

Universidad Nacional de Ingeniería

Facultad de ciencias



Administración de Redes

Protección a nivel de red (IPSec)

Protección a nivel de transporte (SSL / TLS / WTLS)

Redes Privadas Virtuales VPN

Nombres:

- Ávila Santos Alex
- Trejo Chávez Konrad
- Ramos Grados Luis
- Gerald Palomino Monge

Profesor: Manuel Castillo Cara

2019

Índice:

Marco Teórico IPsec	3
Configuración entorno de red IPsec	6
Configuración Clientes IPsec	7
Configuración Servidor IPsec	9
Pruebas IPsec	15
Marco Teórico SSL	17
Configuración entorno de red SSL	19
Configuración Clientes SSL	20
Configuración Servidor SSL	23
Bibliografía.....	29

1. Marco Teórico

¿Qué es IPsec?

La seguridad de IP (IPSec) es un conjunto estándar de protocolos de Internet Engineering Task Force (IETF) entre 2 puntos de comunicación a través de la red IP que proporcionan autenticación, integridad y confidencialidad de datos. También define los paquetes cifrados, descifrados y autenticados. Los protocolos necesarios para el intercambio seguro de claves y la gestión de claves están definidos en él.

Usos de la seguridad IP:

IPsec se puede usar para hacer lo siguiente:

- Para cifrar los datos de la capa de aplicación.
- Para brindar seguridad a los enrutadores que envían datos de enrutamiento a través de la Internet pública.
- Para proporcionar autenticación sin cifrado, como para autenticar que los datos se originan en un remitente conocido.
- Para proteger los datos de la red mediante la configuración de circuitos mediante túneles IPsec en los que todos los datos que se envían entre los dos puntos finales están encriptados, al igual que con una conexión de red privada virtual (VPN).

Componentes de Seguridad IP:

1. Carga de seguridad de encapsulación (ESP):

Proporciona integridad de datos, encriptación, autenticación y anti reproducción. También proporciona autenticación para la carga útil.

2. Encabezado de autenticación (AH):

también proporciona integridad de datos, autenticación y anti reproducción y no proporciona cifrado. La protección contra repetición, protege contra la transmisión no autorizada de paquetes. No protege la confidencialidad de los datos.



Ilustración 1 Paquete con encabezado AH

3. Intercambio de claves de Internet (IKE):

Es un protocolo de seguridad de red diseñado para intercambiar dinámicamente las claves de cifrado y encontrar un camino a través de Security Association (SA) entre 2 dispositivos. La Asociación de seguridad (SA) establece atributos de seguridad compartidos entre 2 entidades de red para admitir la comunicación segura. El Protocolo de administración de claves (ISAKMP) y la Asociación de seguridad de Internet, que proporciona un marco para la autenticación y el intercambio de claves. ISAKMP explica cómo se configuran las Asociaciones de Seguridad (SA) y cómo las conexiones directas entre dos hosts que utilizan IPsec.

Internet Key Exchange (IKE) proporciona protección de contenido de mensajes y también un marco abierto para implementar algoritmos estándar, como SHA y MD5. Los usuarios de IP sec del algoritmo producen un identificador único para cada paquete. Este identificador permite a un dispositivo determinar si un paquete ha sido correcto o no. Los paquetes que no están autorizados se descartan y no se entregan al receptor. ge la confidencialidad de los datos.

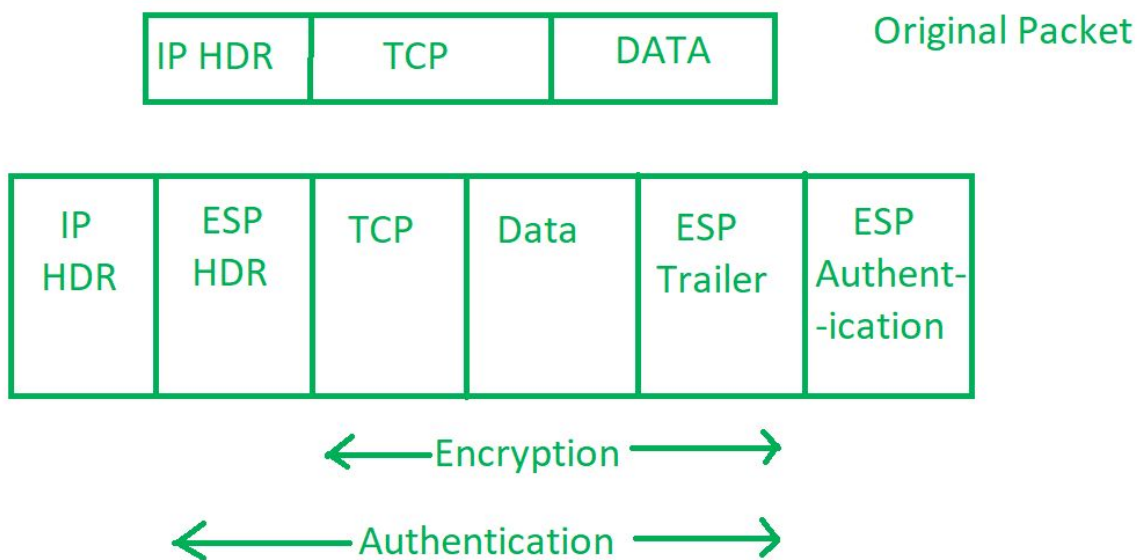
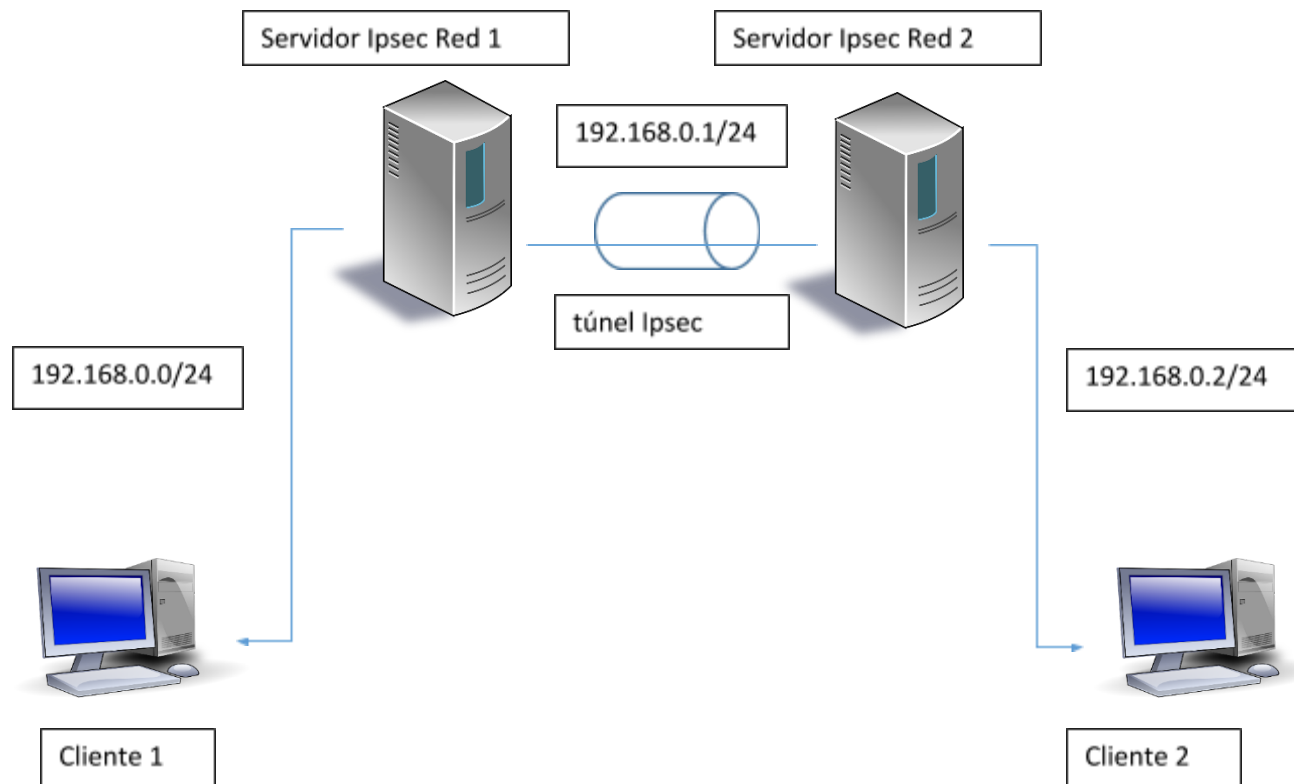


