Tricks with VNC

Virtual Network Computing (VNC), originally from the AT&T research lab is a handy remote access platform with many more uses than most people know. This is a brief howto on some of the more advanced, interesting features of VNC. For basic usage, see the man pages or the end of this article for a list of references.

Make sure you have VNC.

The first step is to get VNC installed on your system. Most distributions ship, many have it installed by default.

To see if it's installed, type:

```
$which vncserver
```

If it returns something like "/usr/bin/vncserver", your good, if it returns nothing, you will need to get VNC installed on your system.

As VNC is an open source application, there are many versions available. All are supposed to be compatible with each other. You can get the true, or original, program from http://www.realvnc.com/download.html . I actually prefer the TightVNC version from http://www.tightvnc.com/download.html .

TightVNC has a different form of compression called the Tight encoding algorithm that results in much better performance over slow connections, such as a modem.

From the RealVNC sight you can get RedHat 7.3 RPMs or source code tarballs. From TightVNC, they claim RedHat 7.x RPMs and again source tarballs. My recommendation is to go to http://rpmfind.net if you are running an RPM-based distro (e.g. RedHat, Mandrake, SuSE) and get your packages there. You technically need only the vncserver package, but you will most likely want the vncclient as well.

If you are running a DPKG-based distro (e.g. Debian, Knoppix, LibraNet, Lindows, Lycoris), it is even easier: (as root)

```
$apt-get update
$apt-get install vncserver \
>xvncviewer vnc-java
```

If you want to use TightVNC instead, the package names are:

```
tightvncserver xtightvncviewer tightvnc-java
```

I believe the Tight versions are only available in Sarge (testing) or higher.

To install the rpm:

```
rpm -Uvh vncserver-blah.blah.rpm
rpm -Uvh vncviewer-blah.blah.rpm
```

To see if it's installed correctly, open a terminal (not as root) and type:

```
$vncserver
```

You should get something like the following response:

```
You will require a password to access your desktops.

Password:
Verify:

New 'X' desktop is xian:1

Starting applications specified in /etc/X11/Xsession
Log file is /home/john/.vnc/xian:1.log
```

Note: It may not be exactly like this. That is probably fine.

To kill your test server, use:

```
vncserver -kill :1
```

XDMCP

XDMCP is the display protocol often used to connect remote X-Servers to an X-client (typically on a server).

I know it gets a little confusing but it must be understood; an X-client is typically a program that, when run, will display it' s output to an X-Server. The two are often run on the same machine but that need not be the case. When the two are split across a network, the X-client (the application) almost always resides on a server, whereas the X-Server (the display software that actually controls the video card) almost always runs on the client (the client from a network point of view, that is).

There are a multitude of security and administrative issues with running an XDMCP server that I won' t get into. Suffice it to say, there are plenty of reasons NOT to do it. Especially over a public network such as the Internet. That' s where VNC comes in.

VNC actually has the ability to work directly with an XDMCP server on your machine. "Where do I get an XDMCP server?", your thinking. You probably already have one. Do you use KDE? Gnome? Some other Window Manager? If, when you boot your Linux (this actually applies to almost any UNIX-based system, but not Mac OS X) box up, it greets you with some sort of graphical login, you almost certainly have an XDMCP server. XDM, GDM, and KDM are the big three.

If you don' t know which one you are running, open a terminal window and type:

```
$ps ax | grep '.dm\b'
```

You should see an instance of one of them running. Note that on some systems (such as Debian), the display manager may be called something like ' ' gdm-binary' ' .

Configuring KDM

For KDM, find your kdmrc file. Maybe in ' ' /etc/X11/kdm/kdmrc' ' , but maybe not. If you can' t find it try (as root):

```
        find / | grep 'kdmrc'
```

In the ' ' XDMCP' ' section of kdmrc, edit (or add) the line:

```
        Enable=false
```

to:

```
        Enable=true
```

You may also have to change the port from ' ' 0' ' to ' ' 177' '

For KDE, you also need to follow the instructions in the following section ' ' Configuring XDM' '

Configuring XDM

You can skip over this section if you are using GDM. With KDM you may need to edit these files, or you may not. If you have them, go ahead and make the changes. If you have a different directory, such as ' ' /etc/kde/kdm' ' or ' ' /opt/kde/share/config/kdm' ' that are used by recent versions of KDE, you will use those instead.

