Forensic Science International: Digital Investigation 53 (2025) 301909

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

I know where you have been last summer: Extracting privacy-sensitive information via forensic analysis of the Mercedes-Benz NTG5*2 infotainment system

Dario Stabili [*], Filip Valgimigli [*], Mirco Marchetti

Department of Engineering "Enzo Ferrari" - University of Modena and Reggio Emilia, Italy

ARTICLE INFO

Keywords:
Privacy analysis of infotainment systems
Privacy exposure
SQL database carving

ABSTRACT

Modern vehicles are equipped with In-Vehicle Infotainment (IVI) systems that offers different functions, such as typical radio and multimedia services, navigation and internet browsing. To operate properly, IVI systems have to store locally different types of data, reflecting user preferences and behaviors. If stored and managed insecurely, these data might expose sensitive information and represent a privacy risk. In this paper we address this issue by presenting a methodology for the extraction of privacy-sensitive information from the popular $NTG5$ COMMAND IVI system (specifically, the $NTG5*2$ version by Harman), deployed in some Mercedes-Benz vehicles from 2013 to 2019. We show that it is possible to extract information related to geographic locations and various vehicles events (such as ignition and doors opening and closing) dating back to the previous 8 months, and that these data can be cross-referenced to precisely identify the activities and habits of the driver. Moreover, we develop a novel forensic tool to automate this task.[1] Given the past usage of the $NTG5$ system, our work might have real life implications for the privacy of millions of drivers, owners and passengers. As a final contribution, we develop a novel technique for SQLite data carving specifically designed to identify deleted data. Comparison with existing state-of-the-art tools for SQLite3 data recovery demonstrates that our approach is more effective in recovering deleted traces than general purpose tools.

1. Introduction

With the introduction of electronic components in passenger vehicles, car manufacturers began to push advanced features to increase both safety and comfort. From simple actuators with limited impact on the vehicle dynamics (e.g. electric windows, seats, and air conditioning) to more complex Advanced Driver Assistance Systems (ADAS) controlling steering, braking and acceleration, modern vehicles are prominent examples of Cyber-Physical systems. Another crucial aspect being developed in modern vehicles is the In-Vehicle Infotainment system (IVI), composed of hardware and software components providing access to multimedia entertainment and applications for driver and passengers. Modern IVIs typically include satellite navigation, audio and video players, many external interfaces such as USB, Bluetooth, Wi-Fi and Cellular, and advanced input methods ranging from steering wheel controls, hands-free voice control, touch screens and voice assistants. Many car manufacturer developed proprietary solutions, such as Ford (SYNC), STELLANTIS (UConnect), and Mercedes (NTG). Modern IVI systems also provide gateway functionality between different in-vehicle communication protocols (such as the Controller Area Network (Bosch, 1991) or the Local Interconnect Network (LIN Consortium, 2010)) and act as a host for different antennas used for location services (GPS) and external connectivity (GSM).

IVIs are commonly connected to multiple in-vehicle sub-networks, and they can collect data related to several vehicle subsystems which can be logged on local storage and/or accessed remotely via companion apps. In one of the first proposal for proper management of data generated by the vehicle system, the European Automobile Manufacturers' Association (ACEA (2024a), an association representing 15 major Europe-based automobile manufacturers including Mercedes-Benz, the BMW group and Ferrari), proposed a regulation on data access in a white paper (ACEA, 2024b). This regulation is based on manufacturers

* Corresponding authors.
E-mail addresses: dario.stabili@unimore.it (D. Stabili), filip.valgimigli@unimore.it (F. Valgimigli), mirco.marchetti@unimore.it (M. Marchetti).

[1] Upon Mercedes-Benz request, the tool available on demand to a restricted audience upon verification of the required usage to avoid misuses by malicious actors after filling this form: https://forms.gle/p2X8BSRTEPcWBTvL9.

https://doi.org/10.1016/j.fsidi.2025.301909
Received 13 August 2024; Received in revised form 27 December 2024; Accepted 23 February 2025

Available online 14 March 2025

providing fair, reasonable, and non-discriminatory access to in-vehicle data and resources to evening the playing field between vehicle manufacturers and third party service providers. In this document, data is categorized in different levels according the expected availability, being (A) data available on the vehicle manufacturer's backend via an external interface; (B) data that can be made accessible via an external interface but that has not yet been transferred to the interface; and (C) data generated by applications hosted in a vehicle's electronic control unit. Despite the document is primarily addressing data generated from the in-vehicle architecture or a third party application and available to both manufacturer and other third party entities, it is evident that modern vehicles will produce more and more data that, if not managed properly, might hinder the privacy of the driver and other vehicle's occupants.

In an effort to raise awareness about this issue afflicting the automotive industry, in this paper we demonstrate how sensitive information about the driver can be extracted from an undocumented data format used to store GPS entries in the $NTG5*2$ infotainment system developed by Harman, installed in just over a million of vehicles manufactured by Mercedes-Benz only.

The work has three main contributions. First, we provide a step-by-step description of the process used to reverse engineer the proprietary and undocumented data structure used by the $NTG5$ COMAND IVI system of a Mercedes-Benz C-Class to log GPS coordinates, as well as physical events related to the vehicle's body and engine. Second, we expose privacy implications of having a local and unprotected log on the IVI by analyzing information related to driving sessions performed by the authors of this work in a controlled environment. We remark that we were able to extract the same information for clear log files containing data related to previous drivers over a time span of more than 8 months before our test. The third and final contribution is the design of a novel file carving tool that exploits the structure used to save data on a SQLite3 database to recover additional entries from the disk and extract deleted traces from the system dated up to 2 years prior our first data acquisition. We compared our novel approach with existing generic tools for SQLite3 data recovery, demonstrating an improvement compared to the state-of-the-art of more than 30% recovered database entries.

The rest of this paper is organized as follows. Section 2 presents the state-of-the-art of forensic analysis of IVI systems and previous investigations on privacy perception of users. Section 3 describes the data acquisition and reverse engineering processes for extracting GPS and other personal data. The definition of the methodologies used to reconstruct forensic time lines from the data, and the impact on the privacy of the driver is demonstrated in Section 4. Section 5 presents a novel method for the recovery of deleted traces from the target system by exploiting the knowledge of the encoding format, demonstrating its improved performance against the current state-of-the-art. Section 6 discusses possible solutions to the underlying weaknesses that have been exploited in this work. Finally, Section 7 summarizes final remarks and outlines for future research.

2. Related work

The application of digital forensic techniques and tools covers a multidisciplinary area of research, from the extraction of personal data from secondhand consoles (Read et al., 2024) to the privacy analysis of data managed in popular mobile applications (Dragonas et al., 2024; Ebbers et al., 2024). With the advent of the COVID pandemic and the increasing adoption of remote meeting applications, there has been a growing interest in the privacy of the data stored by remote meeting platform applications (Kang et al., 2024; Soni et al., 2024) and smart cameras (Salem and Hamarsheh, 2024; Stabili et al., 2024), presenting how existing forensic methodologies can be applied successfully in different contexts. Many of the aforementioned applications of digital forensic methodologies has also been applied to the SQLite3 (SQLite Consortium, February 25, 2025) database, a C-based implementation that offers a small, fast and self-contained SQL database engine. Due to its ease of configuration and no overhead required for installing the whole DBMS, SQLite is the most used database engine, being used by many different applications on mobile phones, computers and embedded devices, thus including also automotive-based systems. Due to the high deployment of SQLite in different application contexts, many forensic researchers started discussing the possibility of data recovery from SQLite databases. The authors of Jeon et al. (2012) demonstrated how to exploit the unallocated area of the SQLite database to identify deleted data, while the authors of Wu et al. (2013) demonstrated the same approach on a YAFFS2 file systems (only applicable to Android devices older than version 2.3). A more recent work is presented in Lee et al. (2017), where the authors extend on Jeon et al. (2012) by also exploring the freelist of the SQLite database and using the generic encoding of the fields in the database to identify potential data. Our contribution extends Lee et al. (2017) by exploiting the structure of the table containing the data to be recovered, thus leading to a more accurate identification of the actual data and raising a lower number of false positives.

One of the latest areas of application of forensics methodologies is represented by the automotive systems. While in the automotive context there are many established research fields, such as cybersecurity for automotive architectures and protocols (Radu and Garcia, 2016; Groza et al., 2021) and the implementation of intrusion detection algorithms for in-vehicle communications (Cho and Shin, 2016; Stabili and Marchetti, 2019; Pollicino et al., 2023), forensic analysis of automotive systems is often hindered by proprietary and undocumented protocols and data structures, hence motivating huge reverse engineering efforts. The topics of forensic analysis and reverse engineering in literature are widely addressed, with different proposals focused on the application of forensic techniques of specific contexts (Juma et al., 2020; Kwon et al., 2021; Babun et al., April 24-28, 2022), the definition of formal methods for the evaluation of filtering methodologies designed to aid the forensic analyst (Karafili et al., 2018), or for the evaluation of existing techniques (Pagani and Balzarotti, 2019; Mattei et al., 2022).

Focusing on the area of reverse engineering and its application on in-vehicle networks, the current state-of-the-art comprises different algorithms designed to extract information from Controller Area Network (Marchetti and Stabili, 2019; Pesé et al., 2019) or to identify the structure of in-vehicle networks (Kulandaivel et al., 2019). Moreover, existing literature already presented techniques to extract sensitive information exchanged via CAN communication between the navigation units and the dashboard display (Hoppe et al., 2012), or to identify the personal information gathered from in-vehicle networks from OEMs (Frassinelli et al., 2020). All these approaches intercept data *in motion*. However, the latter work sparked the idea of the investigation presented in this paper, focused on the analysis of data that are stored inside the vehicles instead of relying only on the information gathered from in-vehicle networks. Hence, the main focus of this work is on the IVI systems.

IVI systems have already been scrutinized in the cybersecurity literature, mainly to demonstrate vulnerabilities that might allow remote attackers to take partial control of a vehicle (Checkoway et al., 2011). An exploit for the communication protocols used in IVI systems is presented in Mazloom et al. (2016), where the authors demonstrated how to send remote messages on in-vehicle networks via a smartphone connected to the IVI system. Other papers demonstrate attacks targeting navigation systems via GPS spoofing (Zeng et al., 2018). However, these papers do not address the privacy of the data found inside the vehicle.

