OT project report

Remote attacks on KIA Soul car

Svetlana Burikova
Innopolis University
s.burikova@innopolis.ru

Sergey Grebennikov
Innopolis University
s.grebennikov@innopolis.ru

Saif Saad
Innopolis University
s.s.mohammed@innopolis.ru

Iuliia Sharafitdinova
Innopolis University
i.sharafitdinova@innopolis.ru

May 13, 2018

Abstract

A wireless connection to a vehicle can improve existing vehicle management services and create many new opportunities for remote diagnostics. But among other things, these very attractive features can become a problem with serious security problems. The internal network of the vehicle control system must be protected from unauthorized access as well as provide a secure channel between vehicle components and external devices. During this project, we will conduct a brief analysis of the safety of remote interaction with the KIA Soul car based on the best practices from the SANS Institute and research articles from the Infosec conferences.

I. Introduction

Modern cars have up to 80 electronic control units (ECUs) that control electric systems in the car: the engine, the braking system, the steering wheel, multimedia, climate, etc. Embedded software in the ECU continues to increase the number of code lines and complexity [1], thereby reducing the stability and reliability of the system. These units are connected via serial buses and communicate with a standard protocol called the Controller Area Network (CAN).

Current technologies provide communication between the car and the Internet through the built-in Internet modem or Bluetooth-pair cellular phone. In addition to Internet access, the vehicle can be connected to external devices, such as smart-phones or tablet PCs. One example is the Open Source Car Control (OSCC) project that allows developers to send vehicle control commands, read control messages from the vehicle's CAN OBD-II (On-Board Diagnostics) network and send reports for the current vehicle control state [2]. Applications using OSCC API functions can retrieve data from the vehicle (OBD-II port) and display them on a smartphone (under the control of the Android or iOS system) via the vehicle interface (VI) as depicted in Figure 1.

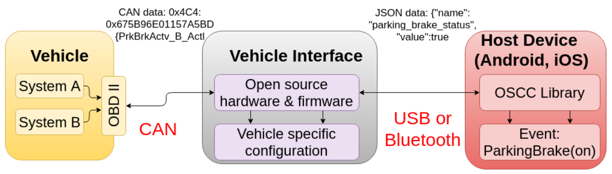


Figure 1: *This architecture diagram illustrates how the vehicle, VI, and the mobile are connected.*

As wireless communication technologies evolve, new security risks emerge. According to the Kaspersky Lab [3] in 2017, 27% of the total number of exploits are for Android applications and the annual growth of such exploits is 6%. With the increase in the number of vehicles with support for wireless technologies, the likelihood of attacks aimed at a vehicle using mobile applications is also increasing. Therefore, there is a need to find out how modern vehicles are vulnerable to these threats.

II. Related work

The potential fragility of the vehicle environment is an urgent topic for research. Several scientific groups were involved in the search for vulnerabilities in vehicles, [4], [5], [6], [7], [8], [9], [10]. In most cases, they address the security issues of the popular CAN bus protocol.

Using wireless and wired physical layer relays, Danev et al. [4] create two attacks that allow an attacker to open and run a car, and to transfer messages between the car and the smart key. They further analyze the critical characteristics of the system.

Charlie Miller and Chris Valasek [5] spoke at the information security conference (DEF CON 22) and made a practical overview of remote attacks on vehicles. Their analysis includes how large the surface of the remote attack on the ECUs and the vehicle features which allow computers to physically control it. In addition, this document describes protection mechanisms, including the intrusion detection and prevention systems.

Phung et al. [7] quite accurately classify cyber attacks on vehicles and the safety-security levels for ECU categories based on remote diagnostics attacks and Firmware updates Over The Air (FOTA) attacks.

Olovsson et al. [9] review a study on the security of a connected vehicle and, in particular, its automotive network. Their goal was to highlight the current state of this situation; identified problems, and what solutions were proposed before. A classification was made into the following five categories: automotive network problems, architectural security functions, intrusion detection systems, lures, threats, and attacks. Also, several areas were identified which, in their opinion, are of immediate concern.

