Is it possible to forward ssh requests that come in over a certain port to another machine?

Asked 16 years, 3 months ago Modified 16 years, 3 months ago

Viewed 5k times

6

I have a small local network. Only one of the machines is available to the outside world (this is not easily changeable). I'd like to be able to set it up such that ssh requests that don't come in on the standard port go to another machine. Is this possible? If so, how?

Oh and all of these machines are running either Ubuntu or OS X.

macos ubuntu ssh port

Share

Improve this question

Follow

asked Sep 5, 2008 at 4:30

Paul Wicks

65.4k • 55 • 124 • 148

6 Answers

Sorted by: Highest score (default)

Another way to go would be to use ssh tunneling (which happens on the client side).

12

You'd do an ssh command like this:

```
ssh -L 8022:myinsideserver:22 paul@myoutsideserver
```

That connects you to the machine that's accessible from the outside (myoutsideserver) and creates a tunnel through that ssh connection to port 22 (the standard ssh port) on the server that's only accessible from the inside.

Then you'd do another ssh command like this (leaving the first one still connected):

```
ssh -p 8022 paul@localhost
```

That connection to port 8022 on your localhost will then get tunneled through the first ssh connection taking you over myinsideserver.

There may be something you have to do on myoutsideserver to allow forwarding of the ssh port. I'm double-checking that now.

Edit

Hmmm. The ssh manpage says this: **Only the superuser can forward privileged ports. **

~~That sort of implies to me that the first ssh connection has to be as root. Maybe somebody else can clarify that.~~

It looks like superuser privileges aren't required as long as the forwarded port ***(in this case, 8022)*** isn't a

privileged port (like 22). Thanks for the clarification Mike Stone.

Share Improve this answer

Follow

edited May 23, 2017 at 11:55

Community Bot

1 ● 1

answered Sep 5, 2008 at 4:40

Mark Biek

151k ● 54 ● 158 ● 201

1 I've been trying to figure out how to do this and finally came across this! I wish I could click the up arrow more than once! – David Jones Feb 13, 2018 at 20:23

5

@Mark Biek

I was going to say that, but you beat me to it! Anyways, I just wanted to add that there is also the -R option:

```
ssh -R 8022:myinsideserver:22 paul@myoutsideserver
```

The difference is what machine you are connecting to/from. My boss showed me this trick not too long ago, and it is definitely really nice to know... we were behind a firewall and needed to give external access to a machine... he got around it by ssh -R to another machine that was accessible... then connections to that machine were forwarded into the machine behind the firewall, so

you need to use -R or -L based on which machine you are on and which you are ssh-ing to.

Also, I'm pretty sure you are fine to use a regular user as long as the port you are forwarding (in this case the 8022 port) is not below the restricted range (which I think is 1024, but I could be mistaken), because those are the "reserved" ports. It doesn't matter that you are forwarding it to a "restricted" port because that port is not being opened (the machine is just having traffic sent to it through the tunnel, it has no knowledge of the tunnel), the 8022 port IS being open and so is restricted as such.

EDIT: Just remember, the tunnel is only open so long as the initial ssh remains open, so if it times out or you exit it, the tunnel will be closed.

Share Improve this answer

Follow

edited May 23, 2017 at 10:32

Community Bot

1 ● 1

answered Sep 5, 2008 at 5:10

Mike Stone

44.6k ● 30 ● 114 ● 140

4

(In this example, I am assuming port 2222 will go to your internal host. $externalip and $internalip are the ip addresses or hostnames of the visible and internal machine, respectively.)

