1. How X Over SSH really works

September 27, 2008

Imagine you are sitting in front of a machine named "**home**" that has a keyboard, mouse, display, and is running an X server. Now you open a terminal and ssh to a machine called "**remote**" (which doesn't need to have an X server running), and run a program like "firefox" which pops a window in your screen. How does this all work?


First, let's clear up the client/server terminology confusion. When talking about X:

* **X client**: a process (like firefox or xemacs) which uses the X client API to display things and receive mouse/keyboard events.
* **X server**: a process (usually just "X") which X clients connect to. At times (as we'll see below) other processes can act as X servers.

Whenever an X client starts up, it reads the local **$DISPLAY** environment variable, whose value looks like: **[hostname]:display\_number[.screen\_number]**. The X client immediately opens a connection to that X server. If it can't, it fails:

**user@home: echo $DISPLAY
:0 # hostname is "localhost" by default**

**user@home: xcalc # pops up a calculator on my screen**

**user@home: DISPLAY="nosuchhost:99"**

**user@home: xcalc**
**Error: Can't open display: nosuchhost:99**

Now let's see what happens when you ssh to another machine:

**user@home: ps aux | grep X # yep, X server running @home**
**root ... /usr/bin/X :0 ...**

**user@home: ssh -X user@remote # -X enables X-forwarding**

**user@remote: ps aux | grep X # nope, no X server here**

**user@remote: echo $DISPLAY**
**:11.0**

**user@remote: xcalc # pops up screen @home**

Wait a minute, the **$DISPLAY** variable is pointing to the localhost ("**remote**"). Natural questions to ask at this point:

* There is no X server **@remote**, so why didn't xcalc just fail on startup?
* Why was the display number "11".
* How did xcalc show something **@home**?

The answer has to do with the ssh-daemon running **@remote**:

**user@remote: ps aux | grep user**
**root ... sshd:user@pts/11**

What's happening is that there's an "X emulator" running **@remote** that was setup just for your ssh session, that is listening on display 11.

To review, here's a play-by-play:

1. You type "**ssh -X user@remote**" in your terminal
2. The ssh process connects to the sshd server **@remote**.
3. sshd spawns a new process that is an X-server-emulator listening on some display number, e.g. "11"
4. sshd sets the **$DISPLAY** to point to that local "X-server" (e.g. ":11")
5. xcalc reads this **$DISPLAY** and conncects to this X-server. xcalc thinks it's displaying to the local machine.
6. the X-server-emulator simply forwards the X commands from xcalc through the ssh connection, to the original ssh process.
7. The ssh process **@home** now acts as a normal X-client and sends those commands to the X-server **@home**.

[](http://1.bp.blogspot.com/_E3Va1433EXQ/SN6jXcdzzII/AAAAAAAABd8/B-IlTHDjIWc/s1600-h/xoverssh.png)Some of you might be wondering: wasn't the X protocol designed to go over the network? Can't you do all this without ssh? You might be tempted to try something like:

**user@remote: DISPLAY="home:0"**
**user@remote: xcalc**

This doesn't work because the X server **@home** won't let other hosts connect to it. To change this (not that you should -- see below) you can do:

**user@home: xhost +remote**

xhost is a command which says "that host can connect to our X-server". However, everybody uses ssh X forwarding instead. Here are some **security reasons** why:

* normally, X-traffic (like your keystrokes) is sent unencrypted from X-client to X-server.
* ssh nicely sends that data through an encrypted channel, so it doesn't go over the internet in the clear.
* "**xhost +remote**" is putting a lot of trust in 'remote' being a nice guy. If remote ever gets hacked, it could connect to the X-server **@home** and listen to all its keystrokes.

There's also an issue with **firewalls**: by doing "**DISPLAY=home:0**", you're assuming that a connection can be established from **remote** -> **home**. But this isn't always possible -- **home** might be sitting behind a firewall (like your Netgear router). Since the ssh-connection was setup from **home**-> **remote**, it takes advantage of this already-established connection.

