Redes de Computadoras

—

Prácticas

Juana López Redondo
María Laura Da Silva Hernández
Vicente González Ruiz
2 de junio de 2011

SSH

SSH (Secure SHell) [8] es un programa semejante al programa Telnet, pero que a diferencia de éste, SSH cifra toda la comunicación entre el cliente y el servidor. Para ello se vale de algoritmo de cifrado (o encriptación) y posiblemente, compresión. SSH es muy utilizado para el acceso remoto a hosts, permitiendo a los usuarios trabajar como si estuvieran físicamente sentados frente el teclado del host remoto.

En esta práctica vamos a aprender a usar el conjunto de utilidades que pertenencen al paquete SSH.

La implementación del SSH que vamos a utilizar en este módulo se llama OpenSSH (http://www.openssh.com) e incorpora, entre otras, las siguientes utilidades:

ssh: Un cliente. El sustituto de Telnet.

sshd: Un servidor.

scp: (Secure CoPy). Una utilidad semejante a la utilidad rcp que permite copiar ficheros entre hosts remotos de forma segura.

sftp: (Secure FTP). Una versión segura del programa Ftp.

sshfs: (SSH File System). Una versión del NFS sobre SSH.

11.1. Algoritmos de cifrado

Los algoritmos de cifrado (o criptográficos) se utilizan para esconder la información a aquel conjunto de personas que no deberían tener acceso a ella. La idea es muy simple y consiste en alterar la representación de la información usando una *clave* (o *key*). Así, sólo quien conozca la clave es capaz de restaurar la representación original (descrifrada) de la información.

Existen dos tipos distintos de algoritmo de cifrado, dependiendo del número de claves que se utilizan:

Algoritmos de clave simétrica (privada): También se llaman algoritmos de clave secreta y se caracterizan por usar la misma clave para cifrar y descifrar. Así,

si enviamos un e-mail cifrado a alguien también tenemos que indicarle la clave que usamos para cifrarlo.

Algoritmos de clave asimétrica (pública): Los sistemas de clave simétrica presuponen que el receptor conoce la clave secreta, pero, ¿cómo le hacemos llegar dicha clave sólo a él a través de un canal inseguro como puede ser Internet? Usando un algoritmo de clave simétrica es imposible.

Para evitar este problema se idearon los algoritmos de clave asimétrica (o algoritmos de clave pública). En éstos, cada extremo de la comunicación maneja dos claves, una pública y otra privada (en total, 4 claves).[1] Se cumple que: (1) lo que se cifra utilizando la clave pública que un interlocutor posee sólo puede descifrarse con la clave privada que únicamente el otro interlocutor posee y (2) es imposible derivar una clave privada a partir de la correspondiente clave pública (sin información adicional que sólo conoce quien genera la clave pública a partir de su clave privada).

Así, cuando dos personas desean intercambiarse información primero deben intercambiar sus claves públicas. Habiendo hecho esto, cuando yo (por ejemplo) deseo enviar un mensaje a alguien lo cifro usando la clave pública de mi interlocutor. Dicho mensaje puede ser interceptado por una tercera persona, pero sólo mi interlocutor va a ser capaz de descifrarlo porque es la única persona que posee la clave privada que puede descifrarlo. De la misma manera, sólo yo podré descifrar lo que mi interlocutor me envía porque sólo yo poseo la clave privada que permite descifrar el mensaje.

11.2. Características del SSH

Existen dos versiones de SSH: la versión 1 y la versión 2. La segunda es la más evolucionada y utilizada. SSH permite además comprimir el stream de datos al vuelo (on-the-fly). Con esto ganaremos en velocidad efectiva de tranmisión y aumentaremos la seguridad.

SSH usa DES (Data Encryption Standard) en su versión 1 y además AES (Advanced Encryption Scheme) y Blowfish en su versión 2. DES es considerado el menos seguro y AES el más seguro. Debe tenerse en cuenta que el consumo de CPU será proporcional al nivel de seguridad.