First we need to edit the Xaccess file. It might be at ' ' /etc/X11/xdm/Xaccess' ' .

If you can' t find the files to edit, as root, try:

```
$find / | grep 'Xaccess'
```

If it doesn' t find the file, you probably don' t have XDM installed. The other configuration files should be in the same directory as Xaccess.

This file is used to control access to the X-Server. Since we will be running a VNC server on the local host, we only need one line in the file:

```
        localhost  #allow connections only from this box
```

If you want to allow any host to connect directly to the X-Server (NOT with VNC), you can use the following entry:

```
        * # allow anyone to connect (have a firewall??)
```

Note: the line is just an asterisk, everything after the # is a comment.

Next, we need to configure the xdm-config file. Make sure you have a line like the following:

```
        DisplayManager.requestPort:  177
```

Note that on many systems, you will normally have this line set to 0 to disable remote logins via X. Often you can simply comment this line out and XDM will listen to UDP port 177 by default. We just say it explicitly.

Also note: If you have a very tight firewall on this machine, you may need to open up the port for the localhost. See the section ' ' Firewall Settings' ' for info on doing just that.

In order for these changes to take effect you must restart your XDM server.

*****Warning: This will kill any active X sessions you are running. Save your work if you need to *****

On Debian et al (as root):

```
        /etc/init.d/xdm restart
```

On RedHat et al (as root):

```
service xdm restart
```

If you are using KDM instead make sure you restart kdm.

Configuring GDM

GDM is the easiest to configure. You need to edit the file gdm.conf (probably in ' ' /etc/X11/gdm/gdm.conf' ' . There is a section called [xdmcp]. Change:

```
Enable=false
```

to:

```
Enable=true
```

Then restart gdm:

Debian et al:

```
/etc/init.d/gdm restart
```

RedHat et al:

```
service gdm restart
```

To control access to the X-Server you would use hosts.allow and hosts.deny (tcp wrappers) instead of Xaccess.

Configuring VNC

For our purpose here, we will start the vncserver from the inetd superserver. That gives us some advantages and some disadvantages. The main advantage is resources. Instead of constantly running a vncserver for each user who needs one, each server will be started on the fly when it is needed. It will also function just like an X-Terminal, in that when a connection is terminated, the VNC server will be shut down. This is also the disadvantage as well, though; if you are used to using VNC with stateful sessions (when you login again, you pick up right where you left off), this will be different. Each time you login it will be a completely different session.

/etc/services

To configure multiple servers with different resolutions, you can add multiple lines to /etc/services. Add the following lines (you can put them wherever you want in the file):

```
# VNC Servers
vnc-svga            5900/tcp
vnc-xga             5901/tcp
```

xinetd

Some distros use xinetd instead of inetd. If you are one, you need to create a file called ' ' /etc/xinetd.d/vnc-svga' ' with the following contents:

```
service vnc-svga
{
  disable          = no
```

```
    socket_type      = stream
    protocol         = tcp
    wait             = no
    user             = nobody
    server           = /usr/bin/Xvnc
    server_args      = :1 -inetd -query localhost \
      -geometry 800x600 -depth 24 -once
  }
```

Note: If you have trouble with fonts in your VNC sessions try the following fix. Read your /etc/X11/XF86Config file and look at the ' ' Files' ' section. Add all of your FontPath directories to the server_args section with the argument -fp. Mine looks like this:

```
  ... removed for brevity
    server_args      = :1 -inetd -query localhost \
      -geometry 800x600 -depth 24 -once -fp \
           /usr/lib/X11/fonts/misc \
           /usr/lib/X11/fonts/75dpi \
           /usr/lib/X11/fonts/Type1 \
           /usr/lib/X11/fonts/TrueType \
           /usr/share/fonts/truetype \
           /usr/share/fonts/truetype/openoffice \
  }
```

For RedHat, I think you can use:

```
  ... removed for brevity
    server_args      = :1 -inetd -query localhost \
      -geometry 800x600 -depth 24 -once -fp \
           unix/:7100
  }
```

