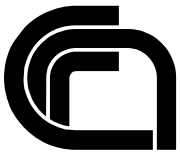

Consiglio Nazionale delle Ricerche

KOFFEE - Kia OFFensivE Exploit

G. Costantino, I. Matteucci

IIT TR-20/2020

Technical Report

Novembre 2020

Istituto di Informatica e Telematica

KOFFEE - Kia OFFensivE Exploit

Dr. Gianpiero Costantino
e-mail: gianpiero.costantino@iit.cnr.it
website: http://webhost.services.iit.cnr.it/gianpiero.costantino/
Dr. Ilaria Matteucci
e-mail: ilaria.matteucci@iit.cnr.it
website: http://webhost.services.iit.cnr.it/staff/ilaria.matteucci/

1 Introduction

A recent study [7] claims that modern In-Vehicles Infotainment (IVI) systems mounts Linux or Android operating system (OS). Even though Linux provides several advantages, Android OS is going to impose its supremacy also in the automotive market [2]. This is mainly caused by the advantages that such OS provides in terms of features in the connected-car scenario. Several OEMs already mounts on their cars IVI with Android OS and others are going to do it soon, e.g., General Motors in 2021 [6].

In this paramount, years ago we have started our security research activity on possible vulnerabilities that IVI, mounting Android OS, may expose. Our initial studies were on after-market IVIs based on Android OS and we found important vulnerabilities [3],[4] on the devices that may allow for instance, an attacker to gain remote root privileges to the IVI.

As next step, we moved our activity on a KIA Cee'd car, which we bought in the summer of 2019 and we started reverse engineering starting from its HU based on Android. Our KIA Cee'd is not connected to the Internet by default and does not have a telematic unit. However, it can be connected to the Internet through a smartphone, as hotspot mode, or 3G,4G and 5G dongle that generates a Wifi network in which the head unit is connected.

In this report, we describe our exploit, named KOFFEE, perpetrated to a KIA Cee'd. This is part of our research activity on offensive cybersecurity in the automotive domain. Therefore, we decided to not detail all aspects of our work in this report, instead of a full disclosure which would be considered as irresponsible to vehicle users. The full report will be released at a proper time in the year 2021 if things will go as planned.

2 Vehicle's anatomy

Our KIA Cee'd has an internal network separation that aims to divide untrusted zones, such as the multimedia one, named M-CAN, from the trusted one, such as P-CAN and others.

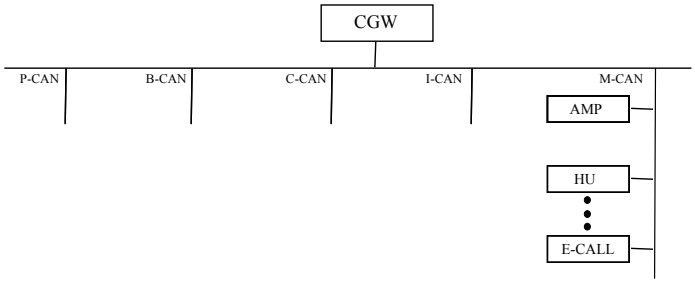

Figure 1: In-Vehicle Network

The partitions are connected through a Central Gateway (CGW) that forwards CAN frames from a partition to another. Not all frames are allowed to cross partition and this is controlled by the CGW.

3 Head Unit

The Head Unit (HU) sold with the KIA Cee'd is last available platform of Kia vehicles. This platform is the fifth generation, i.e, Gen 5.0, referred with the name *iAVN*. It is designed for more than 40 Countries, has Apple Carplay and Android support as well as Tom Tom services for the navigation part. The hardware specification are the following:

- *Chipset*: Telechips TCC8931
- *CPU* Dual ARM Cortex A9 @ 1.2GHz to 1.5GHz with 64KB RAM,
- *MCU*: ARM Cortex M3 with 16KB I-RAM & 16KB D-RAM
- *GPU*: ARM Mali-400MP2
- *Display*: 8" touch-screen

Android is the operating system and the version available in our Head Unit (HU) is Android 4.2.2.

3.1 Third-party applications

The head unit as is does not allow users to install third-party applications. In fact, users are limited to use only those apps already pre-installed in the system and the possibility to access to other Android's menu is blocked by default. However, in the Internet there are available several guides that allow users to access the full Android operating system from a hidden menu, called *Engineering Menu* and referred as "eng menu" throughout this paper. Access to the eng menu may change from vehicle to vehicle model and it also changes from HU software version.

