Source Open

About Technology and Life With A Touch Of My Own

Feeds: Posts Comments

SSH Tunneling Explained

March 21, 2012 by Buddhika Chamith

Recently I wanted to set up a remote desktop sharing session from home pc to my laptop. While going through the set up guide I came across ssh tunneling. Even though there are many articles on the subject still it took me a considerable amount of googling, some experimenting and couple of Wireshark sessions to grasp what's going under the hood. Most of the guides were incomplete in terms of explaining the concept which left me desiring for a good article on the subject with some explanatory illustrations. So I decided to write it my self. So here goes...

Introduction

A SSH tunnel consists of an encrypted tunnel created through a SSH protocol connection. A SSH tunnel can be used to transfer unencrypted traffic over a network through an encrypted channel. For example we can use a ssh tunnel to securely transfer files between a FTP server and a client even though the FTP protocol itself is not encrypted. SSH tunnels also provide a means to bypass firewalls that prohibits or filter certain internet services. For example an organization will block certain sites using their proxy filter. But users may not wish to have their web traffic monitored or blocked by the organization proxy filter. If users can connect to an external SSH server, they can create a SSH tunnel to forward a given port on their local machine to port 80 on remote web-server via the external SSH server. I will describe this scenario in detail in a little while.

To set up a SSH tunnel a given port of one machine needs to be forwarded (of which I am going to talk about in a little while) to a port in the other machine which will be the other end of the tunnel. Once the SSH tunnel has been established, the user can connect to earlier specified port at first machine to access the network service.

Port Forwarding

SSH tunnels can be created in several ways using different kinds of port forwarding mechanisms. Ports can be forwarded in three ways.

1. Local port forwarding
2. Remote port forwarding
3. Dynamic port forwarding

I didn't explain what port forwarding is. I found Wikipedia's definition more explanatory.

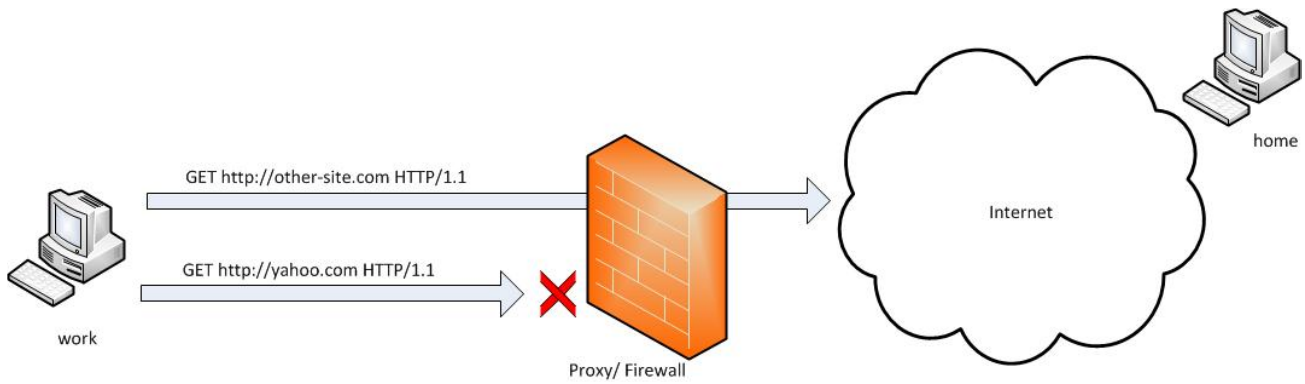
> Port forwarding or port mapping is a name given to the combined technique of
>
> 1. translating the address and/or port number of a packet to a new destination
> 2. possibly accepting such packet(s) in a packet filter(firewall)
> 3. forwarding the packet according to the routing table.

Here the first technique will be used in creating an SSH tunnel. When a client application connects to the local port (local endpoint) of the SSH tunnel and transfer data these data will be forwarded to the remote end by translating the host and port values to that of the remote end of the channel.

So with that let's see how SSH tunnels can be created using forwarded ports with an examples.

