www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Using SSH to Securely Tunnel a TCP Application

Often times, I'm caught in a situation where an application or service is not directly accessible to me from the network I'm currently on. For instance, if I'm at a customer site, I may need VNC or RDP access to a collocated server. Sure, I could open up port 5901 to the Internet on my firewall, but that would be ill advised, as VNC sends passwords in clear text. Even still, the site I'm at may have port 5901 outbound blocked.

Similarly, there have been situations where customers have two or more separate networks, joined only by certain, multi-homed machines, or machines which are privileged in terms of router access lists. How do you then go about accessing a web app for instance via those machines?

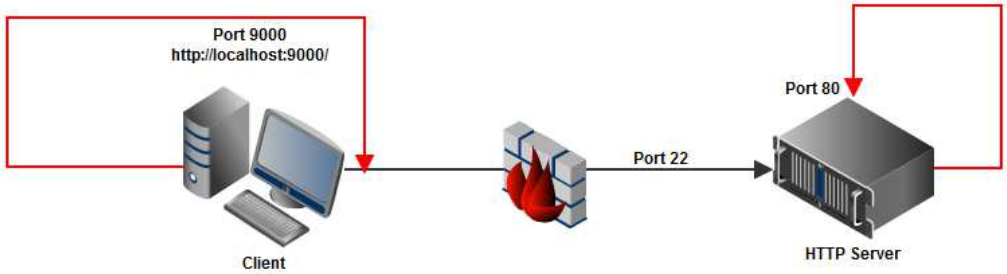
Take another situation -- you're at the airport looking for public wifi. You join a wireless network which you believe is legitimate. Can you be certain? When privacy is of concern, I immediately open up an ssh tunnel to my personal linux machine where a squid proxy is running, and I tunnel all web traffic through that.

Not only does SSH provide encryption, but it can also provide compression. So applications such as VNC and HTTP further benefit from this, especially when you're on a WAN link with limited bandwidth.

The list of potential situations where an ssh tunnel would be useful is endless. Continue reading my howto and you'll be an ssh tunnel guru in no time.

Scenario 1: Basic SSH tunneling

In this situation, we want to access a service running on a server, but all that's available to us (or all that we choose to use) is SSH.



In order to tunnel the HTTP traffic, the SSH client listens on port 9000 on the client. 9000 is an arbitrary port—you can choose any port that is not already taken on the client. When we point our browser at http://localhost:9000, the traffic is sent through the tunnel, where the SSH server is connected to our web server port listening on port 80.

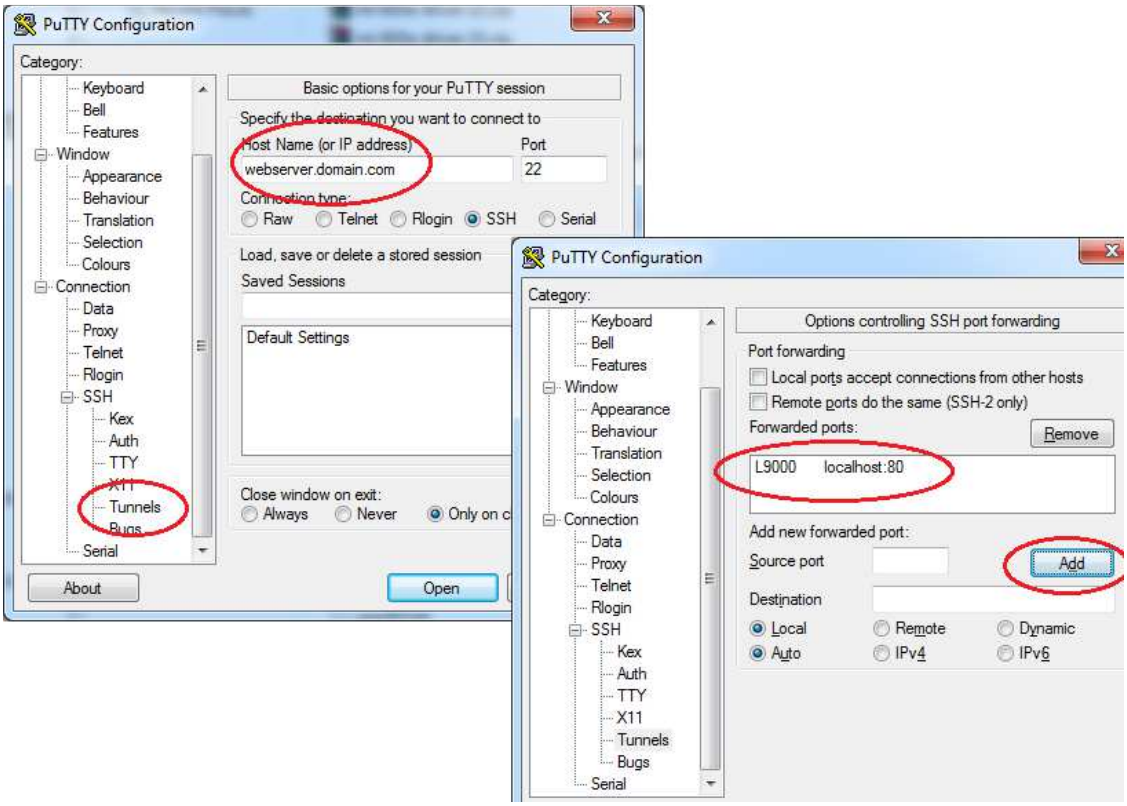
www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

If you have Cygwin or any other command line SSH client installed, your connection would look something like this:

```
bash#  ssh –C mike@webserver.domain.com -L 9000:localhost:80
```

Similarly, you could use PuTTY to achieve the same thing:



Note the -C parameter in the ssh command listed above. This will allow for SSH compression. SSH compression does add latency to the connection, so your mileage may vary depending on your application. VNC, HTTP, and other text based protocols compress quite well. SSL traffic on the other hand, compresses poorly.

