A Comparison of Trusted X Security Policies, Architectures, and Interoperability

Jeffrey Picciotto
The MITRE Corporation
Bedford, MA 01730
jp@mitre.org

Jeremy Epstein
TRW Systems Division
Fairfax, VA 22033
epstein@trwacs.fp.trw.com

Abstract

During the past several years, interest in architectures for multi-level secure versions of the X Window System (X) has grown dramatically, primarily in the Compartmented Mode Workstation (CMW) community. The architectures and security policies implemented in existing CMWs are similar to each other, although they differ in certain key details. Alternatives to the current approaches are being investigated. Most notably, a TRW-led team of researchers has designed and prototyped a trusted X implementation aimed at the Trusted Computer Systems Evaluation Criteria (TCSEC) B3 level. The architecture and security policies adopted by this implementation differ radically from those of all the existing CMWs.

One of X's greatest achievements is that it provides seamless interoperability among all X implementations in a distributed heterogeneous environment. Unfortunately, the divergence of security policies enforced by trusted X implementations has virtually eliminated X interoperability among these systems.

This paper surveys the architectures and security policies implemented in existing CMWs and the TRW prototype, identifies areas of commonality and divergence, and notes areas where interoperability can be achieved.

1 Introduction

The X Window System (X)[14] is the industry standard windowing system for workstations. X is designed around a client-server model, where the clients are the application programs, and the server maintains control of the display, keyboard, and mouse. Interoperability is a key feature of X: the X protocol is a well-defined byte stream, allowing clients and servers from many different vendors to interoperate. X client implementations are marketed on workstations, X terminals¹, and personal computers.

There are an increasing number of trusted X implementations²:

- About a half dozen commercial Compartmented Mode Workstation (CMW) systems designed to meet the CMW requirements (CMWREQS) [13] and the Trusted Computer System Evaluation Criteria (TCSEC) B1 requirements [9].
- One commercial CMW designed to meet the CMW Evaluation Criteria (CMWEC) [6] and the TCSEC B1 requirements.
- One X terminal designed to meet the CMWREQS and two X terminals designed to meet the CMWEC.³
- TRW's prototype designed to meet the TCSEC B3 requirements.

Unfortunately, when X and security join, interoperability suffers. As a result, users of CMWs and other trusted X systems are unable to interoperate with the same freedom as in the commercial X environment. To resolve this problem and others, a vendor working group, the Trusted Systems Interoperability Group (TSIG), has been working for three years to define interoperability among trusted systems for inter-process communication, networked file systems, administration, and X. Participants in the TSIG X group include most of the X implementors (including

¹An X terminal is an embedded system which runs the X server, frequently using special purpose hardware. It is similar to a diskless workstation, but without the general purpose operating system.
²The number of implementations is constantly increasing: the numbers shown here are believed to be correct as of publication.
³These X terminals use the CMW X servers as their basis, and hence are functionally equivalent. All comments in this paper about CMW systems apply equally to CMW X terminals.
non-CMW companies), as well as other interested parties.

In an earlier paper[3], we surveyed the issues involved in building trusted X systems. This paper defines interoperability, describes the security policies and architectures of existing implementations, and explains the interoperability issues and solutions between implementations.

2 What is Interoperability?

X clients operating in heterogeneous trusted X environments face many constraints imposed by security restrictions. Issues of interoperability arise when an X client on one machine wishes to display information on a second machine such as an X terminal. Portability issues arise when an X client implemented on one machine is reimplemented on a second, different, machine. While portability and interoperability issues are similar, each raises a unique set of concerns. This paper is limited to an examination of interoperability concerns.

Existing trusted X implementations are targeted both at the B1 and B3 levels of the TCSEC,4 and at the CMWREQS [13] and CMWEC [6]. An overview of the requirements that must be met by such implementations is provided in [3] and will not be repeated here. Many of the requirements identified in [3] directly impact client interoperability because they permit sufficient latitude for widely varied approaches to their implementation. Some elements of security policies, such as auditing of security-relevant events, have no impact on X interoperability because there are no functional or interface differences visible to X clients resulting from enforcement of those policies.

This paper, then, is limited to an examination of the differences among existing B1, B3, and CMW security policies that affect interoperability, and exposition of the impact on interoperability of those policies.

