NAME

X security − X display access control

SYNOPSIS

X provides mechanism for implementing many access control systems. Release 6 includes five mechanisms:

Host Access	Simple host-based access control.
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.

ACCESS SYSTEM DESCRIPTIONS

Host Access

Any client on a host in the host access control list is allowed access to the X server. This system can work reasonably well in an environment where everyone trusts everyone, or when only a single person can log in to a given machine, and is easy to use when the list of hosts used is small. This system does not work well when multiple people can log in to a single machine and mutual trust does not exist. The list of allowed hosts is stored in the X server and can be changed with the *xhost* command. When using the more secure mechanisms listed below, the host list is normally configured to be the empty list, so that only authorized programs can connect to the display.

MIT-MAGIC-COOKIE-1

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; *xdm* generates such cookies automatically when this form of access control is used. The user’s copy of the cookie is usually stored in the *.Xauthority* file in the home directory, although the environment variable **XAUTHORITY** can be used to specify an alternate location. *Xdm* automatically passes a cookie to the server for each new login session, and stores the cookie in the user file at login.

The cookie is transmitted on the network without encryption, so there is nothing to prevent a network snooper from obtaining the data and using it to gain access to the X server. This system is useful in an environment where many users are running applications on the same machine and want to avoid interference from each other, with the caveat that this control is only as good as the access control to the physical network. In environments where network-level snooping is difficult, this system can work reasonably well.

XDM-AUTHORIZATION-1

Sites in the United States can use a DES-based access control mechanism called XDM-AUTHORIZATION-1. It is similar in usage to MIT-MAGIC-COOKIE-1 in that a key is stored in the *.Xauthority* file and is shared with the X server. However, this key consists of two parts - a 56 bit DES encryption key and 64 bits of random data used as the authenticator.

When connecting to the X server, the application generates 192 bits of data by combining the current time in seconds (since 00:00 1/1/1970 GMT) along with 48 bits of "identifier". For TCP/IP connections, the identifier is the address plus port number; for local connections it is the process ID and 32 bits to form a unique id (in case multiple connections to the same server are made from a single process). This 192 bit packet is then encrypted using the DES key and sent to the X server, which is able to verify if the requestor is authorized to connect by decrypting with the same DES key and validating the authenticator and additional data. This system is useful in many environments where host-based access control is inappropriate and where network security cannot be ensured.

SUN-DES-1

Recent versions of SunOS (and some other systems) have included a secure public key remote procedure call system. This system is based on the notion of a network principal; a user name and NIS domain pair. Using this system, the X server can securely discover the actual user name of the requesting process. It involves encrypting data with the X server’s public key, and so the identity of the user who started the X server is needed for this; this identity is stored in the *.Xauthority* file. By extending the semantics of "host address" to include this notion of network principal, this form of access control is very easy to use.

To allow access by a new user, use *xhost*. For example,

xhost keith@ ruth@mit.edu

adds "keith" from the NIS domain of the local machine, and "ruth" in the "mit.edu" NIS domain. For keith or ruth to successfully connect to the display, they must add the principal who started the server to their *.Xauthority* file. For example:

xauth add expo.lcs.mit.edu:0 SUN-DES-1 unix.expo.lcs.mit.edu@our.domain.edu

This system only works on machines which support Secure RPC, and only for users which have set up the appropriate public/private key pairs on their system. See the Secure RPC documentation for details. To access the display from a remote host, you may have to do a *keylogin* on the remote host first.

MIT-KERBEROS-5

Kerberos is a network-based authentication scheme developed by MIT for Project Athena. It allows mutually suspicious principals to authenticate each other as long as each trusts a third party, Kerberos. Each principal has a secret key known only to it and Kerberos. Principals includes servers, such as an FTP server or X server, and human users, whose key is their password. Users gain access to services by getting Kerberos tickets for those services from a Kerberos server. Since the X server has no place to store a secret key, it shares keys with the user who logs in. X authentication thus uses the user-to-user scheme of Kerberos version 5.

When you log in via *xdm*, *xdm* will use your password to obtain the initial Kerberos tickets. *xdm* stores the tickets in a credentials cache file and sets the environment variable *KRB5CCNAME* to point to the file. The credentials cache is destroyed when the session ends to reduce the chance of the tickets being stolen before they expire.

Since Kerberos is a user-based authorization protocol, like the SUN-DES-1 protocol, the owner of a display can enable and disable specific users, or Kerberos principals. The *xhost* client is used to enable or disable authorization. For example,

xhost krb5:judy krb5:gildea@x.org

adds "judy" from the Kerberos realm of the local machine, and "gildea" from the "x.org" realm.

THE AUTHORIZATION FILE

Except for Host Access control, each of these systems uses data stored in the *.Xauthority* file to generate the correct authorization information to pass along to the X server at connection setup. MIT-MAGIC-COOKIE-1 and XDM-AUTHORIZATION-1 store secret data in the file; so anyone who can read the file can gain access to the X server. SUN-DES-1 stores only the identity of the principal who started the server (unix.*hostname@domain* when the server is started by *xdm*), and so it is not useful to anyone not authorized to connect to the server.

Each entry in the *.Xauthority* file matches a certain connection family (TCP/IP, DECnet or local connections) and X display name (hostname plus display number). This allows multiple authorization entries for different displays to share the same data file. A special connection family (FamilyWild, value 65535) causes an entry to match every display, allowing the entry to be used for all connections. Each entry additionally contains the authorization name and whatever private authorization data is needed by that authorization type to generate the correct information at connection setup time.

The *xauth* program manipulates the *.Xauthority* file format. It understands the semantics of the connection families and address formats, displaying them in an easy to understand format. It also understands that SUN-DES-1 and MIT-KERBEROS-5 use string values for the authorization data, and displays them appropriately.

The X server (when running on a workstation) reads authorization information from a file name passed on the command line with the −*auth* option (see the *Xserver* manual page). The authorization entries in the file are used to control access to the server. In each of the authorization schemes listed above, the data needed by the server to initialize an authorization scheme is identical to the data needed by the client to generate the appropriate authorization information, so the same file can be used by both processes. This is especially useful when *xinit* is used.

MIT-MAGIC-COOKIE-1

This system uses 128 bits of data shared between the user and the X server. Any collection of bits can be used. *Xdm* generates these keys using a cryptographically secure pseudo random number generator, and so the key to the next session cannot be computed from the current session key.

XDM-AUTHORIZATION-1

This system uses two pieces of information. First, 64 bits of random data, second a 56 bit DES encryption key (again, random data) stored in 8 bytes, the last byte of which is ignored. *Xdm* generates these keys using the same random number generator as is used for MIT-MAGIC-COOKIE-1.

SUN-DES-1

This system needs a string representation of the principal which identifies the associated X server. This information is used to encrypt the client's authority information when it is sent to the X server. When *xdm* starts the X server, it uses the root principal for the machine on which it is running (unix.*hostname@domain*, e.g., "unix.expire.lcs.mit.edu@our.domain.edu"). Putting the correct principal name in the *.Xauthority* file causes Xlib to generate the appropriate authorization information using the secure RPC library.

MIT-KERBEROS-5

Kerberos reads tickets from the cache pointed to by the *KRB5CCNAME* environment variable, so does not use any data from the *.Xauthority* file. An empty entry must still exist to tell clients that MIT-KERBEROS-5 is available.

FILES

.Xauthority

SEE ALSO

X(1), xdm(1), xauth(1), xhost(1), xinit(1), Xserver(1)