TP 02 : Linux : Scripts and remote control

Introduction: In this TP we will begin by learning how to use SSH, and continue with more bash functionalities.

1 Secure Shell

In this part we'll see how to remotely connect to a computer by using SSH(Secure Shell). The service was created as a secure replacement for the unencrypted Telnet and uses cryptographic techniques to ensure that all communication to and from the remote server happens in an encrypted manner. SSH uses encryption to ensure secure transfer of information between the host and the client. Host refers to the remote server you are trying to access, while the client is the computer you are using to access the host. There are three different encryption technologies used by SSH :

Symmetrical encryption, is a form of encryption where a secret key is used for both encryption and decryption of a message by both the client and the host

Asymmetrical encryption, unlike symmetrical encryption, uses two separate keys for encryption and decryption. These two keys are known as the public key and the private key. Together, both these keys form a public-private key pair.

Hashing, a function with which it is easy to generate a unique value of a fixed length for each input, but impossible(very hard) to generate the input from the hash.

In broad strokes (very broad), when connecting to an ssh server, the server first uses Diffie-Hellman key exchange algorithm to create a common secret key in order to have symmetrical encryption. When symmetrical encryption it obtained the server authenticates the user by requiring the username and password.

The SSH configuration files are located in the directory "/etc/ssh/". The client's customization files are in the user's homedir, in the ".ssh/" directory, i.e. "~/.ssh/"

1.1 Connecting

The syntax for using the "ssh" client is as follows :

```
ssh [ options ] [ login@hostname ]  [ commande ]
```

If the file "~/.ssh/known_hosts" exists, rename it :

```
mv ~/.ssh/known_hosts{,.bak}
```

Explain the proposed syntax.
Connect to the SSH server of the department("ssh.dptinfo.ens-cachan.fr"). What is happening ? Explain.

1.2 Config file

Remembering all your hostname/usernames/any other options for a specific host can turn very cumbersome really fast. Fortunately, ssh has a way to store these options, in a file called "config". The default location for this file is at " /.ssh/config". If it does not exists for you, now it's the time to create it. The file accepts entries of the following format :

```
Host "hostName/aliases"
  SSH_OPTION value
  SSH_OPTION value
```

Where "hostName/aliases" stands for the aliases you want to give to this specific host, and "SSH_OPTION"s are the all kind of options associated to this specific host. Example of some of the "SSH_OPTION"s are :

- Hostname : the address of the computer you are trying to connect. Note that if the Hostname doesn't exists shh will try to connect to the "hostName/aliases" written in the title of this entry(i.e. the first aliases)
- User : the user name you want to use.
- Port : to which port you are trying to connect.

For an example the following :

```
Host ssh.pentagon.com secretStuff endGame
  Hostname ssh.pentagon.com
  User Julian_Assange
  Port 666
```

is an entry for the host "ssh.pentagon.com" with the aliases "ssh.pentagon.com", "secretStuff" and "endGame", which uses the user name "Julian_Assange" and connects to the port 666. Now, if we pass the command "ssh endGame", ssh automatically knows to connect to the correct host, with the correct username, and to the correct port.

Exercise : Create a entry for the SSH server of the department, and give it couple of aliases.

1.3 SSH Key Pairs

SSH has the ability to use keypairs to authenticate your session with the server, and hence removes the need of using your password.

In order to achieve this we first need to create a privet and public keys, one of the possible ways to do that is running the ssh-keygen command :

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

After running this command you'll be promoted to give a file in which to save the key, where the defualt is "~/.ssh/id_rsa", unless you have a reason don't change it. Next, it will ask you "Enter passphrase (empty for no passphrase)" if you don't leave it empty it will ask you for your password every-time you try to use the privet key generated.

When you are done go to the folder in which you created your keys(by defualt ""~/.ssh"). There you'll find two files "id_rsa" and "id_rsa.pub". The first file is your privet key, keep it safe and don't shear it. The second one is the public key, there is no reason to keep it privet, in fact this is the one who is going to be sheared.

Now when we have created a pair of keys, we are going to upload them to the desired ssh server. First make sure that the desired server has the directory " /.ssh". If it doesn't, create one. And finally we add our public key to the list of authorized keys :

```
cat <<pubKey_location>>| ssh <<host_address>> "cat >> ~/.ssh/authorized_keys"
```

Note that, for this command to work you need to be in your computer. Finally, re-login to the host, and enjoy your secured connection :)

Exercise : Create a pair of keys and upload them to the ssh server.