Ilustración 2 Paquete con encabezado ESP

Trabajo de Ipsec

1. El host verifica si el paquete debe ser transmitido usando IPsec o no. Este tráfico de paquetes activa la política de seguridad por sí mismos. Esto se hace cuando el sistema que envía el paquete aplica un cifrado apropiado. El host también verifica los paquetes entrantes que están encriptados correctamente o no.
2. Luego comienza la fase 1 de IKE, en la que los 2 hosts (utilizando IPsec) se autentican entre sí para iniciar un canal seguro. Tiene 2 modos. El modo Principal que proporciona mayor seguridad y el modo Agresivo que permite al host establecer un circuito IPsec más rápidamente.
3. El canal creado en el último paso se utiliza para negociar de manera segura la manera en que el circuito IP encripta los datos a través del circuito IP.
4. Ahora, la Fase 2 de IKE se lleva a cabo a través del canal seguro en el que los dos hosts negocian el tipo de algoritmos criptográficos para usar en la sesión y acuerdan el material de codificación secreta que se usará con esos algoritmos.
5. Luego, los datos se intercambian a través del túnel cifrado IPsec recién creado. Estos paquetes están cifrados y descifrados por los hosts mediante las SA de IPsec.
6. Cuando se completa la comunicación entre los hosts o se agota el tiempo de espera de la sesión, se termina el túnel IPsec descartando las claves de ambos hosts.

Hoy configuraremos una VPN ipsec de Sitio a Sitio con Strongswan, que se configurará con Autenticación de Clave Pre-Compartida.



2. Configuración del entorno de red

Vamos a realizar la instalación de 4 sistemas operativos Ubuntu Server en 4 máquinas virtuales. Configuraremos inicialmente la máquina virtual con la que vamos a trabajar de la siguiente forma:

- El tipo de sistema operativo es Ubuntu Server de 64 bits
- La memoria virtual de la máquina tiene que ser suficiente para ejecutar el entorno gráfico de instalación (mínimo 1024MB)
- Se configura la interfaz de red para:
 - Cliente 1, una interfaz red interna Red 1
 - Cliente 2, una interfaz red interna Red 2
 - Servidor 1, una interfaz red interna Red 1 y una interfaz red interna Red 3
 - Servidor 2, una interfaz red interna Red 2 y una interfaz red interna Red 3

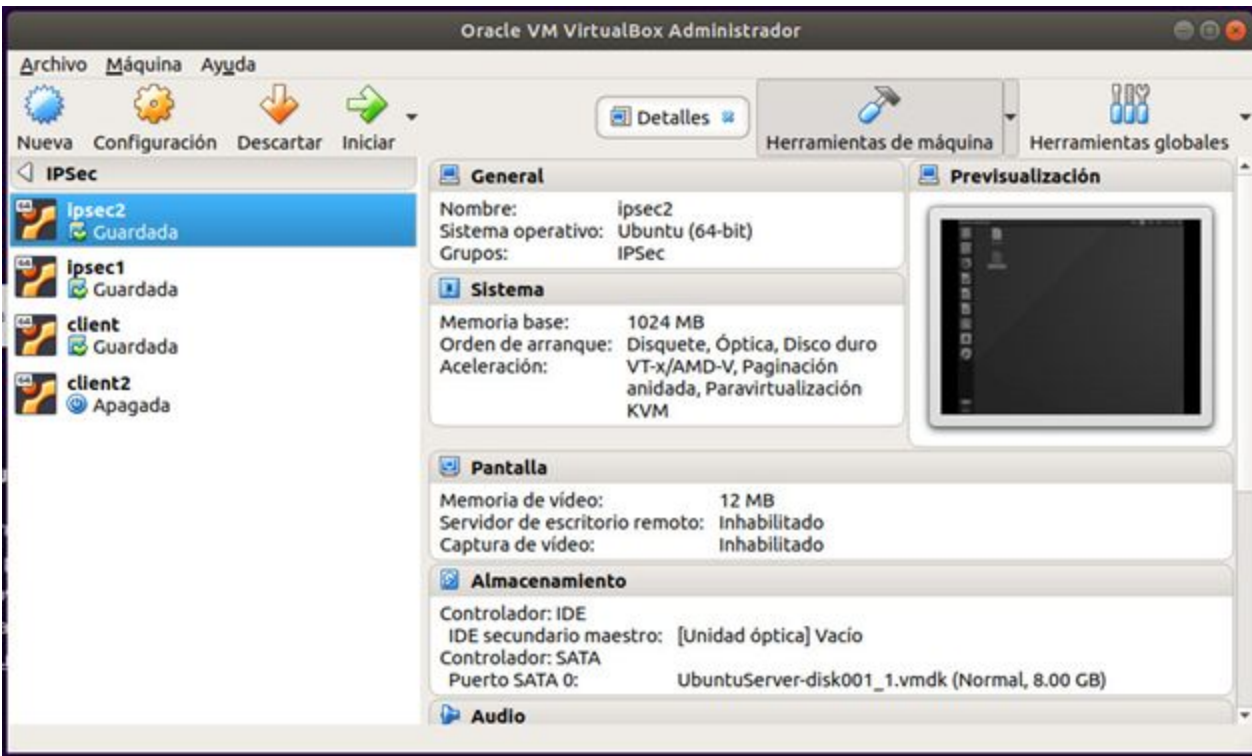


Ilustración 3 Las 4 máquinas virtuales que vamos a usar.

