
114: 1
115: 1
116: 1
117: 1
118: 1
119: 1
120: 1
121: 1
122: 1
123: 1
124: 1
125: 1
126: 1
127: 1
128: 1
129: 1
130: 1
131: 1
132: 1
133: 1
134: 1
135: 1
136: 1
137: 1
138: 1
139: 1
140: 1
141: 1
142: 1
143: 1
144: 1
145: 1
146: 1
147: 1
148: 1
149: 1
150: 1
151: 1
152: 1
153: 1
154: 1
155: 1
156: 1
157: 1
158: 1
159: 1
160: 1
161: 1
162: 1
163: 1
164: 1
165: 1
166: 1
167: 1
168: 1
169: 1
170: 1
171: 1
172: 1
173: 1
174: 1
175: 1
176: 1
177: 1
178: 1
179: 1
180: 1
181: 1
182: 1
183: 1
184: 1
185: 1
186: 1
187: 1
188: 1
189: 1
190: 1
191: 1
192: 1
193: 1
194: 1
195: 1
196: 1
197: 1
198: 1
199: 1
200: 1
201: 1
202: 1
203: 1
204: 1
205: 1
206: 1
207: 1
208: 1
209: 1
210: 1
211: 1
212: 1
213: 1
214: 1
215: 1
216: 1
217: 1
218: 1
219: 1
220: 1
221: 1
222: 1
223: 1
224: 1
225: 1
226: 1
227: 1
228: 1
229: 1
230: 1
231: 1
232: 1
233: 1
234: 1
235: 1
236: 1
237: 1
238: 1
239: 1
240: 1
241: 1
242: 1
243: 1
244: 1
245: 1
246: 1
247: 1
248: 1
249: 1
250: 1
251: 1
252: 1
253: 1
254: 1
255: 1
256: 1
257: 1
258: 1
259: 1
260: 1
261: 1
262: 1
263: 1
264: 1
265: 1
266: 1
267: 1
268: 1
269: 1
270: 1
271: 1
272: 1
273: 1
274: 1
275: 1
276: 1
277: 1
278: 1
279: 1
280: 1
281: 1
282: 1
283: 1
284: 1
285: 1
286: 1
287: 1
288: 1
289: 1
290: 1
291: 1
292: 1
293: 1
294: 1
295: 1
296: 1
297: 1
298: 1
299: 1
300: 1
301: 1
302: 1
303: 1
304: 1
305: 1
306: 1
307: 1
308: 1
309: 1
310: 1
311: 1
312: 1
313: 1
314: 1
315: 1
316: 1
317: 1
318: 1
319: 1
320: 1
321: 1
322: 1
323: 1
324: 1
325: 1
326: 1
327: 1
328: 1
329: 1
330: 1
331: 1
332: 1
333: 1
334: 1
335: 1
336: 1
337: 1
338: 1
339: 1
340: 1
341: 1
342: 1
343: 1
344: 1
345: 1
346: 1
347: 1
348: 1
349: 1
350: 1
351: 1
352: 1
353: 1
354: 1
355: 1
356: 1
357: 1
358: 1
359: 1
360: 1
361: 1
362: 1
363: 1
364: 1
365: 1
366: 1
367: 1
368: 1
369: 1
370: 1
371: 1
372: 1
373: 1
374: 1
375: 1
376: 1
377: 1
378: 1
379: 1
380: 1
381: 1
382: 1
383: 1
384: 1
385: 1
386: 1
387: 1
388: 1
389: 1
390: 1
391: 1
392: 1
393: 1
394: 1
395: 1
396: 1
397: 1
398: 1
399: 1
400: 1
401: 1
402: 1
403: 1
404: 1
405: 1
406: 1
407: 1
408: 1
409: 1
410: 1
411: 1
412: 1
413: 1
414: 1
415: 1
416: 1
417: 1
418: 1
419: 1
420: 1
421: 1
422: 1
423: 1
424: 1
425: 1
426: 1
427: 1
428: 1
429: 1
430: 1
431: 1
432: 1
433: 1
434: 1
435: 1
436: 1
437: 1
438: 1
439: 1
440: 1
441: 1
442: 1
443: 1
444: 1
445: 1
446: 1
447: 1
448: 1
449: 1
450: 1
451: 1
452: 1
453: 1
454: 1
455: 1
456: 1
457: 1
458: 1
459: 1
460: 1
461: 1
462: 1
463: 1
464: 1
465: 1
466: 1
467: 1
468: 1
469: 1
470: 1
471: 1
472: 1
473: 1
474: 1
475: 1
476: 1
477: 1
478: 1
479: 1
480: 1
481: 1
482: 1
483: 1
484: 1
485: 1
486: 1
487: 1
488: 1
489: 1
490: 1
491: 1
492: 1
493: 1
494: 1
495: 1
496: 1
497: 1
498: 1
499: 1
500: 1
501: 1
502: 1
503: 1
504: 1
505: 1
506: 1
507: 1
508: 1
509: 1
510: 1
511: 1
512: 1
513: 1
514: 1
515: 1
516: 1
517: 1
518: 1
519: 1
520: 1
521: 1
522: 1
523: 1
524: 1
525: 1
526: 1
527: 1
528: 1
529: 1
530: 1
531: 1
532: 1
533: 1
534: 1
535: 1
536: 1
537: 1
538: 1
539: 1
540: 1
541: 1
542: 1
543: 1
544: 1
545: 1
546: 1
547: 1
548: 1
549: 1
550: 1
551: 1
552: 1
553: 1
554: 1
555: 1
556: 1
557: 1
558: 1
559: 1
560: 1
561: 1
562: 1
563: 1
564: 1
565: 1
566: 1
567: 1