Other notes for the curious:

* if **$DISPLAY** is set to "**localhost:0**" (or any explicitly named host) it uses tcp-ip to send the X-traffic locally.
* if **$DISPLAY** is just "**:0**" it uses a special (more efficient, non-tcp-ip) connection.
1. Settings on debian

/etc/ssh/sshd\_config : X11Forwarding is set to yes

* UNIX
enable X11 forwarding op de server, dan is op zijn minst het xauth programma vereist.
1. installeer xbase-clients op de server (of de package dat xauth bevat)
2. problemen wanneer je verandert van user
X authenticatie is gebaseerd op cookies, voor de server vandaar deze toevoeging

root@debian:/home# xauth list

root@debian:/home# **xauth list $DISPLAY**
debian/unix:0 mit-magic-cookie-1 4b73137421627917931dbc5b67ccab8c

root@debian:/home# **# moet in hoofdletters zijn**

root@debian:/home# **xauth add** debian/unix:0 MIT-MAGIC-COOKIE-1 4b73137421627917931dbc5b67ccab8c

root@debian:/home# xauth list
debian/unix:0 MIT-MAGIC-COOKIE-1 4b73137421627917931dbc5b67ccab8c

root@debian:/home# xauth remove debian/unix:0

root@debian:/home# xauth list

<http://www.debian-administration.org/articles/494>

1. maak een connectie met de server gebruik makend van SSH
ssh -X servername
2. run het programma
* WINDOWS
install [Cygwin/X](http://x.cygwin.com/)
1. XDMCP

/etc/gdm3/daemon.conf, and in the XDMCP section Enable = 1

telinit 1

cd /etc/X11

Xorg

1. X Window

Officially the "X Window System," but also called "X Windows," "X11" or simply "X," it is an open source windowing system developed at MIT in the early 1980s. It was created to provide a common graphics rendering engine for Unix applications. Prior to X, CAD and scientific modeling applications that required graphics output used proprietary software to render images. X is also the de facto graphics engine in Linux desktops.

Version X11 was released in 1987 and remains the current standard, having undergone many revisions. The X.Org Foundation (www.x.org) governs the X Window standards for Unix/Linux desktops, which evolved from XFree86 implementations (www.xfree86.org). Hummingbird's Exceed (www.hummingbird.com) and AttachmateWRQ's Reflection (www.wrq.com) are commercial X Window implementations for Windows desktops.

* 1. Network Transparency
	One of the unique features of X is that it allows applications to run on a network server, but be displayed on a desktop machine. This was very significant in the 1980s and 1990s when servers were far more powerful than user machines. In the early days of X, dedicated X Window hardware, known as "X terminals," were widely used. They accepted input, rendered output and performed no application processing.
	2. The X Window Manager
	X Window, by itself, generates borderless windows in fixed screen locations. It requires a "window manager" to add borders and buttons and the ability for users to resize and move the windows on screen. The Tabbed Window Manager (twm) has been the default X window manager, but more than three dozen others have been used, including AfterStep, Blackbox and Enlightenment. The KDE and GNOME user interfaces for Linux use Kwin and Metacity respectively as their window managers.
	3. Server Runs in Client; Client Runs in Server
	X Window was designed as a client/server architecture. The application is the "X client," and the software that accepts keyboard and mouse input and renders the images on screen is called the "X server." Communications between X clients and the X server is via the X protocol.

	Since the user's machine handles user input and output, the X server always runs in the client machine. Applications (X clients) generally run in the server; however, they can also run in the client machine. For example, all applications in Linux desktops are X clients running in the same machine as the X server. See [XIE](http://encyclopedia2.thefreedictionary.com/XIE) and [thin client](http://encyclopedia2.thefreedictionary.com/thin%2Bclient).

|  |  |
| --- | --- |
| http://img.tfd.com/cde/XWINDOW.GIF | **X Window Servers Run on Clients** The X Window server runs on your client, and the X Window client runs on your server. All's fair in love and computers |
| http://img.tfd.com/cde/_XWINSCR.GIF | **An X Window Screen** Using the twm window manager, the two "xterm" windows (top and bottom) display output from applications in two remote servers. They emulate input/output terminals and provide command lines for launching apps. (Screen shot courtesy of Peter Hermsen.) |

1. bron

<http://dustwell.com/how-x-over-ssh-really-works.html>