The privacy analysis of navigation systems is still an under represented topic. Current literature only partially addressed this aspect by focusing on the TomTom One (Hannay, 2008, 2009, 2017) and the TomTom NA mobile application (Le-Khac et al., 2014). Authors demonstrate how to extract sensitive information from the devices, however we remark that extracted information can not be used as forensic evidence since they are not coupled with time information. An experimental analysis of privacy-related information found in modern IVI systems is presented in Lacroix et al. (2016) and Whelan et al. (2018). The for-

mer describes the methodology to find privacy-related data on the Ford SYNC system, while the latter compares security solutions applied in a generic automotive system with solutions found on smartphone architectures. However, these works rely on a proprietary forensic tool developed by BERLA (2022) (which license is released only to government agencies) to extract data from the target IVI, thus preventing the scientific community from reproducing their experiments. Another work focused on the proposal of a methodology for digital forensic in the automotive fields is presented in Gomez Buquerin et al. (2021), where the authors exploited the absence of secure storage in modern IVI system to demonstrate that it is possible to extract information from the vehicle. However, they do not evaluate their methodology nor the implications of their findings on a real IVI.

Our work relates more closely to open methodologies for automotive forensics developed by security researchers. Within this field, previous work focus on reverse engineering methods to identify a defeat device to cheat emissions test (Contag et al., 2017). This work presents an analysis of two families of software defeat devices for diesel engines (one used by the Volkswagen Group and one found on Fiat Chrysler Automobiles) based on static firmware analysis. Another work targeting Volkswagen Group vehicles is presented in Jacobs et al. (2017), where the authors described the challenges associated with vehicle data forensics focusing on the IVI of a Volkswagen Golf. We remark that the proposed methodology does not allow to extract GPS logs from the analyzed IVI system.

Compared to the current state-of-the-art, this work focuses on the description of the steps required for extract and reverse engineer a proprietary encoding format used to store GPS data on the Mercedes-Benz $NTG5*2$ IVI system. First, we identify the main sources of information that can be used in a forensic environment to reconstruct the habit of the driver. Then we document the proprietary format used for storing GPS coordinates of each trip of the vehicle. We demonstrate how to correlate the extracted GPS logs with additional data sources found in the IVI system to enhance the travel history of the vehicle, and we demonstrate that the knowledge of the proprietary format can be used to extract personal information also from logically deleted data. Finally, we demonstrate how to exploit the structure of the database table storing the data to recover previously-deleted entries, thus increasing the number of valid traces that can be used for forensic applications. We remark that the findings of this work afflict all Mercedes-Benz vehicles mounting the same version of the IVI system. To the best of our knowledge, this is the first paper achieving similar results on a commercial IVI found in modern vehicles.

3. The Mercedes-Benz NTG 5*2 case study

In this section we present the process used for the forensic analysis of the COMAND APS $NTG5*2$ infotainment system installed in a Mercedes-Benz C-Class, MY 2016 and produced by Harman. We first describe the data acquisition and extraction process used to create a forensic copy of the infotainment system (Section 3.1), then we present a high level description of the content of the acquired disk image.

The data extracted from the vehicle is gathered over a week of usage for commuting between two cities in the northern part of Italy and a few trips during the weekend day within a single municipality. This results in the gathering of approximately 1250 kilometers worth of data with several stops. During the data collection process we activated and deactivated as many features available in the IVI system associated with location activities (such as the navigator system and the Bluetooth) every other day. However, we did not find any clear evidence that by deactivating these features the number of data gathered from the IVI system is reduced, thus leading to the conclusion that these types of data are gathered automatically and are not strictly related to driver-activated features.

Disclaimer. We remark that we were able to access the home and work addresses of many different drivers of the rented vehicle, and that by replicating the methodology presented in the following sections it is possible to extract these data from any $NTG5*2$ IVI system produced by Harman. To prevent mismanagement of personal information of people outside of this process, we only analyze GPS entries and log events generated during our rental period. As an additional countermeasure to prevent any leak of personal information, we stored a single forensic copy of the hard drive on an off-line storage disk with only one of the authors responsible to access the data. Moreover, we remark that despite this paper focuses on a single vehicle model, the issues that we identify afflict all vehicles mounting the same IVI system (NTG5*2) produced by a popular Tier-1 in the automotive ecosystem (Harman).

Fig. 1. Intermediate step of the extraction process.

3.1. Data acquisition and extraction

The first step in our analysis relies in the creation of a forensic copy of the $NTG5*2$ system, to prevent any data loss or damage while analyzing its content. We first extracted the head unit from the dashboard of the vehicle following repair steps found in publicly-available Web forums, thus requiring the removal of the central command panel to access the infotainment head unit as depicted in Fig. 1. Our analysis methodology could also by applied by remotely accessing the filesystem, without requiring physical interventions. We remark that remote access to IVI systems of Mercedes has been theoretically discussed (Lab, 2020), but there is no public demonstration of this attack. We acknowledge that full disk encryption (which is a widely deployed best practice) would prevent access to the data through offline physical disk copies. We also remark that the proposed approach does not depend on physical access but on the analysis of a limited number of files (the SQLite database analyzed in 3.1.1 and the event log files discussed in 3.1.2) independently of the method used to access them. As an example, any logical access to the files while the IVI system is on would bypass full disk encryption.

The head unit found behind the central panel is produced by *Harman Becker* (model $NTG5*2\ HU$, as found on the labels of the metal enclosure), equipped with a 2.5-inch 5400 RPMs hard disk drive. We acquired a forensic copy of this disk by using a USB-to-SATA adapter and the *EnCase* disk imaging tool.

In the preliminary analysis of the MBR partition table of the disk, we identified 4 partitions formatted with the *QNX6FS*, a filesystem typically used by the QNX OS 1. Partition #1 is the largest, and contains icons

and images used for the graphical interface of the infotainment system, such as navigation, media player, etc. It also contains databases storing the POI loaded by the car maker, a *GraceNote* music catalog, the user manual in HTML format, and dictionaries used for text-to-speech and voice commands. Partition #2 is the second-largest partition, and contains files and references to both system and user configurations and the travel history of the vehicle. Partition #3 is the third-largest partition of the disk, it appears to be empty. Partition #4 is the smallest partition of the disk, and contains system event log backups, compressed and archived on a regular basis. We remark that all of these partitions are used to store data used by the $NTG5*2$ system, with the firmware being stored on a different memory unit.

```
$ fdisk ewf1

Disk ewf1: 186.31 GiB, 200049647616 bytes, 390721968 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc0481de5

Device Boot     Start       End   Sectors   Size Id Type
ewf1p1             32 324841471 324841440 154.9G b1 qnx6fs
ewf1p2      324841472 363282431  38440960  18.3G b2 qnx6fs
ewf1p3      363282432 386719743  23437312  11.2G b3 qnx6fs
ewf1p4      386719744 390721535   4001792   1.9G b4 qnx6fs
```

Listing 1: Detailed description of the extracted disk.

In this work we focus on the personal data contained in the extracted disk. Following the definition of personal data available on the European's General Data Protection Regulation (GDPR) (European Parliament and Council, April 27, 2016) *"personal data" means any information relating to an identified or identifiable natural person [...] by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific [...] identity of that natural person*. This definition clearly indicates that, of all the data found in the disk image, the ones that can be categorized as "personal data" are the GPS coordinates found in *partition #2* and the content of the log files found in *partition #4*. Hence, in the following we discuss the process of extracting the GPS coordinates from the travel history of the user and the enhancement of this information with the content of the log files to produce a detailed forensic report of the system and its previous users. This combined approach allows us to fully reconstruct the travel history of a driver and to extend the scope of our analysis by including additional information that allow us to estimate the number of people inside the vehicle (given by the number of door openings before the vehicle is moved) and the presence of items in the trunk.

3.1.1. Travel history

The files with the travel history are located inside the *nav* folder of partition #2 2, namely *"trails.sqlite"* and *"trips.sqlite"*. These files are in the SQLite database format, and are used by the IVI system to log all the data coming from the GPS antenna. Each file has the same database structure, composed of three different tables: `Description`, `IncompleteTrail`, and `Trails`. The same folder contains additional elements related to the navigation system which are not of interested in our analysis.

```
$ tree ewf1p2/nav/

ewf1p2/nav/
|-- ClassicDbProvider.cache
|-- DealerPOI_ECE.db3
|-- driveshow/
|   |-- HU
|-- ico/
|   |-- POI_H_Prede_0524_082245_55x58_256.png
|   |-- ...
|   |-- POI_H_Prede_0524_082247_55x58_305.png
|   |-- RDNR_H_PosCt_0706_111833_72x55_255.png
|   |-- RDNR_H_PosCt_0706_111833_79x55_254.png
|-- sds
|   |-- genericInLang_hasId.template
|   |-- generic.template
|   |-- tempFile
|-- trails.sqlite
|-- trips.sqlite

4 directories, 59 files
```

Listing 2: Content of the `nav` folder.

The `Description` table contains metadata of the database as a key-value pairs encoded as a binary blob. The `Description` table contains 2 different columns, one called `Name` and the other called `Value`. This table has 7 entries, each one characterized by a different value of the `Name` column, encoded in ASCII plain text: (1) *Type*; (2) *Identifier*; (3) *AccessCounter*; (4) *FormatVersion*; (5) *VehicleNumbers*; (6) *SoftwareReleases*; (7) *RoadNetworkDatabaseReleases*.

