# 2. Configuración IPSec con ESP

## 2.1. Generación de certificados en pc3, r1 y r4

**Crea el certificado autofirmado (pc1)**
root@pc1:/etc/ipsec.d# ipsec pki --gen --type rsa --size 4096 --outform pem > private/myCAKey.pem
root@pc1:/etc/ipsec.d# chmod 600 private/myCAKey.pem
root@pc1:/etc/ipsec.d# ipsec pki --self --ca --lifetime 3650 --in private/myCAKey.pem --type rsa --dn "C=ES, O=myCA, CN=My Root CA" --outform pem > cacerts/myCACert.pem

**Creación de una pareja de claves RSA de 2048 bits y el certificado de r1 firmado por la autoridad de certificación. (Igual para pc3 y r4, cambiando r1 donde merezca).**
root@pc1:/etc/ipsec.d# ipsec pki --gen --type rsa --size 2048 --outform pem > private/r1Key.pem
root@pc1:/etc/ipsec.d# chmod 600 private/r1Key.pem
root@pc1:/etc/ipsec.d# ipsec pki --pub --in private/r1Key.pem --type rsa | ipsec pki --issue --lifetime 730 --cacert cacerts/myCACert.pem --cakey private/myCAKey.pem --dn "C=ES, O=myCA, CN=r1" --san r1 --flag serverAuth --flag ikeIntermediate --outform pem > certs/r1Cert.pem

**1. Indica en la memoria los nombres de los ficheros de certificados y claves privadas que has generado y en qué carpetas los has almacenado en cada una de las máquinas.**

Pc3:
/etc/ipsec.d/cacerts/myCACert.pem
/etc/ipsec.d/private/pc3Key.pem
/etc/ipsec.d/certs/pc3Cert.pem

r1:
/etc/ipsec.d/cacerts/myCACert.pem
/etc/ipsec.d/private/r1Key.pem
/etc/ipsec.d/certs/r1Cert.pem

r4:
/etc/ipsec.d/cacerts/myCACert.pem
/etc/ipsec.d/private/r4Key.pem
/etc/ipsec.d/certs/r4Cert.pem

**2. Incluye en la memoria el resultado de 1 de forma legible cada uno de los certificados que has creado.**

**MyCACert:**

root@pc1:/etc/ipsec.d/cacerts# ipsec pki --print --in /etc/ipsec.d/cacerts/myCACert.pem
cert: X509
subject: "C=ES, O=myCA, CN=My Root CA"
issuer: "C=ES, O=myCA, CN=My Root CA"
validity: not before Apr 27 17:09:58 2017, ok

not after  Apr 25 17:09:58 2027, ok (expires in 3649 days)
serial:  00:96:ac:c4:74:c1:08:f9:02
flags:  CA CRLSign self-signed
authkeyId: 2b:55:fd:ab:b4:db:ae:54:79:ae:1c:9e:2a:19:3b:e6:ba:c9:5c:c9
subjkeyId: 2b:55:fd:ab:b4:db:ae:54:79:ae:1c:9e:2a:19:3b:e6:ba:c9:5c:c9
pubkey:  RSA 4096 bits
keyid:  28:88:a2:9b:ad:3e:5b:6e:7f:0f:82:a9:ec:0d:12:db:95:59:ed:63
subjkey:  2b:55:fd:ab:b4:db:ae:54:79:ae:1c:9e:2a:19:3b:e6:ba:c9:5c:c9

**r1Cert:**

root@r1:/etc/ipsec.d/cacerts# ipsec pki --print --in /etc/ipsec.d/certs/r1Cert.pem
cert:  X509
subject:  "C=ES, O=myCA, CN=r1"
issuer:  "C=ES, O=myCA, CN=My Root CA"
validity:  not before Apr 27 17:17:52 2017, ok
not after  Apr 27 17:17:52 2019, ok (expires in 729 days)
serial:  14:e7:0d:6d:23:90:3f:f5
altNames:  r1
flags:  serverAuth
authkeyId: 2b:55:fd:ab:b4:db:ae:54:79:ae:1c:9e:2a:19:3b:e6:ba:c9:5c:c9
subjkeyId: 2f:20:06:3a:1b:ac:a7:a6:e2:9f:71:08:d4:e3:5a:69:7c:42:c2:5b
pubkey:  RSA 2048 bits
keyid:  79:7c:f2:b5:f2:3d:4f:a9:a1:66:ea:bd:75:3d:62:33:c9:e3:f2:8e
subjkey:  2f:20:06:3a:1b:ac:a7:a6:e2:9f:71:08:d4:e3:5a:69:7c:42:c2:5b

**r4Cert:**

root@r4:/etc/ipsec.d/cacerts# ipsec pki --print --in /etc/ipsec.d/certs/r4Cert.pem
cert:  X509
subject:  "C=ES, O=myCA, CN=r4"
issuer:  "C=ES, O=myCA, CN=My Root CA"
validity:  not before Apr 27 17:23:52 2017, ok
not after  Apr 27 17:23:52 2019, ok (expires in 729 days)
serial:  19:fc:48:2d:85:80:3f:3f
altNames:  r4
flags:  serverAuth
authkeyId: 2b:55:fd:ab:b4:db:ae:54:79:ae:1c:9e:2a:19:3b:e6:ba:c9:5c:c9
subjkeyId: 51:a4:5f:56:43:fa:f0:8b:b5:97:f0:9a:97:7b:80:42:38:e0:8a:42
pubkey:  RSA 2048 bits
keyid:  72:d7:6c:58:e9:05:bb:24:22:6c:b8:b1:1f:6c:6d:ae:e4:ab:a3:8b
subjkey:  51:a4:5f:56:43:fa:f0:8b:b5:97:f0:9a:97:7b:80:42:38:e0:8a:42