Tunnelling with Local port forwarding

Let's say that yahoo.com is being blocked using a proxy filter in the University.
(For the sake of this example. :). Cannot think any valid reason why yahoo would be blocked). A SSH tunnel can be used to bypass this restriction. Let's name my machine at the university as 'work' and my home machine as 'home'. 'home' needs to have a public IP for this to work. And I am running a SSH server on my home machine. Following diagram illustrates the scenario.



(https://chamibuddhika.files.wordpress.com/2012/03/scenario.jpg)

To create the SSH tunnel execute following from 'work' machine.

```
ssh -L 9001:yahoo.com:80 home
```

The ‘L’ switch indicates that a local port forward is need to be created. The switch syntax is as follows.

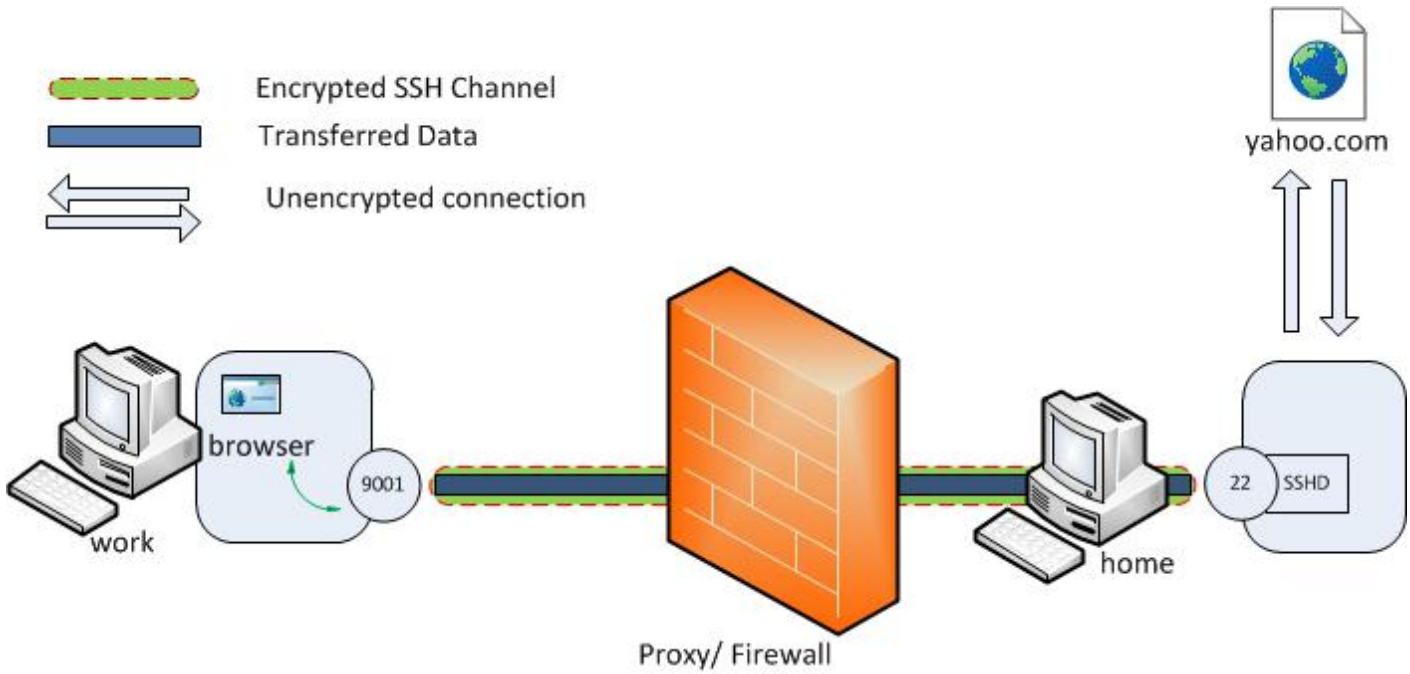
```
-L <local-port-to-listen>:<remote-host>:<remote-port>
```

Now the SSH client at ‘work’ will connect to SSH server running at ‘home’ (usually running at port 22) binding port 9001 of ‘work’ to listen for local requests thus creating a SSH tunnel between ‘home’ and ‘work’. At the ‘home’ end it will create a connection to ‘yahoo.com’ at port 80. So ‘work’ doesn’t need to know how to connect to yahoo.com. Only ‘home’ needs to worry about that. The channel between ‘work’ and ‘home’ will be encrypted while the connection between ‘home’ and ‘yahoo.com’ will be unencrypted.

