UNPSJB

LICENCIATURA EN SISTEMAS OPGCPI

ADMINISTRACIÓN DE REDES Y SEGURIDAD

Trabajo Práctico 4

SSH y túneles

Cátedra
Lic. Bruno Damián Zappellini

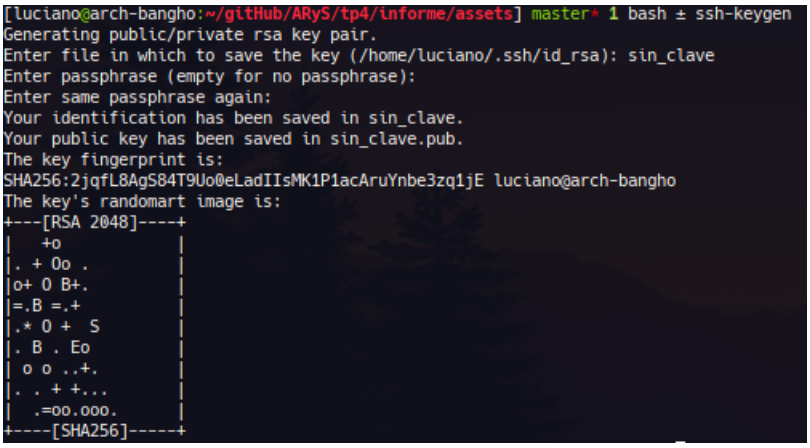
Integrantes:
Luciano Serruya Aloisi

4 de diciembre de 2018

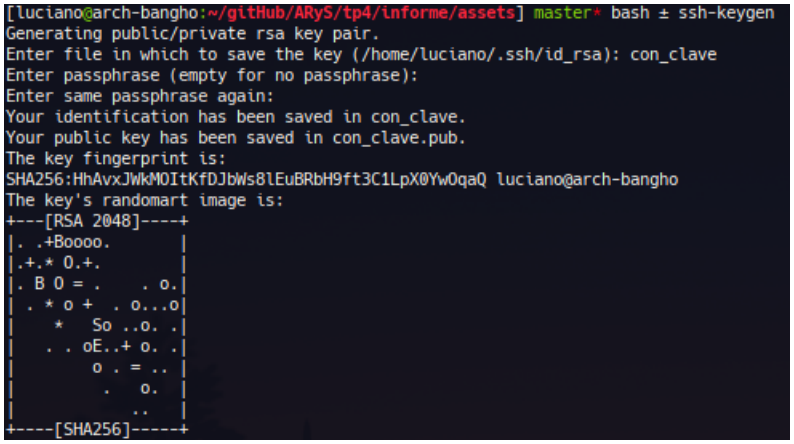
Índice

1. Sistema de autenticación de SSH

Según las páginas `man`, el comando `ssh-keygen` *genera, gestiona, convierte, y autoriza claves para ssh. ssh-keygen puede generar claves para que las use SSH protocolo versión 2*



Creación de una llave SSH sin contraseña



Creación de una llave SSH con contraseña “unaClave”

1.1. Archivos de autorización

Para gestionar el ingreso con *clave pública* y para llevar registro de los sitios confiables a los cuales ya se ingresó, SSH mantiene dos archivos, respectivamente:

- *authorized_keys*: lleva registro de las claves públicas aceptadas para conectarse vía SSH al sistema. Se podrán conectar con autenticación de clave pública sólo aquellos

usuarios que tengan la correspondiente clave privada de la clave pública registrada en el archivo

- *known_hosts*: indica a los sitios que se conectó el usuario y son seguros (antes de establecer la conexión SSH con un sitio nuevo, el sistema pregunta si desea confiar en el sitio)

1.2. Ingresando a un servidor remoto con clave pública

Luego de registrar la clave pública personal en el servidor remoto y de haber configurado este último para que sólo permita ingresar (por SSH) mediante clave pública, se puede ingresar al servidor sin problemas.

