SAFE USE OF X WINDOW SYSTEM PROTOCOL ACROSS A FIREWALL

Brian L. Kahn
blk@mitre.org

The MITRE Corporation
202 Burlington Road
Bedford, Massachusetts 01730

Abstract

This paper describes a method for safe use of the X window system protocol across a firewall. The approach uses an agent in the communications stream to enforce an access control policy. Topics covered include risks, policy, implementation, and a summary of prototype efforts. Results can be applied to other distributed or client/server applications.

Background

Many organizations are now using or installing a *firewall* to protect their local networks against threats from public and wide area networks. A firewall is positioned between networks and protects resources by controlling access across the boundary. A typical firewall might allow electronic mail to pass through and support Telnet sessions from the protected network to the other side. A firewall allows corporate and institutional users to have controlled access to outside resources, such as the Internet, while protecting their private assets.

Many of these same users would like to use the remote display capabilities of the X window system to run X clients on hosts outside the firewall and display the results on a host inside the firewall. The X window system provides graphical user interface and other display services to clients running on local or remote machines. Unfortunately for the users, there are a number of serious security risks associated with X window system connections across a firewall.

This paper first explains some of these risks. We establish the context for this research, and we define some useful terms. Next, we describe an X Gateway module that is able to control the risks by enforcing restrictions on the X protocol. We describe a policy for X Gateway, both in general and for a specific implementation. A discussion of our experience with a prototype of the X Gateway completes the paper.

Firewall Protection

Connecting systems together in a network has large benefits, but this also creates a large number of risks. A network is a shared medium, so there is a risk of eavesdropping or data capture. Network services allow interaction between the systems on the network, but they are vulnerable to spoofing, denial of service, and penetration attacks. Before an organization allows systems to be connected together into a network, there should be an assessment of the risks and benefits. For many organizations, there is no need for an explicit risk assessment because a level of trust is given to the users just by being allowed on the premises. In such an organization, there may be a few sensitive systems which are not networked, but the physical plant security is adequate to deal with the risks of internal networking.

All assumptions change when networks are connected together, especially when an organization's private network is connected to a public network. The resources which become available to the user community may be very tempting, but the vulnerability must be carefully considered. A firewall can provide protection for a network while allowing some resources on the two networks to be shared.

A firewall provides controlled communications between two points and is usually used to regulate traffic between networks. A firewall is typically built from network components, such as routers and dual-headed boundary hosts, configured to partially isolate the private network. Such a firewall can provide access across a boundary with reasonably low cost and processing overhead.

However, firewall is not a panacea. The main problem with using a firewall for protection is the course granularity of control. A filtering router can be programmed to block packets based on the source or destination address and low level protocol type. A connection layer protocol may serve many different purposes, some of which are desired and some of which are not. A boundary host can be employed to provide application proxy service, or clients, to let the firewall pass specific services discriminated at a much higher level of the protocol stack. For example, the client and server implementing File Transfer Protocol (FTP) use the TCP/IP protocol. FTP service can be provided by the boundary host at the application level even though general TCP/IP use is blocked at the firewall.

A Context for this Research

A company operates a distributed network of personal computers, workstations, and servers for the computing needs of technical and support staff. The Corporate LAN Interconnection Network (CLIN) spans the two main sites and several remote sites. The configuration is always changing and growing. The CLIN also connects to the Internet public network through several routers and 56Kb high speed modems.

The firewall filters the network traffic, blocking many packets but allowing certain protocols and connections to pass through. The CLIN routers form a *security perimeter* which defines the edge of the firewall to protect the rest of the CLIN against misuse or intrusion by Internet users. The complete firewall consists of the routers and their routing tables, the physical layout of the CLIN, several dual-headed boundary hosts, and corporate policy covering network access by company employees.

The company firewall design and policy prohibits direct communication across the firewall between a company workstation and an outside host. This protects the company network, but it also constrains access to the Internet. To support access to public resources, a company boundary host provides application level proxies for ftp, Telnet, and mosaic.

The X window system is a particularly interesting case. The X protocol enables software (the X client) to run on one machine while using the screen and keyboard of another machine (the X server). The X protocol allows X clients to share the keyboard, mouse, and screen of the host running an X server. The clients may be on the same machine as the X server or on a remote machine. Enabling X connections across the firewall would allow an employee to sit at a company workstation and see the graphical output from software running on a super-computer at a university. The current company security policy does not allow employees to start an X client on an outside host and display the client's output on an X server within the firewall because of concerns about the security risks posed by X connections with the outside.

