Servidor de conexión remota

Tema 3

Prof. Ing. Angel Brito Segura

1. Introducción

Al principio del auge del Internet, la mayoría de los sistemas operativos, como UNIX, operaban en un entorno de tiempo compartido, donde los usuarios forman parte del sistema con cierto derecho a acceder a los recursos. Cada usuario autorizado tiene una identificación (lo define dentro del sistema operativo como miembro del mismo) y, probablemente, una contraseña. Para acceder al sistema, el usuario inicia sesión en el sistema con un ID de usuario o un nombre de inicio de sesión y el sistema realiza una comprobación de contraseñas para evitar que un usuario no autorizado acceda a los recursos.

Por otra parte, los usuarios quieren ejecutar sus programas en un sitio remoto y crear resultados que se puedan transferir a su sitio local. Una forma de satisfacer esa demanda fue la creación de programas de aplicación cliente/servidor para cada servicio deseado, como fue el caso del servidor de correo electrónico (SMTP), que nos permite intercambiar mensajes entre usuarios. Sin embargo, sería imposible escribir un programa cliente/servidor específico para cada demanda solicitada por los usuarios.

Para solucionar esto, se creó un programa cliente/servidor de propósito general que permita al usuario acceder a cualquier aplicación en la computadora remota; en otras palabras, permitir que el usuario inicie sesión en una computadora remota. Después de iniciar sesión, un usuario puede utilizar los servicios disponibles en la computadora remota y transferir los resultados de regreso a la computadora local.

2. TELNET

TErminaL NETwork es el protocolo TCP/IP estándar para el servicio de terminal virtual (ISO), siendo un programa de aplicación cliente/servidor de propósito general. Este protocolo de comunicación bidireccional permite el establecimiento de una conexión a un sistema remoto de tal manera que la terminal local parezca una terminal en el servidor remoto.

2.1. RFC 854

Esta Solicitud de Comentarios define el Protocolo TELNET, que es un protocolo de comunicación de red que permite a los usuarios establecer sesiones interactivas en sistemas remotos a través de una conexión basada en texto. Funciona sobre TCP (generalmente en el puerto 23) y proporciona un canal de comunicación interactivo con su protocolo simple y versátil, pero no cifra los datos, lo que lo hace inseguro en redes modernas.

2.2. Inicio de sesión

Cuando un usuario inicia sesión en un sistema local de tiempo compartido (inicio de sesión local) escribe en una terminal o en una estación de trabajo que ejecuta un emulador de terminal, las pulsaciones de teclas son aceptadas por el controlador de terminal. El controlador de terminal pasa los caracteres al sistema operativo que, a su vez, interpreta la combinación de caracteres e invoca el programa de aplicación o utilidad deseado.

Cuando un usuario desea acceder a un programa de aplicación o utilidad ubicado en el servidor remoto (inicio de sesión remoto), protocolo TELNET le permite al usuario controlar el servidor remoto enviando comandos en formato ASCII.

El usuario envía las pulsaciones de teclas al controlador de terminal, donde el sistema operativo local acepta los caracteres pero no los interpreta. Los caracteres se envían al cliente de TELNET, que transforma los caracteres en un conjunto de caracteres universales (NVT, caracteres de terminal virtual de red) y los entrega a la pila de protocolo TCP/IP local.

Los comandos o texto, en formato NVT, viajan a través de Internet y llegan a la pila TCP/IP de la máquina remota. Aquí los caracteres se entregan al sistema operativo y se pasan al servidor TELNET, que cambia los caracteres recibidos a los caracteres correspondientes que puede entender la computadora remota. Sin embargo, los caracteres no se pueden pasar directamente al sistema operativo porque el sistema operativo remoto no está diseñado para recibir caracteres de un servidor TELNET: está diseñado para recibir caracteres de un controlador de terminal.

Para solucionar esto, se cuenta con un ***controlador de pseudoterminal*** que simula que los caracteres provienen de una terminal. Luego, el sistema operativo pasa los caracteres al programa de aplicación apropiado para realizar la acción solicitada.