Also, for RedHat and similar, you may need to symlink Xvnc or else use a different path in the inetd config file:

```
  ln -s /etc/X11R6/bin/Xvnc /usr/bin/
```

You should do an entry like above for each line you put in /etc/services.

inetd

For inetd you would add the following line to /etc/inetd.conf

```
vnc-svga stream tcp nowait nobody /usr/sbin/tcpd \
         /usr/bin/Xvnc :1 -inetd -query localhost \
         -geometry 800x600 -depth 24 -once
```

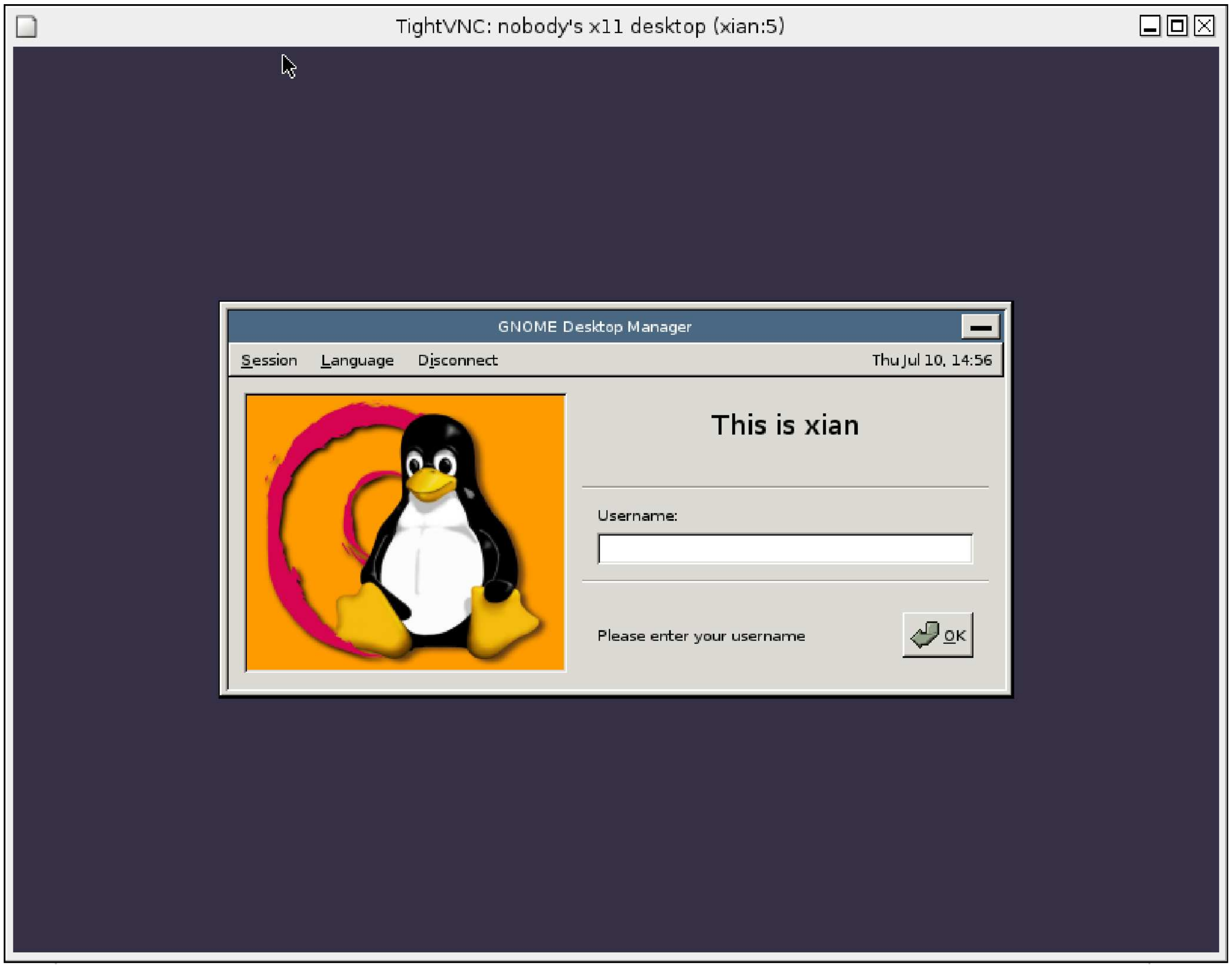
Note that I split this across multiple lines so it would fit on the page. In the actual file it should NOT be split. Also, for RedHat and similar, you may need to symlink Xvnc or else use a different path in the inetd config file:

```
        ln -s /etc/X11R6/bin/Xvnc /usr/bin/
```

