- Login to Edit

Ubuntu Documentation > Community Documentation > SSHOpenSSHPortForwarding

SSHOpenSSHPortForwarding

Parent page: Internet and Networking >> SSH

Contents

Introduction

SSH allows you to "forward" several types of information. "Forwarding" in this sense means sending information on from an intermediary to a final destination, like how you might forward letters when you move house, or forward phone calls when you're out of the office.

Forwarding Internet connections

It's sometimes useful to smuggle other types of Internet traffic into or out of a network using SSH. For example, to work around network monitoring or snooping, or to avoid badly configured routers on the Internet. You might need to change the settings in other programs (like your web browser) in order to circumvent these filters.

Warning: Filtering and monitoring is usually implemented for a reason. Even if you don't agree with that reason, your IT department might not take kindly to you flouting their rules.

SSH lets you smuggle traffic around by *port-forwarding* a connection to another computer, which works a bit like a telephone *call-forwarding* a connection to another phone number. There are three types of Internet port forwarding:

- **Local port-forwarding**: connections *from the SSH client* are forwarded *via the SSH server*, then *to a destination server*
- **Remote port-forwarding**: connections *from the SSH server* are forwarded *via the SSH client*, then *to a destination server*
- **Dynamic port-forwarding**: connections from *various programs* are forwarded *via the SSH client*, then *via the SSH server*, and finally *to several destination servers*

Local port-forwarding is the most common type. For example, local port-forwarding lets you bypass a company firewall that blocks Wikipedia.

Remote port-forwarding is less common. For example, remote port-forwarding lets you connect from your SSH server to a computer on your company's intranet.

Dynamic port-forwarding is rarely used. For example, dynamic port-forwarding lets you bypass a company firewall that blocks web access altogether. Although this is very powerful, it takes a lot of work to set up, and it's usually easier to use local port-forwarding for the specific sites you want to access.

Port-forwarding is a widely useful technique that is supported in all major SSH clients and servers, although you will have to consult your client's documentation to find out exactly how your client does it. For examle, the PuTTY manual has a section on port-forwarding in PuTTY.

To use port-forwarding, you need to make sure port-forwarding is enabled it in your server. You also need to tell your client the source and destination *port numbers* to use. If you're using local- or remote-forwarding, you need to tell your client the *destination server*. If you're using dynamic port-forwarding, you need to configure your programs to use a SOCKS proxy server. Exactly how to do this depends on which SSH client you use, so you may need to consult your documentation.

Local port-forwarding

Local port-forwarding lets you connect from your *local* computer to another server. To use local port-forwarding, you need to know your destination server, and two port numbers. You should already know your destination server, and for basic uses of port-forwarding, you can usually use the port numbers in Wikipedia's WikiPedia:list of TCP and UDP port numbers.

For example, say you wanted to connect through your laptop to http://www.ubuntu.com, using the command-line SSH client. You would use source port number `8080` (the alternate *http* port), destination port `80` (the *http* port), and destination server *www.ubuntu.com*:

```
ssh -L 8080:www.ubuntu.com:80 laptop
```

The `-L` option specifies *local* port-forwarding. For the duration of the SSH session, pointing your browser at http://localhost:8080/ would download pages from http://www.ubuntu.com/.

In the above example, we used port `8080` for the source port. Ports numbers less than `1024` or greater than `49151` are reserved for the system, and some programs will only work with specific source ports, but otherwise you can use any source port number. For example, you could do:

```
ssh -L 8080:www.ubuntu.com:80 -L 12345:ubuntuforums.org:80 laptop
```

This would forward two connections, one to www.ubuntu.com, the other to www.ubuntuforums.org. Pointing your browser at http://localhost:8080/ would download pages from www.ubuntu.com, and pointing your browser to http://localhost:12345/ would download pages from www.ubuntuforums.org.

The destination server can even be the same as the SSH server. For example, you could do:

```
ssh -L 5900:localhost:5900 laptop
```

This would forward connections to the shared desktop on your SSH server (if one had been set up). Connecting an SSH client to *localhost* port `5900` would show the desktop for that computer. The word "localhost" is the computer equivalent of the word "yourself", so the SSH server on your laptop will understand what you mean, whatever the computer's actual name.

Remote port-forwarding

Remote port-forwarding lets you connect from the *remote* SSH server to another server. To use remote port-forwarding, you need to know your destination server, and two port numbers. You should already know your destination server, and for basic uses of port-forwarding, you can usually use the port numbers in Wikipedia's WikiPedia:list of TCP and UDP port numbers.

For example, say you wanted to let a friend access your remote desktop, using the command-line SSH client. You would use port number `5900` (the first *VNC* port), and destination server *localhost*:

```
ssh -R 5900:localhost:5900 guest@joes-pc
```

The `-R` option specifies *remote* port-forwarding. For the duration of the SSH session, Joe

would be able to access your desktop by connecting a VNC client to port 5900 on his computer (if you had set up a shared desktop).

Dynamic port-forwarding

Dynamic port-forwarding turns your SSH client into a *SOCKS proxy server*. *SOCKS* is a little-known but widely-implemented protocol for programs to request any Internet connection through a *proxy server*. Each program that uses the proxy server needs to be configured specifically, and reconfigured when you stop using the proxy server.