The Problem with X Across the Firewall

Enabling X window connections across a firewall is straightforward and well understood. The X protocol requires a reliable transport connection, which in practice is either DECNet or TCP/IP between machines. To move X packets across a firewall, it is only necessary to start a proxy running on a boundary host that listens at an agreed port and forwards the packets to the intended X server. The outside client sees the boundary host as the X server host, and the client neither knows nor cares that the X packets are being forwarded inside the firewall. Enabling X across the firewall is not a problem; the problem is making a safe way for outside clients to share the X server with inside clients.

Here is a brief description of the risks related to X. Later in this paper, a more detailed description of X vulnerabilities is given:

- The X window system has no access control beyond client/server connection time. After a client connects to the server, it has full access to all X objects and resources.
- X clients share the keyboard, mouse, display screen, and an operating environment containing graphical objects and data items.
- An X client may elect to receive the user input from any X window, so it is easy for one client to eavesdrop on keystrokes intended for another client.
- A client can view or draw into the windows created by any other client, capturing or faking output.
- One X client may even change the run-time behavior of another client by binding a different function to a button or action.

To summarize the risks, consider this simplified description of the X window system environment—all X clients have access to the screen, keyboard, and mouse of the X server including any objects created by other X clients.

The X Gateway Solution

The risks associated with an X protocol connection can be controlled by restricting the capabilities of the X client. A large set of X clients are self-contained, in that they do not need to interact with other X clients or the resources created by other X clients. These clients can be considered "safe" because they do not interfere with any other client. This set of clients includes many useful programs, and it is very desirable to allow these clients (but only these clients) to cross the firewall.

The X Gateway enables safe X clients and disables clients that engage in "dangerous" behavior. The X gateway is a software module which resides on a boundary host. Clients outside of the firewall see the X Gateway as an X server running on the boundary host. The X Gateway actually acts as a proxy agent, forwarding packets between an X server within the firewall and the outside X clients.

The X Gateway enforces a security policy on the client/server channel that isolates the outside X clients from the inside clients. The X Gateway does this by imposing restrictions on use of the X protocol. Each packet in the X protocol channel across the firewall is examined by the X gateway to ensure adherence to the isolation policy. The external client connecting through the X gateway does not have all the capabilities of an inside client, but the available functionality is adequate for safe applications.

Terms and Concepts

The X window system is an architecture independent system for display of graphical user interfaces (GUIs). The X window system is created, maintained, and freely distributed by MIT and the X Consortium. An X server is a display and keyboard controller that implements the X protocol. An X client is a program that uses the X protocol to communicate with an X server. The server provides display and input handling services to the client. The X server runs on the computer attached to the console display screen, keyboard, and mouse. The X client may run on the same machine as the X server or on another machine. The X protocol is a two way exchange between an X client and an X server over some reliable transport, such as TCP/IP. The X client initiates all X protocol connections.

The X protocol constraint enforcement software is referred to as the X Gateway or Xgate. The X Gateway allows X clients running on hosts outside the company security perimeter to display on X server console screens within the company firewall perimeter. A client connected to a server through the X Gateway is called an X Gateway client or an outside client.

The user is a human seated at the X display console. A host located within the company firewall and connected to the CLIN is called a company host. A host used by the company as part of the firewall (connected to the company network and the outside network) is a boundary host. An X client running on a company host is an inside client, and an X client running on an outside host is an outside client.

An *X object* in this document is a resource referenced by an identifier known as an XID. An X object is a data structure created and held by the X server. *Client objects* are created in response to a client request, using an ID provided by the client. Client objects are windows, pixmaps, cursors, graphics contexts, and client colormaps. Properties are considered to be named attributes of a window. *Server objects* are created by the server for common use by all clients. Server objects are fonts, the default colormap, and the root window. The server also maintains lists and structures to control server operations. The following server resources are referred to as server objects in this document: the host list, the save list, the key and button grab list, the keymap, the keyboard state, and the cursor position. There may be other server resources not listed here, depending on server implementation. Most fonts are loaded in response to a client request, but they are not considered to be client objects.

The *client ID* is an identifier assigned to a client by the X server at client/server connection time. The server permanently associates the client ID with the communication channel being used by the client. A XID is a 32 bit identifier used within the X protocol to identify the objects managed by the X server. XIDs uniquely identify objects. The XID is provided by the client in the request to create an object. The X server ensures that the high bits of a XID match the client ID associated with the channel and that the ID is not already in use. Thus, each XID reliably indicates which client created the object. All remaining client objects are destroyed when a client disconnects from the server so that clients cannot "inherit" objects from previous clients.

