



ECOLE
POLYTECHNIQUE
DE BRUXELLES

IRELE - MA1 Electrical Engineering

ELEC-H417

Report Lab 4

Virtual Private Network (VPN)

Authors :

Amaury ARICO

Alexis BOLLENGIER

Emmeran COLOT

Sefa GÖNEN

Professor :

Jean-Michel DRICOT

Assistant :

Navid LADNER

Academic year :

2024 - 2025

Contents

1 Mission 1 - Clear traffic observation	1
1.1 Question - TCP 3-way handshake procedure	1
1.2 Question - Sequence and acknowledgement number	1
2 Mission 2 - VPN configuration	2
2.1 Question - Cryptographic parameters	2
2.2 Bonus question - Why AES	2
3 Mission 3 - Encrypted traffic observation	3
3.1 Question - Wireshark Analysis	3
3.2 Question - VPN Uses	3
3.3 Bonus question - VPN Security	3

Mission 1 - Clear traffic observation

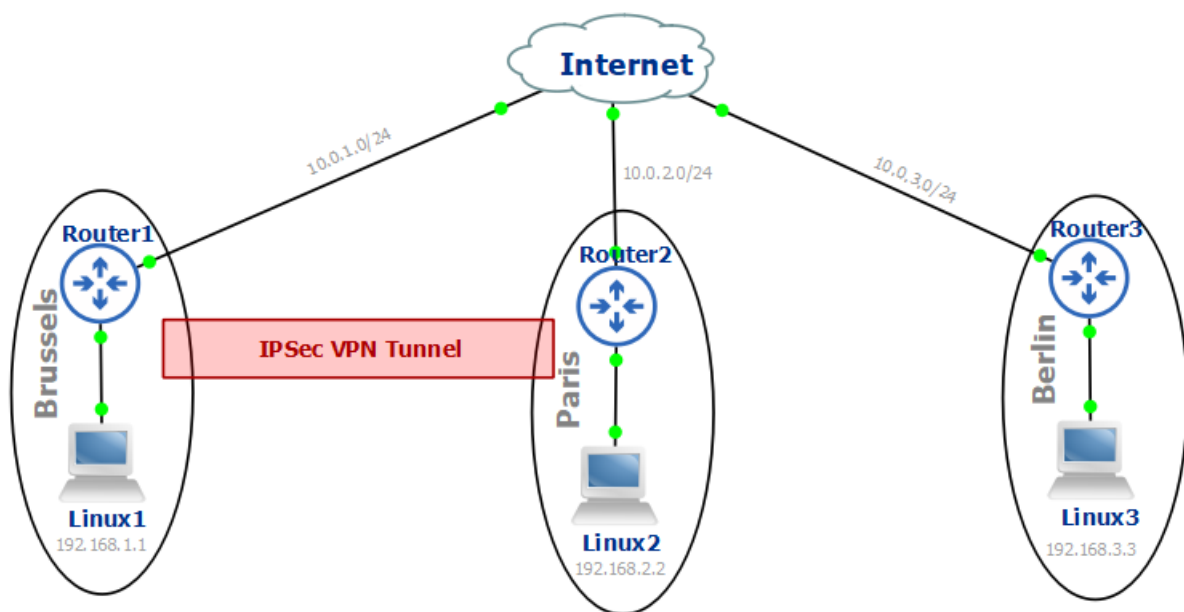


Figure 1.1: Topology of the network for the lab

1.1 Question - TCP 3-way handshake procedure

TCP 3-way handshake procedure

When you send your "Hello world!" message, you can observe 7 packets in Wireshark. Put your Wireshark capture in the report and highlight the payload value (i.e. the cleartext message). Could you explain why and what are all these packets ?

The payload¹ is at the end (layer 7 data) of the 4th packet. To ensure that the message is received, it is sent in **TCP** and this is the reason of the number of packet sent (7) despite the fact that the content of the message was only 28 bytes.

