

Network Security – CSCI_6541_80

Namana Y Tarikere – G21372717

Homework Assignment – 5

IPSec Assignment

Setup:

- Download this file from within Ubuntu:
http://archive.ubuntu.com/ubuntu/pool/universe/i/ipsec-tools/ipsec-tools_0.8.2+20140711-10build1_amd64.deb
- You can use the command: wget
http://archive.ubuntu.com/ubuntu/pool/universe/i/ipsec-tools/ipsec-tools_0.8.2+20140711-10build1_amd64.deb
- Install it
You can use the command: `dpkg -i ipsec-tools_0.8.2+20140711-10build1_amd64.deb`
- Make sure it is installed
You can run the command: `setkey -DP`. It should return “No SPD entries”

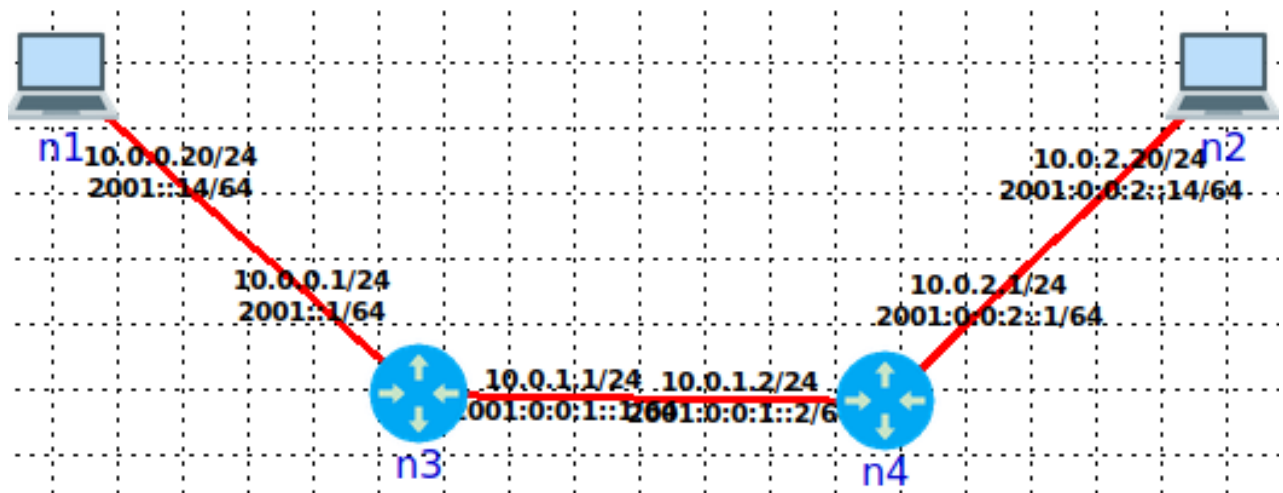


```
core@core-VM: ~  
core@core-VM:~$ wget http://archive.ubuntu.com/ubuntu/pool/universe/i/ipsec-tools/ipsec-tools_0.8.2+20140711-10build1_amd64.deb  
--2024-11-05 18:52:24-- http://archive.ubuntu.com/ubuntu/pool/universe/i/ipsec-tools/ipsec-tools_0.8.2+20140711-10build1_amd64.deb  
Resolving archive.ubuntu.com (archive.ubuntu.com)... 91.189.91.83, 185.125.190.82, 185.125.190.83, ...  
Connecting to archive.ubuntu.com (archive.ubuntu.com)[91.189.91.83]:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 62112 (61K) [application/vnd.debian.binary-package]  
Saving to: 'ipsec-tools_0.8.2+20140711-10build1_amd64.deb.1'  
ipsec-tools_0.8.2+2 100%[=====] 60.66K --KB/s in 0.1s  
2024-11-05 18:52:24 (568 KB/s) - 'ipsec-tools_0.8.2+20140711-10build1_amd64.deb.1' saved [62112/62112]  
core@core-VM:~$ sudo dpkg -i ipsec-tools_0.8.2+20140711-10build1_amd64.deb  
[sudo] password for core:  
(Reading database ... 172230 files and directories currently installed.)  
Preparing to unpack ipsec-tools_0.8.2+20140711-10build1_amd64.deb ...  
Unpacking ipsec-tools (1:0.8.2+20140711-10build1) over (1:0.8.2+20140711-10build1) ...  
Setting up ipsec-tools (1:0.8.2+20140711-10build1) ...  
update-rc.d: warning: start and stop actions are no longer supported; falling back to defaults  
Processing triggers for systemd (245.4-4ubuntu3.20) ...  
Processing triggers for man-db (2.9.1-1) ...  
core@core-VM:~$ sudo setkey -DP  
No SPD entries.  
core@core-VM:~$ s
```

- You can read about setkey here:
 - <https://manpages.debian.org/testing/ipsec-tools/setkey.8.en.html>
 - <https://www.kame.net/newsletter/19991007/>
 - <http://www.ipsec-howto.org/x304.html>
- Useful commands:
 - `setkey -F`: flushes SAD database
 - `setkey -FP`: flushes SPD database
 - `setkey -D`: dumps SAD database
 - `setkey -DP`: dumps SPD database
 - `setkey -h`: displays help menu

Assignment (10 points)

Create the topology shown below in CORE:



The setkey command takes a config file. You can run it as follows to have it read and apply the configuration in the config file:

- setkey -f my_config.file

You will create the file "my_config.file" for some of the nodes in each the questions below. The content of the config file will be different for each of the questions below and for each node. I recommend having a "n1_config.file" for node n1, n2_config.file for n2, etc.

Using the "setkey" command, create the following configurations:

1. (4 points) You will create two configs: n1_config.file to use on n1, and n2_config.file to use on n2. The configurations will create a transport association in AH mode between nodes n1 and n2.
 - 1.1. (1 point) Show the content of both config files

Content of n1_config.file in AH mode:

```
core@core-VM: ~/HW5
GNU nano 4.8 n1_config.file
#!/usr/sbin/setkey -f

# Configuration for 192.168.1.100

# Flush the SAD and SPD
flush;
spdflush;

# Attention: Use this keys only for testing purposes!
# Generate your own keys!

# AH SAs using 128 bit long keys
add 10.0.0.20 10.0.2.20 ah 0x200 -A hmac-md5
0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 10.0.2.20 10.0.0.20 ah 0x300 -A hmac-md5
0x96358c90783bbfa3d7b196ceabe0536b;

# ESP SAs using 192 bit long keys (168 + 24 parity)
# add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc
# 0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
# add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc
# 0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 10.0.0.20 10.0.2.20 any -P out ipsec
        ah/transport//require;

spdadd 10.0.2.20 10.0.0.20 any -P in ipsec
        ah/transport//require;

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell
```

Content of n2_config.file in AH mode:

```
core@core-VM: ~/HW5
GNU nano 4.8 n2_config.file
#!/usr/sbin/setkey -f

# Configuration for 192.168.1.100

# Flush the SAD and SPD
flush;
spdflush;

# Attention: Use this keys only for testing purposes!
# Generate your own keys!

# AH SAs using 128 bit long keys
add 10.0.0.20 10.0.2.20 ah 0x200 -A hmac-md5
0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 10.0.2.20 10.0.0.20 ah 0x300 -A hmac-md5
0x96358c90783bbfa3d7b196ceabe0536b;

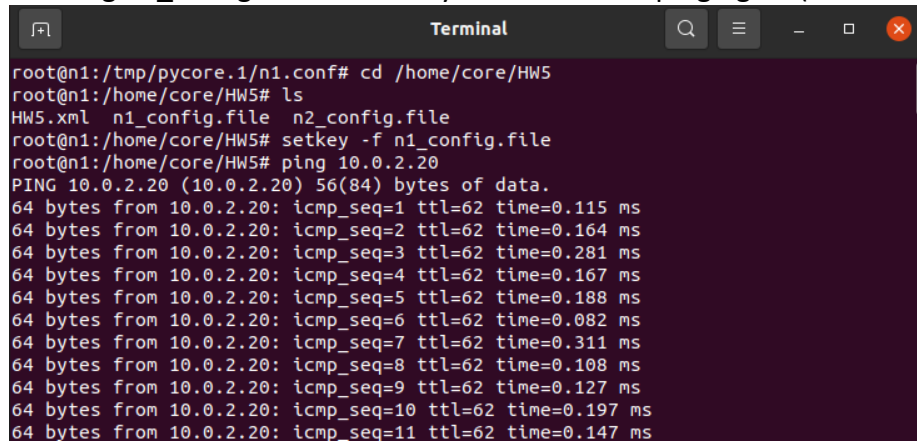
# ESP SAs using 192 bit long keys (168 + 24 parity)
# add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc
# 0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
# add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc
# 0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 10.0.0.20 10.0.2.20 any -P in ipsec
        ah/transport//require;

spdadd 10.0.2.20 10.0.0.20 any -P out ipsec
        ah/transport//require;

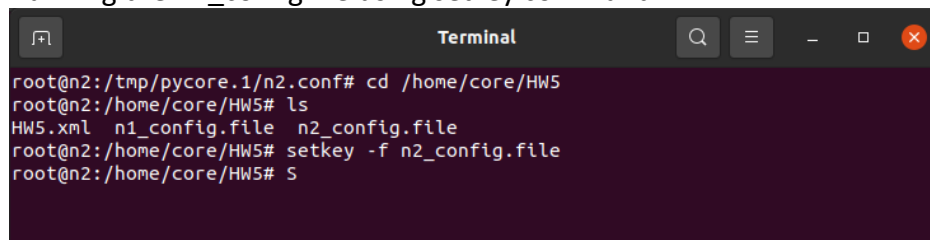
[ Read 29 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell
```

