UNIVERSIDAD LUTERANA SALVADOREÑA

Facultad de las Ciencias del Hombre y la Naturaleza

Carrera:
Licenciatura en Ciencias de la Computación

Materia:
Introducción al software libre

Catedrática:
Lic. Pedro Noble

Ciclo:

02-2018

Alumnos:
Hersson Omar Cañas Gutiérrez.
José Danilo García Hernández.
Franklin Antonio Cortez Quiñonez.
Henrry Geovanny Castañeda Cáceres.

Objetivos

- **Objetivos General.**

  - Dar a conocer que es un servidor ssh y cómo funciona.

- **Objetivos específicos**.

  - Obtener una comprensión en profundidad de cómo es el funcionamiento de SSH y cómo puede ayudar a los usuarios.

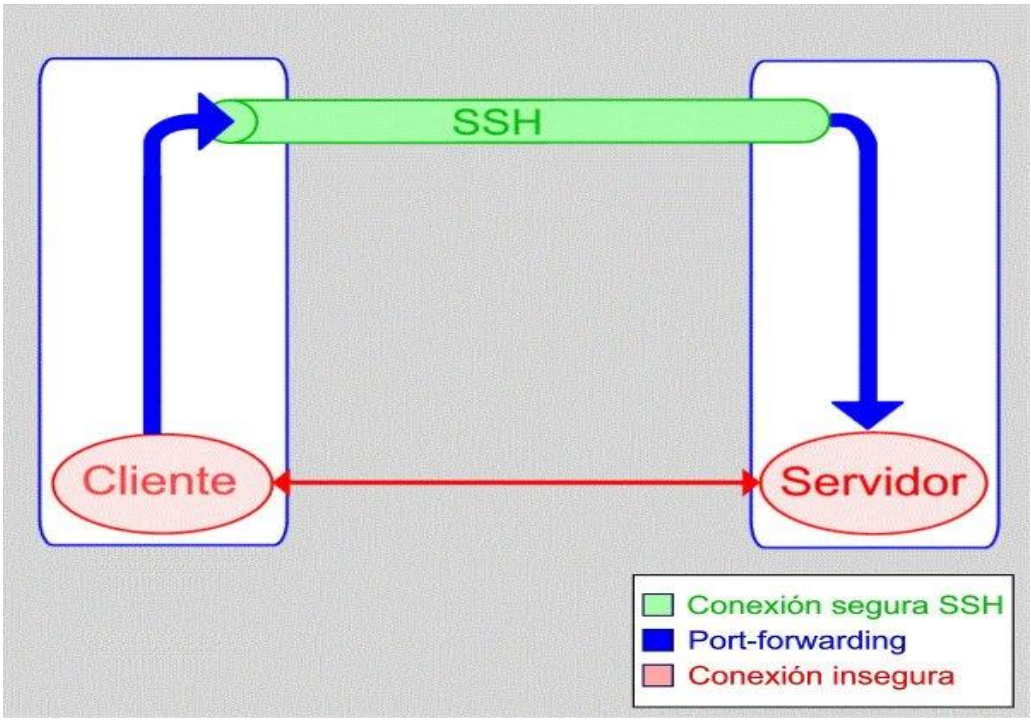
  - Comprender los aspectos de seguridad de esta tecnología.

Introducción

En el presente trabajo daremos a conocer que es un servidor ssh, como funciona y como se puede instalar en un ordenador con una distribución Linux, aprenderemos como un usuario se conecta a un sistema mediante ssh y que los usuarios tienen una variedad de herramientas que les permiten interceptar y redirigir el tráfico de la red para ganar acceso al sistema veremos cómo se programa a pasos desde una terminal utilizando diferentes comandos ya que esto nos permitirá manejar de una misma manera que lo haría si estuviera operando físicamente el equipo remoto.