3 Security Policies

The fundamental security policies enforced by existing trusted X systems that impact interoperability are described below. In each case, we characterize the particular policy and implementation differences that can be found among existing implementations. In a later section, we examine the impacts of differing policies on interoperability.

We note that in addition to the security policies adopted by trusted X, interoperable clients will need to accommodate differences in the security policies and implementations embodied in the transport layer through which they communicate with X. These important differences in the transport layer functionality are not addressed by this paper except where they directly impact clients' abilities to exercise X functionality.

Finally, it is not our intent to evaluate or pass judgement on the design and policy alternatives discussed in the subsections below. However, when the benefits or limitations of a particular choice are not immediately self-evident, they are noted.

3.1 Discretionary Access Control Policies

Three basic approaches are being used to provide discretionary access control (DAC). The most easily implemented approach is to enforce DAC on a per-connection basis. That is, each time an X client attempts to connect to the X server, it must successfully pass DAC checks. After a client has established the connection, no further DAC checks are imposed. This type of restriction can often be implemented without code modifications by placing DAC restrictions on the communication endpoint to the server (e.g., socket or stream in Unix, port in Trusted Mach). While such an approach is acceptable for B-level systems and single-user-at-a-time CMWs, it has not been shown to be acceptable for true multi-user CMWs.

In an enhanced version of the approach described above, DAC is provided on a per-X object basis, but is fixed as owner only DAC5. That is, any client can connect to the server and create X objects, but may only access and modify objects created by the same owner. In systems where the owner is defined to be the creating client, this essentially assures client isolation. In systems where the owner is defined to be the user on whose behalf the client is operating, the policy is somewhat less restrictive.

The final approach is to provide full DAC on X objects by associating either permission bits or access control lists (ACLs) with every X object.6

A few CMWs provide full DAC using using permission bits or ACLs. However, most CMWs implement a mixture of owner-only and full DAC to provide maximal flexibility where necessary, while minimizing the complexity of, and number of modifications to X. For example, some CMWs protect cursors and graphics contexts as owner-only, but provide full DAC on properties and colormaps. There is little consistency across existing CMWs as to which objects are subject to full DAC and which are protected as owner only. In keeping with the B3 requirement to minimize TCB complexity, the TRW X implementation imposes DAC

4We are unaware of the existence of any B2 trusted X implementations.

5Arguably this is not true DAC inasmuch as there is no "discretion" involved. Such a policy has historically been referred to as DAC, possibly because like DAC it provides user identity-based access control.

6For multi-user systems, this approach is the only approach known to satisfy the B-level and CMW criteria.
restrictions only on connection to the X server.

All CMWs that provide changeable DAC on any X objects provide a mechanism for clients to modify the DAC attributes of those objects. This mechanism is invariably provided via an X extension. Although the functionality offered by the extensions are similar (e.g., Get/SetDacAttributes), the precise syntax and semantics of each request, reply, and event, vary across systems. Because the TRW B3 system does not support DAC on X objects, it neither provides nor supports this type of extension.

3.2 Mandatory Access Control Policies

All B-level and CMW systems must support multiple sensitivity levels (SLs), therefore, all must enforce mandatory access control (MAC).

In CMWs that enforce client-only DAC, the MAC checks are implicit in the DAC checks and effectively enforce a read equal and write equal policy. In the remaining CMWs, each object has associated with it a sensitivity level, and client access to the object is mediated based on the client's SL and the object's SL. These CMWs offer read down, read equal, and write equal. The TRW prototype, by virtue of polyinstantiating each object at every requested sensitivity level, effectively provides read and write equal.

Note that even within the constraints of read and write equal and read down, the MAC policies differ across systems. The reason is that the X objects, as defined by the vendors, are different. For example, whereas some CMW vendors treat a complete colormap as a single object with a single MAC label, others treat each colormap entry as an object with its own MAC label, with a MAC label for the entire colormap. This difference has a significant impact on clients.

For systems that label colormaps as objects, it is probable that all but clients running at system low will receive an error on attempting to add colors to the default colormap. Systems which provide colormap entry labeling can share the default colormap (or any other colormap), but at the cost of introducing complexity. Problems of this type impact not just colormaps, but all X objects. By contrast, TRW's prototype provides a default colormap at each SL which may be read and written freely by all clients at that SL.