2.3. Conjunto de Caracteres de Terminal Virtual de Red

El mecanismo para acceder a una computadora remota es complejo. Esto se debe a que cada computadora y su sistema operativo aceptan una combinación especial de caracteres como tokens, por lo que se esta tratando con sistemas heterogéneos. Para acceder a cualquier ordenador remoto del mundo, primero se debe de saber a qué tipo de servidor se realizará la conexión y debe de estar instalado el emulador de terminal específico que utiliza ese servidor.

Para resolver este problema, el protocolo define la interfaz universal: ***conjunto de caracteres de terminal virtual de red (NVT)***. A través de esta interfaz, el cliente TELNET traduce caracteres (datos o comandos) que vienen del terminal local al formato NVT y los entrega a la red. El servidor TELNET, por otro lado, traduce datos y comandos del formato NVT al formato aceptable para el ordenador remoto.

Este conjunto de caracteres NVT utiliza dos conjuntos de caracteres, uno para datos y el otro para control (ambos usan bytes de 8 bits). Para los datos, NVT es un conjunto de caracteres de 8 bits en el que los 7 bits de orden más bajo son iguales a ASCII y el bit de orden más alto es 0. Para enviar caracteres de

control entre computadoras (del cliente al servidor o viceversa), NVT utiliza un conjunto de caracteres de 8 bits en el que el bit de orden más alto se establece en 1.

Character	*Decimal*	*Binary*	*Meaning*
EOF	236	11101100	End of file
EOR	239	11101111	End of record
SE	240	11110000	Suboption end
NOP	241	11110001	No operation
DM	242	11110010	Data mark
BRK	243	11110011	Break
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase character
EL	248	11111000	Erase line
GA	249	11111001	Go ahead
SB	250	11111010	Suboption begin
WILL	251	11111011	Agreement to enable option
WONT	252	11111100	Refusal to enable option
DO	253	11111101	Approval to option request
DONT	254	11111110	Denial of option request
lAC	255	11111111	Interpret (the next character) as control

Figura 1: Caracteres de control en el NVT

TELNET utiliza sólo una conexión TCP donde envia los datos y los caracteres de control que incorpora en el flujo de datos. Sin embargo, para distinguir los datos de los caracteres de control, cada secuencia de caracteres de control está precedida por un carácter de control especial llamado ***interpretar como control (lAC)***. La implementación predeterminada de TELNET, no permite al usuario editar localmente, la edición se realiza en el servidor remoto.

2.4. Negociación de Opciones

El protocolo permite que el cliente y el servidor negocien opciones antes o durante el uso del servicio. Las opciones son funciones adicionales disponibles para un usuario con una terminal más sofisticada (usuarios con terminales más simples pueden utilizar funciones predeterminadas) como las mostradas en la figura 2:

Code	*Option*	*Meaning*
0	Binary	Interpret as 8-bit binary transmission.
1	Echo	Echo the data received on one side to the other.
3	Suppress go ahead	Suppress go-ahead signals after data.
5	Status	Request the status of TELNET.
6	Timing mark	Define the timing marks.
24	Terminal type	Set the terminal type.
32	Terrninalspeed	Set the terminal speed.
34	Line mode	Change to line mode.

Figura 2: Opciones disponibles

Para utilizar cualquiera de las opciones, primero se requiere una negociación de opciones entre el cliente y el servidor. Para este fin se utilizan cuatro caracteres de control que se muestran en la figura 5:

Character	*Decimal*	*Binary*	*Meaning*
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DONT	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable

Figura 3: Caracteres de control

Una parte puede ofrecer habilitar o deshabilitar una opción si tiene el derecho de hacerlo. La otra parte puede aprobar o rechazar la oferta. Para ofrecer habilitar, la parte receptora envía el comando `WILL` (*¿habilitas la opción?*), La otra parte envía el comando `DO` (*Hazlo por favor*) o el comando `DONT` (*No lo hagas por favor*). Para ofrecer deshabilitar, la parte receptora envía el comando `WONT` (*No usaré más esta opción*) y la respuesta debe ser el comando `DONT`.

Una parte puede solicitar a la otra parte que habilite o deshabilite una opción. Para solicitar la habilitación, la parte solicitante envía el comando `DO`, mientras que la otra parte envía el comando `WILL` o `WONT`.