Now you need to restart inetd or xinetd:

```
        killall -HUP inetd
        killall -HUP xinetd
```

At this point, you should be able to login with a VNC client and get a graphical login screen. You won' t need to enter a password in VNC, because your display manager will prompt you for one anyway.



Security

There are a few security concerns to be aware of with this setup.

VNC

VNC does not encrypt data. There is some form of encryption during the password exchange, but none after. This especially presents a problem for our setup, because we are logging in after the VNC session is started and thereby sending our password completely in the clear. If you are connecting from a machine with SSH, it is very easy to overcome this limitation. Let' s say you are connecting to myserver.yaklug.org Issue the following command on the computer you are connecting from:

```
        $ssh -L 5950:myserver.yaklug.org:5900 myserver.yaklug.org
```

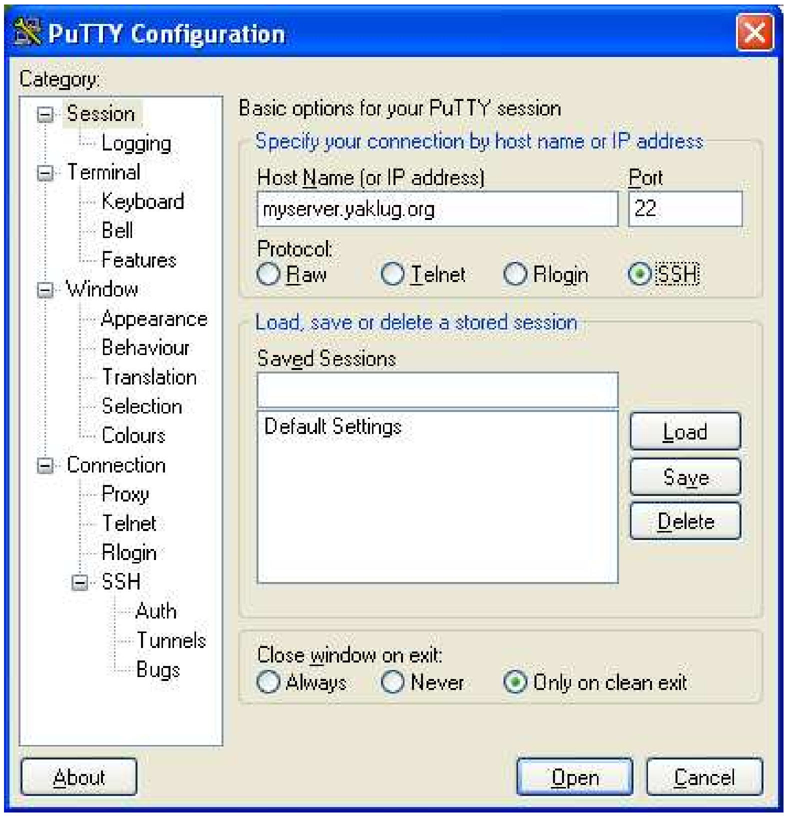
Then, in your VNC client use the local port you specified instead of the remote machine. e.g. you would connect to:

```
        127.0.0.1:5950
```

That would redirect you through a secure tunnel to myserver.yaklug.org:5900 (or display :1. 5901 would be display :2, etc.)