Running n1_config.file with setkey command and pinging n2 (IP: 10.0.2.20) from n1:



```
root@n1:/tmp/pycore.1/n1.conf# cd /home/core/HW5
root@n1:/home/core/HW5# ls
HW5.xml  n1_config.file  n2_config.file
root@n1:/home/core/HW5# setkey -f n1_config.file
root@n1:/home/core/HW5# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data:
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.115 ms
64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.164 ms
64 bytes from 10.0.2.20: icmp_seq=3 ttl=62 time=0.281 ms
64 bytes from 10.0.2.20: icmp_seq=4 ttl=62 time=0.167 ms
64 bytes from 10.0.2.20: icmp_seq=5 ttl=62 time=0.188 ms
64 bytes from 10.0.2.20: icmp_seq=6 ttl=62 time=0.082 ms
64 bytes from 10.0.2.20: icmp_seq=7 ttl=62 time=0.311 ms
64 bytes from 10.0.2.20: icmp_seq=8 ttl=62 time=0.108 ms
64 bytes from 10.0.2.20: icmp_seq=9 ttl=62 time=0.127 ms
64 bytes from 10.0.2.20: icmp_seq=10 ttl=62 time=0.197 ms
64 bytes from 10.0.2.20: icmp_seq=11 ttl=62 time=0.147 ms
```

Running the n2_config.file using setkey command:



```
root@n2:/tmp/pycore.1/n2.conf# cd /home/core/HW5
root@n2:/home/core/HW5# ls
HW5.xml  n1_config.file  n2_config.file
root@n2:/home/core/HW5# setkey -f n2_config.file
root@n2:/home/core/HW5# S
```

1.2. (1 points) Show ping command from n1 to n2 successfully sending and receiving

1.2.1. Show a Wireshark screenshot of ICMP echo request and echo response authenticated at the 10.0.0.1 interface on n3.

ICMP echo request and reply with authentication header:

The image shows a Wireshark packet capture window titled "Capturing from veth3.0.1". The packet list pane displays 113 packets. The packet details pane shows the structure of packet 48, which is an ICMP Echo (ping) request. The packet structure is as follows:

- Frame 48: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface veth3.0.1, id 0
- Ethernet II, Src: 00:00:00:aa:00:01 (00:00:00:aa:00:01), Dst: 00:00:00:aa:00:00 (00:00:00:aa:00:00)
- Internet Protocol Version 4, Src: 10.0.2.20, Dst: 10.0.0.20
- Authentication Header
 - Next header: ICMP (1)
 - Length: 4 (24 bytes)
 - Reserved: 0000
 - AH SPI: 0x00000300
 - AH Sequence: 6
 - AH ICV: 27bc9d3623f112f426b468d9
- Internet Control Message Protocol

The packet bytes pane shows the raw data of the packet, with the SPI value highlighted in red in the original image.

1.2.2. Highlight the SPI value in the header in red.

Authentication Header SPI highlighted in red:

The image shows a Wireshark packet details pane for packet 48. The Authentication Header section is expanded, and the SPI value is highlighted in red. The packet structure is as follows:

- Frame 48: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface veth3.0.1, id 0
- Ethernet II, Src: 00:00:00:aa:00:01 (00:00:00:aa:00:01), Dst: 00:00:00:aa:00:00 (00:00:00:aa:00:00)
- Internet Protocol Version 4, Src: 10.0.2.20, Dst: 10.0.0.20
- Authentication Header
 - Next header: ICMP (1)
 - Length: 4 (24 bytes)
 - Reserved: 0000
 - AH SPI: 0x00000300
 - AH Sequence: 6
 - AH ICV: 27bc9d3623f112f426b468d9
- Internet Control Message Protocol

The packet bytes pane shows the raw data of the packet, with the SPI value highlighted in red in the original image.

1.3. Redo 1.2 but using ESP mode

Content of n1_config.file in ESP mode:

```
core@core-VM: ~/HW5
GNU nano 4.8 n1_config.file
#!/usr/sbin/setkey -f

# Configuration for 192.168.1.100

# Flush the SAD and SPD
flush;
spdflush;

# Attention: Use this keys only for testing purposes!
# Generate your own keys!

# AH SAs using 128 bit long keys
#add 10.0.0.20 10.0.2.20 ah 0x200 -A hmac-md5
#0xc0291ff014dccdd03874d9e8e4cdf3e6;
#add 10.0.2.20 10.0.0.20 ah 0x300 -A hmac-md5
#0x96358c90783bbfa3d7b196ceabe0536b;

# ESP SAs using 192 bit long keys (168 + 24 parity)
add 10.0.0.20 10.0.2.20 esp 0x201 -E 3des-cbc
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
add 10.0.2.20 10.0.0.20 esp 0x301 -E 3des-cbc
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 10.0.0.20 10.0.2.20 any -P out ipsec
        esp/transport//require;

spdadd 10.0.2.20 10.0.0.20 any -P in ipsec
        esp/transport//require;

^G Get Help      ^O Write Out    [ Wrote 29 lines ]
^X Exit          ^R Read File    ^W Where Is
                  ^\ Replace      ^K Cut Text
                  ^U Paste Text   ^J Justify
                  ^T To Spell
```