Marco Histórico

SERVIDOR SSH



SSH o Secure Shell, es un protocolo de administración remota que permite a los usuarios controlar y modificar sus servidores remotos a través de Internet. El servicio se creó como un reemplazo seguro para el Telnet sin cifrar y utiliza técnicas criptográficas para garantizar que todas las comunicaciones hacia y desde el servidor remoto sucedan de manera encriptada. Proporciona un mecanismo para autenticar un usuario remoto, transferir entradas desde el cliente al host y retransmitir la salida de vuelta al cliente.

El servidor de Shell seguro o SSH (Secure Shell) es un servicio muy similar al servicio telnet ya que permite que un usuario acceda de forma remota a un sistema Linux pero con la particularidad de que, al contrario que telnet, las comunicaciones entre el cliente y servidor viajan cifradas desde el primer momento de forma que si un usuario malintencionado intercepta los paquetes de datos entre el cliente y el servidor, será muy difícil que pueda extraer la información ya que se utilizan sofisticados algoritmos de cifrado.

Para que un usuario se conecte a un sistema mediante ssh, deberá disponer de un cliente ssh. Durante el proceso de autentificación, cuando el usuario proporciona el nombre y la contraseña, se utiliza cifrado asimétrico, pero en el resto de la sesión se utiliza cifrado simétrico por su menor necesidad de procesamiento.

SSH (Secure SHell, en español: intérprete de órdenes seguro) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder servidores privados a través de una puerta trasera (también llamada backdoor). Facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente.

HISTORIA DEL SSH

El protocolo SSH fue desarrollado en el año 1995 por el finlandés TatuYlönen, quien publicó su trabajo bajo una licencia de libre uso, pero ante el éxito del programa desarrollado pronto registró la marca SSH y fundó la empresa SSH Communications Security con fines comerciales, la cual permitía el uso del protocolo gratuitamente para uso doméstico y educativo.

Ante este cambio en la política de uso del protocolo, los desarrolladores del sistema operativo OpenBSD empezaron a desarrollar en el año 1999 una versión libre de este protocolo que recibió el nombre de OpenSSH.

Desde la aparición de este protocolo han sido dos las versiones que han estado activas. En la primera de ellas, se ofrecía una alternativa a las sesiones interactivas mediante el uso de herramientas como TELNET, RSH o RLOGIN entre otras, sin embargo pronto se descubrió que este protocolo tenía un punto débil que permitía a los hackers introducir datos en los flujos cifrados.

Ante este problema, en el año 1997 fue lanzada la versión 2, donde una serie de medidas solucionaban el problema descubierto en la primera versión. Además de corregir ese problema, esta segunda versión incorporaba el protocolo SFTP (Secure File Transfer Protocol – Protocolo seguro de transferencia de archivos) que proporciona la funcionalidad necesaria para la transferencia y manipulación de archivos de forma segura. SSH (Secure SHell,: intérprete de órdenes segura).

¿QUE REEMPLAZA SSH?

SSH está diseñado para reemplazar los métodos más viejos y menos seguros para registrarse remotamente en otro sistema a través de la shell de comando, tales como telnet o rsh. Un programa relacionado, el scp, reemplaza otros programas diseñados para copiar archivos entre hosts como rcp. Ya que estas aplicaciones antiguas no encriptan contraseñas entre el cliente y el servidor, evite usarlas mientras le sea posible. El uso de métodos seguros para registrarse remotamente a otros sistemas reduce los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto.

¿POR QUE USAR SSH?

Los usuarios tipo ‘hackers’ tienen a su disposición una variedad de herramientas que les permiten interceptar y redirigir el tráfico de la red para ganar acceso al sistema. En términos generales, estas amenazas se pueden catalogar del siguiente modo: Intercepción de la comunicación entre dos sistemas.

En este escenario, existe un tercero en algún lugar de la red entre entidades en comunicación que hace una copia de la información que pasa entre ellas. La parte interceptora puede interceptar y conservar la información, o puede modificar la información y luego enviarla al recipiente al cual estaba destinada. Este ataque se puede montar a través del uso de un paquete sniffer una utilidad de red muy común. Personificación de un determinado host Con esta estrategia, un sistema interceptor finge ser el recipiente a quien está destinado un mensaje. Si funciona la estrategia, el sistema del usuario no se da cuenta del engaño y continúa la comunicación con el host incorrecto. Ambas técnicas interceptan información potencialmente confidencial y si esta intercepción se realiza con propósitos hostiles, el resultado puede ser catastrófico.