A Review of the Risks

There are many vulnerabilities associated with using the standard X protocol. The general risk areas are described here, and the risk profile is then evaluated to establish a safety policy.

The main risks posed to the company by unrestricted X protocol connections across the firewall are penetration attacks and eavesdropping by an outside agent. A simple example of eavesdropping is an outside X client requesting keystroke events from the window of an inside xterm client in order to steal the password from a Telnet session. A simple example of a penetration attack is an outside X client sending fake mouse button events to an inside mail reader client which causes the inside client to send out some interesting files in email.

Most of the risks result from a total lack of access control within the X environment. The only security decision taken by the X server is whether to allow an X client to make a connection to the server. There are several initial client authorization methods optionally performed at client/server connect time—the method selected depends upon configuration parameters set either by a server start-up database or by a connected client. Once connected, all X clients have equal access to the X objects. Clients can manipulate, modify, or destroy any object (window, scrollbar, etc.) regardless of which client is the owner.

The X server reports events (such as keystrokes or mouse clicks) to any client that registers a request for the event type with regards to a particular window. There is no privacy within X unless a client grabs[1] the keyboard, pointer, or entire server for its exclusive use. In general, every client may eavesdrop and monitor all of the user keystrokes, pointer movement, and mouse button presses regardless of which client or window is actively responding to the user. From the user perspective, the keystrokes and button clicks appear to be picked up by one client application at a time, usually in the window under the mouse. Internally, all events are available to every client that expresses an interest.

Another risk results from abstraction of the input devices, the pointer device ("mouse"), and keyboard. This helps X to be operating system and architecture independent because the physical keyboard and mouse are mapped by software to key symbol events and pointer events. This approach also allows clients to restructure the input system. This can be used to trick the user into "typing" an unintended key sequence or to disable the mouse buttons for denial of service.

Another set of risks stems from the "callback" paradigm adopted by many X client libraries. This approach dynamically binds events to functions within the code. Many X clients accept detailed run-time configuration parameters from a shared database stored in the RESOURCE_MANAGER property on the root window. Some clients will reconfigure themselves in response to a certain kind of message from another client after their initial startup configuration. Client messages are a special event type intended for interclient communications, and there is a protocol called EditRes which allows one client to alter the internal structure of another client. EditRes is part of the basic X tool kit library (Xt). It is used by the Athena widget library and is easily included into other tool kits. For example, xterm from the X11 distribution and Gnu Emacs respond to the EditRes protocol. Run-time binding can change many things in subtle ways, but here is a simple example: a rogue outside client could alter the operation of an inside client by rebinding the Save & Exit panel button to the function exit_without_saving_changes().

Clients may request the server to create and send synthetic events that closely resemble real events, and many clients will respond to these fake events. This leads to a serious penetration risk when there are command shell clients such as xterm, hpterm, decterm, or commandtool running and connected to the display. A rogue outside client can send fake keystrokes to an inside client and cause arbitrary commands, or programs, to be run on the system running the inside client, just as if the user had typed the keys. This vulnerability extends well beyond the client's running command shells because any sequence of key presses and pointer events can be sent to any inside client. This vulnerability may be used with any client to effect penetration or other types of attacks. Most of the shell tools in the xterm

[1] Use of the X grab functions tends to hog the server and is understandably considered to be antisocial behavior for any extended period, so well-behaved clients only issue a grab for short term events like popping menus or dialogue panels.

family ignore synthetic events by default, but this risk can be combined with the reconfiguration risks described in the previous paragraph to make current or future clients respond to synthetic events.

Combinations of attacks using the standard X protocol risk the use of company resources and permissions to mount attacks from one outside host against another outside host. The problem is most easily expressed in a scenario: if user, JohnA, starts up clients on two outside machines, SponsorHostB and CollegeHostC, then another user, JaneD, may be able to launch an attack against SponsorHostB from CollegeHostC by exploiting shared resources through the X Gateway.

A denial of service attack is easy to carry out but difficult to counter in a generic way. X clients routinely grab the server for exclusive use when menus are dropped or dialogue panels are raised, so it is not an abnormal occurrence. Besides this, there are a number of ways to prevent other clients from engaging in normal X operations. However, this attack is not a serious concern for company users because recovery from a worst case scenario only requires restarting the X server.