Content of n2_config.file in ESP mode:

```
core@core-VM: ~/HW5
GNU nano 4.8 n2_config.file Modified
#!/usr/sbin/setkey -f

# Configuration for 192.168.1.100

# Flush the SAD and SPD
flush;
spdflush;

# Attention: Use this keys only for testing purposes!
# Generate your own keys!

# AH SAs using 128 bit long keys
#add 10.0.0.20 10.0.2.20 ah 0x200 -A hmac-md5
#0xc0291ff014dccdd03874d9e8e4cdf3e6;
#add 10.0.2.20 10.0.0.20 ah 0x300 -A hmac-md5
#0x96358c90783bbfa3d7b196ceabe0536b;

# ESP SAs using 192 bit long keys (168 + 24 parity)
add 10.0.0.20 10.0.2.20 esp 0x201 -E 3des-cbc
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
add 10.0.2.20 10.0.0.20 esp 0x301 -E 3des-cbc
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 10.0.0.20 10.0.2.20 any -P in ipsec
        esp/transport//require;

spdadd 10.0.2.20 10.0.0.20 any -P out ipsec
        esp/transport//require;

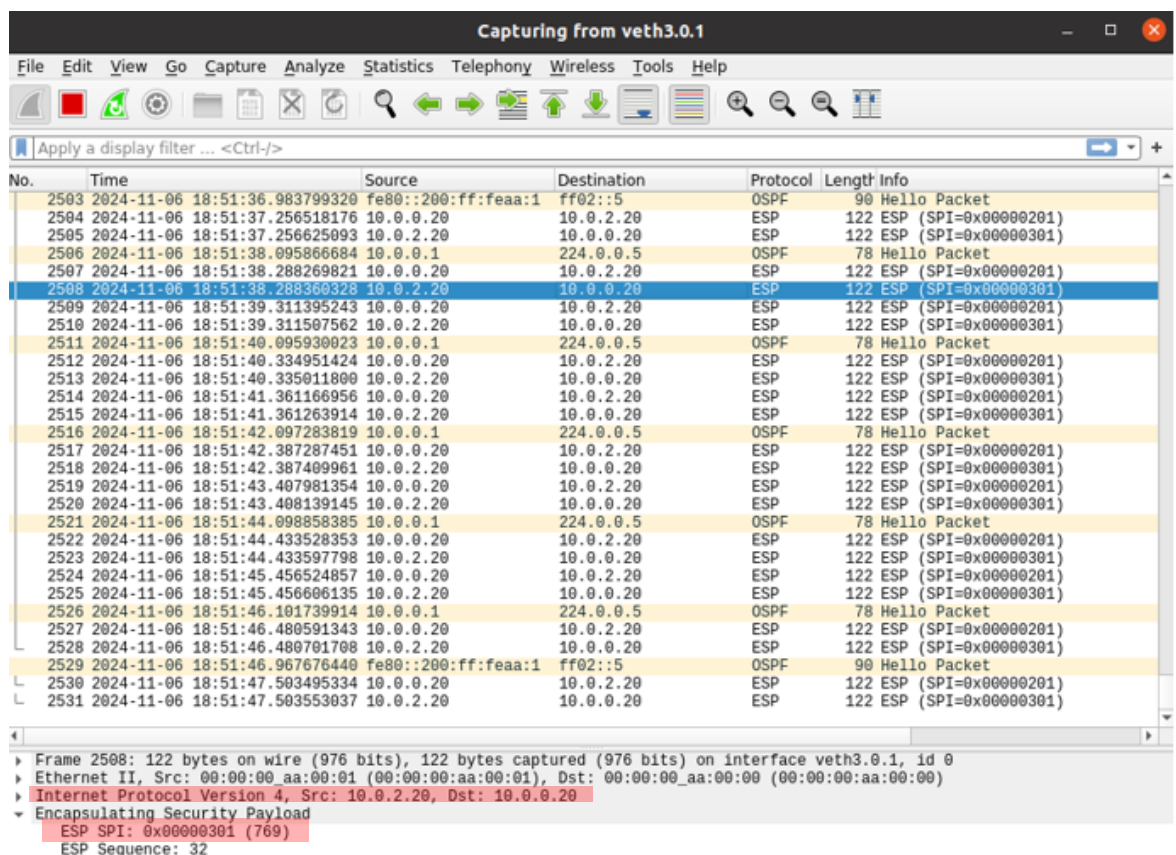
File Name to Write: n2_config.file
^G Get Help      ^M-D DOS Format  ^M-A Append      ^M-B Backup File
^C Cancel        ^M-M Mac Format  ^M-P Prepend     ^T To Files
```

Running n1_config.file and n2_config.file using the setkey command and pinging n2 (IP: 10.0.2.20) from n1:

```
Terminal
root@n1:/tmp/pycore.1/n1.conf# cd /home/core/HW5
root@n1:/home/core/HW5# ls
HW5.xml  n1_config.file  n2_config.file
root@n1:/home/core/HW5# setkey -f n1_config.file
root@n1:/home/core/HW5# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=0.115 ms
64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=0.164 ms
64 bytes from 10.0.2.20: icmp_seq=3 ttl=62 time=0.281 ms
64 bytes from 10.0.2.20: icmp_seq=4 ttl=62 time=0.167 ms
64 bytes from 10.0.2.20: icmp_seq=5 ttl=62 time=0.188 ms
64 bytes from 10.0.2.20: icmp_seq=6 ttl=62 time=0.082 ms
64 bytes from 10.0.2.20: icmp_seq=7 ttl=62 time=0.311 ms
64 bytes from 10.0.2.20: icmp_seq=8 ttl=62 time=0.108 ms
64 bytes from 10.0.2.20: icmp_seq=9 ttl=62 time=0.127 ms

Terminal
root@n2:/tmp/pycore.1/n2.conf# cd /home/core/HW5
root@n2:/home/core/HW5# ls
HW5.xml  n1_config.file  n2_config.file
root@n2:/home/core/HW5# setkey -f n2_config.file
root@n2:/home/core/HW5#
```

ESP with ESP header is captured in Wireshark as shown below:



ESP SPI Header value highlighted in red:

```
Frame 2508: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface veth3.0.1, id 0
Ethernet II, Src: 00:00:00_aa:00:01 (00:00:00:aa:00:01), Dst: 00:00:00_aa:00:00 (00:00:00:aa:00:00)
Internet Protocol Version 4, Src: 10.0.2.20, Dst: 10.0.0.20
Encapsulating Security Payload
  ESP SPI: 0x00000301 (769)
  ESP Sequence: 32
```

1.4. Flush the SPD and SAD at nodes n1 and n2 (look at useful commands above)

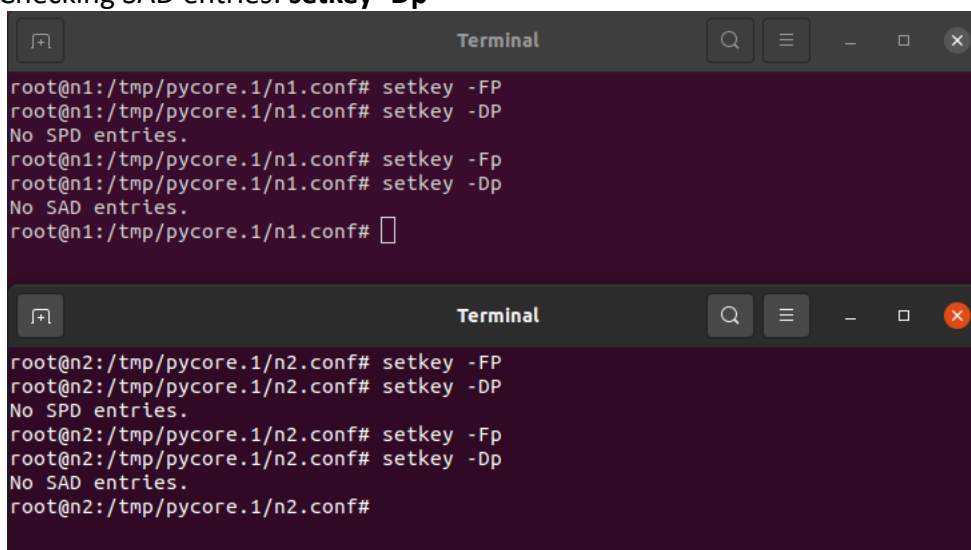
Commands for flushing SPD and SAD from both n1 and n2 nodes:

