Data leakage prevention model for vehicular networks

Maryam Najafi[1], Marc Lemercier[1], and Lyes Khoukhi[2]

[1]Computer science and digital society (LIST3N) Laboratory, University of Technology of Troyes (UTT), Troyes, France
[2]Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
*M*aryam.najafi@utt.fr, marc.lemercier@utt.fr, lyes.khoukhi@ensicaen.fr

***Abstract*—The vehicle ad-hoc network (VANET) is a promising technology that enables numerous vehicular network applications to improve road safety, navigation, and many other purposes. Android automotive supports many applications for vehicles. Each application has a list of accesses, called permissions, required for specific interfaces or sensitive data. However, some applications request permissions unrelated to functionalities or unnecessary permissions. Furthermore, they improperly collect user data. Given the privacy risk associated with applications, it is necessary to study the permissions requested by the application before installation. A permission system is a solution to deal with abusive applications. However, such a system suffers from limitations as users may ignore it during the installation phase due to the complexity of understanding the permissions. This article proposes a graph-based model to determine abusive applications by automatically analyzing the requested permissions. This aims to build a confidence indicator to choose the applications with more respect for privacy. This model would inform the user about the possibility of data leakage risks by assigning a privacy score.**

***Index Terms*—VANET, google auto, applications, privacy , data leakage**

I. Introduction

With the development of intelligent transportation systems, Vehicle Ad-hoc Network (VANET) has attracted a lot of attention. VANET is a promising research site that can offer many useful applications in In-Vehicle Infotainment (IVI) platforms.

IVI platforms have become increasingly popular. To accommodate this trend, Google has moved to standardization among proprietary IVI operating systems. Google presents Android Automotive Operating System (AAOS), which runs natively on the IVI platform. Android Automotive is a full operating system that can power the entire experience when driving [1]. Android Automotive is in charge of everything during driving. AAOS will access the In-Vehicle Network (IVN) and can read and collect vehicular sensors and event data. Moreover, Android Automotive shares data with selected third-party applications. In addition, Google presented Android Auto which is designed to display several applications on the car screen. However, Android Auto is limited to working with third-party applications and does not have access to data generated inside the vehicle. While Android Auto always requires an Android phone to run the applications and a connection to the IVI (over USB, WiFi, or Bluetooth AVRCP), Android Automotive runs directly on the head unit without needing a cable or Bluetooth connection or anything else.

Android auto and Android Automotive support many useful applications. Applications with similar functionality fall into the same category. The supported applications are classified into five different types as follows:

1) **Media applications:** This type of application allows users to browse and play various audio content in the vehicle, such as music, radio, and audiobooks.
2) **Messaging applications:** This type of application allows users to receive and send messages easily without the risk of interrupting attention. For example, messaging applications are able to convert any written text into spoken words and vice versa, converting human speech into text.
3) **Navigation, parking, and charging applications:** This type of application helps users to drive more comfortably. For example, they help the driver find his way, find a place to park, or find the nearest charging station.
4) **Point of Interest (POI) applications:** This type of application helps users to find and discover their favorite places with the intent to navigate. These applications are not limited to finding a parking place, fuel station or charging station, although these are included.
5) **Video applications:** This type of application allows users to watch streaming videos while the vehicle is parked for security reasons.

This paper is structured as follows. Section II presents the related works. Section III describes the detail of our proposed model in this article. In Section IV, the paper highlights the results of the proposed model. Section V concludes the article and also presents future works.