Now it is possible to browse yahoo.com by visiting http://localhost:9001 (http://localhost:9001) in the web browser at ‘work’ computer. The ‘home’ computer will act as a gateway which would accept requests from ‘work’ machine and fetch data and tunnelling it back. So the syntax of the full command would be as follows.

```
ssh -L <local-port-to-listen>:<remote-host>:<remote-port> <gateway>
```

The image below describes the scenario.



(https://chamibuddhika.files.wordpress.com/2012/03/localportforwarding.jpg)

Here the ‘host’ to ‘yahoo.com’ connection is only made when browser makes the request not at the tunnel setup time.

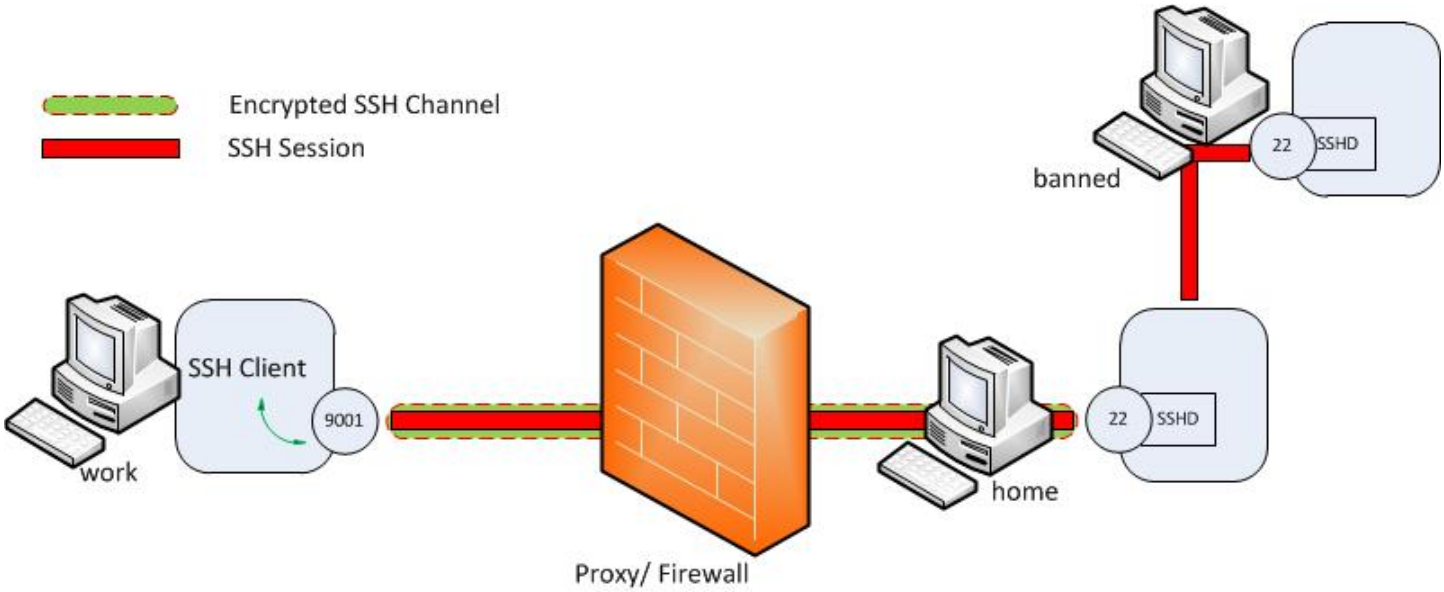
It is also possible to specify a port in the ‘home’ computer itself instead of connecting to an external host. This is useful if I were to set up a VNC session between ‘work’ and ‘home’. Then the command line would be as follows.

```
ssh -L 5900:localhost:5900 home (Executed from 'work')
```

So here what does localhost refer to? Is it the ‘work’ since the command line is executed from ‘work’? Turns out that it is not. As explained earlier is relative to the gateway (‘home’ in this case) , not the machine from where the tunnel is initiated. So this will make a connection to

port 5900 of the ‘home’ computer where the VNC client would be listening in.