Flushing SPD: **setkey -FP**

Checking SPD entries: **setkey -DP**

Flushing SAD: **setkey -Fp**

Checking SAD entries: **setkey -Dp**



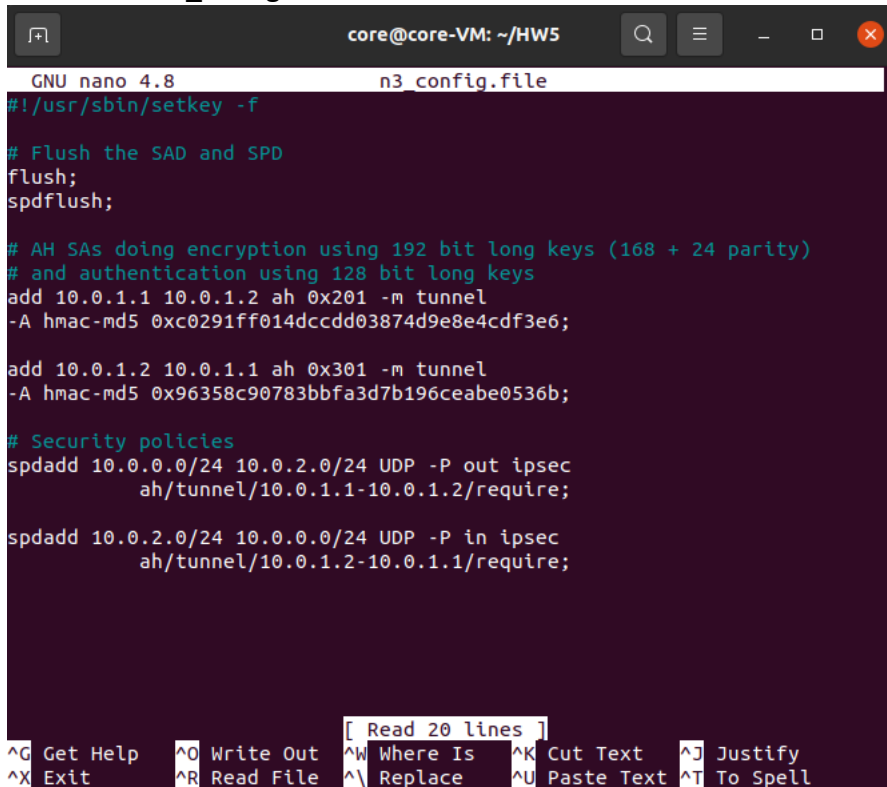
```
Terminal
root@n1:/tmp/pycore.1/n1.conf# setkey -FP
root@n1:/tmp/pycore.1/n1.conf# setkey -DP
No SPD entries.
root@n1:/tmp/pycore.1/n1.conf# setkey -Fp
root@n1:/tmp/pycore.1/n1.conf# setkey -Dp
No SAD entries.
root@n1:/tmp/pycore.1/n1.conf#

Terminal
root@n2:/tmp/pycore.1/n2.conf# setkey -FP
root@n2:/tmp/pycore.1/n2.conf# setkey -DP
No SPD entries.
root@n2:/tmp/pycore.1/n2.conf# setkey -Fp
root@n2:/tmp/pycore.1/n2.conf# setkey -Dp
No SAD entries.
root@n2:/tmp/pycore.1/n2.conf#
```

2. (4 points) You will create two configs: n3_config.file to use on n3, and n4_config.file to use on n4. The config files will create a tunnel association in AH mode between nodes n3 and n4. At n3, tunnel UDP traffic in 10.0.0.0/24 subnet. At n4, tunnel UDP traffic in 10.0.2.0/24 subnet.

2.1. (1 points) Show the content of both config files

Content of n3_config.file in AH mode with UDP traffic:



The screenshot shows a terminal window titled 'core@core-VM: ~/HW5' with a search bar and window controls. The terminal is running GNU nano 4.8 editing 'n3_config.file'. The file content is as follows:

```
#!/usr/sbin/setkey -f

# Flush the SAD and SPD
flush;
spdflush;

# AH SAs doing encryption using 192 bit long keys (168 + 24 parity)
# and authentication using 128 bit long keys
add 10.0.1.1 10.0.1.2 ah 0x201 -m tunnel
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

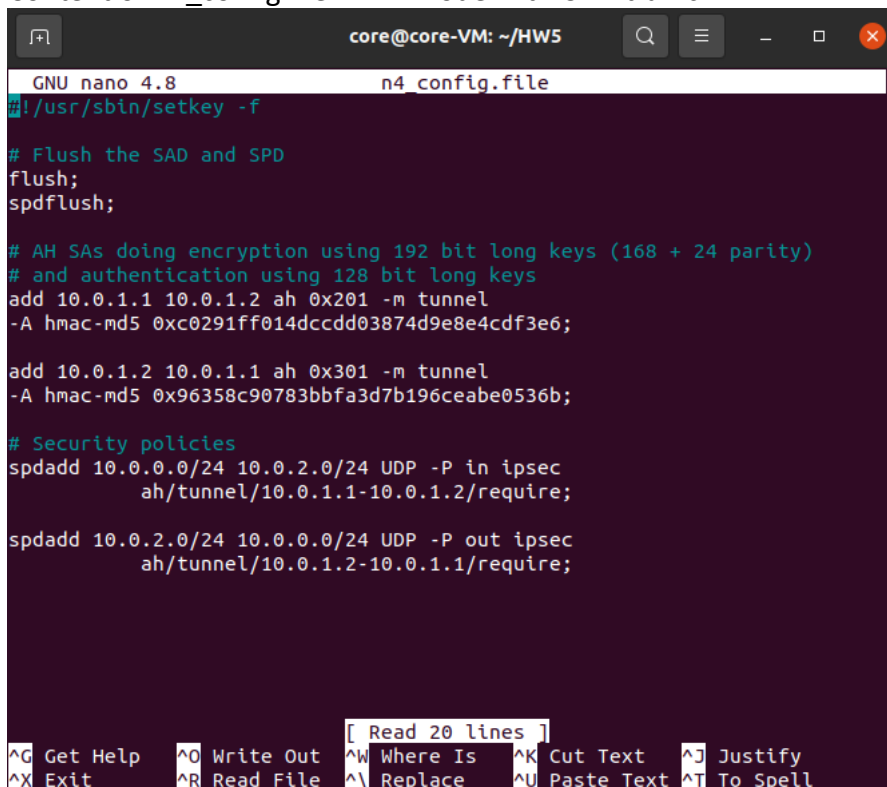
add 10.0.1.2 10.0.1.1 ah 0x301 -m tunnel
-A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;

# Security policies
spdadd 10.0.0.0/24 10.0.2.0/24 UDP -P out ipsec
        ah/tunnel/10.0.1.1-10.0.1.2/require;

spdadd 10.0.2.0/24 10.0.0.0/24 UDP -P in ipsec
        ah/tunnel/10.0.1.2-10.0.1.1/require;
```

The bottom of the terminal shows the nano editor's command palette with options: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^X Exit, ^R Read File, ^_ Replace, ^U Paste Text, and ^T To Spell. A '[Read 20 lines]' indicator is also visible.

Content of n4_config.file in AH mode with UDP traffic:



The screenshot shows a terminal window titled 'core@core-VM: ~/HW5' with a search bar and window controls. The terminal is running GNU nano 4.8 editing 'n4_config.file'. The file content is as follows:

```
#!/usr/sbin/setkey -f

# Flush the SAD and SPD
flush;
spdflush;

# AH SAs doing encryption using 192 bit long keys (168 + 24 parity)
# and authentication using 128 bit long keys
add 10.0.1.1 10.0.1.2 ah 0x201 -m tunnel
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

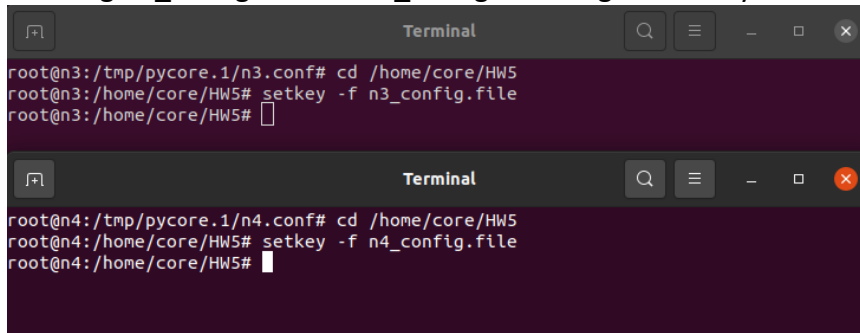
add 10.0.1.2 10.0.1.1 ah 0x301 -m tunnel
-A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;

# Security policies
spdadd 10.0.0.0/24 10.0.2.0/24 UDP -P in ipsec
        ah/tunnel/10.0.1.1-10.0.1.2/require;

spdadd 10.0.2.0/24 10.0.0.0/24 UDP -P out ipsec
        ah/tunnel/10.0.1.2-10.0.1.1/require;
```

The bottom of the terminal shows the nano editor's command palette with options: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^X Exit, ^R Read File, ^_ Replace, ^U Paste Text, and ^T To Spell. A '[Read 20 lines]' indicator is also visible.

