OpenSSH
Teil 3: Firewalls durchbohren

Johannes Franken
<jfranken@jfranken.de>

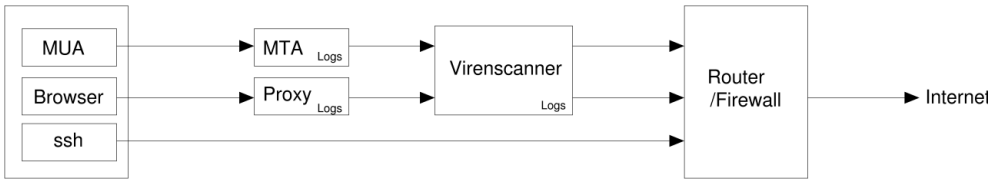
Auf dieser Seite demonstriere ich, wie man ssh in anderen Protokollen versteckt durch Firewalls tunnelt.

Achtung:
Das Umgehen fremder Sicherheitssysteme ist verboten. In diesem Vortrag möchte ich ein Verständnis für Sicherheitslücken vermitteln, damit Sie sich schützen können.

Inhalt

Motivation

ssh-Verbindungen stellen für Firmen ein enormes Risiko dar, schließlich können darüber Tunnels an Virenscannern und Logfiles vorbei gelegt und Dateien übertragen werden.



Viele Firmen haben daher auf ihren Routern oder Firewalls den Zugriff auf Port 22 aller Internetteilnehmer verboten. Da heute jedoch praktisch kein Unternehmen auf einen Internetanschluß verzichten kann, bestehen meist doch gewisse Löcher in der Firewall. Im Folgenden zeige ich, wie man diese entdeckt und eine ssh-Verbindung darüber aufbaut.

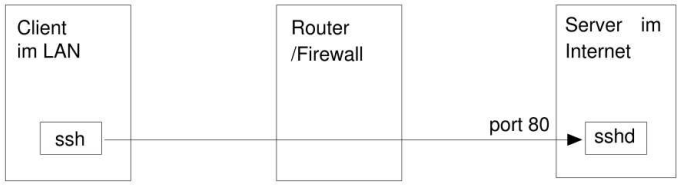
Mit den Kenntnissen aus Teil 2 (insb. ppp-over-ssh) ist es damit trivial, beliebige IP-Verbindungen durch nahezu jede Firewall zu betreiben.

Offene Ports benutzen

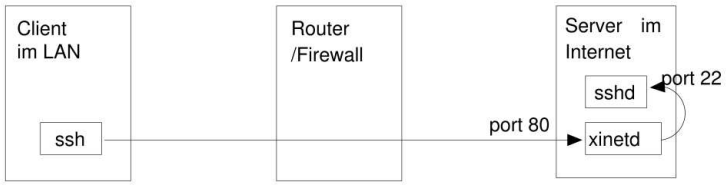
Zunächst sollte man prüfen, welche Ports die Firewall immer durchlässt. Ich verwende hierzu den **nmap**-Portscanner von www.nmap.org:

```
root@hamster$ nmap -sA -p 1-65535 www.jfranken.de
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on www.jfranken.de (195.88.176.20):
(The 21 ports scanned but not shown below are in state: filtered)
Port       State       Service
25/tcp     UNfiltered  smtp
80/tcp     UNfiltered  http
443/tcp    UNfiltered  https
Nmap run completed -- 1 IP address (1 host up) scanned in 5138 seconds
```

Sobald auch nur ein Port **unfiltered** ist (hier z.B. 80), kann ich auf meinem Server für diesen Port einen **sshd** starten:



oder den Port auf dem Server mit **ssh**, **inetd**/**nc**, **xinetd** oder Firewallregeln auf Port 22 umleiten:



Beispiel: siehe Teil 2.

Dem Client gibt man entweder bei jedem Aufruf den Parameter **-p 80** oder trägt den Port dauerhaft in die **~/.ssh/config** ein:

```
Host www.jfranken.de
        Port 80
        User jfranken
```

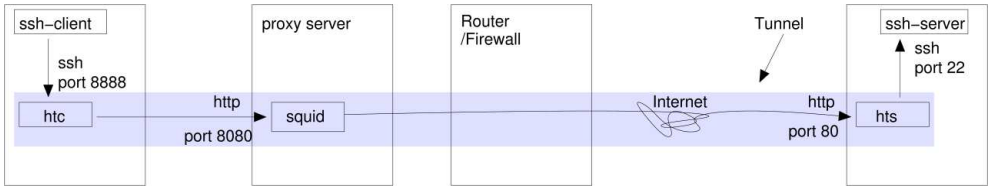
Vorhandene Proxyserver benutzen

Wenn der eigene Rechner (laut **nmap**) keinen einzigen Port im Internet erreichen, aber auf einen Proxyserver zugreifen kann, liegt doch nichts näher, als die ssh-Verbindungen über diesen Proxyserver ins Internet zu leiten.

ssh over http

httptunnel

httptunnel stellt einen tcp Port eines entfernten Servers lokal zur Verfügung. Für die Verbindung sorgen zwei kleine Programme (**hts** und **htc**), die im Netz wie Browser und Webserver über http kommunizieren.