The created tunnel can be used to transfer all kinds of data not limited to web browsing sessions. We can also tunnel SSH sessions from this as well. Let’s assume there is another computer (‘banned’) to which we need to SSH from within University but the SSH access is being blocked. It is possible to tunnel a SSH session to this host using a local port forward. The setup would look like this.



(https://chamibuddhika.files.wordpress.com/2012/03/sshsessionforwarding.jpg)

As can be seen now the transferred data between ‘work’ and ‘banned’ are encrypted end to end. For this we need to create a local port forward as follows.

```
1 ssh -L 9001:banned:22 home
```

Now we need to create a SSH session to local port 9001 from where the session will get tunneled to ‘banned’ via ‘home’ computer.

```
1 ssh -p 9001 localhost
```

With that let’s move on to next type of SSH tunnelling method, reverse tunnelling.

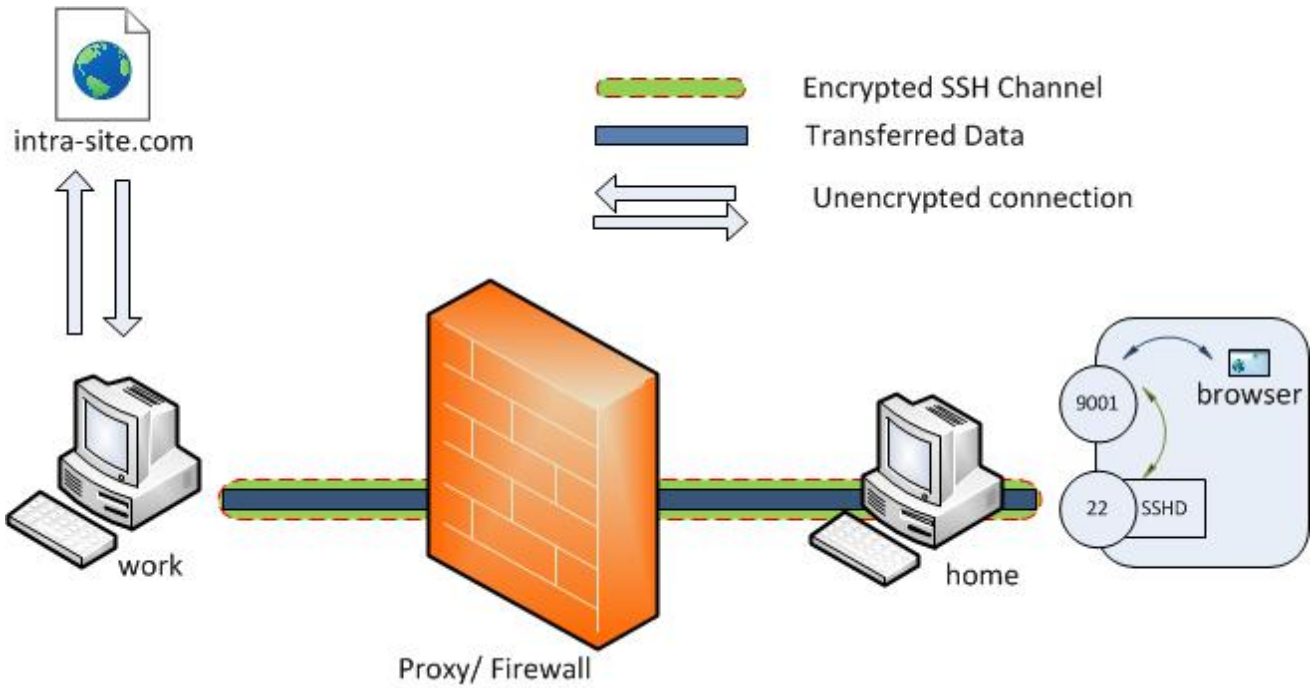
Reverse Tunnelling with remote port forwarding

Let’s say it is required to connect to an internal university website from home.
The university firewall is blocking all incoming traffic. How can we connect from ‘home’ to internal network so that we can browse the internal site? A VPN setup is a good candidate here. However for this example let’s assume we don’t have this facility. Enter SSH reverse tunnelling..

