Indonesian Journal of Electrical Engineering and Computer Science
Vol. 21, No. 1, January 2021, pp. 321~327
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v21.i1.pp321-327

Implementation of a bluetooth attack on controller area network

Zniti Asmae, Pr. El Ouazzani Nabih
Faculty of sciences and Technology (FST), University Sidi Mohamed Ben Abdellah, Signals Systems and Components Laboratory (LSSC), Fez, Morocco

Article Info

Article history:

Received Jun 30, 2020
Revised Aug 11, 2020
Accepted Aug 21, 2020

Keywords:

Automobile
Bluetooth
Controller area network
Security
Wireless attack

ABSTRACT

In this paper a general overview of the vulnerability of the controller area network (CAN) bus is presented and a practical short-range attack is proposed. There are more and more potential attacks on the CAN bus, which may cause leakage of information and thereby there may be danger for safe driving. The attack combines several techniques, such as how to update a node firmware using a Bluetooth module and inject a priority fake frame, in order to block the legitimate messages.

Corresponding Author:

Zniti Asmae
Signals Systems and Components Laboratory (LSSC)
University Sidi Mohamed Ben Abdellah
Faculty of sciences and Technology (FST), Fez, Morocco
Email: znitiasmae@gmail.com

1. INTRODUCTION

The controller area network (CAN) is a multi-master serial data bus with real-time capabilities [1, 2] CAN bus was designed in late 1980 by Bosch [3], for robust and flexible performances in harsh environments and particularly for automotive applications and it also turned out to be very useful in industrial systems [4-6]. Automotive applications consist of electronic control units (ECU) [7] that monitor all of the car’s systems and sensors. The CAN bus acts as a central networking system, enabling communication between all the components [8]. Nevertheless, recently, several researchers have pointed out potential vulnerabilities in automotive CAN buses. In [9-13] some security limitations of the protocol, that are common to numerous implementations, have been highly outlined. As well known, CAN packets contain neither authenticator [14] nor identifier fields, meaning that any component can send data.

Hence, internal vehicles networks are under the threat of damaging attacks through various techniques that can be divided into three categories:

a) Indirect Physical Attacks: Some of the demonstrated attack models of physical access take place through two main channels [15]: the multimedia player and the PASS THRU device, which connects directly to the OBD-II port and allows communication with the car's internal network.
b) Short Range Attacks: As a result of the inherent characteristics of short and long wireless embedded in the vehicle’s infotainment system, researchers discovered that a multitude of attacks could be launched such as Bluetooth [15] and Tire Pressure Monitoring System (TPMS) [16]. The TPMS consists mainly

of a sensor built into tires that is responsible for sending information about pressure, temperature and rotation via a radio frequency (RF) transmitter. The receiver's computer analyzes the data and sends results or commands to other computers via the CAN bus.

c) Long Range Attacks [17]: In 2014, The analysts Miller, C. and Valasek, C. managed to show that remote attacks on vehicles are possible, by closely analyzing the entertainment system of the Jeep Cherokee called UCONNECT. The researchers discovered that port 6667 is open and allows executing inter-process commands, without any need for authentication.

The proposed work describes the methodology for formulating an attack on the network and how to inject arbitrary malware into a CAN node. It also shows the danger of having access to the car's internal network and taking control of it. The attack is implemented and tested on Arduino UNO [18] using HC-05 module [19]. The remainder of this paper is structured as follows: Section 2 introduces the CAN protocol, Section 3 describes the relevant details, and the results are provided in Section 4. Finally, Section 5 concludes the paper.

2. CAN BUS PROTOCOL

Exchange of information and communication between several nodes within a vehicle takes place according to the CAN protocol involving four frames [20-22]:

2.1. Data frame

The data frame contains node data for transmission along with other fields as shown in Figure 1.