3. Configuración de los clientes

Cliente 1:

- a. Modificar el fichero `/etc/network/interfaces` indicando la interfaz y que se va a asignar el IP de forma manual (ver Ilustración 4).

```
root@AdR: /home/adr# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet static
address 192.168.1.10
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
root@AdR: /home/adr#
```

Ilustración 4 Configuración de interfaz de red cliente 1

- b. Una vez configurada la interfaz verificamos con `ifconfig` y `route`

```
root@AdR: /home/adr# systemctl restart networking
root@AdR: /home/adr# ifconfig
enp0s3: Link encap:Ethernet direcciónHW 08:00:27:e2:8e:26
        Direc. inet:192.168.1.10 Difus.:192.168.1.255 Másc:255.255.255.0
        Dirección inet6: fe80::a00:27ff:fee2:8e26/64 Alcance:Enlace
        ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
        Paquetes RX:12 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:133 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatX:1000
        Bytes RX:1386 (1.3 KB) TX bytes:15709 (15.7 KB)

lo: Link encap:Bucle local
        Direc. inet:127.0.0.1 Másc:255.0.0.0
        Dirección inet6: ::1/128 Alcance:Anfitrión
        ACTIVO BUCLE FUNCIONANDO MTU:65536 Métrica:1
        Paquetes RX:12200 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:12200 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatX:1
        Bytes RX:906368 (906.3 KB) TX bytes:906368 (906.3 KB)

root@AdR: /home/adr# route
Tabla de rutas IP del núcleo
Destino Pasarela Genmask Indic Métric Ref Uso Interfaz
default 192.168.1.1 0.0.0.0 UG 0 0 0 enp0s3
link-local * 255.255.0.0 U 1000 0 0 enp0s3
192.168.1.0 * 255.255.255.0 U 0 0 0 enp0s3
root@AdR: /home/adr#
```

Ilustración 5 Verificar configuración de interfaz de red cliente 1

Ciente 2:

- Modificar el fichero `/etc/network/interfaces` indicando la interfaz y que se va a asignar el IP de forma manual (ver Ilustración 6).

```
root@AdR: /home/adr
root@AdR:/home/adr# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet static
address 192.168.2.10
netmask 255.255.255.0
network 192.168.2.0
broadcast 192.168.2.255
gateway 192.168.2.1
root@AdR:/home/adr#
```

Ilustración 6 Configuración de interfaz de red cliente 2

- Una vez configurada la interfaz verificamos con `ifconfig` y `route`

```
root@AdR: /home/adr
root@AdR:/home/adr# systemctl restart networking
root@AdR:/home/adr# ifconfig
enp0s3  Link encap:Ethernet  direcciónHW 08:00:27:60:a9:5f
        Direc. inet:192.168.2.10  Difus.:192.168.2.255  Másc:255.255.255.0
        Dirección inet6: fe80::a00:27ff:fe60:a95f/64  Alcance:Enlace
        ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
        Paquetes RX:10 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:276 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colaTX:1000
        Bytes RX:1046 (1.0 KB)  TX bytes:38207 (38.2 KB)

lo      Link encap:Bucle local
        Direc. inet:127.0.0.1  Másc:255.0.0.0
        Dirección inet6: ::1/128  Alcance:Anfitrión
        ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
        Paquetes RX:9344 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:9344 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colaTX:1
        Bytes RX:695408 (695.4 KB)  TX bytes:695408 (695.4 KB)

root@AdR:/home/adr# route
Tabla de rutas IP del núcleo
Destino      Pasarela      Genmask      Indic Métric Ref      Uso Interfaz
default      192.168.2.1   0.0.0.0      UG    0      0      0 enp0s3
link-local   *             255.255.0.0  U     1000   0      0 enp0s3
192.168.2.0  *             255.255.255.0 U     0      0      0 enp0s3
root@AdR:/home/adr#
```

Ilustración 7 Verificar configuración de interfaz de red cliente 2

4. Configuración de los Servidores

Primero tenemos que generar una clave previamente compartida la cual la usarán ambos servidores : PSK 'ipsec'.

- 1) Install strongswan (sudo apt-get install strongswan)
- 2) Configuramos las interfaces en el fichero /etc/network/interfaces. (ver ilustración 8)

```
root@r1:/home/adr
root@r1:/home/adr# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet static
address 192.168.0.1
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255

auto enp0s8
iface enp0s8 inet static
address 192.168.1.1
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
root@r1:/home/adr#
```

Ilustración 8 Fichero de configuración de las interfaces

- 3) Una vez configurada la interfaz verificamos con ifconfig.

```
root@r1:/home/adr# ifconfig
enp0s3: Link encap:Ethernet direcciónHW 08:00:27:9e:54:34
        Direc. inet:192.168.0.1 Difus.:192.168.0.255 Másc:255.255.255.0
        Dirección inet6: fe80::a00:27ff:fe9e:5434/64 Alcance:Enlace
        ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
        Paquetes RX:363 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:272 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatX:1000
        Bytes RX:52802 (52.8 KB) TX bytes:29196 (29.1 KB)

enp0s8: Link encap:Ethernet direcciónHW 08:00:27:1a:f6:d8
        Direc. inet:192.168.1.1 Difus.:192.168.1.255 Másc:255.255.255.0
        Dirección inet6: fe80::a00:27ff:fe1a:f6d8/64 Alcance:Enlace
        ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
        Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:62 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatX:1000
        Bytes RX:0 (0.0 B) TX bytes:6988 (6.9 KB)

lo: Link encap:Bucle local
        Direc. inet:127.0.0.1 Másc:255.0.0.0
        Dirección inet6: ::1/128 Alcance:Anfitrión
        ACTIVO BUCLE FUNCIONANDO MTU:65536 Métrica:1
        Paquetes RX:16036 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:16036 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatX:1
        Bytes RX:1189872 (1.1 MB) TX bytes:1189872 (1.1 MB)

root@r1:/home/adr#
```

Ilustración 9 Interfaces correctamente configuradas

- 4) Configuraremos nuestra puerta de enlace VPN en el archivo de secretos /etc/ipsec.secrets en donde irá la clave precompartida e indicaremos desde donde a donde va.

```
root@r1:/home/adr# cat /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.
192.168.0.1 192.168.0.2 : PSK 'ipsec'
root@r1:/home/adr#
```

Ilustración 10 Configuración del archivo de secretos

- 5) Ahora para configurar nuestra configuración VPN en /etc/ipsec.conf

```
root@r1:/home/adr# cat /etc/ipsec.conf
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
    charondebug="all"
    uniqueids=yes
    strictcrlpolicy=no
    # strictcrlpolicy=yes
    # uniqueids = no

conn s1-a-s2
    authby=secret
    left=192.168.0.1
    leftid=192.168.0.1
    leftsubnet=192.168.1.0/24
    right=192.168.0.2
    rightsubnet=192.168.2.0/24
    ike=aes256-sha2_256-modp1024!
    esp=aes256-sha2_256!
    keyingtries=0
    ikelifetime=1h
    lifetime=8h
    dpddelay=30
    dpdtimeout=120
    dpdaction=restart
    auto=start
root@r1:/home/adr#
```

Ilustración 11 Configuración de la VPN va desde S1 a S2

6) Reglas de Firewall:

```
root@r1:/home/adr# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@r1:/home/adr# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
root@r1:/home/adr# sudo iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -d 192.168.1.0/24 -j MASQUERADE
root@r1:/home/adr# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
MASQUERADE all -- 192.168.2.0/24 192.168.1.0/24
root@r1:/home/adr#
```

