SSH

- Creating SSH-key
- SSH Tunnel
- Using `dd` and `ssh` to copy a disk image over a network

Creating SSH-key

- To generate an SSH-key, enter the following command on the "home" terminal:

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

-t stands for type and this determines the type of key 2. -b stands for bits. This can be used to determine the length of the key.

Saving the SSH-key

```
Enter file in which to save the key (/home/me/.ssh/id_rsa):
```

Here you can select a different location and an alternative name for the file containing the private key. Just press "Enter" to accept the given suggestion.

```
Enter passphrase (empty for no passphrase):
```

Optionally, a password for the public key can be assigned here. This is always queried when the public key file is used to establish a connection.

```
Enter same passphrase again:
```

Enter the same password again. If the field is empty, simply press "Enter"

Copying the SSH-key on your server

`ssh-copy-id youruser@ip-address`

Copy the public key to the desired server. For this the password of the server is necessary. NOTE: this will only work if the public key lays on the default location

Login without password-authentication

Now, if all of the steps are done right you´ll be able to login over ssh without your password. Simply connect over ssh (if you choose a password in the key, use the keys password

```
ssh youruser@ip-address
```

SSH Tunnel

tunnel with ssh (local port 3337 -> remote host's 127.0.0.1 on port 6379)

```
ssh -L 3337:127.0.0.1:6379 root@domain.tld -N
```

SSH Tunnel - Advanced

SSH-Tunnel Syntax:

```
ssh -L [bind_address:]port:host:port user@server
ssh -R [bind_address:]port:host:port user@server
```

the option -L creates a local, and the Option -R a remote Port Forwarding. The encrypted tunnel is created always between Client and Server. The connection from "tunnel end" to host happens unencrypted, this is why you set it in most cases to localhost. Therefore localhost should not be confused with the local Computer. You have to see this localhost from server perspective, so the Server itself.

Die Option -L bzw. -R sets the direction. if you choose -L the direction is from your own Computer to the remote one, if you choose -R in the opposite direction. (you can think of it as normaL backwaRds.)

The first Port Argument is the entryport in the connection. You have to keep in mind, that the opening of a "privileged" port, so under 1024, only is allowed by root, so you should choose a higher one.

With the optional parameter bind_address you can seon which specific network address the connection should use, whereas localhost is default. A * or an empty bind_address-argument before the colon means, that the forwarding is on all Interfaces / Network Adresses. Probably whis will only work with IPv4 Adresses because the IPv6-Adresses aren't capable of beeing forwarded, Therefore you should use the Argument -4 .

The second port-parameter tells which Port tells, which port from host the tunneling should go on

Another useful argument is the option -N, which refuses a terminal-session, if you only want to use the Portforwarding to the remote systeme.

Examples

Forwarding fro Port 8000 on the local system to the Webserver (port 80) on Server:

```
ssh -L 8000:localhost:80 server -N &
netstat -anp --inet | egrep '(^Proto|8000)'

Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program
name
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN     10843/ssh

fg

ssh -L 8000:localhost:80 server -N
[Strg-C]
Killed by signal 2.
```

Same, but it isn't just a connection from local Host forwarded, but from all Interfaces (hint: you need to set the option - GatewayPorts ; use this option with caution!):

```
ssh -L *:8000:localhost:80 server -N -4 &
netstat -anp --inet | egrep '(^Proto|8000)'

Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program
name
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN     10906/ssh
```

Reverse direction. You allow Users on the Server, via localhost:3306 to connect to the clients MySQL-Server:

```
ssh -R 3306:localhost:3306 server

Last login: Sat Mar 11 23:24:20 2006 from 192.168.4.56
netstat -an --inet | egrep '(^Proto|3306)'
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
exit
```

```
logout
Connection to server closed.
```

Here you can see an example of a double SSH Reverse tunnel:

```
needsupportpc$ ssh -R 22:localhost:2222 user@vps
helpdeskpc$ ssh user@vps -t ssh needsupportpcuser@localhost:2222
```

Using `dd` and `ssh` to copy a disk image over a network

```
dd if=/dev/sdX | ssh user@remotehost "dd of=ops-tools.img"
```

Another Way

On the Sourcemachine

```
  tar zcf - /* | nc $IP 4444
```

And on the Destination Machine

```
  nc -lp 4444 > backup.tar.gz
```