For example, say you wanted Firefox to connect to every web page through your SSH server. First you would use dynamic port-forwarding with the default SOCKS port:

```
ssh -C -D 1080 laptop
```

The `-D` option specifies *dynamic* port-forwarding. `1080` is the standard SOCKS port. Although you can use any port number, some programs will only work if you use `1080`. `-C` enables compression, which speeds the tunnel up when proxying mainly text-based information (like web browsing), but can slow it down when proxying binary information (like downloading files).

Next you would tell Firefox to use your proxy:

- go to Edit -> Preferences -> Advanced -> Network -> Connection -> Settings...
- check "Manual proxy configuration"
- make sure "Use this proxy server for all protocols" is cleared
- clear "HTTP Proxy", "SSL Proxy", "FTP Proxy", and "Gopher Proxy" fields
- enter "127.0.0.1" for "SOCKS Host"
- enter "1080" (or whatever port you chose) for Port.

You can also set Firefox to use the DNS through that proxy, so even your DNS lookups are secure:

- Type in about:config in the Firefox address bar
- Find the key called "network.proxy.socks_remote_dns" and set it to true

The SOCKS proxy will stop working when you close your SSH session. You will need to change these settings back to normal in order for Firefox to work again.

To make other programs use your SSH proxy server, you will need to configure each program in a similar way.

Port-forwarding theory

To get the most out of port-forwarding, it's helpful to know a bit about how the Internet works.

The Internet assigns computers virtual "ports", a bit like the USB ports on the back of your computer:

To let a digital camera share pictures with your PC, you connect the USB port on the camera to any USB port on the PC. The computer then talks to the camera about your photos, and shows you the result.

To let a web server share pages with your PC, you connect the web server port on the server to any Internet port on the PC. The computer then talks to the server about your page, and shows you the result.

Unlike a USB port, there is no physical component to an Internet port. There's no actual wire, or actual hole on the back of your computer. It's all just messages being sent over the Internet. Like other "virtual" computer concepts, Internet ports are just an analogy that help to

explain what your computer is doing. Sometimes, that analogy breaks down:

There are two types of Internet port: normal "TCP" ports and strange "UDP" ports (which won't be covered here).

Unlike USB ports, every computer has exactly 65,535 numbered TCP ports, some of which have a special purpose. For example, port number 80 is your web server port, so your web browser knows it should connect to port number 80 in order to download a web page.

Connections between Internet ports can be patched together, so a connection from computer A to computer B on port 12,345 could be patched through to port number 80 on computer C. This is known as **port forwarding**.

Forwarding GUI Programs

SSH can also forward graphical applications over a network, although it can take some work and extra software to forward programs to Windows or Mac OS.

Single Applications

If you are logging in from a Unix-like operating system, you can forward single applications over SSH very easily, because all Unix-like systems share a common graphics layer called X11. This even works under Mac OS X, although you will need to install and start the X11 server before using SSH.

To forward single applications, connect to your system using the command-line, but add the **-X** option to forward X11 connections:

```
ssh -X laptop
```

Once the connection is made, type the name of your GUI program on the SSH command-line:

```
firefox &
```

Your program will start as normal, although you might find it's a little slower than it would be if it were running locally. The trailing `&` means that the program should run in "background mode", so you can start typing new commands in straight away, rather than waiting for your program to finish.

If you only want to run a single command, you can log in like this:

```
ssh -f -T -X laptop firefox
```

That will run Firefox, then exit when it finishes. See the SSH manual page for information about **-f** and **-T**.

If you start an application and it complains that it cannot find the display, try installing the **xauth** package from the Main repository (click here to install xauth). Xauth is installed by default with desktop installations but not server installations.

If you suspect that programs are running slowly because of a lack of bandwith, you can turn SSH compression on with the **-C** option:

```
ssh -fTXC joe@laptop firefox
```

Using **-fTXC** here is identical to **-f -T -X -C**.

Nested windows

Xephyr is a program that gives you an X server within your current server. It's available in the **xserver-xephyr** package in the Main repository (click here to install xserver-xephyr).

Two ssh forwarded desktops on dual monitors, click to enlarge

Setting up Xephyr was explained briefly in the Ubuntu forums.

Troubleshooting

If you get a message like this when you try to forward a port:

```
bind: Address already in use
channel_setup_fwd_listener: cannot listen to port: <port number>
Could not request local forwarding.
```

then someone is already listening on that port number. You won't be able to listen on that port until the other person has finished with it.

If forwarding doesn't seem to work, even though you didn't get a warning message, then your SSH server might have disabled forwarding. To check, do the following:

```
grep Forwarding /etc/ssh/sshd_config
```

If you see something like this:

```
X11Forwarding no
AllowTcpForwarding no
```

then forwarding is disabled on your server. See the SSH configuration page for more information.

SSH/OpenSSH/PortForwarding (last edited 2009-06-22 15:27:30 by https://login.launchpad.net/+id/YH4fWht @ cpc5-hitc4-0-0-cust385.lutn.cable.ntl.com[81.105.201.130]:Andrew Sayers)

- Parent Page
- Page History