Si se utiliza SSH para inicios de sesión de consola remota y para copiar archivos, se pueden disminuir estas amenazas a la seguridad notablemente. Esto es porque el cliente SSH y el servidor usan firmas digitales para verificar su identidad. Adicionalmente, toda la comunicación entre los sistemas cliente y servidor es encriptada. No servirán de nada los intentos de falsificar la identidad de cualquiera de los dos lados de la comunicación ya que cada paquete está cifrado por medio de una llave conocida sólo por el sistema local y el remoto.

Marco Teórico

¿COMO FUNCIONA SSH?

Si está usando Linux o Mac, entonces usar SSH es muy simple. Si utiliza Windows, deberá utilizar un cliente SSH para abrir conexiones SSH.

Para usuarios de Mac y Linux, dirígete a tu programa de terminal y sigue el procedimiento siguiente:

El comando SSH consta de 3 partes distintas: ssh

{user}@{host}

El comando de clave SSH le indica a su sistema que desea abrir una Conexión de Shell Segura y cifrada. {user} representa la cuenta a la que desees acceder. Por ejemplo, puede que

desees acceder al usuario root, que es básicamente para el administrador del sistema con derechos completos para modificar cualquier cosa en el sistema. {Host} hace referencia al equipo al que desees acceder. Esto puede ser una dirección IP (por ejemplo, 244.235.23.19) o un nombre de dominio (por ejemplo, www.xyzdomain.com).

Cuando pulsa enter, se te pedirá que introduzcas la contraseña de la cuenta solicitada. Al escribirla, nada aparecerá en la pantalla, pero tu contraseña, de hecho, se está transmitiendo. Una vez que hayas terminado de escribir, pulsea enter una vez más. Si su contraseña es correcta, recibirá una ventana de terminal remota.

PROTOCOLO SSH (SECURE SHELL)

Es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

SSH está diseñado para reemplazar los métodos más viejos y menos seguros para registrarse remotamente en otro sistema a través de la shell de comando, tales como TELNET o RSH.

Un programa relacionado, el SCP, reemplaza otros programas diseñados para copiar archivos entre hosts como RCP. Ya que estas aplicaciones antiguas no encriptan contraseñas entre el cliente y el servidor, evite usarlas mientras le sea posible.

El uso de métodos seguros para registrarse remotamente a otros sistemas reduce los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto.

COMO FUNCIONA EL PROTOCOLO SSH:

- El cliente inicia una conexión TCP sobre el puerto 22 del servicio. Este puerto es el que utiliza por defecto.

- El cliente y el servidor se ponen de acuerdo en la versión del protocolo a utilizar, así como el algoritmo de cifrado utilizado para el intercambio de la información.

- El servidor, que tiene en su poder dos claves (una privada y una pública), manda su clave pública al cliente.

- Cuando el cliente recibe la clave enviada por el servidor, la compara con la que tiene almacenada para verificar su autenticidad. El protocolo SSH exige que el cliente la confirme la primera vez.

- Con la clave pública del servidor en su poder, el cliente genera una clave de sesión aleatoria, creando un mensaje que contiene esa clave y el algoritmo seleccionado para la encriptación de la información. Toda esa información es enviada al servidor haciendo uso de la clave pública que envió en un paso anterior de forma cifrada.

- Si todo es correcto, el cliente queda autenticado, iniciando la sesión para comunicarse con el servido.

SEGURIDAD.

SSH trabaja de forma similar a como se hace con telnet La diferencia principal es que SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión; aunque es posible atacar este tipo de sistemas por medio de ataques de REPLAY y manipular así la información entre destinos.