Once a user got access to the eng menu and unlocked the access to the fully operating system, she/he is able to install third-party applications also from unknown sources. This represents the gateway of our attack for Head Unit software versions listed in Section 4.4. For Head Units with software versions 20.06 and beyond, the installation of 3rd party applications after getting access to engineering mode was no longer possible, making the Kia Offensive Exploit described in Section 4 no longer effective.

3.2 Apps reverse engineering

By default the HU comes with a set of apps already pre-installed that allow users to use the iAVN. For example, there is the navigation app and the radio app to listen to the available radios. In addition, there are also other apps needed for the HU configuration or just to have information about the software version as well as the possibility to upgrade the software with a new version.

By default, Third-party applications cannot be installed on the system. Using the eng menu we opened a way to do it. Moreover, by accessing the file-system we were able to obtain the apk files of the original apps installed in the HU. Thus, we started our reverse engineering phase exploring the "source code". It is important to highlight that all apps were in the format *apk+odex* [8] that made the reverse engineering phase quite complicated.

Once, we were able to reverse all apps, we started digging into the code trying to understand its main functionalities and any kind of detail related to CAN bus access (if any). We discovered that apps communicate using the Inter-Process Communications (IPC) [1] method and apps can receive and send CAN frames, and/or control some functionalities of the head unit.

3.3 File System

The file-system structure resembles that one of common Android file system. By default, the user does not have privileged access to the system and some files and directories cannot be accessed nor modified. However, there are some accessible directories that provides executable files or scripts that can be run.

3.4 Access to the Can bus

As we described in §2 the Head Unit is a node that belongs to the M-CAN. It can send CAN frames into the M-bus as well as to receive frames from this bus and the other ones of the in-vehicle network. Frame injection is executed using a specific hardware component, called *MICOM*. It can be trigger for the operating system as well as apps running on it through a socket that is available at the position "/dev/socket/micomd".

At the app level, CAN frame are sent by a JAVA object that aims to send frames once the send method is invoked. This method mainly takes two parameters, which are the "ID" and the payload formed by an array of up to 8 bytes. However, the ID and the payload do not correspond to the same fields of the CAN bus 2.0 protocol, but they refers to an internal HU structure that, controlled by the MICOM firmware, converts those fields into a so called MICOM-signal or to a CAN frame. This choice is regulated by a sort of internal DBC available in the HU, which depending on the ID, generates a MICOM signal or the corresponding CAN frame. From our reverse engineering studies, we observed that all MICOM-signals are addressed to the HU components, e.g., active reverse parking camera or increase, decrease the audio volume. Instead the CAN frames are mostly for the external nodes.

4 KOFFEE

Kia OFFensivE Exploit (KOFFEE) is the exploit we wrote to take control of the KIA Cee'd Head Unit that we used for our research activity.

4.1 Attack chain

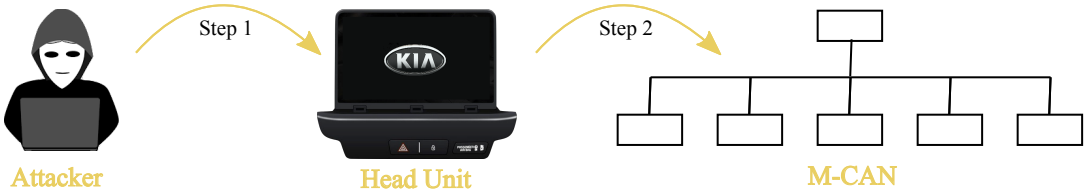

Figure 2: KOFFEE work-flow

Step 1. Exploiting the possibility to install third-party application, we wrote an app that, once installed in the system and run, for instance through a social engineering method, opens both the licit app plus other malwares, which we wrote and inserted in the main app. These malwares allow the remote connection to the HU plus other spy-actions activated on the HU.

Step 2. The remote control allows us to access and navigate the HU file system. Starting from this and from all details got during our reverse engineering phase, we are able to inject CAN frame into the M-bus as well as to trigger some functionalities, such as, increasing the radio volume or switching it off. All these operations are performed working on the *Micom* of the HU.