As in the earlier case we will initiate the tunnel from ‘work’ computer behind the firewall. This is possible since only incoming traffic is blocking and outgoing traffic is allowed. However instead of the earlier case the client will now be at the ‘home’ computer. Instead of -L option we now define -R which specifies
a reverse tunnel need to be created.

```
ssh -R 9001:intra-site.com:80 home (Executed from 'work')
```

Once executed the SSH client at ‘work’ will connect to SSH server running at home creating a SSH channel. Then the server will bind port 9001 on ‘home’ machine to listen for incoming requests which would subsequently be routed through the created SSH channel between ‘home’ and ‘work’. Now it’s possible to browse the internal site
by visiting http://localhost:9001 (http://localhost:9001) in ‘home’ web browser. The ‘work’ will then create a connection to intra-site and relay back the response to ‘home’ via the created SSH channel.



(https://chamibuddhika.files.wordpress.com/2012/03/remoteportforwarding.jpg)

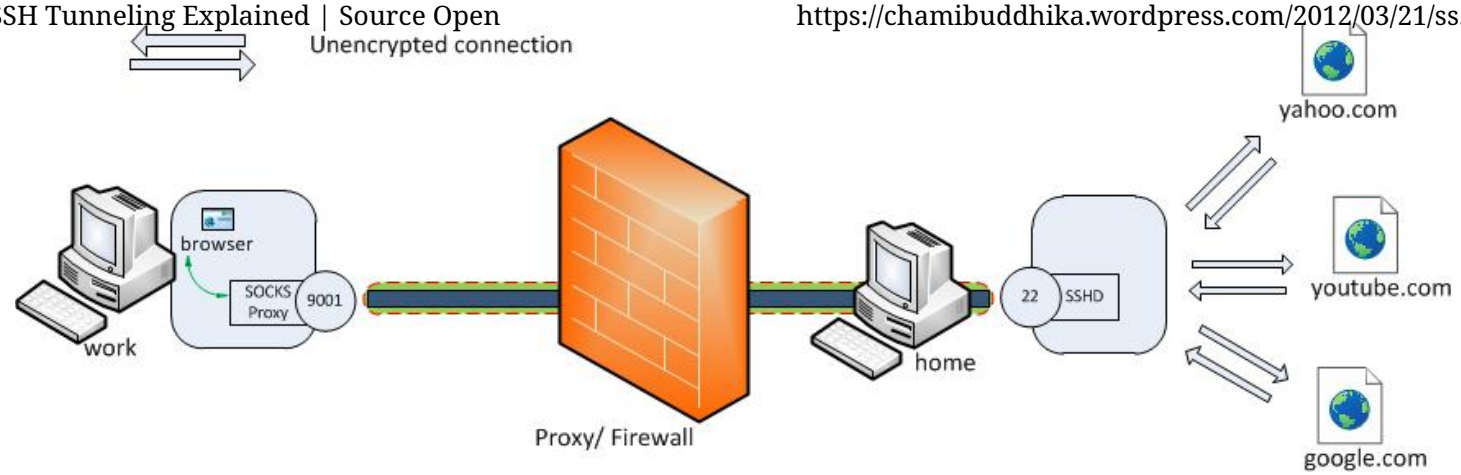
As nice all of these would be still you need to create another tunnel if you need to connect to another site in both cases. Wouldn’t it be nice if it is possible to proxy traffic to any site using the SSH channel created? That’s what dynamic port forwarding is all about.

Dynamic Port Forwarding

Dynamic port forwarding allows to configure one local port for tunnelling data to all remote destinations. However to utilize this the client application connecting to local port should send their traffic using the SOCKS protocol. At the client side of the tunnel a SOCKS proxy would be created and the application (eg. browser) uses the SOCKS protocol to specify where the traffic should be sent when it leaves the other end of the ssh tunnel.

```
ssh -D 9001 home (Executed from 'work')
```

Here SSH will create a SOCKS proxy listening in for connections at local port 9001 and upon receiving a request would route the traffic via SSH channel created between ‘work’ and ‘home’. For this it is required to configure the browser to point to the SOCKS proxy at port 9001 at localhost.