Ilustración 12 Reglas firewall de Servidor 1

7) Reiniciamos el ipsec para ver que todo va bien (ipsec restart) y vemos el estado (ipsec statusall)

```
root@r1:/home/adr# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.3.5 IPsec [starter]...
root@r1:/home/adr# ipsec statusall
Status of IKE charon daemon (strongSwan 5.3.5, Linux 4.4.0-146-generic, x86_64):
  uptime: 7 seconds, since Jun 10 19:43:57 2019
  malloc: sbrk 1486848, mmap 0, used 343392, free 1143456
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, schedule d: 4
  loaded plugins: charon test-vectors aes rc2 sha1 sha2 md4 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem openssl fips-prf gmp agent xcbc hmac gcm attr kernel-netlink resolve socket-default connmark stroke updown
  Listening IP addresses:
    192.168.0.1
    192.168.1.1
  Connections:
    s1-a-s2: 192.168.0.1...192.168.0.2 IKEv1/2, dpddelay=30s
    s1-a-s2: local: [192.168.0.1] uses pre-shared key authentication
    s1-a-s2: remote: [192.168.0.2] uses pre-shared key authentication
    s1-a-s2: child: 192.168.1.0/24 === 192.168.2.0/24 TUNNEL, dpdaction=restart
  Security Associations (1 up, 0 connecting):
    s1-a-s2[1]: ESTABLISHED 7 seconds ago, 192.168.0.1[192.168.0.1]...192.168.0.2[192.168.0.2]
    s1-a-s2[1]: IKEv2 SPIs: 0d209f0d6c749531_i* fa02802453b4e5af_r, pre-shared key reauthentication in 33 minutes
    s1-a-s2[1]: IKE proposal: AES_CBC_256/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/MODP_1024
    s1-a-s2[1]: INSTALLED, TUNNEL, reqid 1, ESP SPIs: cb1389a0_i c4dc0553_o
    s1-a-s2[1]: AES_CBC_256/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 7 hours
    s1-a-s2[1]: 192.168.1.0/24 === 192.168.2.0/24
root@r1:/home/adr#
```

Ilustración 13 Verificando ipsec funcione correctamente

Para el servidor 2

- 1) Instalar strongswan (`sudo apt-get install strongswan`)
- 2) Configuramos las interfaces en el fichero `/etc/network/interfaces`. (ver ilustración 30)

```
root@r2: /home/adr
root@r2:/home/adr# nano /etc/network/interfaces
root@r2:/home/adr# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet static
address 192.168.0.2
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255

auto enp0s8
iface enp0s8 inet static
address 192.168.2.1
netmask 255.255.255.0
network 192.168.2.0
broadcast 192.168.2.255
root@r2:/home/adr#
```

Ilustración 30 Fichero de configuración de las interfaces

- 3) Una vez configurada la interfaz verificamos con `ifconfig`.

```
root@r2: /home/adr
root@r2:/home/adr# ifconfig
enp0s3  Link encap:Ethernet  direcciónHW 08:00:27:80:fa:8b
        Direc. inet:192.168.0.2  Difus.:192.168.0.255  Másc:255.255.255.0
        Dirección inet6: fe80::a00:27ff:fe80:fa8b/64 Alcance:Enlace
        ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
        Paquetes RX:35 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:91 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colTX:1000
        Bytes RX:3774 (3.7 KB)  TX bytes:10188 (10.1 KB)

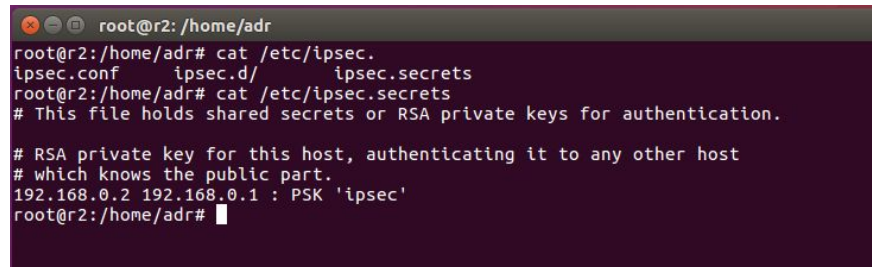
enp0s8  Link encap:Ethernet  direcciónHW 08:00:27:dd:c1:bf
        Direc. inet:192.168.2.1  Difus.:192.168.2.255  Másc:255.255.255.0
        Dirección inet6: fe80::a00:27ff:fedd:c1bf/64 Alcance:Enlace
        ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
        Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:56 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colTX:1000
        Bytes RX:0 (0.0 B)  TX bytes:6406 (6.4 KB)

lo      Link encap:Bucle local
        Direc. inet:127.0.0.1  Másc:255.0.0.0
        Dirección inet6: ::1/128 Alcance:Anfitrión
        ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
        Paquetes RX:3076 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:3076 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colTX:1
        Bytes RX:230832 (230.8 KB)  TX bytes:230832 (230.8 KB)

root@r2:/home/adr#
```

Ilustración 31 Interfaces correctamente configuradas

- 4) Configuraremos nuestra puerta de enlace VPN en el archivo de secretos `/etc/ipsec.secrets` en donde irá la clave precompartida e indicaremos desde donde a donde va.

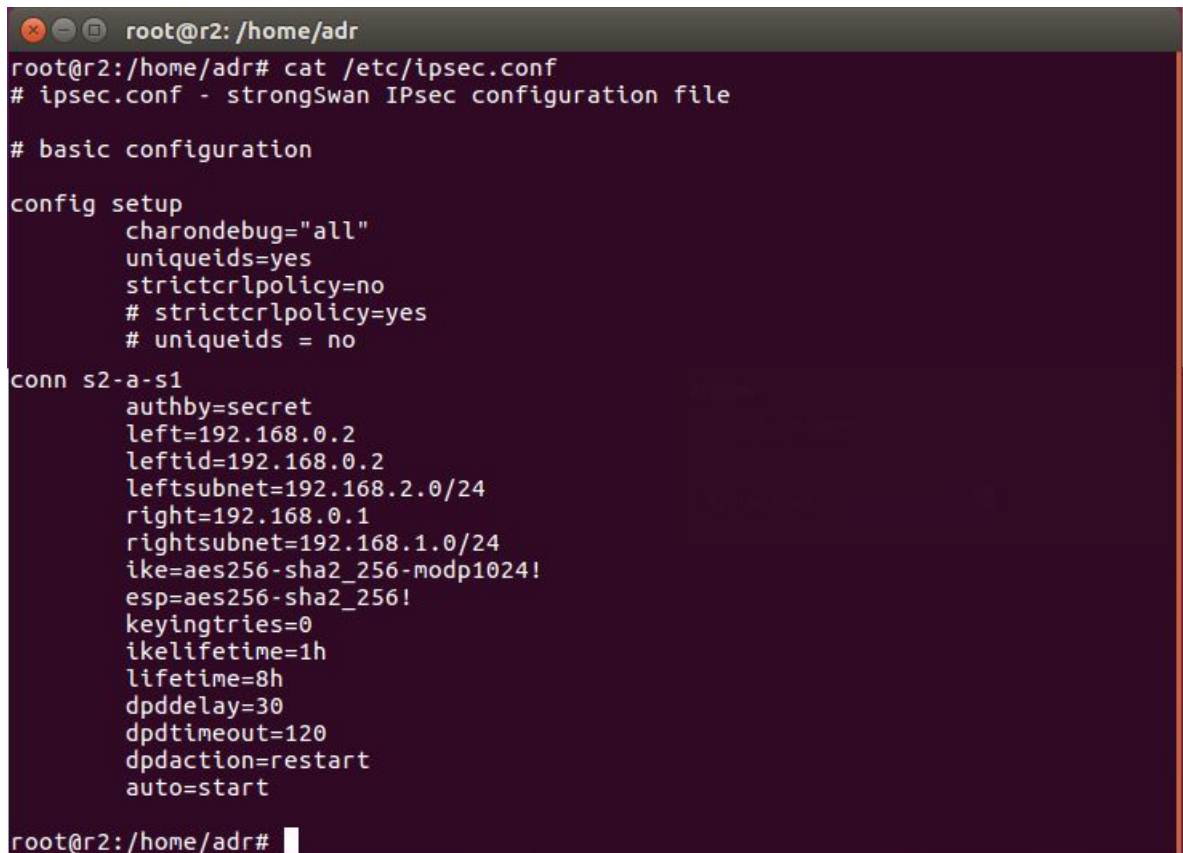


```
root@r2: /home/adr
root@r2:/home/adr# cat /etc/ipsec.secrets
ipsec.conf    ipsec.d/    ipsec.secrets
root@r2:/home/adr# cat /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.
192.168.0.2 192.168.0.1 : PSK 'ipsec'
root@r2:/home/adr#
```