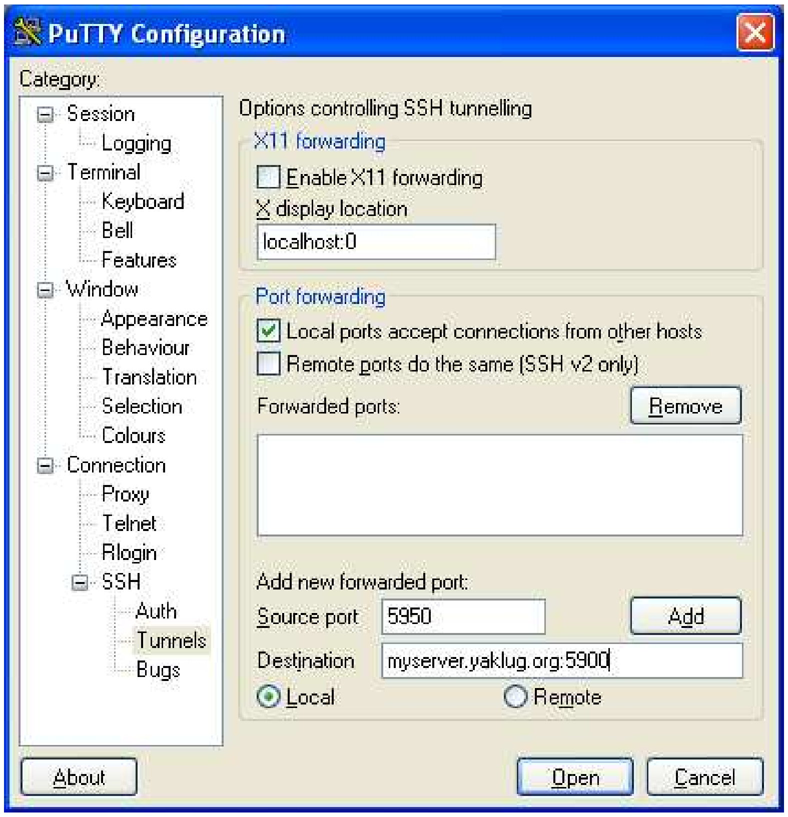
With Windows, you can use Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty) to do the same thing. The same example would look like this:

Setup a session to your server:



Host Name (or IP address): myserver.yaklug.org
Port: 22
Protocol: Raw Telnet Rlogin [✓ SSH]
Close window on exit: Always Never [✓ Only on clean exit]

Next, setup the tunnel with the example below.



Enable X11 forwarding: ☐

X display location: localhost:0

Local ports accept connections from other hosts: ☑

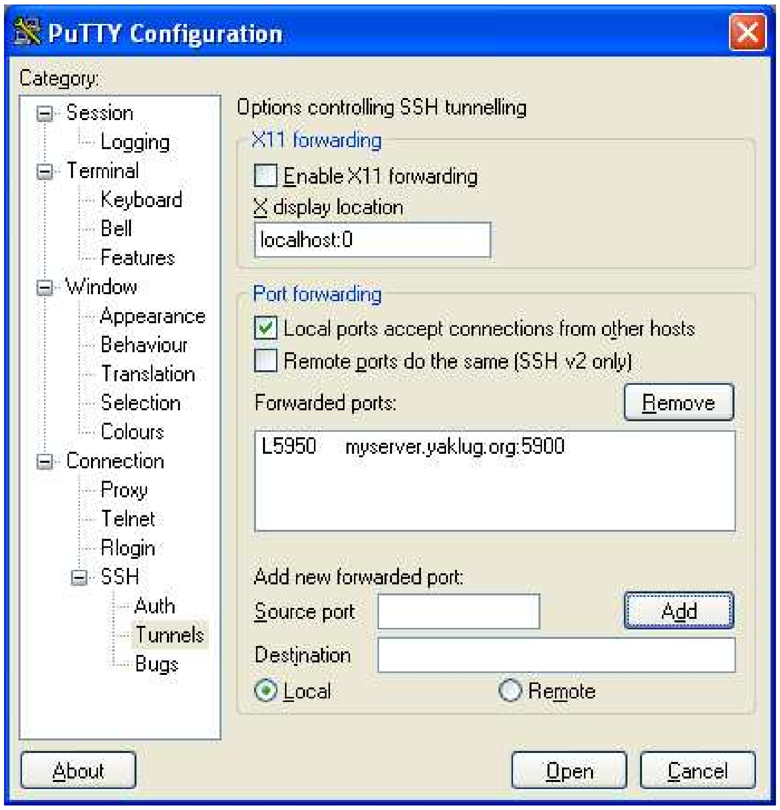
Remote ports do the same (SSH v2 only): ☐