(https://chamibuddhika.files.wordpress.com/2012/03/dynamicportforwarding.jpg)

Posted in Concepts, Security | Tagged Port forwarding, SSH, tunnelling | 37 Comments

37 Responses

Tunneling SSH – transfert de ports — Bruno ARLIGUY *on March 23, 2012 at 12:21 pm | Reply*
[…] Le but de ce document est d'expliquer les tunnels SSH. C'est à dire comment utiliser SSH pour faire passer différents protocoles, ce qui permet de sécuriser la communication (une sorte de VPN software). Si vous souhaitez plus de détails sur les différentes possibilités, cet article de Buddhika Chamith vous éclairera : SSH Tunneling Explained. […]

Dica: Acessando ssh através de um proxy autenticado « O Futuro é a Liberdade *on April 9, 2012 at 12:40 pm | Reply*


[…] SSH Tunnelling Explained (chamibuddhika.wordpress.com) […]

raghavan *on May 28, 2012 at 3:09 pm | Reply*
very well explained. well done.

Buddhika Chamith *on May 28, 2012 at 11:41 pm | Reply*

Thanks. Glad that you found it useful. :).

Emmanuel *on June 8, 2012 at 4:29 pm | Reply*
Excellent article on port forwarding

Buddhika Chamith *on June 8, 2012 at 9:05 pm | Reply*

Thanks.

Nick *on July 21, 2012 at 12:07 am | Reply*
This is a very detailed document.. Since I am new to this technology.can anyone please explain me where are the commands (ssh -L 5900:localhost:5900 home (Executed from 'work')

) executed.Is there a software in which the commands are executed..????

Buddhika Chamith *on July 26, 2012 at 11:24 am* | *Reply*

You need to install SSH server and client in order to follow this tutorial. Specially in this case SSH client at 'work' and SSH server in 'home'. http://www.cyberciti.biz/faq/ubuntu-linux-openssh-server-installation-and-configuration/ may be of help to you if you are on an Ubuntu distribution. Or else you can try installing Putty. HTH.

root_taker *on July 25, 2012 at 4:03 pm* | *Reply*

Great tutorial! I have a similar tutorial on my blog that explains things similarly for windows users who would be using putty: http://technologyordie.com/ssh-tunneling-and-proxying

Buddhika Chamith *on August 2, 2012 at 2:02 am* | *Reply*

Thanks. Nice work on your part too. :).

Bjarne Valle *on August 2, 2012 at 1:03 am* | *Reply*

One of the best, and easy understandable article about ssh Tunneling I have come across.

Buddhika Chamith *on August 2, 2012 at 1:58 am* | *Reply*

Thanks Bjarne!! Glad you found it useful.

David AMAR *on August 22, 2012 at 7:45 am* | *Reply*

Very clear and useful! Thanks a lot

Umapathi Anand *on September 2, 2012 at 12:57 am* | *Reply*

Very Good Stuff and Style. The World Expects more contribution from You.

Buddhika Chamith *on September 2, 2012 at 3:10 am* | *Reply*

Thanks Anand.

simpleissweet *on September 20, 2012 at 8:06 am* | *Reply*

This is a fantastic article… saved me tons of time from googling for answers and worked like a charm!

nodmonkey *on October 13, 2012 at 8:34 am* | *Reply*

Clearly explained, good examples, well illustrated. The Internet was missing this resource and now we have it. Great!

One thing I would suggest adding is the syntax of reverse and dynamic tunnelling using angle brackets, as you did for the first local tunnelling example.

Ricardo Murillo *on November 6, 2012 at 9:15 am* | *Reply*

Excelente explanation, this is the first time I read about SSH tunnelling but I know understand it clearly.

> **Ricardo Murillo** *on November 6, 2012 at 9:16 am* | *Reply*
>
> (Correcting previous post) Excellent explanation, this is the first time I read about SSH tunnelling but now I understand it clearly.