Running n3_config.file and n4_config.file using the setkey command:



```
Terminal
root@n3:/tmp/pycore.1/n3.conf# cd /home/core/HW5
root@n3:/home/core/HW5# setkey -f n3_config.file
root@n3:/home/core/HW5#

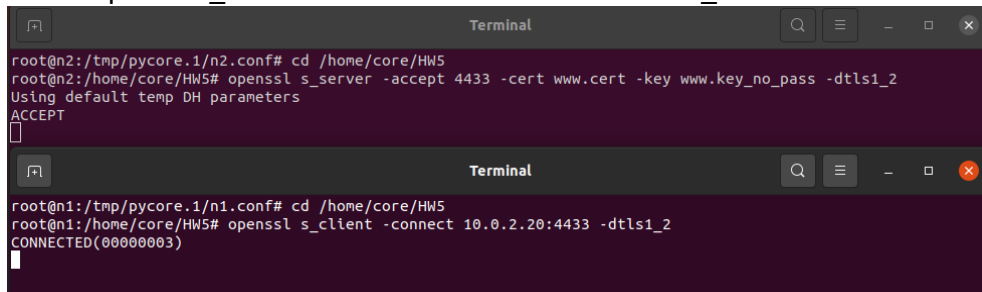
Terminal
root@n4:/tmp/pycore.1/n4.conf# cd /home/core/HW5
root@n4:/home/core/HW5# setkey -f n4_config.file
root@n4:/home/core/HW5#
```

2.2. (1 points) Do a DTLS session from n1 to n2 successfully

Running a DTLS session from n1 (client) to n2 (server) using the following commands:

Server: openssl s_server -accept 4433 cert www.cert -key www.key.no_pass -dtls1_2

Client: openssl s_client -connect 10.0.2.20:4433 -dtls1_2

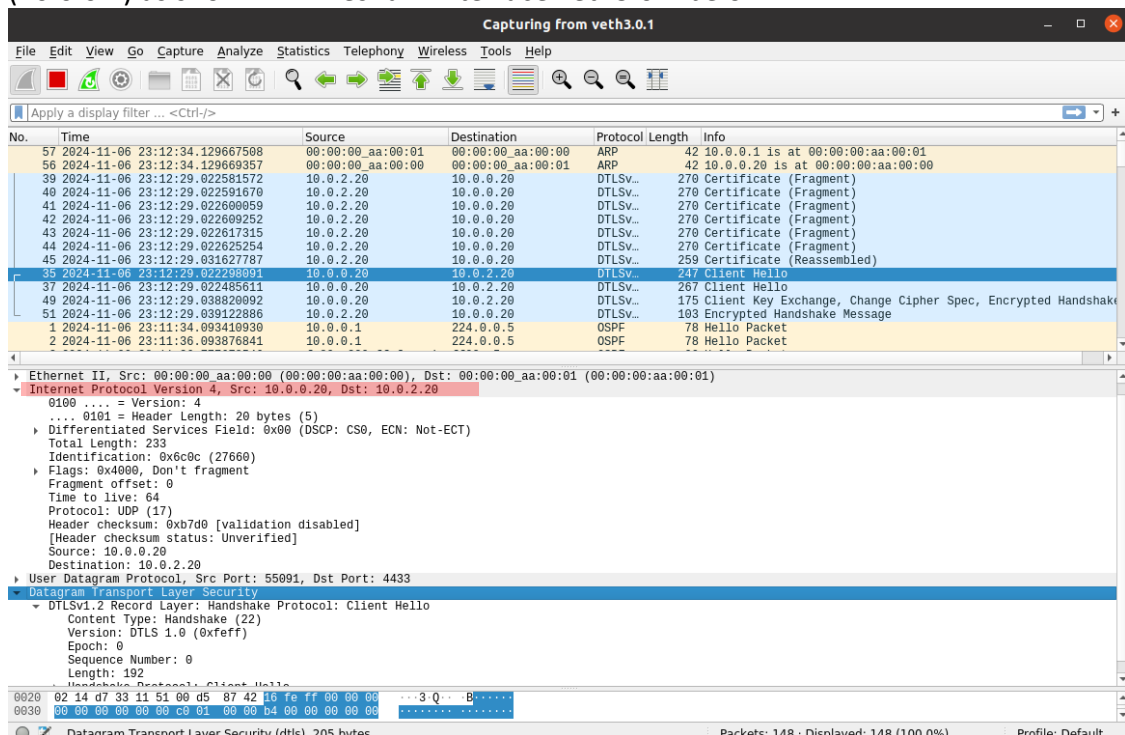


```
Terminal
root@n2:/tmp/pycore.1/n2.conf# cd /home/core/HW5
root@n2:/home/core/HW5# openssl s_server -accept 4433 -cert www.cert -key www.key.no_pass -dtls1_2
Using default temp DH parameters
ACCEPT
^C

Terminal
root@n1:/tmp/pycore.1/n1.conf# cd /home/core/HW5
root@n1:/home/core/HW5# openssl s_client -connect 10.0.2.20:4433 -dtls1_2
CONNECTED(00000003)
```

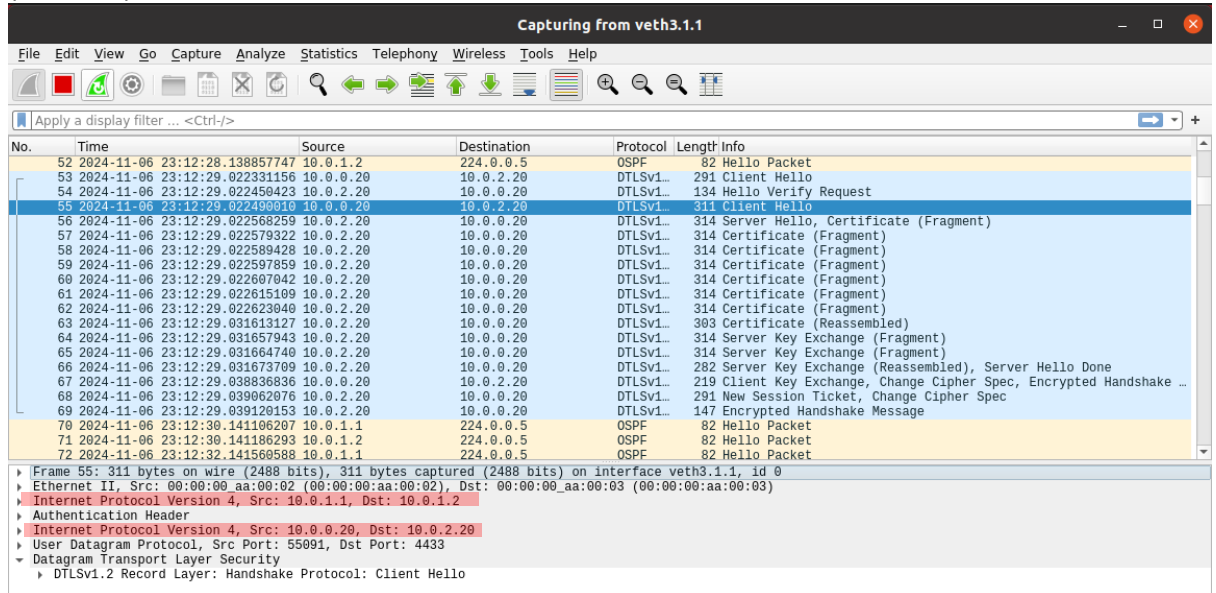
2.2.1. Show a Wireshark screenshot of Client Hello packets entering the 10.0.0.1 interface on n3.

DTLS Client Hello packets moving from n1 (10.0.0.20) to n2 (10.0.2.20) via n3 (10.0.0.1) as shown in wireshark interface veth3.0.1 below:



2.2.2. Show a Wireshark screenshot of Client Hello packets leaving the 10.0.1.1 interface on n3.

DTLS Client Hello packets moving from n1 (10.0.0.20) to n2 (10.0.2.20) via n3 (10.0.1.1) as shown in wireshark interface veth3.1.1 below:



2.2.2.1. Show inner and outer IP headers addresses

Outer IP header address:

