Linux para Ingeniería:

Cliente y Servidor SSH

Luis Garreta
luis.garreta@javerianacali.edu.co

Ingeniería de Sistemas y Computación
Pontificia Universidad Javeriana – Cali

4 de abril de 2017

Source: Linux System Administration by Paul Cobbaut
(http://linux-training.be/linuxsys.pdf)

ssh client and server

- The **secure shell** or **ssh** is a collection of tools using a secure protocol for communications with **remote Linux computers**.

Secure Shell

- Avoid using **telnet, rlogin** and **rsh** to remotely connect to your servers
- They **do not encrypt** the login session
- Your user id and password can be **sniffed** by tools like wireshark or tcpdump.
- To securely connect to your servers, **use ssh**.

SSH Security Schema

- The ssh protocol is **secure in two ways**:
  - Firstly the **connection is encrypted** and
  - secondly the connection is **authenticated both ways**.
- An ssh connection always starts with **a cryptographic handshake**, followed by **encryption of the transport layer** using a symmetric cypher.
- In other words, **the tunnel is encrypted** before you start typing anything.

SSH Authentication

- Then authentication takes place:
  - Using user id/password or public/private keys, and
  - communication can begin over the encrypted connection.
- The ssh protocol will remember the servers it connected to,
  - and warn you in case something suspicious happened
- The openssh package is maintained by the OpenBSD people
  - and is distributed with a lot of operating systems,
  - it may even be the most popular package in the world.

System Config File: /etc/ssh/

- Configuration of ssh client and server is done in the **/etc/ssh directory**

Public and private keys

ssh uses the well known system of **public and private keys:**

- Alice and Bob like to communicate with each other.
- Using public and private keys they can communicate with encryption and with authentication.
- When Alice wants to send an encrypted message to Bob:
  - she uses the public key of Bob.
- Bob shares his public key with Alice, but keeps his private key private!
- Since Bob is the only one to have Bob's private key, Alice is sure that Bob is the only one that can read the encrypted message.

rsa and dsa algorithms

- Both of these are encryption systems that are in common use when encrypting content.

Log on to a remote server

- The local user is named paul and he is logging on as user admin42 on the remote system.

```
paul@ubu1204:~$ ssh admin42@192.168.1.30
The authenticity of host '192.168.1.30 (192.168.1.30)' can't be established.
RSA key fingerprint is b5:fb:3c:53:50:b4:ab:81:f3:cd:2e:bb:ba:44:d3:75.
Are you sure you want to continue connecting (yes/no)?
```

- the user paul is presented with an rsa authentication fingerprint from the remote system.
- The user can accepts this bu typing yes
- An entry will be added to the ~/.ssh/known_hosts file
- The user can get log out of the remote server by typing exit or by using Ctrl-d.

Executing a command in remote

- Execute the pwd command on the remote server.
- There is no need to exit the server manually.

```
paul@ubu1204:~$ ssh admin42@192.168.1.30 pwd
admin42@192.168.1.30's password:
/home/admin42
paul@ubu1204:~$
```

Copy To or From Remote System: **csp**

- The scp command works just **like cp**,
- but allows the source and destination of the copy to be behind ssh.
- Here is an example where we copy the /etc/hosts file from the remote server to the home directory of user paul:

```
paul@ubu1204:~$ scp admin42@192.168.1.30:/etc/hosts /home/paul/serverhosts
admin42@192.168.1.30's password:
hosts                                         100%  809     0.8KB/s   00:00
```

- Here is an example of the reverse, copying a local file to a remote server:

```
paul@ubu1204:~$ scp ~/serverhosts admin42@192.168.1.30:/etc/hosts.new
admin42@192.168.1.30's password:
serverhosts                                   100%  809     0.8KB/s   00:00
```

Setting up passwordless ssh

- Use **ssh-keygen** to generate a key pair without a passphrase,
- and then copy your public key to the destination server.
- Let's do this step by step:

1. Execute ssh-keygen
2. Create ~/.ssh directory
3. Rename id_rsa and id_rsa.pub
4. Copy the public key to the other computer
5. Create authorized_keys
6. Execute passwordless ssh commands

Setting up passwordless ssh:
Execute ssh-keygen command

- Alice uses ssh-keygen to generate a key pair. Alice does not enter a passphrase.

```
[alice@RHEL5 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alice/.ssh/id_rsa):
Created directory '/home/alice/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alice/.ssh/id_rsa.
Your public key has been saved in /home/alice/.ssh/id_rsa.pub.
The key fingerprint is:
9b:ac:ac:56:c2:98:e5:d9:18:c4:2a:51:72:bb:45:eb alice@RHEL5
[alice@RHEL5 ~]$
```

- You can use ssh-keygen -t dsa in the same way.

