PRACTICA 1:

Servicios de red y programación distribuida en Unix

1 Servicios de red

1.1 Introducción

Se pretenden estudiar algunas facilidades estándares que proporciona Unix para soportar operaciones distribuidas sobre una red. Todas estas facilidades se basan en el modelo cliente-servidor para aplicaciones distribuidas.

Para localizar un servicio en la red, lo primero que tiene que conocer el cliente es la máquina a contactar. Esto se puede proporcionar bien por una cadena de caracteres o por la dirección de red traducida a un número de 32 bits (dirección IP).

Dentro de cada máquina habrá varios terminales de comunicaciones donde los servidores estarán esperando conexiones. Estos terminales se identifican por un número de **puerto** (normalmente 16 bits). Por ejemplo, el servidor *sendmail* siempre espera conexión en el puerto 25, y el demonio *rexecd*, que ofrece un servicio de ejecución de orden remota, en el puerto 512. La correspondencia entre servicios y puertos está almacenada en el archivo */etc/services*. Aquí también se incluye el protocolo que se utiliza y un nombre alternativo para el servicio.

Este método de localizar un servicio es extremadamente primitivo, ya que no se puede proporcionar un nombre de servicio y pedir al sistema que localice una máquina que ofrezca ese servicio (esto sí existe para sistemas de archivos compartidos como RFS de AT&T, y también en el entorno DCE de OSF, el cual proporciona un servicio directorio de celda).

El número de puerto, por ejemplo de *sendmail*, se dice que es **conocido** debido a que este servidor, en todas las máquinas, siempre espera conexión en ese puerto.

Cuando un cliente se conecta a un servidor tiene que crear otro terminal de transporte de su propiedad, el cual tendrá también asignado un número de puerto que será arbitrario, con la excepción de que los números menores de 1024 son **reservados** (sólo procesos ejecutándose con *uid* efectivo de *root* podrán utilizar estos números de puertos). La mayoría de los servidores utilizan puertos reservados. Ésta es la base para proporcionar seguridad a bajo nivel en los servicios de red. Algunos servidores piden que los clientes utilicen puertos reservados cuando intentan la conexión, lo cual asegura un tipo cliente concreto.

1.2 Ejecución de orden remota

La ejecución de órdenes remotas es uno de los principales servicios de red que ofrecen los sistemas operativos. A continuación se dan las diferentes posibilidades para ejecución de órdenes remotas:

1) La principal orden de ejecución remota es *rsh*, lógicamente éste es el programa cliente que se ejecuta en la máquina local. El servidor se denomina *rshd*. El formato de la orden es:

```
rsh <máquina remota> <orden>
```

El método que utiliza el servidor para comprobar la identidad del usuario se denomina **máquinas de confianza**. Este método evita tener que especificar el nombre de usuario y la clave cada vez que se ejecuta una orden remota. Así, para poder ejecutar la orden remota se tienen que satisfacer dos condiciones:

   a. El usuario tiene que tener una cuenta en la máquina remota (normalmente con el mismo nombre de usuario).

   b. El archivo */etc/hosts.equiv* de la máquina remota tiene que tener una entrada con el nombre de la máquina local, o si esto falla, en el directorio de la cuenta en la máquina remota tiene que haber un archivo *.rhosts* que contenga dicha entrada.

2) Una variante de la orden *rsh* es *on*. Es un servicio basado en RPC soportado por del demonio *rexd* y tiene el mismo formato que *rsh*. La diferencia es que crea un entorno en la máquina remota similar al que hay en la máquina local, por ejemplo, todas las variables de entorno locales se pasan explícitamente a la máquina remota. Otra diferencia es que soporta órdenes remotas interactivas, por ejemplo *vi*.

