2014

Hands On Labs+

Debian and RPM Distributions

And Secure VNC

Table of Contents

Introduction

Security is a commonly overlooked topic of conversation when discussing Linux servers. Although it is not uncommon to talk about hardening the server itself, client access to the server can sometimes be an afterthought. SSH is by far the most common and secure method of accessing your server, but there will be instances where you may have to run a full desktop and do not have access to the terminal and tunneling X over SSH will not be enough.

We will talk about the security around accessing a remote desktop session running on your server and how to protect that access on your client.

Goals

This Hands On Lab will show you how to set up a remote VNC connection to your Debian (Ubuntu, Cinnamon or Plain Debian) or RPM based (CentOS, Red Hat) Linux distributions. The two methods differ in the server configuration, although securing the connection locally and remotely are the same. Where differences exist, they will be clearly noted.

Although we will talk about getting X Windows running on your system, we will cover that portion at the most basic level (i.e. just enough to get an X Windows desktop running to connect to over VNC).

PLEASE NOTE: The lab servers you will use are Ubuntu (Debian based) distributions. The RPM (Red Hat or CentOS) instructions are included as a bonus reference but do not apply to the hands on portion of this lab.

Packages, Resources and Prerequisites

- Tigervncserver (RPM Based Distributions)
- Vnc4server (Debian Based Distributions)
- MobaXTerm (Windows SSH Client w/ VNC Client Built In)
    - http://mobaxterm.mobatek.net/ - Price: $0

The resources you will be accessing during the course of this lab are:

- Ubuntu 13.10 Server: Test Client
    - You will use this to test your VNC server over SSH once completed

Prerequisites to this lab:

- A LinuxAcademy.com Lab+ Subscription
- Internet Access and SSH Client

    - You will need to connect to the public IP of the server in order to configure VNC, the only method of connectivity is over SSH
        - SSH client can be Windows (i.e. Putty) or from another Linux system shell
    - For Windows VNC Client configuration and testing, use the MobaXTerm client referenced above
- Login Information (Provided When The Server Starts Up)

Document Conventions

Just a couple of housekeeping items to go over so you can get the most out of this Hands On Lab without worrying about how to interpret the contents of the document.

When we are demonstrating a command that you are going to type in your shell while connected to a server, you will see a box with a dark blue background and some text, like this:

```
linuxacademy@ip-10-0-0-0:~$ sudo apt-get install package
[sudo] password for linuxacademy: <PASSWORD PROVIDED>
```

That breaks down as follows:

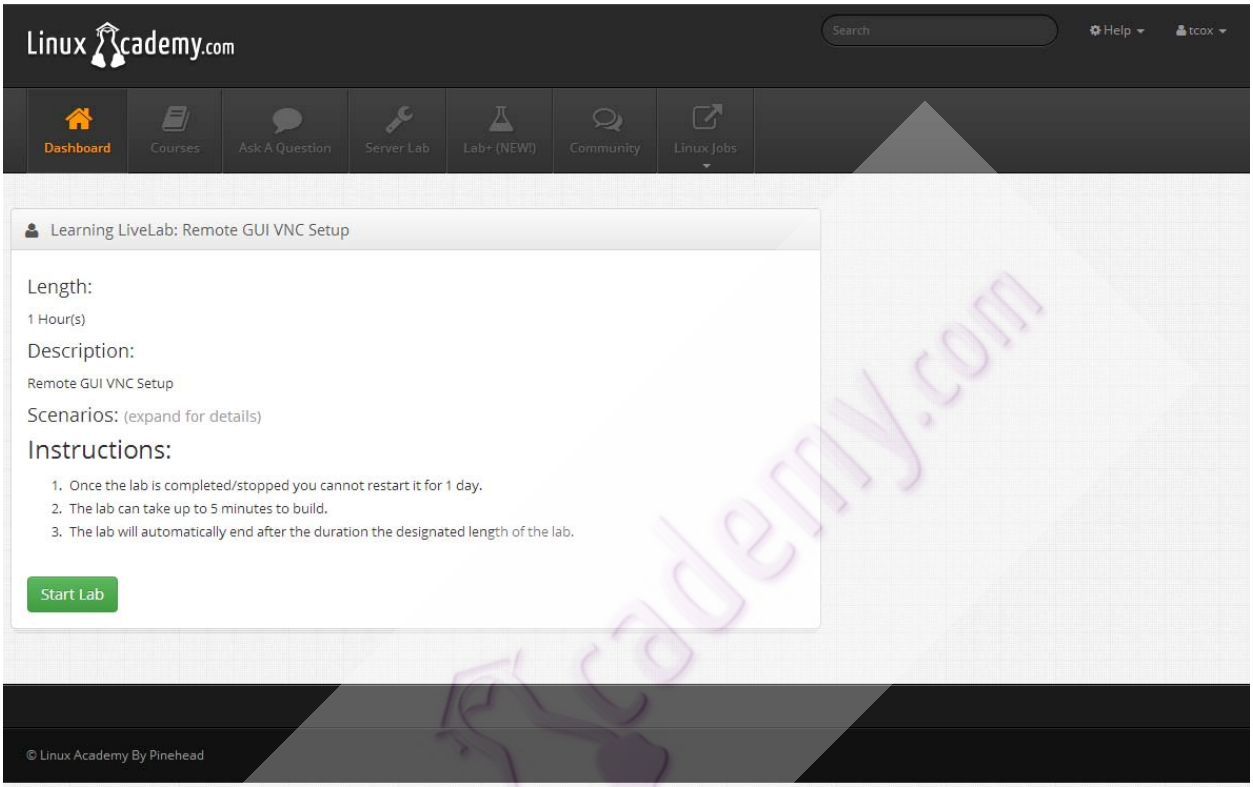
- The white text that looks something like "linuxacademy@ip-10-0-0-0:~$: "should be interpreted as the console prompt your cursor will be sitting at when you are logged into the server. You are not typing that into the console, it is just there. Note that the portion that says "ip-10-0-0-0" will be different for you based on the IP address of your system.
- The bold yellow text in any command example is the actual command you will type in your shell.
- Any lines subsequent to the bold yellow command that you type in is to be considered as the response you can expect from your command entry.

If you do not see the command prompt as the first line in the example, then all the white text is an example of a text, script or configuration file and is intended to be typed in its entirety (in the event you are instructed to create a previously non-existent file) or to compare against the contents of the file on your system.

One final convention, if you see a "~" at the end of a line in a command or text example, that will indicate that the line overflowed the end of line. Meaning you should just keep typing without hitting enter to start a new line until the natural end of the command.

General Process

When you are ready to begin the Hands On Lab, log into your Linux Academy Lab+ subscription and navigate to the "Live Labs" section on the Dashboard. Once you choose the "Secure VNC" Lab from the list, you will see the screen below:

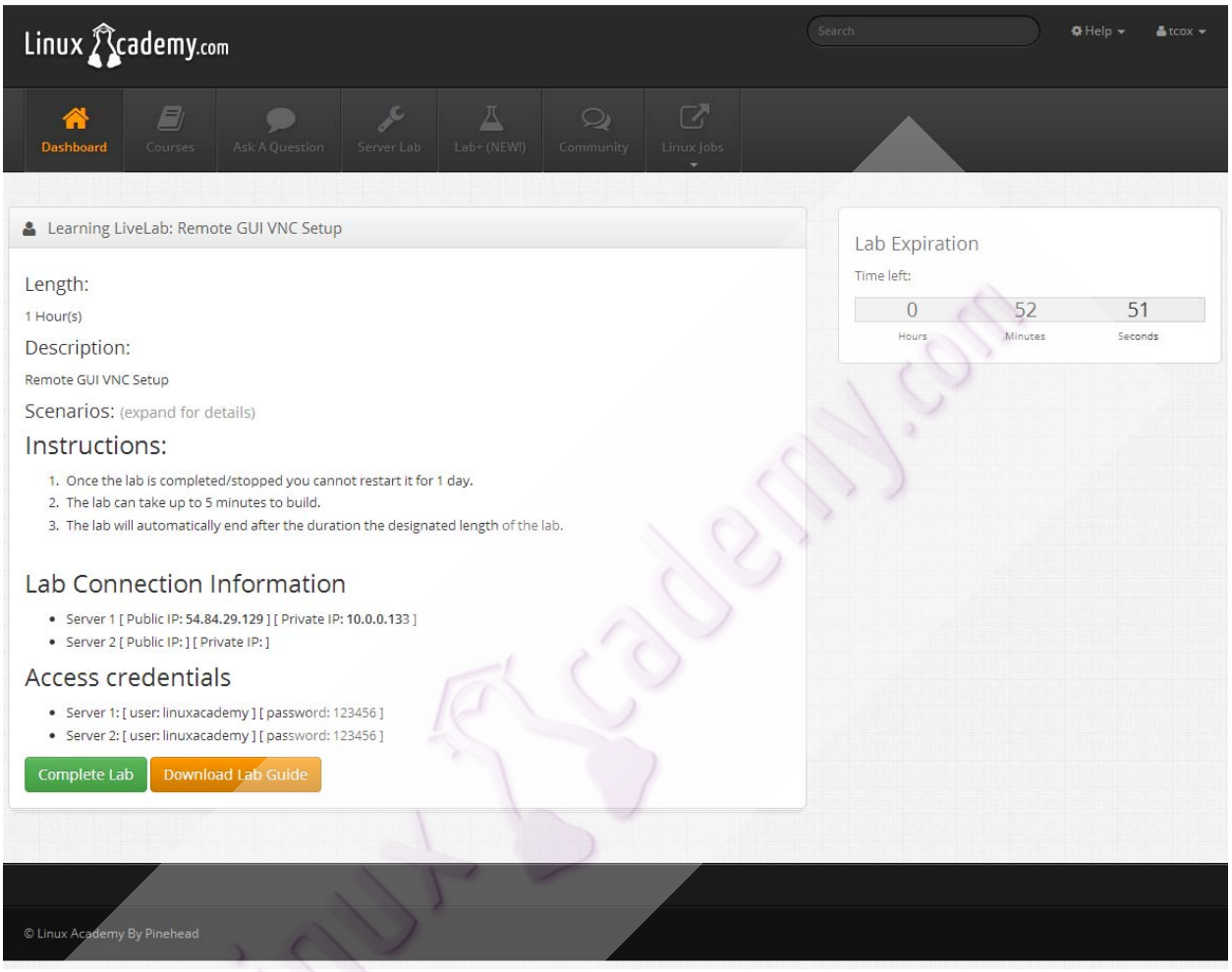


[Linux Academy Live Labs+ - Start Screen]

A few things to note before you start this process:

- When you launch the lab from this screen, it may take up to FIVE MINUTES for your servers to be deployed and be available for your use.
- Do not leave your desktop and come back, once the servers are launched, you will only have TWO HOURS to complete this lab from start to finish. After that point, the servers time out and are deleted permanently. Any and all work that you have done will then be lost.
- You can only use this lab ONCE PER DAY. If you try to use it more than that after completing the lab or the servers timing out, the screen will tell you when it will be available to you again.
- Other than those descriptions, you may retry any of the Labs+ labs as many times as you wish as long as you are a subscriber.

Once you have clicked on the 'Start Lab' button that you see above, a process will launch on our servers that will deploy the two servers we will use in our lab for testing. After a few minutes of processing (and you will see a status message that says "Creating Lab… Please Wait"), you should see a screen that looks like this one:



[Linux Academy Live Labs+ - Start Screen]

You will see all the information you need to access your servers from another system. Specifically, you need:

- The server public IP addresse
- Access credentials

One thing to note is that, in addition to the IP that you see above, the server will have another IP assigned to it in the 10.0.0.x subnet. This is a private IP address and will not route outside of your private server pool. Your server will have a static private address of 10.0.0.100. We will be using the external IP address to connect over SSH as well as when configuring VNC.

Linux Academy Lab Server – Running the Desktop GUI

Connect to your Linux Academy Lab server over SSH using a Windows client (like Putty) or another Linux system using the shell. We are going to install the KDE Desktop (you can install Gnome if you want but KDE is less resource intensive to run and the desktop manager runs well over VNC).

RPM Based Distributions (Bonus Content – Not For Lab)

Once you have your RPM based distribution running, you will want to install a desktop. We are going to use KDE in our examples throughout this lab, but in this section, we will discuss the packages necessary to install KDE or Gnome. If you choose Gnome, please simply substitute Gnome for KDE in subsequent discussions.