X Gateway - Policy and Implementation

This section describes the policy developed for the X Gateway and presents issues relevant to building our Xgate prototype. The policy is presented in three ways: a high level statement of seven policy objectives, a longer explanation of each policy objective, and design statements reflecting how each objective is supported in the prototype implementation. Experience with the Xgate prototype is briefly summarized. There is also some discussion of policy alternatives and connections from inside clients to outside servers.

Developing a security policy for X is difficult due to a great complexity in the protocol (over 120 packet types), together with complex interactions between clients and objects. The following policy was developed, after several false starts, by starting with a simple policy that describes a well-behaved client and elaborating with more detailed clauses to cover the peculiarities of the X window system functionality.

It is not always possible or practical to add security after the fact, and it is almost always more difficult than addressing security during the design phase. The X protocol is suited for an in-line policy enforcement module because the information needed for an access control decision is available within each X protocol packet. The operation indicated by a packet is determined by the packet type and some of the parameters. The client invoking the operation is determined by the point-to-point communication channel (typically TCP/IP). The objects of the operation are, in most cases, explicitly identified in the packet, and the XID used to identify each object also indicates which client created that object. The successful X gateway prototype suggests that it is sometimes practical to add security to a client/server relation by using a constraint processor in-line with the protocol stream.

The X Gateway policy is based on one fundamental observation: most clients do not need all of the X protocol functionality. The X Gateway is considered a success if it enables many of the programs which users want to run across the firewall. Most X clients are only interested in their own windows and objects; and, except for the standard cut-and-paste mechanism, most clients make no attempt to interact with other clients.

X Gateway Client Isolation Policy

The X Gateway policy describes isolation of outside clients.

1. Client object access: X Gateway shall isolate outside clients by restricting object use. An outside client may not use objects created by any other client.

2. Server object access: X Gateway shall protect normal operations by restricting access from outside clients to server state or control objects.

3. Selection requests: X Gateway shall protect the client selection mechanism by restricting use of the interclient exchange initiated by ConvertSelection.

4. User Notification: X Gateway shall notify the user of significant changes that are allowed by policy but significantly affect the access control profile.

5. Image Capture: X Gateway shall prohibit outside clients from accessing portions of the screen image generated by other clients.

6. Denial of Service: X Gateway shall provide some counter to denial of service attacks.

7. Audit: X Gateway shall provide a mechanism for audit security-relevant events and should also support audit of normal X protocol and audit reduction.

Explanation of Client Isolation Policy

The X Gateway policy describes isolation of outside clients in respect to objects created by other clients, server objects, the physical screen, and control flow. Note that this policy does not restrict inside clients in any way.

1. The primary clause of the policy is the restriction on interactions between clients. Window managers need to interact with other clients a great deal, and so, it is not possible to run a window manager as a remote client through the X Gateway. Most X applications do not interact with other clients, with the notable exception of the cut-and-paste exchange. Indeed, the loss of the cut-and-paste capability from outside clients to inside clients is the greatest drawback of the X Gateway policy. This issue is addressed below in Policy Alternatives.

2. The server creates a number of objects shared by many clients, notably the root window, the default colormap, and character fonts.[2] There are also a number of configuration structures maintained by the server that affect various operations. It is possible to write an X Gateway that captures references to the default server objects and redirects these to X Gateway objects, but allowing selective access to server objects has the advantage of lesser complexity and code size.

3. The X selection mechanism is intended for interclient communications. The cut-and-paste exchange is the most common use for selections. The problem with cut-and-paste (or other selections) is that exchange goes on between clients and may not have been initiated or intended by the user. The X Gateway controls when an outside client may request a paste, but it does not control the paste action itself which is performed by the inside client.

4. User notification is a general-purpose salve for areas that are sensitive to exploitation but cannot be avoided. Two areas are called out here; but other situations may be handled the same, depending upon the policy most appropriate for the user site.

   Most X servers support a number or mechanisms for authentication of client identity at client/server connection time. Unfortunately, strong authentication mechanisms may be a burden to administer or may not be supported by every server, and many sites or users will end up with one of the weaker mechanisms. It is considered worthwhile to alert the user (or to obtain user approval) before any new client connects via the X Gateway.

   The keyboard focus is held by a single window, and any key stroke events are associated with that window. Most of the time, the focus is transferred by the window manager when the user moves the mouse pointer or presses a key sequence requesting a focus change. The X server will also change the focus in response to a client request, and the user may not be aware that key events are channeled to an outside client without a visible indicator.

   Some implementations may rely on the window manager (WM) to indicate keyboard focus. The WM usually indicates the keyboard focus through a change in the window border or title bar. The disadvantage of this approach is the implied trust of the WM, which is complex software.

