Apéndice de Redes: SSH y VPN

VII Curso de Introducción a la Administración de Servidores GNU/Linux
Extensión Universitaria. Universidad de Sevilla
Mayo 2010

por Enrique Ostúa

Contenidos

1. SSH: Introducción

- SSH es un protocolo de nivel de aplicación para facilitar la conexión remota a un servidor
- Proporciona una comunicación muy segura gracias a que garantiza la **encriptación** y la **integridad** de la conexión
- Facilita la autenticación por varios métodos
- También permite realizar de forma segura:
  - la redirección de puertos, creando túneles
  - ejecución remota de aplicaciones X-window
  - transferencia de ficheros (SFTP)

Modos de autenticación

- SSH soporta por defecto 4 modos de autenticación, que se prueban en este orden:
  - GSS_API, opción de implementar lo que se quiera
  - host-based, mediante Rhosts (inseguro!) y Shosts. autentica por “nombre host + nombre de usuario”, mediante certificados de los hosts.
  - clave pública, el servidor tiene la clave pública del usuario, que usa su clave privada para conectarse. Soporta encriptación RSA y DSA.
  - contraseña, la del usuario en el servidor

Instalando SSH

- Se recomienda usar OPENSSH, hay otras versiones no libres o comerciales.
- El cliente SSH viene instalado 'de serie', paquete “openssh-client” (para Windows existen varios clientes, por ejemplo “Putty”)
- El servidor SSH es “openssh-server”. Al instalarlo se crea un par de claves de host publica/privada para usar por RSA o DSA
  - En /etc/ssh: ssh_host_rsa_key y ssh_host_dsa_key son las claves privadas (nadie debe verlas!)
  - Los .pub son las claves públicas del servidor

Configuración

- /etc/ssh/**sshd_config** : la configuración por defecto del servidor es bastante completa
  - “X11Forwarding yes”, activa la redirección de aplicaciones X-window en remoto
  - “PermitRootLogin no”, prohibe la conexión de root
  - La autenticación basada en Rhosts viene desactivada por defecto:
    - “IgnoreRhosts yes”, “RhostsRSAAuthentication no”, “HostbasedAuthentication no”

Conectando por SSH

$ ssh [-X] [usuario@]servidor.de.ssh

- Establece una conexión segura con el servidor y abre una shell de comandos.
- La opción “-X” reenvia las aplicaciones de X remotas para ser manejadas desde el sistema X local.
- El nombre de usuario es opcional, si no se indica se usa el mismo que el local.
  - Ej: $ ssh mcr-120 $ ssh jruiz@mcr-120
    $ ssh -X jruiz@mcr-120 -> gcalctool

Autenticación por clave pública

- Esto es a nivel particular de cada usuario.
- Crear ~/.ssh con permisos 700 en CLI y SERV
- En CLIENTE: “ssh-keygen -t dsa”, nos pedirá una **contraseña** para asociarla al par. Crea “id_dsa” y “id_dsa.pub” en el ~/.ssh
- En SERVIDOR: copia/añade el contenido de tu clave pública en ~/.ssh/authorized_keys

La clave pública ID_DSA.PUB es así... todo va en una sóla línea!

“ssh-dss AAAAB3NzaC1kc3MAAACBANRwVmKMQY
Z6rM1fDcullUiAKOD8A8l4ZdJU4yVKl8bO2imsPXX
OU.....BhtFaaLrvjtcS75DMDpw ostua@mcr-120”

Clave pública sin contraseña

- Se puede generar la clave RSA/DSA sin asociar una contraseña. Es arriesgado!
- Se debe usar sólo para ejecutar un comando de forma automática, con algún script o similar, por ejemplo un backup nocturno.
- Se añade en ~/.ssh/authorized_keys antes de la propia clave el comando que se va a ejecutar, todo en la misma línea:

command="ls -la /tmp",no-port-forwarding,no-x11-forwarding,no-agent-forwarding ssh-dss AAAAB3NzaC1k..... ostua@mcr-120”

Túnel SSH

- Al conectar por SSH podemos redirigir un puerto local de forma que el servidor redirige la petición a otro puerto en la red remota.
- Esto crea un **túnel** seguro para esa conexión
- Ejemplos de uso: para encriptar un protocolo no seguro (Web, SMTP, VNC, ...), o también para entrar a un servicio de la red privada desde internet (Servidor Web de la Intranet).

  $ ssh -L 5555:servweb:80 juan@servssh.micasa.es

  – En la URL “http://localhost:5555/” veré la intranet de forma seguro, mientras dure la sesión SSH.

