The topology as you were able to derive it

IP addresses of all routers

Host/IPs in the As

Using if config and show ip bgp on each router to derive the following information

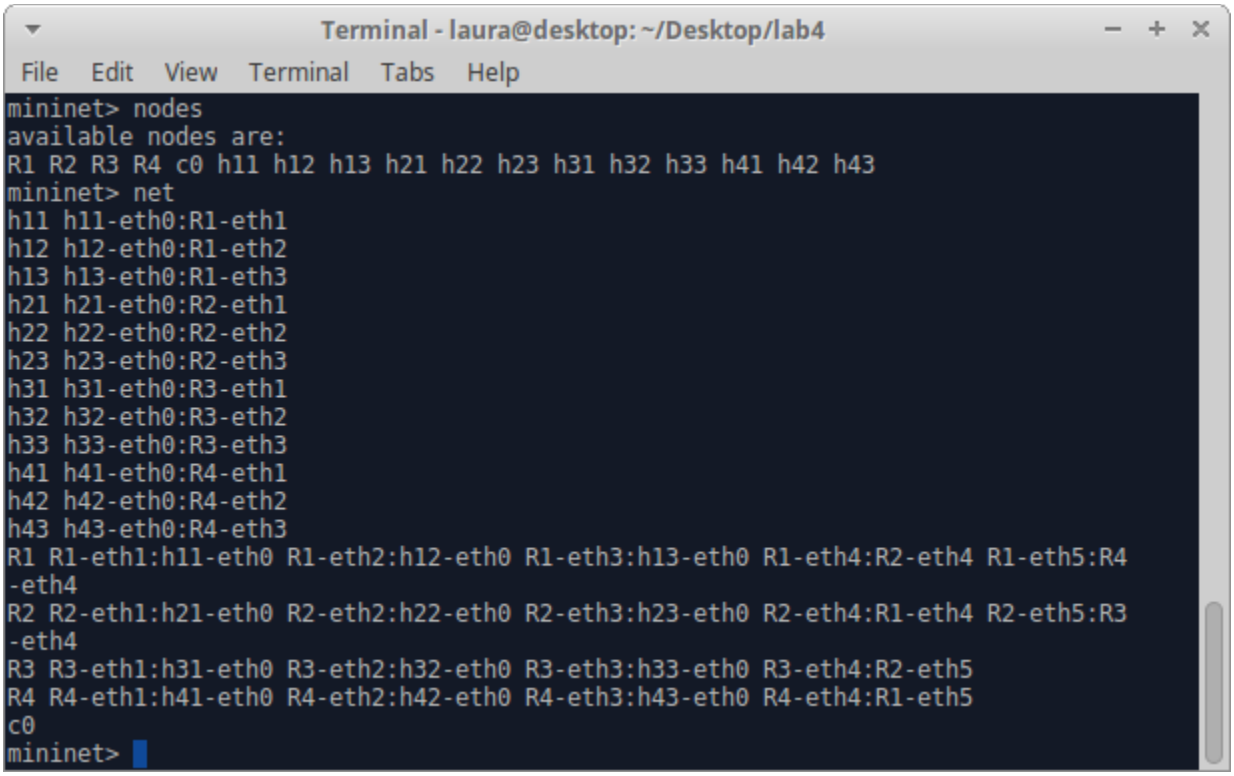

Figure 1: nodes and net information

<u>AS1</u>

AS1: 11.0.0.0/8

R1-eth1(h11): 11.0.1.254/24

R1-eth2(h12): 11.0.2.254/24

R1-eth3(h13): 11.0.3.254/24

R1-eth4(R2): 9.0.0.1/24

R1-eth5(R4): 9.0.4.1/24

<u>AS2</u>

AS2: 12.0.0.0/8

R2-eth1(h21): 12.0.1.254/24

R2-eth2(h22): 12.0.2.254/24

R2-eth3(h23): 12.0.3.254/24

R1-eth4(R1): 9.0.0.2/24

R1-eth5(R3): 9.0.1.1/24

AS3

AS3: 13.0.0.0/8

R3-eth1(h31): 13.0.1.254/24

R3-eth2(h32): 13.0.2.254/24

R3-eth3(h33): 13.0.3.254/24

R1-eth4(R2): 9.0.1.2/24

AS4

AS1: 14.0.0.0/8 (spoofed as AS3 (13.0.0.0/8)

R4-eth1(h41): 13.0.1.254/24

R4-eth2(h42): 13.0.2.254/24

R4-eth3(h43): 13.0.3.254/24

R4-eth4(R2): 9.0.4.2/24

What was it initially not possible to reach 13.0.1.1 from AS1? How did you find out/what did you do to fix this?