5. Clients may obtain images from the display in two ways by inheriting an image of the screen at window creation or by querying the server for a copy of some portion of the screen. Few clients need this capability,

[2] Atoms are a relation between strings and identifiers maintained by the X server. Atoms may appear to be an important object type or resource, but the X Gateway is not concerned with them. Atoms can only be created, never changed or destroyed, so their information content is very small.

and we chose to preclude this as a matter of policy. On the other hand, some applications (for example, several groupware programs) may have good reason to extract images with other clients. Support for such clients would call for a change in this policy clause.

6. Denial of service is difficult to prevent, as a matter of policy, without disrupting the operation of the client's user interface. It is not uncommon for well-behaved clients to grab complete control over the keyboard, pointer, or entire server for brief periods. The X Gateway may be implemented to block or immediately release any explicit or implicit server grabs while maintaining the correct keyboard focus and visual effect. Another approach is to provide the user a way out when under attack by providing a kind of trusted path to the X Gateway with options to kill off offending processes or specific server grabs.

7. Audit is an important, and often overlooked, property of security software. The determination of which events are interesting to audit is open for debate—certainly, new client connections but possibly, violations of policy. It should be understood that many clients will make minor violations of the policy that are blocked or countered by the X Gateway.

Implementation of Client Isolation Policy

Here are details of the simple client isolation policy with minimal extensions. Implementation clauses marked with a square bracket are not yet supported in the Xgate prototype.

1. Client object access: check the high bits of all XIDs used as parameters.

   a. A match with the client ID indicates that the client created the object allowed.

   b. Server objects have a unique value in place of the client ID (usually zero), and these are covered in clause 2, or the request is blocked.

2. Server object access: restrict access to objects that cannot contain sensitive information.

   a. All references to fonts and the default color map are allowed.

   b. The root window may be used as the parent in CreateWindow request or as the drawable in CreateGC and CreatePixmap requests.

   c. The cursor position may be accessed normally.

   d. Server grabs are prohibited.

   e. The keymap and keyboard state may not be accessed.

   *Sanitize response to QueryKeymap request.

   *KeymapNotify events are blocked or sanitized.

   f. The host list and the access control mode may not be accessed.

   g. The root window may not be used in any other way including window properties, redirected events, image queries, etc.

3. Xgate will block ConvertSelection requests when the client is not the selection owner.

   a. Simply checking ownership with GetSelectionOwner creates a race condition.

   b. Xgate may return a NoOwner error for all ConvertSelection requests.

   c. Xgate may use the following protocol in place of a ConvertSelection request.

   *GrabServer and GetSelectionOwner.

   *If owner is the outside client, then do ConvertSelection or else return NoOwner.

   *UngrabServer.

d. Xgate may change the selection type to a selection held by Xgate.

*Xgate may act as an intermediary for a interclient subpolicy.

*Site policy may allow a cut-and-paste with explicit user authorization.

4. Xgate notifies the user when new clients connect and when the keyboard events are being associated with an outside client window.

   a. Xgate posts a yes/no query button asking the user for permission to initiate the connection for each client, identifying the IP address of the client's host.

   b. Xgate provides a visual indication when keyboard focus is held by an outside client.

   *A small, simple focus indicator graphic is shown when key events are going out.

   +Xgate will request FocusChange events on all windows of all outside clients.

   +Focus indicator is kept visible whenever it is being shown (mapped).

   +Repeated obscuring of the focus indicator is a security-relevant event.

   *The focus indicator is posted when key events are sent to an outside client if the focus is held by root (this happens if the client is under the pointer).

   *These rules are fully enforced during all grabs.

5. Xgate will restrict explicit and implicit pixel reads.

   a. New window creates requests with a parent value of ROOT, and a background pixel value of NONE will be modified to a background of BLACK_PIXEL or PARENT_RELATIVE.

   b. Explicit reads are prohibited by clauses 1 and 2.

6. Xgate provides an escape for the user.

   a. Event stream is monitored during keyboard grabs.

   b. Xgate "hot key" sequence to kill one or all Xgate clients.

7. The audit mechanism is two tiered.

   a. Xgate logs client connections and terminations directly to a file.

   b. Xgate performs configurable audit reduction.

   *Xgate accepts audit rules on stdin; generates audit log on stdout.

   *Five levels of detail in the formatted output.

   *Logging level can be set for the four major packet groups and for individually identified packets.

   *Supports selective blocking of packets by type.

   *Xgate provides a GUI interface to the daemon for ease of use.