II. Related works

Android automotive applications have a list of accesses, called permissions, required for specific interfaces or sensitive data. For example, ”access network state”, this permission allows applications to access information about networks. Applications also record a certain amount of data, such as speed with the permission called ”car speed”. Likewise, the vehicle’s state, whether parked, idling, or moving with permission, is called ”car driving state”. Based on the protection

levels, permissions are divided into four groups on the official Android site as follows:

1) **Normal:** Default low-risk permission has a minimal risk of privacy for other applications, users, or the system. This type of permission is automatically granted to the requesting applications during installation without the user's express consent, although, the user has the option to check these permissions before installation. For example, the following permissions have a normal protection level: car info, full network access and car exterior environment, .
2) **Dangerous:** High-risk permission has a potential risk to users' privacy. This type of permission gives access to the user's private data or control over the device, which may adversely affect the user's privacy. Therefore, these permissions need the express consent of the user before installation. For example, precise location (GPS and network-based), car energy and car speed.
3) **Signature:** This type of permission is granted without notifying the user or asking for the user's consent on one condition. The condition is if the requesting application is signed and matched with the same certificate as the application that declared the permission. For example, the following permissions have a signature protection level: bind projection service, bind VMS client, and bind car input service.
4) **Signature or system:** Granting this permission is complex and requires two conditions. First, the requesting application must signed and matched with the same certificate as the application that declared the permission. Second, the program must be located in a special folder on the Android system image. This permission is recommended for a specific situation. For example when vendors need to explicitly share specific attributes because they are created with each other. There is a list of permissions with the signature or system protection level as follows: modify system setting, car tires, car identification and update component usage statistics.

A permission system is a security solution against abusive applications [2]. Each application must define a list of permissions required for specific interfaces or sensitive data. Before installing any application, the user must grant access to the permissions. The permissions list provides information about an app's behavior and may be appropriate for automatic app analysis. It could be a good solution to deal with privacy risk, but unfortunately, it has not been successful because users ignore it during the installation phase due to the complexity of understanding the permissions [3]. In fact, malicious third-party entities may cause financial and operational damage to the vehicle and compromise the driver's privacy and safety. This is because improperly, they can remotely collect user data and request unrelated permissions to functionalities or unnecessary permissions. In addition, each new version of the application contains additional permissions related to new features [4]. Information on required permissions is integrated into each application and is always available for verification by users. Their identification of key permissions for various features as well as expected permission requests can help us detect the program's normal/abnormal behavior and provide a danger indicator for users [5]. This list of permissions is supposed to warn users about abusive applications but has proven to be ineffective.

The data leakage risk is considered one of the most significant challenges that can lead to serious network impacts. Hence, privacy is of primary concern since attackers can alter life-critical information [6]. Imagine a third-party application that requests engine speed permission called "sensor type RPM" (i.e., represents engine RPM of the car.) and gear position permission called "current gear". Both of these permissions are in the normal permission level group: in other words, they are granted automatically without the user's explicit consent. It is not difficult to calculate the car's speed with this information, because car speed, engine speed, and gear position share a physical relationship. However, on the official Android site the "car speed" permission has a dangerous protection level (this sensor represents vehicle speed in m/s.). Therefore, a third-party application can easily gain access to a sensor with dangerous permission. In [7, 8] authors proved that GPS location can be extracted from car speed. It is worth noting that permissions related to the location, such as precise location (GPS and network-based) and approximate location (network-based), have a dangerous protection level. Information such as the speed or location of the vehicle is sensitive information and if it is in the wrong hands, the privacy and safety of the driver will be compromised. For example GPS location allows the driver to be tracked down, a severe data leakage that must be prevented at all costs.

Recently, some studies have been done on Android Auto, mostly focusing on analyzing potential security threats. In [9] the authors proposed a static analyzer for the Android Auto infotainment system. The architecture of their system is based on analyzing the Android Manifest. They extract the Java bytecode by reverse engineering through dex2jar. Also, they extract the jar files from the apk file by reverse engineering through apktool. So, they use the manifest file to determine the entry points for parsing the Java bytecode of the apps. The authors have checked all the infotainment apps available on the Google Play Store. Based on this research, the authors show that almost 80% of the applications are potentially vulnerable and expose some warnings, out of which 25% applications pose security threats and serious risk related to the execution of JavaScript. Similarly, in [10] authors analyze the Auto infotainment and On-Board Diagnostics II applications. They found that 60% of these applications are vulnerable and have access to the cache directory.

