Global Information Assurance Certification Paper

X Windows Security: How to Protect your Display

by Arturo Guillen

November 16, 2001

Introduction

The X Windows is as described in the X man pages a "portable, network-transparent window system". X Windows is such a powerful and flexible interface that the user does not have to be in front of the machine to interact with it. Instead, the user can be remotely located working in a transparent way. X Windows uses a client/server model, where the client-server communications are done following the X protocol.

X Windows as the time of this writing is in its version eleven, release 6 (X11R6). The X consortium is in charge of maintaining the X Windows System and provides it free of charge and open source. X Windows is the standard protocol among UNIX-like systems, but it is so popular that there are also implementations for personal computers and Macintosh that allow X emulation. There is even dedicate hardware equipment that only knows how to talk the X protocol language, the so called X-terminals.

This paper analyzes the threads and describe the security involved in the X Window Systems. It takes a practical overview of the X Windows security to emphasize the risks and indicate the measurements that must be taken.

Terminology

For clarification purposes and before any further explanation, let us revise the very basic X terminology (an exhaustive list can be found in [2]):

- **X clients**: These are the programs that are running in the X client machine (can be local or remote). i.e. xclock, xterm, xeyes, etc.
- **X server**: Is the machine where the display resides on. It is the place where the interaction with the user is done. In plain words, a X server is the workstation, PC or X-terminal in front of the user.
- **Display**: Normally consists of a monitor, a keyboard and a mouse.

  A display name is described in one of the following formats:

  - "hostname:D.S", where S is the screen number on display number D. The X server for this display is listening at TCP port 6000+D of hostname where the display is physically connected.
  - "host/unix:D.S", where S is the screen number on display number D. The X server for this display is listening at UNIX domain socket /tmp/.X11-unix/XD on host.
  - ":D.S" is equivalent to host/unix:D.S, where host is the local hostname.

Note that in X Windows the client/server model is reversed from the traditional idea of the model. Here the client is the one that runs the application (normally the "big machine") and the server is the machine that gets the user's input and displays the results (normally the "small machine").

How X Windows Works

In order to analyze the security in the X Windows System it is necessary that we understood how the system works. To do so, let us go through a practical example. Note that this is just an example to explain how X Windows works and it is **not** the most desirable approach.

Step 1. The user grants access to the X server to the remote machine, where the X client is going to be run. This is one of the two mechanisms X Windows has to control access to the server. More on this later on.

```
xserver.host> xhost +xclient.host

Step 2. The user connects to a remote machine (the X client).

xserver.host> ssh -X xclient.host -l arturo

Step 3. The user sets the DISPLAY environmental variable (or alternatively use the
-display switch in the X client call).

xclient.host> setenv DISPLAY xserver.host:0.0

Step 4. The user runs a X client application.

xclient.host> xclock &

At this point, even though the application is running in xclient.host, the output
(a window with a clock in it) is displayed in xserver.host. We can see in this
example how the X protocol makes all the connections transparent to the user.
Behind the scenes there are multiple connections between xclient.host, using the X
library (Xlib), to send/receive X events to/from xserver.host. The details of this
connections are beyond the scope of this paper.

Note that we have used ssh or openssh to connect to the remote machine since the
connection is encrypted and have the capability to forward X requests (although be
aware that ssh can problems when interacting with X11 [7]).
```

X Windows Threads

```
Among the threads the user is exposed if he/she does not secure properly his/her
display are the following (as described in the SATAN documentation [3]):

   • Read the user's keyboard, including any passwords that the user types.
   • Read everything that is sent to the screen.
   • Write arbitrary information to the screen.
   • Start or terminate arbitrary applications.
   • Take control of the user's session.

Let us imagine that a remote.host has an unprotected display:

remote.host> xhost +

Now anybody can change the remote.host background:

local.host> xsetroot -display remote.host:0.0 -solid green

Anybody can see what remote.host's console screen looks like:

local.host> xwd -display remote.host:0.0 -root | xwud

Anybody can create windows with messages on remote.host's display:

local.host> xmessage -display remote.host:0.0 "Your display is unprotected!!!"
```

Anybody can activate the screen saver on remote.host:

```
local.host> xset -display remote.host:0.0 s activate
```

Anybody can reverse the mouse buttons functionality:

```
local.host> xmodmap -display remote.host:0.0 -e "pointer = 3 2 1 4 5"
```

Anybody can snoop at remote.host display (grabbing keystrokes):

```
local.host> xkey remote.host:0.0
```

Note: xkey can be easily found doing a search on Google.

There are several scanners that look for unprotected displays on the Internet (i.e. Xscan, SATAN). SATAN's brief tutorial in X security is worth reading [3]. So it looks like it is a good idea to protect one's display. How to do that is explained in the next section.