3) Utilizar el servidor *rexecd*. Su ejecución no depende de que el cliente tenga privilegios de supervisor. Este servidor no tiene un programa cliente, pero tiene una función *rexec()* para que los usuarios puedan escribir sus propios programas cliente. Aquí la identificación se realiza introduciendo explícitamente el nombre de usuario y la clave cada vez que se llama al servidor.

4) Utilizar el cliente de propósito general *telnet*. Normalmente, *telnet* conecta con el servidor *telnetd* en el número de puerto TCP 23, pero opcionalmente, se puede utilizar con un número de puerto alternativo donde se encuentre algún otro servicio orientado a texto (para obtener salidas legibles). Por ejemplo, la ejecución de orden remota siguiente devuelve la fecha y hora de la máquina remota *ws1* (pruebe utilizando el nombre de otra máquina del laboratorio):

```
telnet ws1 daytime
```

5) Actualmente el servicio más usado para ejecución remota es el *Secure Shell* (*SSH*), que es similar a *rsh*, pero es un protocolo seguro. SSH fue diseñado para reemplazar a *telnet* y a otros protocolos de acceso remoto como *rlogin*, *rsh* o *rexec* que envían *passwords* en texto plano y son fáciles de interceptar. *SSH* proporciona un canal encriptado con clave pública entre el cliente y el servidor, proporcionando confidencialidad e integridad de los datos sobre redes inseguras, como internet. Sin embargo, como todos los protocolos, no es infalible, como quedó patente en el caso Snowden.

Se pueden obtener los servicios que ofrece actualmente una máquina con la orden:

```
netstat -a
```

O con la siguiente orden:

```
sudo service --status-all
```

La siguiente tabla resume los métodos de ejecución remota:

Servidor	Función cliente	Aplicación cliente	Comentarios
rshd	rcmd()	rsh, rcp	Parte de las utilidades r* de BSD. Identificación basada en el uso de puertos reservados y en el archivo /etc/hosts.equiv
rexd	rex()	on	Servicio basado en RPC. Pasa datos del entorno y soporta uso interactivo
rexecd	rexec()	No existen	Similar a rshd, pero la identificación por explicito nombre de usuario y clave
El que se especifique	No se conoce	telnet	El servicio solicitado debe de ser orientado a texto para obtener una salida legible
sshd	ssh()	ssh	Conexiones seguras, se creó para sustituir a los anteriores.

1.3 Ejemplo SSH

Si el sistema no tiene instalado un servidor *ssh*, lo primero es instalar uno, como *openssh-server o ssh*.

```
sudo apt-get install openssh-server
```

Después de la instalación el servidor debería estar activo. Se puede comprobar su estado con:

```
sudo service ssh status
```

La configuración del servicio se puede cambiar en el fichero /etc/ssh/*sshd_config*. Por ejemplo, cambiar el puerto, permisos de entrada, etc. Se puede editar el fichero con el editor de texto nano.

```
sudo nano /etc/ssh/sshd_config
```

Para permitir a otros usuarios acceder a la máquina a través de ssh, se debe insertar los permisos en el fichero /etc/ssh/*ssh_config* añadiendo al final la línea:

```
AllowUsers <usuario1> <usuario2>
```

Siempre que se cambie la configuración hay que levantar de nuevo el servicio:

```
sudo service ssh restart
```

Para ver todos los servicios activos se puede usar la orden:

```
sudo service --status-all
```

Consulte las opciones de *service* utilizando man.

Para acceder al servidor *sshd* desde otra máquina, solo hay que tener instalado un cliente ssh, la mayoría de los sistemas Linux lo tienen ya instalado. Simplemente se accede con un nombre de usuario permitido en el sistema y la dirección IP del servidor.

```
ssh –l <usuario> <nombre-servidor> -p2222
```

Esta orden convierte su terminal en un terminal remoto, por lo tanto ahora puede utilizar diferentes comandos en la máquina remota, por ejemplo, *ls*, *cd*, *ps*, etc.

