IPsec and Transport Layer Security

Lab Report 4

Information Systems Security course (01TYM, 02KRQ)

prepared by:

Anuar Elio Magliari (s317033@studenti.polito.it)

George Florin Eftime (s303483@studenti.polito.it)

Alekos Interrante Bonadia (s319849@studenti.polito.it)

IPSec and security at IP level

3.1 Setting IPsec policies and Security Associations

- Which commands do you need to run to flush the SAD and SPD?
  - In order to flush the SAD the command needed is:
    - *ip xfrm state flush*
  - For the SPD database the command needed is:
    - *ip xfrm policy flush*
- Which commands do you need to run to check the content of the SAD and SPD databases?
  - *sudo ip xfrm state show*
  - *sudo ip xfrm policy show*
- which type of traffic is protected with IPsec?
  - In IPsec we protect the payload not the header (AH encapsulates also the real header). We do it with ESP or AH strategy ESP is ok for confidentiality, authentication, integrity and also for protection against replay attacks. Using AH we does not give encryption so usually ESP is preferred. In this solution we are using only ESP algorithm.
- which IPsec protocol is used?
  - The IPsec protocol used are AH and ESP protocols
- which security services are provided?
  - AH: authentication, integrity and protection against replay attacks

    - ESP: almost same functions provided by AH, plus confidentiality
- Which are the cryptographic algorithms used?
    - In this case is used AES using a 128 bit key.
- What are respectively the roles of the Security Association Database (SAD) and of the Security Policy Database (SPD) in the IPsec architecture?
    - SPD (Security Policy Database) is used to define rules for handling specific types of traffic. These rules are then applied to determine the appropriate Security Associations (SAs) for the packets with its security parameters (algorithms, keys, ...). Additionally, the SAD (Security Association Database) stores all active SAs along with their associated security parameters.
    - The association is saved in the “DB” and could be then checked using an index (SPI). In this case we are using the indexes 0x1000 for Alice DB and 0x2000 for Bob DB.
- Which directives are processed every time Alice sends a new IP packet?
    - The directives that are processed every time Alice sends a new IP packet are:
        - Checking the SPD to determine if there are rules to apply to this packet. If rules exist, proceed to the SAD; otherwise, proceed to Layer 2.
        - If rules are found, verify a valid index in the SAD and the policies, perform encryption, encapsulate the payload IP packet, and finally, send it to the destination.
- Why are two SAs necessary?
    - The SAs associations are two because even if the association is unidirectional, for a secure bidirectional channel we need to have SA policies in both nodes so that the channel can be secure and both DB (for Alice IP and Bob IP device) should be filled with the other index, key, IP and algorithm. Then, there are two tables, one for outgoing packets and one for incoming packets in each of the two nodes connected.
- Which is the purpose of the field SPI present in the IP packets exchanged?

- It is used to gather all information about the packet by consulting the SAD. SPI is a specific unique index for each connection channel so there is always a single and unique index of 32 bits associated with a IP in the SAD.

- Which is the scope of the field Seq Number?
  - The purpose of the sequence number within packets is to prevent replay and filtering attacks. In other words, it is a method to protect packets from manipulation by an attacker. It is also used for the integrity check of a packet by the endpoints.

