



OpenSSH

Section 3: Breaking Firewalls

Johannes Franken
<jfranken@jfranken.de>

On this page I show you, how to hide ssh in other protocols and tunnel it through firewalls.

Caution:

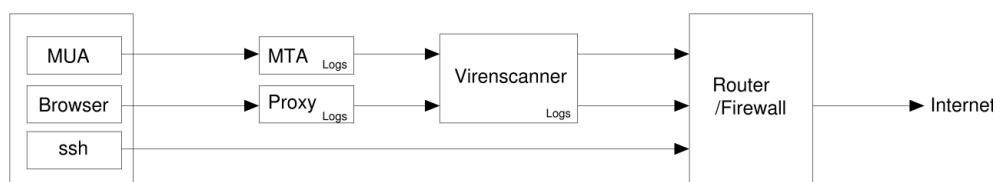
Passing security systems is forbidden. In this talk I try to establish an understanding for security vulnerabilities, so that you can protect yourself.

Contents

1. [Motivation](#)
2. [Using open ports](#)
3. [Using existing proxy-servers](#)
 - a) [ssh over http](#)
 - i) [httptunnel](#)
 - b) [ssh over https](#)
 - i) [ssh-https-tunnel](#)
 - ii) [transconnect](#)
 - iii) [proxytunnel](#)
4. [Recommendations](#)
5. [Further readings](#)

Motivation

ssh-connections pose an enormous risk to companies, as employees can use them to setup tunnels and transfer files, disregarding of any Logfiles and virus scanners.



That's why many companies disallow access to port 22 of any Internet addresses on their router- and firewall-devices. But, since today practically no company can abstain from access to the Internet, they usually have certain holes in their firewall policies. In this talk I will show you, how to explore and (ab?)use them for ssh connections.

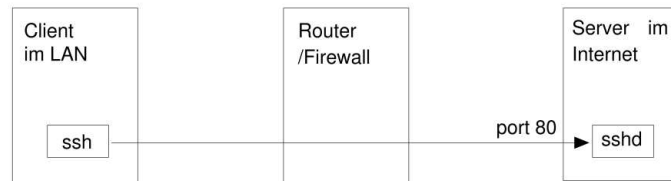
Applying your knowledge from [part 2](#) (particularly ppp-over-ssh), it is trivial to make arbitrary IP-connections through almost any firewall.

Using open ports

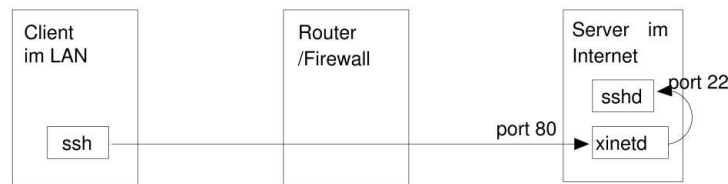
First you should check if the firewall basically lets certain ports pass. I recommend using the **nmap** portscanner of www.nmap.org:

```
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
```

If there's even one single port found to be **unfiltered** (like port 80 in this example), you can run another **sshd** on that port at your server:



or redirect queries from that port to port 22 using **ssh**, **inetd/nc**, **xinetd** or firewall rules:



For an example, see [Part 2](#).

For the client side, either supply the parameter **-p 80** to every call to **ssh**, or permanently configure the port in **~/.ssh/config**:

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

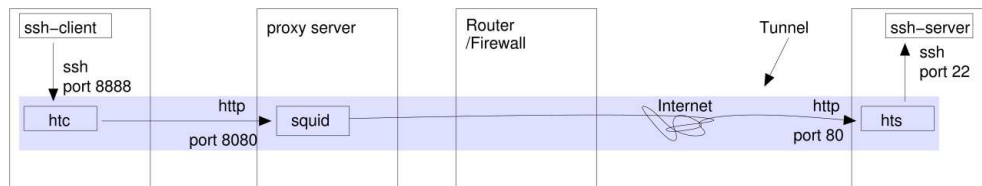
Using existing proxy-servers

If (according to **nmap**) your box can't access any port in the Internet, but you've got web-access via proxy-servers, you can have that proxy-server forward your ssh-connections to the Internet.

ssh over http

httptunnel

httptunnel makes a remote server's tcp port locally available. The connection runs over two little programs (**hts** and **htc**), which communicate in http like a browser and webserver.



Setup:

- on the server (as root, because port 80 < 1024):

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

- on the client:

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

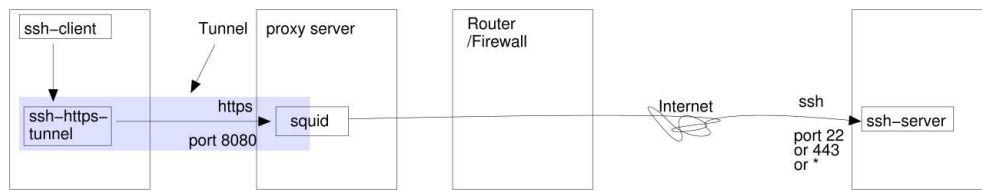
My version of httptunnel (v3.3) still has some bugs:

- only one connection at a time can use the tunnel
- a bug makes **hts** miss the end of proxied connections. You have to kill and restart it after each session.

More about:

see: [httptunnel homepage](http://www.jfranken.de/homepages/johannes/vortraege/ssh3.en.html)

ssh over https



ssh-https-tunnel

The perl-script [ssh-https-tunnel](#) sends a CONNECT-request to your proxy-server, thus directing it to open a TCP-connection with a remote host's port. Then it will allow communication to that port at stdin/stdout.

An [update](#) by Gerd Aschemann <gerd@aschemann.net> simplifies the configuration, so that you don't need to store the proxy settings in the source code any more, because it will read them from your `HTTP_PROXY` environment variable or the file `~/.ssh/https-proxy.conf`.

```
$ 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
$
```

This is what the proxyservers logs, after the connection is over:

```
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 -
```

Of course, you can also drive ssh through such a tunnel. If the proxyserver is well configured or sitting behind a firewall, https-connections might be allowed to port 443 only. In this case, you can start another ssh-demon on port 443 or forward it to your already running sshd on port 22 (for example with `ssh -gL 443:localhost:22 localhost`) or xinetd).

Setup:

- append the following lines to your `~/.ssh/config` file:

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

- store the proxyserver in the file `~/.ssh/https-proxy.conf`:

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

And from now on the connection to the ssh-demon at `gate:443` will take it's way via the proxyserver:

```

$ 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 is a C-library, which replaces the connect-function of glibc in a way, so that any connection establishment is directed over a proxy server. Thus it will work for dynamically linked binaries only (check with `ldd`). It's advantage over any port-based tunneling tool is it's flexibility: it will automatically direct *any* port over the proxyserver.

```

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

```

see [transconnect Project Home Page](#)

proxytunnel

proxytunnel is a C-program, and makes a remote port behind an https proxyserver locally available - either as pipe or listening port. You can even configure it to start on demand (from inetd).

It's got a straight syntax:

```

$ ./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 ]

```

Examples:

- as 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:~/ $
```

- as 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:~/ $
```

Of course, you can configure that permanently in your `~/.ssh/config` file:

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

More about:

see: see [proxytunnel project homepage](#)

Recommendations

The best way to keep unwanted ssh-connections to the Internet from your network is to completely disallow Internet access in your LAN. Don't have proxyservers accessible to any LAN client. Have the browsers running in a DMZ and redirect their output to the workstations via X11, Citrix Metaframe, VNC a.s.o.

Further readings

- [The Secure Shell TM Frequently Asked Questions](#)
- [OpenSSH Project Page](#)