Setting Up a TigerVNC Server for Remote Control of Linux Desktop Environment

## Install the Desktop Environment

This tutorial will use the XFCE desktop environment. XFCE is a lightweight, fast, and low-resource desktop environment, which is well suited for a device like the Raspberry Pi.

sudo apt update

sudo apt install xfce4 xfce4-goodies

`xfce4-goodies` is a package that provides a collection of additional tools, utilities, and plugins specifically designed to enhance the XFCE desktop environment.

## Install the TigerVNC Server

The `tigervnc-xorg-extension` adds support for Xorg. Xorg is an open-source implementation of the X Window System, which provides the foundational graphical environment for UNIX-like operating systems, including Linux. Xorg acts as an intermediary between the operating system and the display hardware (like your monitor), handling graphical rendering, input devices, and window management. It is essential for running graphical desktop environments like XFCE, GNOME, or KDE.

sudo apt install tigervnc-standalone-server tigervnc-xorg-extension

## Create a Script to Execute Upon Creation of a VNC Session

TigerVNC is configured to execute a script at the directory in the command below upon creation of any new VNC session.

This script defines what should happen when a new VNC session is started (such as launching of the desktop environment). The file name should be `xstartup`.

nano ~/.vnc/xstartup

The following content should be added to the `xstartup` script:

#!/bin/sh

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup

[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources

vncconfig -iconic &

startxfce4

**`[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup`**: If the `/etc/vnc/xstartup` file exists and is executable, it is executed. This is a fallback to system-wide startup settings if they exist.

**`[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources`**: If the `.Xresources` file in the user's home directory is readable, the X resources (settings for X11 applications) are loaded.

*\*X11 applications are graphical programs that run on the X Window System, commonly referred to as X11 or X.*

**`vncconfig -iconic &`**: Starts the VNC configuration in the background with the iconified (minimised) option. `vncconfig` is a utility that is part of the VNC suite of tools. It is primarily used for configuring and managing certain aspects of a VNC session, particularly related to clipboard sharing between the VNC client and the server.

**`** **startxfce4`**: Starts the XFCE desktop environment when the VNC session begins.

The `xstartup` script file needs to be made executable in order for it to run when a VNC session starts:

chmod +x ~/.vnc/xstartup

## Start the VNC Server / Session

This command starts the VNC server daemon (if it is not already running) and creates a new virtual desktop session. The VNC server listens for incoming connections from VNC clients and manages the display, input, and session settings.

vncserver

Each VNC session is associated with a specific display number, such as `:1`, `:2` etc. A port number is derived from the display number of each session. VNC typically uses port 5900 as the base port, and the port number for any given session can be found by adding the session number to the base port number:

For display `:1`, the port number is `5900 + 1 = 5901`.

For display `:2`, the port number is `5900 + 2 = 5902`.

And so on.

Explicitly restricts VNC access to connections originating from the same machine, although this is done implicitly:

## Restricting VNC Access to Local (localhost) Connections

By default, the VNC server only accepts connections from the local machine (127.0.0.1). This enhances security by ensuring that the VNC server is not exposed to external connections, and it will only be accessible through an SSH tunnel.

This restriction can also be stated explicitly using `-localhost`:

vncserver -localhost

## Connecting to the VNC Server from the Client Machine Using an SSH Tunnel

An SSH tunnel is a method of forwarding network traffic from one network to another securely through an SSH connection. It allows you to redirect traffic from a local port on your machine to a remote port on another machine via SSH, essentially creating a secure "tunnel" for your data.

### How it Works:

* **Local Port Forwarding**: The SSH client listens on a specific port on your local machine (in this case, `5901`) and forwards any data received on that port to a specified port on the remote machine (also `5901` in this case).
* **Secure Connection**: Since the traffic is forwarded through an SSH connection, it is encrypted and secure, preventing eavesdropping or man-in-the-middle attacks.

When you create an SSH tunnel, the terminal behaves like a standard SSH session. You log in to the remote machine, and if you don't specify any special options (like `-N`), you get a command line interface (CLI) just as you would with a normal SSH connection.

ssh user@remote-server -L 5901:127.0.0.1:5901

In my case, this is:

ssh mattpatchava@192.168.0.32 -L 5901:127.0.0.1:5901

* **`ssh`**: This command starts an SSH session.
* **`mattpatchava@192.168.0.32`**: This specifies the remote user and the IP address of the remote machine.
* **`-L 5901:127.0.0.1:5901`**: This is the port forwarding option:
* **`5901`**: This is the local port on your machine that will listen for VNC connections.
* **`127.0.0.1`**: This specifies that the connection should be forwarded to the local interface (localhost) on the remote machine.
* **`5901`**: This is the port on the remote machine where the VNC server is running.

By connecting your VNC client to `127.0.0.1:5901`, you are essentially connecting to your local machine's port 5901. However, because of the SSH tunnel, any data sent to this port is securely forwarded to the remote machine's port 5901.

## Connect using a VNC Client

Using a VNC client (such as Screen Sharing on macOS), connect to:

`localhost:5901`

or

`127.0.0.1:5901`

This is the local endpoint of the SSH tunnel, which is forwarded to the VNC server on the remote machine.

## View Active VNC Sessions

This command lists all the active VNC sessions on your Raspberry Pi. It shows the display numbers and associated process IDs, helping you manage multiple sessions.

vncserver -list

## Stop the VNC Session

When you are ready to stop a VNC session, you can use the following syntax, specifying the display number of the session you want to terminate.

vncserver -kill :1