TEMA 3: SSH
ADMINISTRACIÓN DE SERVICIOS

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS

2020/2021

Cristian Martín Fernández

Contenido

- SSH
- Instalación Servidor OpenSSH
- Convecciones Ubuntu
- Conexión sin contraseña
- SCP
- SFTP
- SSH Tunneling
- Gestión de usuarios SFTP

SSH

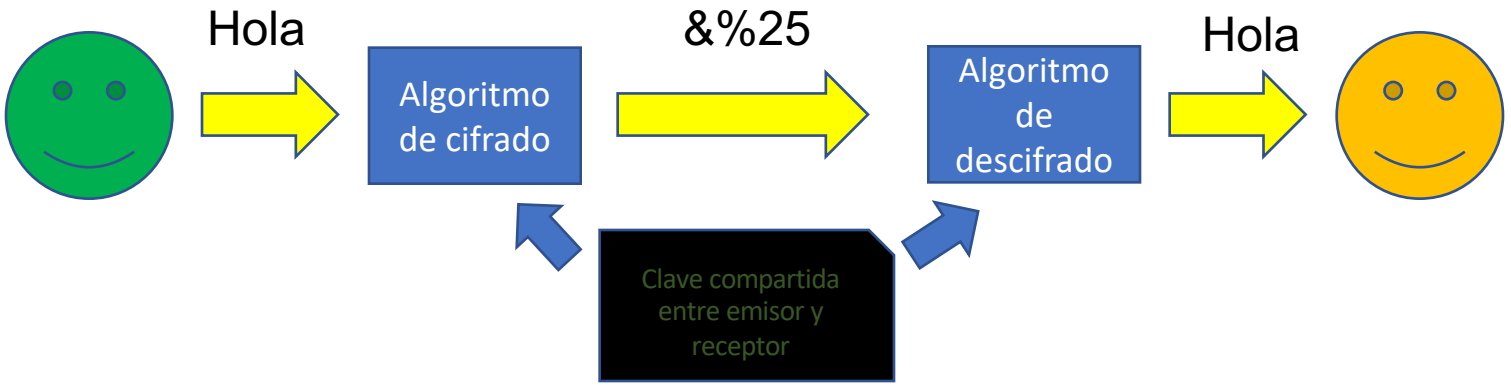
- Secure SHell (SSH) nos proporciona un canal encriptado y seguro por el que podemos acceder y ejecutar comandos desde una maquina virtual remota.
- SSH funciona por defecto en el puerto TCP 22.
- Sustituto de Telnet, que transmite la información en texto plano.
- Además del uso de una consola remota estándar, SSH también permite un número de acciones interesantes y muy útiles:
  - instalar túneles,
  - ejecutar aplicaciones remotas,
  - transferencias de archivos.
- Utiliza tres tipos de cifrado:
  - Cifrado simétrico
  - Cifrado asimétrico
  - Hashing

SSH

- Objetivos de la encriptación o cifrado:
  - Confidencialidad
    - Sólo puede acceder al contenido real su legítimo destinatario.
  - Integridad
    - La información no puede ser alterada por un tercero sin ser detectado.
  - No repudio
    - El emisor de la información no puede dejar de reconocer su autoría. El receptor no puede negar la recepción.
  - Autenticación
    - Tanto emisor como receptor pueden estar seguros de la identidad del otro.