3.2 Setting IPsec policies and Security Associations

- ESP with AES-128-CBC and HMAC-SHA1;
  - *ip xfrm state add src 192.168.1.43 dst 192.168.1.42 proto esp spi 0x1000 enc aes 0xaa22334455667788990aabbccddeeff auth sha1 0xbbccddeeff00112233445566778899aabbccddeeff*
  - *ip xfrm state add src 192.168.1.42 dst 192.168.1.43 proto esp spi 0x2000 enc aes 0xbb22334455667788990aabbccddeeff auth sha1 0xaaccddeeff00112233445566778899aabbccddeeff*
  - *ip xfrm policy add src 192.168.1.43 dst 192.168.1.42 dir out tmpl proto esp mode transport level required*
  - *ip xfrm policy add src 192.168.1.42 dst 192.168.1.43 dir in tmpl proto esp mode transport level required*
- AH with HMAC-SHA1:
  - *ip xfrm state add src 192.168.1.43 dst 192.168.1.42 proto ah spi 0x3000 auth sha1 0xbbccddeeff00112233445566778899aabbccddeeff*
  - *ip xfrm state add src 192.168.1.42 dst 192.168.1.43 proto ah spi 0x4000 auth sha1 0xaaccddeeff00112233445566778899aabbccddeeff*
  - *ip xfrm policy add src 192.168.1.43 dst 192.168.1.42 dir out tmpl proto ah mode transport level required*
  - *ip xfrm policy add src 192.168.1.42 dst 192.168.1.43 dir in tmpl proto ah mode transport level required*
- ESP with AES-128-CBC and AH with HMAC-SHA1:
  - *ip xfrm state add src 192.168.1.43 dst 192.168.1.42 proto esp spi 0x1000 enc aes 0xaa22334455667788990aabbccddeeff*

- *ip xfrm state add src 192.168.1.42 dst 192.168.1.43 proto esp spi 0x2000 enc aes 0xbb22334455667788990aabbccddeeff*
- *ip xfrm state add src 192.168.1.43 dst 192.168.1.42 proto ah spi 0x3000 auth sha1 0xbbccddeeff00112233445566778899aabbccddeeff*
- *ip xfrm state add src 192.168.1.42 dst 192.168.1.43 proto ah spi 0x4000 auth sha1 0xaaccddeeff00112233445566778899aabbccddeeff*
- *ip xfrm policy add src 192.168.1.43 dst 192.168.1.42 dir out tmpl proto esp mode transport level required*
- *ip xfrm policy add src 192.168.1.42 dst 192.168.1.43 dir in tmpl proto ah mode transport level required*

- ESP with AES-128-GCM (RFC-4106) Authenticated Encryption with Associated Data (AEAD).
  - *ip xfrm state add src 192.168.1.43 dst 192.168.1.42 proto esp spi 0x1000 aead rfc4106(gcm(aes)) 0xaa22334455667788990aabbccddeeff1234567896*
  - *ip xfrm state add src 192.168.1.42 dst 192.168.1.43 proto esp spi 0x2000 aead rfc4106(gcm(aes)) 0xbb22334455667788990aabbccddeeff1234567896*
  - *ip xfrm policy add src 192.168.1.43 dst 192.168.1.42 dir out tmpl proto esp mode transport level required*
  - *ip xfrm policy add src 192.168.1.42 dst 192.168.1.43 dir in tmpl proto esp mode transport level required*
- How many SAs and SPs do you need to define to implement the security mechanism 3 (above)?
  - 4 SAs and 2 SPs
- Which is the difference among the configurations 1 and 3 in terms of the structure of the IP packets obtained?

```
▾ Internet Protocol Version 4, Src: 172.22.16.169, Dst: 172.22.16.190
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 136
    Identification: 0x5f83 (24451)
  ▸ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: Encap Security Payload (50)
    Header Checksum: 0x612d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.22.16.169
    Destination Address: 172.22.16.190
▾ Encapsulating Security Payload
    ESP SPI: 0x00001000 (4096)
    ESP Sequence: 372
```

- ip xfrm policy update src 172.22.16.169 dst 172.22.16.190 dir in tmpl proto esp mode transport

