
SSH Tunneling (Port forwarding)

By rac

Published: 05.01.2008 - 04:00

local forward

```
ssh -L port:target:targetport host
```

Dieser Befehl baut eine Verbindung zu host auf und leitet alle Verbindungen zu localhost:port an target:targetport, über die ssh-Verbindung, weiter.

remote forward

```
ssh -R port:target:targetport host
```

Dieser Befehl baut eine Verbindung zu host auf und leitet alle Verbindungen zu host:port an target:targetport, über die ssh-Verbindung, weiter.

X-Forwarding

Mittels X-Forwarding ist es möglich, ein oder mehrere Programme auf einem anderen Rechner zu starten aber auf dem eigenen anzeigen zu lassen.

```
ssh [user]@[host] -X <Programm>
```

Stellt das Programm am lokalen Rechner dar, die auf dem Host aber ausgeführt werden.

Beispiel IRC

```
xterm -e ssh -f -X -C [user]@[host] xchat
```

Hier wird xchat auf dem host gestartet, aber auf dem lokalen Rechner angezeigt.

Beispiel KDE

Ganze KDE remote starten:

```
user@localhost:~$ Xnest :2&
user@localhost:~$ DISPLAY=:2 ssh remoteuser@remotehost -X
remotehost:~$ startkde
```

- Xnest :2& öffnet ein X Fenster, in dem man z.B. KDE starten kann.
- DISPLAY=:2 ssh remoteuser@remotehost -X verbindet mit dem Server
- startkde startet dann die KDE auf dem Xnest, da die ssh unter verwendung von DISPLAY=:2 gestartet wurde.

Public Key Authentifizierung einrichten

Textkopie von trash.net

Keypair erzeugen

Ich gehe in diesem Beispiel von OpenSSH unter einem Unix-Derivat aus, es kann also durchaus Unterschiede geben, wenn du andere SSH Software benutzt. Als erstes brauchst du mal dein Keypair, bestehend aus einem public Key und einem private Key. Wie der Name sagt, gehört der public Key in die Welt hinaus, und den private Key sollst du hüten wie deinen Augapfel.

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

Dies startet einen kleinen Assistenten, der dir hilft dein SSH2 Keypair zu erzeugen. Dein private Key kann nun mit Hilfe eines Passwortes verschlüsselt werden. Dies ist ein absolutes Muss, da passwortlose private Keys ein enormes Sicherheitsrisiko darstellen. Wähle ein kompliziertes Passwort, bestehend aus Sonderzeichen, Zahlen, und Gross- und Kleinschreibung. Du hast nun zwei Dateien:

```
~/.ssh/id_dsa    <- Hier steckt dein verschlüsselter private Key drin
```

~/ssh/id_dsa.pub <- Dies ist dein public Key, der auf die Remote Server gehört

Den public Key platzieren

Nun musst du deinen public Key auf dem Zielsystem platzieren. Hierfür verwendest du am besten ein dir bekanntes Protokoll. Auf die Bedienung von scp, das für solche Zwecke eigentlich ideal ist, komme ich später zu sprechen. Kopiere die Datei ~/ssh/id_dsa.pub via ftp auf den Zielrechner. Dort speicherst du die Datei als ~/ssh/authorized_keys2 ab. Nun kennt der Server deinen public Key.

Der erste Einloggversuch

Versuche nun, dich mit einem normalen "ssh \$ZIEL" einzuloggen. Wenn du alles richtig gemacht hast, fragt SSH nun nach dem Passwort für die Entschlüsselung deines private Keys. Nachdem der SSH Client deinen private Key entschlüsselt hat kann er eine Verbindung aufbauen, ohne dass du das Passwort eingeben musst, das auf dem Zielhost definiert ist. Jedesmal das Passwort für den private Key anzugeben, ist auf die Dauer etwas mühsam, deswegen haben die Leute hinter OpenSSH ein nettes Programm erschaffen, das dir diese Mühe abnimmt.