Ilustración 32 Configuración del archivo de secretos

- 5) Ahora para configurar nuestra configuración VPN en `/etc/ipsec.conf`



```
root@r2: /home/adr
root@r2:/home/adr# cat /etc/ipsec.conf
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
    charondebug="all"
    uniqueids=yes
    strictcrlpolicy=no
    # strictcrlpolicy=yes
    # uniqueids = no

conn s2-a-s1
    authby=secret
    left=192.168.0.2
    leftid=192.168.0.2
    leftsubnet=192.168.2.0/24
    right=192.168.0.1
    rightsubnet=192.168.1.0/24
    ike=aes256-sha2_256-modp1024!
    esp=aes256-sha2_256!
    keyingtries=0
    ikelifetime=1h
    lifetime=8h
    dpddelay=30
    dpdtimeout=120
    dpdaction=restart
    auto=start
root@r2:/home/adr#
```

Ilustración 33 Configuración de la VPN va desde S2 a S1

6) Reglas de Firewall:

```
root@r2: /home/adr
root@r2:/home/adr# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@r2:/home/adr# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
root@r2:/home/adr# sudo iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -d 192.168.2.0/24 -j MASQUERADE
root@r2:/home/adr# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
MASQUERADE all -- 192.168.1.0/24 192.168.2.0/24
root@r2:/home/adr#
```

Ilustración 34 Reglas firewall de Servidor 2

7) Reiniciamos el ipsec para ver que todo va bien (ipsec restart) y vemos el estado (ipsec statusall)

```
root@r2: /home/adr
root@r2:/home/adr# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.3.5 IPsec [starter]...
root@r2:/home/adr# ipsec statusall
Status of IKE charon daemon (strongSwan 5.3.5, Linux 4.4.0-146-generic, x86_64):
  uptime: 7 seconds, since Jun 10 19:17:25 2019
  malloc: sbrk 1486848, mmap 0, used 343392, free 1143456
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 4
  loaded plugins: charon test-vectors aes rc2 sha1 sha2 md4 md5 random nonce x509 revocation
  constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem openssl fips-prf gmp agen
  t xcbc hmac gcm attr kernel-netlink resolve socket-default connmark stroke updown
  Listening IP addresses:
    192.168.0.2
    192.168.2.1
  Connections:
    s2-a-s1: 192.168.0.2...192.168.0.1 IKEv1/2, dpddelay=30s
    s2-a-s1: local: [192.168.0.2] uses pre-shared key authentication
    s2-a-s1: remote: [192.168.0.1] uses pre-shared key authentication
    s2-a-s1: child: 192.168.2.0/24 === 192.168.1.0/24 TUNNEL, dpdaction=restart
  Security Associations (1 up, 0 connecting):
    s2-a-s1[1]: ESTABLISHED 7 seconds ago, 192.168.0.2[192.168.0.2]...192.168.0.1[192.168.0.1]
    s2-a-s1[1]: IKEv2 SPIs: 89c0299ee4220b85_i* 4a11e4683da50e57_r, pre-shared key reauthen
    tication in 36 minutes
    s2-a-s1[1]: IKE proposal: AES_CBC_256/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/MODP_1024
    s2-a-s1[1]: INSTALLED, TUNNEL, reqid 1, ESP SPIs: c44d8bc0_i c6ae036d_o
    s2-a-s1[1]: AES_CBC_256/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 7 hours
    s2-a-s1[1]: 192.168.2.0/24 === 192.168.1.0/24
root@r2:/home/adr#
```

Ilustración 35 Verificando ipsec funcione correctamente

5. Pruebas y resultado

```
root@AdR: /home/adr
root@AdR:/home/adr# ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=62 time=2.36 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=62 time=2.51 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=62 time=2.46 ms
64 bytes from 192.168.1.10: icmp_seq=4 ttl=62 time=2.14 ms
64 bytes from 192.168.1.10: icmp_seq=5 ttl=62 time=2.36 ms
64 bytes from 192.168.1.10: icmp_seq=6 ttl=62 time=2.05 ms
64 bytes from 192.168.1.10: icmp_seq=7 ttl=62 time=1.73 ms
64 bytes from 192.168.1.10: icmp_seq=8 ttl=62 time=2.39 ms
64 bytes from 192.168.1.10: icmp_seq=9 ttl=62 time=2.00 ms
64 bytes from 192.168.1.10: icmp_seq=10 ttl=62 time=1.34 ms
^C
--- 192.168.1.10 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 1.344/2.138/2.517/0.352 ms
root@AdR:/home/adr#
```

Ilustración 35 Conexión de cliente 2 a cliente 1

```
root@r2: /home/adr
root@r2:/home/adr# tcpdump esp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
20:30:35.541429 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x1), length 136
20:30:35.542563 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x1), length 136
20:30:36.541473 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x2), length 136
20:30:36.542813 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x2), length 136
20:30:37.543654 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x3), length 136
20:30:37.545107 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x3), length 136
20:30:38.544787 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x4), length 136
20:30:38.545992 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x4), length 136
20:30:39.546987 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x5), length 136
20:30:39.548403 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x5), length 136
20:30:40.549891 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x6), length 136
20:30:40.551214 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x6), length 136
20:30:41.551338 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x7), length 136
20:30:41.552367 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x7), length 136
20:30:42.553685 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x8), length 136
20:30:42.555128 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x8), length 136
20:30:43.556134 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x9), length 136
20:30:43.557282 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x9), length 136
20:30:44.556686 IP 192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0xa), length 136
20:30:44.557474 IP 192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0xa), length 136
```

Ilustración 36 Comprobando si el tráfico por server 2 fluye a través del túnel.

```
root@AdR: /home/adr
root@AdR:/home/adr# ping 192.168.2.10
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=62 time=1.13 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=62 time=2.56 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=62 time=2.37 ms
64 bytes from 192.168.2.10: icmp_seq=4 ttl=62 time=2.39 ms
^C
--- 192.168.2.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.137/2.116/2.567/0.572 ms
root@AdR:/home/adr#
```

Ilustración 37 Conexión de cliente 1 a cliente 2

```
root@r1: /home/adr
root@r1:/home/adr# tcpdump esp -v
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
20:47:36.781347 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ESP (50), length 156)
    192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x32), length 136
20:47:36.782166 IP (tos 0x0, ttl 64, id 52655, offset 0, flags [none], proto ESP (50), length 156)
    192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x32), length 136
20:47:37.783970 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ESP (50), length 156)
    192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x33), length 136
20:47:37.785454 IP (tos 0x0, ttl 64, id 52807, offset 0, flags [none], proto ESP (50), length 156)
    192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x33), length 136
20:47:38.786291 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ESP (50), length 156)
    192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x34), length 136
20:47:38.787811 IP (tos 0x0, ttl 64, id 52829, offset 0, flags [none], proto ESP (50), length 156)
    192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x34), length 136
20:47:39.788740 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ESP (50), length 156)
    192.168.0.1 > 192.168.0.2: ESP(spi=0xcef0bb9e,seq=0x35), length 136
20:47:39.789866 IP (tos 0x0, ttl 64, id 52864, offset 0, flags [none], proto ESP (50), length 156)
    192.168.0.2 > 192.168.0.1: ESP(spi=0xcab4b035,seq=0x35), length 136
```

Ilustración 37 Comprobando si el tráfico por server 1 fluye a través del túnel.