**pc3Cert:**

root@pc3:/etc/ipsec.d/cacerts# ipsec pki --print --in /etc/ipsec.d/certs/pc3Cert.pem
cert:  X509
subject:  "C=ES, O=myCA, CN=pc3"
issuer:  "C=ES, O=myCA, CN=My Root CA"
validity:  not before Apr 27 17:37:38 2017, ok

not after  Apr 27 17:37:38 2019, ok (expires in 729 days)
serial:  2f:1a:57:2a:e6:17:31:80
altNames:  pc3
flags:  serverAuth
authkeyId: 2b:55:fd:ab:b4:db:ae:54:79:ae:1c:9e:2a:19:3b:e6:ba:c9:5c:c9
subjkeyId: 80:b1:51:98:a7:6e:a5:32:c7:a0:cf:fb:be:8b:a4:6f:83:50:23:08
pubkey:  RSA 2048 bits
keyid:  c2:53:a0:46:af:9a:3d:7e:44:d0:5a:89:e5:f4:b5:b4:7a:a3:b3:b7
subjkey:  80:b1:51:98:a7:6e:a5:32:c7:a0:cf:fb:be:8b:a4:6f:83:50:23:08

## 2.2. ESP en modo túnel entre r1 y r4

**r1:**

En ipsec.conf he añadido las siguientes líneas:

```
conn %default
ikelifetime=24h
keylife=24h
rekeymargin=3m
keyingtries=1
keyexchange=ikev2
ike=aes128-sha256-modp3072!
esp=aes128-sha256-modp3072!

conn net-net
left=100.10.2.1
leftcert=r1Cert.pem
leftid=@r1
leftsubnet=10.10.1.0/24
right=100.10.4.4
rightid=@r4
rightsubnet=10.10.2.0/24
type=tunnel
auto=add
```

En ipsec.secrets he añadido la siguiente línea:

```
: RSA r1Key.pem
```

**r4:**

En ipsec.conf he añadido las siguientes líneas:

```
conn %default
ikelifetime=24h
keylife=24h
rekeymargin=3m
keyingtries=1
keyexchange=ikev2
ike=aes128-sha256-modp3072!
esp=aes128-sha256-modp3072!

conn net-net
left=100.10.4.4
leftcert=r4Cert.pem
leftid=@r4
leftsubnet=10.10.2.0/24
right=100.10.2.1
rightid=@r1
```

rightsubnet=10.10.1.0/24
type=tunnel
auto=add

En ipsec.secrets he añadido la siguiente línea:

: RSA r4Key.pem

### 2.2.1. Gestión de SAs usando IKEv2

**1. Inicia un captura de tráfico en r1(eth1) para capturar el intercambio de mensajes IKE (recuerda usar la opción -s 0 para que se capturen los paquetes completos) y guarda los paquetes capturados en el fichero ipsec-00.cap. Arranca IPsec en los extremos y activa la configuración del túnel desde r1 para que comience el intercambio de mensajes IKE. Comprobarás que los extremos han acordado la SA que van a usar. Interrumpe la captura y cárgala en Wireshark para analizarla.**

ipsec start (En r1 y r4)
tcpdump -i eth1 -s 0 -w /hosthome/ipsec-00.cap &

root@r1:/etc# ipsec up net-net
initiating IKE_SA net-net[1] to 100.10.4.4
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) ]
sending packet: from 100.10.2.1[500] to 100.10.4.4[500]
received packet: from 100.10.4.4[500] to 100.10.2.1[500]
parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) CERTREQ N(MULT_AUTH) ]
received cert request for "C=ES, O=myCA, CN=My Root CA"
sending cert request for "C=ES, O=myCA, CN=My Root CA"
authentication of 'r1' (myself) with RSA signature successful
sending end entity cert "C=ES, O=myCA, CN=r1"
establishing CHILD_SA net-net
generating IKE_AUTH request 1 [ IDi CERT N(INIT_CONTACT) CERTREQ IDr AUTH SA TSi TSr N(MOBIKE_SUP) N(ADD_4_ADDR) N(MULT_AUTH) N(EAP_ONLY) ]
sending packet: from 100.10.2.1[4500] to 100.10.4.4[4500]
received packet: from 100.10.4.4[4500] to 100.10.2.1[4500]
parsed IKE_AUTH response 1 [ IDr CERT AUTH SA TSi TSr N(AUTH_LFT) N(MOBIKE_SUP) N(ADD_4_ADDR) ]
received end entity cert "C=ES, O=myCA, CN=r4"
  using certificate "C=ES, O=myCA, CN=r4"
  using trusted ca certificate "C=ES, O=myCA, CN=My Root CA"
checking certificate status of "C=ES, O=myCA, CN=r4"
certificate status is not available
  reached self-signed root ca with a path length of 0
authentication of 'r4' with RSA signature successful
IKE_SA net-net[1] established between 100.10.2.1[r1]...100.10.4.4[r4]
scheduling reauthentication in 86123s
maximum IKE_SA lifetime 86303s