Let's analyze each request:

1. **Linux1** → **Linux2**, wanting to instantiate the connection with the **SYN** flag raised (for synchronization)

¹The payload was sent from Linux1 (192.168.1.1) to Linux2 (192.168.2.2) and contained "Network architecture is fun"

2. **Linux2** → **Linux1**, acknowledging (**ACK** flag + $Ack=1$) the reception of the synchronization (**SYN** flag) request
3. **Linux1** → **Linux2**, acknowledging (**ACK** flag + $Ack=1$) the acknowledgment of Linux2
4. **Linux1** → **Linux2**, pushing the payload (**PSH** flag) and still acknowledging the reception of the 2nd packet (**ACK** flag + $Ack=1$)
5. **Linux1** → **Linux2**, informing it of the finish of the transmission (**FIN** flag). It is necessary because if the payload is too long, it would have needed multiple push requests. This packet still acknowledges the last received packet (**ACK** flag + $Ack=1$)
6. **Linux2** → **Linux1**, acknowledging the reception of the 4th packet (**ACK** flag + $Ack=29$)
7. **Linux2** → **Linux1**, acknowledging the reception of the 5th packet (**ACK** flag + $Ack=30$)

To finish this analysis, let's just recall how the *acknowledgment number* is determined: when sending a packet of *sequence number* N and payload of length M , the expected acknowledgment response number is $M + N$. This number corresponds also to the sequence number of the next packet to send.

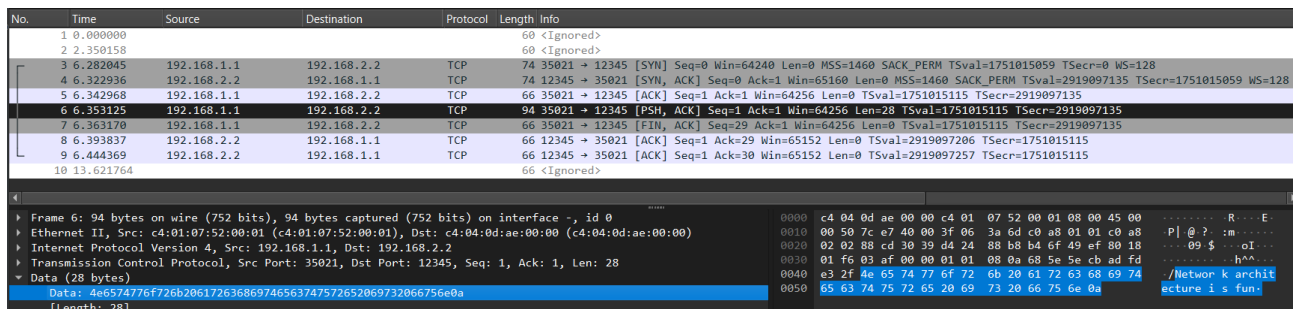


Figure 1.2: Wireshark capture of an *echo* request

1.2 Question - Sequence and acknowledgement number

Sequence and acknowledgement number

Explain what are the sequence number and acknowledgement number flags used for.

The explanation at the end of the previous question can be studied more in depth:

The **sequence number** is a value linked to the sender side² that indicates the packet that he is sending. This number is in fact a pointer that is set to 0 at the start of the communication.

The **acknowledgment number** is a value linked to the receiver side that indicates the next packet's sequence number that the receiver expects to receive. The calculation has been explained in the previous section (seq + length).

It can be seen on the Wireshark screenshot ?? that Linux1 send the packet with sequence number 29 without waiting for the acknowledgment number 29 to avoid waiting twice the travel time. In the case this acknowledgment never arrived, Linux1 would have sent the packet with sequence number 1 again.

²both PCs are senders and receivers

Mission 2 - VPN configuration

2.1 Question - Cryptographic parameters

Cryptographic parameters

Explain briefly each of these policy lines (i.e. What is AES ? What is a hash function and what could it be used for in this case ? What does pre-shared key means ? What is Diffie-Hellman ? What is the lifetime used for ?)

- **AES encryption:** transform the data in *cyphertext* in a way that is resistant to brute-force, making it unreadable for anyone intercepting it. It uses the same key for encryption and decryption and is based on *XORing*, substituting, shifting and mixing operations.
It is here used to make sure that the data cannot be read.
- **SHA hash function** (or *Secure Hash Algorithm*): a hash function converts a message in a fixed-size output in a deterministic way. SHA has the particularity that even if a bit of the original data is flipped, the output of the hash is entirely different.
It is here used to make sure that the data has not been modified during transmission.
- **Pre-shared key authentication:** verifies identity using a shared secret key known by both parties. This is used during the initial setup of the communication.
- **Group 2 Diffie-Hellman:** allows to share keys over an unsecured channel. The choice of group 2 is a good compromise between security and computation time.
Note that it is not used to share the authentication key as it is already known by the two correspondents.
- **Lifetime of 86400:** Sets the duration of the security association to 24 hours. After this time, both parts must check the identity of the other again (for example by sending the authentication key).