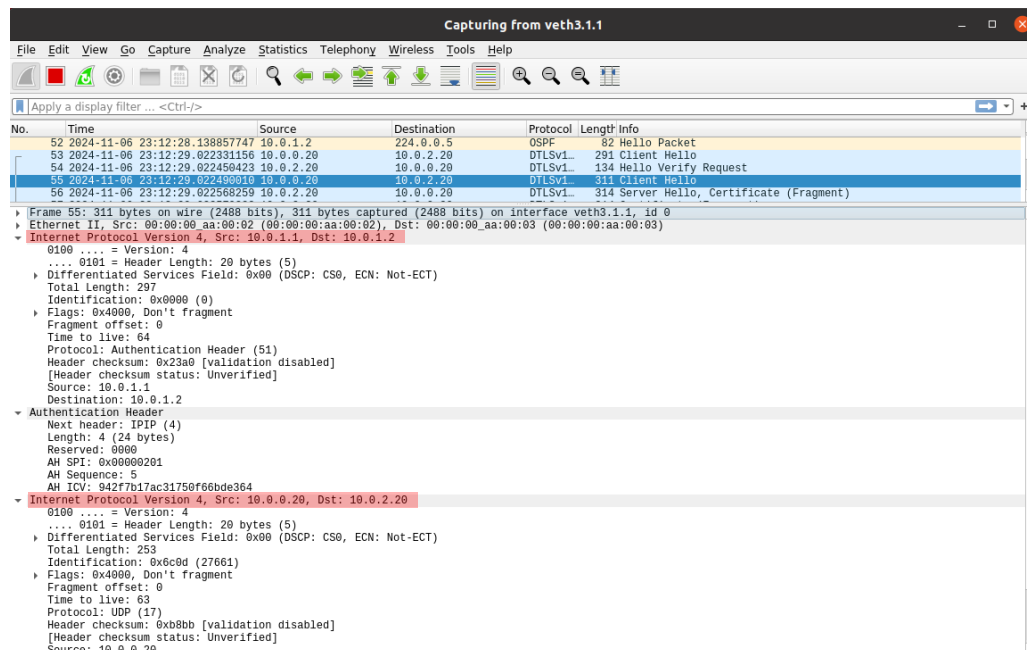
Source: 10.0.1.1

Destination: 10.0.1.2

Inner IP header address:

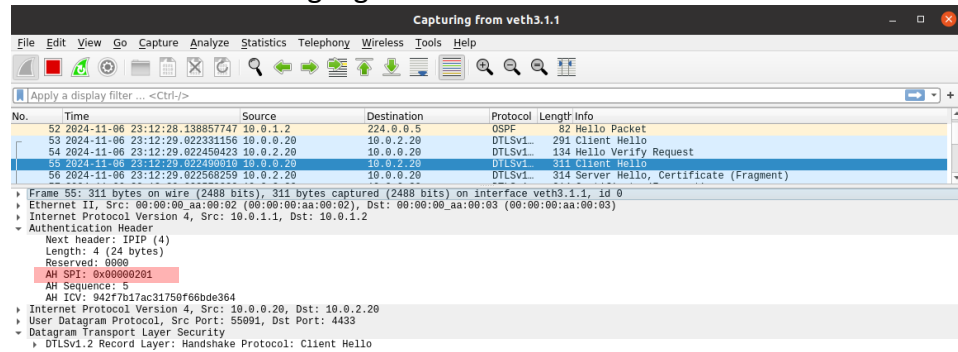
Source: 10.0.0.20

Destination: 10.0.2.20



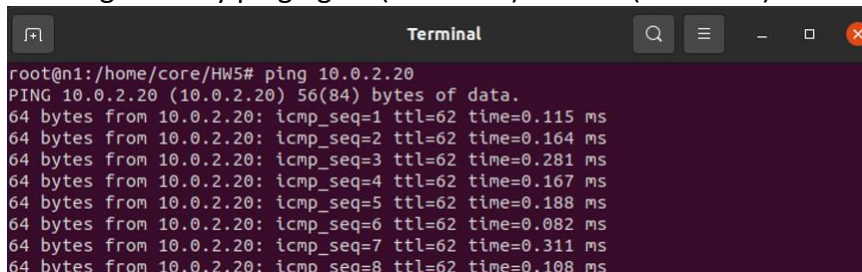
2.2.2.2. Highlight the SPI value in the header in red.

AH SPI Header value highlighted in red:

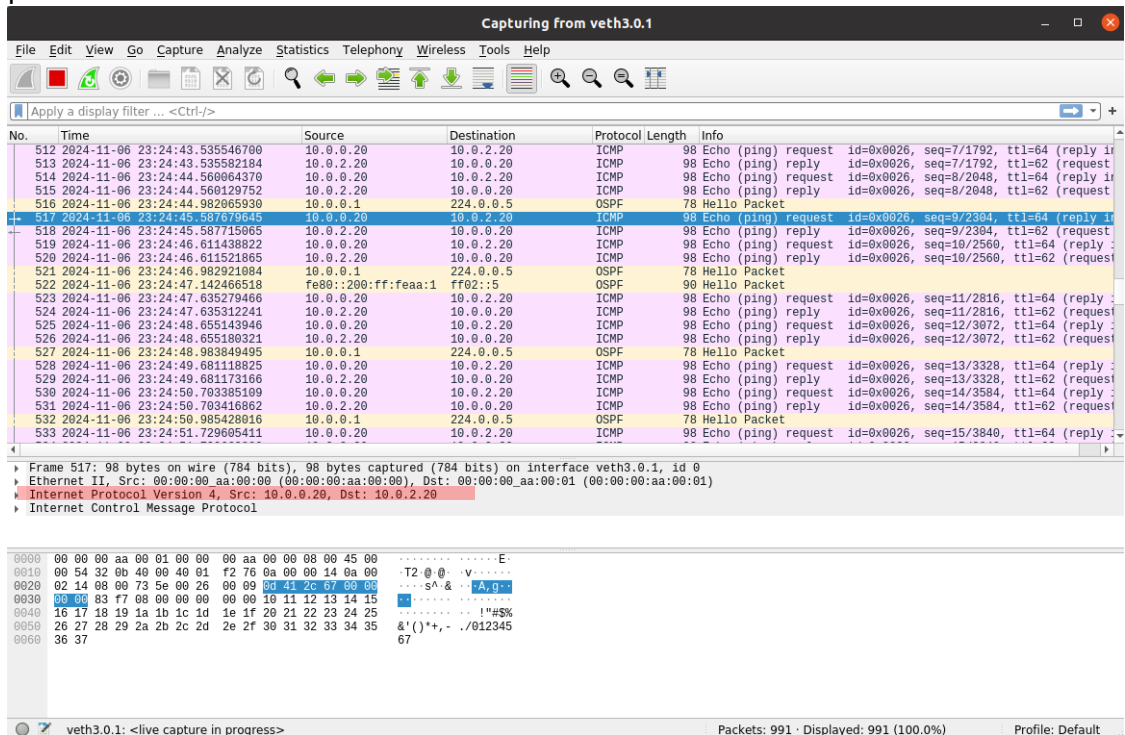


2.3. Repeat 2.2 using ICMP, you should not see any AH headers

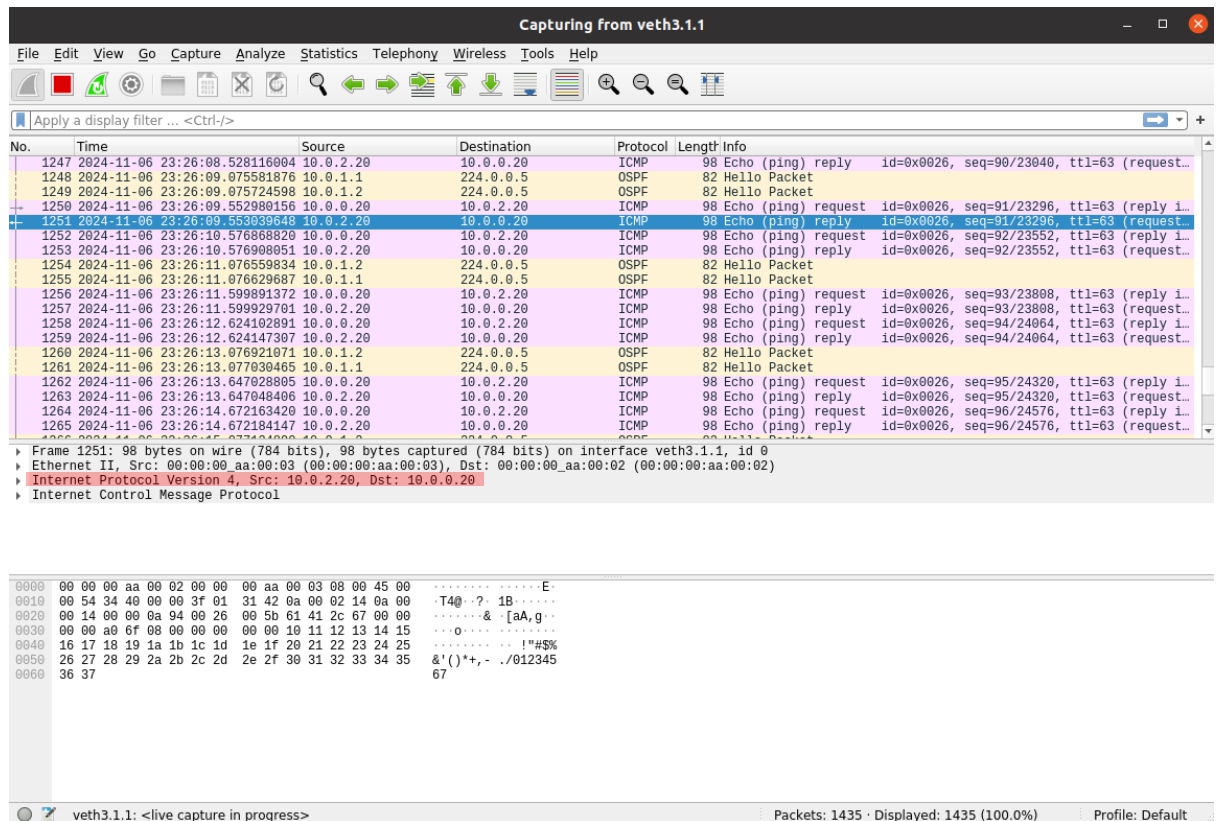
Running ICMP by pinging n2 (10.0.2.20) from n1(10.0.0.20):



