SSH

HW 8 - CNS Sapienza

Daniele Oriana 1457625

15 Dec 2016

Description of SSH with reference to both sides authentication

Answer

SSH (Secure SHell) is a protocol that allows to enstabilish an encrypted remote connection with another host through a command line interface. SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server. SSH was created with the purpose to replace Telnet and other unsecured remote shell protocols. For the duration of your SSH session, any commands that you type into your local terminal are sent through an encrypted SSH tunnel and executed on your server. On the server we have a piece of software called an SSH daemon that listens for connections on a specific network port, in addiction it also authenticates connection requests and predispose the correct environment after a successfull login done by the user. The user's machine, of course, must have an SSH client. This is a piece of software that knows how to communicate using the SSH protocol and has the goal to start and manage the connection in the client side.
SSH provides authentication for both sides of the protocol (client and server), let's look how it takes place:

- Authentication client side:
  Client uses an authentication based on a passphrase that the user can insert in order to connect with the server, but automated bots and malicious users will often repeatedly try to authenticate finding this passphrase; for this reason SSH provides also an authentication based on a set of cryptographic keys (SSH keys). In this authentication each

set contains a public and a private key. The public key can be shared freely, while the private key must be kept secret. To authenticate using SSH keys, a user must have an SSH key pair on their local computer. On the remote server, the public key must be copied to a file within the user's home directory at /.ssh/authorized_keys. This file contains a list of public keys, one-per-line, that are authorized to log into this account. When a client wants to enstabilish a new connection it asks for it to the server that checks if has the public key of the client it in the authorized keys file, if there is this public key, the server generates a random string encrypting it with the public key of the client and sends the result to the client. At this point the client uses its private key in order to decrypt and, after that, it combines the random string with a session ID previously negotiated. It then generates an MD5 hash of this value and transmits it back to the server. The server already had the original message and the session ID, so it can compare an MD5 hash generated by those values and determine that the client must have the private key. Now the connection can take place.

- Authentication server side:
  Server authentication is performed using the Diffie-Hellman key exchange with the addiction of the digital certificates. This kind of authentication follows these steps:
  - The server sends its certificate (which includes also its public key) and a random data associated with the session signed by the server's private key to the client.
  - Since the server certificate is signed with the private key of a certification authority (CA), the client can check it using the CA certificate.
  - After that the client verifies that the server has a valid private key checking the signature in the initial packet through the public key sent by the server in the first step.

  So the authentication, at this point, is completed. We can notice that in this key exchange the man-in-middle attack is not a problem, because the system checks that the server certificate has been issued by a trusted CA.

Running examples of two types authentication of SSH clients and report what you did

Answer (first example)

In this first example i will show you how to implement a local SSH server-client pair, providing an authentication based on a public/private key pair and on a passphrase on linux.
The procedure is the following one:

- First of all we need to install openssh-server and we can do this with the command:

  ```
  sudo apt-get install openssh-server
  ```

- After that we need some directories and files with some permissions and, once we have added them, we need to restart the service, i have used the following commands:

  - mkdir .ssh
  - chmod 700 .ssh
  - touch .ssh/authorized_keys
  - chmod 600 .ssh/authorized_keys
  - sudo service sshd restart

  restarting the service allows the server to run.

- Now, we have to setup the client, first of all we need to create a public-private key, so open a new terminal (the client one) and type:

  ```
  ssh-keygen -t dsa
  ```

  follow the subsequent instructions in order to save the DSA keys pair and in order to choose a passphrase for the future login in the secure shell. Once we have done this we obtain the situation like the one descibed in the Picture 1.

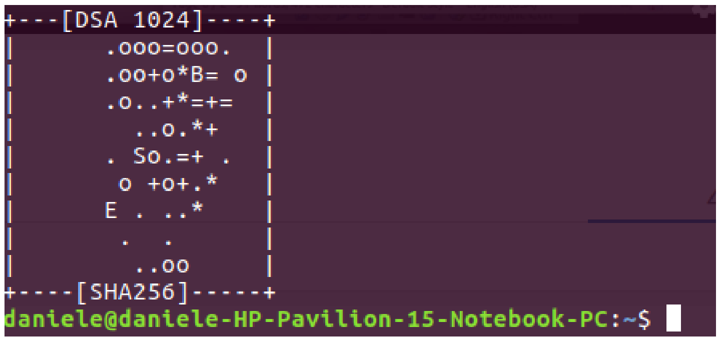
```
+---[DSA 1024]----+
|       .ooo=ooo.  |
|       .oo+o*B= o |
|       .o..+*=+=  |
|         ..o.*+   |
|       . So.=+ .  |
|        o +o+.*   |
|       E . ..*    |
|        .  .      |
|         ..oo     |
+----[SHA256]-----+
daniele@daniele-HP-Pavilion-15-Notebook-PC:~$
```

Figure 1: Picture 1.

As a conseguence of that we have two files located in /.ssh directory, called id_dsa and id_dsa.pub; these are the client's public and private key, respectively.

- The next step is to add the client's public key to the SSH server's authorized_keys file, but in order to do that, we need to know the server's IP addres, we can achieve this IP using the famous ifconfig command. Here we will call this address with X.
- Once we know the server's IP address, we can add into it the public key of the client typing:
  ssh-copy-id [your user name]@X
- We are ready now to connect with the server, and we do this typing:
  ssh [your user name]@X
  in order to connect with the server we need also to type the passphrase after the previous instruction.

We have to note that when we are connecting to the server through SSH we have to enter the passphrase of the generated keys and not the server's user password. What makes the connection authenticated is the existence of the client's public key on the remote server (authorized_keys file) and the client's private key in the local machine. The passphrase is there to ensure that someone who steals your private key will still no able to log into the SSH server unless she/he knows also the the passphrase.

Answer (second example)

In this example i consider a (local) SSH based on the so called Host-Based-Authentication that is used to restrict client access only to certain host. This method is similar to public key authentication, however the server maintains a list of hosts and their public keys (so using the public key on other host won't authenticate the client). In order to implement this modality the procedure is like the one described in the previous example, but we have to consider some other things, that are the following:

- On the client, enable host-based authentication, you can do it in the configuration file, /etc/ssh/ssh_config, typing the following entry:
  HostbasedAuthentication yes
- Do the same (like previous point) in the server configuration file, /etc/ssh/sshd_config, type:
  HostbasedAuthentication yes
- On the server, we have to configure a file that enables the client to be recognized as a trusted host, in order to do that we add the client as an entry to the server's /etc/ssh/shosts.equiv file
- Then on the server we have to allow the sshd daemon to access the list of trusted hosts, to do that we modify the /etc/ssh/sshd_config file adding:
  IgnoreRhosts no

  After these steps you can enstabilish a connection between server and client following the steps described in the previous example.