568: 1
569: 1
570: 1
571: 1
572: 1
573: 1
574: 1
575: 1
576: 1
577: 1
578: 1
579: 1
580: 1
581: 1
582: 1
583: 1
584: 1
585: 1
586: 1
587: 1
588: 1
589: 1
590: 1
591: 1
592: 1
593: 1
594: 1
595: 1
596: 1
597: 1
598: 1
599: 1
600: 1
601: 1
602: 1
603: 1
604: 1
605: 1
606: 1
607: 1
608: 1
609: 1
610: 1
611: 1
612: 1
613: 1
614: 1
615: 1
616: 1
617: 1
618: 1
619: 1
620: 1
621: 1
622: 1
623: 1
624: 1
625: 1
626: 1
627: 1
628: 1
629: 1
630: 1
631: 1
632: 1
633: 1
634: 1
635: 1
636: 1
637: 1
638: 1
639: 1
640: 1
641: 1
642: 1
643: 1
644: 1
645: 1
646: 1
647: 1
648: 1
649: 1
650: 1
651: 1
652: 1
653: 1
654: 1
655: 1
656: 1
657: 1
658: 1
659: 1
660: 1
661: 1
662: 1
663: 1
664: 1
665: 1
666: 1
667: 1
668: 1
669: 1
670: 1
671: 1
672: 1
673: 1
674: 1
675: 1
676: 1
677: 1
678: 1
679: 1
680: 1
681: 1
682: 1
683: 1
684: 1
685: 1
686: 1
687: 1
688: 1
689: 1
690: 1
691: 1
692: 1
693: 1
694: 1
695: 1
696: 1
697: 1
698: 1
699: 1
700: 1
701: 1
702: 1
703: 1
704: 1
705: 1
706: 1
707: 1
708: 1
709: 1
710: 1
711: 1
712: 1
713: 1
714: 1
715: 1
716: 1
717: 1
718: 1
719: 1
720: 1
721: 1
722: 1
723: 1
724: 1
725: 1
726: 1
727: 1
728: 1
729: 1
730: 1
731: 1
732: 1
733: 1
734: 1
735: 1
736: 1
737: 1
738: 1
739: 1
740: 1
741: 1
742: 1
743: 1
744: 1
745: 1
746: 1
747: 1
748: 1
749: 1
750: 1
751: 1
752: 1
753: 1
754: 1
755: 1
756: 1
757: 1
758: 1
759: 1
760: 1
761: 1
762: 1
763: 1
764: 1
765: 1
766: 1
767: 1
768: 1
769: 1
770: 1
771: 1
772: 1
773: 1
774: 1
775: 1
776: 1
777: 1
778: 1
779: 1
780: 1
781: 1
782: 1
783: 1
784: 1
785: 1
786: 1
787: 1
788: 1
789: 1
790: 1
791: 1
792: 1
793: 1
794: 1
795: 1
796: 1
797: 1
798: 1
799: 1
800: 1
801: 1
802: 1
803: 1
804: 1
805: 1
806: 1
807: 1
808: 1
809: 1
810: 1
811: 1
812: 1
813: 1
814: 1
815: 1
816: 1
817: 1
818: 1
819: 1
820: 1
821: 1
822: 1
823: 1
824: 1
825: 1
826: 1
827: 1
828: 1
829: 1
830: 1
831: 1
832: 1
833: 1
834: 1
835: 1
836: 1
837: 1
838: 1
839: 1
840: 1
841: 1
842: 1
843: 1
844: 1
845: 1
846: 1
847: 1
848: 1
849: 1
850: 1
851: 1
852: 1
853: 1
854: 1
855: 1
856: 1
857: 1
858: 1
859: 1
860: 1
861: 1
862: 1
863: 1
864: 1
865: 1
866: 1
867: 1
868: 1
869: 1
870: 1
871: 1
872: 1
873: 1
874: 1
875: 1
876: 1
877: 1
878: 1
879: 1
880: 1
881: 1
882: 1
883: 1
884: 1
885: 1
886: 1
887: 1
888: 1
889: 1
890: 1
891: 1
892: 1
893: 1
894: 1
895: 1
896: 1
897: 1
898: 1
899: 1
900: 1
901: 1
902: 1
903: 1
904: 1
905: 1
906: 1
907: 1
908: 1
909: 1
910: 1
911: 1
912: 1
913: 1
914: 1
915: 1
916: 1
917: 1
918: 1
919: 1
920: 1
921: 1
922: 1
923: 1
924: 1
925: 1
926: 1
927: 1
928: 1
929: 1
930: 1
931: 1
932: 1
933: 1
934: 1
935: 1
936: 1
937: 1
938: 1
939: 1
940: 1
941: 1
942: 1
943: 1
944: 1
945: 1
946: 1
947: 1
948: 1
949: 1
950: 1
951: 1
952: 1
953: 1
954: 1
955: 1
956: 1
957: 1
958: 1
959: 1
960: 1
961: 1
962: 1
963: 1
964: 1
965: 1
966: 1
967: 1
968: 1
969: 1
970: 1
971: 1
972: 1
973: 1
974: 1
975: 1
976: 1
977: 1
978: 1
979: 1
980: 1
981: 1
982: 1
983: 1
984: 1
985: 1
986: 1
987: 1
988: 1
989: 1
990: 1
991: 1
992: 1
993: 1
994: 1
995: 1
996: 1
997: 1
998: 1
999: 1
1000: 1

```