El SSH utiliza un algoritmo de clave simétrica para cifrar los datos que el cliente y el servidor se intercambian. Dicha clave se genera (de forma aleatoria) en el cliente y se envía al servidor usando un algoritmo de clave asimétrica. Otro aspecto importante de SSH radica en que posee un sistema para autentificar a los extremos de la comunicación (evita el *spoofing*). De esta manera, se asegura que tanto el emisor como el receptor son realmente quien dicen ser. [2] Veamos con un poco más de detalle cómo se produce este proceso.

El SSH utiliza tres mecanismos (tres protocolos) [8]:

[1] El emisor tiene una clave pública y otra privada, y el receptor una clave pública y otra privada.

[2] Nótese que esto no se puede verificar simplemente con el algoritmo de cifrado.

Transport Layer Protocol (TLP) [10]: Autentifica el servidor, cifra la comunicación y conserva la integridad de la información transmitida. Suele usar el TCP y adicionalmente puede comprimir los datos.

User Authentication Protocol (UAP) [9]: Utiliza el TLP y autentifica el cliente al servidor.

Connection Protocol (CP) [11]: Se ejecuta sobre el UAP y permite multiplexar muchos canales lógicos sobre un único tunel cifrado.

A continuación se presenta todo el proceso de comunicación (http://es.tldp.org/Tutoriales/doc-ssh-intro/intro-ssh/node9.html) entre un cliente y un servidor (versión 2 del protocolo):

1. **Fase de conexión TCP**:
   a) El cliente establece una conexión TCP con el servidor.
2. **Fase de identificación del protocolo**:
   a) Si el servidor acepta la conexión TCP envía al cliente un mensaje (no cifrado) que indica los parámetros fundamentales de la conexión (como por ejemplo, la versión del protocolo SSH).
   b) El cliente envía al servidor otro mensaje con su propia información acerca de los parámetros que desea utilizar en la comunicación.
3. **Fase de autentificación del servidor**:
   a) Si el servidor acepta las condiciones del cliente, el servidor envía su *host key* pública, sin cifrar.
   b) El cliente comprobará que la *host key* recibida es igual a la que recibió en la última conexión con dicho servidor. Si es la primera vez que se conecta, mostrará la *host key* al usuario para que pueda, por ejemplo, telefonear al administrador del host remoto para preguntarle la *host key*. Si no es la primera vez y es diferente (el cliente mira en ~/.ssh/known_hosts), también mostrará la *host key* para evitar el posible spoofing.
   c) Si el cliente continua con la conexión, cifrará y almacenará la *host key* en ~/.ssh/known_hosts para futuras comprobaciones.
4. **Fase de generación de la clave de sesión**:
   a) El cliente genera una clave de cifrado simétrica llamada *session key* y la cifra utilizando la *host key* pública del servidor.
   b) El cliente envía la *session key* cifrada al servidor.
   c) El servidor descifra la *session key* utilizando su *host key* privada. En este punto ambos extremos conocen la clave de sesión que se utiliza para cifrar y descifrar el resto de la comunicación.

5. **Fase de identificación y autentificación del usuario** (http://www.ssh.fi/support/documentation/online/ssh/adminguide/32/User_Authentication.html). Se intentarán secuencialmente cada uno de los siguientes métodos[3]:

   Autentificación basada en el host del usuario: El usuario queda autentificado en estos casos:

   a) Si el host de usuario está en /etc/hosts.equiv o en /etc/ssh/shosts.equiv, y si el nombre del usuario es el mismo en la máquina local y en la máquina remota.

   b) Si los ficheros ~/.rhosts o ~/.shosts en el directorio *home* del usuario en la máquina remota contiene una entrada de la forma usuario_maquina_cliente@maquina_cliente.

   En cualquiera de estos casos es necesario que el fichero /etc/ssh/ssh_known_hosts o el fichero ~/.ssh/known_hosts contenga la *host key* pública del cliente.

   Autentificación de clave pública: Para poder usar esta forma de autentificación, el usuario en alguna sesión previa ha debido generar una clave pública y otra privada (generalmente usando RSA). En dicha sesión previa, el usuario ha informado al servidor de su clave de usuario pública. Durante esta fase de autentificación el servidor genera un número aleatorio (llamado *challenge*) que es cifrado usando la clave pública que le envió el usuario. El cliente recibe el desafío, lo descifra usando su clave privada, y lo vuelve a cifrar usando la *host key* pública del servidor. Finalmente se lo envía al servidor. El servidor lo descifra usando su *host key* privada y si el número coindice, entonces el usuario queda autentificado.