X Windows Authentication

Authentication is the process of verifying that an entity is the one it really claims to be. The X authentication is independent of the operating system's authentication. Thus, even if a machine is closed down, there is still another door to close: the one the X Window System is using.

Since X11R4 the X Window System provides two main different mechanisms of access control. Both of them restrict access to the X server's display.

Host-based Authentication

The first method of authentication is based on the origin of the communication. An X server will check the requester's IP address to determine if that request can have access to the display. The command to manage this mechanism is the **xhost** command. Here are some examples:

To grant access to everybody (this means any single machine on the Internet!):

```
arturo@local.host> xhost +
access control disabled, clients can connect from any host
```

To disable any incoming request, try:

```
arturo@local.host> xhost -
access control enabled, only authorized clients can connect
```

To grant remote.host machine access to the display:

```
arturo@local.host> xhost +remote.host
remote.host being added to access control list
```

To check the status of your display and the access list, try:

```
arturo@local.host> xhost
access control enabled, only authorized clients can connect
INET:remote.host
```

To remove permissions from remote.host machine to access the display:

```
arturo@local.host> xhost -remote.host
remote.host being removed from access control list
```

Let us make sure that our access list is empty (after removing access to remote.host):

```
arturo@local.host> xhost
access control enabled, only authorized clients can connect
```

It is worth to mention that once a connection is granted there is no mechanism to close the X Windows connection. "xhost -" will not affect current connections to the X server.

The xhost command uses the "/etc/XD.hosts" files that contain the X server authentication information (D is the display number).

Refer to the xhost manual page for detailed information [6].

This mechanism is intended to be used when in the X client machine all the users are trusted. When you issue a "xhost +remote.host" **all** the users in the remote.host are allowed to access the X server.

User-based Authentication

To work around the host-based authentication broadness, X Windows offers a more specific user-based authentication.

In X11R6 there are four user-based authentication mechanisms:

MIT-MAGIC-COOKIE-1	Shared plain-text "cookies".
XDM-AUTHORIZATION-1	Secure DES based private-keys.
SUN-DES-1	Based on Sun's secure rpc system.
MIT-KERBEROS-5	Kerberos Version 5 user-to-user.

Among these methods the first one is the most common preferred and is the one explained hereafter. Read the Xsecurity man page for the detailed explanation of the different methods.

MIT-MAGIC-COOKIE-1 authentication:

From the Xsecurity man page [6]:

"When using MIT-MAGIC-COOKIE-1, the client sends a 128 bit 'cookie' along with the connection setup information. If the cookie presented by the client matches one that the X server has, the connection is allowed access. The cookie is chosen so that it is hard to guess".

As we can see, this mechanism uses one of the most common mechanisms of authentication: the challenge/response method. The **xauth** command is in charge of managing the $HOME/.Xauthority file that keeps the authentication information. Let us see some practical examples:

To check the status of the $HOME/.Xauthority file:

```
local.host> xauth list
local.host/unix:10 MIT-MAGIC-COOKIE-1 6ef186cae5a46e2da83417200d178205
local.host/unix:0 MIT-MAGIC-COOKIE-1 762f6ebeb93d3368030208e8ff5d377a
```

The format of this file is as follows:

Display name	user authentication method	cookie

As explained later on some programs take care of the user authentication automatically, like the Xwindows Display Manager (xdm), but it is also possible to

generate the cookie manually. You can do something like this:

```
local.host> (nfsstat -r; nfsstat -n; date; netstat -a) | md5sum -
b525128ef24031cd4228a4c4f2c46f96 -
```

Then you can use that string as a cookie to the $HOME/.Xauthority file:

```
local.host> xauth add `hostname`:0.0 MIT-MAGIC-COOKIE-1
b525128ef24031cd4228a4c4f2c46f96
```

To check again the status of the $HOME/.Xauthority file:

```
local.host> xauth list
local.host/unix:10 MIT-MAGIC-COOKIE-1 6ef186cae5a46e2da83417200d178205
local.host/unix:0 MIT-MAGIC-COOKIE-1 762f6ebeb93d3368030208e8ff5d377a
local.host:0 MIT-MAGIC-COOKIE-1 b525128ef24031cd4228a4c4f2c46f96
```

Now on the remote machine (the X client) add the same cookie to the user's $HOME/Xauthority file (using cut/paste):

```
arturo@remote.host> xauth add `hostname`:0.0 MIT-MAGIC-COOKIE-1
b525128ef24031cd4228a4c4f2c46f96
```

A way to automate the process from local.host is the following command:

```
local.host> xauth extract - local.host:0 | ssh arturo@remote.host -s xauth merge -
```

We have just granted user arturo on remote.host unlimited access to the X server running on local.host. We can also realize the importance of the file $HOME/.Xauthority in the users home directory of both local.host and remote.host. Check the permissions for this file, although it is in binary format, nevertheless it should not be readable for anybody besides the owner.