The content of the `Value` column is encoded as an opaque binary blob, starting with a `01 01 01 00` hexadecimal header and followed by a different encoding for each field:

1. `Type`: contains the header followed by the null-terminated encoding of the `Trails` string (visible also in ASCII);
2. `Identifier`: contains the header followed by a counter word (little endian, in our data is equal to `00 0c`) expressing the size of the remaining field expressed in bytes. The first 4 bytes following the counter word are equal to the value encoded in the `FormatVersion` field;
3. `AccessCounter`: contains the header followed by a counter (double word little endian, in our data is equal to `00 00 07 D8`). We hypothesize this being used for counting the accesses to the database;
4. `FormatVersion`: contains the header followed by 4 bytes containing the value `08 01 01 00`. These 4 bytes are the same found at the beginning of the `Identifier` field (after the counter word), and we hypothesize being the version number of the infotainment system;
5. `VehicleNumbers`: contains the header followed by a null byte;
6. `SoftwareReleases`: contains the header followed by a byte containing the number of entries of this field (in our data, only 1 entry). Each entry is encoded with an initial counter word (little endian) expressing the size in bytes of the remaining of the entry. Following the counter we found an ASCII-encoded set of chars containing the build info of our infotainment system (`Build info: ntg5 /M060_B0 Wed 03/08/2017 18:26:09.39 @HIMGWSCC08 CL_4893156 CoC_Nav_P003B_17103a`);
7. `RoadNetworkDatabaseReleases`: contains the header followed by a byte containing the number of entries of this field, encoded similarly to the `SoftwareRelease` field. Each entry starts with an initial counter word (little endian) expressing the size in bytes of the remaining of the entry. Following the counter we found an ASCII-encoded set of chars containing a timestamp (the first one being `20180601044506` and decoded as `01 June 2018, 04:45:06` and the second being `20170514180531` and decoded as `14 May 2017, 18:05:31`).

The other two tables of the database (`IncompleteTrail` and `Trails`) are used to store the history of GPS coordinates in the vehicle, thus being the primary focus of this section. We hypothesize that the `IncompleteTrail` table is used to store the temporary readings from the GPS antenna, with the entries being moved to the `Trails` table after a trip is completed. We remark that this hypothesis requires online analysis to be verified, and that we based this assumption on the fact that the `IncompleteTrail` contains only the structure of the table, while `Trails` contains also data. Both tables present the same structure:

- *TrailID*: identifier of the trail and primary key of the schema;
- *DriverID*: identifier of the driver (always equal to 0);

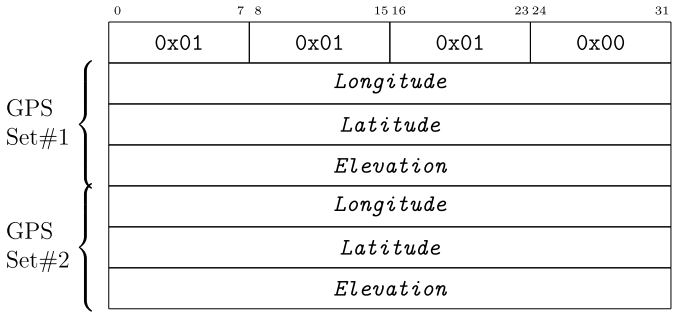

Fig. 2. Final format of the reversed *bounding* field.

- *Length*: length of the trail expressed in centimeters;
- *Begin Time*: timestamp of the first GPS entry;
- *End Time*: timestamp of the last GPS entry;
- *Bounding Box*: area containing the GPS entries;
- *Path*: binary blob composed of GPS coordinates;
- *Valid*: boolean value (always equal to 1).

The most interesting columns (*Path* and the *Bounding Box*) are encoded in a proprietary format and converted into an opaque binary blob, thus requiring a reverse engineering process to extract GPS coordinates.

This process starts with the assumption that the value of the last entry of the database in the *Bounding Box* field includes the GPS coordinates of the location in which the vehicle is parked before extraction of the head unit. This assumption helped us to identify the encoding structure, which is then used to extract all the GPS entries stored in the *path* column.

We hereby present the details of the encoding used to save the *bounding box* and *path* columns of the `Trails` table of the database, obtained after a manual reverse engineering process.

Bounding Box Each entry in the *bounding box* field starts with four bytes set to `01 01 01 00` followed by two set of GPS coordinates defining the bounding box of its relative trail. The set of GPS coordinates are encoded as three 32-bit little-endian integers. Each set of coordinates contains the longitude, latitude, and elevation. The elevation field is encoded as an unsigned integer and its value is measured in centimeters. Reversing latitude and longitude required an additional effort of extensive trial and error based on the known coordinates of the place in which we extracted the IVI from the vehicle. Latitude and longitude are encoded as a signed integer value that needs to be converted to a GPS coordinate using the formula:

$$val_d = \frac{val_e * 180}{Int32.MAX\_VALUE} \quad (1)$$

where val_d is the decoded value of the GPS coordinate expressed in decimal degrees, val_e is the value encoded in the binary word expressed as a signed integer, 180 is the maximum value of the GPS coordinate (used for both longitude and latitude), and $Int32.MAX\_VALUE$ is the maximum signed integer value that can represented in a signed 32-bits integer. To the best of our knowledge, this encoding is not standard nor commonly used by other software and libraries related to GPS data. As a final remark, the value of the elevation field found in the GPS coordinates in the bounding box is always equal to 0 since the bounding box only defines the two-dimensional area covered by the *path* field. Fig. 2 shows a graphical representation of the content of the reversed *bounding* field.

Path Despite the GPS coordinates in both *Bounding Box* and *Path* partially shares the same format, there are additional information available in the *path* table that are worth further inspections. The data encoded in the binary blob of the *Path* field is organized as a series of events. Each entry in the database starts with a fixed header equal to `04 01 01 00`, and is followed by an unsigned 16-bit little endian integer value containing the number of segments encoded in the binary string. Different segments encode different information and have different length, specified in the first 32 bits of the segment. Additional 32 bits are found at the end of the segment (repeating the size of the segment) as a terminator. The size value is expressed in bytes and specifies the size of the segment including both delimiters. Following the initial size field, there are three additional fields of 32 bits each containing different values, with the second word being always equal to 0. Following these three words there is a list of navigation events containing different data recorded from the GPS antenna. We were able to correctly identify 7 different types of events, each one with its own structure and semantics. The structure of the events is composed by at least two fields, namely the *identifier* and the *distance*, while following additional fields are related to the event type. The identifier is found in the first byte of the event, and identifies the size of the additional fields following the distance, which is directly encoded after this value. The distance field contains the absolute distance (expressed in centimeters) from the start of the recording. These two fields are part of the *basic structure* of the events, with a fixed size of 40 bits (8 bits for the identifier and 32 bits for the value). The values of the identifiers (i.e. the event IDs) that are found in the path field of the analyzed databases are hereby presented:

- **Event 01**: contains the GPS coordinates, expressed as longitude, latitude, and elevation. The GPS coordinates are encoded as already described for the *bounding box* field, with a set of three different 32 bits integers. The total size of this event is 3×32 bits (for the GPS coordinates) plus the size of the basic structure;
- **Event 02**: contains the milliseconds since the start of the recording. The millisecond value is encoded as an unsigned 32 bit integer. The total size of this event is 32 bits plus the size of the basic structure;
- **Event 03**: contains timestamp of the start of the recording. This value is encoded in the 32 bits following a 32 bit value equal to zero. The total size of this event is 2×32 bits plus the size of the basic structure. We remark that this event can only be observed in events having "distance" value equal to 0.
- **Event 14**: does not contain any field except the basic structure;
- **Event 15**: contains 32 bits that are not found interesting in any particular scenario. The total size of this event is 32 bits plus the size of the basic structure;
- **Event 16**: does not contain any field except the basic structure;
- **Event 18**: contains a single byte encoding a boolean value. However, we only found this event twice in all our data, thus its actual meaning is still unknown. The total size of this event is 8 bits plus the size of the basic structure.

A graphical representation of the format of the *Path* field is depicted in Fig. 3.

3.1.2. Event logs

The event log files are located inside multiple archives placed in the root directory of partition #4. Each archive is named following a precise pattern, composed of a progressive identifier, the source device (HU, *Head Unit*), a timestamp (encoded in UTC), and a system code 3.

```
$ tree ewf1p4/

ewf1p4/
|-- ArmMultStartup_0
|-- curtrigger
    |-- arm
    |-- eventlog
    |-- intel
    |-- scr
|-- eventlog
    |-- eventlogringbuffer.dat
|-- FF-ARM.log
|-- FF-COMPLETE.log
|-- FF-COPY.log
|-- FF-NAND.log
```

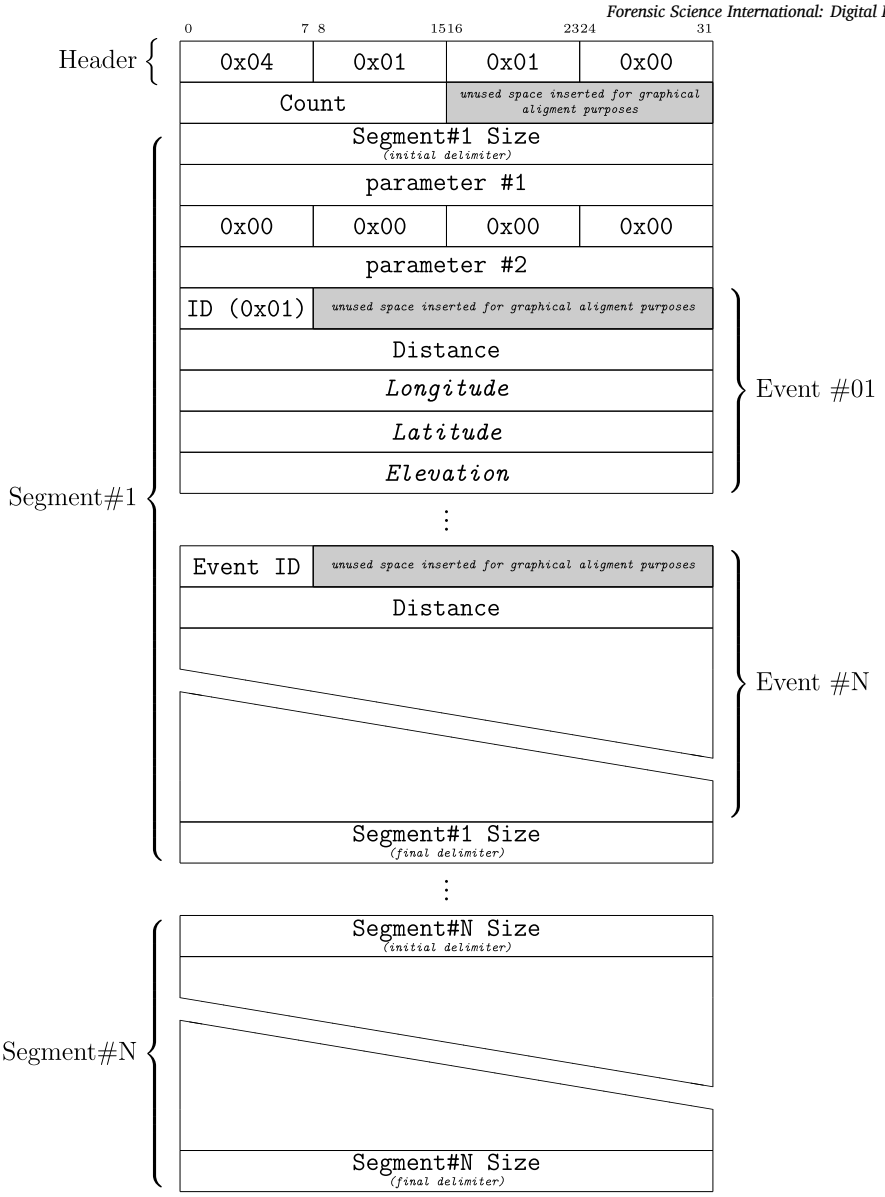

Fig. 3. Example of the *Path* format containing an Event with ID `01` in the first segment.

```
|-- FF-NOR.log
|-- FF-SYSLOG.log
|-- trigger_001_HU_20110101_000033_SYS_CA01.tgz
|-- trigger_002_HU_20110101_000032_SYS_CA02.tgz
|-- ...
|-- trigger_998_HU_20201205_132328_SYS_CWD.tgz
|-- trigger_999_HU_20201205_144507_SYS_CA05.tgz

6 directories, 1080 files
```

Listing 3: Content of Partition#4.