   Autentificación basada en password: Consiste en enviar un password que sólo el usuario conoce (generalmente el password usado en la cuenta de la máquina remota). Dicho password viaja cifrado con la clave de sesión simétrica.

6. **Fase de acceso al sistema remoto**:

   a) El host remoto ejecuta un shell cuya entrada estandar es proporcionada por el servidor SSH y cuya salida es enviada al servidor SSH.

7. **Fase de desconexión**:

   a) Cuando el usuario realiza un logout, el shell en el host remoto muere y la conexión TCP se cierra.

11.3. Instalación de SSH (cliente y servidor)

En Linux tenemos disponibles un servidor y un cliente SSH a través del paquete OpenSSH (http://www.openssh.com). Su instalación es muy sencilla:

[3] Esta lista no es exhaustiva, es decir, pueden existir otras formas de autentificación.

Debian Linux:

```
root# apt-get install ssh
```

El demonio se llama ssh.

Fedora Core Linux:

```
root# yum install openssh
```

El demonio se llama sshd.

Gentoo Linux:

```
root# emerge openssh
```

El demonio se llama sshd.

11.4. Configuración del servidor

El servidor SSH se configura modificando el fichero de configuración (ojo con la d):

```
/etc/ssh/sshd_config
```

Dicho fichero está codificado en ASCII y suficientemente autocomentado. A continuación comentaremos las diferentes opciones (más información en man sshd_config):

Port: Puerto de escucha del servicio. 22 por defecto.

ListenAddress: Direcciones que serán atendidas. Todas por defecto.

Protocol: Versión del protocolo. 2 por defecto.

HostKey: Especifica los ficheros con las host keys. /etc/ssh/ssh_host_rsa_key y /etc/ssh/ssh_host_dsa_key, por defecto.

UsePrivilegeSeparation: Especifica si el fichero ~/.ssh/environment y las opciones environment= en el fichero ~/.ssh/authorized_keys son procesadas por sshd. Por defecto, no.

KeyRegenerationInterval: Periodo de regeneración del server key para la versión 1 del protocolo. Por defecto, 1 hora.

ServerKeyBits: Número de bits de la server key para la versión 1 del protocolo. Por defecto 768 bits.

SyslogFacility: Tipo de registro de actividades. Por defecto AUTH.

LogLevel: Nivel de registro de actividades. Por defecto, INFO.

LoginGraceTime: Tiempo de espera para la autentificación. 120 segundos, por defecto.

PermitRootLogin: ¿Se permite acceder al root? Por defecto, sí.

StrictModes: Enviar ficheros con todos los permisos a todos los usuarios. Sí, por defecto.

RSAAuthentication: ¿Queremos usar RSA (versión 1)? Sí, por defecto.

PubkeyAuthentication: ¿Queremos usar clave pública (versión 2)?. Sí, por defecto.

AuthorizedKeysFile: Fichero con las claves para la autentificación. ~/.ssh/authorized_keys, por defecto.

IgnoreRhosts: ¿Se ignoran los ficheros ~/.rhosts y ~/.shosts? Sí, por defecto.

RhostsRSAAuthentication: Permitir la autentificación por /etc/rhosts (versión 1). Por defecto, no.

HostbasedAuthentication: Permitir la autentificación por /etc/rhosts (versión 2). Por defecto, no.

IgnoreUserKnownHosts: Poner a yes si no confiamos en el fichero ~/.ssh/known_hosts para la RhostsRSAAuthentication.

PermitEmptyPasswords: No, por defecto (y no se recomienda poner yes).

ChallengeResponseAuthentication: Habilita los passwords "challenge-response". Por defecto, no.

PasswordAuthentication: Permite deshabilitar los passwords en texto plano cuando se está utilizando tunneling. No, por defecto (los passwords están cifrados, por defecto).