You can see that there are no authentication headers (AH) in the ICMP echo request packets from wireshark veth3.0.1 interface as shown below:



You can see that there are no authentication headers (AH) in the ICMP echo reply packets from wireshark veth 3.1.1 interface as shown below:



2.4. Flush the SPD and SAD at nodes n1 and n2 (look at useful commands above)

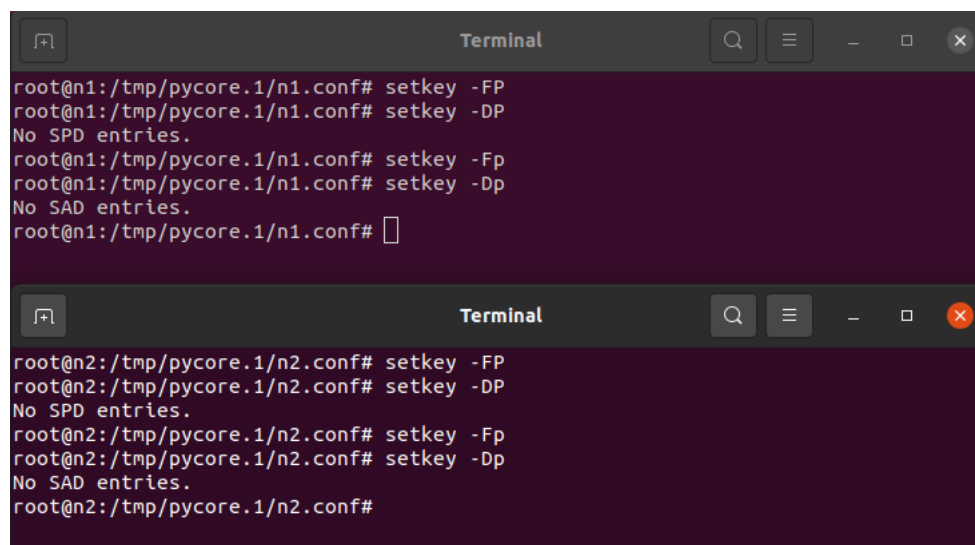
Commands for flushing SPD and SAD from both n1 and n2 nodes:

Flushing SPD: **setkey -FP**

Checking SPD entries: **setkey -DP**

Flushing SAD: **setkey -Fp**

Checking SAD entries: **setkey -Dp**

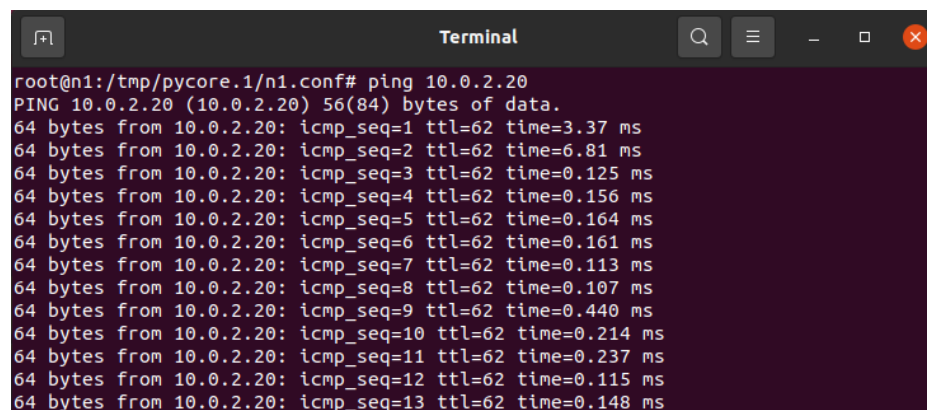


3. (2 points) Combine (1.2) in AH mode and (2.2) in AH mode but apply for any protocol instead of just UDP.

In n1_config.file and n2_config.file, change the content back into AH Security Association and Security policies as shown in step 1.1 above. In n3_config.file and n4_config.file, change the security policies traffic from **UDP to any** as shown in step 1.1

3.1. Ping from n1 to n2

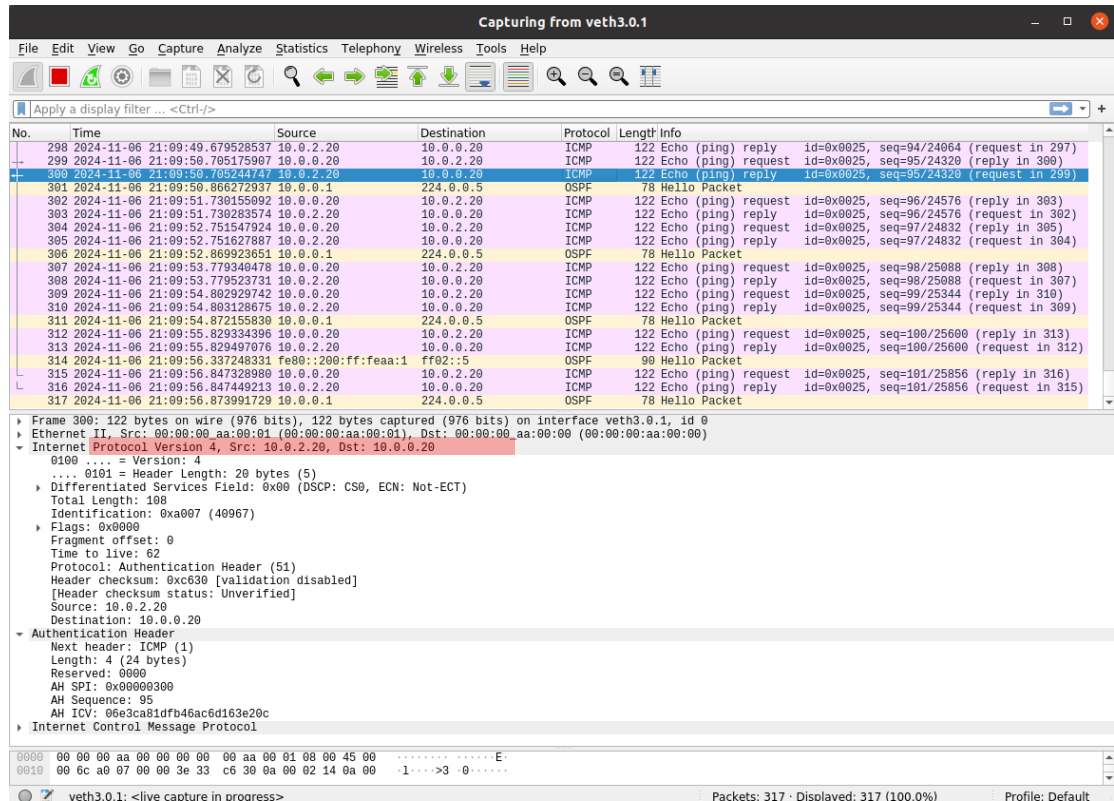
Pinging n2 (10.0.2.20) from n1(10.0.0.20) as shown:

A terminal window titled "Terminal" with a dark background and light text. It shows the command "root@n1:/tmp/pycore.1/n1.conf# ping 10.0.2.20" and its output. The output indicates that 13 ping requests were sent to 10.0.2.20, each receiving a 64-byte response from 10.0.2.20 with an ICMP sequence number from 1 to 13 and a TTL of 62. The response times vary, with the first being 3.37 ms and the last being 0.148 ms.