EL PROTOCOLO SSH PROPORSIONA LOS SIGUIENTES TIPOS DE PROTECCIÓN:

- Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor al que se conectó anteriormente.

- El cliente transmite su información de autenticación al servidor usando una encriptación robusta de 128 bits.

- Todos los datos enviados y recibidos durante la sesión se transfieren por medio de encriptación de 128 bits, lo cual los hacen extremamente difícil de descifrar y leer. Ya que el protocolo SSH encripta todo lo que envía y recibe, se puede usar para asegurar protocolos inseguros.

El servidor SSH puede convertirse en un conducto para convertir en seguros los protocolos inseguros mediante el uso de una técnica llamada reenvío por puerto, como por ejemplo POP, incrementando la seguridad del sistema en general y de los datos.

GUÍA DE INTALACIÓN DE SSH EN LINUX

- root@server:~# apt install openssh-server openssh-client

Configuración

Todas las posibilidades de configuración del servidor *ssh* están en el archivo /etc/ssh/sshd_config.

Para configurar el servidor hay que indicar las direcciones ip donde el servicio va a "escuchar" y "responder". En este caso son conexiones ligadas a la dirección *192.168.1.100*, a través del puerto *22* y utilizando la versión 2 del protocolo *ssh*:

/etc/ssh/sshd_config

```
1  #Port 22
2  #AddressFamily any
3  #ListenAddress 0.0.0.0
4  #ListenAddress ::
5  ListenAddress 192.168.1.100
6
7  #[...]
```

Por seguridad, conviene desactivar el *login* como *root*. En esta configuración, para adquirir los privilegios de *root*, hay que iniciar una sesión *login* con una cuenta de usuario normal y,

después, adquirir los privilegios de *root*. Así se previenen los ataques a la contraseña *password* de la cuenta *root*.

/etc/ssh/sshd_config

```
#[...]

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password

#[...]
```

También hay que impedir las sesiones o *logins* de cuentas sin contraseña (*Empty Passwords*):

/etc/ssh/sshd_config

```
#[...]

#PermitEmptyPasswords no

#[...]
```

Por último, para que el servicio tome los cambios que se han hecho en el archivo de configuración, se reinicia el servicio:

```
root@server:~# systemctl restart ssh
```

Verificación

Clientes Linux

Debe ser posible establecer una conexión *ssh* a la dirección 192.168.1.100.

La primera vez que se establece la conexión, ésta tiene que ser confirmada, porque el servidor no está registrado en la lista de los sistemas conocidos por el cliente.

```
fribeiro@laptop:~$ ssh fribeiro@192.168.1.100
```

```
2  The authenticity of host '192.168.56.101 (192.168.56.101)' can`t be established.
3  ECDSA key fingerprint is SHA256:eC0Wva7QKqCiy5imegiPv/NxfCup3Dmjm1n4aB3LH2I.
4  Are you sure you want to continue connecting (yes/no)? yes
5  Warning: Permanently added '192.168.1.100' (ECDSA) to the list of known hosts.
6  fribeiro@192.168.1.100`s password:
7  Linux stretch 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2 (2017-06-12) x86_64
8
9  The programs included with the Debian GNU/Linux system are free software;
10 the exact distribution terms for each program are described in the
11 individual files in /usr/share/doc/*/copyright.
12
13 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
14 permitted by applicable law.
15 fribeiro@server:~$ logout
16 Connection to 192.168.1.100 closed.
17 fribeiro@laptop:~$
```

Las conexiones con el login root no se aceptarán:

```
1  fribeiro@laptop:~$ ssh root@192.168.1.100
2  root@192.168.1.100´s password:
3  Permission denied, please try again.
4  root@192.168.1.100´s password:
5  Permission denied, please try again.
6  root@192.168.1.100´s password:
7  Permission denied (publickey,password).
8  fribeiro@laptop:~$
```

Para arrancar el servidor

sudo /etc/init.d/ssh start

Para parar el servidor

sudo /etc/init.d/ssh stop

Para reiniciar el servidor

sudo /etc/init.d/ssh restart

DISTINTOS TIPOS DE LLAVES PUBLICAS Y PRIVADAS

El protocolo que sigue este mecanismo es una combinación de Hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital o el no repudio de transacciones electrónicas.