2: Redes Privadas Virtuales (VPN)

- Es un sistema para conectar distintas redes LAN y/o “equipos sueltos” a una única red en la que los equipos de la VPN se ven entre sí como si estuvieran en una red LAN real.
- Todo esto a través de internet y de forma segura, independientemente de dónde esté y de cómo se conecte a la red.
- La VPN tiene su direccionamiento interno propio, generalmente con IP's privadas, además del que tenga cada equipo en Internet. Es decir, los equipos están conectados a dos redes, internet y la VPN.

openVPN

- openVPN es una solución completa para trabajar con VPN's de todo tipo, software libre y portada para muchas arquitecturas
- Los equipos se identifican en la VPN a través de certificados que deben ser emitidos por el servidor.
- Se usa una autoridad certificadora, que emite los certificados para los clientes y los puede revocar en cualquier momento.
- Paquete “openvpn” en servidor y cliente
- Configuración en /etc/openvpn

Routing o Ethernet Bridging

- openVPN puede trabajar de dos formas:
  - Routing, la más sencilla, establece conexiones a nivel TCP/IP con cada cliente, básicamente sólo hay un nuevo interfaz (tun0) para la conexión con la VPN. Será la forma que veremos en adelante.
    - limitación: no funcionan los broadcasts, ni rutar cosas que no sean TCP/IP.
  - Ethernet Bridging, de esta forma se simula en cada cliente un dispositivo ethernet virtual completo, tap0, y la red VPN sería como un switch virtual, llamado br0, donde se conectan los tap's.

Generando los certificados VPN

```
# cp -r /usr/share/doc/openvpn/examples/easy-rsa  /etc/openvpn
# cd /etc/openvpn/easy-rsa
# . ./vars
# ./clean-all
# ./build-ca                    (crea la autoridad certificadora)
# ./build-key-server servidor   (hay que responder al “Common
                                 Name” con 'servidor' también)
# ./build-key-pass cliente1     (repetir para cada cliente)
# ./build-dh
# ls /etc/openvpn/easy-rsa/keys/
ca.crt  ca.key  cliente1.crt  cliente1.csr  cliente1.key  cliente2.crt
   cliente2.csr  cliente2.key  dh1024.pem    index.txt   serial
   servidor.crt  servidor.csr  servidor.key
```

Los ficheros *.KEY son SECRETOS. Hay que protegerlos!

Configurando el servidor

- Necesitamos: ca.crt, dh1024.pem, servidor.crt y servidor.key

```
# mkdir -m 700 /etc/openvpn/keys
# cp ca.crt dh1024.pem servidor.crt servidor.key /etc/openvpn/keys
# zcat /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz >
    /etc/openvpn/server.conf
      port 1194                          keepalive 10 120
      proto udp                          comp-lzo      # usa compresión
      ca keys/ca.crt                     device tun    # crea interfaz tun0,tun1...
      cert keys/servidor.crt
      key keys/servidor.key
      dh keys/dh1024.pem
      server 10.8.0.0 255.255.255.0   # direcciones IP de la VPN (.1 será el servidor)
      client-to-client   # permite a los clientes verse entre sí, no solo cliente-servidor
# /etc/init.d/openvpn restart
```

Configurando cada cliente

- Necesitamos los ficheros: ca.crt, cliente1.crt y cliente1.key

```
# mkdir -m 700 /etc/openvpn/keys
# cp ca.crt cliente1.crt cliente1.key /etc/openvpn/keys
# cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/openvpn
      client
      proto udp
      remote  servidorvpn.midominio.com  1194
      device tun  # creará interfaz tun0
      comp-lzo    # debe estar activo en el servidor
      ca keys/ca.crt
      cert keys/cliente1.crt
      key keys/cliente1.key
# /etc/init.d/openvpn restart
```

Conexión servidor-clientes

- Si ambos demonios están arrancados, el interfaz tun0 del cliente será una conexión 'punto a punto' con el servidor, encriptada.
- La IP del servidor VPN será la 10.8.0.1 y la de los clientes serán de la 10.8.0.6, la .10, .14, .18, etc.

```
# ifconfig tun0
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.6  P-t-P:10.31.0.4  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:892 (892.0 b)  TX bytes:1176 (1.1 KB)
# ping 10.8.0.1
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.484 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.455 ms
# ping 10.8.0.10
64 bytes from 10.8.0.10: icmp_seq=1 ttl=64 time=2.23 ms
64 bytes from 10.8.0.10 icmp_seq=2 ttl=64 time=2.19 ms
```

Más configuraciones

- Para permitir el reenvio entre clientes, hay que activar el forwarding en el servidor:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Si tenemos reglas de iptables en el servidor, hay que permitir lo siguiente:

```
iptables -A INPUT -i tun+ -j ACCEPT                    # tun+ es tun0,tun1,tun2...
iptables -A OUTPUT -o tun+ -j ACCEPT
iptables -A FORWARD -i tun+ -j ACCEPT
```

- Si queremos que la LAN completa del servidor sea visible desde los clientes de la VPN, y no sólo el servidor, se debe añadir en el /etc/openvpn/server.conf la opción:

```
push "route 10.0.11.0 255.255.255.0"      # LAN del servidor 10.0.11.0/24
```

  - Ojo, ¿a lo mejor hay que activar el NAT para que las dos redes se vean? Si el servidor VPN también es el gateway de la LAN del servidor no hace falta, sino habrá que hacer NAT o añadir rutas en el gateway de la LAN.