



# Anexo 1: VPN

## Redes Privadas Virtuales (VPN)

Extensión de la red privada:  
usa medio público para conectar de forma **segura** a usuarios remotos (internet)



- Transporte info → técnicas de encapsulación (**tunneling**)
- conectividad con **host**, **asym. PPP**, **lín. de acceso** o **redes WAN dedicadas**

## Escenarios de VPNs

### Branch Offices e Delegaciones

site-to-site

• redadas geográficamente

• VPN se configura

entre los **GATEWAY**

9A → PB

• **rutadores VPN**

### Road Warriors o Usuarios Móviles

acceso Remoto

• A → B → Red Remota

• VPN se configura

entre el **usuario** y la **Red Universitaria**

### Host to host

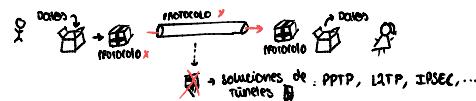
A → A

• **extremo a extremo entre**

los **usuarios finales**

## Túneles

- Tranfieren **datos** de un **Protocolo** encapsulados dentro del **Protocolo** de otro **Protocolo** (el mismo)
- Túnel proporciona **capacidad de transporte** de **datos** q NO deben moverse DIRECTAMENTE por la red



### Nivel 2 (entrañate de datos)

Transm. Tramas

### Típos de Túneles

• Encapsulado sobre IP

### Nivel 3 (capa de Red)

Transm. de IP

### TIPOS DE TÚNELES

• Encapsulado sobre IP

### Túnel de nivel 2

### Túnel de nivel 3

### Túnel de nivel 2

### Túnel de nivel 3

## Requisitos Básicos de la VPN

Autenticación de usuarios bloquear accesos no autorizados

Gestión de direcciones establecer IP en la que viene

Red Remota

Cifrado de datos

Admin. de claves jerarquismo para renovar claves usadas en el intercambio

Soporte multiprotocolo si se requiere encapsulamiento de otro tipo de tráfico, además de IP (IPX, NetBIOS, ...)

## Otras soluciones de VPN

SSTP → Secure Socket Tunneling Protocol, encapsula conexiones PPP sobre SSL/TLS

OpenVPN → VPN basada en SSL/TLS. Autenticación → clave compartida o estandarizada. No estandarizada.

WireGuard → Solución VPN robusta con criptografía routing

Túneles SSH → comunicación mediante redirecc. puertos TCP/UDP

Túneles SSL → secuenciar cualquier conexión TCP (stunnel)

Servicios Web SSL/TLS (clientless VPN) → permiten establecer túneles, redireccionar aplicaciones una vez establecida una conexión HTTP

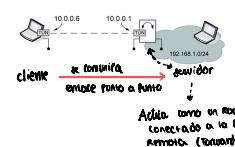
WebVPN (Cisco), Adito/OpenVPN ALS /SSL Exploit

## Open VPN

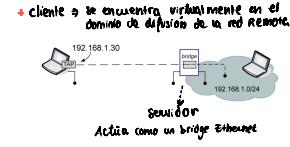
- OpenSource + Multiplataforma (Linux, Mac, Android, Windows)
- Simple & Flexible (funciones UDP & TCP - 1194-, portada NAT, opera N3 ...)
- No es normalizada, utiliza OpenSSL (SSL/TLS)
- Autenticación → clave compartida (PSK)
- Modos de Operación → N3 como bridge ethernet (interfaces TAP)
- N3 con enlaces punto a punto y router (interfaz TUN)

## Modo de Operación (TUN / TAP)

### modo TUN (tunneles)



### modo TAP



## Instalación Linux

- Se instala paquete → Cliente openvpn
- Dependencias → LZO
- Archivos config → long  
/etc/openvpn/server → CA  
/etc/openvpn/client → Client
- Archivos de ejemplo: /usr/share/doc/openvpn-2.4.0/
- Gestión certif. autenticación → RSA, easy-rsa
- easy-rsa vs  
Gestión de certificados para la autenticación
- scripts para la gestión en marcado de una PKI  
creación CA → generación claves...  
autenticación de certificados
- P: ./script/easyrsa-req-3  
→ script "easyrsa", realiza tareas gestión CA
- Archivos de nuestro CA  
↳ ./openssl-1.0.2.cnf