4.2 Semi-controlled CAN frames

With the current version of KOFFEE, we are also able to send semi-controlled CAN frames. As far as we understood, the firmware of the HU filters out some frame IDs that are not allowed to be sent. On the contrary, we have basically full control of the payload for those working CAN IDs. Although the restriction imposed by the firmware, we are able to forger frames and trigger other nodes in the M-bus.

4.3 Weaponising KOFFEE

KOFFEE is also available in Metasploit [5] as module[1] that allows us to exploit the discovered vulnerabilities. Figure 3 shows the main functionalities of our module. The displayed ones are only a subset of a bigger number of actions that the attacker can exploit through KOFFEE.

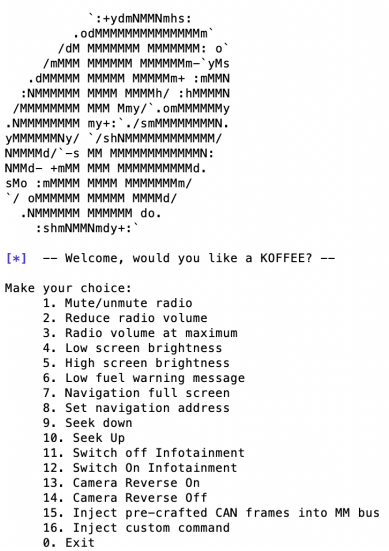

Figure 3: KOFFEE for Metasploit

4.4 Vulnerability Assessment

We found the following vulnerabilities that KOFFEE exploits:

ID	Vulnerability	Node	Function	Channel	Vulnerability importance
1	Code Execution	HU	Controlling HU functionalities	Wireless	High
2	HU library code execution	HU	Developing APP to work with IPC	Wireless	High
3	Semi-Controlled CAN frame injection	HU	M-CAN nodes triggered	Wireless	Medium
4	DBC to generate and interpret CAN frames and MICOM signals	HU	Ease the reverse eng phase of signals and CAN frames	Local	Medium

Table 1: List of discovered vulnerabilities

We tested KOFFEE on HUs with the following software versions on a Kia Cee'd with Gen 5.0 Head Unit:

- 180703
- 181019
- 191219
- 200401

[1] The module is not public available at the moment.

5 Disclosure Process

This is an ethical hacking research activity that we have done on the KIA Cee'd. We decided to follow the "Responsible Disclosure" practice in which contacted the manufacturer, in this case KIA Motors before publishing our fully research activity.

References

[1] Android IPC: Android Interface Definition Language (AIDL). https://developer.android.com/guide/components/aidl (2020)

[2] Christoph Hammerschmidt: Study: Android challenges automotive OS market. https://www.eenewsautomotive.com/news/study-android-challenges-automotive-os-market (2019)

[3] Costantino, G., Marra, A.L., Martinelli, F., Matteucci, I.: CANDY: A social engineering attack to leak information from infotainment system. In: 87th IEEE Vehicular Technology Conference, VTC Spring 2018, Porto, Portugal, June 3-6, 2018. pp. 1–5. IEEE (2018). https://doi.org/10.1109/VTCSpring.2018.8417879, https://doi.org/10.1109/VTCSpring.2018.8417879

[4] Costantino, G., Matteucci, I.: CANDY CREAM - hacking infotainment android systems to command instrument cluster via can data frame. In: Qiu, M. (ed.) 2019 IEEE International Conference on Computational Science and Engineering, CSE 2019, and IEEE International Conference on Embedded and Ubiquitous Computing, EUC 2019, New York, NY, USA, August 1-3, 2019. pp. 476–481. IEEE (2019). https://doi.org/10.1109/CSE/EUC.2019.00094, https://doi.org/10.1109/CSE/EUC.2019.00094

[5] RAPID4 Metasploit: Metasploit. https://www.metasploit.com (2020)

[6] Sean O'Kane: GM will use Google's embedded Android Automotive OS in cars starting in 2021. https://www.theverge.com/2019/9/5/20851021/general-motors-android-auto-google-infotainment (2019)

[7] Strategy Analytics Press Releases: Linux, Android poised to dominate automotive infotainment systems. https://www.telecomtv.com/content/device-software-apps/linux-android-poised-to-dominate-automotive-infotainment-systems-34987/ (2019)

[8] XDA Developers: The differences between Odex and Deodex Files. https://forum.xda-developers.com/showthread.php?t=2336411 (2013)