Another example is windows. In X, windows are arranged in a hierarchy, starting with the top level window which is a child of the root window (the screen background). In some systems, every window in a hierarchy (starting with a top-level window, downwards) must be at the same SL. Other CMW vendors are implementing systems in which that constraint need not hold (e.g., the child window's SL must dominate the parent window's SL). Therefore, there are two fundamental policy differences among systems: differences in the MAC constraints and differences in the object definitions to which those constraints apply.

In addition, all CMWs protect a class of objects as accessible only by privileged clients. These objects typically include globally shared resources such as mouse sensitivity, keyboard repeat rate, etc. The CMWs protect these as readable by all clients, writable only with privilege. The TRW B3 X system, in contrast, polyinstantiates these objects. Although this allows all clients in the system to modify these objects as needed, polyinstantiating objects which users expect to be global raises usability concerns. The effects of each change to a global resource is limited to the sensitivity level at which the change was made, as opposed to affecting the entire system, as expected. (So, for example, a user who remaps the keyboard from QWERTY to Dvorak would have to do so at every sensitivity level in order to achieve results consistent with unmodified X.)

Events, replies, and errors are also labeled. The SLs applied to this data, in both CMW implementations and TRW's prototype, is simply the SL at which the client established its connection to the X server. Most existing implementations prevent a client from changing the SL of its connection once the connection has been established. In those implementations where clients may change the connection SL, the client must possess numerous privileges to perform the label change operation and to continue using previously available resources.

It becomes evident, given the above, that the MAC policies of existing CMWs and the TRW approach are fundamentally different. Furthermore, while the policies CMWs implement are largely similar, due to different object definitions, the MAC constraints on access to particular X objects may vary dramatically.

Finally, we note that, as with DAC, all CMWs must provide security extensions to the X protocol to support clients that wish to set or retrieve the MAC attributes of X objects. As previously noted, although the functionality is largely the same across CMWs, the precise syntax and semantics vary. Because the TRW implementation supports neither read down nor privileged clients, it does not provide such extensions.

3.3 Information Labeling Policies

Information labeling is required by the CMW Requirements [13] and Evaluation Criteria [6]. However, unlike many of the other security policies discussed here, information labeling does not enforce any access control restrictions, nor does it limit the base X functionality in any way. Thus, from the perspective of interoperability, information labeling has relatively little impact.
However, all CMWs must provide a mechanism for clients to set and retrieve the information labels associated with X objects. Each CMW provides this functionality through very similar, though slightly different, X extensions. As with MAC, these differences are compounded by the different granularity of objects identified by each implementation.

The TRW implementation, being targeted at B3, does not support information labeling. Providing information labeling would require a fundamental change in TRW’s design and implementation philosophy.

### 3.4 Privilege Policies

Systems that enforce policies restricting subjects’ access to objects generally provide a mechanism for privileged subjects to bypass the restrictions. Although the TCSEC does not require that any privilege mechanism be provided, the original CMW requirements (CMWREQS) [11] did, and the updated CMW evaluation criteria (CMWEC) are even more explicit and require the implementation of a specified set of independently-grantable privileges.

As a result of these diverse requirements, several different approaches have been undertaken. The TRW prototype, being targeted exclusively at the TCSEC, provides no privilege mechanism. As such it fails to meet the CMW requirements and, more importantly, cannot support any trusted X applications.

One CMW, evaluated against CMWREQS, provides only one X-related privilege. This privilege is asserted when a client first connects to the X server and remains in effect for the client for the life of that connection. The privilege permits the client to bypass all security policies enforced by the trusted server. Finally, existing CMWs being evaluated against the CMWEC provide finer-grained privileges. These implementations provide anywhere from a few to many privileges, depending upon the lengths to which each vendor pursues the principle of least privilege.

While vendors do not agree on the number, syntax, or semantics of X privileges, the underlying mechanism used to communicate privileges from clients to the server are generally very similar. All CMWs currently under evaluation provide a trusted communication protocol (e.g., trusted sockets), that envelopes each atomically written sequence of bytes with a security header. The privileges a process has in effect at the time of the write operation are taken by the server and remains in effect for the client for the life of that connection. The privilege permits the client to bypass all security policies enforced by the trusted server. Finally, existing CMWs being evaluated against the CMWEC provide finer-grained privileges. These implementations provide anywhere from a few to many privileges, depending upon the lengths to which each vendor pursues the principle of least privilege.

