Challenge Sheet 4 (10 Points)

Date due: January 2nd 2022

Submission Instructions

Solutions to this challenge sheet must be submitted by **January 2nd 2022** *before 11:59 am* to the CMS. **Make sure that your solution contains your name and matriculation number at the top of the files!** It is not allowed to submit these solutions in a group, as you do with the regular exercise sheet solutions. In the .py files, you may add as many additional methods as you want, but please do not rename the existing methods or add code outside of methods. How many points you get for the tasks is determined by tests on our system. You can also leave some feedback about this sheet here: Survey 4

1 Ethernet

In this task we will play around with MACs and Ethernet frames. To do this, please implement the following functions in **ethernet.py**.

(2 points) (a) **bytesToMAC(...)** should convert a MAC address from bytes to the familiar, more human-friendly format.

(2 points) (b) **MACToBytes(...)** should do it the other way around and convert the human-friendly representation of a MAC into its byte version.

(3 points) (c) Last but not least, let's get a little familiar with Ethernet frames. In particular, Ethernet II frames as shown in the lecture (there are many different representations on the Internet, please stick to the representation from the lecture). For this please implement **parseEthernetFrame(...)** which should extract all important details from a given frame. You can assume that your method will only be confronted with correct frames with the same fields as shown in the lecture.

Hint: The imported methods might be helpful!

2 ARP

In the lecture, you heard about ARP spoofing and also some ways to protect against it, such as using a smart switch. Implementing such a switch will be part of today's challenge.

(3 points) (a) We have already modified the software of some switch to filter out ARP responses like "`B.B.B.B is at b0:b0:b0:b0:b0:b0`". It will take these responses and put their IP and MAC into the **detect(...)** method of **ARPSpoofingDetection** in **arp.py**. This method should return *True* if it detects responses for an IP address from different MACs or a MAC responding for more than one IP address. If it

detects nothing suspicions it should return *False*. To make it a bit easier, you can assume that the network will be shut down after something suspicious is detected. This means that you don't have to keep track of already suspicious MACs or similar.

Hint: You can use the arp_table attribute to store some information between method calls. You can also add as many additional attributes as you like!