- It was initially not possible to reach 13.0.1.1 from R1 AS1 because the packets sent from R1 is able to route to 13.0.1.1 but from R3, it does not know how to route back to R1 As1 due to a missing routing table entry in R3 which is supposed to tell R3 how to route packets that have destination IP address of R1 (9.0.0.0/24)
- No routing entry in R3 to route the packets to R1 but there is a routing entry in R3 to route the packets to the host H11 in As1

Verify using h11 (Figure 2) which shows that h11 in As1 is able to contact h33 in AS3 and that R3 has no problems finding the route to H11 (but not R1). Also, the command R1 ping 13.0.1.1 (also Figure 2) is

unable to receive any replies.

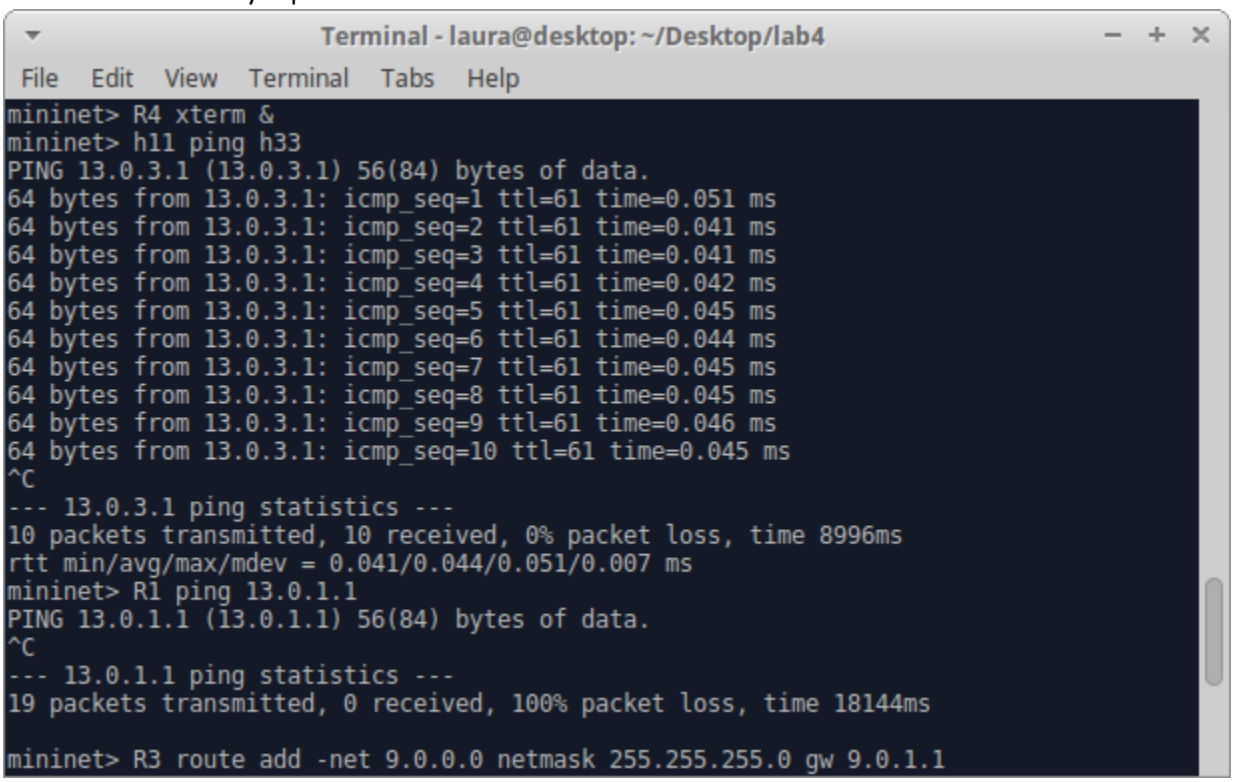

Figure 2: h11 ping h33, R1 unable to contact 13.0.1.1 at first

After configuring R3 to have a routing entry to R1 using *route* (See Figure 3), we are able to successfully ping 13.0.1.1 from R1 and get a response, meaning that the ICMP packets from R1 are now able to