Samuel Woo et al. [10] show that a wireless attack over long distances is physically possible using a real vehicle and a malicious application for smartphones in a related automotive environment. They also proposed a safety protocol for CAN as a countermeasure designed in accordance with the current CAN specifications. Their results show that the proposed security protocol is more efficient than existing security protocols with respect to authentication delay and communication load.

III. Background

The Figure 2 shows a conceptual model of interaction of a vehicle internal elements with external devices [7]. The network model in the vehicle consists of the following main elements: the Controller Area Network (CAN), Media-Oriented Systems Transport (MOST), and Local Interconnect Network (LIN). Gateways connect different networks for inter-working.

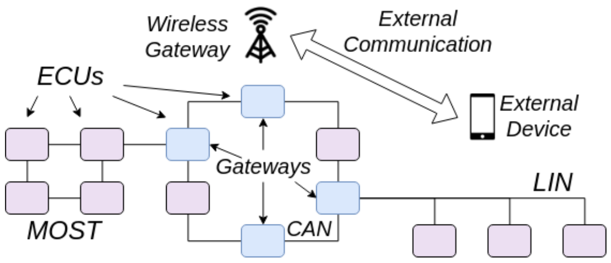


Figure 2: *Conceptual model of an in-vehicle network*

CAN is the most common network in a vehicle developed in the early 1980s for efficient communication between automotive electronic control units. In those days, when the CAN protocol was designed, the risks of information security was not considered. Therefore, most of the research in the vehicle security field is based on an analysis of the CAN protocol.

The MOST is a bandwidth multimedia network technology that uses a ring topology to carry audio, video, voice and data signals via physical layers. The LIN is a low-speed serial network protocol used for non-safety critical sensor and actuator systems. In an in-vehicle network, a wireless gateway is used to communicate with external devices. External devices use a wireless gateway to access the ECUs for firmware updates or for remote diagnostic procedures.

Each ECU is responsible for a certain functionality in the vehicle. For example, one ECU is responsible for door locking and another one for the speed control. More complex functions are performed by several ECUs jointly. A large risk for the vehicle, passenger, and manufacturer poses the ECUs, which interact with the outside world, as well as with the internal network of vehicles. For intruders, this is a potential entry point and the ability to exploit the vulnerabilities of the system.

IV. Attack model

Functionality, such as infotainment, remote diagnostics, telematics, fleet management, requires that the internal network of the ECU interacts with external networks via wireless interfaces. Wi-Fi, Bluetooth, custom radio frequencies and in principle any wireless interface is open for compromises, if not properly provided. Consequently, there is no single point or unit for security, but number some of them. The attack surface grows in favor of opponents. We investigated different vectors of attacks and listed some of them below.

Access OBD II interface. One way of attacking is through OBD II interface when the vehicle is being serviced. To attack the wireless end-to-end device, the attacker would need to install an access point (AP) next to the workshop, which has a signal strong enough to overcome the default AP signal used in the workshop. Thus, the wireless transit device will connect to the attacker AP, allowing an attacker to gain access to the CAN bus or even sniff and intercept the connection between the car and the mechanic's computer.

Attack 3rd party GSM to CAN adapter. The second interface to the internal CAN network can be a 3rd party CAN to GSM adapter, if one exists on the vehicle.

Attack Tire Pressure Monitoring System (TPMS) gateway. This attack can be done using a custom antenna. Implementing such an attack requires special expensive equipment, and with enough effort, the sensor signals can be reverse engineered.

Relay attack on Remote Keyless Entry (RKE) system. RKE systems allow the vehicle owner to lock and unlock the car doors at a distance, and to enable and disable the alarm if available. RKE systems apply unidirectional data transfer using a remote control that is built into the car key. When the button is pressed, the active radio frequency (RF) transmitter in the remote control generates signals in the freely usable frequency band.