2.2 Bonus question - Why AES

Why AES

There is two family of encryption algorithms (symmetric and asymmetric). Which family does AES belong to ? Why do we use this family in our case, what is the main benefit ?

As mentioned before, it is a symmetric encryption method, meaning that the same key is used on both sides (for encryption and decryption). The benefit of it is that it is faster to decrypt/encrypt and easier to use than asymmetric encryption.

Mission 3 - Encrypted traffic observation

3.1 Question - Wireshark Analysis

Wireshark Analysis

Put both Wireshark captures of your communications with Paris and Berlin in your report and highlight in both the secret message send (encrypted or not). Put them in parallel and explain the differences between the message that used the VPN (aka. towards Paris) and the one that didn't (aka. towards Berlin).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000				66	<Ignored>
2	0.774957				60	<Ignored>
3	8.665174	192.168.1.1	192.168.3.3	TCP	74	36279 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3409914317 TSecr=0 WS=128
4	8.695974	192.168.3.3	192.168.1.1	TCP	74	12345 → 36279 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3951373055 TSecr=3409914317 WS=128
5	8.716091	192.168.1.1	192.168.3.3	TCP	66	36279 → 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3409914360 TSecr=3951373055
6	8.726140	192.168.1.1	192.168.3.3	TCP	79	36279 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=13 TSval=3409914360 TSecr=3951373055
7	8.736699	192.168.1.1	192.168.3.3	TCP	66	36279 → 12345 [FIN, ACK] Seq=14 Ack=1 Win=64256 Len=0 TSval=3409914360 TSecr=3951373055
8	8.747419	192.168.3.3	192.168.1.1	TCP	66	12345 → 36279 [ACK] Seq=1 Ack=14 Win=65152 Len=0 TSval=3951373106 TSecr=3409914360
9	8.797782	192.168.3.3	192.168.1.1	TCP	66	12345 → 36279 [ACK] Seq=1 Ack=15 Win=65152 Len=0 TSval=3951373160 TSecr=3409914360
10	10.766049				60	<Ignored>

Frame 6: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface -, id 0	0000	c4 04 0d ae 00 00 c4 01 07 52 00 01 08 00 45 00	R...E
Ethernet II, Src: c4:01:07:52:00:01 (c4:01:07:52:00:01), Dst: c4:04:0d:ae:00:00 (c4:04:0d:ae:00:00)	0010	00 41 24 77 40 00 3f 06 91 eb c0 a8 01 01 c0 a8	Asw@ ?
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.3.3	0020	03 03 8d b7 30 39 73 26 3c b6 cf 03 cc 63 80 18	...09s& <...c...
Transmission Control Protocol, Src Port: 36279, Dst Port: 12345, Seq: 1, Ack: 1, Len: 13	0030	01 f6 5e 7c 00 00 01 01 08 0a cb 3f 29 f8 eb 85?..
Data (13 bytes)	0040	2a ff 48 65 6c 6c 6f 20 77 6f 72 6c 64 21 0a	* Hello world!

Figure 3.1: **Berlin**: Wireshark capture of the *echo* request without the VPN

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	c4:01:07:52:00:01	c4:01:07:52:00:01	LOOP	60	Reply
2	0.870540				66	<Ignored>
3	6.372855				60	<Ignored>
4	10.211623				60	<Ignored>
5	11.484880	10.0.1.1	10.0.2.2	ESP	134	ESP (SPI=0x84e88058)
6	11.515424	10.0.2.2	10.0.1.1	ESP	134	ESP (SPI=0x3fd109b6)
7	11.526204	10.0.1.1	10.0.2.2	ESP	134	ESP (SPI=0x84e88058)
8	11.536374	10.0.1.1	10.0.2.2	ESP	150	ESP (SPI=0x84e88058)
9	11.546970	10.0.1.1	10.0.2.2	ESP	134	ESP (SPI=0x84e88058)
10	11.566782	10.0.2.2	10.0.1.1	ESP	134	ESP (SPI=0x3fd109b6)
11	11.618094	10.0.2.2	10.0.1.1	ESP	134	ESP (SPI=0x3fd109b6)
12	20.133406				60	<Ignored>