SSL

Secure Sockets Layer (en español **capa de puertos seguros**) Es un estándar que brinda seguridad a la comunicación cliente servidor , a través del encriptado de sus datos. En encriptado se logra a través de un algoritmo matemático así como un sistema de claves solo conocidas por los implicados en la comunicación , usualmente el servidor un cliente.

Certificado ssl

Para establecer la conexión se instala un certificado ssl en el servidor este sirve como un "pasaporte" electrónico que establece las credenciales de una entidad en línea al hacer negocios en la Web. Cuando un usuario de Internet intenta enviar información de credenciales a un servidor web, el navegador del usuario accede al certificado digital del servidor y establece una conexión segura.

Un certificado SSL contiene la siguiente información:

- El nombre del titular del certificado
- El número de serie del certificado y la fecha de vencimiento
- Una copia de la clave pública del titular del certificado
- La firma digital de la autoridad que emite el certificado

¿ Para qué sirve ?

- Autenticar la identidad del sitio web, garantizando a los visitantes que no están en un sitio falso.
- Información de inicio de sesión y contraseñas.
- Información financiera (como números de tarjetas de crédito y cuentas bancarias).
- Datos personales (como nombres, direcciones, números de seguridad social y fechas de nacimiento).
- Información patentada.
- Listas de clientes.

Proceso de Inicio de la comunicación

Ejemplo Navegador accediendo a un sitio

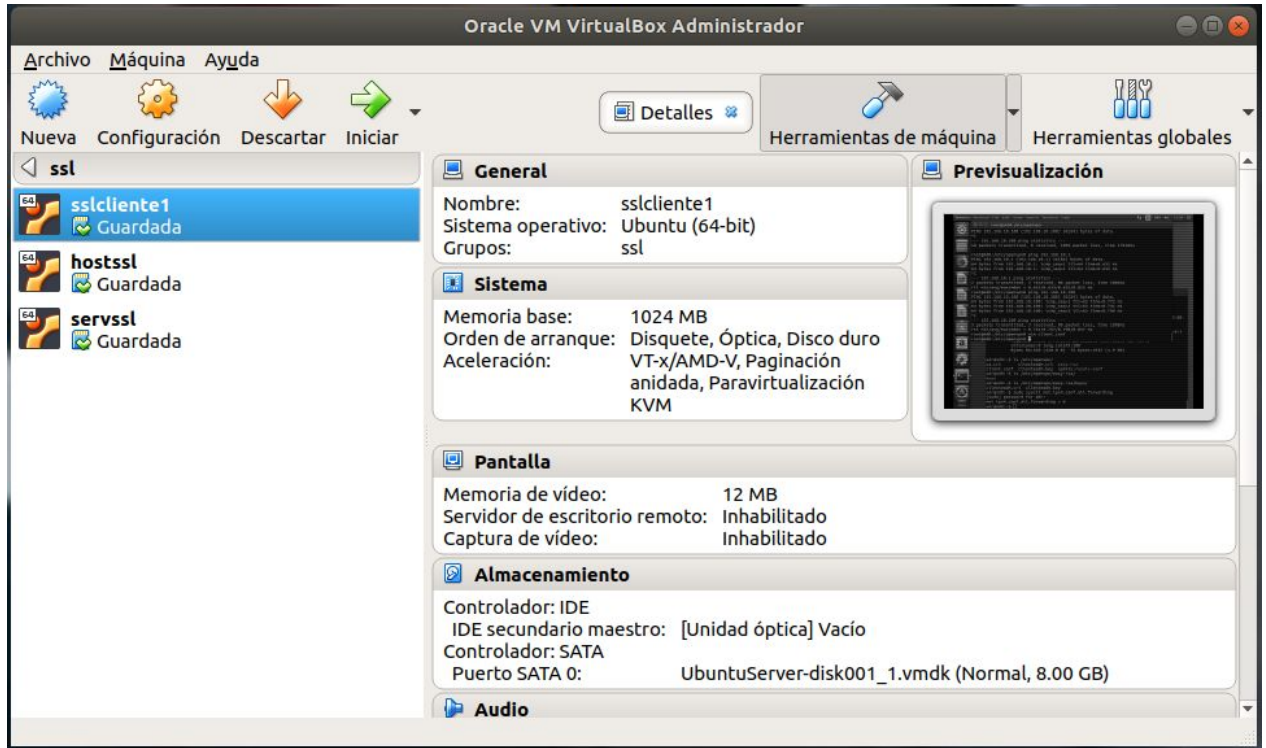


Ilustración 14

- El navegador inicia una petición al un sitio seguro (ej. Facebook).
- El sitio responde con certificado ssl con los parámetros necesarios.
- El navegador valida el Certificado verificando lo siguiente :
 - Integridad del certificado
 - Descifra la firma digital incluida en él mediante la llave pública de la AC y comparándola con una firma del certificado generada en ese momento.
 - Vigencia del certificado
 - La fecha de emisión y la fecha de expiración incluidos en él.
 - Verifica emisor del certificado
 - Hace uso de una lista de Certificados Raíz almacenados en tu computadora y que contienen las llaves públicas de las ACs conocidas y de confianza.
- En base a esto el navegador certifica la confianza del sitio.