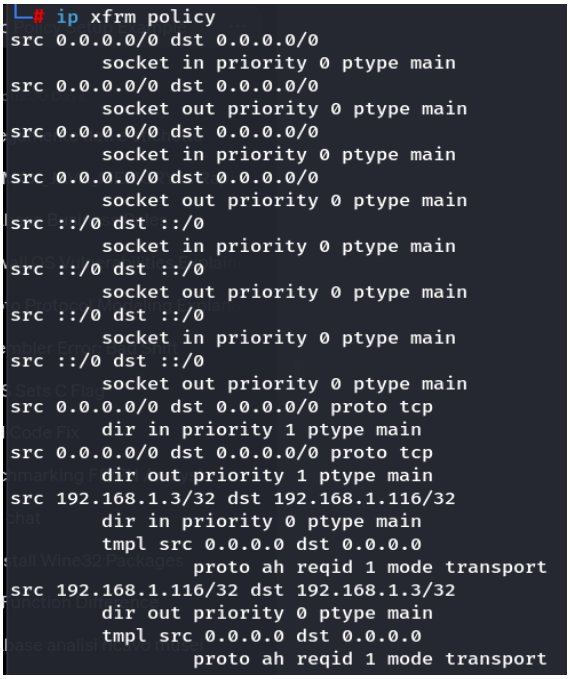
```
Internet Protocol Version 4, Src: 172.22.16.169, Dst: 172.22.16.190
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 124
    Identification: 0xac2c (44076)
  ▸ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: Encap Security Payload (50)
    Header Checksum: 0x1490 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.22.16.169
    Destination Address: 172.22.16.190
Encapsulating Security Payload
    ESP SPI: 0x00001000 (4096)
    ESP Sequence: 45
```

- ip xfrm policy update src 172.22.16.190 dst 172.22.16.169 dir out tmpl proto ah mode transport

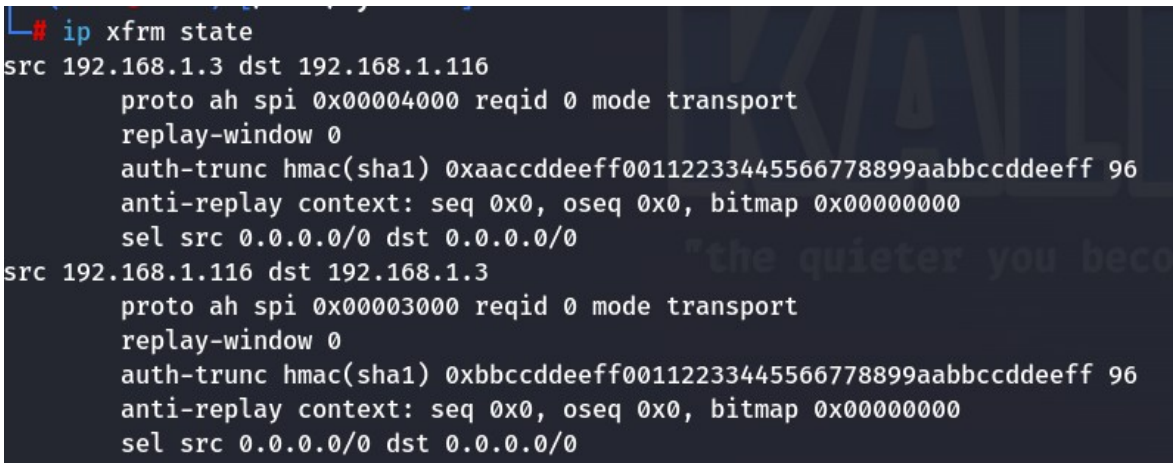
```
▸ Internet Protocol Version 4, Src: 172.22.16.190, Dst: 172.22.16.169
▾ Authentication Header
    Next header: ICMP (1)
    Length: 4 (24 bytes)
    Reserved: 0000
    AH SPI: 0x00004000
    AH Sequence: 42
    AH ICV: 1cd7a24e90a58b5310c3e53d
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xb7db [correct]
    [Checksum Status: Good]
    Identifier (BE): 49931 (0xc30b)
    Identifier (LE): 3011 (0x0bc3)
    Sequence Number (BE): 19 (0x0013)
    Sequence Number (LE): 4864 (0x1300)
    Timestamp from icmp data: Dec  1, 2023 12:39:13.000000000 CET
    [Timestamp from icmp data (relative): -33.404614242 seconds]
```