Un ejemplo muy simple de cómo funciona este proceso sería que al enviar una información importante pero el mensajero resulta inseguro para nosotros, escribimos el mensaje en un papel, se introduce en una caja de metal con un candado y se envía. La caja llega a su destino sin problema, pero el destinatario no puede leerla, pues no puede abrir el candado. Lo primordial sería enviarle la llave, pero aunque sea por otro medio u otro mensajero, puede que haya otro nivel de riesgo, al comprometer la seguridad de la llave, confiándola a extraños. Así funciona el ciframiento convencional. Cuando decidimos brindar información confidencial, pero queremos que solo el receptor pueda leerla, ahí es cuando entra en juego este método.

A través de un juego entre distintos caracteres informáticos, la llave pública solamente puede cifrar. La llave privada puede descifrar o hacer las dos cosas, aunque esto último no es tan importante. Se recibe la llave pública del destinatario y con ella cifra la información que se le enviará. Una vez cifrada, no se puede ver la información. Al enviarla, en un correo por ejemplo, el destinatario la recibe y la descifra con su llave privada. Es por ello, que las llaves permiten a los usuarios autenticarse frente a otros usuarios y usar la información de los certificados de identidad (por ejemplo, las claves públicas de otros usuarios) para cifrar y descifrar mensajes, firmar digitalmente información, garantizar el no repudio de un envío,

y otros usos. Dependerá en parte importante, cómo se guarden las claves privada, puesto que existen dispositivos especiales denominados tokens de seguridad para facilitar la seguridad de la clave privada, así como evitar que ésta pueda ser exportada. Estos dispositivos pueden incorporar medidas biométricas, como la verificación de huella dactilar, que permiten aumentar la confiabilidad, dentro de las limitaciones tecnológicas, en que sólo la persona dueña del certificado pueda utilizarlo.

GENERANDO LAS CLAVES SSH.

Se analiza el proceso de generación de claves públicas y privadas

Para poder utilizar SSH, primero debemos generar las claves. Al generarlas, obtenemos dos ficheros, uno con el contenido de la clave privada, y el otro con el contenido de la clave pública.

En el servidor SSH pondremos nuestra clave pública, y la clave privada deberemos guardarla y tendremos cuidado de no dársela a nadie.

De esta forma, si tuviéramos 5 servidores a gestionar, configuraríamos nuestra clave publica en los 5, y a través de la clave privada configurada en nuestra máquina, accederíamos a todos los servidores sin ningún problema.

GENERANDO LAS CLAVES EN LINUX

Para crear las claves, tenemos que estar validados como el usuario que las va a utilizar. Si las generamos como root, solo el root podrá utilizarlas.

```
ssh-keygen -t rsa
```

aceptamos la carpeta que nos propondrá dentro del home *~/.ssh/id_rsa*

Luego introducimos la clave ssh o paraphrase que nos pide. Conviene que esta clave sea distinta a la que tenemos como usuarios en el servidor.

Tras hacer esto, ges:

- Clave privada: ~/.ssh/id_rsa
- Clave pública: ~/.ssh/id_rsa.pub

Luego hay que modificar los permisos de la carpeta ssh:

chmod 755 ~/.ssh.

GENERANDO TU LLAVE PUBLICA

Tal y como se ha comentado, muchos servidores Git utilizan la autentificación a través de claves públicas SSH. Y, para ello, cada usuario del sistema ha de generarse una, si es que no la tiene ya. El proceso para hacerlo es similar en casi cualquier sistema operativo. Ante todo, asegurarte que no tengas ya una clave. Por defecto, las claves de cualquier usuario SSH se guardan en la carpeta ~/.ssh de dicho usuario. Puedes verificar si tienes ya unas claves, simplemente situandote sobre dicha carpeta y viendo su contenido:

```
$ cd ~/.ssh
$ ls
authorized_keys2  id_dsa      known_hosts
config            id_dsa.pub
```