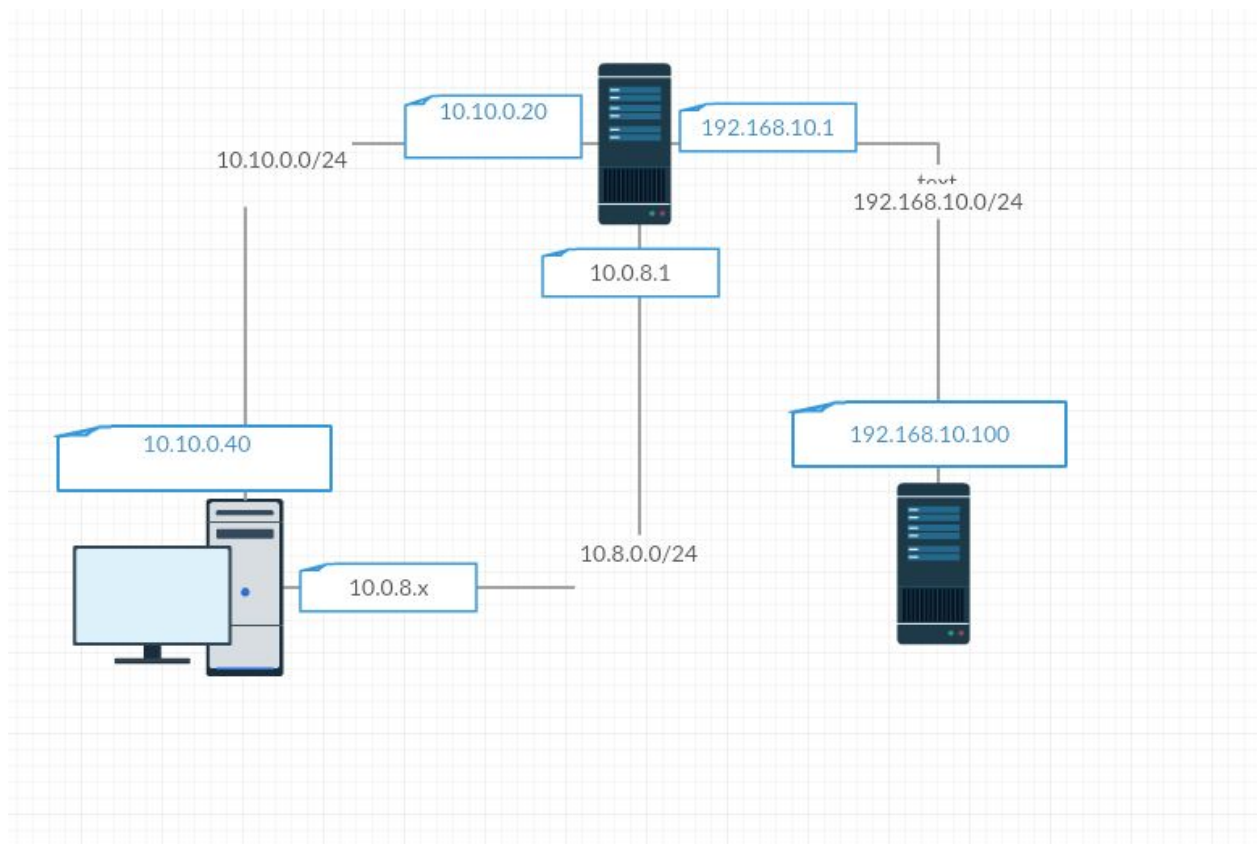
Procedimiento

1. Configuración del entorno de Red



Vamos a realizar la instalación de 3 sistemas operativos Ubuntu Server en 3 máquinas virtuales. Configuraremos inicialmente la máquina virtual con la que vamos a trabajar de la siguiente forma:

- El tipo de sistema operativo es Ubuntu Server de 64 bits
- La memoria virtual de la máquina tiene que ser suficiente para ejecutar el entorno gráfico de instalación (mínimo 1024MB)
- Se configura la interfaz de red para:
 - Cliente , una interfaz red interna Redint
 - Servidorssl, una interfaz red interna Red 1 y una interfaz red interna Redint
 - Hostssl, una interfaz red interna Red 1



red1: 192.168.10.0/24

redintermedia: 10.10.0.0/24

red para la vpn:10.8.0.0/24

Maquina host_interna 192.168.10.100

Maquina servers 192.168.10.1

10.10.0.20

Maquina cliente 10.10.0.40

2. En todas las máquinas:

1) Actualizar las máquinas:

```
sudo apt update
```

```
sudo apt full-upgrade
```

```
reboot
```

2) Instalar openvpn, easy-rsa, openssh-server:

```
sudo apt install openvpn
```

```
sudo apt install easy-rsa
```

```
sudo apt install openssh-server
```

2. Configuración del Servidor

1) A Partir de aqui ejecutar todo como root

```
sudo su
```

Creamos la carpeta:

```
mkdir /etc/openvpn/easy-rsa/
```

2) Copiamos los ficheros de ejemplo en la carpeta creada

```
cp -r /usr/share/easy-rsa/* /etc/openvpn/easy-rsa/
```

3) Editamos vars según se desee (ilustración 1)

```
/etc/openvpn/easy-rsa/vars
```

```
# These are the default values for fields
# which will be placed in the certificate.
# Don't leave any of these fields blank.
export KEY_COUNTRY="PE"
export KEY_PROVINCE="LI"
export KEY_CITY="lima"
export KEY_ORG="unissl"
export KEY_EMAIL="adm@unissl"
export KEY_OU="MyOrganizationalUnit"

# X509 Subject Field
export KEY_NAME="uni"
```

ilustración 1 (cat /etc/openvpn/easy-rsa/vars)

4) Creamos los certificados de autoridad(CA)

```
source /etc/openvpn/easy-rsa/vars
```

```
./etc/openvpn/easy-rsa/clean-all
```

```
./etc/openvpn/easy-rsa/build-ca
```

5) Se creará la carpeta keys, en donde se genera el CA(ca.crt) en caso que no se cree crearla manualmente

```
mkdir /etc/openvpn/easy-rsa/keys
```

6)Luego volver a ejecutar los tres comandos anteriores creamos los certificados y claves privadas para el servidor.

```
./etc/openvpn/easy-rsa/build-key-server servidor
```

Hará preguntas sobre las variables modificadas en vars, saltar por defecto con enter

Luego preguntará "Sign the certificate? [y/n]" responder "y" luego "1 out of 1 certificate requests certified, commit? [y/n]" responder "y" también

Se crean los ficheros servidor.crt server.key en la carpeta keys

7) Generamos los parámetros Diffie Hellman para el servidor

```
./etc/openvpn/easy-rsa/build-dh
```

se creará el fichero dh2048.pem en keys

8) Ahora copiaremos todos estos ficheros generados a la carpeta de openvpn

```
cd /etc/openvpn/easy-rsa/keys/
```

```
cp server.crt server.key ca.crt dh2048.pem /etc/openvpn/
```

9) Creamos los certificados para los clientes

```
source /etc/openvpn/easy-rsa/vars
```

```
./etc/openvpn/easy-rsa/build-key clienteadm
```

```
root@AdR:/home/adr# ls -ltr /etc/openvpn/
total 48
-rwxr-xr-x 1 root root 1301 may  8 08:50 update-resolv-conf
-rw-r--r-- 1 root root 1708 jun 11 23:31 server.key
-rw-r--r-- 1 root root 424 jun 11 23:31 dh2048.pem
-rw-r--r-- 1 root root 1818 jun 11 23:31 ca.crt
-rw-r--r-- 1 root root 5728 jun 12 02:06 server.crt
-rw-r--r-- 1 root root 10484 jun 12 11:13 server.conf
drwxr-xr-x 3 root root 4096 jun 12 11:49 easy-rsa
-rw-r--r-- 1 root root 20 jun 12 13:14 ipp.txt
-rw-r--r-- 1 root root 359 jun 12 13:14 openvpn-status.log
root@AdR:/home/adr# ls -ltr /etc/openvpn/easy-rsa/keys/
total 84
-rw-r--r-- 1 root root 1704 jun 11 23:16 ca.key
-rw-r--r-- 1 root root 1818 jun 11 23:16 ca.crt
-rw-r--r-- 1 root root 1708 jun 11 23:19 server.key
-rw-r--r-- 1 root root 1098 jun 11 23:19 server.csr
-rw-r--r-- 1 root root 3 jun 11 23:19 serial.old
-rw-r--r-- 1 root root 149 jun 11 23:19 index.txt.old
-rw-r--r-- 1 root root 21 jun 11 23:19 index.txt.attr.old
-rw-r--r-- 1 root root 5728 jun 11 23:19 server.crt
-rw-r--r-- 1 root root 5728 jun 11 23:19 01.pem
-rw-r--r-- 1 root root 424 jun 11 23:23 dh2048.pem
-rw-r--r-- 1 root root 1704 jun 11 23:36 clienteadm.key
-rw-r--r-- 1 root root 1106 jun 11 23:36 clienteadm.csr
-rw-r--r-- 1 root root 3 jun 11 23:39 serial
-rw-r--r-- 1 root root 21 jun 11 23:39 index.txt.attr
-rw-r--r-- 1 root root 302 jun 11 23:39 index.txt
-rw-r--r-- 1 root root 5626 jun 11 23:39 clienteadm.crt
```

ilustración 2 (Directorios de servidor)

10) Ahora copiamos los ficheros generados y ca.crt a la máquina cliente

```
scp clienteadm.crt adr@10.10.0.40:~  
scp clienteadm.key adr@10.10.0.40:~  
scp ca.crt adr@10.10.0.40:~  
ssh adr@10.10.0.40  
sudo su  
source /etc/openvpn/vars  
cp ca.crt /etc/openvpn/  
cp clienteadm.crt clienteadm.key /etc/openvpn/easy-rsa/keys/  
exit
```