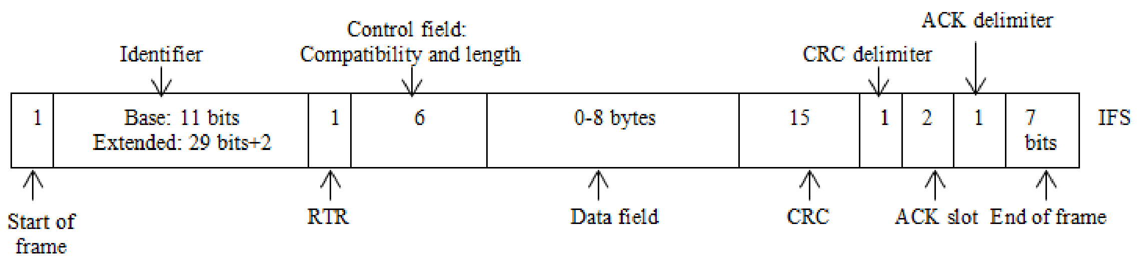


Figure 1. Data frame format of CAN protocol

The frame fields involved in the communication network function as described below:

Start of Frame bit: indicates the beginning of a message with a dominant bit (logic 0)

Identifier: identifies the message and indicates the message's priority. Frames come in two formats, standard which uses an 11-bit arbitration ID, and extended, which relies on a 29-bit arbitration ID

RTR (Remote Transmission Request)**:** This bit is used to differentiate a remote frame from a data frame. A logic 0 (dominant bit) indicates a data frame, whereas a logic 1 (recessive bit) indicates a remote frame

Control field: Informs the length of the Data in bytes (0 to 8 bytes)

Data Field: May contain from 0 to 8 bytes of data

CRC (Cyclic Redundancy Check) [23]: The CRC field is used for error detection. It contains 15-bit cyclic redundancy check code and a recessive delimiter bit.

ACK slot (Acknowledgement slot)**:** Any CAN controller that correctly receives the message sends an ACK bit at the end of the message. The transmitting node checks for the presence of the ACK bit on the bus and reattempts transmissions if no acknowledge is detected.

End of Frame: Marks the end of the CAN frame.

IFS (Intermission Frame Space)**:** Minimum number of bits separating consecutive messages.

2.2. Remote frame

The remote frame requests without a data field, the transmission of a data frame for a given identifier, Figure 2.

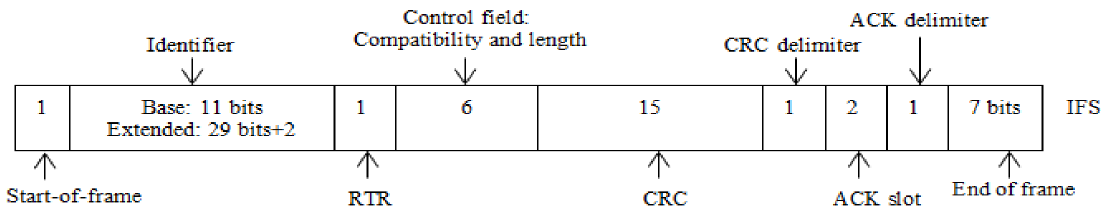


Figure 2. Remote frame format

2.3. Error frame

The error frame is transmitted by any node detecting an error. The error frame consists of two different fields as indicated in Figure 3. The first field is given by the superposition of ERROR FLAGS (6–12 dominant / recessive bits) contributed from different stations. The second field is the ERROR DELIMITER (8 recessive bits).

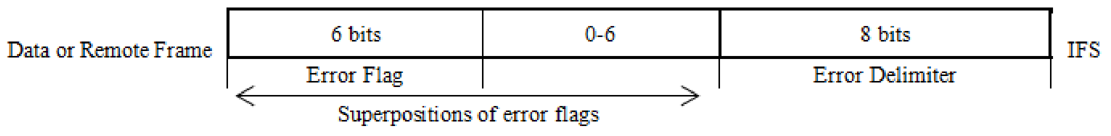


Figure 3. Error frame format

2.4. Overload frame

The overload frame allows a delay between handling data or a remote frame. The format is very similar to that of an Error Frame and contains two fields: Overload flag and Overload Delimiter, Figure 4.