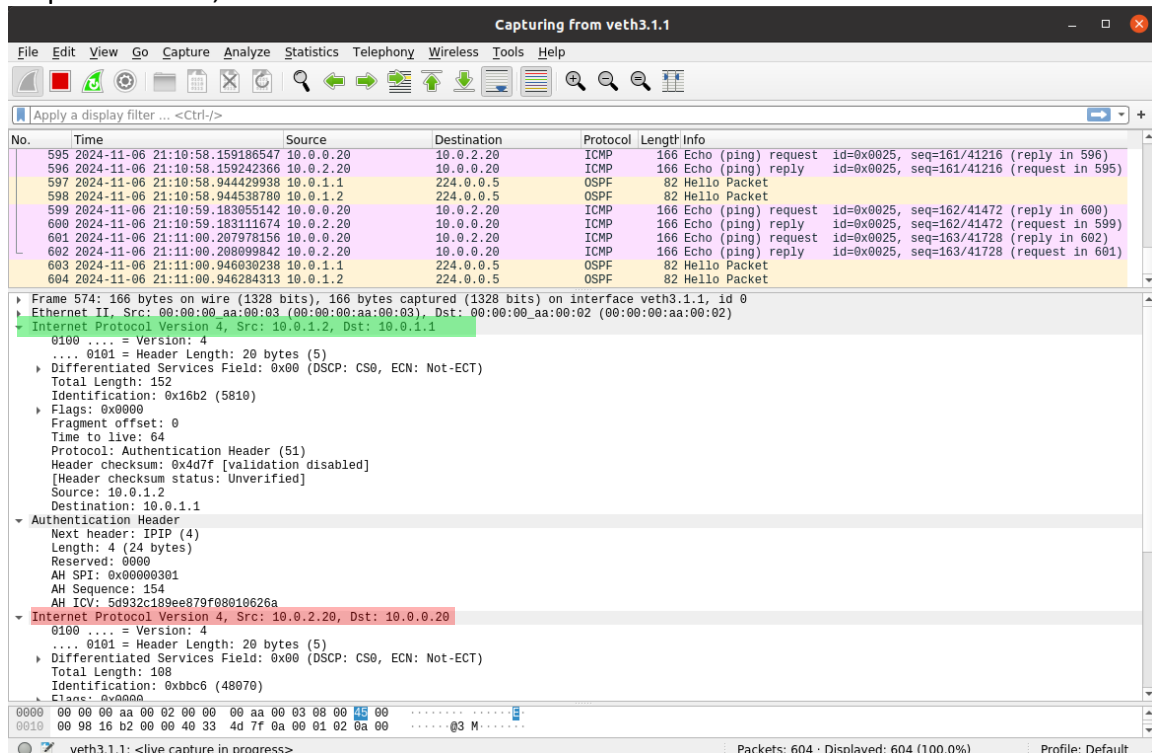
```
root@n1:/tmp/pycore.1/n1.conf# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data.
64 bytes from 10.0.2.20: icmp_seq=1 ttl=62 time=3.37 ms
64 bytes from 10.0.2.20: icmp_seq=2 ttl=62 time=6.81 ms
64 bytes from 10.0.2.20: icmp_seq=3 ttl=62 time=0.125 ms
64 bytes from 10.0.2.20: icmp_seq=4 ttl=62 time=0.156 ms
64 bytes from 10.0.2.20: icmp_seq=5 ttl=62 time=0.164 ms
64 bytes from 10.0.2.20: icmp_seq=6 ttl=62 time=0.161 ms
64 bytes from 10.0.2.20: icmp_seq=7 ttl=62 time=0.113 ms
64 bytes from 10.0.2.20: icmp_seq=8 ttl=62 time=0.107 ms
64 bytes from 10.0.2.20: icmp_seq=9 ttl=62 time=0.440 ms
64 bytes from 10.0.2.20: icmp_seq=10 ttl=62 time=0.214 ms
64 bytes from 10.0.2.20: icmp_seq=11 ttl=62 time=0.237 ms
64 bytes from 10.0.2.20: icmp_seq=12 ttl=62 time=0.115 ms
64 bytes from 10.0.2.20: icmp_seq=13 ttl=62 time=0.148 ms
```

- 3.1.1. Show a Wireshark screenshot of ICMP packets on interfaces between n1-n3, n3-n4, and n4-n2 in that order. Highlight the inner and outer IP headers addresses on each.

ICMP packets on veth3.0.1 interface of wireshark between n1-n3 is shown below. For n1-n3, Inner IP address is highlighted in red, and no outer IP address is present since it follows AH transport mode.



ICMP packets on veth3.1.1 interface of wireshark between n3-n4 is shown below. For n3-n4, Inner IP address is highlighted in red, and outer IP address is highlighted in green. Outer IP address is present here, since n3-n4 follows AH tunnel mode.



ICMP packets on veth4.1.1 interface of wireshark between n4-n2 is shown below. For n4-n2, Inner IP address is highlighted in red, and no outer IP address is present since it follows AH transport mode.

Capturing from veth4.1.1

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
577	2024-11-06 21:11:43.086948471	10.0.2.20	10.0.2.20	ICMP	122	Echo (ping) reply id=0x0025, seq=205/52480 (request in 576)
578	2024-11-06 21:11:44.114320551	10.0.0.20	10.0.2.20	ICMP	122	Echo (ping) request id=0x0025, seq=206/52736 (reply in 579)
579	2024-11-06 21:11:44.114346058	10.0.2.20	10.0.0.20	ICMP	122	Echo (ping) reply id=0x0025, seq=206/52736 (request in 578)
580	2024-11-06 21:11:44.990471888	10.0.2.1	224.0.0.5	OSPF	78	Hello Packet
581	2024-11-06 21:11:45.116618482	10.0.0.20	10.0.2.20	ICMP	122	Echo (ping) request id=0x0025, seq=207/52992 (reply in 582)
582	2024-11-06 21:11:45.116697218	10.0.2.20	10.0.0.20	ICMP	122	Echo (ping) reply id=0x0025, seq=207/52992 (request in 581)
583	2024-11-06 21:11:46.131291346	10.0.0.20	10.0.2.20	ICMP	122	Echo (ping) request id=0x0025, seq=208/53248 (reply in 584)
584	2024-11-06 21:11:46.131318299	10.0.2.20	10.0.0.20	ICMP	122	Echo (ping) reply id=0x0025, seq=208/53248 (request in 583)
585	2024-11-06 21:11:46.321150196	fe80::200:ff:feaa:4	ff02::5	OSPF	90	Hello Packet
586	2024-11-06 21:11:46.991318305	10.0.2.1	224.0.0.5	OSPF	78	Hello Packet
587	2024-11-06 21:11:47.153981925	10.0.0.20	10.0.2.20	ICMP	122	Echo (ping) request id=0x0025, seq=209/53504 (reply in 588)
588	2024-11-06 21:11:47.154063392	10.0.2.20	10.0.0.20	ICMP	122	Echo (ping) reply id=0x0025, seq=209/53504 (request in 587)

Frame 588: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface veth4.1.1, id 0
Ethernet II, Src: 00:00:00:aa:00:05 (00:00:00:aa:00:05), Dst: 00:00:00:aa:00:04 (00:00:00:aa:00:04)
Internet Protocol Version 4, Src: 10.0.2.20, Dst: 10.0.2.20
0100 = Version: 4
... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 108
Identification: 0xd684 (54916)
Flags: 0x0000
Fragment offset: 0
Time to live: 64
Protocol: Authentication Header (51)
Header checksum: 0x8db3 [validation disabled]
[Header checksum status: Unverified]
Source: 10.0.2.20
Destination: 10.0.2.20
Authentication Header
Next header: ICMP (1)
Length: 4 (24 bytes)
Reserved: 0000
AH SPI: 0x00000300
AH Sequence: 198
AH ICV: 0e2f04b6b740aca141120d21
Internet Control Message Protocol

0000 00 00 00 aa 00 04 00 00 00 aa 00 05 08 00 45 00E
0010 00 6c d6 84 00 00 04 33 8d b3 0a 00 02 14 0a 00 -1....03

veth4.1.1: <live capture in progress> Packets: 588 · Displayed: 588 (100.0%) Profile: Default