Make sure you have logged in and the user either has SUDO access or you perform the following steps as root. Our examples will show a normal user performing the steps with SUDO access. In order to install Gnome, you would execute the following:

```
sudo yum -y groupinstall "Desktop" "Desktop Platform" "X Window System" "Fonts"
```

This will install all the necessary packages and services to run Gnome. As you can see, there is quite a bit to install in order to run Gnome. For our purposes, let's install KDE as follows:

```
sudo yum -y groupinstall kde-desktop
```

Once the packages are installed, we do not need to change the runlevel or force the desktop to come up on boot. Since we are going to access the desktop remotely, having the packages installed is all we really need and will help cut down on the memory needed for our system.

Debian Based Distributions

Once you have your Debian based distribution running, you will want to install a desktop. We are going to use KDE in our examples throughout this lab, but in this section, we will discuss the packages necessary to install KDE or Gnome. If you choose Gnome, please simply substitute Gnome for KDE in subsequent discussions.

Make sure you have logged in and the user either has SUDO access or you perform the following steps as root. Our examples will show a normal user performing the steps with SUDO access. In order to install Gnome, you would execute the following:

```
sudo apt-get install gnome-core
```

This will install the core Gnome desktop for Debian without most of the overhead and end user applications. You can then install them as needed in order to cut down on resource and disk space utilization. For our purposes, let's install KDE as follows:

```
sudo apt-get install kde-plasma-desktop
```

Again, once the packages are installed, we do not need to change the runlevel or force the desktop to come up on boot. Since we are going to access the desktop remotely, having the packages installed is all we really need and will help cut down on the memory needed for our system.

Installation and Configuration of VNC Server

This section will talk about the VNC server configuration on both systems and how to start and configured it securely. Once we complete the server configuration, the client configuration to access the server is identical regardless of the distribution.

RPM Based Distributions (Bonus Content – Not for Lab)

Installing the server is very easy, simply log in and make sure you are running the following commands as a user with SUDO privileges or root. You can install the VNC server with the following command:

```
sudo yum install tigervnc-server
```

This will complete the server installation, creating the necessary directories and services and start the VNC server itself. However, as you will find, there are no configurations in place so there is nothing for a user to connect to. At this point, we need to stop the VNC server, on RPM based distributions, we can either:

```
sudo service vncserver stop
```

Or:

```
sudo /etc/init.d/vncserver stop
```

Either way will get the service to stop. We now need to create our local directory structure for our user VNC configuration. No need to do this manually, we can have VNC do this for us by executing the command:

```
vncpasswd
```

Now in this case, you will notice we are NOT using the 'sudo' command to execute the 'vncpasswd' utility. The reason being is that we are creating both a password we can use when connecting over VNC to the server as ourselves, but this way, VNC will build our ~/.vnc directory that we need to store our log file, our PID file and the xstartup command.

Next, we need to edit a configuration file in order to configure the VNC desktop we are going to connect to. Edit the file "/etc/sysconfig/vncservers" and change it to look like the following example (substitute your user as appropriate):

```
VNCSERVERS="1:tcox"
VNCSERVERARGS[1]="-geometry 1800x950 -localhost"
```

When you launch a VNC server, by default, it launches itself on ports in the 5900 range. The first line of this configuration will launch a single VNC server for the user 'tcox' (which is why we needed to build our directory and create our password earlier) running on port 5901 (base port 5900 + 1 for us). In addition, we are telling that server (argument [1]) to start the desktop at the size of 1800x950 and to answer only on the localhost.

The last parameter may be a bit confusing since it is effectively stopping any remote VNC connection. However, this is how we are going to force a secure connection, we will tunnel our connection over SSH as we will see in the client configuration section.