Misuse infotainment - Instrument Cluster (ICL) system. Considering the Bluetooth way in, an attacker could first create a malicious application for commercial smartphones, which hides its malicious intentions. Then, using social engineering the attacker could trick the driver into installing the application on their phone, with the driver becoming a non-malicious insider attacker. Once the phone is paired and connected to the infotainment system via Bluetooth it would scan for the available Bluetooth services offered by the system and look for exploits. Then, the application would misuse the Bluetooth connection to inject another piece of malicious software on the infotainment system, using, for example, a buffer overflow attack. Last, the malicious software running on the infotainment ECU could misuse the diagnostic message functionality to access and attack the CAN segment through the ICL ECU.

Another similar way of using the Bluetooth interface is to reverse engineer the software running on the ECU and figure out vulnerabilities of the Bluetooth implementation. First, by using social engineering, an attacker could acquire documentation or the source code of the infotainment operating system or the documentation for that code. As an alternative, an attacker could rent or borrow a truck, dismount the infotainment system, dump the memory and work out the source code from the binary code. As a result, the created application can skip scanning the Bluetooth services and go straight to the code injection.

The third alternative to access the internal CAN via the infotainment system is by attacking the update procedure of the firmware that uses a CD. First, an attacker would have to get documentation or source code of the infotainment system. Then, they could create a custom firmware, which looks and acts normal but contains trojan malicious software. The next step would be to burn it on a CD with the right format for an update. The updated format can also be reverse engineered because the update functionality is part of the firmware. Then the attacker would have to trick the driver into installing the custom firmware on the vehicle's ECU.
To understand the scale of the investigated attacks

we made the attack model, where we counted possible risk of each attack. The risk for a given attack, according to work of C. Venakis and N. Vavoulas[11], is defined as the product of the total probability of success with the consequences shown in the Equation 1.

$$Risk(Attack) = Probability(Attack) * Surface(Attack) \quad (1)$$

The possible values for the two measures are given in tables below with columns for the real world value for each measure, and the probability. The probability for each real-world value is defined by a description, i.e. low/medium/high, a numeric range that the level of probability covers, and the approximation value used in calculations for that range, i.e. the average value of the range.

The probability of attack takes into account two factors - the cost of the attack based on equipment that is used and the detectability which is based on the distance that separates the attacker and victim. Table 1 and Table 2 shows values.
Table 3 demonstrates the attack surface .
The final attack model is presented in Table 4.

Table 1: *Probability of an attacker affording an attack based on cost*

Cost	Probability		
	Description	Range	Approximation
low	high prob	$0.67 \leq p \leq 1$	0,825
medium	medium prob	$0.33 \leq p \leq 0.67$	0,495
high	low prob	$0 \leq p \leq 0.33$	0,165

Table 2: *Probability of attack success without raising suspicions*

Suspicions raised	Probability		
	Description	Range	Approximation
low	high prob	$0.67 \leq p \leq 1$	0,825
medium	medium prob	$0.33 \leq p \leq 0.67$	0,495
high	low prob	$0 \leq p \leq 0.33$	0,165

Table 3: *Surface of attack*

Surface	Consequences		
	Description	Range	Approximation
low	low cons	$0.67 \leq p \leq 1$	0,825
medium	medium cons	$0.33 \leq p \leq 0.67$	0,495
high	high cons	$0 \leq p \leq 0.33$	0,165

Table 4: *Attack model*

Attack	Cost Prob	Susp Prob	Surface	Risk value
ICL: expolit BT by known vulnerabilities	low	high	high	0.561
Access OBD II interface	low	high	high	0.476
ICL: expolit BT by scanning	low	medium	high	0.343
Attack 3rd party GSM to CAN adapter Attack 3rd party GSM to CAN adapter	low	high	medium	0.286
Attack TPMS gateway	medium	high	low	0.220
Relay attack on RKE	medium	high	low	0.220
ICL: create custom firmware	medium	medium	high	0.202

V. Practical attack experiment

During our preparation for the security audit, we found out that we only have two attack vectors due to the lack of hardware devices that would allow us to conduct other types of remote attacks. The first one is on automotive RKE systems and the other one is Bluetooth.