Sensors have always played an important role in the new generation of modern vehicles. Also, there are numerous sensors in the design of modern machines. These sensors produce huge amounts of data about vehicles, drivers, and the environment. The data generated is used by automotive applications. In [11], the authors proposed a Sensing as a

Service (S2aaS) model to collect and transmit data from vehicle sensors. They proposed an optimization mechanism for the network. This optimization is based on sensors data cache and distributed decision-making systems.

In [12], the authors concentrate on malicious third-party apps, driver disturbance, availability, and privacy attacks. They analyze the potential risk to the privacy, safety, and security of Android automotive applications. Their analyses have shown that vehicular applications are not safe and can impact road safety and, to some extent, user privacy. They presented a static analysis tool to detect dangerous use of vehicle-specific APIs, which is called "AutoTame". In [1], the authors analyze the architecture of the Android Automotive platform. They found that the current permission model is not sufficiently secure and needs improvement. The authors have shown its vulnerabilities through three theoretical attacks (i.e., privacy, Financial/Operational, and Safety) and presented possible mitigations.

Regarding the privacy risk associated with applications, it is necessary to study the permissions required by the application before installation. In [13] the authors proposed a generic model for measuring the data leakage risk due to the mobile application. However, there is no research about studying the permissions requested by applications in vehicular network.

We propose a new measure of data leakage risk for the VANET applications inspired by a model presented by Sokolova et al. [14]. Any application presents an initial risk calculated concerning the data access authorizations required and the legitimacy of this request compared to other applications with similar functionalities. In other words, we propose an indicator to measure the data leakage risk of applications. This model would inform the user about the possibility of data leakage risks by assigning a privacy score and risk warning threshold.

III. Measurement of the data leakage risk with a permission pattern graph

In this section, we present a model to build a permission graph for each application as well as a permission pattern graph for the categories of applications in VANET. We suppose applications grouped in the same category provide similar functions. Therefore they demand similar collections of permissions. To build this pattern for a category, we analyse co-occurring pairs of permissions. Here we present an analysis of legal permission applications by category. Based on legitimate permission requests, we determine the normal behavior for the applications of a category.

A. Build a permission graph for an application

For each application App, a graph denoted $G_{App}(Ps\ \ App, L\ \ App)$ is created where the set of $Ps\ \ App$ nodes represents the permissions required by an application, and the set of $L\ \ App$ links represents two permissions used together in this application, in other words these two permissions are linked together. We produce a graph for the Telegram application in the category of

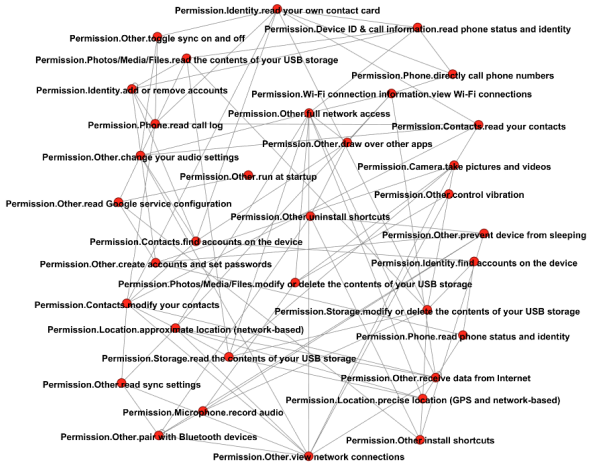

Fig. 1. Graph of the permission requested by Telegram

messaging applications. As shown in Figure 1 each node present one permission (e.g., Permission.Other.receive data from Internet, Permission.Contacts.modify your contacts, Permission.Phone.read phone status and identity). We show all the nodes in red color because they are requested by Telegram before installation. There are 34 permissions. Also the edges present the links (e.g., permission.Other.view network connection and Permission.Location.approximate location (network-based), Permission.Contacts.read your contacts and Permission.Contacts.modify your contacts). As shown in this figure we have a connected graph. However, in order to better display the graph, we displayed important links.

$$App = G_{App}(Ps\ App, L\ App) \tag{1}$$

B. Build a permission pattern graph for a category of applications

For each category cat_X, a graph denoted $G_{cat_X}(P\ \ cat, L\ \ cat)$ is created where the set of $P\ \ cat$ nodes represents the permissions, and the set of links $L\ cat$ represent two permissions used together in the same category; in other words, these two permissions are linked together. Where $x \in \{$media apps, messaging apps, navigation,parking and charging apps, point of interest apps, video apps$\}$.