- Which is the difference if terms of security of the data contained in the IP packet?
  - In terms of security, there are no differences between the two configurations, but the first configuration can simplify the handling, while the second configuration, despite separating the concepts of confidentiality and authentication with integrity, cab be more complex to manage.
- When is the configuration 2 useful?
  - It is useful when the purpose is to have authentication and integrity for the packets (and consequently also authentication for the intermediate nodes). Furthermore, it’s faster because it does not need to encrypt and it can guarantee a partial protection by replay attacks.

- What advantage do we have using the configuration 4?
  - With configuration 4 we have a single algorithm for encryption and integrity so low overhead, more efficiency and simpler to implement.
- Choose one of the previous configurations and modify the SPs so that to protect only the TCP protocol messages (and not the messages corresponding to the other protocols). Verify subsequently that the ping packets are not protected, while the TCP traffic remains protected (for example the HTTP traffic).
  - ip xfrm policy add proto tcp dir out priority 1
  - ip xfrm policy add proto tcp dir in priority 1



- Finally, indicate which information is contained in the SAD and in the SPD.



Performance measurement

Now, execute the same performance measurements for the various IPsec configurations encountered in the previous exercises; write down the results in the Table 1, and compare them with the previous ones.

	10 KB	100 KB	1MB	10 MB	100 MB
No IPsec/TLS					
time [s]	0,12	0,15	0,51	4,03	37,18
speed [kB/s]	84,7	655	1929	2426	1911
ESP transport AES-128-CBC					
time [s]	0,35	0,82	0,37	5,21	38,70
speed [kB/s]	28,2	1190	2682	1846	3680
ESP transport AES-128-CBC HMAC-SHA1					
time [s]	0,58	0,19	0,45	3,1	28,32
speed [kB/s]	17,2	516	2194	3163	3699
AH transport HMAC-SHA1					
time [s]	0,07	0,08	0,37	2,58	27,46
speed [kB/s]	130	1268	2646	3789	2941
ESP AES-128-CBC + AH HMAC-SHA1					
time [s]	0,38	0,09	0,42	3,57	27,67
speed [kB/s]	26,45	1122	2336	2737	2614

The IKE protocol

- How many SAs have been negotiated?
  - There have been a total of 2 SAs negotiations.
- Which kind of security header and which mode is used in the SA? ESP or AH?
  - ESP
- Which algorithms are used?
  - AES-CBC-128
  - HMAC-SHA2-256-128
- Which is the key length used for each algorithm?
  - 128 bits
- In which phase of the IKE protocol is used the configured PSK?
  - The PSK is used during Phase 1 of the IKE protocol. During Phase 1, the PSK is used to authenticate the parties involved and to establish a secure initial communication channel. The PSK is a shared secret that is known to both the initiator and the responder of the IKE negotiation. This shared secret is used to prove the authenticity of each party and to derive the initial keying material for securing the subsequent communication.

The TLS protocol

- Which cipher suite has been negotiated?
  - Client

```
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 441
    Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 437
        Version: TLS 1.2 (0x0303)
      Random: 1e9ace8e26219fb54ac5e430646f43e00c0bb838addbe42f8e06462d49313f80
        Session ID Length: 32
        Session ID: 13e012e0cd1958f0cb67459a54afa8005b178e8c5742c7c8e6fcd3aac84e274d
        Cipher Suites Length: 182
      Cipher Suites (91 suites)
          Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
          Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
          Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
          Cipher Suite: TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 (0x00a3)
          Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
```