A. *Attack on the RKE system*

The first car remote controls did not use cryptography at all: the car was unlocked after successfully receiving a constant "fixed code" signal. Such systems are subject to a replay attack. To prevent replay attacks modern RKE systems apply rolling code (hopping code) technology that uses cryptography and a counter value that increases with each press of a button.

At the initial stage, it was required to determine which RKE technology was implemented in the Kia

Soul car. However, the search for public documentation did not lead to any results. Apparently, these systems are a complete black box without any publicly documented security analysis. In order to analyze such a closed system, it was necessary to perform a number of practical experiments.

To receive (and send) RKE signals, we used various equipment, including Software-Defined Radios (SDRs) (HackRF, RTL-SDR DVB-T USB sticks) and some RF modules (transmitter/receiver). On the Kia Soul remote control, there is a CMIIT[1] identifier, through which the operating frequency of RKE system is recognized (Figure 3). This identifier points to the band 433.92MHz with a range of up to several tens of meters.

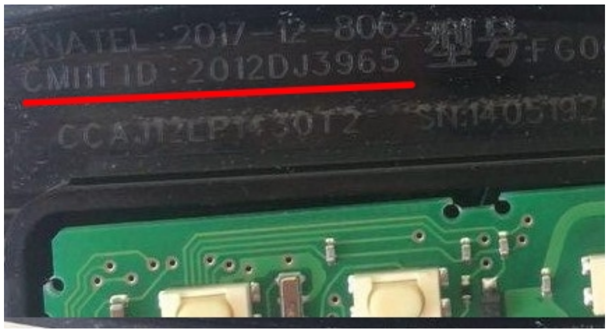


Figure 3: *A CMIIT identifier*

By setting the desired frequency (433.92MHz) in the SDR sharp program and using the RTL-SDR DVB-T USB stick, we can determine the type of modulation that is used by the RKE system (Figure 4).

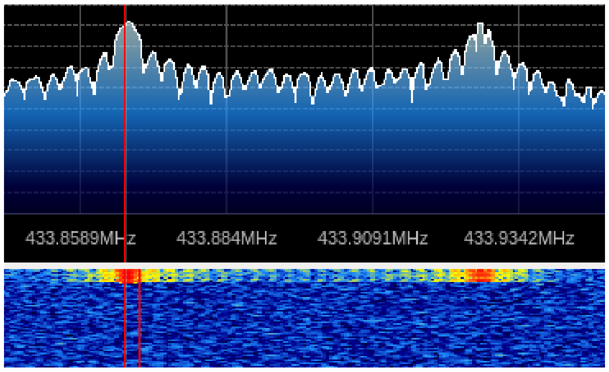
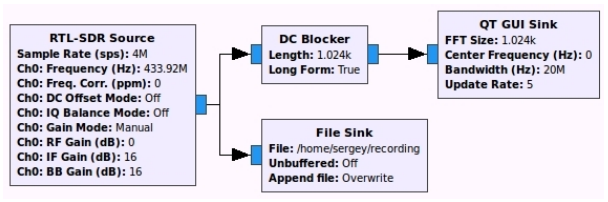


Figure 4: *The remote control signal capturing using the SDR Sharp program and the RTL-SDR DVB-T USB stick.*

As can be seen from the Figure 4, the system uses Gaussian Frequency-Shift Keying (GFSK) as a modulation scheme. The meaning of this modulation is that different symbols are coded as different frequencies. For example, the signal at 433.86 MHz is 1, and the signal at 433.92 MHz is 0. This results in a frequency-modulated signal with a central frequency of 433.89 MHz and a deviation of 300 kHz. The Gaussian filter helps to reduce interference to neighboring frequencies caused by abrupt changes in the carrier frequency.

