SSH Tunnel - Port Forwarding

What is Port Forwarding?

Port forwarding via SSH (SSH tunneling) creates a secure connection between a local computer and a remote machine through which services can be relayed.

Types of Port Forwarding

There are three types of port forwarding with SSH:

- **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 to access a particular site.

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.

Local forwarding

Local forwarding is used to forward a port from the client machine to the server machine. Basically, the SSH client listens for connections on a configured port, and when it receives a connection, it tunnels the connection to an SSH server. The server connects to a configured destination port, possibly on a different machine than the SSH server.

Typical uses for local port forwarding include:

Tunneling sessions and file transfers through jump servers
Connecting to a service on an internal network from the outside
Connecting to a remote file share over the Internet
Quite a few organizations for all incoming SSH access through a single jump server. The server may be a standard Linux/Unix box, usually with some extra hardening, intrusion detection, and/or logging, or it may be a commercial jump server solution.

Many jump servers allow incoming port forwarding, once the connection has been authenticated. Such port forwarding is convenient, because it allows tech-savvy users to use internal resources quite transparently.

Enabling Port Forwarding

By default you should be able to tunnel network connections through an SSH session. You can enable/disable port forwarding by editing the ssh daemon configuration file. The file is located at:

    /etc/ssh/sshd_config

Disable forwarding

To disable forwarding, look for the following lines in your sshd_config:

    AllowTcpForwarding yes
    X11Forwarding yes

and replace them with:

    AllowTcpForwarding no
    X11Forwarding no

If either of the above lines don't exist, just add the replacement to the bottom of the file.

Restarting SSH Server

You can use the “Service” command to stop, start, restart and check the status of the SSH Server. For example to check the status of the SSH Server, run the following command:

```
service sshd status
```

```
[root@compute903 ssh]# service sshd status
Redirecting to /bin/systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2018-03-23 10:55:08 EDT; 4 months 16 days ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1949 (sshd)
   CGroup: /system.slice/sshd.service
           └─1949 /usr/sbin/sshd -D
```

Available Options

Below is a list of base flags used between the HPC cluster, compute901 - compute904 and our jump node at us0789lnxp. All our port forwards are configured with the following format:

```
ssh -L -nNTf us0789lnxp.america.apci.com:19888:192.168.8.252:19888 root@192.168.8.252
```

-L [bind_address:]port:host:hostport

Required option that specifies the actual binding of the local host and port to the remote host and port.

Manual Page Description

> Specifies that connections to the given TCP port or Unix socket on the local (client) host are to be forwarded to the given host and port, or Unix socket, on the remote side. This works by allocating a socket to listen to either a TCP port on the local side, optionally bound to the specified bind_address, or to a Unix socket. Whenever a connection is made to the local port or socket, the connection is forwarded over the secure channel, and a connection is made to either host port hostport, or the Unix socket remote_socket, from the remote machine.

-f

Takes the port forward command executed on the command line and places it in the background. This allows the binding to continue after the local user terminal has exited.

Manual Page Description
Requests ssh to go to background just before command execution.

-nNT

Every time we create a tunnel you also SSH into the server and get a shell. This isn't usually necessary, as you're just trying to create a tunnel. To avoid this we can run SSH with the -nNT flags, such as the following, which will cause SSH to not allocate a tty and only do the port forwarding.

Manual Page Description
-N Do not execute a remote command. This is useful for just forwarding ports.
-n Redirects stdin from /dev/null (actually, prevents reading from stdin). This must be used when ssh is run in the background.
-T Disable pseudo-terminal allocation.

Ping Command

You can use the ping command to determine the IP address of an server domain name.
[johnsol3@us0789lnxp man]$ ping compute901
PING compute901 (192.168.8.253) 56(84) bytes of data.
64 bytes from compute901 (192.168.8.253): icmp_seq=1 ttl=64 time=0.101 ms
64 bytes from compute901 (192.168.8.253): icmp_seq=2 ttl=64 time=0.064 ms
64 bytes from compute901 (192.168.8.253): icmp_seq=3 ttl=64 time=0.089 ms
64 bytes from compute901 (192.168.8.253): icmp_seq=4 ttl=64 time=0.092 ms
64 bytes from compute901 (192.168.8.253): icmp_seq=5 ttl=64 time=0.065 ms

--- compute901 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.064/0.082/0.101/0.016 ms

Manual Page Description
Ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (``pings'') have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of ``pad'' bytes used to fill out the packet.

Troubleshooting

Is the particular port forward command running?

Port forwarding is accomplished via an ssh server running on the remote server. So the first step in troubleshooting a failed port forward is to determine if there is an ssh process running to execute the port forward. To list all processes running on your system, you can use te linux ‘ps’ command.

- ps -ef

The ‘-e’ option specifies to list all process
The ‘-f’ option specifies to do a full-format listing.

To limit the output of the ps command to processes that specify your requested port number you can pipe the results of the ps command to the ‘grep’ command:

- ps -ef | grep ‘3939’

In the command above, i’m looking for any process running on the server that has ‘3939’ in the results.

```
[johnsol3@us0789lnxp ~]$ ps -ef|grep 3939
johnsol3  25514      1  0 Jun20 ?        00:00:11 ssh -fnNT -L
us0789lnxp.america.apci.com:3939:192.168.8.251:3939 root@192.168.8.251
johnsol3  31894  27646  0 11:26 pts/53   00:00:00 grep --color=auto 3939
[johnsol3@us0789lnxp ~]$
```

SSH Command

In the above results we can see an entry for an ssh command with 3939 in the full format listing. If the port forwarding was not setup and running this entry would not show in the results. In that case you can rerun the ssh command to restart the process:

```
ssh -fnNT -L us0789lnxp.america.apci.com:3939:192.168.8.251:3939 root@192.168.8.251
```