**2. Indica qué campos puedes ver en cada uno de los mensajes IKE capturados. Explícalos.**

Existen dos paquetes IKE_SA_INIT MID=00, Initiator Request y Responder Response, que son los que inician la conexión.
Además, existen otros dos paquetes IKE_SA_INIT MID=00, Initiator Request y Responder Response

**3. Utilizando Wireshark podrás descifrar los paquetes IKEv2 si configuras en Wireshark las claves que se están utilizando para el intercambio de mensajes IKEv2. Para ello deberás tomar las claves que se encuentran en el fichero de logs de r1: /var/logs/charon.log. Localiza en ese fichero las claves:Sk ei, Sk er, Sk ai y Sk ar Con la captura que has realizado cargada en Wireshark selecciona en el menú Edit -> Preferences -> Protocols -> ISAKMP -> IKEv2 Decryption Table Edit y copia allí los valores sin dejar espacios en blanco:**

Sk_ei secret → 16 bytes @ 0x40288870
0: FA 32 DB FC 84 CA 27 C3 58 5E 70 A6 39 C7 84 2B .2....'.X^p.9..+
Sk_er secret → 16 bytes @ 0x40288870
0: DB 6F 01 B2 FD 0D 75 87 9C B1 B6 31 E4 00 14 0A .o....u....1.…
Sk_ai secret → 32 bytes @ 0x4028ae50
0: 63 80 08 95 63 69 2A F7 E3 27 60 02 10 E2 16 54 c...ci*..'`....T
16: 81 C1 A1 2A CF 07 C5 46 30 E3 8C 03 50 DA 60 54 ...*...F0...P.`T
Sk_ar secret → 32 bytes @ 0x4028ae50
0: EB AB F8 D3 17 6D CC 3F 75 71 F7 C3 70 30 57 1F .....m.?uq..p0W.
16: 8C A1 E6 F2 1F F7 C7 85 6F BA A5 A5 AE 2E 69 F9 ........o.....i.
Initiator's SPI (8 bytes) → parsing rule 0 IKE_SPI
=> => 8 bytes @ 0x4028ad50
0: 4F 15 06 E8 20 B9 97 33 O... ..3
Responder's SPI (8 bytes) → parsing rule 1 IKE_SPI
=> => 8 bytes @ 0x4028ad58
0: EB 15 CE 8A 5C 3B FF 77 ....\;.w
Encryption algorithm: AES-CBC-128 [RFC3602] → using encryption algorithm AES_CBC with key size 128
Integrity algorithm: HMAC_SHA_256_128 [RFC4868] → using integrity algorithm HMAC_SHA2_256_128 with key size 256

saul@saulibanez:~$ cat ~/.config/wireshark/ikev2_decryption_table
# This file is automatically generated, DO NOT MODIFY.
4f1506e820b99733,eb15ce8a5c3bff77,fa32dbfc84ca27c3585e70a639c7842b,db6f01b2fd0d75879cb1b631e400140a,"AES-CBC-128 [RFC3602]",6380089563692af7e327600210e2165481c1a12acf07c54630e38c0350da6054,ebabf8d3176dcc3f7571f7c37030571f8ca1e6f21ff7c7856fbaa5a5ae2e69f9,"HMAC_SHA2_256_128 [RFC4868]"

**a) En el mensaje IKE_AUTH que envía r1 a r4:**

**1) Fíjate como cada payload contiene un campo Next Payload que indica lo que viene a continuación. Indica todos los payloads que aparecen y cuál es el campo Next Payload del último.**

El primer payload se encuentra en IKE_SA_INIT MID=00, tanto en initiator como en responder, los dos su campo de siguiente payload es: Next payload: Security Association (33), aunque en el de responder tiene mas campos type payload. Como vemos, lo manda a Security Assiciation, asique vemos el campo e IKE_AUTH podemos ver que su campo de siguiente payload es: Next payload: Encrypted and Authenticated (46)

**2) Cómo se identifica al extremo initiator.**

ID type: FQDN (2)

**3) Qué certificado se incluye**

Certificate Encoding: X.509 Certificate - Signature (4), firmado por la CA
rdnSequence: 3 items (id-at-commonName=My Root CA,id-at-organizationName=myCA,id-at-countryName=ES)

**4) Cómo debe identificarse el otro extremo (responder).**

Certificate Encoding: X.509 Certificate - Signature (4)
rdnSequence: 3 items (id-at-commonName=My Root CA,id-at-organizationName=myCA,id-at-countryName=ES)

**5) Cómo el otro extremo (responder) va a autenticar a initiator.**

Comprobando el certificado y la firma.

