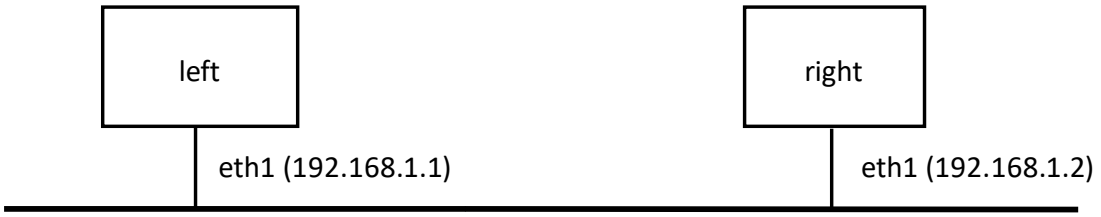
Seguridad en Redes

Práctica 3.6. OpenVPN y OpenSSH

Preparación del entorno

Vamos a usar dos MVs (`left` y `right`) conectadas a una red interna (`internet`).



Importa una MV, haz una clonación enlazada y añade una interfaz de red a cada MV conectado a la red correspondiente.

Configura `left`:

```
sudo ifconfig eth1 192.168.1.1/24 up
sudo apt-get update
sudo apt-get install openvpn ssh
```

Configura `right`:

```
sudo ifconfig eth1 192.168.1.2/24 up
sudo apt-get update
sudo apt-get install openvpn ssh
```

Para instalar una versión más reciente de Wireshark, con capacidad para analizar los paquetes de OpenVPN, añade la siguiente línea al fichero `/etc/apt/sources.list` de `left`:

```
deb http://ftp.es.debian.org/debian wheezy-backports main
```

Después, ejecuta:

```
$ sudo apt-get update
$ sudo apt-get -t wheezy-backports install wireshark
```

OpenVPN

Consulta la página de manual del comando `openvpn`.

Uso de una clave estática previamente compartida

Crea una clave compartida en `left` y cópiala a `right`:

```
$ /usr/sbin/openvpn --genkey --secret static.key
$ scp static.key usuario@192.168.1.2:.
```

Configura e inicia la VPN en `left`:

```
$ sudo openvpn --remote 192.168.1.2 --dev tun --ifconfig
10.4.0.1 10.4.0.2 --secret static.key
```

Y en `right`:

```
$ sudo openvpn --remote 192.168.1.1 --dev tun --ifconfig
10.4.0.2 10.4.0.1 --secret static.key –-verb 4
```

Para obtener más información del proceso se añade la opción `--verb 4`.

- **Entrega:** Copia la salida del comando anterior en `right` desde donde se especifican los algoritmos de cifrado y autenticación hasta que se vea que se ha establecido la conexión.

Observa, con `sudo ifconfig -a`, que se ha creado una interfaz virtual (`tun0`).

- **Entrega:** Copia las características de la interfaz `tun0` que nos devuelve el comando ejecutado.

Comprueba que la VPN funciona. Para ello, abre otro terminal en `left` y ejecuta el wireshark, confíguralo para capturar los paquetes por la interfaz `eth1`. Después abre otro terminal y haz:

```
$ ping –c 4 10.4.0.2
```

¿Están cifrados los mensajes capturados? Cambia la interfaz de captura de wireshark a `tun0` y vuelve a ejecutar el `ping`. ¿Qué mensajes se capturan ahora?

- **Entrega:** Responde las preguntas planteadas.

Los parámetros de la VPN también se pueden especificar en ficheros como los siguientes:

`left.conf`	`right.conf`

`remote 192.168.1.2`
`dev tun`
`ifconfig 10.4.0.1 10.4.0.2`
`secret static.key`
	`remote 192.168.1.1`
	`dev tun`
	`ifconfig 10.4.0.2 10.4.0.1`
	`secret static.key`

Cierra la conexión establecida anteriormente (`Control C`) en las dos máquinas. Crea el fichero `left.conf` en `left` y ejecuta:

```
$ sudo openvpn left.conf
```

Y crea el fichero `right.conf` en `right` y ejecuta:

```
$ sudo openvpn right.conf
```

Comprueba de nuevo que la VPN funciona y se ha creado la interfaz virtual `tun0`.