The next thing we need to do is restart the VNC server. This will create a template "xstartup" file for our user that determines how our desktop will look. Execute the following:

```
sudo service vncserver start
Starting VNC server: 1:tcox
New 'X' desktop is ca-centos6ut01:1

Starting applications specified in /home/tcox/.vnc/xstartup
Log file is /home/tcox/.vnc/ca-centos6ut01:1.log
```

Now that we have started the service and it has started our desktop, we need to make a few changes to the local 'xstartup' file so the entire desktop is started. Edit the file called "/home/yourusername/.vnc/xstartup" so it looks like the following:

```
#!/bin/sh

[ -r /etc/sysconfig/i18n ] && . /etc/sysconfig/i18n
export LANG
export SYSFONT
vncconfig -iconic &
unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS
OS=`uname -s`
if [ $OS = 'Linux' ]; then
```

```
  case "$WINDOWMANAGER" in
    *gnome*)
      if [ -e /etc/SuSE-release ]; then
        PATH=$PATH:/opt/gnome/bin
        export PATH
      fi
      ;;
  esac
fi
if [ -x /etc/X11/xinit/xinitrc ]; then
  exec /etc/X11/xinit/xinitrc
fi
if [ -f /etc/X11/xinit/xinitrc ]; then
  exec sh /etc/X11/xinit/xinitrc
fi
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
twm &
```

Of particular interest in this configuration file is the last line which will actually start the full window manager when the service is started for us. Finally, all we have left is to restart the VNC service so it runs our new configuration:

```
sudo service vncserver restart
```

Our server setup is done for RPM based distributions. The Debian install is a bit easier since it is user command line driven instead of a service.

Debian Based Distributions

There are several VNC servers that will work with Debian systems, however, since Ubuntu is built around Debian, we are limited by some of the non-standard implementations of Ubuntu. The one VNC service we can rely on across the board is a package called "vnc4server". Installing the server is easy to complete as follows:

```
sudo apt-get install vnc4server
```

We now need to create our local directory structure for our user VNC configuration. No need to do this manually, we can have VNC do this for us by executing the command:

```
vncpasswd
```

This will not install a service, but will install a server that we can execute on the command line as needed. Let's create a script we can just execute whenever we are ready to use VNC on this

server specific to our needs. Execute the following (substituting the desktop geometry you want in your case):

```
echo “vncserver :1 -geometry 1850x950 -depth 24 -localhost &” > ~
vncuser.sh && chmod 755 vncuser.sh

sudo cp vncuser.sh /usr/bin
```

This will create our script file. Now we can execute it to create our “xstartup” file:

```
/usr/bin/vncuser.sh
```

We can now kill the VNC session since we need to make some changes to our configuration. You can kill the VNC server by executing:

```
vncserver -kill :1
```

This kills the vncserver at port 5901 (base port 5900 + 1 for this user) running under this user ID. Now let’s edit the file at “/home/yourusername/.vnc/xstartup” to look like the following example:

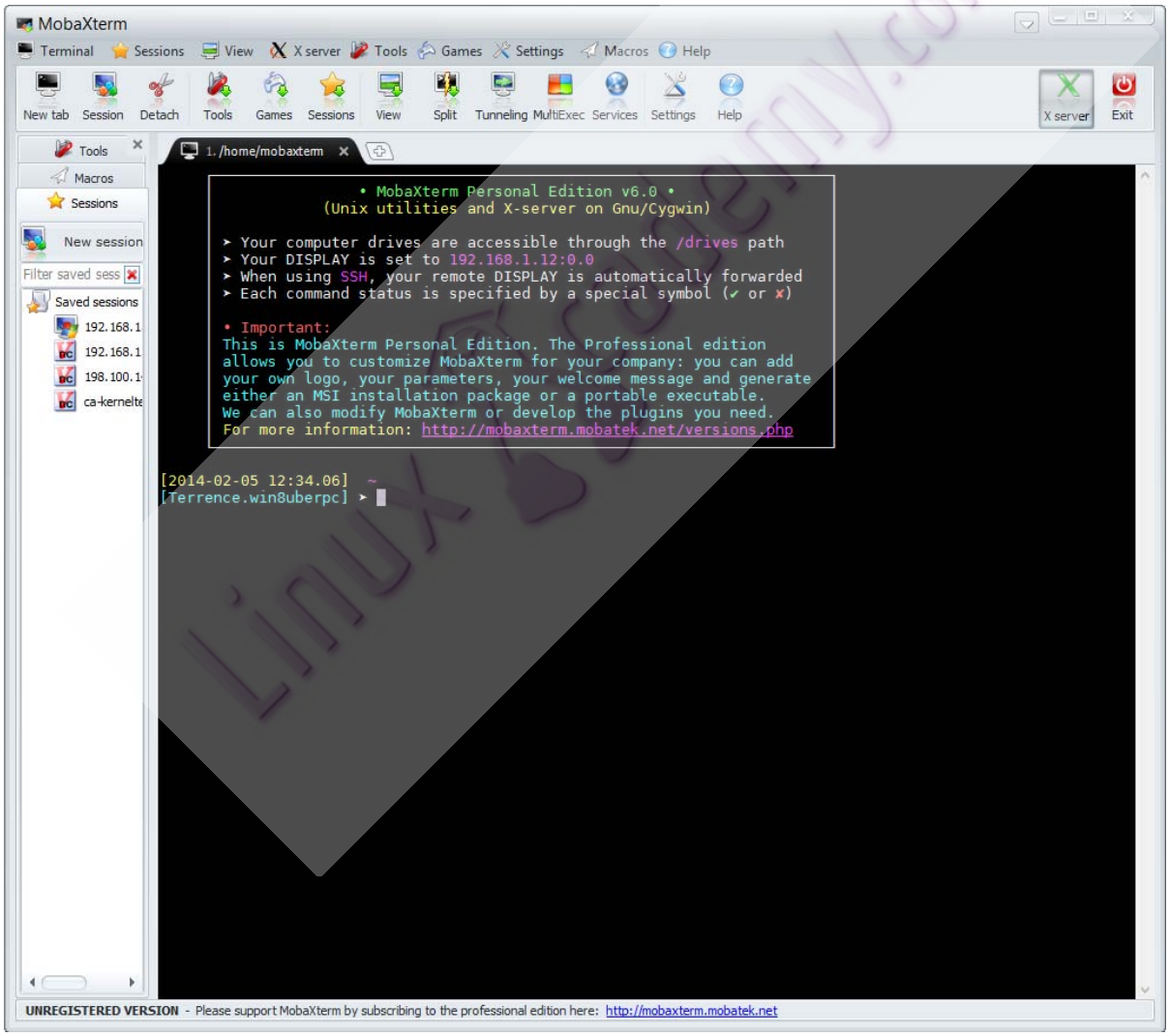
```
#!/bin/sh

xrdb $HOME/.Xresources
xsetroot -solid grey
#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP~
Desktop" &
#x-window-manager &
# Fix to make GNOME work
export XKL_XMODMAP_DISABLE=1
/etc/X11/Xsession
```

Once that file is saved in place, we can restart the VNC server by running the “vncuser.sh” script that we created above and then we are done with configuration on a Debian based distribution. Remember, we set VNC to only respond to connections from localhost in order to force the client to tunnel connections securely over SSH. We will go over the client connections next.

Client Configuration

Now that our servers are configured for secure VNC access, we need to set up our clients so that we can tunnel over SSH and connect to our desktop. Luckily, the configuration for the client is the same regardless of the distribution you use on your desktop machine to connect through with a minor exception.

If you are using Windows, most people's "go to" SSH client is Putty. Although that will certainly work, the configuration can be confusing and requires another application once configured for VNC access. I recommend you download (very small) and use a full Windows SSH client application called "MobaXTerm" from http://mobaxterm.mobatek.net/. This application is completely free and provides you a complete Linux like environment on your Windows system contained in a single application (it even embeds an X Windows server and automatically tunnels X over SSH if available from your server). The client looks like the following screen print:



Besides being free, the best thing about using this client is that you can then set your system up the exact same way as you would a Linux system for tunneling.