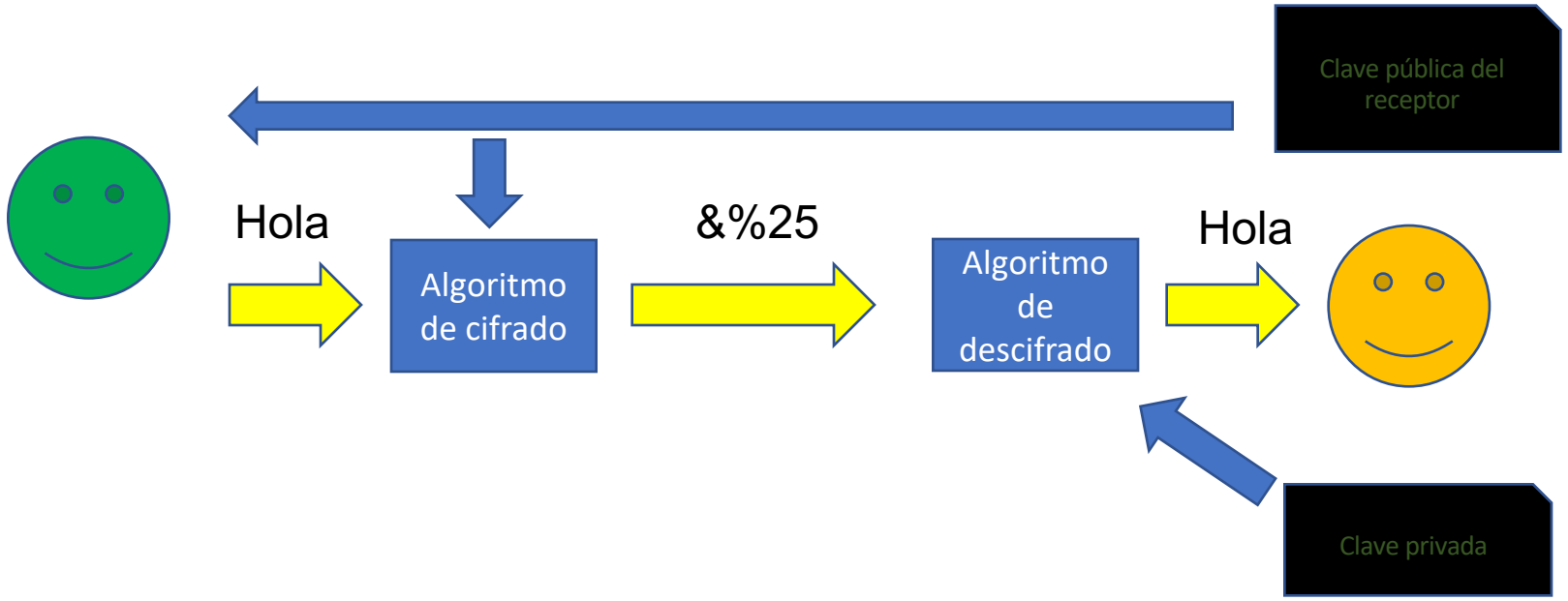
Cifrado simétrico

- Se utiliza la misma clave secreta para cifrar como para descifrar tanto en el cliente como en el servidor.
- La clave nunca se revela a terceros y no debe transmitirse sin cifrar.
- Esta clave se genera en mediante un algoritmo de intercambio de claves (por ejemplo: Diffie-Hellman)
- Una vez generada, todo el tráfico es encriptado de forma eficiente.
- Algoritmos: AES, DES, Blowfish, 3DES, CAST, RC4, etc.



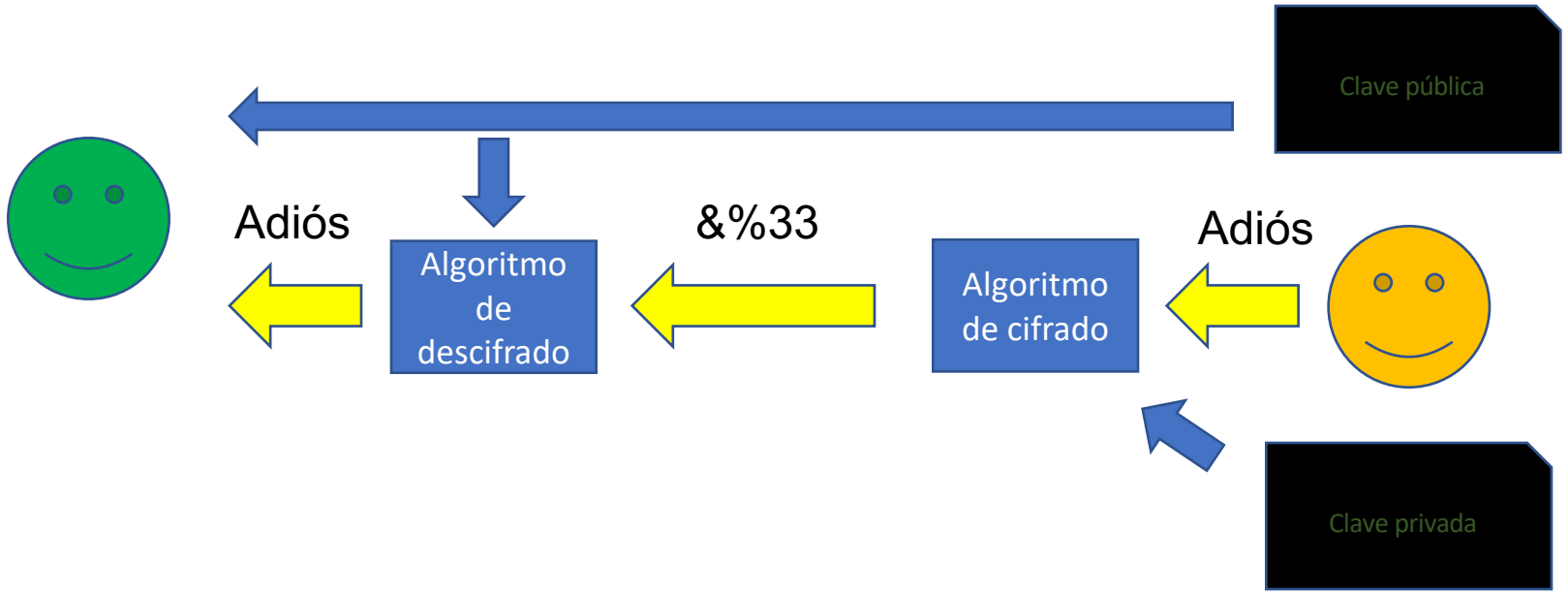
Cifrado asimétrico (confidencialidad)

