Exercise #3 (24 Points)

Date due: June 4., 9:59 am

Submission Instructions

The solutions must be submitted via CMS until **June 4., 9:59 am** in PDF format. Make sure that you clearly indicate the names and matriculation numbers of all students who authored the solutions **on the sheet itself**. Please do not share info that is not publicly known (via the lecture or the exercise sheet), especially not in the forum. If you have questions that might leak deduced information, write a mail to the tutors or visit the office hours instead of creating a post in the forum. If you wrote any code to solve the exercises, please add it to your solution. **Undocumented code is not accepted**. If it is more than a few lines, please put it into a **dedicated appendix** and refer to it from your solution. We only accept solutions written in *C*, *C++* and *Python* if not stated otherwise. Be kind and format your code nicely.
Label your solutions corresponding to the exercises on the task sheet. Handwritten solutions are accepted, but they have to be readable; otherwise points might be deducted.

1 Teach me how to drive (7 points)

Since the day you helped out *MetaCortex* to extract the key from the compromised device installed at their door, your name is on everyone's lips. You recently upgraded your office and are now relaxing in a hammock, as the telephone wakes you from your dreams. The news of your skills got around quickly, so the car manufacturer from the next city, *FolksCar*, asks you for your help. They want to upgrade their car fleet to drive autonomously and have already developed a perfect selfdriving software. Unfortunately they lost the documentation of their CAN bus protocol (rumor has it that there was never one), so they are unable to send the command which controls the speed. You agree kindly to work for them, under the condition that they send you a dump such that you can work on the task at your office.

For this exercise, we provide you with a `txt` file that contains a dump from a CAN Bus of the specific car. It is part of the provided zip file. If you want to create your own dump file, go into the student working room in the second floor. You will find parts of a car there and two arduinos. One can be used to take fresh dumps. You can connect to this Arduino with the USB cable there (baud rate 115200). **Do not overwrite the software on the Arduino.**

The CAN arduino (identifiable via the ethernet cable) receives a UDP packet with a payload that represents a CAN frame (IP 192.168.1.177, port 8888). The first 3 characters are the frame ID, followed by a #, followed by up to 16 characters which build 8 bytes pairwise. Example: 2b9#deadbeefc0cac01a. If something went wrong,

the Arduino responds by sending a debug UDP packet with the error message back to your IP at port 5000. You need a static ip for this exercise. If you are unfamiliar with it, you may want to look up how to configure a static ip.
This exercise works with shared hardware. Start working early.

(2 points) (a) Analyze the dump. Differentiate the messages by their IDs and note down for every message type if their content changes or not. Additionally, state one device that listens to messages with content that is static for most of the time and another device that expects messages with content that changes during driving.

(5 points) (b) Find out which ID the speedometer listens to and the frame that sets the speed to 50km/h. Briefly and precisely explain how you got there and how the data is interpreted.

2 Baby you can drive my car! (8 points)

You successfully completed *FolksCar*'s job and as a reward they gifted you their newest car, the model *Croquet 7*. The self-driving software works like a charm; the only drawback is that you can no longer drive on your own. You know, however, that the car would let you drive in case of malfunction. Luckily, the car has a built-in WiFi function to allow a smartphone enabling the cars seat heating function remotely. You now search for a "hidden function" to fake the engine's temperature such that it thinks it's malfunctioning.

When working on task **1** you got your hands on some code driving the media system. It is included in the zip file. Remember to reboot the CAN Arduino before working on it.

This exercise works with shared hardware. Start working early.

(1 point) (a) Briefly explain what buffer overflows are.

(1 point) (b) Briefly explain how one can find vulnerabilities that can be exploited using a buffer overflow.

(1 point) (c) Briefly explain two ways to prevent buffer overflows.

(5 points) (d) Now analyze the code which is part of the entertainment system.

- State at which line the program is vulnerable to buffer overflows.
- Sketch your attack and what your benefit would be.
- Finally, state the payload of the UDP packet which lets you send the payload **0x33, 0xed, 0x00, 0x00, 0x00, 0x58, 0x52, 0x00** with the id **0x288** as a CAN message.
- If sent in the right frequency for several times, the engine temperature should overheat.

Figure 1: Control light for engine temperature

We highly recommend you to actually mount the attack, as a not working approach results in no points. You may verify your findings listening to the CAN bus with the dump Arduino. If your attack was successful, the engine overtemperature light should come on.

3 The bus CAN do. (9 points)

(1 point) (a) Concerning task **1**: Briefly state and explain a reason why some messages are sent circularly, even though the state of the corresponding device did not change.

(1 point) (b) In task **2** we learned how to manipulate devices hooked up to the CAN bus by inserting frames on the bus in order to send arbitrary CAN messages. In most cars our attack would not set the actual speed of a car. Briefly explain why not.

(2 points) (c) When designing the CAN architecture the manufacturer has to ensure that bus utilization is never too high when designing all messages. What is the CAN bus utilization (percentage wise) for the following cyclic messages on a 500kbit/s bus (you may ignore bit stuffing):

- `12 47 00` every 5ms,
- `ef 90 ff 01` every 5ms,
- `ef ff 4b 80` every 10 ms,
- `00 00 ff ff 4b` every 15 ms,
- `66 9a a0 d2 21 86 00 00` every 15 ms,
- `82 d2 01 ff e1 00` every 15 ms,
- `99 82 00 00 92 ff` every 25 ms,
- `61 9d bc c0 2f 7a 9d` every 50 ms,
- `02 88 00 00 00 62` every 100 ms,

(1 point) (d) *FolksCar* were impressed by your ability to set the speed of the car. However, they are now afraid that if someone finds a flaw in their quality software^{TM}, someone might be able to set the speed of the car while someone is driving. They instructed you again to find solutions for this problem.
You learned that the CAN bus protocol is old and it is not possible to verify the integrity of the message. As cars are getting more and more connected, this topic becomes more and more interesting, hence researchers proposed different ways of tackling this problem. One such technique is explained here [1].
Briefly explain what is used to secure the integrity of the messages.

(4 points) (e) The authors decided to use symmetric cryptography with static keys. List one advantage and one disadvantage each for instead using asymmetric cryptography or using a key exchange to establish a common key.

[1] `https://link.springer.com/chapter/10.1007/978-3-662-53140-2_6`, only free in eduroam