Xgate Prototype

The MITRE Corporation has built a prototype of the X Gateway module named Xgate. This section describes the approach taken and summarizes lessons learned from the effort.

The MITRE Xgate prototype is built on top of a freely available package named Xmon.[3] The purpose of Xmon is to monitor the X protocol stream and audit packets at the selected level of detail. A developer can watch the X protocol

[3] Available on several Internet archive sites. One URL is ftp://ftp.uu.net/Usenet/comp.sources.x/volume9/Xmon.

packets go past and determine exactly what is happening at any point. This is useful, both for debugging problems and for determining how X clients are using the X server. Each of the 120 plus packet types can be individually selected for audit using a graphical user interface, and Xmon can also block transmission of selected packet types. These built-in capabilities make Xmon a good choice for a prototyping effort.

The Xgate prototype required approximately 12 weeks of staff effort, generating close to 6000 lines of code in 150KB of file space. The design analysis, comprising the bulk of this paper, required a similar investment of staff time.

Once development moves beyond the prototype stage it may be possible to increase efficiency by trimming back the code. We expect to improve packet latency, throughput, and resource utilization in later implementations. After the details of the policy and design decisions are well settled, we expect to recode an Xgate module from scratch or rebuild on top of a simpler code base. Xroute is another publicly available package to consider at that point, a tiny X connection router which simply bounces X packets from one machine to another.

Experience with the prototype led to changes in the X Gateway policy interpretation. We used the flexible auditing capabilities inherited from Xmon to examine the specific use of the X protocol by various popular X clients. We observed a number of policy violations by X clients that were not related to any kind of attack.

We had planned to limit Xgate processing to requests flowing from the clients to the server. It is possible to enforce policy simply by blocking some requests. Some implementers may choose to build an X Gateway to only monitor and filter the client-to-server stream, as we originally intended, resulting in simpler and faster code. Our early tests showed us, however, that some clients make a few policy violations but should be allowed to run because these are basically safe clients with no intent to subvert or entangle other clients. We chose to manipulate the query/response stream to control the behavior of certain important clients, "persuading" these clients to act within the policy bounds.

Some well known X clients violate a simple interpretation of the policy to cover unusual situations or make an application more robust. In example, the Xv image viewer client (often used with Mosaic) examines the top level windows of other clients in an attempt to discover which window manager is in use. Blocking the offending packets tends to kill the applications, yet there is no threat from the clients themselves. We figured out several fixed content messages that can be returned to the client without compromising the policy. The specifics are described in the following section, "Implementation Extensions to Enhance Compatibility." The disadvantage in this approach is a small amount of extra code inside Xgate. There is also a performance penalty, because this manipulation of the protocol requires Xgate to scan both the incoming request stream and the outgoing reply/event/error stream.

Particular attention should be given to the documentation and structure of the Xgate software. Confidence in the analysis phase is crucial because the Xgate implementation is a security critical component.

Performance is a concern for our users, but we do not see any problem with Xgate in place. On an Ethernet based network, there is a slight, but noticeable, difference between a plain application and one running through Xgate. Our experience indicates that network connections (such as a T1 link) impose more delay than the Xgate prototype. Furthermore, the prototype has not yet been tuned for performance, yet its performance is more than adequate for our users. Unfortunately for this discussion, performance under X windows is very difficult to measure, both in selection of meaningful tests and interpretation of test results.

There is no perceptible difference in response time with or without Xgate for our typical user: an X connection across an Ethernet network between two Sun SPARC workstations which is creating and destroying complex widgets. There is a slight difference in response for this same client connected to a local server (no network hop) when compared with the client connected to the local server via Xgate. The conclusion is that the latency delay imposed by Xgate is hidden by the latency imposed by a local area network.