- Se utilizan dos claves distintas, para cifrar y descifrar (publica y privada).
- Existe una relación entre clave la clave pública y la privada, pero no es posible obtener la clave privada a partir de la pública.
- La clave pública es conocida por todos.
- La clave privada del receptor solo la conoce el propio receptor.
- El emisor cifra el mensaje utilizando la clave pública.
- El receptor descifra el mensaje utilizando su clave privada.



Cifrado asimétrico (autenticidad)

- También se puede utilizar para estar seguros de la autenticidad del emisor.
- Si el emisor cifra el mensaje con su clave privada solo descifrando con su clave pública se obtiene el original.
- Así se garantiza autenticidad y no repudio (firma digital de documentos).



SSH

- Cifrado asimétrico
  - Se suele usar para intercambiar la clave simétrica al inicio de la sesión SSH. Luego se utiliza cifrado simétrico que es más eficiente.
  - Algoritmos: RSA, DSA, etc.

Hashing

- Asegura que un mensaje entre un emisor y un receptor no ha sido modificado por un tercero.
- Es fácil generar un hash criptográfico a partir de una entrada determinada, pero es imposible generar la entrada a partir del hash.
- Algoritmos más utilizados:
  - MD5 (Message Digest, v5)
  - SHA-1 (Secure Hash Algorithm, v1)
  - SHA-2 (Secure Hash Algorithm, v2)
- El receptor comparará el HASH original con el que se produce al recibir y descifrar la información:
  - Si son iguales el mensaje no ha sido alterado.
  - Si son distintos el mensaje ha sido modificado.

Pasos en la apertura de una sesión SSH

1. El cliente SSH abre conexión con el servidor SSH.
2. Cliente y servidor negocian la versión del protocolo.
3. El servidor envía al cliente su clave pública RSA.
4. El cliente mira que la clave coincida con la almacenada.
5. Si no tenemos la clave pública del servidor, se pide confirmación al usuario (riesgo) y se almacena clave.
6. Ambas partes deciden un algoritmo de cifrado simétrico y generan aleatoriamente una clave para dicho algoritmo.
7. La clave se comparte utilizando la clave pública del servidor.
8. El servidor descifra el mensaje anterior con su clave RSA privada y a partir de entonces toda comunicación es encriptada usando la clave compartida.

Instalación Servidor OpenSSH

- Instalación servidor OpenSSH:
  - `$ sudo apt-get install openssh-server`

- Una vez que el paquete está instalado, el proceso sshd arrancará y se podrá acceder de forma remota al servidor.