recognize the way back and provide a reply

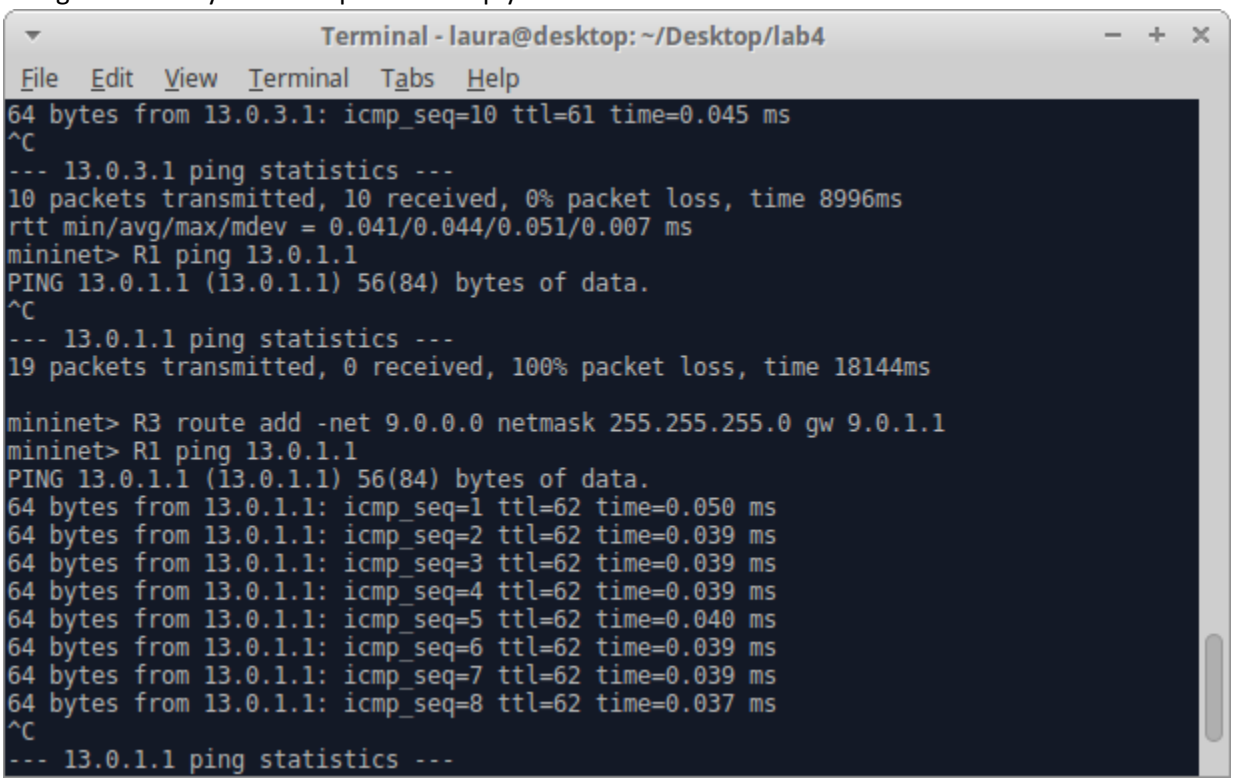

Figure 3: After configuring R3, now R1 is able to contact 13.0.1.1

Describe the BGP traffic you were able to observe during the reestablishment of routes