Setup:

- auf dem Server (als root, weil port 80 < 1024):

```
$ hts --no-daemon --forward-port localhost:22 80
```

- auf dem Client:

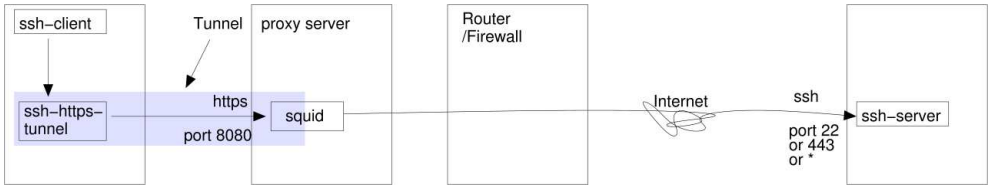
```
$ htc --no-daemon --forward-port 8888 --proxy proxy:8080 --proxy-authorization jfranken:geheim hamster:80 &
$ ssh -p 8888 -o NoHostAuthenticationForLocalhost=yes localhost
```

Problematisch ist zurzeit (httptunnel Version 3.3) , dass

- nur eine Verbindung gleichzeitig über den Tunnel möglich ist
- ein Bug dafür sorgt, dass bei Verwendung eines Proxyservers der **hts** das Ende der Verbindung nicht mitbekommt. Man muß ihn nach jeder Verbindung stoppen und starten.

Mehr zum Thema:
siehe: httptunnel homepage

ssh over https



ssh-https-tunnel

Das Perlscript ssh-https-tunnel weist einen Proxy-Server per CONNECT-Anweisung an, eine TCP-Verbindung zu einem remote Host zu öffnen und ermöglicht die Kommunikation dorthin über stdin/stdout.

Eine Erweiterung von Gerd Aschemann **<gerd@aschemann.net>** vereinfacht die Konfiguration dahingehend, dass man den Proxyserver nicht mehr im Sourcecode eintragen muß, sondern dass dieser aus der Environment-Variable **HTTP_PROXY** oder der Datei **~/.ssh/https-proxy.conf** ausgelesen wird:

```
$ HTTP_PROXY=http://proxy:8080 ./ssh-https-tunnel gate.jfranken.de 25
220 gate.jfranken.de ESMTP Exim 3.35 #1 Fri, 13 Sep 2002 18:08:13 +0200
HELO abc
250 gate.jfranken.de Hello proxy.jfranken.de [192.168.42.2]
QUIT
221 gate.jfranken.de closing connection
$
```

Das Proxy-Log zeigt nach dem Beenden der Verbindung:

```
proxy:~# tail -1 /var/log/squid/access.log| fmt
1031933305.366  11857 192.168.42.20 TCP_MISS/200 213 CONNECT
gate.jfranken.de:25 - DIRECT/192.168.42.1 -
```

Durch so einen Tunnel kann man natürlich auch ssh fahren. Wenn der Proxyserver gut konfiguriert ist oder hinter einer Firewall steht, sind https-Verbindungen möglicherweise nur auf Port 443 möglich. In dem Fall reicht es, einen weiteren ssh-Dämon auf Port 443 zu starten, oder (z.B. per **ssh -gL 443:localhost:22 localhost** or xinetd) den Port auf den bereits auf Port 22 laufenden sshd zu forwarden.

Setup:

- in **~/.ssh/config** folgende Zeilen anfügen:

```
host gate.jfranken.de
  ProxyCommand ~/.ssh/ssh-https-tunnel %h %p
  Port 443
```

- den Proxyserver in die Datei **~/.ssh/https-proxy.conf** eintragen:

```
host=proxy
port=8080
```