**6) En la SA que se propone indica el protocolo IPSec que se va a utilizar, el SPI y los algoritmosque envía r1 a r4 para confidencialidad y autenticación.**

Protocol ID: ESP
SPI Size: 4
SPI: cf54a1c2
Encriptación → Transform ID (ENCR): ENCR_AES_CBC
Integridad → Transform ID (INTEG): AUTH_HMAC_SHA2_256_128

**7) Indica qué selectores de tráfico se envían, tanto para initiator como responder.**

Type Payload: Traffic Selector - Initiator (44) # 1
Next payload: Traffic Selector - Responder (45)
0... .... = Critical Bit: Not Critical
Payload length: 24
Number of Traffic Selector: 1
Traffic Selector Type: TS_IPV4_ADDR_RANGE (7)
Protocol ID: Unused
Selector Length: 16
Start Port: 0
End Port: 65535
Starting Addr: 10.10.1.0
Ending Addr: 10.10.1.255

Type Payload: Traffic Selector - Responder (45) # 1
Next payload: Notify (41)
0... .... = Critical Bit: Not Critical
Payload length: 24
Number of Traffic Selector: 1
Traffic Selector Type: TS_IPV4_ADDR_RANGE (7)
Protocol ID: Unused
Selector Length: 16
Start Port: 0
End Port: 65535
Starting Addr: 10.10.2.0
Ending Addr: 10.10.2.255

**b) En el mensaje IKE_AUTH que envía r4 a r1:**

**1) Cómo se identifica al extremo responder.**

ID_FQDN: r4

**2) Qué certificado se incluye**

Certificate Encoding: X.509 Certificate - Signature (4)
rdnSequence: 3 items (id-at-commonName=My Root CA,id-at-organizationName=myCA,id-at-countryName=ES)

**3) Cómo el otro extremo (initiator) va a autenticar a responder.**

Comprobando su certificado.

**4) En la SA que se propone indica el protocolo IPSec que se va a utilizar, el SPI y los algoritmos que envía r1 a r4 para confidencialidad y autenticación. Indica qué diferencias ves con respecto a lo que initiator envió en su propuesta de SA.**

Protocol ID: ESP (3)
SPI Size: 4
SPI: cce941f7
Transform ID (ENCR): ENCR_AES_CBC (12)
Transform ID (INTEG): AUTH_HMAC_SHA2_256_128 (12)

**5) Indica qué selectores de tráfico se envían, tanto para initiator como responder.**

Type Payload: Traffic Selector - Initiator (44) # 1
Next payload: Traffic Selector - Responder (45)
0... .... = Critical Bit: Not Critical
Payload length: 24
Number of Traffic Selector: 1
Traffic Selector Type: TS_IPV4_ADDR_RANGE (7)
Protocol ID: Unused
Selector Length: 16
Start Port: 0
End Port: 65535

Starting Addr: 10.10.1.0
Ending Addr: 10.10.1.255
Type Payload: Traffic Selector - Responder (45) # 1
Next payload: Notify (41)
0... .... = Critical Bit: Not Critical
Payload length: 24
Number of Traffic Selector: 1
Traffic Selector Type: TS_IPV4_ADDR_RANGE (7)
Protocol ID: Unused
Selector Length: 16
Start Port: 0
End Port: 65535
Starting Addr: 10.10.2.0
Ending Addr: 10.10.2.255

**4. Consulta SAD y SPD en cada uno de los extremos e incluye en la memoria la información relevante para el túnel IPSec. Fíjate en los valores SPI e indica si estos valores coinciden con los de la SA negociada previamente.**

```
root@r1:/etc# ip xfrm state
src 100.10.2.1 dst 100.10.4.4
        proto esp spi 0xcce941f7 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                        auth-trunc      hmac(sha256)
0xe5cda8c3e15c43998deb107f701195568203922e181b7c26496d70721d3a419b 128
        enc cbc(aes) 0x5eb8a973bbf1d3cd18ae4fc4882fee39
src 100.10.4.4 dst 100.10.2.1
        proto esp spi 0xcf54a1c2 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                        auth-trunc      hmac(sha256)
0x26a063d987ce309e82df2eddeb8e616adb3b1e0522a6ef750cc088de4f388ac7 128
        enc cbc(aes) 0xeb2a763822628c102d28b2bd295f4b6b


root@r4:/etc# ip xfrm state
src 100.10.4.4 dst 100.10.2.1
        proto esp spi 0xcf54a1c2 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                        auth-trunc      hmac(sha256)
0x26a063d987ce309e82df2eddeb8e616adb3b1e0522a6ef750cc088de4f388ac7 128
        enc cbc(aes) 0xeb2a763822628c102d28b2bd295f4b6b
src 100.10.2.1 dst 100.10.4.4
        proto esp spi 0xcce941f7 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                        auth-trunc      hmac(sha256)
0xe5cda8c3e15c43998deb107f701195568203922e181b7c26496d70721d3a419b 128
        enc cbc(aes) 0x5eb8a973bbf1d3cd18ae4fc4882fee39


root@r1:/etc# ip xfrm policy
```

```
src 10.10.2.0/24 dst 10.10.1.0/24
        dir fwd priority 1859 ptype main
        tmpl src 100.10.4.4 dst 100.10.2.1
                proto esp reqid 1 mode tunnel
src 10.10.2.0/24 dst 10.10.1.0/24
        dir in priority 1859 ptype main
        tmpl src 100.10.4.4 dst 100.10.2.1
                proto esp reqid 1 mode tunnel
src 10.10.1.0/24 dst 10.10.2.0/24
        dir out priority 1859 ptype main
        tmpl src 100.10.2.1 dst 100.10.4.4
                proto esp reqid 1 mode tunnel
src ::/0 dst ::/0
        socket in priority 0 ptype main
src ::/0 dst ::/0
        socket out priority 0 ptype main
```
(Estas últimas líneas se repiten unas cuantas veces).

```
root@r4:/etc# ip xfrm state
src 100.10.4.4 dst 100.10.2.1
        proto esp spi 0xcf54a1c2 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                                auth-trunc      hmac(sha256)
0x26a063d987ce309e82df2eddeb8e616adb3b1e0522a6ef750cc088de4f388ac7 128
        enc cbc(aes) 0xeb2a763822628c102d28b2bd295f4b6b
src 100.10.2.1 dst 100.10.4.4
        proto esp spi 0xcce941f7 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                                auth-trunc      hmac(sha256)
0xe5cda8c3e15c43998deb107f701195568203922e181b7c26496d70721d3a419b 128
        enc cbc(aes) 0x5eb8a973bbf1d3cd18ae4fc4882fee39
```

```
root@r4:/etc# ip xfrm policy
src 10.10.1.0/24 dst 10.10.2.0/24
        dir fwd priority 1859 ptype main
        tmpl src 100.10.2.1 dst 100.10.4.4
                proto esp reqid 1 mode tunnel
src 10.10.1.0/24 dst 10.10.2.0/24
        dir in priority 1859 ptype main
        tmpl src 100.10.2.1 dst 100.10.4.4
                proto esp reqid 1 mode tunnel
src 10.10.2.0/24 dst 10.10.1.0/24
        dir out priority 1859 ptype main
        tmpl src 100.10.4.4 dst 100.10.2.1
                proto esp reqid 1 mode tunnel
src ::/0 dst ::/0
        socket out priority 0 ptype main
src ::/0 dst ::/0
```

socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
socket out priority 0 ptype main
(Estas últimas líneas se repiten unas cuantas veces).

Coinciden.

**5. Consulta la tabla de encaminamiento 220 (creada por strongswan) para ver qué entradas se han insertadoen r1 y r4.**

root@r1:/etc# ip route list table 220
10.10.2.0/24 via 100.10.2.2 dev eth1  proto static  src 10.10.1.1

root@r4:/etc#  ip route list table 220
10.10.1.0/24 via 100.10.4.3 dev eth0  proto static  src 10.10.2.4

## 2.2.2. Comunicación usando IPSec (ESP en modo túnel)

**1. Explica detalladamente qué es lo que crees que ocurriría en r1 cuando:**
**pc1 le envía un datagrama IP a pc2**
**pc2 le envía un datagrama IP a pc1**

Con el túnel cerrado, no podrán llegar los paquetes a su destino.
Con el túnel abierto, y la conexión activada, si pc1 le hace un ping a pc2, este llegará a su destino, y viceversa también.

SAD (Security Asociation Database) y la SPD (Security Policy Database).

**Política in**: se recibe un paquete dirigido a la dirección IP del router.

De la cabecera se obtienen SPI, las direcciones IP y el protocolo que se está usando ESP/AH. Con esta información se busca en SAD, donde debería existir SA (Security Asociation).
→ Si no existe SA se provoca un error.
→ Si existe, el SA indica qué algoritmos se necesitan para descifrar/autenticar el paquete y se procede a ello.

A partir de SA se obtiene la política correspondiente SP (Security Policy) en SDP para comprobar si el mecanismo de seguridad especificado en la política es el que se tenía que aplicar al paquete.

**política out:** el router genera un paquete.

Se busca en SPD para ver si cumple los selectores de la política (IP origen, IP destino, etc) y saber si es necesario que lo procese IPSec, si se descarta o si se envía en claro.
Si se necesita aplicar IPSec la máquina consultará SAD.
→ Si existe SA se aplicarán los parámetros de seguridad de SA.
→ Si no existe SA, se avisará a IKE para que establezca SA.

**2. Inicia las siguientes capturas de tráfico:**
**En r1(eth1) y guarda los paquetes capturados en el fichero ipsec-01.cap.**
**En r4(eth0) y guarda los paquetes capturados en el fichero ipsec-02.cap.**
**En pc2 y guarda los paquetes capturados en el fichero ipsec-03.cap.**
**Realiza un ping desde pc1 a pc2. Interrumpe las capturas y observa su contenido en Wireshark. Wireshark mostrará para cada paquete recibido el paquete cifrado y el paquete en claro. Sin embargo para el paquete enviado sólo lo muestra cifrado.**
**Explica el contenido que puedes ver de los paquetes ESP que se corresponden con los paquetes echo request/echo reply capturados y cómo varía el campo ESP Sequence.**
**Si una máquina maliciosa intermedia capturara los paquetes de la comunicación entre pc1 y pc2, ¿crees que podría saber qué direcciones IP de están comunicando? ¿por qué?**

r1 → tcpdump -i eth1 -s 0 -w /hosthome/ipsec-01.cap
r4 → tcpdump -i eth0 -s 0 -w /hosthome/ipsec-02.cap
pc2 → tcpdump -i eth0 -s 0 -w /hosthome/ipsec-03.cap

El campo ESP Sequence va variando de uno en uno, en mi caso, entre 12 y 14. En la captura ipsec-03, no tiene el campo ESP Sequence, ya que no tiene ningún protocolo ESP.

No, porque las ip que circulan por el túnel son las de los extremos, en este caso, la de r1 y r4. Una vez ha llegado a los extremos, los manda a la máquina correspondiente.

**3. ¿Con qué TTL crees que se habrán recibido los paquetes en pc2?. Compruébalo observando la captura ipsec-03.cap. Explica el resultado.**

El campo TTL va variando entre 62 (request) y 64 (reply).
El 64 es justo donde se captura las peticiones, por eso tiene el valor máximo, y el de 62 es porque atraviesa los extremos del túnel.