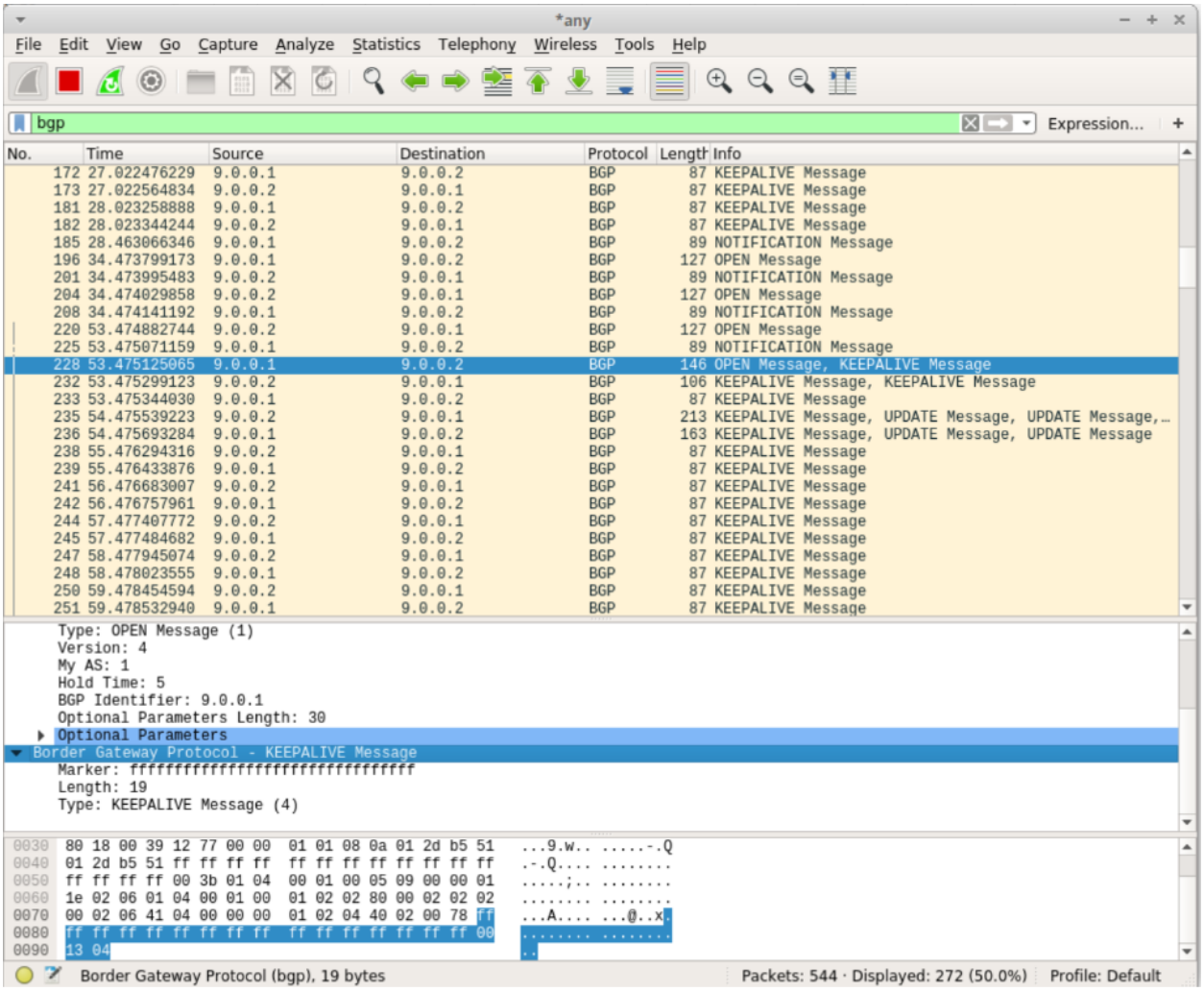

Figure 4: Wireshark packets showing the BGP messages during re-establishment

During re-establishment of routes, it can be seen that there are Notification and open messages, followed by subsequent Keep Alive messages after the routes have been re-established

Notification messages are sent by the BGP system when an error condition is detected, and after the message is sent, the BGP session and the TCP connection between the BGP systems is closed. Notification messages consist of the BGP header plus the error code and subcode, and data that describes the error.

BGP open messages are exchanged after a TCP connection is established between the two BGP systems. Once the connection is established, the two systems can exchange BGP messages and data traffic.