Now we are allowed to run X client programs on the remote.host and display them in local.host:

```
local.host> ssh arturo@remote.host -s "setenv DISPLAY local.host:0 ; xclock &"
```

Once we are done, we can remove the cookie as follows:

```
local.host> xauth remove local.host:0
```

The xauth program can also run in an interactive mode:

```
local.host> xauth
Using authority file $HOME/.Xauthority
xauth> info
```

At the "xauth>" prompt you can issue all the previous commands, i.e. list, add, remove, etc.
Check the xauth man page for more details [6].

Conclusion

As we have seen X Windows offers the users a very powerful interface to interact with the machine, but if not handled properly, it can create a serious risk on a machine's security. A misuse of the xhost command can put your display in a WYSIWIS (What You See Is What I See) mode.

There are many scanners looking for unprotected displays. Once a vulnerable display is found, there are utilities to exploit it. An example of a real compromise where the xkey program was used to grab keystrokes can be found here. It is possible to prevent other X applications from snooping on your keystrokes in applications like

xterm that offer a "Secure Keyboard" option.

The X Windows security problem is not a new topic, it has been around for many years [1]. This fact is reflected in general UNIX security check lists and security articles. In this guide the CERT warns us out about vendors shipping unprotected versions of X servers:

"If your /usr/lib/X11/xdm/Xsession file includes an 'xhost' command
with a '+' entry, such as /usr/bin/X11/xhost + remove that line. (Note that
the 'xhost' command may be in a different directory tree on your system.)
If such a line remains intact, anyone on the network can talk to the X server
and potentially insert commands into windows or read console keystrokes."

In this checklist Dave Dittrich, one of the members of the honeynet project give us some advice about X Windows security:

"Disable broadcast and/or indirect XDM requests for any X terminals that you don't explicitly want to support (the default from MIT is for XDM to accept a broadcast or indirect request from any X terminal)."

X Windows suffers from many security limitations. This problem comes from its insecure design. Like in many other protocols (like HTTP) X Windows was designed to share resources in trusted environment and security was not a priority. Later on some security measures were added, but many insecurities are still inherited in new releases from the original design. In a network protocol like the X protocol, there are three security aspects to take into account:

- **Authentication:** Is the process of verifying that an entity is the one it really claims to be. As we have seen in X Windows there are two main different authentication mechanisms: host-based and user-based. There is no session or connection based mechanism of authentication.
- **Authorization:** Process that allows the user access to various resources based on the user's identity. In X windows, once a user has been granted access to an X server, there is not restriction on what that user can do, meaning there is no authorization whatsoever.
- **Privacy:** Is the process in which the information is kept hidden to other parties which are not involved in the communication. Privacy is normally achieved by encryption. X Windows does not provide privacy. This can be solved tunneling X requests through ssh or layering them on top of SSL.

The recommendation is, as for any other service running in your machine, if you are not going to use X Windows, disable the service. For example, machines behind a firewalls can filter the 6000-6255 range of TCP ports from the outside. Also, the xhost command can be disable.

If X Windows has to be used, choose a user-based authentication mechanism and pipe your X requests through ssh. Do not forget to remove your cookie at the end of the session.

References

[1] Fisher, John. "Securing X Windows." August, 1995.
URL: http://ciac.llnl.gov/ciac/documents/ciac2316.html (15 Nov 2001).

[2] Scheifler, Robert W. "X WINDOW SYSTEM PROTOCOL, VERSION 11." June 1987.
URL: http://www.faqs.org/rfcs/rfc1013.html (15 Nov 2001).

[3] Farmer, Dan and Venema, Wietse. "Security Administrator Tool for Analyzing Networks (SATAN)."
URL: http://www.porcupine.org/satan/ (15 Nov 2001).

[4] comp.windows.x FAQ.

URL: http://www.faqs.org/faqs/x-faq/ (15 Nov 2001).

[5] Mui, Linda and Pearce, Eric. "X Window System Administrator's Guide, Vol. 8.", November 1993.
ISBN: 0937175838, Publisher: Thomson Learning.

[6] X, Xsecurity, xhost and xauth UNIX manual pages.

[7] Flegel, Ulrich. "The Interaction between SSH and X11". September 30, 1997.
URL: http://the.wall.riscom.net/books/unix/rootsh/docs/ssh-x11.ps.gz (15 Nov 2001).

Last Updated: July 26th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
Community SANS Indianapolis SEC401	Indianapolis, IN	Oct 09, 2017 - Oct 14, 2017	Community SANS