To avoid over represented pairs, such as Permission.Other.receive data from internet, Permission.Other.full network access we calculate the importance of a permission pair in the category based on its overall usage. We apply the W score for each permission pair observed in each category to do this. The W score is also used as the weight of the links in the graph. These pairs show the most representative legal permits in each category. We will calculate W score as below:

$$W_{PP} = \frac{F_{cat}^{PP} - \mu_{PP}}{\sigma_{PP}} \tag{2}$$

Where:

- PP is a permission pair $PP \in P\ cat^2$
- F_{cat}^{PP} is the frequency of PP in the cat.
- μ_{PP} is the mean of the pair PP across all cat
- σ_{PP} is the standard deviation of the pair PP across all cat

The value of W_{PP} can be negative, positive, or equal to one.

$$W_{PP} = \begin{cases} > 1 & if \quad F_{cat}^{PP} < \mu_{PP} \\ = 1 & if \quad F_{cat}^{PP} - \mu_{PP} = \sigma_{PP} \\ < 1 & if \quad F_{cat}^{PP} - \mu_{PP} > \sigma_{PP} \end{cases} \tag{3}$$

To obtain the final graph of each category, we filtered the pairs of permissions to keep only the pairs whose μ_{PP} is greater than 1. The non-connected nodes (permissions) of the resulting graph are also omitted. So for each category of applications, we present a category pattern as follows:

$$Pat_{cat} = G_{cat_X}(P\ cat, L\ cat) \tag{4}$$

We calculate the similarity of permissions denoted Sim_P for each application denoted App with the pattern of its category as follows:

$$Sim\_P_{App,Pat}(App, Pat_{cat}) = |Ps\ App \cap P\ cat| \tag{5}$$

Where:

- $Ps\ App$ is a set of permissions required for the installation of this application.
- $P\ cat$ is a set of permissions belonging to a pattern in the category cat.
- $Sim\_P_{App,Pat}$ is a percentage of number of common permissions between an application and a category pattern. This score is defined in the equation 5 above.

An application that follows the pattern perfectly obtains a similarity equal to 1; otherwise, an application that deviates completely from the pattern obtains a similarity equal to 0.

C. Build a confidence indicator

We build a confidence indicator for each application to select the applications with more respect for privacy. Therefore, with this indicator, we can determine the applications that have normal or abnormal behavior after installation.

We calculate the risk of data leakage based on the requested permission by application. We consider a weight for each protection level of permission (e.g., Normal= 1, Dangerous= 2, Signature= 3, and Signature and system= 4). We calculate Data Leakage Risk (DLR) score as follows:

$$DLR_{score} = \sum_{x=1}^{4} Numx * Lx \tag{6}$$

Where:

- DLR_{score} is the data leakage risk score of the application.
- $Numx$ is a number of permissions requested by application except the standard permissions.
- Lx is a weight of each protection level, (i.e., the normal level has the lowest weight equal to 1 and signature or system level has the highest weight equal to 4. Also we consider Normal protection level as L1, Dangerous protection level as L2, Signature protection level as L3, and Signature or system protection level as L4.