- Para verificar que SSH está corriendo se puede utilizar:
  - `$ systemctl status ssh`

Convecciones Ubuntu

- /etc/ssh/
- Este directorio contiene todos los archivos de configuración, tanto del cliente como del servidor de OpenSSH
- /etc/ssh/sshd_config
- Archivo principal de configuración del servidor SSH donde encontraremos los parámetros por defecto.
- Es importante tener en cuenta que por defecto el acceso a root por SSH está habilitado, aunque Ubuntu no instale accesos con root en el sistema por defecto.
- Para deshabilitar los accesos de root, podemos editar el archivo de configuración y definir el valor “PermitRootLogin” a “no:
- PermitRootLogin = no

Convecciones Ubuntu

- /etc/ssh/ssh_config
- En este archivo encontraremos los parámetros por defecto para el cliente SSH.
- ~/.ssh
  - authorized_keys
    - Cada línea contiene una clave pública de un cliente.
    - Especifica qué usuarios pueden conectarse al servidor utilizando una autenticación basada en clave pública.
  - known_hosts
    - Permite al cliente autenticar al servidor, es decir, saber si es un servidor conocido.
    - Contiene una línea por servidor conocido.
    - La primera vez, el cliente debe comprobar si el servidor es quien dice ser.

Convecciones Ubuntu

- /etc/ssh/ssh_host_dsa_key
- /etc/ssh/ssh_host_dsa_key.pub

- Estos archivos proporcionan las claves privadas y públicas respectivamente del servidor SSH.

- De forma análoga a los dos ficheros del puntos anterior, también se proporcionan las claves privadas y públicas respectivamente para los algoritmos RSA, ECDSA y ED255119.

Convecciones Ubuntu

- /etc/init.d/ssh
- Es el script de inicio para el servidor de OpenSSH y proporciona los comanados estándares “start”, “stop”, “restart” y “reload”
- /var/log/auth.log
- El servidor de OpenSSH guarda en este archivo, su información de registros, como por ejemplo: mensajes de notificación, errores y acceso de usuarios.
- Para consultar el log del servicio ssh podemos utilizar:
  - `$ journalctl –u ssh`

Servidor OpenSSH

- Abrir sesión de SSH:
  - `$ ssh username@hostname`
- Si tenemos SSH abierto en otro puerto podemos utilizar la opción -p:
  - `$ ssh –p 2222 username@hostname`
- También podemos ejecutar un comando directamente cuando nos conectamos (termina sesión):
  - `$ ssh username@hostname <comando>`
- Para abandonar o salir de una sesión SSH:
  - `$ exit, o también $^d`

Conexión sin contraseña (basada en clave pública)

- Se puede acceder a SSH a través de claves públicas, dónde no es necesario contraseñas.
- Paso 1: Generar clave pública en el PC del cliente:
  - `ssh-keygen -t rsa -b 4096`

- Paso 2: Copiar clave pública (.ssh/id_rsa.pub) al servidor en el fichero .ssh/authorized_keys. Cada línea representará una clave pública de un cliente.
- `$ cat ~/.ssh/id_rsa.pub | ssh username@remote_host "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"`

- También se puede utilizar la herramienta ssh-copy-id para copiar la clave pública (Paso 2):
  - `ssh-copy-id -i ~/.ssh/.ssh/id_rsa.pub user@host`

Conexión sin contraseña (basada en clave pública)

- Cuando se cambia la autenticación (ej., certificado, método, etc.) es conveniente eliminar la entrada en .ssh/known_hosts ya que sino se confía en el servidor con la configuración anterior.

- Esto habría que hacerlo en los clientes si cambia la clave del servidor.

Deshabilitar autenticación por contraseña

Editamos el fichero de configuración principal:

- `$ sudo nano /etc/ssh/sshd_config`
- Y podemos las siguientes características:
- `PasswordAuthentication no`
- `ChallengeResponseAuthentication no`
- `UsePAM no`
- `PermitRootLogin no`

Deshabilitar autenticación por contraseña

Se puede comprobar si se puede acceder con clave aunque ya se tenga una clave pública instalada. Por ejemplo:

- `$ ssh -p 2222 cristian@localhost  -o PubkeyAuthentication=no`

SCP

- Secure CoPy hace uso de SSH para realizar copias seguras y encriptadas de archivos.
- Se caracteriza por ser una herramienta no interactiva. Realiza su trabajo y termina.
- Para copiar un único archivo:
  - `$ scp <archivo> <usuario>@<IP o hostname>:<Destino>`
- Ejemplo:
  - `$ scp -P 2222 helloworld.cpp cristian@localhost:helloworld.cpp`
- Para copiar directorios y subdirectorios, utilizamos la opción –r. Ejemplo:
  - `$ scp -r -P 2222 tmp/ cristian@localhost:tmp/`

SFTP

- SSH File Transfer Protocol hace uso también de SSH para permitir una sesión interactiva entre el servidor y la máquina local.