BGP systems exchange keepalive messages to determine whether a link or host has failed or is no longer available. Keep alive messages are exchanged often enough so that the hold timer does not expire. These messages consist only of the BGP header.

Describe in detail what happened when you started the attack on BGP

In order to start the attack, I first modified the bgpd-R4.conf file to change its network parameter from 14.0.0.0/8 to 13.0.0.0/8 (refer to Figure 5) which the latter is the webserver we want to disguise as.

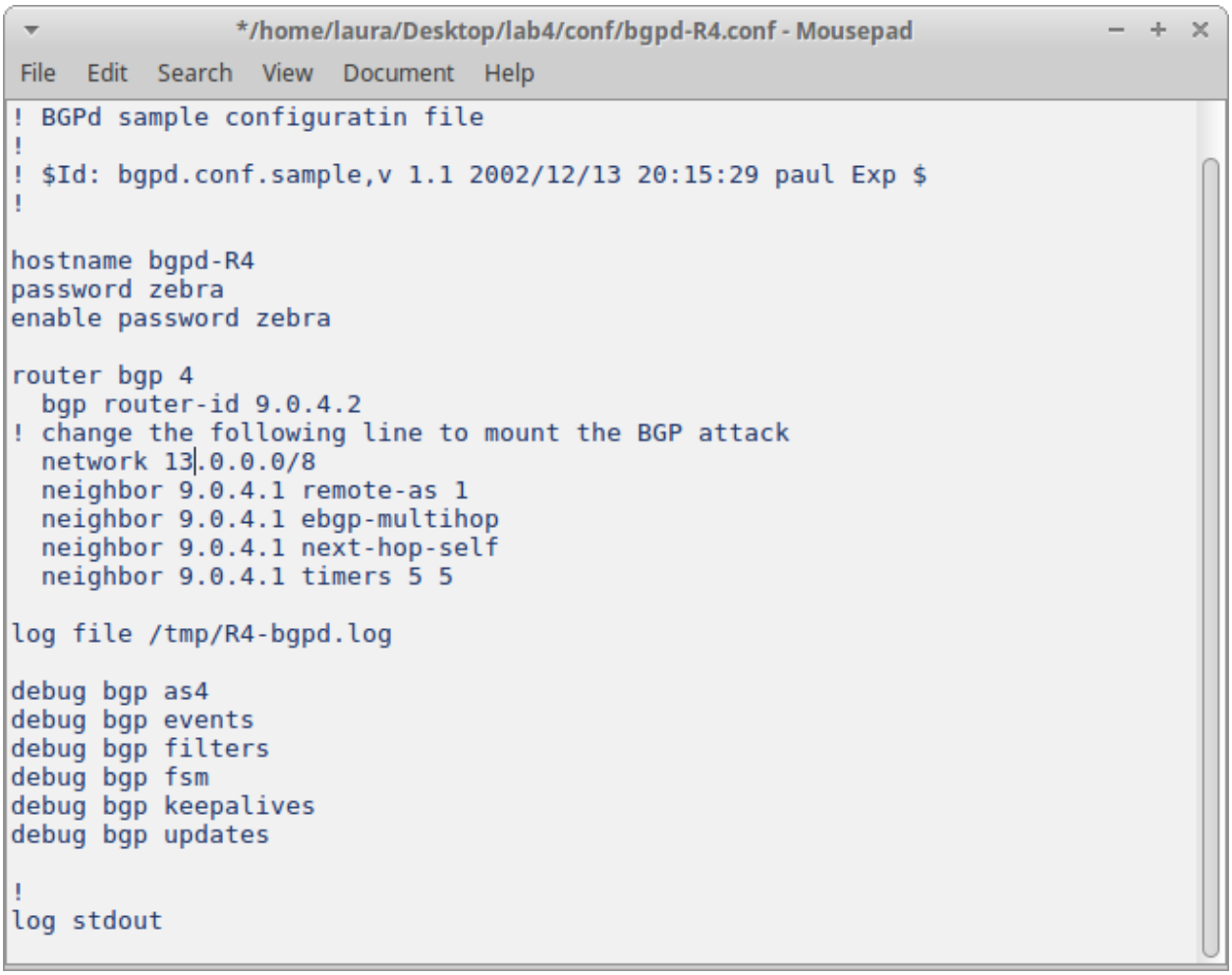

Figure 5: Modified bgpd-R4.conf file

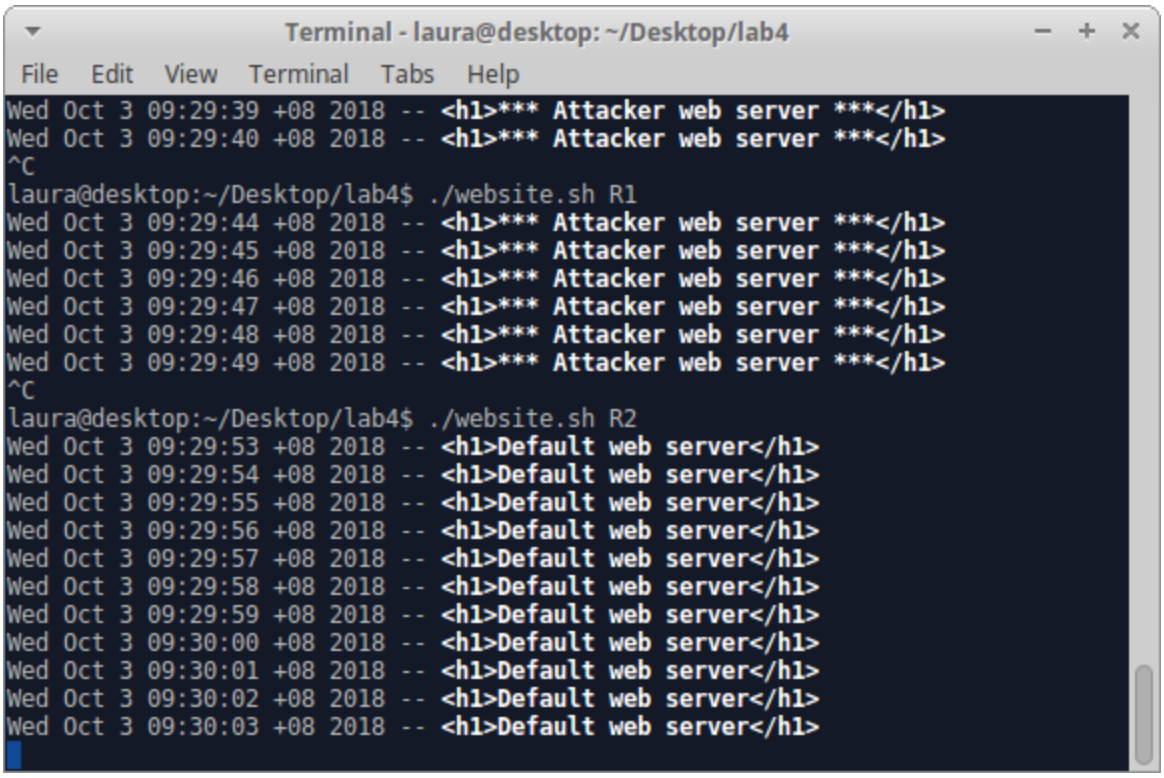

Figure 6: website.sh script running

I then ran the provided website script using *./website.sh R1,* which left it continuously contacting a webserver on 13.0.1.1 from R1. It returned 'Default web server' which is expected since I haven't started the attack yet. Then, I ran ./start_rogue.sh to launch the attack. The website results now return ***Attacker web server***' (from R4) instead of Default web server (R3) See Figure 6.