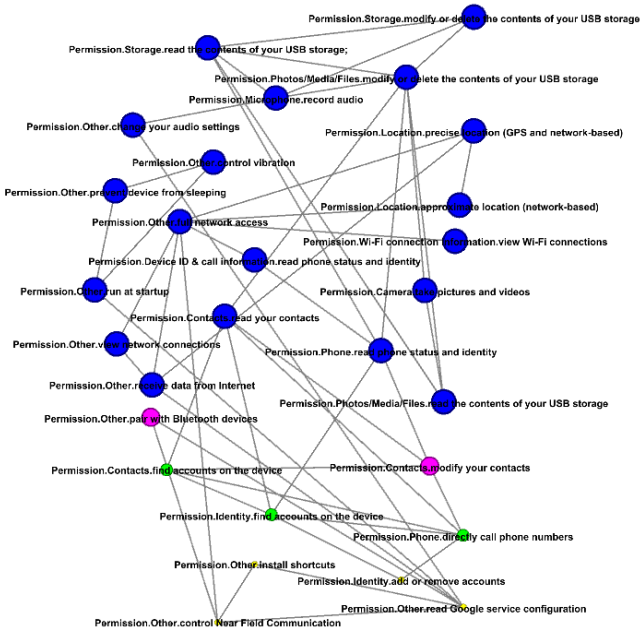

Fig. 2. Permissions' graphs obtained for the Messaging application category

IV. Evaluation

We only present the result related to one of the categories. We chose the messaging applications category because, today, everyone uses at least one of these messaging applications. There are nine applications available in this category: Hangouts, Nextplus, ICQ Video Calls Chat Rooms, Kik, TextPlus, WeChat, Telegram, WhatsApp, and Messenger.

The pattern obtained is highlighted in Figure 2. In this graph, the color and size of the nodes are directly related to the percentage usage of this permission by applications. We used four colors (i.e., blue, magenta, green, and yellow). The blue or the biggest nodes indicate the permissions requested by all applications in the messaging category. Smaller or magenta color nodes represent the permissions requested by 88,88 percent of applications. The green nodes, smaller than magenta, represent the permissions requested by 77,77 percent of applications. The smallest nodes, shown at the bottom of the figure in yellow, show the permissions requested by 66,6 percent of applications. All of these permissions in this obtained graph are standard permissions for applications in this category.

We list some of the permissions requested by applications in the messaging applications category in Table 1. P stands for permission in this table. The % Usage column shows the percentage of requests for this permission by the applications. For example, "P.other.modify system settings" is requested by

TABLE I
EXAMPLES OF PERMISSIONS AND PERCENTAGE USAGE IN THE MESSAGING APPLICATIONS CATEGORY

Permission	% Usage
P.Other.control flashlight	11,11
P.Other.Google Play license check	11,11
P.Other.modify system settings	11,11
P.Other.read sync statistics	11,11
P.SMS.receive text messages (MMS)	22,22
P.Other.measure app storage space	22,22
P.Other.use accounts on the device	22,22
P.Phone.read call log	33,33
P.SMS.send SMS messages	33,33
P.SMS.receive text messages (SMS)	33,33
P.Other.uninstall shortcuts	33,33
P.Other.access Bluetooth settings	44,44
P.Other.change network connectivity	44,44
P.Other.disable your screen lock	44,44
P.Identity.read your own contact card	55,55
P.Other.draw over other apps	55,55
P.Other.read sync settings	55,55
P.Other.send sticky broadcast	55,55
P.Other.toggle sync on and off	55,55
P.Identity.add or remove accounts	66,66
P.Other.install shortcuts	66,66
P.Phone.directly call phone numbers	77,77
P.Contacts.modify your contacts	88,88
P.Other.pair with Bluetooth devices	88,88
P.Camera.take pictures and videos	100
P.Contacts.read your contacts	100
P.Microphone.record audio	100
P.Phone.read phone status and identity	100
P.Other.change your audio settings	100
P.Other.control vibration	100
P.Other.full network access	100
P.Other.prevent device from sleeping	100
P.Other.receive data from Internet	100
P.Other.run at startup	100
P.Other.view network connections	100

11.11 % of applications. Protection level of this permission is signature or system. Granting this permission means access to a camera, microphone, Google Drive to store items there, a Google Account to store detailed reports, and more. It can also change Android options such as your screen timeout duration. It is obvious that this permission has the potential to be abused.

