Security Issues 6

The Solaris environment supports two access control mechanisms: user-based and host-based. It also supports two authorization protocols: MIT-MAGIC-COOKIE-1 and SUN-DES-1. This chapter discusses these access control mechanisms and authorization protocols. It also discusses how to change the server's access control, and how to run clients remotely, or locally as a different user.

Notes About This Chapter

If you run applications in any of the following configurations, you need to read this chapter:

- Linked with a version of `Xlib` *previous* to OpenWindows Version 2 or X11R4. See "Host-Based" on page 94 for details.
- *Statically* linked to OpenWindows Version 2 libraries *and* you want to use the SUN-DES-1 authorization protocol. See "SUN-DES-1" on page 95 for details.
- On a remote server. See "Running Clients Remotely, or Locally as Another User" on page 100 for details.

If you are not using any of the configurations listed above, you do not need to change the default security setup.

6

Access Control Mechanisms

An access control mechanism controls which clients or applications have access to the OpenWindows server. Only properly authorized clients can connect to the server. All unauthorized X clients terminate with the following error message:

```
Xlib: connection to hostname refused by server
Xlib: Client is not authorized to connect to server
```

The server console displays the following message:

```
AUDIT: <Date Time Year>: X: client 6 rejected from IP 129.144.152.193 port 3485
    Auth name: MIT-MAGIC-COOKIE-1
```

The two types of access control mechanisms are: *user-based* and *host-based*. Unless the `-noauth` option is used with `openwin`, both the user-based access control mechanism and the host-based access control mechanism are active. See "Manipulating Access to the Server" on page 97 for more information.

User-Based

A user-based, or authorization-based mechanism allows you to give access explicitly to a particular user on any host. The user's client passes authorization data to the server. If the data matches the server's authorization data, the user obtains access.

Host-Based

A host-based mechanism is a general purpose mechanism. It allows you to give access to a particular host, such that all users on that host can connect to the server. This is a weak form of access control; if a host has access to the server, all users on that host can connect to the server.

The Solaris environment provides the host-based mechanism for backward compatibility. Applications linked with a version of `Xlib` older than OpenWindows Version 2 or X11R4 do not recognize the new user-based access

control mechanism. To enable these applications to connect to the server, a user must either switch to the host-based mechanism, or relink with the newer version of `Xlib`.

Note – If possible, clients linked with an older version of Xlib should be relinked with a newer version of Xlib. This enables them to connect to the server with the new user-based access control mechanism.

Authorization Protocols

The OpenWindows environment supports two different authorization protocols: MIT-MAGIC-COOKIE-1 and SUN-DES-1. While they differ in the authorization data used, they are similar in the access control mechanism used.

The MIT-MAGIC-COOKIE-1 protocol, using the user-based mechanism, is the OpenWindows environment default.

MIT-MAGIC-COOKIE-1

The MIT-MAGIC-COOKIE-1 authorization protocol was developed by the Massachusetts Institute of Technology (MIT). A *magic cookie* is a long, randomly generated binary password. At server startup, the magic cookie is created for the server and the user who started the system. On every connection attempt, the user's client sends the magic cookie to the server as part of the connection packet. This magic cookie is compared with the server's magic cookie. The connection is allowed if the magic cookies match, or denied if they do not match.

SUN-DES-1

The SUN-DES-1 authorization protocol was developed by Sun Microsystems. It is based on Secure Remote Procedure Call (RPC) and requires Data Encryption Software (DES) support. The authorization data is the machine-independent netname, or network name, of a user. This data is encrypted and sent to the server as part of the connection packet. The server decrypts the data, and, if the netname is known, allows the connection.

The SUN-DES-1 authorization protocol provides a higher level of security than the MIT-MAGIC-COOKIE-1 protocol. There is no way for another user to use your machine-independent netname to access a server, but it is possible for another user to use the magic cookie to access a server.

This protocol is available only in libraries in the OpenWindows Version 3 and later environments. Any applications built with static libraries, in particular Xlib, in environments prior to OpenWindows Version 3 cannot use this authorization protocol.

"Allowing Access When Using SUN-DES-1" on page 99 describes how to allow another user access to your server by adding their netname to your server's access list.

Changing the Default Authorization Protocol

The default authorization protocol, MIT-MAGIC-COOKIE-1, can be changed to another supported authorization protocol or to no user-based access mechanism at all. The default is changed by supplying options with the `openwin` command. See the `openwin(1)` man page for more information.

For example, to change the default from MIT-MAGIC-COOKIE-1 to SUN-DES-1, start the OpenWindows environment as follows:

```
example% openwin -auth sun-des
```

If you must run OpenWindows without the user-based access mechanism, use the `-noauth` command line option.

```
example% openwin -noauth
```

Warning – Using `-noauth` weakens security. It is equivalent to running OpenWindows with only the host-based access control mechanism; the server inactivates the user-based access control mechanism. Anyone who can run applications on your local machine will be allowed access to your server.

Manipulating Access to the Server

Unless the `-noauth` option is used with `openwin` (see "Changing the Default Authorization Protocol" on page 96), both the user-based access control mechanism and the host-based access control mechanism are active. The server first checks the user-based mechanism, then the host-based mechanism. The default security configuration uses MIT-MAGIC-COOKIE-1 as the user-based mechanism, and an empty list for the host-based mechanism. Since the host-based list is empty, only the user-based mechanism is effectively active. Using the `-noauth` option instructs the server to inactivate the user-based access control mechanism and initializes the host-based list by adding the local host.