Next, with the GNU Radio program, we can store the captured signal (Figure 5), and then transmit (Figure 6) the same signal using the HackRF device. But we could not transmit the captured signal using HackRF since HackRF stopped working, did not send or receive any signals.



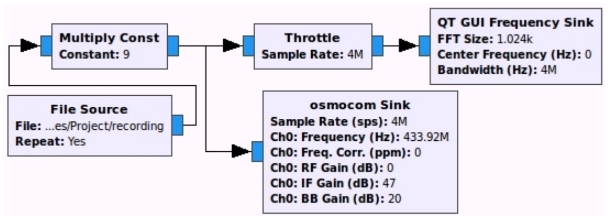
RTL-SDR Source
Sample Rate (sps): 4M
Ch0: Frequency (Hz): 433.92M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 16
Ch0: BB Gain (dB): 16

DC Blocker
Length: 1.024k
Long Form: True

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 20M
Update Rate: 5

File Sink
File: /home/sergey/recording
Unbuffered: Off
Append file: Overwrite

Figure 5: *GNU Radio scheme to store the signal*



Multiply Const
Constant: 9

Throttle
Sample Rate: 4M

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 4M

File Source
File: ...es/Project/recording
Repeat: Yes

osmocom Sink
Sample Rate (sps): 4M
Ch0: Frequency (Hz): 433.92M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 47
Ch0: BB Gain (dB): 20

Figure 6: *GNU Radio scheme to transmit the signal*

As the transmitting device, we could use Arduino with the transmitting radio frequency module but the problem lies in the amount of raw data (about 40 MB) that must be transmitted. For a qualitative recording of the signal to the file, a sample rate of 4 million samples per second was used, which is an excessively large value for Arduino transmitter.

[1]A CMIIT ID is the product ID assigned by the China Ministry of Industry and Information Technology to identify wireless products in the Chinese market.

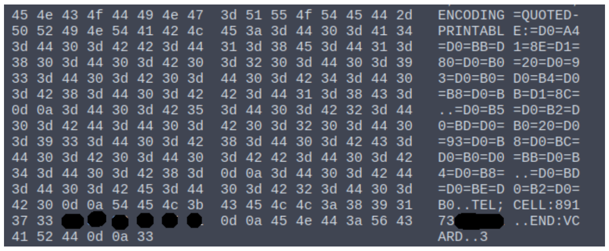
B. *Attack on Bluetooth*

We started our security auditing of Bluetooth by enabling *Bluetooth HCI snoop log* feature, which will allow us to see all the traffic that will be exchanged over Bluetooth. Then we connected the phone to the car by Bluetooth and started doing some stuff, for example, play some music, start a call, receive a call, and call ending, all of these activities was to see if we will be able to see anything in the Bluetooth log file.

Normally we should see encrypted data because the exchanged data is protected by a key that is generated during Bluetooth pairing. Since we are capturing directly from the device we will have the data before being encrypted.

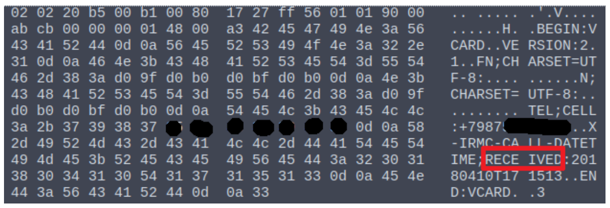
Breaking the encryption can be done easily with the Crackle [12] tool which will give us a long-term key (LTK). This key is used to generate the session key for an encrypted connection. This means that by obtaining this key we can decrypt any further communication between the phone and the car.