- SFTP es ampliamente utilizado por muchos clientes para intercambiar información con servidores. Un ejemplo, es el conocido cliente FileZilla

- Para conectarnos desde una Shell podemos utilizar el siguiente ejemplo:
  - `$ sftp -P 2222 cristian@localhost`

SFTP Comandos útiles:

`get remote-path [local-path]`	Descargar fichero en local
`put local-path [remote-path]`	Subir fichero al servidor
`lls [ls-options [path]]`	Hacer un ls local
`ls [path]`	Hacer un ls del servidor
`lcd path`	Cambia el directorio local a path
`cd path`	Cambia el directorio server a path
`lmkdir path`	Crea un directorio local
`mkdir path`	Crea un directorio remoto
`exit`	Sale de sftp
`!comando`	Ejecutar comando en local
`?`	Ayuda

SSH Port Forwarding == SSH Tunneling

- El túnel SSH es un método que permite crear una canalización de información a través de SSH entre dos hosts a través del reenvío de puertos.

- Un ejemplo clásico: conectarnos a un servidor SSH para encapsular y cifrar todo el tráfico a través de él. Comportamiento similar a VPN.

- Los túneles SSH permiten agregar seguridad de red a aplicaciones que no permiten de forma nativa.

- Los túneles SSH también se pueden utilizar para ocultar la fuente de un ataque.

Local port forwarding

- Un túnel SSH local o “Local port forwarding” permite acceder a puertos/servicios desde la red local que no están expuestos a Internet desde un servidor.

- Se utiliza para servicios concretos, ej. Una base de datos, Apache, etc.

- Ejemplo para acceder a Apache desde la máquina guest (sin reenvío de puertos en VirtualBox) y SSH sí con reenvío de puertos en el puerto 2222 de la MV:
  - `$ ssh -p 2222 -N -L 8888:127.0.0.1:80 cristian@localhost`

- Con la opción -f se ejecuta en background:
  - `$ ssh -p 2222 -f -N -L 8888:127.0.0.1:80 cristian@localhost`

- Con la opción –N se evita que se inicie una sesión de SSH.

- Para hacer un “Remote Port Forwarding”, justo lo contrario que con local port forwarding, se utiliza la opción –R en vez de –L.

Dynamic Port Forwarding

- Un túnel SSH con reenvío dinámico creará un proxy SOCKS. Funciona de manera similar a una VPN.

- Todo el tráfico enviado a través del proxy se enviaría a través del servidor SSH.

- Ejemplo para el reenvío en el puerto 8081:
  - `$ ssh -D 8081 -p 2222 cristian@127.0.0.1`

Gestión de usuarios

Se pueden añadir permisos específicos en el fichero principal de configuración de SSH (sshd_config) para controlar el acceso de usuarios (ej., para que se conecten únicamente por SFTP y a un directorio específico).

```
Subsystem sftp internal-sftp

Match Group sftpusers
ForceCommand internal-sftp
PasswordAuthentication yes
ChrootDirectory /sftp/
```

Hay que darle permisos de root a la carpeta ráiz para que SSH pueda entrar:

```
sudo chown root:root /sftp/
sudo chmod 755 /sftp/
```

Y permisos al usuario para su carpeta:

```
sudo chown aso:sftpusers /sftp/aso
```

Bibliografía

- OpenSSH Server: https://ubuntu.com/server/docs/service-openssh
- How to disable ssh password login: https://www.cyberciti.biz/faq/how-to-disable-ssh-password-login-on-linux/
- SSH Essentials: https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys
- SSH Transfer Files: https://help.ubuntu.com/community/SSH/TransferFiles
- How to use SSH Tunneling https://www.howtogeek.com/168145/how-to-use-ssh-tunneling/
- Enable Access SFTP without Shell: https://www.digitalocean.com/community/tutorials/how-to-enable-sftp-without-shell-access-on-ubuntu-16-04