Ejercicio 1: Compruebe que el servidor ssh está activo. Cree un usuario para que un compañero suyo pueda acceder a su máquina. Acceda a la máquina de un compañero a través de ssh. Pruebe algunos comandos en la máquina remota.

Nota: Normalmente el puerto de ssh es el 22, pero en los laboratorios está redirigido al puerto 2222. En los laboratorios al estar instalado Ubuntu en un máquina virtual no se puede entrar mediante el cliente *ssh* con la dirección IP del servidor, se debe utilizar siempre con el nombre de la máquina. El nombre de máquina se encuentra en una pegatina sobre la CPU y tiene el siguiente formato `eixxxxx`, siendo las x dígitos numéricos.

Ejercicio 2. El entorno gráfico en el servidor está activado en los laboratorios, (con la orden `X11Forwarding yes` en el archivo de configuración del *sshd*, */etc/ssh/sshd_config*), compruébelo!

Para acceder con un cliente al servidor *sshd* haciendo uso del entorno gráfico se debe poner la opción *–X* en la petición de conexión del cliente:

```
ssh –X <usuario>@<IP-servidor> -p2222
```

Compruebe que puede acceder al servidor en entorno gráfico probando ejecutar una aplicación del servidor de entorno gráfico (como *xeyes* o *xclock* o *xterm*, que están instalados en el laboratorio). Puede comprobar que las aplicaciones se abren con los mismos colores y configuración que tenga en el servidor.

Desactive el entorno gráfico remoto cambiando la configuración de */etc/ssh/sshd_config*, desactivando la función X11Forwarding:

```
#X11Forwarding yes
```

Reinicie el servidor sshd, y compruebe que ahora no se pueden activar los servicios X11 remotamente.

Ejercicio 3. *Secure Copy* (*SCP*) es una orden para copiar ficheros de una máquina a otra que utiliza el protocolo ssh para comunicarse. Cree un nuevo archivo de texto en el cliente y cópielo del cliente al servidor mediante la orden *scp*, consulte el *man* para conocer las opciones de uso. Compruebe que el archivo se ha copiado en el servidor. **Nota**: Recuerde que el puerto en los laboratorios es 2222.

Ejercicio 4. Análogamente *Secure File Transfer Protocol* (*SFTP*) es un *ftp* que funciona sobre *ssh*, permite transferir ficheros entre máquinas. Con ayuda del *man* consulte cómo conectar con un servidor *sftp* y transfiera un fichero (orden *get*) desde el servidor al cliente.

Ejercicio 5. Creación de llaves para utilizar en SSH

Si nos conectamos en SSH con llaves, las máquinas realizarán la autenticación, sin necesidad de que tengamos que introducir nuestro password cada vez que nos conectemos. Cuando un cliente intenta acceder a un servidor SSH, comprueba si en el servidor existe una llave pública del cliente, en cuyo caso el servidor dejará acceder al cliente sin necesidad de introducir password, y encriptará la comunicación con la clave pública del cliente, que solo se puede desencriptar con la clave privada correspondiente. Para conseguir esto el cliente debe crear una clave pública y privada, y compartir la clave pública con el servidor siguiendo estos pasos:

1. Lo primero que hay que hacer es crear un llave pública y una llave privada en el cliente, con el comando ssh-Keygen. La llave privada la mantendremos nosotros y nunca se la daremos a nadie, la llave pública es la que compartiremos con otros, para encriptar los mensajes con esa llave, y desencriptarlos nosotros con nuestra llave privada. Estas llaves se crean una sola vez, y se pueden compartir con todas las máquinas que queramos. Compruebe con el *man* las opciones del comando.

```
ssh-keygen -b 4096 -t rsa
```

2. La llave pública generada se la pasamos al servidor, para que la conozca, con el comando *ssh-copy-id*:

```
ssh-copy-id <usuario>@<servidor> -p2222
```

3. Conecte con el servidor mediante ssh y comprobe que no es necesario introducir el password.