Laura Ong 1002464

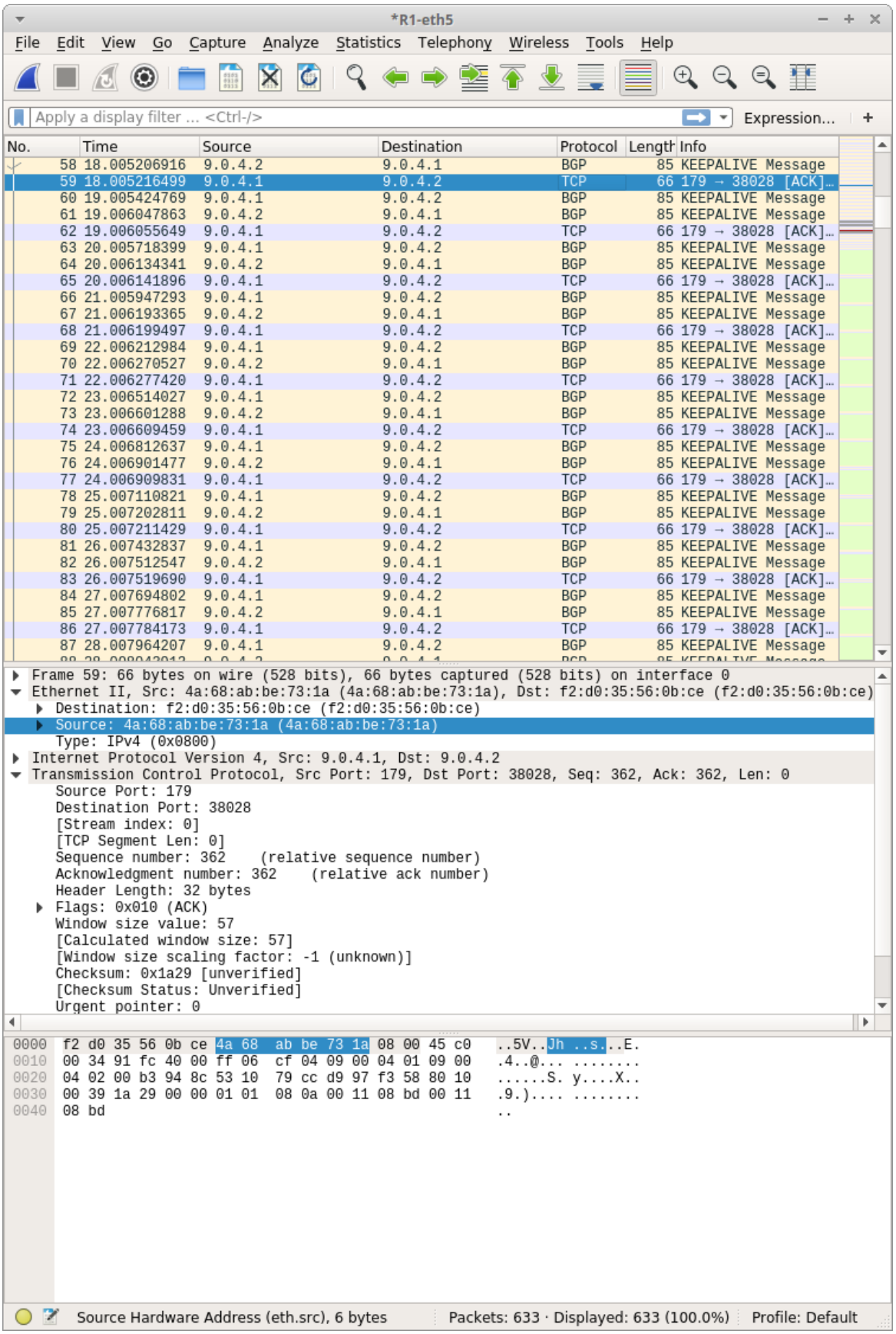

Figure 7: Wireshark packets showing default web server packets

By running Wireshark alongside the attack, I was able to capture and observe the network traffic. Before I launched the attack, I followed the TCP stream and found that the network traffic was as desired by the user, with the packets being forwarded from 9.0.0.1 at R1 to the real destination 13.0.1.1 at AS3. It can also be observed that the source MAC address is 4a:68:ab:be:73:1a. Also, I can see from the TCP stream that the message being sent back to R1 was default web server.