Each archive contains 4 different folders, with only the *eventlog* folder containing data. We remark that the same folder structure can also be found in the *curttrigger* folder of partition #4, which we hypothesize being the temporary folder used by the logging system before an archive is produced. The system logs found in the archives include many information about the vehicle: odometer data, Vehicle Identification Number (VIN), early memory flashing and factory initialization sequences, detailed system library enumeration, display management, multimedia streams, and events of the internal communication buses and IP networking. Each one of these events is associated with a relative timestamp since the boot of the system, which is relative to the last ignition. These logs are stored as progressive snapshots of a ring buffer, which is located in the *root* directory of the *eventlog* folder in the *eventlogringbuffer.dat* binary file. Each archive contains both binary and text representation of the same log entries, for a total of up to 20480 entries.

The event log file are structured as tabular-separated files, with each column representing a different value:

- `line counter`;
- `milliseconds since boot`;
- `event category` (formatted as a hexadecimal code);
- `text description of the event`;
- `payload of the event` (formatted as a hexadecimal string).

The same information is available in the binary representation of the text file, each row corresponding to a 10-byte value, encoding the milliseconds since boot in the first 32 bits, the event category as a 16 bits field, and the event payload in 32 bits.

The content of the event logs is used a journal file to maintain data about all the different interactions between the head unit and other vehicle subsystems, including events gathered from internal vehicular networks such as MOST, Ethernet, and CAN. While some of these events

Table 1
Relevant system log event types.

Category		Type		Payload	
05	Vehicle	01	Ignition	0x00000000 0x00010000 0x00020000 0x00040000 0x00050000	<LOCK> <OFF> <ACC> <ON> <START>
		03	FrontDoor	0x01010000 0x01020000 0x02010000 0x02020000	Left closed Left open Right closed Right open
0f	General	02 06	GpsTimeAvailable InstrumentCusterTimeAvailable	see Fig. 4	YYYY-MM-DD hh:mm
09	Display	21 11 01	RR_DisplRequest RL_DisplRequest HU_DisplRequest	0x00000001	Open
		02	HU_DisplStatus	0x00000001 0x00000003	Closed Open
		03	HU_BacklRequest	0x00000000	Off
		04	HU_BacklStatus	0x00000001	On
01	StartUp	01	CanSignal	0x----00 0x----01 0x----02 0x----ff	Ignition State HK On (Ctrl-C)[a] Door Modules SNA
		03	Reset	same payload as SupDAI_ResetTGW	
02	ShutDown	01 06 07	CanSignal Error Voltage	0x----00 0x33811e01	

[a] Hardware Keyboard.

are related to technical information (e.g., diagnostic messages), other are associated to the diver actions (e.g., commands and status probe messages about *Head Unit* and *Rear* display deployment).

We were able to reverse engineer the value of the `category`, `type` and `payload` fields by cross-referencing the different values with the textual description, thus allowing us to identify the source system of the logged event. While some events include a textual description of the payload, allowing a correct identification of the meaning of the hexadecimal string, others can not be decoded without additional information. Table 1 summarizes the entries that are directly identified from their text description, representing both hexadecimal code and textual description. We remark that the format used to store the content of the event logs does not follow the standardized format Diagnostic Log and Trace (DLT) (AUTOSAR, November 25, 2021) nor other standardized formats for logging information (Gerhards, 2009; Gurbani et al., 2013) that we are aware of.

Following the analysis of the reverse engineered type of events and their content, we selected events related to CAN communication as the most interesting source of information from a forensic perspective. In particular, by accessing these events we can extract all activities related to opening and closing of the doors and of the trunk of the vehicle (*door open* and *door closed* events), and the exact moment the vehicle is started and stopped (*engine status* events). Events related to the *door* and the *engine* status are always found at the beginning of the log file, while after the ignition process is completed it is possible to see events related to the boot process of the head unit.

Another set of interesting log entries are the ones related to the *TimeAvailable* family, collecting timestamps from both the *dashboard* and the *GPS* antenna. The values of these entries are not encoded in a standard format, thus we applied the same assumption adopted for the decoding the *path* field to reverse engineer their encoding. The reverse engineered content of the *TimeAvailable* event logs contains five big-endian bit fields, each one representing a single element of the timestamp as an unsigned integer (right to left): *minute* (6 bits), *hour* (5 bits),

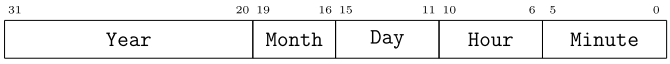

Fig. 4. `General_TimeAvailable` payload binary format.

day (5 bits), *month* (4 bits), and *year* (12 bits, mostly used to fill all the leftover space).

A graphical representation of the encoding of the `TimeAvailable` field is presented in Fig. 4.

4. Implications on personal privacy

In this section we demonstrate how the data extracted from the $NTG5*2$ infotainment Head Unit can be used to identify sensitive information of the driver. In particular, we present the data fusion and enhancement process used to extract the trips from the database in Section 4.1, while a representative analysis of one of the reconstructed and enhanced traces is presented in Section 4.2.

4.1. Data fusion and enhancement

The whole process is composed of two main steps: *data preparation* and *cross-reference and presentation*. The *data preparation* step describes the process required to prepare the data from the different sources for the fusion operation, while the *cross-reference and presentation* phase shows the merging process, resulting in a single timeline that can be used by forensic analysts for a full analysis of the actions of the driver of the vehicle.

4.1.1. Data preparation and cross-reference

This section provides the details about the pre-processing of both `Trails` table and *event* logs. In the first phase of this step, the content of the `Trails` table is imported directly from the SQLite database, decoding it as described in Section 3.1.1. The data fields required for the data fusion procedure are:

- `start timestamp`: available in both the `Trails` table and the **event #03** event entry, denotes the start of the logging process expressed in locale;
- `distance`: available in all the trail events, denotes the relative distance (expressed in meters) from the beginning of the trail;
- `uptime`: available in **event #02**, denotes the relative time (expressed in milliseconds) from the beginning of the trail;
- `GPS coordinates`: available in **event #01**, denotes the coordinates (expressed in longitude, latitude, and elevation) of the event entry.

Each event is associated with a timestamp which is used to map all the events on a single timeline. Each entry is associated with its own timestamp by using the `distance` values and their relative `uptime` as references. We remark that multiple *trail event* with different IDs might share the same `distance` value in the same *path*, thus we assign the same timestamp to all the trail events at the same distance. The timestamp is associated with the set of trail events by computing the absolute timestamp Ts as $Ts = ts_{start} + ts_{up}$, where ts_{start} is the start timestamp and ts_{up} is the value of the `uptime` field. When the value of the `uptime` field is not available, we extrapolate it with its two nearest values by using a linear interpolation technique. Despite linear interpolation might introduce small approximation errors, we remark that the difference between available uptimes is always in the millisecond range, hence the approximation error is negligible. At the end of this process, the content of the `Trails` table is converted into a human-readable format where each event of the *path* field is associated with an absolute timestamp.

The second step of this phase focuses on the preparation of the log event files for merging and presentation. Different log files might contain overlapping entries resulting from the ring buffer structure used in the logging process. Hence, as a preliminary step, we merge all the logs into a single event log file to remove redundancy and duplicated events. After this step, the tool separates the events in different files according to their *power cycle*, i.e. the different sessions of the Head Unit. Entries related to the same power cycle exhibit an increasing value of the *elapsed milliseconds from system boot* value, resulting in the reset of this value in the next power cycle. Following this separation process, each event of the logs is associated with an absolute timestamp, computed as the time difference of the event with the *millis from system boot* of the `TimeAvailable` entry (decoded as already described in Section 3.1.2) of each power cycle. The absolute timestamp Ts_i of the i_{th} entry is computed as $Ts_i = ts_{TAP} + \Delta millis$, where ts_{TAP} is the timestamp value encoded in the `TimeAvailable` payload and $\Delta millis$ is the difference in milliseconds between the *elapsed time* of the row of the `TimeAvailable` message and the i_{th} row. At the end of this process each event (including its category, type, and payload) is associated with a timestamp.

4.1.2. Data fusion and presentation

After the data preparation process is concluded, the tool merges entries from both data sources based on the values of the timestamps in single timeline. For each relevant log entry, a set of coordinates is derived with a linear interpolation from the nearby coordinates, thus associating a physical position to the entry. The result of this process is a file where each entry, being either a GPS coordinate extracted from the `Trails` table or the *event log* files, is associated with a timestamp and a location. This allows a forensic expert to analyze the content of these files to identify the behavior of the driver and interesting location (as show later in Section 3). This tool exports the output of the fusion process in the GPX format (an open format GPS data based on XML), and is extended to support custom annotations. Each GPX file describes a single entry of the `Trails` table, thus representing a single trip enhanced with event logs. Each event of the trip is identified by the GPS coordinates (expressed in decimal degrees), elevation (expressed in meters), and time (expressed in UTC). Event logs have two additional fields: the `<name>`, filled with the category and type of the event, and the `<description>`, if available from the original logs.