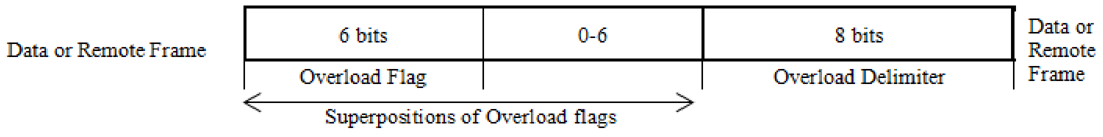


Figure 4. Overload frame format

3. PROPOSED PROTOTYPE DESCRIPTION

3.1. CAN bus-based prototype

The proposed prototype, shown in Figure 5 is made up of three nodes functioning as follows:

a) Node 1 is connected to an LED and communicates in parallel with a smartphone that controls the LED lighting via Bluetooth communication.

b) Node 2 monitors a temperature sensor, that is, it reads the value and sends the message to node 3.

c) Node 3 receives data from node 2 and displays it on the LCD.

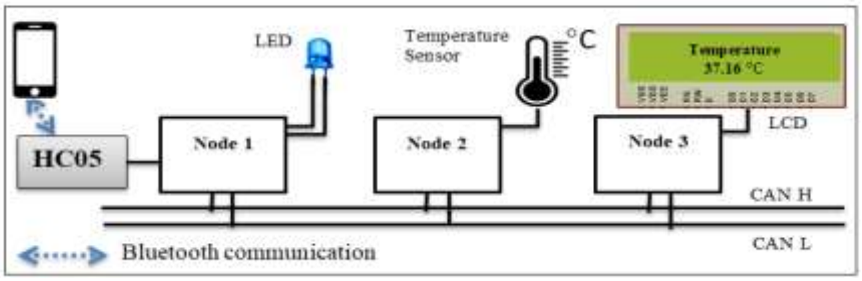


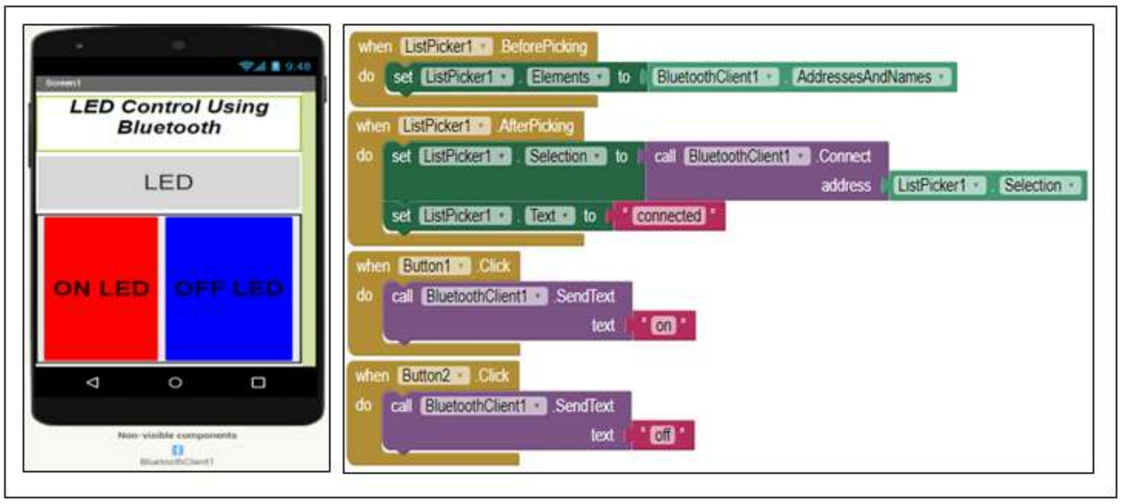
Figure 5. Proposed CAN bus prototype

3.2. Prototype hardware

In order to implement the proposed prototype, the set-up of Figure 7 has been considered with the following devices:
Arduino UNO integrated into each of the three nodes.
MCP2515 is a CAN module interfacing with Arduino through SPI communication.
HC-05 ensures data transmission from the Smartphone to the Arduino Uno via Bluetooth.
LM35 is a temperature sensor.
LCD displays data issued by LM35.
Smartphone runs the prototype's application to command the illumination of the LED.

