SSH-tunneling: What Is SSH and SSH-tunneling?

The Acronym **SSH** stands for **Secure Shell Host**. SSH was originally created to provide a secure way to access server systems at "low level", to be used instead of common (but insecure) **telnet** methods. SSH can use several different forms of encryption, anywhere from 56 to 1024 bit. SSH has been ported to Operating Systems on several platforms including Linux, Microsoft Windows and Macintosh. There are SSH servers and SSH clients available for different types of communication. You can find more information about SSH at https://en.wikipedia.org/wiki/OpenSSH/. Here you may notice this: "OpenSSH includes the ability to forward remote TCP ports over a secure tunnel, allowing that way arbitrary TCP ports on the server side and on the client side to be connected through an SSH tunnel". This is excatly what we make use of.

The term **"SSH tunneling"** in relation to a database server means that in- and outgoing communications to the network that hosts the database server "passes through" the SSH-server and uses the communications port (usually port 22) and the protocol of the SSH-server. The SSH-server then "translates" and "transfers" that in-and outgoing communication to the database server. SSH is actually quite simple to use. However if you are totally unfamiliar with networking terminology you will have to study it somewhat. Actually you may even install an SSH server at your own local machine and use it for connecting to a local MySQL server. Not much use of that, but it will give you an excellent understanding of what SSH is!

There is a built-in SSH-client in SQLyog that lets you connect to a MySQL server using SSH.

Basically there are two benefits of SSH Tunneling:
* SSH can be used to encrypt communications between SQLyog and your remote MySQL server.
* It lets you access the MySQL server even if the MySQL port (3306) is blocked.

Unlike HTTP-tunneling you can't take it for granted that SSH-tunneling is available at your webhost. In general the "more professional" and "more expensive" hosting providers offer SSH and the cheaper ones don't.

Refer to the SQLyog help file for instructions how to set up the SQLyog Connections Manager, if you want to use SSH-tunneling with SQLyog. Each SSH-connection occupies a TCP-port at your local machine. With recent SQLyog versions this port is picked automatically from the pool of high-numbered ports not already in use.

You can read about SSH-related error messages here.

A concluding note on the popular 'Putty' program and SQLyog SSH-tunneling.

Sometimes when people are having problems with SSH-connections, we often hear "I can connect with Putty without problems". Maybe so, but it does not tell very much (almost nothing actually!) because the type of connection with Putty or a similar program referred to here by users **is not tunneling** and does not make use of port forwarding. Putty creates a remote (and secure) shell on the client machine, and connects to the 'mysql' client program on the server. So

here the MySQL client is the 'mysql' client on the remote server. It is true that Putty can be used for setting up a SSH-tunnel as well, but this is not the simple 'connect with Putty' most often referred to and compared with here. With (SQLyog) SSH-tunnel the MySQL client is the client API that is compiled into SQLyog (and SJA). That is why port forwarding is needed and must be functional with SQLyog SSH-tunneling!

Unique solution ID: #1032
Author: Peter Laursen
Last update: 2006-06-22 15:20