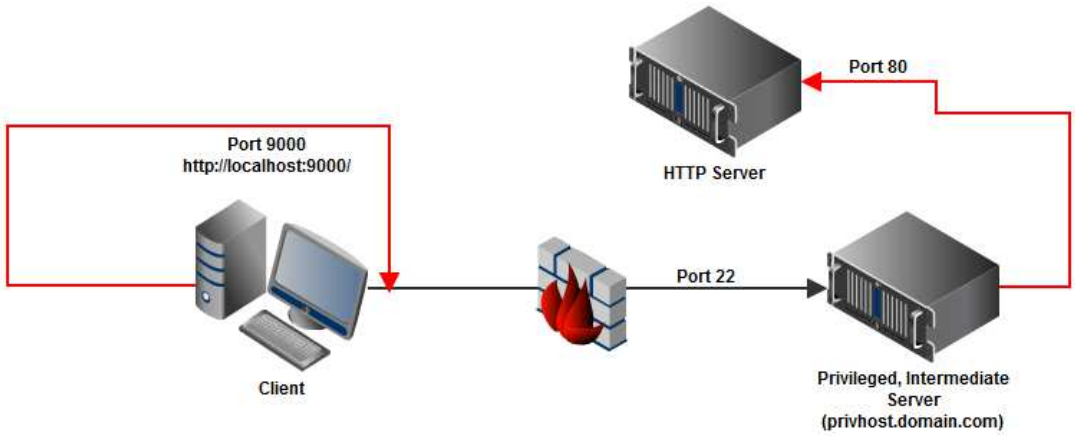
Another thing to note is that you are not limited to only tunneling one port or application. You can append as many -L parameters as you like as long as the local ports are both unique and not in use. For instance, if we wanted to tunnel both VNC and HTTP at the same time, it would look something like this: -L 9000:vncserver:5901 -L 9001:httpserver:80, etc.

Scenario 2: SSH tunnel through one machine to another

In this situation, our client needs to connect to a web server, but that web server is only accessible from a specific host.

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

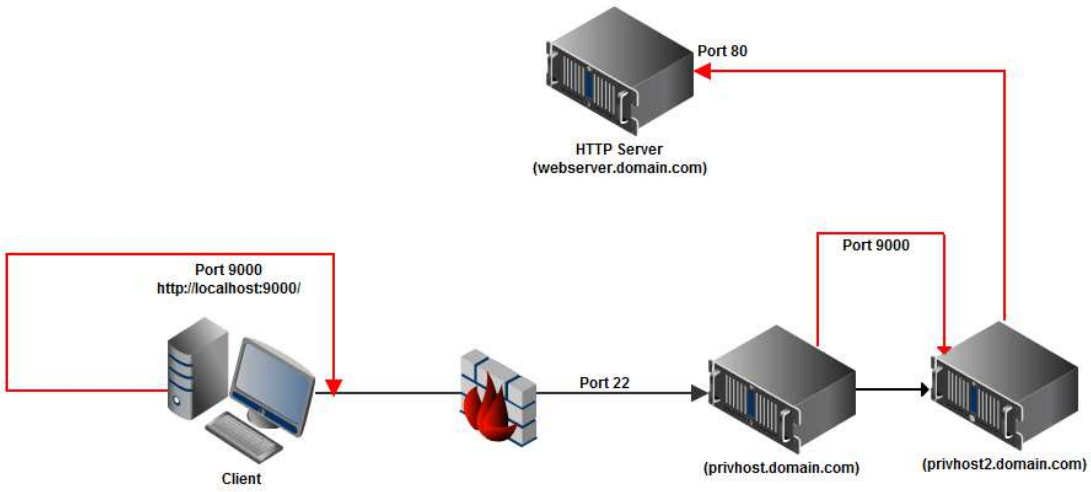


The ssh command in this situation would be similar to:

```
bash#  ssh mike@privhost.domain.com -L 9000:httpserver:80
```

Situation 3: Advanced SSH tunneling, through multiple hosts

Perhaps you have a web host that is several hops away. We can "chain" our ssh tunnel across as many hosts as necessary to make it to the destination.




```
bash#  ssh mike@privhost.domain.com -L 9000:localhost:9000

privhost#  ssh mike@privhost2.domain.com -L 9000:webserver.domain.com:80
```

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Potential Limitations

Unfortunately, ssh tunneling has some limitations. Primarily, it can only tunnel TCP based applications. So any UDP based applications would not work. Another small potential issue is that from the server's perspective, the traffic would appear to come from itself, since the local ssh daemon is the one creating the connection. This may cause issues for virtualhosted web apps depending on the setup.

Lastly, you may want to review network usage policies for your organization, as this has the potential to abuse / circumvent any policies that an enterprise may have in place.

Michael Solinap
Sr. Systems Integrator
SPK & Associates