The application program is built by means of the MIT App Inventor [24] for graphical programming by putting up blocks; Figure 6 illustrates the procedure.

As a result of clicking on:
“ListPicker1: LED”, the connection will be established between the selected device and the application, and then the status label will be set to be connected.
“Button 1: ON LED”, the application sends on in a message by Bluetooth which causes the illumination of the LED.
“Button 2: OFF LED”, the application sends off in message that is interpreted as an order to turn off the LED.



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to call BluetoothClient1 .Connect
                                        address ListPicker1 . Selection
    set ListPicker1 . Text to " connected "

when Button1 .Click
do  call BluetoothClient1 .SendText
                          text " on "

when Button2 .Click
do  call BluetoothClient1 .SendText
                          text " off "
```

Figure 6. Application program in the designer and in the blocks editor

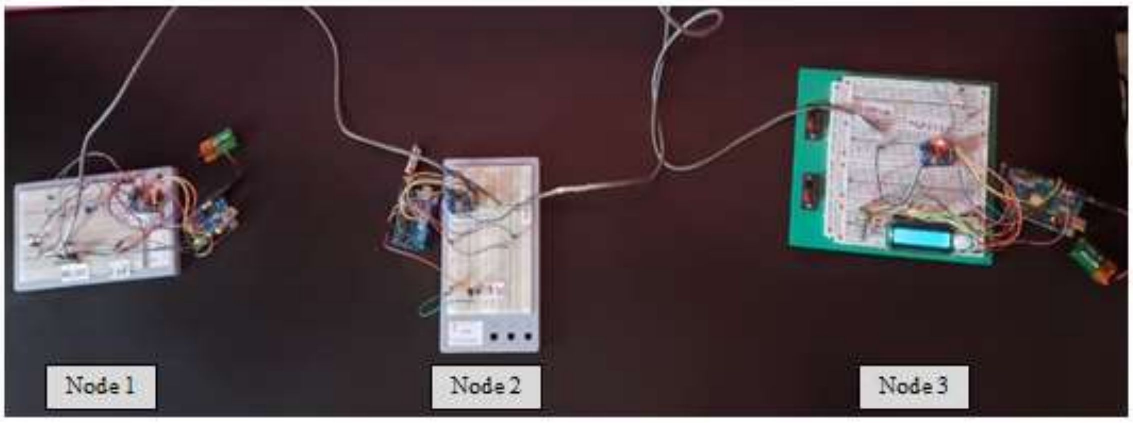


Figure 7. Prototype set-up

4. DESCRIPTION OF THE ATTACK

4.1. Bluetooth interface issue

Bluetooth is one of the most famous technologies built into cars and represents an interface that can be used to launch remote attacks directly or indirectly.

a) Indirect attack: The scenario of the attack implies that the owner of the target vehicle downloaded the program to their smartphone without noticing the operation. When coupled to the vehicle's telematics system, the application triggers its attack and takes control of the ECU.
b) Direct attack: The first step is to identify the Media Access Control (MAC) address of the car (Bluetooth) then pair it. However, it turns out that the car can accept pairing requests without the intervention of the driver, which allows the attacker to find the password generated by the car to confirm the pairing.

4.2. Implementation

The attacker's goal is to take advantage of the CAN bus vulnerabilities in order to inject malicious messages and disrupt the normal operations of the network remotely. As mentioned before, node 1 uses a Bluetooth communication protocol; so it is obvious all the work will run on this node.

The malicious code aims to send a false temperature value permanently on the CAN bus with a high priority ID while the other nodes will be forced to stop their broadcasts and distort the real value of temperature and mislead the user that there is a problem as can be seen in Figure 8. The procedure for setting up the attack is carried out according to two steps, which are:

a) Step 1 deals with pairing the malicious phone with the HC05 module, the device password is set to 4 numbers, which means that the attacker will have to make 10^4 attempts to succeed. The probability of success of a random attack is therefore $1/\ 10^4$, which implies that the hacker should stay close to the target vehicle for a time. What can make it even easier for attackers is the fact that there are several demonstrations of attacks on Bluetooth such as that of [25], which shows that the most common Bluetooth password can be cracked within less than 0.3 seconds.