4.2. Trip reconstruction

In this section we present a detailed analysis of a set of traces generated for these experiments and extracted from the IVI system. The traces used in this analysis are gathered in the same day and are enhanced using both *event* logs and OSINT sources. To avoid exposing personal information of previous drivers of the rented vehicle, the analysis presented in this Section is based on the only data generated during our tests from one of the authors of this work altering the marker position (marker *A*) of his home address to the location of a public parking. The path traveled by the car and the main stops are shown in Fig. 5, in which box *A* represents the full path while boxes *B* and *C* are augmentation of the two main areas covered in *A* to improve readability.

The vehicle starts at *08:32* at marker *A*, which is the altered location of the home address of the driver. Before the ignition of the vehicle, we observed the opening of the trunk and, 8 seconds later, the opening of the front left door. Both doors are closed after 10 seconds. This allows us to conclude that the driver was alone in this travel and that carried something with him. After a couple of minutes, the vehicle stops at marker *B*, which is a gas station. At the gas station, the front left door is open and closed, and after 8 minutes the driver resumes his commute. We observe that only the front left door is opened and closed again before the ignition of the vehicle. The vehicle leaves the gas station at *08:48* and arrives at marker *C* (a parking lot in the proximity of the engineering department of the University of Modena and Reggio Emilia) at *09:42*. Upon arrival at the University, we can see the opening and closing of both front left door and trunk. The vehicle remains parked until *14:34*. Before resuming the driving, we can see the opening and closing of the trunk and then both front doors, implying that there is another passenger in the vehicle. After a short distance, the vehicle stops at marker *D*, which is a data center nearby the train station of Modena. Here, we observe that the front doors and the trunk are opened and closed, with the vehicle left parked until *17:24*. The vehicle is then driven back to marker *A* (masked home address) with only the driver inside (only front left door opens and closes before ignition), arriving at destination at *18:32*. Here the driver stays in the vehicle (the front left door does not open) and, after a couple of minutes, a passenger enters the vehicle (the front right door is opened and closed), and the vehicle resumes driving toward marker *E* (a grocery store). The vehicle is left parked at the grocery store for approximately 40 minutes, with both driver and passenger leaving the vehicle. At *19:42* we observe the trunk being opened (as well as both front doors) and then the vehicle is driven towards marker *F*, arriving at *19:57*. Here we can see that only the passenger leaves the vehicle for approximately 10 minutes. At *20:10* the vehicle is driven to marker *G* where, once again, only the passenger is observed leaving the vehicle. After that, the vehicle is left at marker *A* at *20:31*, where we can see both driver and passenger leave the vehicle, the trunk being opened, and the car remains stationary until the next day.

We remark that reports with the same level of detail can be generated for every other day, including those in which the car was driven by people outside of our research group (presumably other car rental customers). We also remark that it is possible to extract useful information by just analyzing a small subset of data. For example, with a simple statistical analysis of raw GPS coordinates it is possible to identify the home location of the driver (marker *A* is found frequently as the first and last point of each day). Hence, storing this data on any IVI system that does not properly manage these privacy-sensitive data could easily expose sensitive locations and habits.

5. Recovery of deleted traces

In the final step of our forensic analysis of the $NTG5 * 2$ infotainment system, we demonstrate how to exploit the data structure used to save the entries in the database to extract additional references from the database by employing a novel carving technique. At first we present

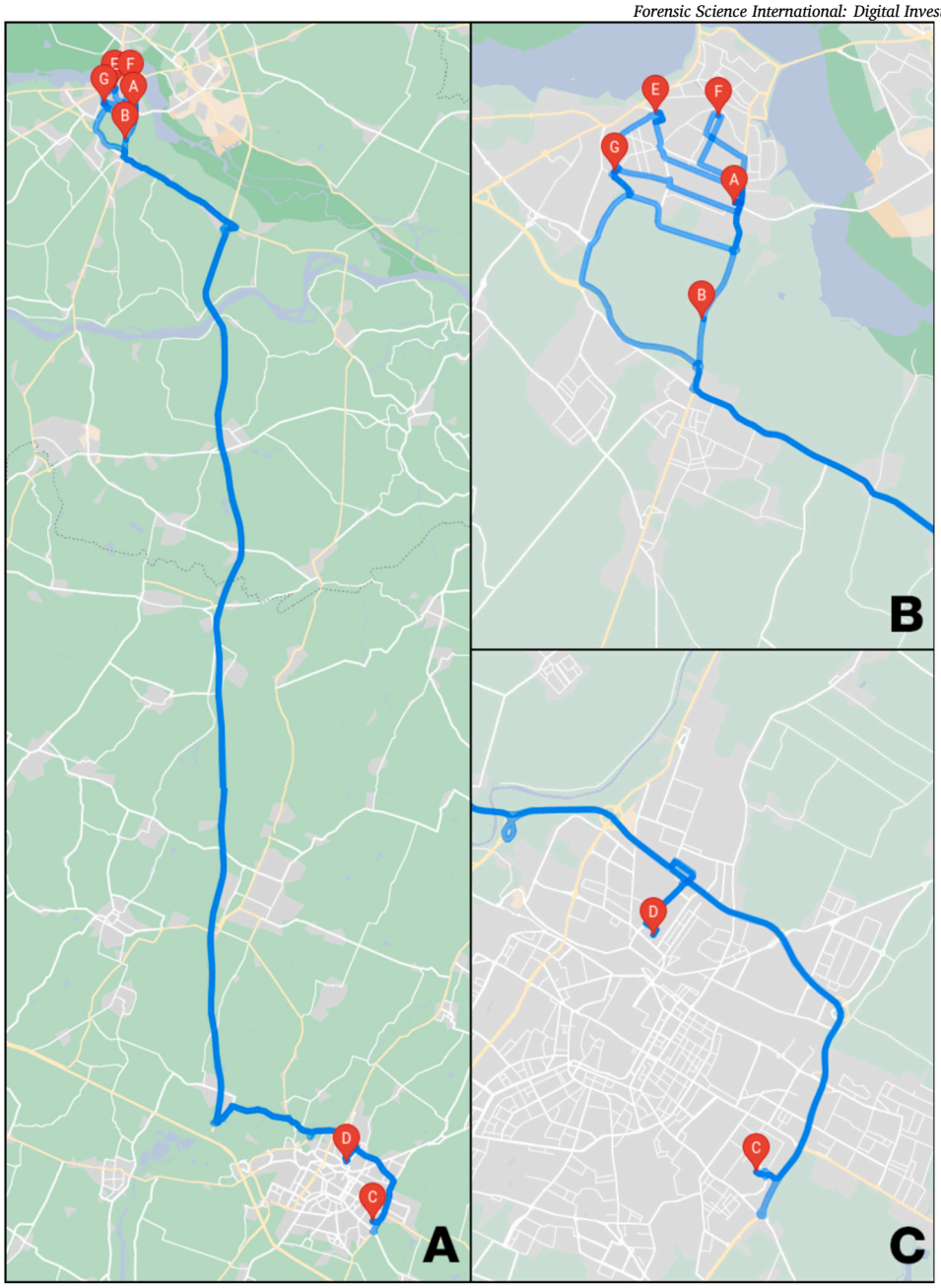

Fig. 5. Graphical representation of a GPS trace extracted from the SQLite database.

the format of the SQLite database in Section 5.1, then we present our carving method in Section 5.2. Finally, we compare the number of extracted entries of our carving method with existing tools and previous research works for the recovery of deleted traces from SQLite databases in Section 5.3.

5.1. SQLite database file format

To extract additional GPS entries we focus on how data are stored by the SQLite database engine to fully understand how entries are deleted from the database. In case a *rollback journal* file is available in the analyzed system, that file should have the highest priority to identify data that might be deleted from the system. However, since there is no *rollback journal* file in our image, we focus on the *main database file*. A generic SQLite database file contains one or more pages of equal size, numbered starting from 1. While the size of all the pages is the same, the memory space in page #1 is lower than the others since the *SQLite database header* is found in the first 100 bytes of that page. The SQLite database header contains different information (SQLite - Database File Format, June 18, 2004), such as the size of each page, which is encoded in a 2-byte integer value found in the first bytes of the database file (each page has a minimum size of 512 bytes and with a maximum of 2^{16} bytes). The maximum number of pages found in a database file is $2^{31} - 1$, despite it is more common to reach the maximum file limit (size of the page × number of pages) due to restrictions of the underlying file system. A detailed representation of the SQLite database header and the corresponding values in our *trails.sqlite* file are presented in Table 2, where the values depicted in the columns offset and size are expressed in bytes.

For the identification of "deleted" entries we focus on the fields containing information related to the *freelist*, a data structure containing the list of all unused pages (i.e. pages that can be overwritten without losing any information). An unused page is either a page that was never used

Table 2
SQLite database header (SQLite - Database File Format, June 18, 2004).

Offset	Size	Description	Value
0	16	The header string	`SQLite format 30`
16	2	Size of each page of the database	`0x04 0x00`
18	1	File format write version	`0x01`
19	1	File format read version	`0x01`
20	1	Number of bytes unused "reserved" at the end of each page	`0x00`
21	1	Maximum embedded payload fraction	`0x40`
22	1	Maximum embedded payload fraction	`0x20`
23	1	Leaf payload fraction	`0x20`
24	4	File change counter	`0x12a4`
28	4	Size of the database file in pages	`0x21 0x59`
32	4	Number of the first freelist trunk page	`0x07 0x9c`
36	4	Total number of pages in the freelist	`0x01 0xcd`
40	4	Schema cookie	`0x03`
44	4	Schema format number	`0x04`
48	4	Default page cache size	`0x00`
52	4	Page number of the larges root b-tree page	`0x00`
56	4	Database text encoding	`0x01`
60	4	The "user version"	`0x00`
64	4	Reserved for incremental-vacuum mode	`0x00`
68	4	Application ID	`0x00`
72	20	Reserved for expansion	`0x00`
92	4	Version-valid-for number	`0x12 0xa4`
96	4	SQLite version number	`0x2d 0xe2 0x21`

by the database or a page containing entries that have been logically deleted.