Para solicitar la desactivación, la parte solicitante envía el comando `DONT` y la respuesta debe ser el comando `WONT`. Algunas opciones requieren información adicional que se indica en la siguiente figura 4:

Character	*Decimal*	*Binary*	*Meaning*
SE	240	11110000	Suboption end
SB	250	11111010	Suboption begin

Figura 4: Opciones adicionales

2.5. Modos de Operación

La mayoría de las implementaciones de TELNET funcionan en alguno de los siguientes tres modos:

- **Modo predeterminado**: cuando no se invocan otros modos a través de la negociación de opciones. En este modo, el cliente realiza el eco. El usuario escribe un carácter y el cliente repite el carácter en la pantalla (o impresora), pero no lo envía hasta que se completa una línea completa.
- **Modo de caracteres**: el cliente envía cada carácter escrito al servidor. El servidor normalmente repite el carácter para que se muestre en la pantalla del cliente. En este modo, el eco del carácter puede demorarse si el tiempo de transmisión es largo (como en una conexión satelital). También crea sobrecarga (tráfico) para la red porque se deben enviar tres segmentos TCP por cada carácter de datos.
- **Modo de línea**: este nuevo modo propuesto se creó para compensar las deficiencias de los dos anteriores modos. En este modo, la edición de línea (hacer eco, borrar caracteres, borrar líneas, etc.) la realiza el cliente. Luego, el cliente envía la línea completa al servidor.

3. SSH

Secure SHell es un protocolo de red que permite a los usuarios comunicarse y controlar de forma remota otros dispositivos a través de una red segura. Este protocolo proporciona un mecanismo de autenticación y cifrado de datos para proteger la información confidencial que se envía entre dos sistemas, evitando que los datos sean interceptados y leídos por alguien más. Además, se considera como un reemplazo para el protocolo Unix de intérprete de comandos remoto (`rsh`) ya que proporciona cifrado completo y reenvío de datos de visualización X11 a través de una tubería segura que se ejecuta en el puerto 22, lo que la hace una alternativa segura a Telnet.

3.1. RFCs

SSH no es solo un protocolo de acceso remoto, sino un conjunto de protocolos que trabajan en conjunto. Se divide en tres capas principales, cada una con su propio RFC:

- ***Transporte***: Seguridad y cifrado (RFC 4253)
- ***Autenticación de usuario***: Métodos de autenticación (RFC 4252)
- ***Conexión***: Multiplexación de sesiones (RFC 4254)

3.2. Seguridad

SSH-1 tenía problemas de seguridad, por lo que SSH-2 fue diseñado como un protocolo más seguro y estandarizado, provocando que diferentes algoritmos criptográficos hayan sido agregados o eliminados con el tiempo, lo que ha generado nuevos RFCs (ej. RSA, Ed25519, Diffie-Hellman). Al permitir diferentes métodos de autenticación (contraseña, clave pública, Kerberos, etc.) cada uno se fue definiendo en un RFC por separado y se han agregado extensiones como *X11 forwarding*, compresión y túneles TCP. El protocolo no solo se usa para acceso remoto, sino también para transferencia segura de archivos (SFTP, SCP), túneles VPN y administración de infraestructura (Ansible).

Debido al uso de mecanismos de cifrado, como el Estándar de cifrado de datos (DES, por sus siglas en inglés: Data Encryption Standard) y la autenticación de host RSA (acrónimo de Rivest, Shamir y Adelman, inventores del algoritmo), SSH puede proteger a los hosts de varios ataques comunes,además, crea una red privada virtual (VPN) entre el cliente y el servidor y es gratuito para uso no comercial.

La autenticación de host RSA proporciona cifrado de clave pública/clave privada, en la que los datos cifrados con la clave pública solo se pueden descifrar con una clave privada. En el caso de la autenticación de host, el host emisor cifra una cadena aleatoria de datos utilizando la clave pública del host receptor (remoto). Si el host remoto puede descifrarla correctamente utilizando su clave privada, los dos hosts saben sin lugar a dudas que son legítimos.