You can use either of two programs to change a server's access control mechanism: `xhost` and `xauth`. For more information, see the `man` pages under `xhost` and `xauth`.. These programs access two binary files created by the authorization protocol. These files contain session-specific authorization data. One file is for server internal use only. The other file is located in the user's `$HOME` directory:

`.Xauthority` (Client Authority File)

Use the `xhost` program to change the host-based access list in the server. You can add hosts to, or delete hosts from the access list. If you start with the default configuration–an empty host-based access list–and use `xhost` to add a machine name, you lower the level of security. The server allows access to the host you added, as well as to any user specifying the default authorization protocol. See "Host-Based" on page 94 for an explanation of why the host-based access control mechanism is considered a lower level of security.

The `xauth` program accesses the authorization protocol data in the `.Xauthority` client file. You can extract this data from your `.Xauthority` file so that other users can merge the data into their `.Xauthority` file, thus allowing them access to your server, or to the server to which you connect.

See "Allowing Access When Using MIT-MAGIC-COOKIE-1" on page 99 for examples of how to use `xhost` and `xauth`.

6

Client Authority File

The client authority file is `.Xauthority`. It contains entries of the form:

connection-protocol *auth-protocol* *auth-data*

By default, `.Xauthority` contains MIT-MAGIC-COOKIE-1 as the *auth-protocol*, and entries for the local display only as the *connection-protocol* and *auth-data*. For example, on host *anyhost*, the `.Xauthority` file may contain the following entries:

anyhost:0	MIT-MAGIC-COOKIE-1	*82744f2c4850b03fce7ae47176e75*
localhost:0	MIT-MAGIC-COOKIE-1	*82744f2c4850b03fce7ae47176e75*
anyhost/unix:0	MIT-MAGIC-COOKIE-1	*82744f2c4850b03fce7ae47176e75*

When the client starts up, an entry corresponding to the *connection-protocol* is read from `.Xauthority`, and the *auth-protocol* and *auth-data* are sent to the server as part of the connection packet. In the default configuration, `xhost` returns an empty host-based access list and states that the authorization is enabled.

If you have changed the authorization protocol from the default to SUN-DES-1, the entries in `.Xauthority` contain SUN-DES-1 as the *auth-protocol* and the netname of the user as the *auth-data*. The netname is in the following form:

unix.*userid@NISdomainname*

For example, on host, *anyhost* the `.Xauthority` file may contain the following entries:

anyhost:0	SUN-DES-1	“unix.*15339@EBB.Eng.Sun.COM*”
localhost:0	SUN-DES-1	“unix.*15339@EBB.Eng.Sun.COM*”
anyhost/unix:0	SUN-DES-1	“unix.*15339@EBB.Eng.Sun.COM*”

where unix.*15339@EBB.Eng.Sun.COM* is the machine-independent netname of the user.

Note – If you do not know your network name, or machine-independent netname, ask your system administrator.

Allowing Access When Using MIT-MAGIC-COOKIE-1

If you are using the MIT-MAGIC-COOKIE-1 authorization protocol, follow these steps to allow another user access to your server.

1. **On the machine running the server, use `xauth` to extract an entry corresponding to *hostname:0* into a file.**
   For this example, *hostname* is *anyhost* and the file is *xauth.info*.

```
myhost% $OPENWINHOME/bin/xauth nextract - anyhost:0 > $HOME/xauth.info
```

2. **Send the file containing the entry to the user requesting access (using Mail Tool, `rcp`, or some other file transfer protocol).**

Note – Mailing the file containing your authorization information is a safer method than using `rcp`. If you do use `rcp`, do *not* place the file in a directory that is easily accessible by another user.

3. **The other user must merge the entry into their `.Xauthority` file.**
   For this example, *userhost* merges `xauth.info` into their `.Xauthority` file.

```
userhost% $OPENWINHOME/bin/xauth nmerge - < xauth.info
```

Note – The *auth-data* is session-specific; therefore, it is valid only as long as the server is not restarted.

Allowing Access When Using SUN-DES-1

If you are using the SUN-DES-1 authorization protocol, follow these steps to allow another user access to your server.

1. **On the machine running the server, use `xhost` to make the new user known to the server.**
   For example, to allow new user *somebody* to run on *myhost*, type:

```
myhost% xhost + somebody@
```

2. **The new user must use** `xauth` **to add the entry into their** `.Xauthority` **file.**
   For this example, the new user *somebody*'s machine-independent netname is unix.*15339@EBB.Eng.Sun.COM*.

```
userhost% echo 'add myhost:0 SUN-DES-1 "unix.15339@EBB.Eng.Sun.COM"' | $OPENWINHOME/bin/xauth
```

Running Clients Remotely, or Locally as Another User

X clients use the value of the `DISPLAY` environment variable to get the name of the server to which they should connect.

To run clients remotely, or locally as another user, follow these steps:

1. **On the machine running the server, allow another user access.**
   Depending on which authorization protocol you use, follow the steps outlined in either "Allowing Access When Using MIT-MAGIC-COOKIE-1" on page 99 or "Allowing Access When Using SUN-DES-1" on page 99.

2. **Set** `DISPLAY` **to the name of the host running the server.**
   For this example, the host is *remotehost*.

```
myhost% setenv DISPLAY remotehost:0
```

3. **Run the client program.**
   The client is displayed on the remote machine, *remotehost*.

```
myhost% client_program&
```