Johannes *on November 11, 2012 at 8:28 am* | *Reply*

Hey,
Thx for your effort! Really helpful. Just on thing:
You do describe the following example:
ssh -L 5900:localhost:5900 home (Executed from ‘work’)

There is the sentence:
“As explained earlier is relative to the , not the machine from where the tunnel is initiated”
You seem to have forgotten a word or sentence after “relative to the…”?!

Could you tell to which it is relative to?

Thanks,
Johannes

> **Buddhika Chamith** *on November 11, 2012 at 8:58 am* | *Reply*
>
> 
>
> Hi Johannes,
>
> Yes there is a slight omission. Thanks for pointing it out. It has been fixed now. It should be the gateway computer. (‘home’ in that particular case).

RichardB *on November 30, 2012 at 7:55 am* | *Reply*

A really well explained topic. Thanks
I aim to play with a socks proxy to really get my understanding with this nailed.

RichardB *on November 30, 2012 at 8:01 am* | *Reply*

BTW, I note the tunnel remains and accepts new connections when I used the -L option even if I kill the original command that created the tunnel. netstat says the local port is now being owned by my sshd daemon. Is that normal behavior?
my ssh command was for mysql client forwarding to a remote server which had a local access only mysql server…

ssh root@192.168.0.36 -L 3306:127.0.0.1:3306 -N

RichardB *on November 30, 2012 at 8:05 am* | *Reply*

Ahh, my bad I had also tried the -R example from the remote server – which of course had connected and created a tunnel to my local ssh daemon – sorry for the confusion.

Mohammad Adnan *on December 11, 2012 at 12:31 am* | *Reply*
Great Explanation

pushparajxap *on January 16, 2013 at 4:55 pm* | *Reply*
Very will explained…It cleared all my doubts..in fact it made me to realize to write blogs on concepts which I know well, so that it will be useful for others….Just like what you posted here is useful for me..

Thanx a lot..

Pushparaj

kentgrav *on January 25, 2013 at 3:26 pm* | *Reply*
I have been trying to do this for about 8 days now and have looked at virtually every single article there is that explains this process. And this is by far the best I have found, thanks for posting it and contributing back to the community.

Ruud *on February 2, 2013 at 6:39 am* | *Reply*
Nice, clear and compact article. Thanks !

Nitin *on February 6, 2013 at 1:14 am* | *Reply*
Crisp and clear.

Simple Quick SSH Tunneling to expose your web app | *on July 12, 2013 at 4:56 am* | *Reply*
Sid__
[…] Before we move to the “how to” here is a quick intro to ssh tunneling […]

Secure Two-Machine Maya commandPort, with SSH *on March 1, 2014 at 8:58 pm* | *Reply*
Tunneling | CG Tech by Jonathan Rice
[…] a port like this, however, we can set up a more secure two-machine connection by making use of ssh tunneling. To be specific, we will use local port forwarding, so that Maya and the remote application each […]

Harro *on March 5, 2014 at 9:04 pm* | *Reply*
Thanks for covering all the scenarios. More here….
http://wiki.glitchdata.com/index.php?title=SSH:_Port_Tunnelling

Andy Crofts *on March 8, 2014 at 5:37 pm* | *Reply*
Best, clearest desciption ever. Thanks – exactly what I was searching for.

SSH Tunneling | Derek Provance *on March 21, 2014 at 3:37 pm* | *Reply*

[...] that I have found have proven to only confuse and frustrate me more. Fortunately, I stumbled upon https://chamibuddhika.wordpress.com/2012/03/21/ssh-tunnelling-explained/ which explained the how in setting up an SSH [...]

Mr. Moderate Misbehaviour *on October 1, 2014 at 10:48 am* | *Reply*

Dude thank you so much, was driving me nuts trying to wrap my head around this but your explanation was perfect!

Yarik *on June 20, 2017 at 9:18 am* | *Reply*

Perfect! Thank you so much for ideal explanation

Joel Stevick *on September 4, 2018 at 11:14 am* | *Reply*

Well done, thank you for you explanation. The vital piece of information that really nailed-it for me was that the host is “relative to the gateway”

Comments RSS

WPThemes.