**4. Utilizando Wireshark podrás descifrar los paquetes ESP si configuras en Wireshark las claves que se están utilizando para el intercambio de mensajes ESP. Para ello deberás tomar las claves que has mostrado en SAD. Con la captura ipsec-01.cap cargada en Wireshark selecciona en el menú: Edit -> Preferences -> Protocols -> ESP**
**marca la opción de descifrar y autenticar paquetes y edita la configuración para copiar allí los valores de cada uno de los SAs que muestra la SAD de r1 sin dejar espacios en blanco:**

- **Protocol**
- **Src IP**
- **Dest IP**
- **SPI**
- **Encryption algorithm: AES-CBC [RFC3602].**
- **Encryption key**
- **Authentication algorithm: HMAC-SHA-256-128 [RFC4868].**
- **Authentication key**

```
root@r1:/etc# ip xfrm state
src 100.10.2.1 dst 100.10.4.4
        proto esp spi 0xce507b0c reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                        auth-trunc      hmac(sha256)
0x6b863d1f2a89f0b101784ef40ed70267fedd0b747f31f39620bdd048b46c85b7 128
        enc cbc(aes) 0x06570729379b222f298cf6b2cc4537b8
src 100.10.4.4 dst 100.10.2.1
        proto esp spi 0xc76d33ac reqid 1 mode tunnel
        replay-window 32 flag af-unspec
                                                        auth-trunc      hmac(sha256)
0x34f326c2a7c27fccf87cd699763ca56c348a1adb17da916ade35cdac2d8fdf4a 128
        enc cbc(aes) 0x2cf3bf3d64b4f07ad54d18d807f8c109
```

**5. Explica el contenido de la cabecera ESP de los paquetes que se corresponden con los paquetes echo request/echo reply capturados, observa cómo ahora puedes ver todos los campos.**

```
Encapsulating Security Payload
   ESP SPI: 0xcc66b38d (3429282701)
   ESP Sequence: 1
```

ESP IV: f812b48f01cd0a1e9f8e4608ee6cbba2
Data (96 bytes)
Data: 4688a4efa0eb6bb9833a27fe2a8908806a4ec8135938061b...
[Length: 96]
Authentication Data
[Good: False]
[Bad: False]

Encapsulating Security Payload
ESP SPI: 0xc6c058bf (3334494399)
ESP Sequence: 1
ESP IV: 7ea9ab931949e7ae6ea9a33546b5f495
Data (96 bytes)
Data: 2fff91ff6028973b3092cf33f38a21ee842d450f2d5add16...
[Length: 96]
Authentication Data
[Good: False]
[Bad: False]

**6. ¿Observando la captura puedes saber si IPsec está trabajando en modo túnel o en modo transporte?**

A modo túnel, observando la captura, podemos ver que el túnel se conecta de r1 a r4, pero luego r1 tiene conexión con pc1 a través de la subred, implica que puede tener a más periféricos que puedan usar la misma subred, y no esta restringido solo a pc1.
En la captura podemos observar las dos cabeceras, y como el mensaje ha sido descifrado enteramente.

**7. ¿Qué crees que ocurrirá si se envía tráfico TCP/UDP de pc1 a pc2? ¿Por qué? Inicia una captura nuevamente en r1(eth1) y guarda los paquetes capturados en el fichero ipsec-04.cap.**

**Compruébalo utilizando nc para lanzar un cliente y un servidor en dichas máquinas, primero con UDP y después con TCP. Examina la captura e indica qué ves.**

```
root@pc1:~# nc -u -l -p 7777
hola
que tal
bien
y tu?
^C
root@pc2:~# nc -u -p 6666 10.10.1.10 7777
hola
que tal
bien
y tu?
^C

root@pc1:~# nc -l -p 7777
hola
que tal
```

bien
y tu?

root@pc2:~# nc -p 6666 10.10.1.10 7777
hola
que tal
bien
y tu?
^C

Aparece el tráfico de UDP, OSPF y ESP cuando uso UDP, y cuando ejecuto el de TCP, se ve el protocolo TCP y ESP.

**8. Inicia una captura nuevamente en r1(eth1) y guarda los paquetes capturados en el fichero ipsec-05.cap. Interrumpe IPSec en r1 e interrumpe la captura de paquetes. Utiliza Wireshark para visualizar los paquetes capturados explica qué ocurre.**

root@r1:/etc#  tcpdump -i eth1 -s 0 -w /hosthome/ipsec-05.cap &
[1] 3456
root@r1:/etc# tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
root@r1:/etc# ipsec stop
Stopping strongSwan IPsec...
root@r1:/etc# fg
tcpdump -i eth1 -s 0 -w /hosthome/ipsec-05.cap
^C57 packets captured
57 packets received by filter
0 packets dropped by kernel
root@r1:/etc#