Setting up passwordless ssh:
Create ~/.ssh directory

- ssh-keygen generates a public and a private key,
- it will also create a hidden .ssh directory
- If you create the .ssh directory manually, then you need to chmod 700 (a+wrx) it!
  - chmod ao-rwx
  - chmod u+awx

```
[alice@RHEL5 ~]$ ls -ld .ssh
drwx------ 2 alice alice 4096 May  1 07:38 .ssh
[alice@RHEL5 ~]$
```

Setting up passwordless ssh: **id_rsa and id_rsa.pub**

- The ssh-keygen command generate two keys in .ssh.
- The public key is named ~/.ssh/ id_rsa.pub
- The private key is named ~/.ssh/id_rsa.
- Rename the public key with your initials to avoid conflicts with the remote files

```
mv id_rsa.pub to aid_rsa.pub
```

```
[alice@RHEL5 ~]$ ls -l .ssh/
total 16
-rw------- 1 alice alice 1671 May  1 07:38   id_rsa
-rw-r--r-- 1 alice alice  393 May  1 07:38  aid_rsa.pub
```

Setting up passwordless ssh:
Copy the public key to the other computer

- use **scp** to copy public key to the remote server:

```
[alice@RHEL5 .ssh]$ scp aid_rsa.pub bob@192.168.48.92:~/.ssh/
bob@192.168.48.92's password:
id_rsa.pub                                    100%  393     0.4KB/s   00:00
```

- Append the public key to the authorized_keys:

```
cat aid_rsa >> ~/.ssh/authorized_keys
```

Setting up passwordless ssh:

Check authorized_keys

- In your ~/.ssh directory, you can create a file called authorized_keys
- This file can contain one or more public keys from people you trust.
- Those trusted people can use their private keys to prove their identity and gain access to your account via ssh (without password)
- The example shows Bob's authorized_keys file containing the public key of Alice:

```
bob@laika:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEApCQ9xzyLzJes1sR+hPyqW2vyzt1D4zTLqk\
MDWBR4mMFuUZD/O583I3Lg/Q+JIq0RSksNzaL/BNLDou1jMpBe2Dmf/u22u4Kmq1JBfDhe\
yTmGSBzeNYCYRSMq78CT9l9a+y6x/shucwhaILsy8A2XfJ9VCggkVtu7X1WFDL2cum08/0\
mRFwVrfc/uPsAn5XkkTscl4g21mQbnp9wJC40pGSJXXMuFOk8MgCb5ieSnpKFniAKM+tEo\
/vjDGSi3F/bxu691jscrU0VUdIoOSo98HUfEf7jKBRikxGAC7I4HLa+/zX73OIvRFAb2hv\
tUhn6RHrBtUJUjbSGiYeFTLDfcTQ== alice@RHEL5
```

Setting up passwordless ssh:
Connect and Execute a Remote Command

- Alice can now use ssh to connect passwordless to Bob's laptop.
- In combination with ssh's capability to execute commands on the remote host
- This can be useful when need to execute something in a more powerful machine.

```
[alice@RHEL5 ~]$ ssh bob@192.168.48.92 "ls -l .ssh"
total 4
-rw-r--r-- 1 bob bob 393 2008-05-14 17:03 authorized_keys
[alice@RHEL5 ~]$
```

X forwarding via ssh

- Another popular feature of ssh is called X11 forwarding
- It is used to execute interactive apps (windowing apps)
- It is implemented with the option -X
  - ```
    sh -X user@server
    ```
- For example, log in to the remote server and execute a graphical app (e.g dillo).
    ```

Troubleshooting ssh

- Use **ssh -v** to get debug information about the ssh connection attempt.

```
sim1
<dell>[~]$ ssh -YC -v -v -rcfour lg@eisc.univalle.edu.co -p 2224
OpenSSH_7.2p2 Ubuntu-4ubuntu2.1, OpenSSL 1.0.2g  1 Mar 2016
debug1: Reading configuration data /home/lg/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Connecting to eisc.univalle.edu.co [138.122.201.90] port 2224.
debug1: Connection established.
debug1: identity file /home/lg/.ssh/id_rsa type 1
debug1: key_load_public: No such file or directory
debug1: identity file /home/lg/.ssh/id_rsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/lg/.ssh/id_dsa type -1
debug1: identity file /home/lg/.ssh/id_ecdsa type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/lg/.ssh/id_ecdsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/lg/.ssh/id_ed25519 type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/lg/.ssh/id_ed25519-cert type -1
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.1
debug1: Remote protocol version 2.0, remote software version OpenSSH_6.0p1 D
debug1: match: OpenSSH_6.0p1 Debian-3ubuntu1.2 pat OpenSSH* compat 0x0400000
```

Practice: ssh

- Asegurese que usted tiene acceso a dos computadoras Linux, o trabaje con un compañero.
- Llamaremos cliente al equipo suyo y servidor al equipo donde se va a conectar remotamente.
- Asegurese que tenga una cuenta en el servidor donde tenga acceso (password, leer, escribir, ejecutar).
    - Si no la tiene cree una nueva
- Use **ssh** para loguearse al servidor, muestre su directorio actual y entonces salgase (exit) del servidor.
- Use **csp** para copiar un archivo desde el servidor a su computador.
- Use **ssh-keygen** para crear un par de claves sin *pasphrase.*
- Configure *passwordless ssh* entre su cuenta en el cliente y en servidor.
- Verifique que los permisos en el servidor de los archivos de claves son correctos: lectura para todos en la llave pública, y solo acceso de root para la llave privada.
- Logueese al servidor con X forwarding via ssh y ejecute remotamente una aplicación interactiva.
- Construya un programa en python que calcule el factorial y construya un script que:
    - Cree un directorio remoto en el equipo llamado pruebas:
    - Copie el programa de python al equipo remoto directorio pruebas:
    - Lo ejecute en el equipo remoto y el resultado lo envíe (pipe) a un archivo
    - Copie el archivo del equipo remoto al equipo cliente.