5.2. Identifying removed entries from an SQLite page

The identification of removed entries from SQLite pages starts with the analysis of the pages in the *freelist*, looking for any row of the table stored in the "unallocated" space by exploiting the *b-tree* data structure. SQLite stores the *b-tree* page header in the first bytes of the page (or after the database file header in the first page). The size of the *b-tree* data structure is 8 bytes for *leaf* pages (i.e. pages containing data) or 12 bytes for *interior* pages (i.e. pages used for traversing the data structure). In our process we only look for pages in the *freelist* structure belonging to the *b-tree* leaf type, with a page header of 8 bytes. The *b-tree* page header contains the following fields:

- **B-tree page type**: One-byte flag indicating the page type. Since we are interested in the analysis of leaf b-tree pages, this value should be equal to $0x0D$;
- **Page freeblock**: Two-byte integer containing the start of the first freeblock on the page (0 if there are no freeblocks);
- **Number of cells**: Two-byte integer containing the number of cells on the page;
- **Content start**: Two-byte integer containing the start of the cell content area;
- **Free bytes**: One-byte integer containing the number of fragmented free bytes in the cell content area.

After the *b-tree page* header, the cell pointer array is encoded. This array has a size of $K * 2$ bytes (with K being the number of cells in the page) and contains the offset of each cell of the page. The cell of a leaf *b-tree page* contains three information: (i) the total number of bytes of the payload, (ii) the value of primary key of the data in the cell, (iii) and the payload. While the total number of bytes of the payload allows us to understand directly if a cell contains any data, to identify the content of the payload it is necessary to analyze its value. The payload of a cell is encoded in the *record format*, representing a sequence of values corresponding to the columns of a table, specifying also the number of columns, the datatype of each column, and its content. The record format is composed by a header and a body. The header starts with a single byte containing the length of the header in bytes, including itself. Following this value, there is a variable number of bytes representing the columns of the table. These additional bytes are encoded as "serial type" numbers, and identify the datatype of each column. By referencing the table presented in SQLite (June 18, 2004) (Section 2.1) and the columns of the `trail` table presented in Section 3.1.1 we can define the values of the record header as follows:

- *TrailID*: $0x00$;
- *DriverID*: $0x08$ (corresponding to the value 0);
- *Length*: all types between $0x01$ and $0x06$ (all representing integers with different length);
- *Begin Time*: $0x04$;
- *End Time*: $0x04$;
- *Bounding Box*: $0x44$, representing a binary blob of 28 bytes (the length is evaluated as $(N\text{-}12)/2$ bytes, where N is the encoded value);
- *Path*: any even value N higher than $0x0C$, representing a binary blob of $(N\text{-}12)/2$ bytes;
- *Valid*: $0x09$ (corresponding to the value 1).

Hence, any record header containing this information identifies an entry of our `trails` database. By looking at this value in the cells contained in both *freeblock* and *content* fields we are able to recover entries that have been logically deleted from the database. The pseudo code of both carving and field extraction functions are presented in Algorithm 1.

By applying this process on our data, we were able to recover a total og 13 additional entries of the *Trails* table.

5.3. Comparison with existing tools

To verify the effectiveness of our SQLite extractor we compare the performance of our carving methodology with other tools publicly available: FQLite 2.6.1 (Pawlaszczyk and Hummert, 2021), Undark 0.7.1 (Daniels, 2023), SysTools SQLite Database Recovery $v1.2$ (SysTools, 2014), SQLite Deleted Records Plugin for Autopsy $v2$ (McKinnon, March 19, 2019), SQLabs SQLite Doctor $v1.4.1$ (SQLabs, 2023), SQLite Deleted Records Parser $v1.3$ (De Grazia, June 22, 2015), (bring2lite, August 05, 2019), SQLite Dissect (DC3 - DoD Cyber Crime Center, 2024) (by using both table and freelist carving options), and the recovery option included in the SQLite CLI since version 3.29.0 (SQLite Consortium, February 25, 2025).

Results of the comparison with our tool against previous works are summarized in Table 3. In particular, the results presented in Table 3

Algorithm 1 NTGCarver.

```
function EXTRACT_FIELDS(r, hSize, dSize, lSize)
    off ← hSize
    tId ← off
    dId ← r[off : off + dSize]
    off ← off + drivSize
    len ← r[off : off + lSize]
    off ← off + lSize
    bTime ← r[off : off + 4]
    off ← off + 4
    eTime ← r[off : off + 4]
    off ← off + 4
    bBox ← r[off : off + 28]
    off ← off + 28
    path ← r[off :]            ▷ All remaining part of the record is the path
    return tId, dId, len, bTime, eTime, bBox, path
function SQLCARVE(database.sqlite)
    dbPages ←extract_pages(database.sqlite)
    for page in dbPages do
        if page.bytes[0] == 0x0d then   ▷ Check if the first byte of the page contains the "table leaf" value
            records ←deleted_records(page)
            carved ← EmptyList()
            for r in records do
                rHead ←extract_header(r)
                hSz, dSz, lSz, vVal ←sizes(rHeader)
                if validVal == 0x09 then
                    t ←extract_fields(r, hSz, dSz, lSz) ▷ Extracts and create a new trail from the record
                    carved.append(t)
    return carved
```

Table 3
Comparison of existing tools for SQLite data recovery against the implementation presented in this work.

Tool	Imported	Valid	Duplicate	New
Pawlaszczyk and Hummert (2021)	296	296	3	10
Daniels (2023)	117	53	2	3
SysTools (2014)	283	283	0	0
McKinnon (March 19, 2019)	283	283	0	0
SQLabs (2023)	5	5	0	0
De Grazia (June 22, 2015)	283	283	0	0
bring2lite (August 05, 2019)	212	212	2	9
DC3 - DoD Cyber Crime Center (2024)	299	298	3	12
SQLite Consortium (February 25, 2025)	295	295	2	10
NTGcarver	301	299	3	13

show, for each tool, the total number of imported entries from the database (*imported*), the number of entries with valid metadata (*valid*), the number of duplicate entries (*duplicate*) and the number of new entries recovered with the tool (*new*). We remark that the overall number of entries found in the database is equal to 283, and that a recovered new entry is considered such if it has valid metadata and if the identifier of the entry is not already found in the original 283.

From the results presented in Table 3 we can observe that many tools are able to extract at least the entries found in the database, with the only not being able to extract all the original entries are undark (Daniels, 2023), SQLite Doctor (SQLabs, 2023) and Bring2Lite (bring2lite, August 05, 2019). We remark however that the last update of both undark (Daniels, 2023) and Bring2Lite (bring2lite, August 05, 2019) is more than 8 and 5 years ago respectively, and that we used the free version of SQLite Doctor, which could have some limitations in terms of recovered entries. Another interesting result is that, of all the tools being able to recover at least the traces available in clear in the database, the recovery option included since version 3.29.0 in SQLite and FQLite (Pawlaszczyk and Hummert, 2021) are able to achieve almost the same results of the state-of-the-art tool SQLite Dissect (DC3 - DoD Cyber Crime Center, 2024).

On the other hand, our approach based on the identification of the data structure for recovering deleted entries is able to identify one additional entry compared to SQLite Dissect (DC3 - DoD Cyber Crime Center, 2024) and 3 more entries compared to the recovery option available in the SQLite CLI (SQLite Consortium, February 25, 2025). Upon manual inspection of the recovered entries, we remark that their content is comparable to the ones available in the database, thus being possible to extract additional coordinates from the $NTG5*2$ infotainment system. We also verified that the additional entries recovered with our tool are associated with a timestamp that is 2 years before the manual extraction of the ECU from our vehicle.

6. Countermeasures

In this section we discuss possible countermeasures to prevent access to privacy-sensitive data stored in the IVI system. We distinguish between *architectural* solutions 6.1 and *functional* solutions 6.2. In both scenarios, we consider four actors, namely the vehicle *owner*, the *OEM* (i.e. the car maker), a *higher authority* (i.e. a third party that requires higher privilege access than the car maker) and a generic *third party* with no particular requirements other than accessing the data. All the countermeasures discussed in this section assume the availability of an hardware security module (HSM) on the IVI system. This assumption is compatible with hardware commonly used for automotive grade Electronic Control Units. We will also compare our discussion with the views of the European Automobile Manufacturers' Association (ACEA) on sharing access to in-vehicle data (ACEA, 2024b).

6.1. Architectural solutions

We consider two different scenarios, one where the access to the data generated from the infotainment system is available remotely (e.g., via telematics services), while the other scenario discusses direct access via a local interface. We remark that the categories of data presented in ACEA (2024b) only cover the scenario where data is managed directly by the OEM via the Extended Vehicle (ExVe) model, while the countermeasures discussed in this section also cover data accessible through any physical interface different from the On-Board Diagnostic port (included in the ExVe model).

Remote access. This scenario covers data accessible through any remote interface included in the EVI, where both third parties and OEM can access the vehicle data through the same interface (managed by the OEM) to access data available on the ExVe interface (data level *A* ACEA, 2024b). As discussed in ACEA (2024b), in this scenario the owner's of the data are required to provide explicit consent to the OEM to share their data with every generic third parties entity, allowing the owner to select which third parties can access privacy-sensitive data. Access to the vehicle data via the ExVe interface should only be used to read data from the IVI system in an unidirectional way, and that communication from the ExVe interface to the IVI system should not be allowed. In the worst case scenario where a secure link is exploited by a malicious entity, the access to the vehicle data is limited to the only data available on the ExVe interface for the exploited third party. With the simple countermeasure of having the OEM managing an access control list for every different third party, access to the data available on the ExVe interface can be limited in time (i.e., a third party is authorized to access the ExVe interface for a limited time) and completely revoked in case of the worst case scenario.