3.3. Comandos de ejecución remota

Antes de la llegada del protocolo SSH, los administradores de sistemas utilizaban comandos de ejecución remota como `rlogin`, `rsh`, `rcp`, `rexec` y `rsync` para conectarse a servidores remotos. Sin embargo, estos comandos no cifran los datos, transmitiendo la información en texto plano y no autentican a los hosts (existe una confianza explícita, que se basA en la creencia de que la dirección IP identifica de forma única al servidor), lo que los hace inseguros y en el mejor de los casos, sólo los sistemas locales de confianza en una red local segura pueden tener acceso a través de estos comandos r.

SSH reemplaza los comandos r estándar con comandos seguros que incluyen cifrado y autenticación, como se mencionó en la sección anterior, utiliza un esquema de autenticación fuerte para garantizar que el host de confianza es realmente el host que dice ser. Adicionalmente, proporciona una serie de esquemas de cifrado de clave pública para garantizar que cada paquete en el flujo de paquetes proviene de la fuente de la que dice provenir. Sus mecanismos de seguridad del protocolo permiten proteger el servidor contra varios ataques comunes, como:

- Suplantación de IP
- Interceptación de contraseñas de texto sin formato

- Manipulación de datos

3.4. Componentes

SSH es seguro y fácil de usar. Actualmente hay dos versiones de Secure Shell en uso generalizado: ***SSH Secure Shell***, que es un producto comercial y **OpenSSH**, que es un producto de código abierto que se incluye por defecto en varias versiones de Linux. Sus componentes básicos son:

- ***sshd***: Demonio de Secure Shell que maneja las conexiones SSH entrantes. Debe iniciarse en el momento del arranque desde uno de los scripts de arranque; no desde `inetd.conf`. Este comando genera una clave de cifrado cada vez que se inicia, lo que provoca un inicio lento, lo que lo hace inadecuado para inetd.conf. Un sistema que brinde conexiones SSH debe ejecutar este comando.
- ***ssh***: Reemplazo de `rsh` y `rlogin`. Se utiliza principalmente para pasar un comando de forma segura a un servidor remoto o para iniciar sesión de forma segura en él. Este comando crea las conexiones salientes que maneja `sshd`. Un sistema cliente que desee utilizar una conexión SSH debe utilizar este comando.
- ***scp***: Reemplazo de `rcp`. Permite copiar archivos de forma segura entre dos servidores y/o máquinas.
- ***ssh-keygen***: Genera las claves de cifrado públicas y privadas que se utilizan para proteger la transmisión para el shell seguro.
- ***sftp***: Una versión de FTP que funciona sobre una conexión de shell seguro.

3.5. Autenticación

Cuando un cliente ***ssh*** se conecta a un servidor ***sshd***, intercambian claves públicas. Los sistemas comparan las claves que reciben con las claves conocidas que tienen almacenadas en el archivo `/etc/ssh_known_hosts` y en `~/.ssh/known_hosts`. Si no se encuentra la clave o ha cambiado, se le pide al usuario que verifique que la nueva clave debe ser aceptada. Si se encuentra la clase o se acepta el cambio , el cliente la utiliza para encriptar una clave de sesión generada aleatoriamente. Luego, la clave de sesión se envía al servidor y ambos sistemas utilizan la clave para encriptar el resto de la sesión SSH.

El cliente que se autentica aparece en el archivo `hosts.equiv`, `shost.equiv`, `~/.rhosts` o `~/.shosts`. Este tipo de autenticación es similar al tipo utilizado por los comandos r. Si el cliente no aparece en uno de los archivos, se utiliza la autenticación por contraseña en la cual, SSH la encripta antes de enviarla a través del enlace de conexión.

Para mejorar la seguridad del sistema, los comandos r deben desactivarse después de instalar SSH, por lo que se debe de comentar `rshd`, `rlogind`, `rexcd` y `rexd` fuera del archivo ***inetd.conf*** para desactivar las conexiones entrantes a los comandos r. Para garantizar que se utilice SSH para las conexiones salientes, reemplace rlogin y rsh por ssh.

SSH es una excelente manera de tener comunicaciones seguras entre sistemas a través de Internet, sin embargo, requiere que ambos sistemas tengan instalado el cliente/servidor del protocolo. Adicionalmente,

tiene la capacidad de proteger la integridad de los datos que se transmiten al apoyar conexiones cifradas entre nodos de red.