Tests performed with X11perf (a server test tool from the X11 distribution) shows that Xgate causes an approximate 7% decrease in maximum performance for a mix of X11 actions. Note that X11perf determines the client/server round trip time and extracts that time from the test results. The results below compare Xroute (a simple X forwarder) with Xgate. These results are of interest but may not apply to an X client running the X server at less than full capacity. The load on the Sun SPARC running the Xgate module ranged from 2% to 11%.

```
xgate[129] time x11perf -display boundary:1 -repeat
1 -f8text -popup -scroll100 -copypixwin100
-fspellipse100 -ostrap100

x11perf - X11 performance program, version 1.3
MIT X Consortium server on security:1.0
from vanity
Wed Oct 19 18:28:32 1994
Sync time adjustment is 10.8001 msecs.
 18000 reps @  0.2856 msec (3500.0/sec): 100-pixel
fill slice partial ellipse
 20000 reps @  0.2153 msec (4650.0/sec): Fill
100x100 opaque stippled trapezoid
504000 reps @  0.0075 msec (134000.0/sec): Char
in 70-char line (8x13)
 10000 reps @  0.5436 msec (1840.0/sec): Scroll
100x100 pixels
  4000 reps @  1.3480 msec ( 742.0/sec): Copy
100x100 from pixmap to window
 20000 reps @  0.2494 msec (4010.0/sec):
Hide/expose window via popup (25 kids)
 30000 reps @  0.2182 msec (4580.0/sec):
Hide/expose window via popup (100 kids)
    111.6 real      1.1 user      1.7 sys

xgate[130] time x11perf -display boundary:2 -repeat
1 -f8text -popup -scroll100 -copypixwin100
-fspellipse100 -ostrap100

x11perf - X11 performance program, version 1.3
MIT X Consortium server on security:1.0
from vanity
Wed Oct 19 18:32:06 1994
Sync time adjustment is 5.6688 msecs.
 18000 reps @  0.2871 msec (3480.0/sec): 100-pixel
fill slice partial ellipse
 30000 reps @  0.2049 msec (4880.0/sec): Fill
100x100 opaque stippled trapezoid
720000 reps @  0.0074 msec (135000.0/sec): Char
in 70-char line (8x13)
 10000 reps @  0.5396 msec (1850.0/sec): Scroll
100x100 pixels
  4000 reps @  1.8645 msec ( 536.0/sec): Copy
100x100 from pixmap to window
 20000 reps @  0.2583 msec (3870.0/sec):
Hide/expose window via popup (25 kids)
 30000 reps @  0.2221 msec (4500.0/sec):
Hide/expose window via popup (100 kids)
    119.9 real      1.3 user      3.0 sys

  PID TT STAT  TIME SL RE PAGEIN SIZE  RSS
LIM %CPU %MEM COMMAND
24419 p2 S    0:04  0 43     0  168  384   xx 10.5
0.6 xgate

  PID TT STAT  TIME SL RE PAGEIN SIZE  RSS
LIM %CPU %MEM COMMAND
24419 p2 S    0:05  2 72     0  168  384   xx 2.0
0.6 xgate
```

The prototype is not available for public release at this time.

Implementation Extensions to Enhance Compatibility

The X Gateway enforces a security policy by imposing restrictions on the use of the X protocol. There are some X clients which cannot work through Xgate, and this is the desired result: we do not want outside clients to manipulate all the windows as a window manager does, and we do not want outside clients to copy the screen image as the xgrab utility does. Unfortunately, there are expected to be some X clients which users want to use; and yet, do not conform to the Xgate policy. The Xgate prototype includes some extensions which are not needed for security but do enhance compatibility.

Unless otherwise stated, X protocol requests that do not meet the policy requirements are simply dropped from the client/server stream without any error notification. The dropped packets are replaced in the stream by X protocol no operation (NOP) packets in order to maintain synchronization of sequence numbering in the client/server stream.

Blocking requests which normally generate a reply from the server will disrupt operation of the offending client. Some of these clients can be persuaded to run within the policy constraints by simulating a sterile environment. The Xgate may be implemented to allow requests to pass into the X server and then replace the reply with a fixed-content packet that corresponds to "no such object" or otherwise null response. This can only be allowed for informational requests that make no changes to objects or server state. Passing these requests, and then changing the reply packet, works well enough; but it would be conceptually cleaner if

Xgate were to replace the request with a NOP and manufacture a null response. The problem is that replies, events, and errors must be returned in the same order as the requests which generated them, and it is difficult to insert the null reply packet into the right spot in the response stream. Examples of some common requests which are disallowed by the Xgate policy, but may be covered by a null response, are ListProperties, GetProperty, and GetSelectionOwner. This approach may allow normal operation of clients, which would otherwise fail to work with the Xgate, but it must be balanced against the undesirable complexity of additional code in the Xgate module.

The QueryTree request is a candidate for a slightly, more complex, form of censorship. The X windows are organized into a tree with the root window at the root of the tree. The QueryTree request normally returns the parent and a list of the children for the target window. There is no way for sensitive information to be compromised by QueryTree, so the request could be allowed through. However, we have found some popular X clients which check values of properties on the windows of other clients to gather information about the user's environment. Modifying the server response to QueryTree using the following two rules will convince a client that there are no other clients displaying on the server.