2 Aliases and .bashrc

A bash alias is nothing but the "shortcut" for commands. The alias command allows the user to launch any command or group of commands (including options and filenames) by entering a single word.
To create an alias we use the following command :

```
alias NAME=VALUE
```

Where NAME stands for the name of the shortcut and VALUE for the command that we want the shortcut to preform, e.g. :

```
alias ..='cd ..'
```

Note : You can rewrite commands by using aliases, e.g. creating the alias "ls=’ls -a’", overwrite "ls" to always show you all the files.

When you make an alias directly in the shell, it won’t "stick", next time you reopen your terminal(or reset you bash by runing the command "bash") it will revert to the default configuration. To make changes to settings of your bash you can change the file : "~/.bashrc". This file runs every-time you reset you shell.

Exercise : If "~/.bashrc" doesn’t exists create it. Edit your bash configuration file in such a way that the command "mkdir" will :

— create automatically all folder hierarchy without needing any extra flags
— output all the new folders it created(be ’verbose’)

3 Scripts

A script is a set of instructions written in a file that will be executed sequentially. For example :

```
cat > myscript
cd /tmp
ls
cd
^d
```

You can run the script by running the command :

```
bash myscript
```

In this example we specify that the shell to be used ("bash") and it passes in the arguments the file containing the instructions. But we could also specify directly in the script, what command interpreter to use by specifying it in the header of the file, as such :

```
cat > myscript
#!/usr/bin/bash

cd /tmp
ls
cd
^d
```

This first line is called **"shebang"**, and it specifies the location of the command interpreter(in our case bash). Generally if you dont know where the some command(program) is located, you can use the command "whereis" to get its location :

```
whereis bash
```

the script can then be invoked like any other program.

```
./myscript
```

It does not work, why ? Correct it.

3.1 Control structure

"man test" gives the list of tests available in bash.

3.1.1 if .. then ... else

Syntax :

```
if [ commande(s) ]; then
commande(s)
elif [ commande(s) ]; then
commande(s)
else
  commande(s)
fi
```

Exercise : Use "if" to test the existence of the directory "/var/logs", if it doesn't exists check whether the directory "/var/lOg" exists. Display a message in case both directories are not found or if one of them is found.

Note : In the case of long command lines, you can use "\" to go down a line and continue your command.

Exercise : Write the message "Alert" if you have the right to write in "/etc".

3.1.2 for ... do ... done

Syntax :

```
for var in list ; do
commande(s)
done
```

Write a "for" which displays the letters "a b c d", one per line.
Note : In bash you can create a range of numbers/letters by typing {1..10}.

3.1.3 while ... do ... done

Syntax :

```
while commande(s) ; do
commande(s)
done
```

What does the following code do ?

```
ls | while read a ; do echo $a ; done
```

3.2 Variable manipulation

Command substitution allows the output of a command to replace the command itself. Command substitution occurs when a command is enclosed as follows $(command), for example :

```
a=$(ls /)
```

will put the content of the root directory in to the variable a.

Arithmetic expansion allows the evaluation of an arithmetic expression and the substitution of the result. The format for arithmetic expansion is $(("expression")). For example :

```
x=$( ls /etc | wc -l)
y=$(ls /tmp | wc -l)
a=$(($x + $y))
echo $a
```

What does this script do ?

3.3 Trash script

Just like normal commands out costume scripts can receive input. To go over all the input of a command we can use "$@", e.g. the following script :

```
#!/bin/bash

for i in $@ ; do
echo "$i"
done
```

will output :

```
./script bla bla bla
bla
bla
bla
```

Note : To pick a specific input you can use "$i", where i stands for its position in the input.

Exercise : Write a small script that takes a set of files as a parameter and moves them to the trash (the ".trash" directory in your privet directory). Display an error message if a file does not exist.
The script should be no longer then 15 lines.

3.4 Games

Exercise : Alice Bob and Charlie, played a series of games. For every game they kept the log of these games in a "log file", where every line has the date, the name of one of the players, and how many point did they got.
Write a script that inputs the log file and outputs who is the winner and how many points did they get.
Useful commands : cut, grep, uniq, sort

3.5 Doodle

Exercise : You are a teacher in a unnamed course. You've been told that you need to split your first session into 2 sessions since there is a just not enough space for all the students in the lab. You publish a doodle to see which students goes to which session and get the following list "doodle answers". But you notice that some of the students didn't answer(since you have the full list of names "list of students"). But you still need to send an email telling each student to which session they should go.
Write a script that outputs two lists of names, one for the first session and the other one for the second session. Where each student that didn't answer needs to be in the second list.
Useful commands : grep, sed, uniq, sort