# Anexo 1: VPN

## (Open VPN)

### Configuración de una PKI

- Crearemos una nueva CA que estará situada en un nuevo directorio **pki** bajo la ubicación donde nos encontramos (ojo: al inicializar se sobrescribe todo el contenido de la pki) :

```
cd ~
/usr/share/easy-rsa/3/easyrsa init-pki
/usr/share/easy-rsa/3/easyrsa build-ca
/usr/share/easy-rsa/3/easyrsa gen-dh
```

(obtenemos ca.crt, ca.key y dh.pem)

Lleva bastante tiempo

- Creamos claves y certificados para el servidor y los clientes:

```
/usr/share/easy-rsa/3/easyrsa build-server-full servidor no
/usr/share/easy-rsa/3/easyrsa build-client-full cliente-01
no pass
```

→ para el Servidor  
→ para un Cliente

- Creamos el listado de certificados revocados

```
/usr/share/easy-rsa/3/easyrsa gen-crl
```

### Archivos obtenidos

Archivo	Contenido	Utilizado por	Secreto
ca.key	Clave privada de la CA	Equipo de generación de certificados	Si
ca.crt	Certificado raíz de la CA	Servidor y todos los clientes	No
dh.pem	Parámetros Difile Hellman	Servidor	No
crl.pem	Certificados revocados	Servidor	No
servidor.crt	Certificado del servidor	Servidor	No
servidor.key	Clave priv. del servidor	Servidor	Si
cliente-01.crt	Certificado de cliente1	Cliente1	No
cliente-01.key	Clave priv. de Cliente1	Cliente1	Si

### Configuración y Arranque del servicio

- Optional: Generar una **pre-shared key** para aumenta la seguridad. De usarlo, deberá estar en ambos extremos.

```
openvpn --genkey secret ta.key
```

→ secreto compartido

- Movemos los archivos a los directorios server y client. Arrancamos los servicios con systemctl indicando a continuación de la @ el nombre del archivo de configuración usado en cada lado (sin la extensión .conf). Por ejemplo, si usámos server.conf y cliente.conf, las llamadas serían:

```
systemctl start openvpn@server
systemctl start openvpn@cliente
```

Para ver los logs: journalctl -u openvpn-server@servidor. Ojo: directiva verb (4 o 5)  
También podemos ejecutar en foreground openvpn < fichero\_configuración.conf>

### Configuración modo TUN

**SERVIDOR**

```
port 1194
proto udp
dev tun
ca ca.crt → ubicación archivo ca
cert servidor.crt
key servidor.key
dh dh.pem
```

server 192.0.0.255.255.255.0 → Direcc. q va a manejar mi red virtual  
listen-tcp-port 1194  
push route 192.168.10.0 255.255.255.0 → Requerido !!  
keepalive 10 120

tls-auth ta.key 0  
cipher AES-256-CBC

persist-key  
persist-tun ↑ persistencia con claves  
y con conexión  
Lo sirve de persistencia, intenta reseñaladecula.

**CLIENTE**

```
client
dev tun
proto udp
remote myserver.com 1194 → Red
resolv-retry infinite
persist-key
persist-tun
```

ca ca.crt  
cert client.crt  
key client.key  
remote-cert-tls server → otro mecanismo más de seguridad  
tls-auth ta.key 1  
cipher AES-256-CBC

→ tiene q ver el = q el servidor

### IPtables (post-forwarding y enmascaramiento)

- Port-forwarding (nat en PREROUTING). Ejemplo: las conexiones recibidas en el puerto 80/tcp de eth1 se reenvian al mismo puerto del equipo 192.168.1.100

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 -j DNAT --to-destination 192.168.1.100:80
```

- Enmascaramiento (nat en POSTROUTING). Ejemplo: Enmascara todo el tráfico que sale por la interfaz eth1

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

- Persistencia de las reglas (iptables-save y iptables-restore). Salvamos las reglas en un archivo de configuración

```
iptables-save > /etc/iptables.rules
iptables-restore < /etc/iptables.rules
```

**IPtables - persistente**

Requer. Para q no se borrar los ipables al reiniciar