ssh-agent

ssh-agent speichert eine entschlüsselte Kopie des private Keys im RAM, und ermöglicht so eine komplett passwortlose & sichere Authentifizierung. ssh-agent zu benutzen ist sehr leicht. Durch die Eingabe von "eval `ssh-agent`" startest du eine Instanz von ssh-agent. Diesem musst du nun noch beibringen, welche Keys er benutzen soll, und wie die dazugehörigen Passwörter heißen.

```
ssh-add ~/ssh/id_dsa
```

Fügt deinen private Key dem Schlüsselbund von ssh-agent hinzu. Hierzu musst du natürlich dein Passwort angeben. Log dich jetzt nochmal mit "ssh \$ZIEL" auf dem Remote Host ein. Nun sollte keinerlei Abfrage kommen, und du befindest dich direkt auf dem Zielrechner. Mit Hilfe von ssh-add -L kannst du dir anzeigen lassen, welche Keys momentan bei ssh-agent registriert sind. Beachte: Jeder der lokalen Zugang zu deinem Computer hat, kann sich ohne Passwörter einloggen. Benutze also xlock und ähnliches, oder beende ssh-agent wenn du nicht vor deinem Computer sitzt.

ssh-agent automatisch starten

Um es noch komfortabler zu machen, empfiehlt es sich, ssh-agent automatisch zu starten. Hierzu gibt es zwei Möglichkeiten:

ssh-agent aus der ~/.profile zu starten. Dies geht nur, wenn man nicht direkt in X startet. Die Anwendung ist relativ einfach, es genügt zwei Zeilen hinzuzufügen:

```
$ cat ~/.profile
eval `ssh-agent`
ssh-add ~/.ssh/id_dsa
```

Solltest du keine sh kompatible Shell benutzen, sieht der Syntax ein wenig anders aus. Anstelle von eval `ssh-agent` solltest du eval `ssh-agent -c` angeben

Die zweite Methode kommt dann zum Zuge, wenn man direkt in X bootet, und sich nicht über eine Shell einloggt. Hierfür benötigst du das Utility gnome-ssh-askpass, welches bei dem OpenSSH Sourcetree im Verzeichnis contrib liegt. Ist Gnome nicht vorhanden, so empfiehlt sich der Einsatz von [x11-ssh-askpass](#). Solltest du nicht ab Sourcen installieren, konsultiere bitte die Dokumentation deiner Distribution. Folgende Einträge in der .xsession führen dazu, das ssh-agent automatisch mit deiner X Session gestartet wird:

```
$ cat ~/.xsession
eval `ssh-agent`
export SSH_ASKPASS=`which gnome-ssh-askpass`
ssh-add ~/.ssh/id_dsa < /dev/null
```

rsync over ssh

Textkopie von trash.net

Ohne rsync Server

SSH lässt sich hervorragend als Transport Layer für rsync benutzen. Damit ist eine sichere und effiziente Übertragung grosser Dateimengen möglich.

```
rsync -r -e ssh $REMOTEHOST:~/blah .
```

Dieser Befehl synchronisiert das Verzeichnis "blah" in das momentan aktive Verzeichnis. Dies ist vorallem dann praktisch, wenn die Funktionalität von scp nicht mehr ausreicht.

Mit rsync Server

Leider hat der rsync Daemon noch keinen integrierten Support für verschlüsselte Verbindungen über SSH. Aber auch hierfür gibt es eine Lösung, vorausgesetzt, du hast einen Shell Account auf dem Zielhost.

```
ssh -L 8730:localhost:873 $ZIELHOST
```

Portforwarding hilft uns auch hier wieder weiter. Allerdings verändert sich der Aufruf von rsync.

```
rsync rsync://$USER@localhost:8730/$ZIELMODUL .
```

Da man bei rsync eine eigene Kompression einschalten kann, mittels der Option -z, ist es unnötig ssh mit -C aufzurufen.

Referenzen / Links

- <http://www.openssh.org>
- http://www.fh-giessen.de/fachschaft/mni/mediawiki/index.php/HOWTO_SSH-Tunnel
- <http://kb.gnuher.de/various/HOWTO%20-%20SSH-Tunnel%20on%20demand.txt>
- <http://www.trash.net/faq/ssh.shtml>

-->

Trackback URL for this post:

<http://www.2030.tk/trackback/29>