SECURE SHELL (SSH)

We are going to study a network protocol that provides secure *shell* features, this protocol is SSH. We are not going into details. We just mention some algorithms. This is a *protocol* running on **TCP**. Since SSH is a protocol, it is a set of smaller protocols, each of which is providing small features like compression, encrypt algorithms and so on. The channel under which SSH is used can be retained secure, and authentication is required. The practical use of SSH is for many things, like login to a shell on a remote host (i.e. Telnet), execute a single command on a remote host, securely transfer files and so on. The main usage of SSH is to implement a remote terminal, i.e. we get a terminal with which we can control a remote machine.

There is an open software implementing it, and it is called OpenSSH. It is an open source project that has become common.

Architecture

We tipically have a client and a Server, and client makes a secure connection to Server. There is some handshaking at the beginning, when the client starts a connection with the Server the client stores locally the public key of the Server, then after that every time the client connects it already has the public key of the Server. (07:00 quote)

So everytime we connect to the Server we check whether the public key of the Server is the same we stored the first time.

The tipical case is SSH over TCP. SSH is based on 3 main components:

- SSH-AUTH: Authentication protocol
- SSH-CONN: Connection protocol
- SSH-SFTP: File transfer protocol

Unterlying them there is SSH-TRANS, the SSH Transport protocol.

Let's see them into details:

SSH-TRANS: It handles the initial connection and ensures Server authentication. After establishing the initial connection, the client can communicate in a secure way. It handles initial key exchange, server authentication and sets up encryption, compression and integrity verification. Transport layer also arranges for key re-exchange.

SSH-AUTH: It is used to authenticate client, actually it is weak because it is based on username/password credentials. The most secure way to authenticate the client is by means of a certificate. After it, the authentication of the client is completed and the SSH-CONN can take place.

SSH-CONN: Connection protocol, it starts to play after Authentication protocol. It is implementing the concept of *channel*. Once the inizial part and the key negotiations have taken place, Client and Server can start performing secure connections.

SSH-SFTP: It can rely on SSH-CONN, in the sanse that it happens upon a connection and is used to transfer files in a safe and secure way.

SSH supports a lot of encrypting algorithms and key exchange algorithms.

SSH Port Forwarding

Recall that a TCP port number identifies an application that is using internet. Port forwarding via SSH (**SSH tunneling**) creates a secure connection between a local computer and a remote machine through which services can be relayed. There are different types of SSH port forwarding:

- Local Port Forwarding: connections from SSH client are forwarded via the SSH server, then to a destination server (usually in a private network). It is the most common type.
- Remote Port Forwarding: connections from the SSH server are forwarded via the SSH client, then to a destination server.
- **Dynamic port forwarding**: connections from various programs are forwarded via the SSH client, then via the SSH server, and finally to several destination servers.

In few words it happens the following: we have a secret Server maybe in a private network and we want to access it from a remote position. How to connect to a private IP number? It is not possible from outside. So I connect to another host that is able to connect to that private host.

Imagine we have our laptop which wants to connect to our remote Server in our office, which has the address 10.22.124.15, and such that it is required an SSH connection. We need an SSH Server in the network of our office that takes the SSH traffic and forwards it to the Server in our office, which is not reacheable from outside. Let's say the SSH server has a certain address *office@sshserver.it*, then with the following command:

```
ssh -L 8080:10.22.124.15:32674 office@sshserver.it
```

we transfer all the data going through port 8080 of our laptot (i.e. http://localhost:8080) to the SSH server office@sshserver.it, which will forward then the data to the Server in our office to the port 32674 (i.e. 10.22.124.15:32674).

Example by D'Amore:

```
ssh -L 54321:10.0.2.92:22 damore@linuxserv.dis.uniroma1.it
```

makes it available at port 54321 of localhost (i.e. http://localhost:54321) the port 22 of the (unreachable) host 10.0.2.92, providing a tunnel between (localhost, 54321) and (10.0.2.92, 22).