As you can see from the markups, I was able to parse out some information of each track including song title, artist, and album information. We also see the URL paths to the streaming app locations from the UI. With a little help from [Sarah Edwards](#), I was able to parse out the song duration in seconds, it is an 8-byte float stored in big-endian.

Track ID and album ID can be used to verify the song/album to some degree, they relate to Google Play Music store.

<https://play.google.com/music/m/Bjekozojoyyma723eg2hgukzeaa>

The most useful information tidbit from this may possibly be at the very top, the timezone of the device when the song was identified by the app. This could possibly put the owner of the device at a rough area during a specific timeframe.

All of the other information at this point I have yet to determine the usefulness.

You will see duplicate entries next to each other for songs. That is because Now Playing will listen for a small period of time every minute to conserve battery life, continually identifying what it can. If a song is 2+ minutes, you should see 2 entries for one song in a row, but each will have different timestamps associated.

Quick as always, [Alexis Brignoni](#) turned around a parser for [ALEAPP](#) in less than a day, so you can grab that now!

[illegible]

DFIR Review

This paper provides insight into an artifact that has not been frequently discussed in the digital forensic community. Use cases would have been useful to show the potential significance of the artifact. The author provided a solid hypothesis which was tested. The tools and sources used are well documented. One point that could have been discussed in more detail is the `track_id` that is available in the column `track_id` of the database and in the blob, allowing more information about the song to be retrieved using the URL [https://play.google.com/music/preview/\[track_id\]](https://play.google.com/music/preview/[track_id]), even if it may not be important information for criminal cases. Providing more information with screenshots regarding where the time zone information can be found would have added more value to the findings. It would be helpful to include a references section at the end of the document.

Future Work

Additional research could be conducted to determine if the feature is the same on all OS versions for the Pixel or if it is available on other devices. It may be helpful to determine how to integrate this data with other datasets that may provide the specific location of the user at the time of detecting the played song. As the author indicated, further analysis would be beneficial to determine the benefit of this data for forensic applications. It may also be helpful to compare what is available when the app data is extracted using forensic tools or from a rooted device (if possible). Future work could also include finding similar SQLite database files on different platforms (such as iOS).

Reviewers

Addisu Afework Birhanu (Methodological Review and Verified Review using Author Provided Datasets)

Timothy Bollé (Methodological Review and Verified Review using Author Provided Datasets)

Yohannes Yemane Brhan (Methodological Review and Verified Review using Author Provided Datasets)

Anthony Knutson (Methodological Review)