root@pc1:~# ping 10.10.2.20
PING 10.10.2.20 (10.10.2.20) 56(84) bytes of data.
64 bytes from 10.10.2.20: icmp_req=1 ttl=62 time=0.952 ms
64 bytes from 10.10.2.20: icmp_req=2 ttl=62 time=0.537 ms
64 bytes from 10.10.2.20: icmp_req=3 ttl=62 time=0.599 ms
64 bytes from 10.10.2.20: icmp_req=4 ttl=62 time=0.450 ms
64 bytes from 10.10.2.20: icmp_req=5 ttl=62 time=0.444 ms
64 bytes from 10.10.2.20: icmp_req=6 ttl=62 time=0.572 ms
64 bytes from 10.10.2.20: icmp_req=7 ttl=62 time=0.496 ms
64 bytes from 10.10.2.20: icmp_req=8 ttl=62 time=0.442 ms
64 bytes from 10.10.2.20: icmp_req=9 ttl=62 time=0.842 ms
64 bytes from 10.10.2.20: icmp_req=10 ttl=62 time=0.769 ms
64 bytes from 10.10.2.20: icmp_req=11 ttl=62 time=0.654 ms
^C
--- 10.10.2.20 ping statistics ---
29 packets transmitted, 11 received, 62% packet loss, time 28088ms
rtt min/avg/max/mdev = 0.442/0.614/0.952/0.165 ms

Cuando se cierra el túnel, pc1 ya no tendrá conexión con pc2.

## 2.3. ESP en modo transporte entre pc3 y r4

Supón que pc3 quiere establecer una comunicación IPSec ESP con r4 en modo transporte para una aplicación TCP que se encuentra ejecutándose en r4 esperando recibir conexiones en el puerto 4444. pc3 se conectará desde el puerto local 3333 a dicho puerto remoto en r4. Se va a utilizar IKEv2 para que establezca las asociaciones de seguridad SA con los siguientes algoritmos de seguridad (cifrado, integridad y grupo Diffie-Hellman): aes128-sha256-modp3071.

**1. Realiza la configuración en los ficheros que sean necesarios para permitir esta comunicación. Incluye estos ficheros en la memoria. No olvides guardar en la máquina real los ficheros que modifiques, de esta forma tendrás una copia de seguridad.**

**Pc3:**

```
root@pc3:/etc# vim ipsec.con
conn %default
        ikelifetime=24h
        keylife=24h
        rekeymargin=3m
        keyingtries=1
        keyexchange=ikev2
        ike=aes128-sha256-modp3072!
        esp=aes128-sha256-modp3072!

conn host-host
        leftprotoport=tcp/3333
        left=100.10.7.30
        leftcert=pc3Cert.pem
        leftid=@pc3
        rightprotoport=tcp/4444
        right=100.10.4.4
        rightid=@r4
        rightcert=r4Cert.pem
        type=transport
        auto=add

root@pc3:/etc# vim ipsec.secrets
: RSA pc3Key.pem
```

**R4:**

```
root@r4:/etc# vim ipsec.conf
conn %default
        ikelifetime=24h
        keylife=24h
        rekeymargin=3m
        keyingtries=1
        keyexchange=ikev2
        ike=aes128-sha256-modp3072!
        esp=aes128-sha256-modp3072!
```

```
conn host-host
        leftprotoport=tcp/4444
        left=100.10.4.4
        leftcert=r4Cert.pem
        leftid=@r4
        rightprotoport=tcp/3333
        right=100.10.7.30
        rightid=@pc3
        rightcert=pc3Cert.pem
        type=transport
        auto=add
```

**2. Inicia una captura en pc3 para guardar su contenido en el fichero ipsec-06.cap y establece la configuración ESP en modo transporte entre ambas máquinas. Ejecuta un ping desde la máquina pc3. Interrumpe la captura y explica su contenido.**

```
root@pc3:/etc# ipsec up host-host
root@pc3:/etc# ipsec start
Starting strongSwan 4.5.2 IPsec [starter]...
root@pc3:/etc# ipsec up host-host
initiating IKE_SA host-host[1] to 100.10.4.4
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) ]
sending packet: from 100.10.7.30[500] to 100.10.4.4[500]
received packet: from 100.10.4.4[500] to 100.10.7.30[500]
parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) CERTREQ
N(MULT_AUTH) ]
received cert request for "C=ES, O=myCA, CN=My Root CA"
sending cert request for "C=ES, O=myCA, CN=My Root CA"
authentication of 'pc3' (myself) with RSA signature successful
sending end entity cert "C=ES, O=myCA, CN=pc3"
establishing CHILD_SA host-host
generating IKE_AUTH request 1 [ IDi CERT N(INIT_CONTACT) CERTREQ IDr AUTH
N(USE_TRANSP) SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N(MULT_AUTH)
N(EAP_ONLY) ]
sending packet: from 100.10.7.30[4500] to 100.10.4.4[4500]
received packet: from 100.10.4.4[4500] to 100.10.7.30[4500]
parsed IKE_AUTH response 1 [ IDr CERT AUTH N(USE_TRANSP) SA TSi TSr N(AUTH_LFT)
N(MOBIKE_SUP) N(ADD_4_ADDR) ]
received end entity cert "C=ES, O=myCA, CN=r4"
  using certificate "C=ES, O=myCA, CN=r4"
  using trusted ca certificate "C=ES, O=myCA, CN=My Root CA"
checking certificate status of "C=ES, O=myCA, CN=r4"
certificate status is not available
  reached self-signed root ca with a path length of 0
authentication of 'r4' with RSA signature successful
IKE_SA host-host[1] established between 100.10.7.30[pc3]...100.10.4.4[r4]
scheduling reauthentication in 86093s
maximum IKE_SA lifetime 86273s
```

Pc3→ tcpdump -i eth0 -s 0 -w /hosthome/ipsec-06.cap &

No aparecen mensajes ESP.