KerberosAuthentication: Usar Kerberos para autentificar los usuarios. Por defecto, no.

KerberosOrLocalPasswd: Opción "OrLocalPasswd" cuando utilizamos Kerberos. Por defecto, sí.

KerberosTicketCleanup: Opción "TicketCleanup" cuando utilizamos Kerberos. Por defecto, sí.

KerberosGetAFSToken: Opción "GetAFSToken" cuando utilizamos Kerberos. Por defecto, no.

GSSAPIAuthentication: Usar GSSAPI para autentificar los usuarios. Por defecto, no.

GSSAPICleanupCredentials: Opción "CleanupCredentials" cuando utilizamos Kerberos. Por defecto, sí.

X11Forwarding: Redirigir la conexión TCP usada por el servidor X-Window. Sí, por defecto. Así, cuando accedamos al host remoto desde una consola gráfica (un xterm, por ejemplo) y ejecutemos una aplicación gráfica, ésta aparecerá en nuestro sistema de ventanas.

X11DisplayOffset: Offset por defecto para la variable de entorno DISPLAY que controla el display gráfico utilizado por las aplicaciones gráficas.[4] Por defecto, 10.

PrintMotd: El fichero `/etc/motd` es un mensaje de bienvenida codificado en ASCII y que es mostrado cuando accedemos a través del terminal remoto. Por defecto, no usar.

PrintLastLog: Imprimir la salida más reciente del comando de `lastlog` que se utiliza para saber desde dónde nos conectamos al servidor la última vez que utilizamos SSH. Por defecto está a `yes`.

TCPKeepAlive: Enviar periódicamente mensajes "TCP keepalive". De esta manera si la conexión falla los extremos son notificados y el terminal no se "cuelga". Por defecto, sí.

UseLogin: ¿Usar la utilidad `login` para la autentificación? Por defecto, no.

MaxStartups: Especifica el número máximo de conexiones concurrentes sin autentificar que serán permitidas. Por defecto, 10.

Banner: El banner es un mensaje que puede enviarse antes de iniciarse la conexión con el servidor (para avisar, por ejemplo, que se está accediendo a un sitio muy chungo :-). Esta opción indica la localización del fichero ASCII que contiene dicho mensaje.

AcceptEnv: Especifica qué variables para la configuración local del lenguaje van a ser aceptadas. Por defecto, todas.

Subsystem: Permite especificar la localización en el sistema de ficheros de una determinada utilidad a usar con SSH. Normalmente se utiliza para indicar el fichero que contiene la utilidad `sftp`.

UsePAM: Autentificar al usuario utilizando el Pluggable Authentication Module. Por defecto, sí.

Tras modificar el fichero de configuración del servidor SSH es necesario relanzar el servicio (véase el Apéndice F).

[4] En el sistema X11 un servidor X11 puede tener varios clientes X11 (varios terminales gráficos). La variable DISPLAY permite seleccionar el cliente.

11.5. Configuración del cliente

El cliente SSH (ssh) utiliza el fichero de configuración:

`/etc/ssh/ssh_config`

y demás, el fichero:

`~/.ssh/config`

si es que existe. Los parámetros especificados en él prevalecen sobre el fichero `ssh_config`. La información completa sobre este fichero puede obtenerse mediante `man ssh_config`.

Veámos las principales opciones que podemos configurar:

Host: Especifica el host al que se aplican los parámetros que aparecen a continuación. Un "*" (que es el valor por defecto) referencia a todos los hosts. Nótese que los sistemas en los que comparte el directorio "/etc" (que típicamente se monta vía NFS) esto puede ser muy útil.

ForwardAgent: Indica si la conexión con el agente de autentificación (si es que existe) será redirigido a la máquina remota. Por defecto, no.

ForwardX11: Se utiliza para hacer que las conexiones X11 sean automáticamente redirigidas sobre el canal seguro (utilizando la variable de entorno DISPLAY). Por defecto es que no.

ForwardX11Trusted: Especifica si el cliente X11 tendrá acceso sin restricciones al servidor. Por defecto es que sí tendrá acceso total.