Und schon läuft die Verbindung über den Proxyserver an den ssh-Dämon, der auf **gate** auf Port 443 lauscht:

```
$ ssh -v gate.jfranken.de
OpenSSH_3.4p1 Debian 1:3.4p1-2, SSH protocols 1.5/2.0, OpenSSL 0x0090605f
debug1: Reading configuration data /export/home/jfranken/.ssh/config
debug1: Applying options for gate.jfranken.de
debug1: Applying options for *
[...]
debug1: Executing proxy command:  ~/.ssh/ssh-https-tunnel gate.jfranken.de 443
[...]
debug1: Remote protocol version 1.99, remote software version OpenSSH_2.9.9p2
[...]
Last login: Fri Sep 13 18:28:31 2002 from p5080d740.dip.t-dialin.net
Have a lot of fun...
jfranken@gate:~$
```

transconnect

transconnect ist eine C-Bibliothek, welche die connect-Funktion der glibc ersetzt, so dass der Verbindungsaufbau per https über einen Proxyserver läuft. Das funktioniert damit nur für dynamisch gelinkte Binaries (das kann man mit **ldd** prüfen). Der Vorteil gegenüber einem vorher einzurichtenden Tunnel liegt darin, dass die Anwendung flexibel in der Auswahl des Ports ist.

```
$ LD_PRELOAD=~/.tconn/tconn.so netcat hamster daytime
Sat Sep 14 11:25:49 2002
```

siehe transconnect Project Home Page

proxytunnel

proxytunnel ist ein C-Programm, das mit Hilfe einer https-Verbindung zu einem Proxyserver einen remote Port wahlweise als Pipe oder lokalen Port verfügbar macht. Letzteren kann man sogar entweder als Daemon oder (automatisch bei Bedarf) über inetd/tcpd bereitstellen.

Die Syntax ist einfach:

```
$ ./proxytunnel-linux-i386 --help
Proxytunnel 1.1.0
Jos Visser (Muppet) <josv@osp.nl>, Mark Janssen (Maniac) <maniac@maniac.nl>

Purpose:
  Build generic tunnels trough HTTPS proxy's, supports HTTP authorization

Usage: Proxytunnel [OPTIONS]...
   -h         --help              Print help and exit
   -V         --version           Print version and exit
   -c         --config=FILE       Read config options from file (FIXME)
   -i         --inetd             Run from inetd (default=off)
   -a INT     --standalone=INT    Run as standalone daemon on specified port
   -f         --nobackground      Don't for to background in standalone mode (FIXME)
   -u STRING  --user=STRING       Username to send to HTTPS proxy for auth
   -s STRING  --pass=STRING       Password to send to HTTPS proxy for auth
   -g STRING  --proxyhost=STRING  HTTPS Proxy host to connect to
   -G INT     --proxyport=INT     HTTPS Proxy portnumber to connect to
   -d STRING  --desthost=STRING   Destination host to built the tunnel to
   -D INT     --destport=INT      Destination portnumber to built the tunnel to
   -n         --dottedquad        Convert destination hostname to dotted quad
   -v         --verbose           Turn on verbosity (default=off)
   -q         --quiet             Suppress messages  (default=off)

Examples:
Proxytunnel [ -h | -V ]
Proxytunnel -i [ -u user -s pass ] -g host -G port -d host -D port [ -n ] [ -v | -q ]
Proxytunnel -a port [ -u user -s pass ] -g host -G port -d host -D port [ -n ] [ -v | -q ]
```

Beispiele zur Verwendung

- als pipe:

```
$ ssh -o ProxyCommand=./proxytunnel-linux-i386 -g proxy -G 8080 -d %h -D %p' gate
Connected to proxy:8080
Starting tunnel
Linux gate 2.2.19 #5 Wed May 1 20:15:01 CEST 2002 i586 unknown
You have mail.
Last login: Sat Sep 14 10:38:58 2002 from tp.jfranken.de
jfranken@gate:~/ $
```

- als portforwarder:

```
$ ./proxytunnel-linux-i386 -a 8888 -g proxy -G 8080 -d hamster -D 22
Forked into the background with pid 1524
$ ssh -p 8888 -o NoHostAuthenticationForLocalhost=yes localhost
Last login: Sat Sep 14 10:54:42 2002 from gate.jfranken.de
jfranken@hamster:~/ $
```

Das kann man natürlich auch dauerhaft in **`~/.ssh/config`** konfigurieren:

```
Host gate-via-proxy
        Hostname gate.jfranken.de
        ProxyCommand "proxytunnel-linux-i386 -g proxy -G 3128 -d %h -D %p"
```

Mehr zum Thema:
siehe: siehe proxytunnel project homepage

Empfehlungen

Der beste Schutz vor ungewollten ssh-Verbindungen ins Internet besteht darin, den Internetzugriff im LAN zu verbieten und beispielsweise den Browser aus einer DMZ über X11, Citrix Metaframe, VNC o.ä. auf dem Client darzustellen.

Weiterführendes

- The Secure Shell TM Frequently Asked Questions
- OpenSSH Projekt Seite