What we need to do now is create a script we can run locally that will allow us to connect to our remote VNC session locally over SSH. We have to set up some local port redirections. Open a shell on your client (Linux shell or MobaXTerm) and run the following command:

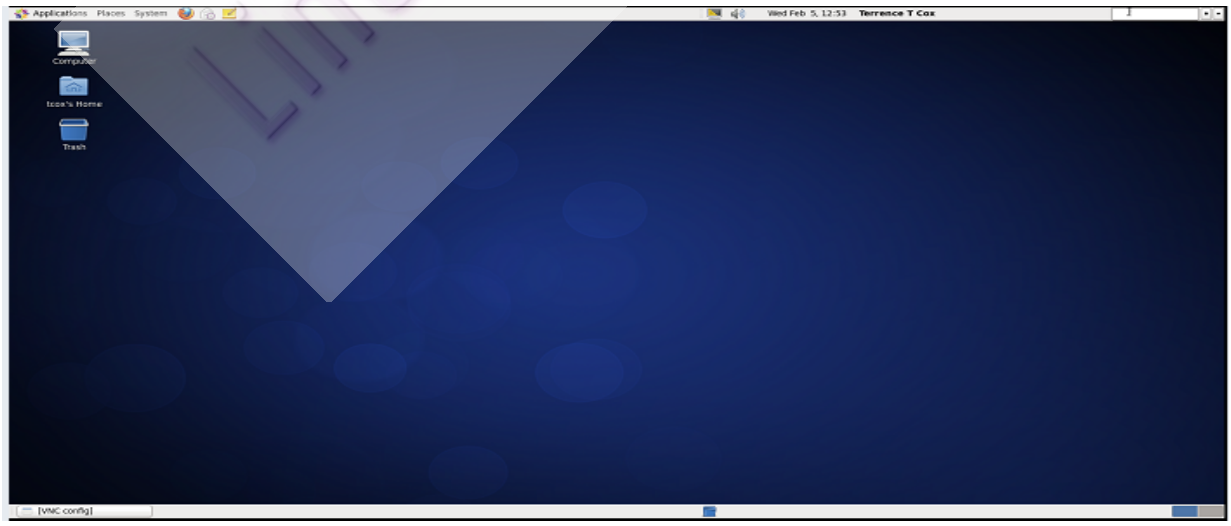
```
echo "ssh -L 5901:localhost:5901 -N -f -l username 10.0.0.5" > ~
hostsetup.sh

chmod 755 hostsetup.sh && sudo cp hostsetup.sh /usr/bin
```

So what this long command does is tell our local system to run SSH in the background on our localhost on port 5901, forwarding that port over SSH to port 5901 to SSH on the server 10.0.0.5 (use your IP address) for user "username". We are now redirecting port 5901 traffic over SSH to VNC remotely.Now, we can use any VNC client and connect to "localhost" on port 5901 and it will forward the traffic to the server and display your desktop on the VNC server.

Be sure you run this command any time you restart your system or client application so the redirect is created otherwise when you set up your VNC client, you will get a "connection refused". We can set up our redirect simply by running the script we just created:

```
hostsetup.sh
```

You can use any number of VNC clients, for Windows users, use the MobaXTerm client as it has VNC built it. When you set up your VNC client connection string, you will connect to the hostname "localhost" (you are effectively connecting to yourself and using SSH to tunnel your connection) and port 5901 (again, redirected by our command above). You will then be prompted for the VNC password you set up on your remote server (the "vncpasswd" command from earlier) and then your desktop will start up!



APPENDIX A – Configuration Files

RPM Based Distributions – xstartup (Bonus Content – Not for Lab)

```
#!/bin/sh

[ -r /etc/sysconfig/i18n ] && . /etc/sysconfig/i18n
export LANG
export SYSFONT
vncconfig -iconic &
unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS
OS=`uname -s`
if [ $OS = 'Linux' ]; then
  case "$WINDOWMANAGER" in
    *gnome*)
      if [ -e /etc/SuSE-release ]; then
        PATH=$PATH:/opt/gnome/bin
        export PATH
      fi
      ;;
  esac
fi
if [ -x /etc/X11/xinit/xinitrc ]; then
  exec /etc/X11/xinit/xinitrc
fi
if [ -f /etc/X11/xinit/xinitrc ]; then
  exec sh /etc/X11/xinit/xinitrc
fi
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
twm &
```

Debian Based Distributions - xstartup

```
#!/bin/sh

xrdb $HOME/.Xresources
xsetroot -solid grey
#x-terminal-emulator -geometry 80x24+10+10 -ls -title
"$VNCDESKTOP~ Desktop" &
#x-window-manager &
# Fix to make GNOME work
export XKL_XMODMAP_DISABLE=1
/etc/X11/Xsession
```