After examining the log file with Wireshark[13], we can see that the data that is related to the contact list and phone call log was sent either in cleartext or using a quoted-printable [14] encoding which is a very weak coding algorithm. In addition, we notice that only the name in the contact list is encoded but we can see the phone number in clear text. Here are some screenshots of our finding (Figure 7 and Figure 8).



```
45 4e 43 4f 44 49 4e 47  3d 51 55 4f 54 45 44 2d   ENCODING =QUOTED-
50 52 49 4e 54 41 42 4c  45 3a 3d 44 30 3d 41 34   PRINTABL E:=D0=A4
3d 44 30 3d 42 42 3d 44  31 3d 38 45 3d 44 31 3d   =D0=BB=D 1=8E=D1=
38 30 3d 44 30 3d 42 30  3d 32 30 3d 44 30 3d 39   80=D0=B0 =20=D0=9
33 3d 44 30 3d 42 30 3d  44 30 3d 42 34 3d 44 30   3=D0=B0= D0=B4=D0
3d 42 38 3d 44 30 3d 42  42 3d 44 31 3d 38 43 3d   =B8=D0=B B=D1=8C=
0d 0a 3d 44 30 3d 42 35  3d 44 30 3d 42 32 3d 44   ..=D0=B5 =D0=B2=D
30 3d 42 44 3d 44 30 3d  42 30 3d 32 30 3d 44 30   0=BD=D0= B0=20=D0
3d 39 33 3d 44 30 3d 42  38 3d 44 30 3d 42 43 3d   =93=D0=B 8=D0=BC=
44 30 3d 42 30 3d 44 30  3d 42 42 3d 44 30 3d 42   D0=B0=D0 =BB=D0=B
34 3d 44 30 3d 42 38 3d  0d 0a 3d 44 30 3d 42 44   4=D0=B8= ..=D0=BD
3d 44 30 3d 42 45 3d 44  30 3d 42 32 3d 44 30 3d   =D0=BE=D 0=B2=D0=
42 30 0d 0a 54 45 4c 3b  43 45 4c 4c 3a 38 39 31   B0..TEL; CELL:891
37 33 [redacted]         0d 0a 45 4e 44 3a 56 43   73[redacted] ..END:VC
41 52 44 0d 0a 33                                  ARD..3
```

Figure 7: *Contact list is sent with name encoding using quoted printable*



```
02 02 20 b5 00 b1 00 80  17 27 ff 56 01 01 90 00   .. ..... .'.V....
ab cb 00 00 00 01 48 00  a3 42 45 47 49 4e 3a 56   ......H. .BEGIN:V
43 41 52 44 0d 0a 56 45  52 53 49 4f 4e 3a 32 2e   CARD..VE RSION:2.
31 0d 0a 46 4e 3b 43 48  41 52 53 45 54 3d 55 54   1..FN;CH ARSET=UT
46 2d 38 3a d0 9f d0 b0  d0 bf d0 b0 0d 0a 4e 3b   F-8:.... ......N;
43 48 41 52 53 45 54 3d  55 54 46 2d 38 3a d0 9f   CHARSET= UTF-8:..
d0 b0 d0 bf d0 b0 0d 0a  54 45 4c 3b 43 45 4c 4c   ........ TEL;CELL
3a 2b 37 39 38 37 [redacted]       0d 0a 58   :+7987[redacted]..X
2d 49 52 4d 43 2d 43 41  4c 4c 2d 44 41 54 45 54   -IRMC-CA LL-DATET
49 4d 45 3b 52 45 43 45  49 56 45 44 3a 32 30 31   IME;RECE IVED:201
38 30 34 31 30 54 31 37  31 35 31 33 0d 0a 45 4e   80410T17 1513..EN
44 3a 56 43 41 52 44 0d  0a 33                     D:VCARD. .3
```

Figure 8: *Call log are sent in clear text*

VI. Discussion

What technology of the RKE system is used in the Kia Soul car remains unexplored due to the lack of working equipment. In this article, we implemented a single remote attack on Bluetooth, but there are also other remote attack vectors, for example, the replay car remote control signal by using HackRF[15] to see the used signal encoding and try to recreate these signals, mobile application auditing by fuzzing the server side or by using hooks and reverse engineer the client-side app, and more Bluetooth testing with more advanced adapters like ubertooth[16]. By using this adapter we will be able to capture and send Bluetooth packets to any Bluetooth enabled device in the range of Bluetooth.