Gracias al *log* que imprime SSH al ejecutarlo con la bandera `-vvv`, se puede ver la siguiente sección con respecto al ingreso con clave pública

```
1  .
2  .
3  .
4  debug1: Authentications that can continue: publickey
5  debug3: start over, passed a different list publickey
6  debug3: preferred publickey,keyboard-interactive,password
7  debug3: authmethod_lookup publickey
8  debug3: remaining preferred: keyboard-interactive,
     password
9  debug3: authmethod_is_enabled publickey
10 debug1: Next authentication method: publickey
11 .
12 .
13 .
```

2. Túneles

2.1. Locales

Ana es desarrolladora de aplicaciones web, y quiere probar algunos cambios con una base de datos de prueba que se encuentra en un servidor remoto. Ana no puede transferir la

base de datos a su computadora, dado el excesivo tamaño de la misma. Sin embargo, Ana sabe que mediante la herramienta ssh puede hacer que su aplicación local utilice la base de datos remota

La creación de un túnel SSH sería de igual manera para trabajar tanto con una base de datos MySQL como con una PostgreSQL, lo único que cambiaría es el puerto al cual redirigir la petición en el servidor remoto.

Lo primero a tener en cuenta es que Ana debe poder ingresar al servidor remoto vía SSH, ya sea teniendo un usuario y contraseña en el servidor, o habiendo instalado su clave pública. Luego, debe saber qué puerto utiliza su aplicación para conectarse a la base de datos local.

Suponiendo que la aplicación web de Ana utiliza el puerto 6060 para conectarse a la base de datos local, debería invocar SSH de la siguiente manera para crear un túnel:

- MySQL: `ssh 6060:<IP_SERVIDOR>:3306 <USUARIO_ANA>@<IP_SERVIDOR>`
- PostgreSQL: `ssh 6060:<IP_SERVIDOR>:5432 <USUARIO_ANA>@<IP_SERVIDOR>`

2.2. Remotos

Matías tiene acceso a un servidor que escucha conexiones únicamente en el puerto 80, para las interfases de red físicas. Matías no puede alterar el firewall ya que no es administrador, sin embargo ha detectado que el servidor ejecuta un servicio SSH y no tiene restricciones para realizar conexiones salientes. Matías desea poder acceder al servidor SSH desde su computadora personal. Indique que con que argumentos debe invocar ssh desde el servidor (túnel reverso) para realizar la conexión desde su máquina personal

En este caso, se necesita crear un túnel de tal forma que se pueda conectar la computadora personal de Matías con el servidor. Creando un túnel reverso **desde el servidor hacia la computadora personal** de Matías, el servidor abre un puerto en la computadora personal, redirigiendo todo el tráfico de ese puerto hacia un puerto propio del servidor.

Para crear el túnel reverso, se debe invocar SSH de la siguiente manera:

- `ssh -R 2020:localhost:22 <USUARIO_MATIAS>@<IP_MATIAS>`

Nótese que el uso del puerto 2020 es arbitrario, se podría utilizar cualquier puerto que no se esté usando en la PC de Matías. Luego, Matías se podrá conectar al servidor haciendo:

- `ssh -p 2020 <USUARIO_SERVIDOR>@<IP_SERVIDOR>`

Marina desea acceder a un servidor que administra José, pero ambos se encuentran detrás de un router que realiza network address translation (NAT). Marina sabe que ambos pueden acceder a un tercer host. Indique qué comandos debe realizar Marina o José, para poder tener acceso ssh entre los servidores de los extremos

Al igual que el escenario anterior, esta situación se puede resolver con un **túnel reverso**.

Supóngase que Marina desea acceder a una aplicación web en el servidor de José, pero esta aplicación corre en el puerto 8080, y José no desea exponer los puertos de la máquina hacia afuera de la red. Teniendo José un acceso vía SSH al servidor público (el tercer host que se indica en el enunciado), puede crear el siguiente túnel reverso desde la máquina que corre la aplicación

- `ssh -R 9000:localhost:8080 <USUARIO_JOSE>@<IP_SERVIDOR>`

Ahora Marina podrá ingresar a la aplicación de José, navegando a la dirección `<IP_SERVIDOR>:9000`

3. Aplicaciones con interfaz gráfica

Ernesto sabe que en un servidor se está ejecutando el proceso X11. Desea ejecutar una aplicación gráfica remota (Por ej.: Firefox). Describa que uso dio al cliente de ssh Ernesto

La bandera `-X` de SSH permite compartir el *socket* de la instancia de X11 que se está corriendo en la máquina del cliente; de esta forma, se puede ejecutar una aplicación con interfaz gráfica en el servidor remoto (al cual se conectó mediante SSH), y ver la GUI en la máquina del cliente.

