SSH-Tunneling hints

Motivation

Sometimes it's very useful to directly connect tools (i.e. a database management tool) that runs on a developer or administrator PC, to a daemon on a server (i.e. mysql server) that is only available locally with the server for security reasons.

Example introduction

For the following tunnel examples we assume:

- A server, `www.clazzes.org`, runs a mysql server listening on `127.0.0.1:3306`
- It provides a database `testdb`, accessible for user `dbtester` with password `testsecret`.
- The server provides a unix account, `webadmin`, and we have a ssh key allowed to connect to `webadmin@www.clazzes.org` (or we have password based ssh access)
- We want to access to the database `testdb` through an ssh tunnel
- When logged in to the server, the local mysql client can connect to the database with: `mysql -h 127.0.0.1 -u dbtester --password=testsecret testdb`
- We want to create a tunnel so when connection to port `3333` on our local system, we actually connect to mysql-server's port `3306` on `www.clazzes.org`

DO NOT bother to try using those credentials, it might get your IP blocked!

OpenSSH

OpenSSH is the default ssh implementation for most Linux distros, and even Microsoft has announced an agreement to include it in Windows.

I'm not sure how equal or similar other Unix ssh clients (like BSD, MacOS) are.

To create a tunnel that stays in foreground:

```
# stay in forground, may be put in background by pressing Ctrl-Z and the command bg
ssh root@clazzes.org -L 3333:127.0.0.1:3306 -N

# go in background, a bit difficult to stop
ssh -f root@clazzes.org -L 3333:127.0.0.1:3306 -N

# evtl. check that ssh listens on 3333
lsof -i -n |grep -i listen |grep 3333
```

Test connect to the database:

```
mysql -h 127.0.0.1 -P 3333 -u dbtester --password=testsecret testdb
```

Voila!

To close the tunnel, abort or kill the according `ssh` process (`ctrl-c`, evtl. after `fg` to get it back to the foreground).

Additional Notes:

Forwards (`-L` resp. `Forward`) and reverse forwards (`-R` resp. `RemoteForward`) can only be applied to sockets, like `~/.gnupg/S.gpg-agent` if the socket to listen on is not yet owned by some daemon.

Hopping can be done automatically using `ProxyJump` option.

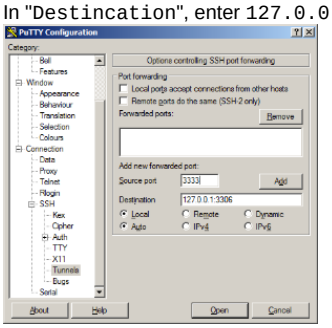
Putty

Putty is the most common ssh client for Windows. Hints for setting up key-based ssh access with Putty can be found everywhere on the internet, we'll focus on tunneling here.
Just one hint anyway: With ssh keys, have `pageant` running. Simply double-klick the `.ppk` file or even put it in your autostart group.

To setup the tunnel:

- Start Putty
- If you don't have a session yet that allows you to connect to `www.clazzes.org` as `webadmin`, set one up and don't forget to save the session before (re)trying to connect.
- In Session, load the session that allows you to connect to `www.clazzes.org` as `webadmin`.
- In "`Category`", open `Connection`, `SSH`, `Tunnels`

- In "`Source port`", enter `3333`
- In "`Destincation`", enter `127.0.0.1:3306` as shown in this screenshot (click to enlarge):



- If you really really want to DANGEROUSLY provide the tunnel for other colleages in you LAN, check "Local ports accept connections from other hosts".
- Click "`Add`"
- In "`Category`", click on "`Session`", "`Save`", then "`Open`"

Now a putty window should open, you should be on `www.clazzes.org` as `webadmin`., and the tunnel should be up.

To check if there's a tunnel, open a Command window and use netstat like this:

```
netstat -a -n |find "3333"

# output should show something like
TCP 0.0.0.0:3333 0.0.0.0:0 LISTENING
```

Voila!

To close the tunnel, just close the according putty terminal, preferrably by entering `exit`.