Frame 8: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface -, id 0	0000	c4 04 0d ae 00 00 c4 01 07 52 00 01 08 00 45 00	R...E
Ethernet II, Src: c4:01:07:52:00:01 (c4:01:07:52:00:01), Dst: c4:04:0d:ae:00:00 (c4:04:0d:ae:00:00)	0010	00 88 01 d4 40 00 ff 32 62 6d 0a 00 01 01 0a 00	@ 2 bm
Internet Protocol Version 4, Src: 10.0.1.1, Dst: 10.0.2.2	0020	02 02 84 e8 80 58 00 00 00 0c 98 fd 3a 18 ae d1	...X...%
Encapsulating Security Payload	0030	36 44 6d fd fc b9 d4 c9 25 e6 87 bf 10 dd ab c1	6Dm...%
ESP SPI: 0x84e88058 (2229829720)	0040	3e e0 85 3e b4 7c e7 03 ff 77 0f 2c f8 cf 33 48	>...>...w...3H
ESP Sequence: 12	0050	a8 17 35 cf 89 a0 da c3 97 7f 6a 80 3e ea 69 c7	..5...>...j...i..
	0060	13 c9 af d3 eb 71 10 5f bc c5 dd ed c9 28 c4 70	...q...>...<p
	0070	b6 50 f4 06 1d 51 92 67 a7 0e f2 4a 7f 9c e9 af	P...Q...g...>...<
	0080	2c 54 9f b9 ba 99 c9 78 d6 89 07 db c1 d1 55 d8	T...>...x...>...U...
	0090	ab cd 6c 0d 0f 67	..l...g

Figure 3.2: **Paris**: Wireshark capture of the *echo* request through the VPN channel

The main difference between the two situations here is that it is impossible for us to retrieve the payload¹ when the payload is sent through the VPN channel as it is encrypted.

Because the **IPSec** has been configured in tunnel mode, even the destination has been encrypted between Router1

¹"Hello world!"

and Router2, which means that we cannot be sure of the destination of the packet in fig ??.

Finally the length of the packages in fig ?? is smaller as the use of a VPN will add new headers and new trailers:

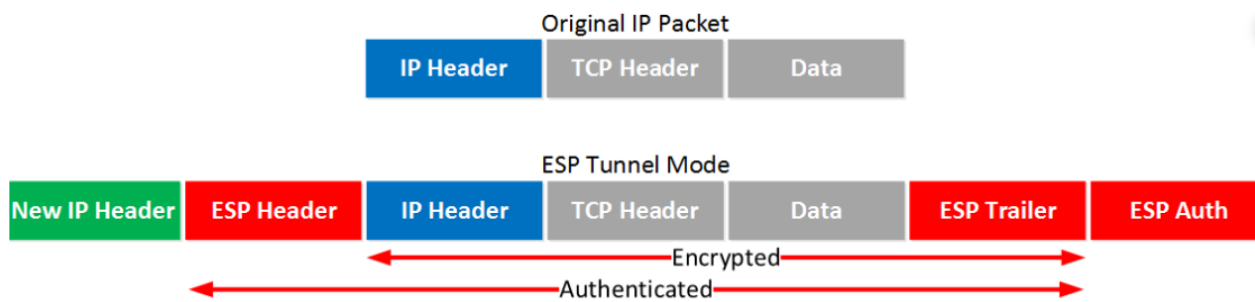


Figure 3.3: Wireshark capture of the *echo* request through the VPN channel

3.2 Question - VPN Uses

VPN Uses

What are the usages of VPN ?

The usage of a VPN make the content inside of the sent packages unreadable and unchangeable by any malicious actor thanks to the protection (encryption + hash). The big difference between a VPN and a "classical" encrypted channel is that it is the whole packet that is encrypted here. This means that even the emitter/receiver addresses are hidden. This implies that is used for security reasons, for example when using a public Wi-Fi. It can also be used to bypass firewalls as the destination in the header is just the exit of the VPN channel and not the real destination of the packet.

3.3 Bonus question - VPN Security

VPN Security

Is VPN really a good way to secure your communication ? Why ? If it is not the case, what could you do instead (or in addition) ?

The provider of the VPN can still see what you are sending in it and might store/sell this data so you must trust the provider. Once the packets have exited the VPN and are on their way to their real destination, they are no longer protected.

A way to prevent it is to encrypt the packets you send through the VPN with an *end to end encryption* (E2EE) such that the content of the packet stays encrypted until the last hop on the network.