3.6. Gestión de llaves pública y privada

SSH utiliza criptografía de clave pública, que proporciona claves criptográficas para autenticar nodos y usuarios remotos. En la criptografía de clave pública, dos claves están involucradas en el proceso de cifrado/descifrado: la clave pública, que puede ser compartida por varios nodos remotos, y una clave privada, que es un secreto utilizado para descifrar una clave pública correspondiente.

Los nodos que admiten SSH tienen asignadas una clave pública y una privada. La clave privada está protegida por una contraseña, que debe ingresar el usuario. La clave privada corresponde con la clave pública, que coincide con la clave pública del extremo remoto. El nodo remoto también tiene una clave privada que descifrará la información enviada a un formato legible para el usuario remoto. Los servidores SSH escuchan las solicitudes que provienen de un cliente SSH. El demonio SSH se ejecuta en el nodo servidor.

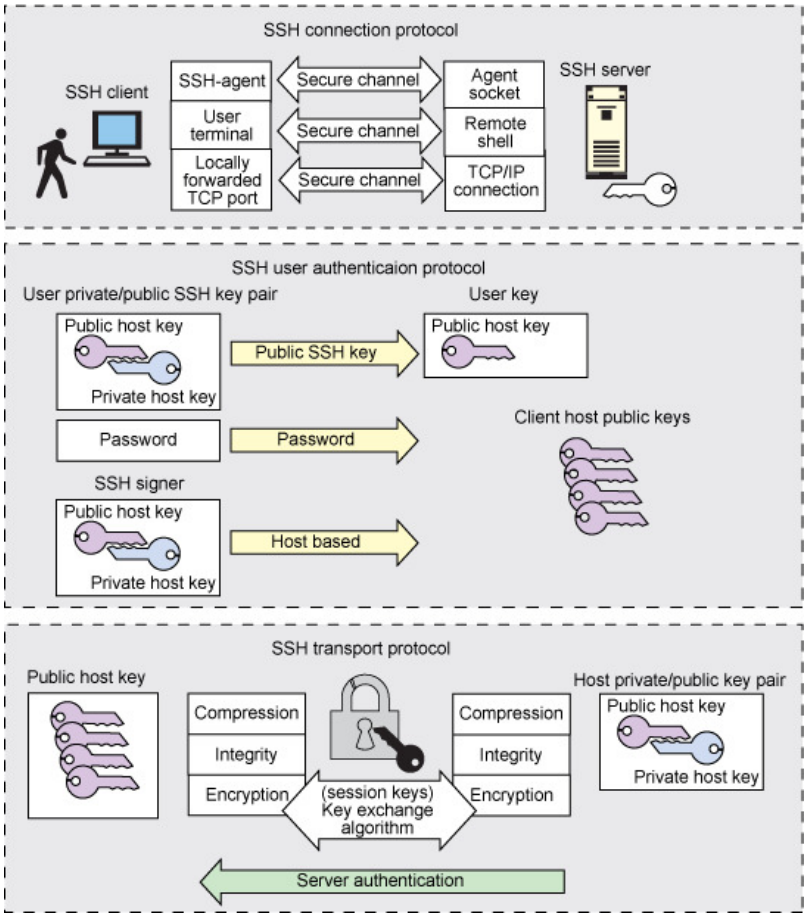

Figura 5: Transacciones de llave pública y privada de SSH

Referencias

[1] B. A. Forouzan, *Data Communications and Networking*. McGraw-Hill, 2003.

[2] D. E. Comer, *Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture*. Pretince Hall, 2000.

[3] C. Hunt, *TCP/IP Network Administration*. O'Reilly Media, 2002.

[4] J. Edwards y R. Bramante, *Networking Self-Teaching Guide OSI, TCP/IP, LANs, MANs, WANs, Implementation, Management, and Maintenance*. Wiley, 2009.

[5] M. Burgess, *Principles of Network and System Administration*. Wiley, 2004.

[6] K. R. Fall y W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Pearson, 2012.

[7] K. Siyan y T. Parker, *TCP/IP Unleashed*. Sams Publishing, 2002.

[8] C. M. Kozierok, *The TCP/IP Guide*. Charles M. Kozierok, 2005.