4. Archivos de configuración de conexiones

Se desea confeccionar un archivo de configuración para manejar la siguiente conexión:

- Se debe ejecutar con el comando `ssh servidor1`

- Dirección IP: 192.168.1.40
- Nombre de usuario: “administrador”
- Puerto de la conexión: 22022
- Autenticación por *passphrase* desactivada
- Utilizar la clave ubicada en *$HOME/.ssh/keys/servidor1*

```
1  Host servidor1
2      # IP a la cual conectarse
3      Host 192.168.1.40
4      # Usuario con el cual conectarse
5      User administrador
6      # Puerto donde escucha SSH
7      Port 22022
8      # Desactivar passphrase
9      BatchMode yes
10     # Utilizar key particular
11     IdentityFile ~/.ssh/keys/servidor1
```

5. Direccionamiento de puertos dinámico

Utilizando túneles locales o remotos, la intención es redirigir el tráfico de un puerto hacia un puerto de otra máquina. Si se desea redireccionar el tráfico de varias aplicaciones, se deben crear varios túneles. Esto resulta siendo tedioso y poco práctico.

Con el **direccionamiento de puertos dinámico** (***Dynamic port forwarding***), se inicia un servidor de proxy que implementa el protocolo **SOCKS** para redirigir el tráfico automáticamente según el protocolo de aplicación del que se trate.

Cabe aclarar lo siguiente, cuando se crea un túnel local o remoto, SSH no interpreta de ninguna manera el tráfico que fluye por el túnel; envía la información desde el puerto origen al puerto destino. Al crear un proxy SOCKS (con el parámetro de SSH `-D`), SSH actúa como un servidor proxy, manejando conexiones de varios puertos distintos (necesita interpretar el tráfico para determinar el puerto destino) [1]

5.1. Configuración de Firefox para usar un proxy SOCKS

1. Iniciar el servidor de proxy con la siguiente línea de SSH
   - `ssh -D <PUERTO> <USUARIO>@<HOST>`
2. En Firefox, ir a `Edit` → `Preferences`
3. Dirigirse a la sección de `Network Proxy` → `Setings...`
4. Tildar `Manual proxy configuration`
5. Borrar todos los campos texto, y en el campo de `SOCKS Host`, ingresar
   - `127.0.0.1`, Puerto `1080`

Al utilizar el navegador con el proxy SOCKS configurado, las peticiones que realiza el navegador no tendrán como IP de origen la máquina en la cual está corriendo el navegador, sino que su IP origen será la de la máquina que esté corriendo el servidor proxy.

La red TOR[1] utiliza servidores de proxy SOCKS para hacer la navegación por Internet anónima; cada petición pasa por tres servidores de proxy distintos antes de llegar a destino

5.2. `proxychains`

La herramienta `proxychains` fueza cualquier conexión TCP hecha por cualquier aplicación a que pase por un proxy como puede ser TOR o cualquier otro proxy SOCKS o HTTP [2].

A diferencia de TOR o configurar SSH como proxy SOCKS, `proxychains` no es un proxy en sí, sino que redirige las comunicaciones TCP a los proxys configurados. La principal característica que tiene es que se pueden definir varios servidores de proxy por los cuales debe pasar la comunicación antes de llegar a destino.

[1] Programa que protege y hace anónima la comunicación por Internet [3]

Referencias

[1] *Difference between "local port forwarding" and "dynamic port forwarding"?* Jul. de 2015. URL: https://unix.stackexchange.com/questions/213213/difference-between-local-port-forwarding-and-dynamic-port-forwarding?answertab=votes#tab-top.

[2] *ProxyChains HowTo.* URL: http://proxychains.sourceforge.net/howto.html.

[3] *What is Tor?* URL: https://www.torproject.org/docs/faq.html.en#WhatIsTor.