Laura Ong 1002464

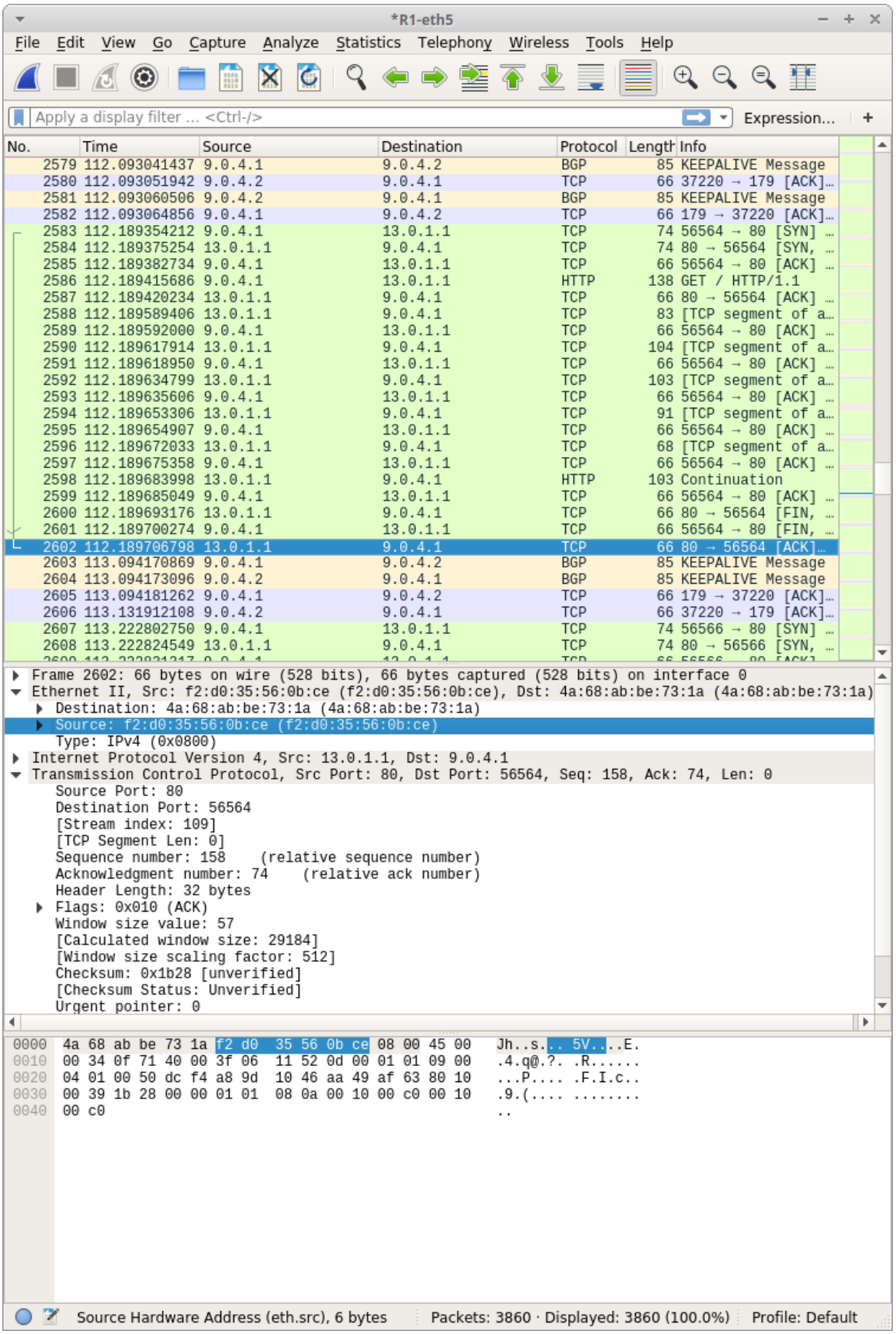

Figure 8: Wireshark packets showing attacker web server packets

After launching the attack, I followed the TCP stream and realized a few differences that confirmed the success of my attack.

The packets are now being forwarded from 9.0.4.1, which the direction is headed for the attacker AS4 to the spoofed 13.0.1.1 destination. Also, we can observe the source MAC address is now f2:d0:35:56:0b:ce, which is different from the previous legitimate MAC address of 4a:68:ab:be:73:1a. The TCP stream shows that the message being sent to R1 is now ***Attacker web server***