b) Step 2, once the pairing is successful, the malware hex file should be generated and raised to node1 with Bluetooth. To make this procedure easier a Bluino loader application is used. Arduino IDE programming software that runs on Android facilitates the writing of sketch code, the compilation of sketch to generate a file Hexadecimal and download it on various Arduino cards via Bluetooth.

The moment the Arduino receives data on its serial port, it checks automatically that the received sequence of characters indicates that there is a new program to be downloaded.

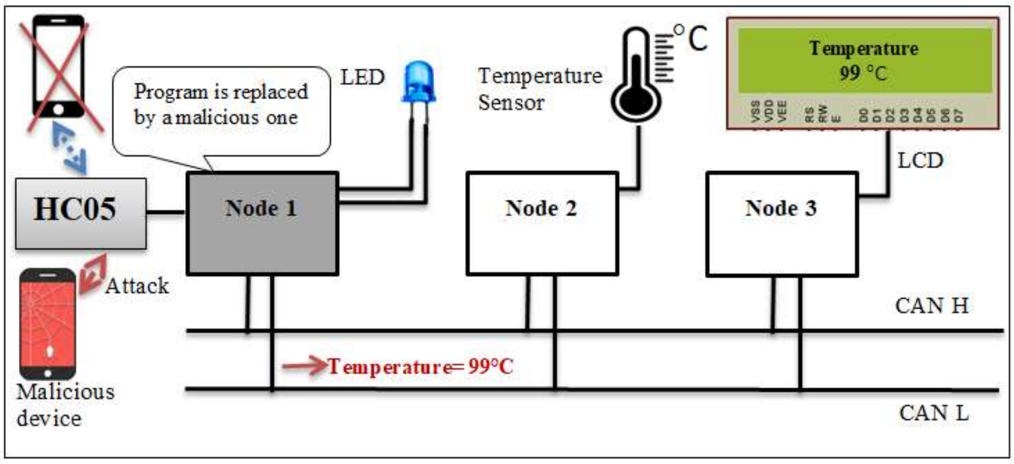


Figure 8. Implementation of the attack

When uploading IDE ARDUINO on the node, the following operations are executed:

a) The ARDUINO bootloader listens to the serial line
b) The IDE sends the .hex file to the bootloader
c) The latter extracts the program and writes it in the FLASH memory
d) The bootloader launches the application
e) Node 1 sends a confusing temperature value over the CAN bus

5. RESULTS AND DISCUSSION

The implementation of the described wireless attack demonstrates the CAN bus vulnerabilities in terms of data transmission and malicious nodes interventions. The most disturbing features of the proposed prototype are related to technical aspects and execution time. Indeed, the attack has been carried out relying on widely available equipments such as ordinary smartphones and ARDUINO cards, making it a low cost operation. In addition, the whole process requires very low time consumption. Thus, hackers have become more interested in such an attack as the implementation is quite simple and accessible as well. It is also important to mention, that in car’s conditions the proposed attack can be multiplexed, which can be used to simultaneously target multiple nearby cars to increase the chances of success in the event that the attacker does not seek a specific vehicle.

6. CONCLUSION

This paper describes how an attacker can take control remotely of a Controller Area network. The main contribution of our work is to prove the potential fragility of the CAN bus and assess the danger that can be caused by modern cars. Indeed, part of the objectives is to make this framing concrete by providing comprehensive experimental results assessing the danger that can be caused by the lack of security on the CAN bus in modern cars. This attack can be used to silently eavesdrop on the data that is being sent by a node, to modify its content, or to take total control of the network.

REFERENCES

[1] R. I. Davis, *et al.*, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised, " *Journal of Real-Time Systems*, vol. 35, no. 3, pp. 239-272, 2007.

[2] M. Di Natale, *et al.*, "Understanding and using the controller area network communication protocol: theory and practice, " *SpringerScience & Business Media*, 2012.

[3] R. Bosch. “Can specification”, version 2.0. Stuttgart, 1991.