RhostsRSAAuthentication: Especifica si usar el fichero `rhosts` con RSA (ver configuración del servidor). Por defecto, no.

RSAAuthentication: Especifica si usar RSA authentication (ver configuración del servidor). Por defecto, sí.

PasswordAuthentication: Indica si usar autentificación basada en password. Por defecto, sí.

HostbasedAuthentication: Especifica si usar la autentificación basada en `rhosts` con la autentificación de clave pública. Por defecto, no.

BatchMode: Permite deshabilitar la pregunta sobre el password. Por defecto, no.

CheckHostIP: Por defecto está habilitada. Si se deshabilita el cliente SSH no comprobará si el servidor figura en el fichero `/etc/ssh/ssh_known_hosts`. Esto permite detectar el DNS spoofing (el que alguien inserte en el DNS una IP falsa para un host).

AddressFamily: Tipos de direcciones IP aceptadas. Por defecto any.

ConnectTimeout: El TCP utiliza un temporizador para mantener activa la conexión enviando un paquete de datos vacío, aunque no exista actividad. Con esta opción hacemos que sea el SSH el que se encarge de este proceso, en lugar del TCP. Un valor a 0 indica que esta opción seguirá siendo controlada por el TCP.

StrictHostKeyChecking: Si este flag se activa a yes, el cliente SSH no añadirá el key host al fichero ~/.ssh/known_hosts, y por tanto, recharará la conexión con aquellos hosts que hallan cambiado su host key (por ejemplo, porque han reinstalado el sistema operativo).

IdentityFile: Especifica el fichero con el identificador DSA. En la versión 1 del protocolo apunta al fichero ~/.ssh/identity. En la versión 2 puede apuntar al fichero ~/.ssh/id_rsa o al fichero ~/.ssh/id_dsa, dependiendo del algoritmo de cifrado usado.

Port: Puerto donde espera encontrar al servidor. 22, por defecto.

Protocol: Versión del protocolo. 2, por defecto.

Cipher: Especifica el cifrador en la versión 1 del protocolo. Por defecto, 3des.

Ciphers: Cifradores para la versión 2, por orden de preferencia. Por defecto, aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc.

EscapeChar: Por defecto vale el carácter ~ (<Alt Gr> + <4>) y se utiliza para realizar acciones como terminar la conexión (~.), enviar un BREAK al sistema remoto (~B), acceder al interprete de comandos del cliente (~C), mostrar las conexiones redirigidas (~#), etc.

Tunnel: Esta opción por defecto vale no y se utiliza para evitar tener que utilizar la capa de red del TCP/IP. Los otros posibles valores son: yes, point-to-point y ethernet.

TunnelDevice: Selecciona el interface de red sobre el que realizar el tunneling. Por defecto, any:any.

PermitLocalCommand: Controla la posibilidad de ejecutar comandos en la máquina local (no en la remota) usando la secuencia de escape C. Por defecto, no.

SendEnv: Sirve para especificar el contenido de una variable de entorno (generalmente LANG que controla el lenguaje utilizado en el shell).

HashKnownHosts: Se usa para aplicar un hashing sobre los nombres de host y direcciones que se especifican en ~/.ssh/known_hosts. No, por defecto.

GSSAPIAuthentication: Permitir autentificación GSSAPI. Por defecto, no.

11.6. Uso de SSH

SSH es uno de los programas con los que tendremos que lidiar todos los días que estamos trabajando con máquinas modernas con Unix o Linux. Veámos algunos ejem- plos de uso típicos.

11.6.1. Accediendo a un host remoto

Para acceder al host remoto utilizamos el cliente ssh. En ese momento se pro- ducirá el proceso de autentificación del servidor y el proceso de identificación y auten- tificación del usuario. Dependiendo del esquema de autentificación de usuario existen distintas alternativas:

Autentificación basada en password: Suele ser la forma más corriente de acceder al host remoto:

```
alumno$ ssh host_remoto
```

Con este comando estamos accediendo al host utilizando el usuario alumno. Cuando queramos acceder con otro nombre de usuario utilizaremos:

```
alumno$ ssh otro_nombre_de_usuario@host_remoto
```