Our evaluation results show that the Kik application (messaging and chat app) has the closest similarity to the obtained permissions' graph. Furthermore, this application did not request any more permissions than the standard permissions shown in Figure 2. Therefore, in this case, the data leakage risk score is equal to zero because the number of other permissions requested by the application at each level is equal to zero. In contrast, the Wechat application requested 21 more permissions than standard permissions in this category. These permissions contain 12 permissions at a normal protection level (i.e., $Num1$), 14 at the dangerous protection level (i.e., $Num2$) and one permission from the signature or system level (i.e., $Num4$). We calculate the data leakage score of the Wechat as follows:

$$DLR_{score} = (12 * 1) + (14 * 2) + (0 * 3) + (1 * 4)$$

So the DLR_{score} is equal to 44. This score shows that there is high probability of data leakage risk from the installation of this application. This application requests permissions such as Permission.Device app history.retrieve running apps, Permission.Other.modify system settings, Permission.Other.download files without notification, Permission.Other.connect and disconnect from Wi-Fi, etc.,

V. CONCLUSION

The study of the permissions requested by the applications for each category allowed us to determine the normal or abnormal behavior expected from an application. This article proposes a graph-based model that can automatically analyze the permissions requested by applications in the Android Automotive. The proposed model deals with data leaks and privacy flaws that can occur with the installation of an application. Furthermore, our model assigns a privacy score to each application. Thus this model can identify possible deviations even before using the application, and helps users choose the applications which have greater respect for privacy. In the future, we will propose a model for the precise classification of applications in Android Automotive.

ACKNOWLEDGMENT

The authors would like to thank FEDER, Region GRAND-EST, and the MSER for the financial support of this research.

REFERENCES

[1] M. Pese, K. Shin, J. Bruner, and A. Chu, "Security analysis of android automotive," pp. 2020–01–1295.

[2] I. M. Almomani and A. A. Khayer, "A comprehensive analysis of the android permissions system," vol. 8, pp. 216671–216688. Conference Name: IEEE Access.

[3] J. Palmerino, "Improving android permissions models for increased user awareness and security," in *2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pp. 41–42.

[4] P. K. Tiwari, S. R. Basireddy, and V. T, "Identification of possibly intemperate permission demands in android apps," in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, vol. 2, pp. 101–106.

[5] O. E. Kural, D. Şahin, S. Akleylek, and E. Kılıç, "Permission weighting approaches in permission based android malware detection," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pp. 134–139.

[6] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named data networking in vehicular ad hoc networks: State-of-the-art and challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 320–351, 2020.

[7] L. Zhou, Q. Chen, Z. Luo, H. Zhu, and C. Chen, "Speed-based location tracking in usage-based automotive insurance," in *2017 IEEE 37th International Conference*

on Distributed Computing Systems (ICDCS), pp. 2252–2257. ISSN: 1063-6927.

[8] R. Dewri, P. Annadata, W. Eltarjaman, and R. Thurimella, "Inferring trip destinations from driving habits data," pp. 267–272.

[9] A. K. Mandal, A. Cortesi, P. Ferrara, F. Panarotto, and F. Spoto, "Vulnerability analysis of android auto infotainment apps," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, CF '18, pp. 183–190, Association for Computing Machinery.

[10] A. K. Mandal, F. Panarotto, A. Cortesi, P. Ferrara, and F. Spoto, "Static analysis of android auto infotainment and ODB-II apps," p. 33.

[11] O. Sadio, I. Ngom, and C. Lishou, "A novel sensing as a service model based on SSN ontology and android automotive," vol. 19, no. 16, pp. 7015–7026. Conference Name: IEEE Sensors Journal.

[12] B. Eriksson, J. Groth, and A. Sabelfeld, "On the road with third-party apps: Security analysis of an in-vehicle app platform:," in *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems*, pp. 64–75, SCITEPRESS - Science and Technology Publications.

[13] K. Sokolova, C. Perez, and M. Lemercier, "Android application classification and anomaly detection with graph-based permission patterns," *Decision Support Systems*, vol. 93, pp. 62–76, Jan. 2017.

[14] K. Sokolova, C. Perez, and M. Lemercier, "Modèle d'analyse pour la prévention contre la fuite de données mobiles," p. 13, 2016.