  - Server

```
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 122
    Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 118
        Version: TLS 1.2 (0x0303)
        Random: 76d88c271c5d36a27a41cdb7ea1c710e99e31bf707b5395c36d933a146de613f
        Session ID Length: 32
        Session ID: d3b36152fe5ddc3491b93c48a874fe9d8909c075303b4ba797ed986c64a1d6fb
        Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Compression Method: null (0)
        Extensions Length: 46
      Extension: supported_versions (len=2)
      Extension: key_share (len=36)
        [JA3S Fullstring: 771,4866,43-51]
        [JA3S: 15af977ce25de452b96affa2addb1036]
  TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.2 (0x0303)
      Length: 1
```

  During the handshake, the client presents to the server 91 available cipher suites where the server must choose one of them. In this case, the server choosed TLS_AES_256_GCM_SHA384

- Which TLS version has been negotiated?
  - The version negotiated is the 1.3 version

```
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 3072 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
```

- Which TLS version has been negotiated?
  - The version negotiated is the 1.2 version

```
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
```

- Which cipher suite has been negotiated?
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

```
Transport Layer Security
▾ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 69
  ▾ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 65
      Version: TLS 1.2 (0x0303)
    ▸ Random: 9e1addbb5c052087cc8d16a01c797476947c36e6c88bcacf6d249758bfbed3ee
      Session ID Length: 0
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
```

- Which algorithm has been used for key exchange?
  - EC Diffie-Hellman

```
Transport Layer Security
▾ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 300
  ▾ Handshake Protocol: Server Key Exchange
      Handshake Type: Server Key Exchange (12)
      Length: 296
    ▾ EC Diffie-Hellman Server Params
        Curve Type: named_curve (0x03)
        Named Curve: x25519 (0x001d)
        Pubkey Length: 32
        Pubkey: 783b7465af76c4c4dae2892dbfa1147a5dabe29e00b9ac2ffcb09a203c936808
      ▸ Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)
        Signature Length: 256
        Signature: 36cdcd34937453f7022f7faf693f6a019a1ab01b55aa1c8237441629edb871b5c52a819b…
Transport Layer Security
▾ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 4
  ▾ Handshake Protocol: Server Hello Done
      Handshake Type: Server Hello Done (14)
      Length: 0
```

```
Transport Layer Security
▾ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 37
  ▾ Handshake Protocol: Client Key Exchange
      Handshake Type: Client Key Exchange (16)
      Length: 33
    ▾ EC Diffie-Hellman Client Params
        Pubkey Length: 32
        Pubkey: a01454ad200966297fc76ad90282cc526cb0dfb278bb2bbca41ad34ac51e427b
▾ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
▾ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 40
    Handshake Protocol: Encrypted Handshake Message
```

Configuration of a TLS server

- Which is the purpose of the parameter -no_dhe used so far?
  - By default is set dhe (Diffie Hellman Ephemeral) that is used for the key exchange, instead using the command -no_dhe will disable this option
- What are and in which case can be used the ephemeral mechanisms for key exchange in TLS?
  - Ephemeral Diffie-Hellman (DHE)

        - DHE is used when the forward secrecy is desired
    - Ephemeral Elliptic Curve Diffie-Hellman (ECDHE)
        - ECDHE is used when the computational resource are limited so that it provides strong security with shorter keys
    - RSA Key Exchange with Forward Secrecy (RSA-FS)
        - DHE is generally preferred over RSA because it provides forward secrecy as well. While RSA can be used as an alternative, its reliance on long-term keys poses a security concern.

Use of Session ID

- For what purpose it is used the “Session ID” in TLS?
    - The session ID is very useful for skipping the phase of algorithm negotiation, enabling the use of the same algorithm decided in the previous session. However, new keys will be generated for the connection

Client authentication in TLS

- Now try to configure an SSL/TLS server with client authentication. What do you need in the first place?
    - We need to use the certificate for the server (it’s mandatory in the handshake).
- Identify the steps that have changed in the handshake phase of the SSL/TLS protocol.
    - The client has sent its certificate, so during the handshake phase, the server must also verify the client's certificate by checking the entire certificate chain. Thus, we achieve mutual authentication, enhancing security because the peers can engage in confidential communication.