# Remote Desktop Support With VNC

VNC is a well-known tool for remote desktop view and control. The two computers establish a TCP connection so that one of them can access the display of the other. However, almost always these two computers are behind a firewall/router and do not have a real IP to be accessed from the Internet. In such a case one of them can do port forwarding on the router/firewall and this would allow the connection to be established. The problem is that more often than not, none of the parts that want to establish such a remote connection have access to the firewall that separates/protects it from the real internet. Sometimes it can even be several levels deep behind the firewall (several layers of firewalls). However, if you have access to an external server (for example a server in the cloud) there is still a workaround and things can be fixed to work. Here I will explain the tricks and hacks that can be used in such a case in order to establish a secure connection to a remote desktop.

## The Situation

Let's say that we have a computer **A** that wants to connect to the desktop of a computer **B**. Both of them are behind a NAT router/firewall and cannot modify its settings. However both of them can access a third computer **C** that is somewhere on the cloud and has a real IP.

It is possible to use **C** as a redirecter of the traffic, so that both computers **A** and **B** can establish a direct (peer-to-peer) connection between them. This is done using the wonderful features for tunneling and port-forwarding of the SSHprotocol. What is more, this connection will be encrypted and secure, which is the main feature of the SSH protocol.

I will describe in the other sections what should be done in each of the computers in order to setup such a connection.

## Basic Setup

On **A** we need to have a VNC client. Let's install **vncviewer**:

```
sudo apt-get install vncviewer
```

On **B** we need to have a VNC server. Let's install **x11vnc**:

```
sudo apt-get install x11vnc
```

To connect from **A** to **C** and from **B** to **C** we can use any existing account on **C**. However it is better to create a new one. Let's call it **vnc**:

```
sudo adduser vnc
```

Let's assume that there is a **sshd** server running on **C** (otherwise there is no way to access it), and let's say that it is running on the port **2201**. We will refer to it as **middle-server** (which can be an IP or a domain name).

To access the desktop of **B** from **A**, we first start the VNC server on **B**:

```
ssh -p 2201 -t -R 5933:localhost:5900 vnc@middle-server
x11vnc -rfbport 5900 -localhost
```

Then we start the VNC viewer on **A**:

```
ssh -p 2201 -t -L 5900:localhost:5933 vnc@middle-server
vncviewer -encodings "copyrect tight zrle hextile" localhost:0
```

**Note:** In both cases above, the **ssh** command and the other command are given on separate terminal tabs. The **ssh**command creates a ssh tunnel to **C** for the VNC port 5900, which is mapped to the port 5933 on **C**.

Now we can enjoy the access to the remote desktop.

## Using A Private-Public Key Pair For The Ssh Connections

The **SSH** protocol allows authentication by a private-publik key pair. This would be more convenient because this way we don't have to remember the passsword. Also we can restrict the ssh access only for what we need it: tunneling and port-forwarding, and prohibit login etc.

Let's create a private-public key pair for user `vnc` on **C**:

```
su - vnc
mkdir ~/.ssh
chmod 700 ~/.ssh
ssh-keygen -t rsa

ls -al .ssh
total 16
drwx------ 2 vnc vnc 4096 Jan 22 09:31 .
drwxr-xr-x 3 vnc vnc 4096 Jan 22 09:31 ..
-rw------- 1 vnc vnc 1675 Jan 22 09:31 id_rsa
-rw-r--r-- 1 vnc vnc  394 Jan 22 09:31 id_rsa.pub
```

The public key `id_rsa.pub` should go to the file `authorized_keys`, and the private key `id_rsa` will be used by **A** and **B** to login:

```
cd ~/.ssh/
cp id_rsa.pub authorized_keys
cp id_rsa vnc.key
```

Now we can copy `vnc.key` to **A** and **B** and then run the commands like this:

```
On B:
ssh -p 2201 -t -R 5933:localhost:5900 -i vnc.key vnc@middle-server
x11vnc -rfbport 5900 -localhost

On A:
ssh -p 2201 -t -L 5900:localhost:5933 -i vnc.key vnc@middle-server
vncviewer -encodings "copyrect tight zrle hextile" localhost:0
```

This time no password will be asked for establishing the ssh connection.

## Making It More Secure

When we run the **ssh** command, by default it opens a shell on **C** where you can run commands, besides creating a tunnel and doing port-forwarding. Even if the **vnc** account that we use for this purpose may have limited access rights on **C**, this still is a potential threat (security problem), especially if we don't know or cannot trust the person that is sharing the desktop.

Fortunately, it is possible to limit the SSH features that each key can use. We do this by modifying the file `authorized_keys`to look like this:

```
command="/bin/sleep 4294967295",no-agent-forwarding,no-user-rc,
no-X11-forwarding,permitopen="localhost:5933" ssh-rsa B3NzaC1...
```

The long list of no-xyz statements disallow it from doing just about anything except connect to a VNC server.

## Referencies


- http://blog.markloiseau.com/2012/03/vnc-current-session/
- https://help.ubuntu.com/community/VNC
- http://www.karlrunge.com/x11vnc/faq.html



Author: Dashamir Hoxha <dashohoxha@gmail.com>


Date: 22 January 2014
HTML generated by org-mode 6.33x in emacs 23