VII. Ethical issues

This project will have to comply with all ethical standards of morality and in its research will not be violated any laws concerning the exploitation of found vulnerabilities in real life.

VIII. Conclusions and suggestion for further research

The remote attacking vectors is a very large attacking surface, testing every possible attack requires a lot of time and a lot of hardware. But it was found that when a mobile device is connected to the car by Bluetooth, some data leaks can be detected.

For the future work, we could try to retransmit the captured packets and signals using ubertooth and HackRF to have a deep understanding of how each command is implemented. Also, we could do security auditing of the mobile application or the rolling code technology of the RKE system.

References

[1] Capers Jones Christof Ebert. "Embedded Software: Facts, Figures, and Future". In: *IEEE Computer Society Press Los Alamitos* (2009). DOI: `10.1109/MC.2009.118`.

[2] The OSCC Project. *Open Source Car Control.* 2018. URL: `https://github.com/PolySync/oscc` (visited on 05/02/2018).

[3] Kaspersky Security Lab. *Overall statistics for 2017.* 2017. URL: `https://kasperskycontenthub.com/securelist/files/2017/12/KSB_statistics_2017_EN_final.pdf` (visited on 05/02/2018).

[4] Boris Danev Aurélien Francillon and Srdjan Capkun. "Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars". In: *Proceedings of the Network and Distributed System Security Symposium* (2011).

[5] Chris Valasek Charlie Miller. "A Survey of Remote Automotive Attack Surfaces". In: *DEF CON 22 Hacking Conference Presentation* (2014).

[6] Franziska Roesner Karl Koscher Alexei Czeskis. "Experimental Security Analysis of a Modern Automobile". In: *Proc. of the 31st IEEE Symposium on Security and Privacy* (2010). DOI: `10.1109/SP.2010.34`.

[7] Phu H. Phung Dennis K. Nilsson and Ulf E. Larson. "Vehicle ECU classification based on safety-security characteristics". In: *Road Transport Information and Control - RTIC 2008 and ITS United Kingdom Members' Conference, IET* (2008). DOI: `10.1049/ic.2008.0810`.

[8] David Oswald. "Wireless Attacks on Automotive Remote Keyless Entry Systems". In: *TrustED'16 - Proceedings of the 6th International Workshop on Trustworthy Embedded Devices* (2016). DOI: `10.1145/2995289.2995297`.

[9] Tomas Olovsson Pierre Kleberger and Erland Jonsson. "Security Aspects of the In-Vehicle Network in the Connected Car". In: *IEEE Intelligent Vehicles Symposium (IV)* (2011). DOI: `10.1109/IVS.2011.5940525`.

[10] Hyo Jin Jo Samuel Woo and Fellow Dong Hoon Lee. "A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN". In: *IEEE Transactions on Intelligent Transportation Systems* (2014). DOI: `10.1109/TITS.2014.2351612`.

[11] C. Xenakis N. Vavoulas. "A quantitative risk analysis approach for deliberate threats". In: *Critical Information Infrastructures Security* (2011).

[12] Mike Ryan. *Crackle cracks BLE Encryption.* URL: `https://github.com/mikeryan/crackle`.

[13] *Wireshark homepage.* URL: `https://www.wireshark.org/`.

[14] *Quoted printable encoding.* URL: `https://en.wikipedia.org/wiki/Quoted-printable`.

[15] *hackrf github repository.* URL: `https://github.com/mossmann/hackrf`.

[16] *Ubertooth github repository.* URL: `https://github.com/greatscottgadgets/ubertooth`.