Direct access. In this scenario we consider the data generated by the IVI system that can be made available via the ExVe interface but has not yet been transferred to the interface (data level *B*) and data generated by applications hosted in a vehicle's ECU (data level *C*). In this scenario, the data can be accessed only via the OBD port (covered by the ExVe model) or any other physical interface (e.g., a JTAG connection on the ECU, not covered by the ExVe model) as remote access via the ExVe is not available. We discuss this scenario by considering the IVI system

leveraging of full disk encryption to prevent extraction of sensitive data in case the disk is physically extracted from the system. We remark however that any logical access to the files while the IVI system is on would bypass full disk encryption (as discussed in Section 3.1), thus failing to prevent unauthorized access to these data while the IVI is fully operational. In this scenario, access to the data should only be granted to the OEM and the higher authority (either with a separate physical access interface or via a shared interface managed by the OEM), while the owner should be allowed to configure different options to select the data that can be stored on the ECU (more on this in 6.2). While stored data should be encrypted to prevent unauthorized access to the stored information, the OEM should be the only actor with the ability to decipher the data on the IVI system, while higher authorities should be able to leverage the OEM, by either obtaining the deciphered data directly from the OEM or by obtaining the decipher keys. As prominent example of this type of protection is implemented by TomTom, where GPS traces are stored in a proprietary format (*triplog* (TomTom Forum, October 03, 2010,N)) which is encrypted with a public key encryption mechanisms (ForensicFocus, December 02, 2011; GIS StackExchange, November 19, 2013), which can only be deciphered with the corresponding private key managed by TomTom. Despite this solution would prevent any unauthorized access to the data stored on the vehicle, we remark that the OEM should only store data that are compliant with existing regulations and laws, such as the European GDPR (European Parliament and Council, April 27, 2016), California's CCPA (California State Legislature, November 03, 2020), or Japan's APPI (Personal Information Protection Commission, June 2020).

6.2. Functional solutions

In this section we will discuss about possible solutions at a functional level, mostly impacting the *owner* perspective to address the privacy of data stored in modern IVI system. We will organize this discussion by considering different aspects and configuration options that the *owner* of the vehicle should be able to access to ensure that privacy-sensitive data are stored in accordance with its interests.

Optional data gathering. The *owner* of the vehicle should be able to choose whether to participate or not in the data gathering process of the OEM, and whether these data can be shared with the third parties. This choice should be presented on the IVI system at its first configuration from the *owner* and should be possible to modify it at any given time. From an user perspective we remark that some *owners* might be interested to prevent data from a particular trip to be gathered by the IVI system. In this case, instead of requiring the *owner* to opt-out and eventually opt-in back once the trip is completed, *OEMs* should implement an *incognito mode* for data collection, preventing any data to be recorded until the vehicle is turned off (or until deactivation of the *incognito mode*).

Data gathering parameters. *Owners* should be able to configure different parameters related to the data collection from the IVI system, such as:

- *frequency of the collected data:* set the time interval between consecutive sensors readings (e.g., time between consecutive GPS readings);
- *precision of the collected data:* set the accuracy of the sensor's readings (e.g., number of decimal digits in GPS readings);
- *types of the collected data:* select the sensor's readings that can be collected (e.g., location data, events data, and music history);
- *events triggering the data collection:* set the types of events that can be collected (e.g., door openings, engine status change, location data only when the navigation feature active).

We remark that these types of parameters are only a small representative example of the whole set of parameters that the *owner* should be able to control while using the vehicle, and that the full coverage of these parameters is outside the scope of this work.

Data deletion. The *OEM* should implement a secure and definitive deletion process to make sure that the *owner* of the vehicle could effectively remove all data gathered by the vehicle at any given time. This feature would prevent deleted entries to be recovered (as presented in the previous section 5) and enables *owners* to have full control over any privacy-sensitive information gathered by the IVI system. Moreover, it is responsibility of the OEM to also delete any data available on the ExVe interface and to revoke third parties access to the deleted data.

7. Conclusions

In this paper we discuss the extraction of privacy-sensitive information from a modern In-Vehicle Infotainment system via means of forensic analysis. In particular, we focus on the $NTG5*2$ COMAND IVI system extracted from a Mercedes-Benz C-Class, presenting the steps required to physically extract the infotainment system from its lodging, accessing the data stored inside the system, and reversing the proprietary format used to encode the GPS positions stored in the system. Then, we demonstrate how these data can be enhanced by means of events logged by the system or by using OSINT sources to build a detailed report which allows to infer if the vehicle is carrying passengers, luggage, and how much time is spent in each different visited places. We develop a tool[2] to automate this process for all the 283 traces extracted from the $NTG5*2$ system, allowing us to extract sensitive information of all the drivers of our rental vehicle up to 8 months before our acquisition. Despite these final reports can support a forensic analyst in the reconstruction of the dynamic of the moments before a particular event, they can also be abused by adversaries to profile their victim, leak private data and gather knowledge that might be used later for social engineering attacks. Finally, we exploit the reverse engineered data structure to recover additional entries from the SQLite database used to store GPS entries. We compare existing general purposes solutions for recovery data from SQLite database against our approach and demonstrate that, in terms of numbers of entries recovered, our by focusing on the data structure we are able to recover more data than existing tools, recovering 13 new entries from the database that are associated with a timestamp that goes up to 2 years before our data acquisition. To the best of our knowledge, no previous work achieved similar results on a real IVI deployed in a high number of passenger vehicles. Moreover, differently from some previous work, the methodology presented in this paper does not require access to proprietary tools, and can be applied directly by security practitioners to any $NTG5*2$ head unit.

CRediT authorship contribution statement

Dario Stabili: Writing – original draft, Visualization, Validation, Resources, Methodology, Funding acquisition. **Filip Valgimigli:** Software, Investigation, Data curation. **Mirco Marchetti:** Writing – review & editing, Supervision, Conceptualization.

Responsible disclosure

The results of this work have been shared to Mercedes-Benz via their vulnerability disclosure program. We first contacted Mercedes-Benz in November 2023 and discussed our results with them. We are still in the responsible disclosure process and in close contact with Mercedes-Benz. As the time of submission of this manuscript (September 2024), Mercedes-Benz acknowledges the vulnerability discussed in our work, specifying that only vehicle equipped with the $NTG5*2$ ECU by Harman are affected with the vulnerability. The overall number of vehicles equipped with our test version of the head unit afflicted by this vulnerability is just over one million. Mercedes-Benz also specifies that head

[2] https://forms.gle/bLoP5uDnZ12iCtti6.

units newer than the $NTG5 * 2$ by Harman are not affected by the vulnerability discussed in the paper. Moreover, the tool is only meant for a limited number of audience, specifically those with intent of only legitimate research purposes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is partially supported by the projects SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, and FuSeCar - Future generation security for smart and connected cars funded by MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2022 - grant 2022W3EPEP.

Data availability

The data that has been used is confidential.

References

SQLite Consortium, February 25, 2025. SQLite library. https://sqlite.org/index.html.

ACEA, 2024a. European automobile ManufacturersÁssociation. https://www.acea.auto.

ACEA, 2024b. Access to in-vehicle data - position paper. https://www.acea.auto/files/ACEA_position_paper-Access_to_in-vehicle_data.pdf.

AUTOSAR, November 25, 2021. Log and trace protocol specification. https://www.autosar.org/fileadmin/standards/R22-11/FO/AUTOSAR_PRS_LogAndTraceProtocol.pdf.

Babun, L., Sikder, A.K., Acar, A., Uluagac, A.S., April 24-28, 2022. The truth shall set thee free: enabling practical forensic capabilities in smart environments. In: 29th Annual Network and Distributed System Security Symposium. The Internet Society. https://www.ndss-symposium.org/ndss-paper/auto-draft-192/, 2022.

BERLA, 2022. Berla.co. https://berla.co/.

Bosch, 1991. CAN specification version 2.0. http://esd.cs.ucr.edu/webres/can20.pdf.

bring2lite, August 05, 2019. bring2lite - GitHub repository. https://github.com/bring2lite/bring2lite.

California State Legislature, November 03, 2020. CCPA - California consumer privacy act. https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5.

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., 2011. Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of the 20th USENIX Conference on Security, SEC'11. USENIX Association, Berkeley, CA, USA, p. 6. http://dl.acm.org/citation.cfm?id=2028067.2028073.

Cho, K.-T., Shin, K.G., 2016. Error handling of in-vehicle networks makes them vulnerable. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16. Association for Computing Machinery, New York, NY, USA, pp. 1044–1055.

Contag, M., Li, G., Pawlowski, A., Domke, F., Levchenko, K., Holz, T., Savage, S., 2017. How they did it: an analysis of emission defeat devices in modern automobiles. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 231–250.

Daniels, Paul L., 2023. Undark - sqlite3 database data recovery tool. https://github.com/inflex/undark.

DC3 - DoD Cyber Crime Center, 2024. Sqlite dissect. https://github.com/dod-cyber-crime-center/sqlite-dissect.

De Grazia, Mari, June 22, 2015. SQLite-parser - GitHub repository. https://github.com/mdegrazia/SQLite-Deleted-Records-Parser.

Dragonas, E., Lambrinoudakis, C., Nakoutis, P., 2024. Forensic analysis of openai's chatgpt mobile application. Forensic Sci. Int. Digit. Investig. 50, 301801. https://doi.org/10.1016/j.fsidi.2024.301801. https://www.sciencedirect.com/science/article/pii/S2666281724001252.

Ebbers, S., Gense, S., Bakkouch, M., Freiling, F., Schinzel, S., 2024. Grand theft api: a forensic analysis of vehicle cloud data. In: Dfrws EU 2024 - Selected Papers from the 11th Annual Digital Forensics Research Conference Europe. Forensic Sci. Int. Digit. Investig. 48, 301691. https://doi.org/10.1016/j.fsidi.2023.301691. https://www.sciencedirect.com/science/article/pii/S266628172300210X.

European Parliament and Council, April 27, 2016. GDPR - general data protection regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

ForensicFocus, December 02, 2011. TomTom GPS encrypted triplog files. https://www.forensicfocus.com/forums/general/tomtom-gps-encrypted-triplog-files/.

Frassinelli, D., Park, S., Nürnberger, S., 2020. I know where you parked last summer: automated reverse engineering and privacy analysis of modern cars. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 1401–1415.

Gerhards, R., 2009. The syslog protocol. RFC 5424. https://doi.org/10.17487/RFC5424. https://www.rfc-editor.org/info/rfc5424, Mar. 2009.

GIS StackExchange, November 19, 2013. How does TomTom stores trackpoint information on its internal drive. https://gis.stackexchange.com/questions/74911/how-does-tomtom-store-trackpoint-information-on-its-internal-drive.

Gomez Buquerin, K.K., Corbett, C., Hof, H.-J., 2021. A generalized approach to automotive forensics. In: DFRWS 2021 EU - Selected Papers and Extended Abstracts of the Eighth Annual DFRWS Europe Conference. Forensic Sci. Int. Digit. Investig. 36, 301111. https://doi.org/10.1016/j.fsidi.2021.301111. https://www.sciencedirect.com/science/article/pii/S2666281721000056.