Has de buscar un par de archivos con nombres tales como 'algo' y 'algo.pub'; siendo ese "algo" normalmente 'iddsa' o 'idrsa'. El archivo terminado en '.pub' es tu clave pública, y el otro archivo es tu clave privada. Si no tienes esos archivos (o no tienes ni siquiera la carpeta '.ssh'), has de crearlos; utilizando un programa llamado 'ssh-keygen', que viene incluido en el paquete SSH de los sistemas Linux/Mac o en el paquete MSysGit en los sistemas Windows:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/schacon/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/schacon/.ssh/id_rsa.
Your public key has been saved in /Users/schacon/.ssh/id_rsa.pub.
```

```
The key fingerprint is:
43:c5:5b:5f:b1:f1:50:43:ad:20:a6:92:6a:1f:9a:3a schacon@agadorlaptop.local
```

Como se vé, este comando primero solicita confirmación de dónde van a a guardarse las claves ('.ssh/id_rsa'), y luego solicita, dos veces, una contraseña (passphrase), contraseña que puedes dejar en blanco si no deseas tener que teclearla cada vez que uses la clave.

Tras generarla, cada usuario ha de encargarse de enviar su clave pública a quienquiera que administre el servidor Git (en el caso de que este esté configurado con SSH y así lo requiera). Esto se puede realizar simplemente copiando los contenidos del archivo terminado en '.pub' y enviándoselos por correo electrónico. La clave pública será una serie de números, letras y signos, algo así como esto:

```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAklOUpkDHrfHY17SbrmTIpNLTGK9Tjom/BWDSU
GPl+nafzlHDTYW7hdI4yZ5ew18JH4JW9jbhUFrviQzM7xlELEVf4h9lFX5QVkbPppSwg0cda3
Pbv7kOdJ/MTyBlWXFCR+HAo3FXRitBqxiX1nKhXpHAZsMciLq8V6RjsNAQwdsdMFvSlVK/7XA
t3FaoJoAsncM1Q9x5+3V0Ww68/eIFmb1zuUFljQJKprrX88XypNDvjYNby6vw/Pb0rwert/En
mZ+AW4OZPnTPI89ZPmVMLuayrD2cE86Z/il8b+gw3r3+1nKatmIkjn2so1d01QraTlMqVSsbx
NrRFi9wrf+M7Q== schacon@agadorlaptop.local
```

ENJAULAMIENTO DE USUARIOS EN SSH

Para enjaular a los usuarios dentro de un directorio, es decir que no puedan ver el árbol de directorios superior debemos editar el archivo /etc/ssh/sshd_config.

Al final del archivo, debemos agregar las directivas para dicho propósito:

Match group usuariox

ChrootDirectory /var/www/directorioConAccesoUsuarioX

X11Forwarding no

AllowTcpForwarding no

ForceCommand internal-sftp

Luego reiniciamos el servidor ssh: /etc/init.d/ssh restart

Y entonces ahora, tendremos a determinado usuario, enjaulado dentro de el directorio definido. Lo más conveniente es que coincida el directorio home del usuario con el del enjaulado. Si queremos cambiar el home del usuario debemos modificar el archivo /etc/passwd con permisos de root.

conclusión

Obtener una comprensión en profundidad de la forma subyacente de funcionamiento de SSH puede ayudar a los usuarios a comprender los aspectos de seguridad de esta tecnología. La mayoría de la gente considera este proceso extremadamente complejo y poco comprensible, pero es mucho más simple de lo que la mayoría de la gente piensa. Si te estás preguntando cuánto tiempo toma una computadora para calcular un hash y autenticar a un usuario, bueno, sucede en menos de un segundo. De hecho, la cantidad máxima de tiempo se gasta en la transferencia de datos a través de Internet.

Bibliografía

- https://www.hostinger.es/tutoriales/que-es-ssh#gref
- https://www.ecured.cu/SSH