**3. Observa el contenido de SAD y SDP en pc3 y explica su contenido.**

```
root@pc3:/etc# ip xfrm state
src 100.10.7.30 dst 100.10.4.4
        proto esp spi 0xc9db3898 reqid 1 mode transport
        replay-window 32
                                                                auth-trunc      hmac(sha256)
0xaac2934416e0cd315328768a8bb14bd27717d2d1fbc1751c34987a8989bf2c7c 128
        enc cbc(aes) 0xf7d89175173d6b0d6bfc3bfe8dad001a
        sel src 100.10.7.30/32 dst 100.10.4.4/32 proto tcp sport 3333 dport 4444
src 100.10.4.4 dst 100.10.7.30
        proto esp spi 0xcb58a3f7 reqid 1 mode transport
        replay-window 32
                                                                auth-trunc      hmac(sha256)
0x80978a1f5af9b9baff524aa25abb4a622b994d9859c0efb3a184c4c2d49b7aaa 128
        enc cbc(aes) 0x731c2702fd177bca3c94caf9498fdaf4
        sel src 100.10.4.4/32 dst 100.10.7.30/32 proto tcp sport 4444 dport 3333
```

En esta tabla (SDP) aparecen los puertos y el protocolo que hemos configurado.

```
root@pc3:/etc# ip xfrm policy
src 100.10.4.4/32 dst 100.10.7.30/32 proto tcp sport 4444 dport 3333
        dir in priority 1792 ptype main
        tmpl src 0.0.0.0 dst 0.0.0.0
                proto esp reqid 1 mode transport
src 100.10.7.30/32 dst 100.10.4.4/32 proto tcp sport 3333 dport 4444
        dir out priority 1792 ptype main
        tmpl src 0.0.0.0 dst 0.0.0.0
                proto esp reqid 1 mode transport
src ::/0 dst ::/0
        socket out priority 0 ptype main
src ::/0 dst ::/0
        socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
        socket out priority 0 ptype main
```

En esta tabla (SAD) vemos que ahora se refiere al modo transport.

**4. Inicia una captura en pc3 para guardar su contenido en el fichero ipsec-07.cap y establece la configuración ESP en modo transporte entre ambas máquinas. Usando nc ejecuta un servidor TCP en r4 en el puerto 4444 al que se conecta un cliente desde la máquina pc3 y el puerto 3333. Escribe algo desde el cliente para que se transmita al servidor. Interrumpe la captura y responde las siguientes cuestiones:**

root@r4:/etc# nc -l -p 4444

hola que tal
bien y tu

root@pc3:/etc# nc -p 3333 100.10.4.4 4444
hola que tal
bien y tu
^C
root@pc3:/etc# fg
tcpdump -i eth0 -s 0 -w /hosthome/ipsec-07.cap
^C10 packets captured
10 packets received by filter
0 packets dropped by kernel

**a) ¿Qué campos observas en los paquetes ESP?**

Encapsulating Security Payload
ESP SPI: 0xc9db3898 (3386587288)
ESP Sequence: 7

**b) Descifra el contenido de ESP para ver exactamente qué es lo que se envía. ¿Qué diferencias observas con respecto al modo túnel?**

root@pc3:/etc# ip xfrm state
src 100.10.7.30 dst 100.10.4.4
proto esp spi 0xc9db3898 reqid 1 mode transport
replay-window 32
auth-trunc hmac(sha256) 0xaac2934416e0cd315328768a8bb14bd27717d2d1fbc1751c34987a8989bf2c7c 128
enc cbc(aes) 0xf7d89175173d6b0d6bfc3bfe8dad001a
sel src 100.10.7.30/32 dst 100.10.4.4/32 proto tcp sport 3333 dport 4444
src 100.10.4.4 dst 100.10.7.30
proto esp spi 0xcb58a3f7 reqid 1 mode transport
replay-window 32
auth-trunc hmac(sha256) 0x80978a1f5af9b9baff524aa25abb4a622b994d9859c0efb3a184c4c2d49b7aaa 128
enc cbc(aes) 0x731c2702fd177bca3c94caf9498fdaf4
sel src 100.10.4.4/32 dst 100.10.7.30/32 proto tcp sport 4444 dport 3333

Encapsulating Security Payload
ESP SPI: 0xc9db3898 (3386587288)
ESP Sequence: 7
ESP IV: 9ba81a974aa1fbb531e554bc96f954b9
Pad: 010203040506
ESP Pad Length: 6
Next header: TCP (0x06)
Authentication Data [correct]
Transmission Control Protocol, Src Port: 3333 (3333), Dst Port: 4444 (4444), Seq: 0, Len: 0
Source Port: 3333

Destination Port: 4444
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Acknowledgment number: 0
Header Length: 40 bytes
Flags: 0x002 (SYN)
Window size value: 14480
[Calculated window size: 14480]
Checksum: 0x50fc [validation disabled]
Urgent pointer: 0
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale

En modo transporte despues de Encapsulatin Security Payload, tenemos el campo Transmission Control Protocol, sin embargo, observamos que esta opción no tiene Internet Control Message Protocol.

**c) ¿Con qué TTL se reciben los paquetes IP en r4? ¿Por qué?**

Time to live: 64 3333 → 4444
Time to live: 62 4444 → 3333

Hay una diferencia de 2 porque son los saltos que se producen antes de llegar al destino.