Source port: 5950

Destination: myserver.yaklug.org:5900

[✓ Local] Remote

After you press the ' ' Add' ' Button, It should look like this:



Enable X11 forwarding: ☐

X display location: localhost:0

Local ports accept connections from other hosts: ☑

Remote ports do the same (SSH v2 only): ☐

Forwarded ports:

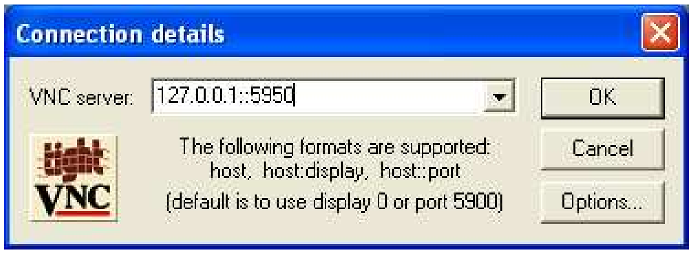
```
L5950     myserver.yaklug.org:5900
```

Source port:

Destination:

[✓ Local] Remote

When you connect with VNC, specify the local machine and port instead of the remote machine. The following example is with the TightVNC Viewer:



VNC server: 127.0.0.1::5950

The following formats are supported:
host, host:display, host::port
(default is to use display 0 or port 5900)

Notice that the IP address and port number are separated by two colons, not one as you may be accustomed to. This is because with X (and VNC) displays are specified as server:display. If you only use one colon, the server thinks you want display 5950 on server 127.0.0.1!

That' s all there is to it!

Firewalls

If you want to make sure you allow only encrypted VNC sessions to your box, you could start with a ruleset like this:

```
IPT=/sbin/iptables
# Drop everything to XDMCP port not from localhost
$IPT -append INPUT -source ! 127.0.0.1 -proto udp \
         --dport 177 -j DROP
# Drop everything for VNC not from localhost
$IPT -append INPUT -source ! 127.0.0.1 -proto tcp \
         --dport 5900:5910 -j DROP
```

A better way to lock it down would be to deny everything and allow only what you want:

```
IPT=/sbin/iptables
# Set Policies
$IPT -policy INPUT DROP     # Drop everything by default
$IPT -policy OUTPUT ACCEPT # Outgoing packets ok
$IPT -policy FORWARD DROP # Don't route packets

# a stateful catchall chain
$IPT -new stateful # create the new chain
# Accept anything related to a current connection
$IPT -append stateful -m state -state \
         ESTABLISHED,RELATED -j ACCEPT
# Block anything else
$IPT -append stateful -m state -state NEW -j DROP
```

```
# What to allow in
# Allow anything from the localhost
$IPT -append INPUT -source 127.0.0.1 -j ACCEPT
# ICMP Stuff
$IPT -append INPUT -proto icmp -icmp-type echo-request \
         -j ACCEPT  # Allow pings
$IPT -append INPUT -proto icmp -icmp-type host-unreachable \
         -j ACCEPT  # Allow MTU Path Discovery
$IPT -append INPUT -proto tcp -dport ssh -j ACCEPT # ssh
# Now we send everything else to the stateful chain to see
# if it's part of an existing connection, otherwise drop it
$IPT -append INPUT -j stateful

# That's all we need!  Since everything is blocked by default,
# no one can get directly to either XDMCP or VNC.  Only ssh
# can be used.  Since ssh redirects the connection through the
# tunnel to “localhost”, there is no need to open up any other
# ports and everything will still work.
```

Miscellany

There are other ways to secure the connection as well. Some ways that come to mind are Zebedee. A cross-platform tunnel program for Linux and Windows. There is even a VNC client that has Zebedee built into it called Zvnc.   I played around with this set up, but frankly found Putty with SSH much simpler to setup and just as easy to use.

You could also run VPN software on your server (or a firewall in front of it, etc.) such as FreeS/WAN, PoPToP, or lt2pd. These can all be made to work but require MUCH more work to setup.