Uso de TLS con certificados

Copia los certificados y claves de prueba proporcionados por OpenVPN en `left`:

```
$ cp /usr/share/doc/openvpn/examples/sample-
keys/{client.*,ca.crt} .
$ gunzip client.crt.gz
```

Y en `right`:

```
$ cp /usr/share/doc/openvpn/examples/sample-
keys/{server.*,ca.crt,dh1024.pem} .
$ gunzip server.crt.gz
```

`client.crt` y `server.crt` son los certificados de cliente y servidor (desde el punto de vista de TLS, no de OpenVPN), `client.key` y `server.key` son las respectivas claves privadas, `ca.crt` es el certificado de la CA que firmó los certificados y `dh1024.pem` contiene los parámetros Diffie-Hellman.

Configura e inicia la VPN en `left`:

```
$ sudo openvpn --remote 192.168.1.2 --dev tun --ifconfig
10.4.0.1 10.4.0.2 --tls-client --ca ca.crt --cert client.crt --key
client.key
```

Y en `right`:

```
$ sudo openvpn --remote 192.168.1.1 --dev tun --ifconfig
10.4.0.2 10.4.0.1 --tls-server --dh dh1024.pem --ca ca.crt --cert
server.crt --key server.key –verb 4
```

- **Entrega:** Copia la salida del comando anterior en `right` desde la línea que contiene `Diffie-Hellman initialized with 1024 bit key`.

La configuración anterior es vulnerable a ataques de intermediario o de suplantación del servidor (https://openvpn.net/howto.html#mitm). Entre otras cosas, el cliente debería asegurarse de que se le presenta un certificado de servidor (con `--remote-cert-tls server`) y de que el nombre (*common name*, CN) coincide con el que aparece en el certificado del servidor (con `--tls-remote Test-Server`).

Modifica los archivos de configuración (`left.conf` y `right.conf`) para que el túnel utilice TLS con certificados y el cliente verifique el certificado del servidor (añadiendo `--remote-cert-tls server` y `--tls-remote Test-Server`).
Reinicia la captura con el wireshark por la interfaz `eth1` (Capture -> Restart). Cierra la conexión OpenVPN abierta y vuelve a abrirla a partir de los archivos de configuración definidos.

- **Entrega:** Copia el contenido de los archivos `left.conf` y `right.conf` que has usado para crear la nueva VPN.

Observa el tráfico con Wireshark. Los tipos de mensajes intercambiados de OpenVPN son:

- P_CONTROL_HARD_RESET_CLIENT_V2: Notifica el reinicio del cliente.
- P_CONTROL_HARD_RESET_SERVER_V2: Notifica el reinicio del servidor.
- P_CONTROL_V1: Mensaje de control (normalmente, son mensajes TLS).
- P_ACK_V1: Confirmación de la recepción de los mensajes de control (OpenVPN proporciona una capa de fiabilidad sobre UDP para estos mensajes).

Como la autenticación se ha hecho con certificados TLS, entre estos mensajes hay otros del protocolo TLSv1 (usa “ssl” como filtro en Wireshark para verlos más fácilmente).

- **Entrega:** Describe los mensajes intercambiados en el acuerdo TLS, indicando lo qué hace cada mensaje.
- **Entrega:**
  - ¿Cuántos conjuntos de algoritmos le ofrece el cliente al servidor para elegir? Enumera 4 de ellos.
  - ¿Qué algoritmos de firma, intercambio de clave, cifrado, autenticación de mensajes acuerdan usar las partes?
  - ¿Qué envía el cliente en la parte denominada “Certificate”?

Es posible que Wireshark no interprete correctamente algunos mensajes TLS.

Comprueba que la conexión OpenVPN funciona haciendo en `left` un `ping 10.4.0.2`. Los mensajes capturados en el wireshark ahora están cifrados y son de tipo:

- P_DATA_V1: Mensaje de datos (en este caso, los paquetes IP de la conexión lógica).