Groza, B., Popa, L., Murvay, P.-S., Elovici, Y., Shabtai, A., 2021. CANARY - a reactive defense mechanism for controller area networks based on Active RelaYs. In: 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, pp. 4259–4276. https://www.usenix.org/conference/usenixsecurity21/presentation/groza.

Gurbani, V.K., Burger, E., Anjali, T., Abdelnur, H., Festor, O., 2013. The common log format (CLF) for the session initiation protocol (SIP): framework and information model. RFC 6872. https://doi.org/10.17487/RFC6872. https://www.rfc-editor.org/info/rfc6872, Feb. 2013.

Hannay, P., 2008. Forensic acquisition and analysis of the TomTom one satellite navigation unit. In: Proceeding of the 6th Australian Digital Forensics Conference.

Hannay, P., 2009. Satellite navigation forensics techniques. In: Proceeding of the 7th Australian Digital Forensics Conference.

Hannay, P., 2017. A non-device specific framework for the development of forensic locational data analysis procedure for consumer grade small and embedded devices. Ph.D. thesis. School of Science, Edith Cowan University.

Hoppe, T., Kuhlmann, S., Kiltz, S., Dittmann, J., 2012. IT-forensic automotive investigations on the example of route reconstruction on automotive system and communication data. In: Ortmeier, F., Daniel, P. (Eds.), Computer Safety, Reliability, and Security. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 125–136. https://link.springer.com/chapter/10.1007/978-3-642-33678-2_11.

Jacobs, D., Choo, K.-K.R., Kechadi, M.-T., Le-Khac, N.-A., 2017. Volkswagen car entertainment system forensics. In: 2017 IEEE Trustcom/BigDataSE/ICESS, pp. 699–705.

Jeon, S., Bang, J., Byun, K., Lee, S., 2012. A recovery method of deleted record for sqlite database. Pers. Ubiquitous Comput. 16, 707–715. https://doi.org/10.1007/s00779-011-0428-7.

Juma, N., Huang, X., Tripunitara, M., 2020. Forensic analysis in access control: foundations and a case-study from practice. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS '20. Association for Computing Machinery, New York, NY, USA, pp. 1533–1550.

Kang, S., Hur, U., Kim, G., Kim, J., 2024. Forensic analysis and data decryption of tencent meeting in windows environment. Forensic Sci. Int. Digit. Investig. 51, 301818. https://doi.org/10.1016/j.fsidi.2024.301818. https://www.sciencedirect.com/science/article/pii/S2666281724001422.

Karafili, E., Cristani, M., Viganò, L., 2018. A formal approach to analyzing cyber-forensics evidence. In: Lopez, J., Zhou, J., Soriano, M. (Eds.), Computer Security. Springer International Publishing, Cham, pp. 281–301.

Kulandaivel, S., Goyal, T., Agrawal, A.K., Sekar, V., 2019. CANvas: fast and inexpensive automotive network mapping. In: 28th USENIX Security Symposium (USENIX Security 19). USENIX Association, Santa Clara, CA, pp. 389–405. https://www.usenix.org/conference/usenixsecurity19/presentation/kulandaivel.

Kwon, Y., Wang, W., Jung, J., Lee, K.H., Perdisci, R., 2021. C^2SR: cybercrime scene reconstruction for post-mortem forensic analysis. In: 28th Annual Network and Distributed System Security Symposium. NDSS 2021, Virtually, February 21–25, 2021. The Internet Society.

Lab, T.S.K., 2020. Mercedes-Benz MBUX security research report. https://keenlab.tencent.com/en/whitepapers/Mercedes_Benz_Security_Research_Report_Final.pdf.

Lacroix, J., El-Khatib, K., Akalu, R., 2016. Vehicular digital forensics: what does my vehicle know about me? In: Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications, DIVANet '16. Association for Computing Machinery, New York, NY, USA, pp. 59–66.

Le-Khac, N.-A., Roeloffs, M., Kechadi, T., 2014. Forensic analysis of the TomTom navigation application. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics X. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 267–276.

Lee, S., Lee, S., Lee, J.-R., 2017. Spare: efficient sqlite recovery using database schema patterns. KSII Trans. Int. Inf. Syst. 11, 1557–1569. https://doi.org/10.3837/tiis.2017.03.017.

LIN Consortium, 2010. LIN specification package - rev. 2.2A. https://www.cs-group.de/wp-content/uploads/2016/11/LIN_Specification_Package_2.2A.pdf.

Marchetti, M., Stabili, D., 2019. READ: reverse engineering of automotive data frames. IEEE Trans. Inf. Forensics Secur. 14 (4), 1083–1097. https://doi.org/10.1109/TIFS.2018.2870826.

Mattei, J., McLaughlin, M., Katcher, S., Votipka, D., 2022. A qualitative evaluation of reverse engineering tool usability. In: Proceedings of the 38th Annual Computer Security Applications Conference, ACSAC '22. Association for Computing Machinery, New York, NY, USA, pp. 619–631.

Mazloom, S., Rezaeirad, M., Hunter, A., McCoy, D., 2016. A security analysis of an in Vehicle Infotainment and App platform. In: Proceedings of the 10th USENIX Conference on Offensive Technologies, WOOT'16. USENIX Association, USA, pp. 232–243.

McKinnon, Mark, March 19, 2019. Autopsy plugins - GitHub repository. https://github.com/markmckinnon/Autopsy-Plugins/tree/master/Parse_SQLite_Del_Records.

Pagani, F., Balzarotti, D., 2019. Back to the whiteboard: a principled approach for the assessment and design of memory forensic techniques. In: 28th USENIX Security Symposium (USENIX Security 19). USENIX Association, Santa Clara, CA, pp. 1751–1768. https://www.usenix.org/conference/usenixsecurity19/presentation/pagani.

Pawlaszczyk, D., Hummert, C., 2021. Making the invisible visible - techniques for recovering deleted sqlite data records. Int. J. Cyber Forensics Adv. Threat Invest. 1 (1–3), 27–41. https://doi.org/10.46386/ijcfati.v1i1-3.17. https://conceptechint.net/index.php/CFATI/article/view/17.

Personal Information Protection Commission, June 2020. APPI - act on the protection of personal information. https://www.ppc.go.jp/files/pdf/APPI_english.pdf.

Pesé, M.D., Stacer, T., Campos, C.A., Newberry, E., Chen, D., Shin, K.G., 2019. LibreCAN: automated CAN message translator. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19. Association for Computing Machinery, New York, NY, USA, pp. 2283–2300.

Pollicino, F., Stabili, D., Marchetti, M., 2023. Performance comparison of timing-based anomaly detectors for controller area network: a reproducible study. ACM Trans. Cyber-Phys. Syst. https://doi.org/10.1145/3604913.

Radu, A.-I., Garcia, F.D., 2016. LeiA: a lightweight authentication protocol for CAN. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (Eds.), Computer Security – ESORICS 2016. Springer International Publishing, Cham, pp. 283–300.

Read, H.O., Xynos, K., Sutherland, I., Bovee, M., Tamburro, C., 2024. Nintendo 3ds forensics: a secondhand case study. In: DFRWS APAC 2024 - Selected Papers from the 4th Annual Digital Forensics Research Conference APAC. Forensic Sci. Int. Digit. Investig. 50, 301815. https://doi.org/10.1016/j.fsidi.2024.301815. https://www.sciencedirect.com/science/article/pii/S2666281724001392.

Salem, Y., Hamarsheh, M.M., 2024. Forensically analyzing iot smart camera using maoidff-iot framework. Forensic Sci. Int. Digit. Investig. 51, 301829. https://doi.org/10.1016/j.fsidi.2024.301829. https://www.sciencedirect.com/science/article/pii/S2666281724001537.

Soni, N., Kaur, M., Aziz, K., 2024. Decoding digital interactions: an extensive study of teamviewer's forensic artifacts across windows and Android platforms. Forensic Sci. Int. Digit. Investig. 51, 301838. https://doi.org/10.1016/j.fsidi.2024.301838. https://www.sciencedirect.com/science/article/pii/S2666281724001653.

SQLabs, 2023. SQLite doctor. https://sqlabs.com/sqlitedoctor.

SQLite, June 18, 2004. Sqlite database file format. https://www.sqlite.org/fileformat.html.

SQLite - Database File Format, June 18, 2004. SQLite - database file format. https://www.sqlite.org/fileformat.html.

Stabili, D., Marchetti, M., 2019. Detection of missing CAN messages through inter-arrival time analysis. In: 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), pp. 1–7.

Stabili, D., Bocchi, T., Valgimigli, F., Marchetti, M., 2024. Finding (and exploiting) vulnerabilities on ip cameras: the tenda cp3 case study. In: Minematsu, K., Mimura, M. (Eds.), Advances in Information and Computer Security. Springer Nature Singapore, Singapore, pp. 195–210.

SysTools, 2014. SysTools SQLite database recovery. https://www.systoolsgroup.com/sqlite-database-recovery.html.

TomTom Forum, October 03, 2010. TripLog - where did I go? https://www.tomtomforums.com/threads/triplog-where-did-i-go.23463/.

TomTom Forum, November 28, 2007. TripLoging enabled. https://www.tomtomforums.com/threads/does-my-one-have-trip-logging.4721/.

Whelan, C.J., Sammons, J., McManus, B., Fenger, T.W., 2018. Retrieval of infotainment system artifacts from vehicles using iVe. J. Appl. Digit. Evid. 1 (1). https://mds.marshall.edu/jade/vol1/iss1/2.

Wu, B., Xu, M., Zhang, H., Xu, J., Ren, Y., Zheng, N., 2013. A recovery approach for sqlite history recorders from yaffs2. In: Mustofa, K., Neuhold, E.J., Tjoa, A.M., Weippl, E., You, I. (Eds.), Information and Communication Technology. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 295–299.

Zeng, K.C., Liu, S., Shu, Y., Wang, D., Li, H., Dou, Y., Wang, G., Yang, Y., 2018. All Your GPS Are Belong to Us: Towards Stealthy Manipulation of Road Navigation Systems, in: 27th USENIX Security Symposium (USENIX Security 18). USENIX Association, Baltimore, MD, pp. 1527–1544. https://www.usenix.org/conference/usenixsecurity18/presentation/zeng.