1. If a child window does not belong to the requesting client, remove it from the list.

2. If the parent window does not belong to the requesting client, change it to the root window.

Outside clients cannot be allowed to read every property on the root window because some properties may contain information we want to protect. It may be desirable to support the use of the RESOURCE_MANAGER for program defaults (set by xrdb). Some clients may need access to an application specific root window property. Selective access to root window properties, either read-only or read-write, may be simulated by redirecting specific requests to a window owned by Xgate. A trusted cut-and-paste mechanism with explicit user approval could be supported by Xgate using a similar redirection technique.

Policy Alternatives

Several alternatives to the policy described above are reasonable for sites with different priorities or concerns. Care should be taken when making changes to any security policy because changes which seem modest may have subtle interactions with other policy clauses. That said, these alternatives are worth considering. Any of the following can be used to enable cut-and-paste between inside and outside clients, a capability expected to top the list of features requested by users.

1. Allow groups of X Gateway clients to communicate as per standard X.

    a. All outside clients allowed to interact.

    b. A group of clients specifically selected by the user allowed to interact.

    c. Clients on the same remote host allowed to interact.

2. Allow certain inside clients to interact with outside clients.

    a. A trusted cut-and-paste clipboard could be started up by the X Gateway.

    b. The trusted clients must be specially "hardened" to resist attacks via the X protocol.

3. Enhance the client authentication mechanism.

    a. Use a one-time key in place of the XAUTH key because the XAUTH key can be reused by any client from the same host until the X server restarts.

    b. Replace the enhanced authentication string with a standard mechanism (probably XAUTH) when forwarding the connect request to the server.

Policy for Connecting Inside Clients to Outside Servers

The X Gateway enforces policy over X connections which are *inbound*, that is X clients outside of the firewall displaying on a server within the firewall.

Users are also interested in *outbound* X connections, that is an X client within the firewall displaying on an outside server. The risks associated with outbound X connections are more difficult to counter because less of the environment is under the controlled conditions within the firewall. The direct risk is uncertainty about the integrity of the inputs received from the X server. The indirect risks are the actions which the client may take in response to this low integrity input.

There are several aspects to this risk. These aspects may be placed into an order by the difficulty of the attack:

1. Outside clients connected to the outside X server using the X protocol.

2. The outside X server may have been corrupted.

3. The connection between the server and the inside client may be compromised.

The specific attacks which may be mounted against an outbound X client are the same as the penetration attacks described for the X Gateway policy. In the worst case, the X client may be induced to take any action permitted by the host operating system.

Since it is not possible to provide guarantees on the integrity of input events when company clients display on outside servers, the following policy assumes that the outside server is sound and that protections are to counter rogue X clients connected to the outside server. For this reason, there should also be a convincing argument showing why a company client cannot release sensitive information, regardless of the input stream, before use is allowed in an outbound X connection. It should also be noted that anything displayed on the outside server is visible to all other clients connected to that server (with very little effort) and to every host connected to the same networks (with a little more effort). The input events sent to the company client are also publicly visible.

A sensible precaution, whatever other measures are used, is to take extra steps to isolate the outbound client from company resources and assets. The company client should be run on a host which supports isolation of users (such as UNIX or VMS), and the client should execute from a restricted account. This relies on operating system protections to prevent release of information in case the outbound client is compromised. Another approach is to demonstrate that the client software is limited to a safe set of operations by analysis and review of the code.

Within these constraints, two simple policies will help to protect against attack by outside clients when displaying on an outside server. One policy seizes the server; the other hides behind a filter. To prevent attacks during use of a company client:

1. Purge or set the RESOURCE_MANAGER database property.

2. Grab the server for exclusive use by the company client.

A shell X client can be written to avoid race conditions. The shell can grab the server, verify the resource database, and execute the intended client within a single X connection. This approach is simple and sound, but it prevents other clients from using the X server when the company client is running.

An alternative policy is more complex to implement and analyze, but more pleasant to use. Assuming the company trusts the integrity of the remote server (or the client cannot be attacked via false inputs), the remaining risks to the client are synthetic events and client messages. Both are easily identified and filtered because the server sets the send_event flag in the event structure.

Summary

Xgate is still in the prototype stage, but it is an apparent success. The policy has shown to be sufficiently restrictive for security, while compatible enough to run numerous useful X clients. The approach is proven to be effective and sound without relying on unduly complex software for the security critical policy enforcement. There is a performance penalty, large enough to be detectable by the user; but overall, the package is thoroughly usable and practical. This approach also shows promise for application to other client/server or distributed applications with well defined protocols.