OpenSSH

Autenticación con clave pública

Ejecuta `ssh-keygen` en `left` seleccionando la ubicación por defecto (/home/usuario/.ssh/id_rsa) para la clave privada y proporciona una contraseña para la misma. Se creará un fichero `~/.ssh/id_rsa` con la clave privada y un fichero `~/.ssh/id_rsa.pub` con la clave pública correspondiente.

Añade el contenido del fichero de clave pública al fichero `~/.ssh/authorized_keys` de `right` (el fichero `id_rsa.pub` debe tener permisos `600`) ejecutando en `left`:

```
$ ssh-copy-id 192.168.1.2
```

Accede con `ssh` a `right` desde `left`:

```
$ ssh 192.168.1.2
```

Pedirá la contraseña para descifrar la clave privada, pero la autenticación se realizará mediante la clave pública. Para ver el proceso de autenticación, usa la opción `-v` (*verbose*).

- **Entrega:** Copia la salida del comando anterior con la opción `-v`.

Para escribir la contraseña de la clave privada una sola vez, ejecuta en `left`:

```
$ eval `ssh-agent`
$ ssh-add
```

Reenvío de puertos

La opción `-L port:host:hostport` especifica el reenvío de un puerto **local** a un destino fijo. Es decir, las conexiones al puerto `port` del cliente SSH, se reenvían sobre el canal seguro establecido con el servidor y de ahí, ya sin seguridad, hasta el puerto `hostport` de `host`.

En `left`, establece un túnel SSH con `right` en el puerto 8080 de `left` y desde `right` al puerto 80 de `www.ucm.es`:

```
$ ssh -v -N -L 8080:www.ucm.es:80 usuario@192.168.1.2
```

La opción `-N` evita que se lance una sesión interactiva de *shell* remota, lo cual es útil si solo se quiere establecer el reenvío de puertos. También se suele usar la opción `-f`, que hace que el comando `ssh` se desasocie del terminal y pase a segundo plano.

Accede en `left` a http://localhost:8080 con el navegador.

- **Entrega:** Copia la salida del comando `ssh`. ¿A qué página se ha conectado el navegador?

La opción `-R port:host:hostport` especifica el reenvío de un puerto **remoto** a un destino fijo. Es decir, las conexiones al puerto `port` del servidor SSH, se reenvían sobre el canal seguro establecido con el cliente y de ahí, ya sin seguridad, hasta el puerto `hostport` de `host`.

En `left`, establece un túnel SSH con `right` en el puerto 8080 de `right` y desde `left` al puerto 80 de `www.ucm.es`, iniciando remotamente un navegador:

```
$ ssh -v -X -R 8080:www.ucm.es:80 usuario@192.168.1.2 chromium
```

La opción `-X` permite el reenvío de conexiones X11 (interfaz gráfica).

Accede a http://localhost:8080 con el navegador iniciado remotamente.

- **Entrega:** Copia la salida del comando `ssh` desde que introduces la contraseña de la clave. ¿En qué máquina se abre el navegador? ¿De qué máquina es el puerto 8080 en el que se está escuchando?

La opción `-D port` especifica el reenvío de un puerto local, con el destino especificado **dinámicamente** por la aplicación. Es decir, las conexiones al puerto `port` del cliente, se reenvían sobre el canal seguro establecido con el servidor y de ahí, a donde la aplicación indique usando el protocolo SOCKS. De esta manera, SSH actuará como un *proxy* a nivel de circuito. Muchas aplicaciones (por ejemplo, los navegadores) incluyen soporte para conectarse a través de un *proxy* SOCKS.

En `left`, establece un túnel SSH con `right` en el puerto 1080 (puerto típico de los servidores SOCKS) de `left` y desde `right` a donde indique la aplicación:

```
$ ssh -v -N -D 1080 usuario@192.168.1.2
```

Configura el navegador en `left` para que se conecte a través de un *proxy* SOCKS:

```
$ chromium --proxy-server="socks5://localhost:1080"
```

Accede en `left` a http://www.ucm.es con el navegador.

- **Entrega:** Copia la salida del comando `ssh` desde que introduces la contraseña de la clave. Para el servidor www.ucm.es, ¿qué máquina es la que se está intentando conectar?