You have a couple of options, depending on how permanent you want the proxying to be:

- Some sort of TCP proxy. On Linux, the basic idea is that *before* the incoming packet is processed, you want to *change its destination*—i.e. prerouting destination NAT:

  ```
  iptables -t nat -A PREROUTING -p tcp -i eth0 -d $externalip --dport 2222 --sport 1024:65535 -j DNAT --to $internalip:22
  ```

- Using SSH to establish temporary port forwarding. From here, you have two options again:
  - Transparent proxy, where the client thinks that your visible host (on port 2222) is just a normal SSH server and doesn't realize that it is passing through. While you lose some fine-grained control, you get convenience (especially if you want to use SSH to forward VNC or X11 all the way to the inner host).
    - From the internal machine: `ssh -g -R 2222:localhost:22 $externalip`
    - Then from the outside world: `ssh -p 2222 $externalip`

    Notice that the "internal" and "external" machines do not have to be on the same LAN. You can port forward all the way around the world this way.

- Forcing login to the external machine first. This is true "forwarding," not "proxying"; but the basic idea is this: You force people to log in to the external machine (so you control on who can log in and when, and you get logs of the activity), and from there they can SSH through to the inside. It sounds like a chore, but if you *set up simple shell scripts on the external machine with the names of your internal hosts, coupled with password-less SSH keypairs* then it is very straightforward for a user to log in. So:
  - On the external machine, you make a simple script, `/usr/local/bin/internalhost` which simply runs `ssh $internalip`
  - From the outside world, users do: `ssh $externalip internalhost` and once they log in to the first machine, they are immediately forwarded through to the internal one.

  Another advantage to this approach is that people don't get key management problems, since running two SSH services on one IP address will make the SSH client angry.

FYI, if you want to SSH to a server and you do not want to worry about keys, do this

```
ssh -o UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no
```

I have an alias in my shell called "nossh", so I can just do `nossh somehost` and it will ignore all key errors. Just understand that you are ignoring security information when you do this, so there is a theoretical risk.

Much of this information is from a talk I gave at Barcamp Bangkok all about fancy SSH tricks. You can see my slides, but I recommend the text version as the S5 slides are kind of buggy. Check out the section called "Forward Anything: Simple Port Forwarding" for info. There is also information on creating a SOCKS5 proxy with OpenSSH. Yes, you can do that. OpenSSH is awesome like that.

(Finally, if you are doing a lot of traversing into the internal network, consider setting up a VPN. It sounds scary, but OpenVPN is quite simple and runs on all OSes. I would say it's overkill just for SSH; but once you start port-forwarding through your port-forwards to get VNC, HTTP, or other stuff happening; or if you have lots of internal hosts to worry about, it can be simpler and more maintainable.)

edited Sep 5, 2008 at 6:00

answered Sep 5, 2008 at 5:33

JasonSmith
73.7k • 23 • 126 • 149

0

You can use Port Fowarding to do this. Take a look here:

http://portforward.com/help/portforwarding.htm

There are instructions on how to set up your router to port forward request on this page:

http://www.portforward.com/english/routers/port_forwarding/routerindex.htm

Share Improve this answer

Follow

answered Sep 5, 2008 at 4:32

Espo

41.9k ● 21 ● 136 ● 161

0

In Ubuntu, you can install Firestarter and then use it's Forward Service feature to forward the SSH traffic from a non standard port on your machine with external access to port 22 on the machine inside your network.

On OS X you can edit the /etc/nat/natd.plist file to enable port fowarding.

Share Improve this answer

Follow

edited Sep 5, 2008 at 6:21

answered Sep 5, 2008 at 5:57

Jay Hofacker

3,469 ● 21 ● 14

0

Without messing around with firewall rules, you can set up a ~/.ssh/config file.

Assume 10.1.1.1 is the 'gateway' system and 10.1.1.2 is the 'client' system.

```
Host gateway
  Hostname 10.1.1.1
  LocalForward 8022 10.1.1.2:22

Host client
  Hostname localhost
  Port 8022
```

You can open an ssh connection to 'gateway' via:

```
ssh gateway
```

In another terminal, open a connection to the client.

```
ssh client
```

Share Improve this answer

Follow

answered Sep 16, 2008 at 18:39

jtimberman

8,258 ● 2 ● 45 ● 37