[4] T. Ken and B. Alan, "Guaranteeing message latencies on control area network (CAN)," *Proc. 1st International CAN Conference*, 1994.

[5] Z. Khawar and S. Kang, "Scheduling messages on controller area network for real-time CIM applications," *IEEE Trans. Robotics and Automation*, vol. 13, no. 2, pp. 310-314, April 1997.

[6] M. Di Natale, "Understanding and using the controller area network, " *inst. eecs. berkeley*. edu/˜ ee249/fa08/Lectures/handout canbus2. pdf, 2008.

[7] K. Koscher, *et al.*, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on. IEEE*, pp.447-462, 2010.

[8] R. Currie, "Developments in Car Hacking," *SANS Institute*, 2016.

[9] P. R. Thorn and C. A. MacCarley, "A spy under the hood: Controlling risk and automotive EDR," *Risk Management*, vol. 55, no. 2, pp.22-26, 2008.

[10] M. Wolf, *et al.*, "Security in automotive bus systems, " in *Proceedings of the Workshop on Embedded Security in Cars,* 2004.

[11] M. Wolf, *et al.*, "State of the art: Embedding security in vehicles," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 1. pp.1-16, 2007.

[12] Zhao, Y. "Telematics: safe and fun driving," *Intelligent Systems, IEEE,* vol. 17, no. 1, pp.10-14, 2002.

[13] I. Foster and K. Koscher, "Exploring Controller Area Networks," *login. USENIX Association*, vol. 40, no. 6, 2015.

[14] A. Zniti and N. El Ouazzani, "Improvement of the Authentication on In-Vehicle Controller Area Networks," *Embedded Systems and Artificial Intelligence, 2020. ESAI 19.* Springer, pp. 23-32, 2020.

[15] S. Checkoway, *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Conference on Security. USENIX Association*, vol. 2011, pp.1-6, 2011.

[16] I. Rouf, *et al.*, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study,". in *I. Goldberg, editor, USENIX Security,* pp.323-338, 2010.

[17] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," in *Black Hat USA*, Vol. 2014, pp.1-90, 2014.

[18] Arduino, "Arduino, " 2015, https://www.arduino.cc/.

[19] HC-05 Serial Bluetooth Products User Instructional Manual. Available: www.electronicaestudio.com/docs/istd016A.pdf.

[20] C. Pan, *et al.*, "Modeling and verification of CAN bus with application layer using UPPAAL," *Electronic Notes in Theoretical Computer Science*, vol. 309, pp. 31-49, December 22, 2014.

[21] Li R, *et al.*, "Design method of can bus network communication structure for electric vehicle," in *International Forum on Strategic Technology IEEE*, pp. 326-329, 2010.

[22] Texas Instruments. Introduction to the Controller Area Network (CAN). SLOA101A. Dallas, TX. 2008.

[23] M. F. Hashmi and A. G. Keskar " An Optimized FPGA Implementation of CAN 2.0 Protocol Error Detection Circuitry, " *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS),* vol. 6, no. 3, pp. 602-614, 2017.

[24] S. Pokress and J. Veiga, "MIT App Inventor: Enabling personal mobile computing," in *Workshop on Programming for Mobile and Touch*, 2013.

[25] Y. Shaked and A. Wool, "Cracking the bluetooth pin," in *Proc. 3rd USENIX/ACM Conf. Mobile Systems, Applications, and Services (MobiSys)*, pp. 39-50, 2005.