11) Ahora configuraremos el servidor

```
cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/openvpn/  
sudo gzip -d /etc/openvpn/server.conf.gz  
vim /etc/openvpn/server.conf
```

12) Verificamos que las siguientes líneas coinciden con nuestros ficheros generados para el servidor (ilustración 3)


```
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys. Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ca.crt
cert server.crt
key server.key # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
# openssl dhparam -out dh2048.pem 2048
dh dh2048.pem
```

ilustración 3 (ficheros generados para el servidor)

13) Verificamos también la ip por la cual debe escuchar openvpn (ilustración 4)

```
# Comments are preceded with '#' or ';'
#####

# Which local IP address should OpenVPN
# listen on? (optional)
local 10.10.0.20
```

ilustración 4

14) verificamos que red le vamos a otorgar a la vpn (ilustración 5)

```
# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0
```

ilustración 5 (asignación de vpn)

15) finalmente verificamos con que subredes internas a las que se podrá comunicar el cliente vpn (ilustración 6)

```
# Push routes to the client to allow it
# to reach other private subnets behind
# the server. Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.10.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"
push "route 192.168.10.0 255.255.255.0"
```

ilustración 6 (subredes internas se pueden comunicar con el cliente)

16) Activamos el forwardin para ipv4

```
sysctl net.ipv4.conf.all=1
```

17) Habilitamos la llegada de upd por el puerto 1194 en caso no lo permita el firewall

```
iptables -A INPUT -i enp0sx -p udp -m state --state NEW -m udp --dport 1194 -j ACCEPT
```

enp0sx es la interfaz que tiene como IP a 10.10.0.20

18) Reiniciamos el servicio

```
/etc/init.d/openvpn restart
```

19) Verificamos que no tenga errores:

```
systemctl status openvpn@server
```

20) Observamos que se crea tun0 con ifconfig (ilustración 7)

```
tun0      Link encap:UNSPEC  direcciónHW 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
00-00-00
        Direc. inet:10.8.0.1  P-t-P:10.8.0.2  Másc:255.255.255.255
        ACTIVO PUNTO A PUNTO FUNCIONANDO NOARP MULTICAST  MTU:1500  Métrica:1
        Paquetes RX:30 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:32 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colaTX:100
        Bytes RX:2520 (2.5 KB)  TX bytes:2688 (2.6 KB)
```

ilustración 7 (ifconfig tun0)

3. Configuración de los clientes

En la máquina del cliente

1) Ingresar como root

```
sudo su
```

2) Copiar el fichero de ejemplo de configuración cliente como se muestra

```
cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/openvpn/
```

3) En el fichero client.conf verificamos que coincidan los certificados generados (

ilustración 8)

```
vim /etc/openvpn/client.conf
```

```
# SSL/TLS parms.
# See the server config file for more
# description. It's best to use
# a separate .crt/.key file pair
# for each client. A single ca
# file can be used for all clients.
ca ca.crt
cert clienteadm.crt
key clienteadm.key
```

ilustración 8(certificados generados)

5) Verificamos que tiene correctamente asignado el servidor (ilustración 9)

```
# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 10.10.0.20 1194
;remote my-server-2 1194
```

ilustración 9 (Verificación de servidor)

6) Reiniciamos el servicio.

```
systemctl restart openvpn
```

7) Verificamos que no tenga errores.

```
systemctl status openvpn@cliente
```

8) Observamos que se cree tun0 en ifconfig(ilustración 10)

```
tun0      Link encap:UNSPEC  direcciónHW 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
00-00-00
   Direc. inet:10.8.0.6  P-t-P:10.8.0.5  Másc:255.255.255.255
   ACTIVO PUNTO A PUNTO FUNCIONANDO NOARP MULTICAST  MTU:1500  Métrica:1
   Paquetes RX:12 errores:0 perdidos:0 overruns:0 frame:0
   Paquetes TX:30 errores:0 perdidos:0 overruns:0 carrier:0
   colisiones:0 long.colaTX:100
   Bytes RX:1008 (1.0 KB)  TX bytes:2520 (2.5 KB)
```

(Se le asigna la IP 10.8.0.6)

8) Verificamos si puede hacer ping al servidor openvpn(ilustración 11)

```
ping 10.8.0.1
```

```
root@AdR:/etc/openvpn# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.551 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.546 ms
```

ilustración 11 (ping al servidor openvpn)

9) Veamos qué rutas se han (ilustración 12)

route

```
root@AdR:/etc/openvpn# route
```

Tabla de rutas IP del núcleo

Destino	Pasarela	Genmask	Indic	Métric	Ref	Uso	Interfaz
default	10.0.3.2	0.0.0.0	UG	100	0	0	enp0s8
10.0.3.0	*	255.255.255.0	U	100	0	0	enp0s8
10.8.0.1	10.8.0.5	255.255.255.255	UGH	0	0	0	tun0
10.8.0.5	*	255.255.255.255	UH	0	0	0	tun0
10.10.0.0	*	255.255.255.0	U	0	0	0	enp0s3
link-local	*	255.255.0.0	U	1000	0	0	enp0s3

ilustración 12 (route)

Maquina host interna:

1) Solamente tenemos que agregar la red que crea la vpn.

```
sudo route add -net 10.8.0.0/24 gw 192.168.10.1
```

4. Pruebas

1) Ahora podemos comprobar que la máquina cliente puede hacer ping a la máquina host interna (ilustración 13)

```
ping 192.168.10.100
```




```
root@AdR:/etc/openvpn# ping 192.168.10.100
PING 192.168.10.100 (192.168.10.100) 56(84) bytes of data.
64 bytes from 192.168.10.100: icmp_seq=1 ttl=63 time=0.772 ms
64 bytes from 192.168.10.100: icmp_seq=2 ttl=63 time=0.741 ms
```

ilustración 13 (ping a 192.168.10.100 host interno)

11) También podemos comprobar que la máquina host interna puede hacer ping a la máquina cliente (ilustración 14)

ping 10.8.0.6

```
adr@AdR:~$ ping 10.8.0.6
PING 10.8.0.6 (10.8.0.6) 56(84) bytes of data.
64 bytes from 10.8.0.6: icmp_seq=1 ttl=63 time=3.48 ms
64 bytes from 10.8.0.6: icmp_seq=2 ttl=63 time=0.825 ms
64 bytes from 10.8.0.6: icmp_seq=3 ttl=63 time=0.742 ms
```

Ilustración 14 (ping a 10.8.06 cliente)

5. Bibliografía

- <https://es.wikipedia.org/wiki/IPsec>
- <https://www.redeszone.net/redes/openvpn/>
- https://es.wikipedia.org/wiki/Red_privada_virtual
- http://laurel.datsi.fi.upm.es/proyectos/teldatsi/teldatsi/protocolos_de_comunicaciones/pro_tocolo_ipsec
- <https://www.geeksforgeeks.org/computer-network-ip-security-ipsec/>
- <https://techclub.tajamar.es/presentacion-de-ipsec/>
- <https://wiki.strongswan.org/projects/strongswan/wiki/IpssecSecrets>
- <https://wiki.strongswan.org/projects/strongswan/wiki/IpssecConf>
- <https://www.slashroot.in/ssl-openvpn-linux-installation-and-configuration>
- <https://www.liquidweb.com/kb/ssl-vs-tls/>

- <https://revista.seguridad.unam.mx/numero-10/el-cifrado-web-sslts>