Performing UDP tunneling through an SSH connection

Intro

The Swiss ISP Bluewin sucks. Their DNS are often down. A friend even received advice from Bluewin technicians to not use their own DNS!... But then, it is quite hard to gain access to another DNS for free, if you don't have access to a co hosted machine.

In this document, we'll access another machine's network internal DNS services (UDP port 53) with only SSH access to it. We will forward local UDP/53 traffic to TCP, then TCP traffic with the port-forwarding mechanism of SSH to the other machine, then TCP to UDP/53 on the other end. Typically, you can do it with openvpn. But here, we'll do it with simpler tools, only openssh and netcat.

Step by step

Open a TCP forward port with your SSH connection

On your local machine (`local`), connect to the distant machine (`server`) by SSH, with the additional `-L` option so that SSH will TCP port-forward:

```
local# ssh -L 6667:localhost:6667 server
```

This will allow TCP connections on the port number `6667` of your local machine to be forwarded to the port number `6667` on `server` through the secure channel.

Setup the TCP to UDP forward on the server

On the server, we open a listener on the TCP port 6667 which will forward data to UDP port 53 of a specified IP. If you want to do DNS forwarding like me, you can take the first nameserver's IP you will find in `/etc/resolv.conf` - in this example, this is `192.168.1.1`. But first, we need to create a fifo. The fifo is necessary to have two-way communication between the two channels. A simple shell pipe would only communicate left process' standard output to right process' standard input.

```
server# mkfifo /tmp/fifo
server# nc -l -p 6667 < /tmp/fifo | nc -u 192.168.1.1 53 > /tmp/fifo
```

This will allow TCP traffic on server's port 6667 to be forwarded to UDP traffic on `192.168.1.1`'s port 53, and responses to come back.

Setup the UDP to TCP forward on your machine

Now, we need to do the opposite of what was done upper on the local machine. You need priviledged access to bind the UDP port 53.

```
local# mkfifo /tmp/fifo
local# sudo nc -l -u -p 53 < /tmp/fifo | nc localhost 6667 > /tmp/fifo
```

This will allow UDP traffic on local machine's port 53 to be forwarded to TCP traffic on local machine's port 6667.

Enjoy your local DNS server :)

As you've probably guessed it now, when a DNS query will be performed on the local machine, e.g. on local UDP port 53, it will be forwarded to local TCP port 6667, then to server's TCP port 6667, then to server's DNS server, UDP port 53 of 192.168.1.1. To test DNS service on your local machine, use `host`:

```
# host m6.fr 127.0.0.1
```

If the address is resolved, you can put the following line in your `/etc/resolv.conf` so that your first nameserver is actually you own machine:

```
nameserver 127.0.0.1
```

Alternative solution with `socat`

Brian Marshall has an alternative solution using socat. It eliminates the fifo file requirement. Here's how to do:

Server side: `socat tcp4-listen:5353,reuseaddr,fork UDP:nameserver:53`

Client side: `socat udp4-listen:53,reuseaddr,fork tcp:localhost:5353`

Last update: Thu Nov 8 09:41:24 2007