While vendors do not agree on the number, syntax, or semantics of X privileges, the underlying mechanism used to communicate privileges from clients to the server are generally very similar. All CMWs currently under evaluation provide a trusted communication protocol (e.g., trusted sockets), that envelopes each atomically written sequence of bytes with a security header. The privileges a process has in effect at the time of the write operation are taken by the server and remains in effect for the client for the life of that connection. The privilege permits the client to bypass all security policies enforced by the trusted server. Finally, existing CMWs being evaluated against the CMWEC provide finer-grained privileges. These implementations provide anywhere from a few to many privileges, depending upon the lengths to which each vendor pursues the principle of least privilege.

While vendors do not agree on the number, syntax, or semantics of X privileges, the underlying mechanism used to communicate privileges from clients to the server are generally very similar. All CMWs currently under evaluation provide a trusted communication protocol (e.g., trusted sockets), that envelopes each atomically written sequence of bytes with a security header. The privileges a process has in effect at the time of the write operation are taken by the server and remains in effect for the client for the life of that connection. The privilege permits the client to bypass all security policies enforced by the trusted server. Finally, existing CMWs being evaluated against the CMWEC provide finer-grained privileges. These implementations provide anywhere from a few to many privileges, depending upon the lengths to which each vendor pursues the principle of least privilege.

### 3.5 Object Reuse

Satisfying object reuse requirements does not typically impact applications since commercial software rarely capitalizes on loopholes of this nature. X suffers from the usual object reuse failures (e.g., undefined fields in protocol events and replies), but all trusted X implementations address these vulnerabilities without loss of functionality by specifying that undefined fields will be zero-filled.

Unfortunately, X also offers a limited set of functions explicitly designed to permit object reuse. The most glaring example is the creation of windows without backgrounds. When such windows are created and displayed on the screen, the visible contents of the screen overlayed by the new window becomes the new window’s contents. A client that creates such a window can then read the contents of the new window and thus read whatever was visible on the screen. All CMW implementations prevent this by disallowing untrusted clients from creating such windows.

The TRW B3 system offers a different solution: because the logical screen is polyinstantiated (i.e., a different image is available at each sensitivity level), and because the system offers no DAC protection within the X server, the ability to create these windows can be viewed as a complex read operation permissible under the other policies. Thus, whereas clients of existing CMWs are prohibited from using the functionality, clients of TRW’s B3 system may safely do so. Note, however, that polyinstantiating the screen can sometimes have negative impact on users. For example, even apparently innocuous operations, such as changing the root window color, are negatively impacted by this policy.

Unlike the policies previously described, no implementation provides extensions related to object reuse. Comparatively, then, differences in object reuse policies across trusted X implementations are not terribly noteworthy.

### 3.6 Trusted Path

All trusted X implementations provide a trusted path mechanism. The CMWs do so by providing trusted X servers which ensure that all text and graphics to be displayed in a window are properly rendered therein. This assurance is invisible to clients. In addition, the CMWs all offer a reserved portion of the screen. In some cases, this portion of the screen can be used by appropriately privileged clients to display data. In other cases, trusted applications can, by asserting a privilege, request that the TCB use the reserved portion of the screen to provide immediate unambiguous visual feedback indicating to the user that he or she is interacting with the TCB via trusted path.

TRW’s approach differs substantially from that of existing CMWs. Unlike the CMWs, TRW’s imple-
mentation does not assure the correct operation of the X server. Thus, though a client may request that a particular text string be displayed, the implementation does not assure that that string will be displayed nor that, if any text is displayed, it be the requested string. The TRW system only ensures that the labels associated with windows are the true and correct labels for those windows. In addition, TRW's implementation provides a reserved portion of the screen for trusted path feedback to the user. However, the system provides no facilities by which trusted clients can interact with users via trusted path. Rather, trusted path applications must rely on a very simple special purpose protocol (in place of the X protocol) for user interactions. This vastly minimizes the amount of code which must be trusted, at the cost of a non-standard protocol containing five operations (compared to the X protocol containing 120 operations).

Clearly, the existing CMWs' approaches differ dramatically from TRW's B3 approach. Even among CMWs, differences abound in functionality (e.g., whether or not reserved screen areas are addressable by trusted clients), privilege definitions, usage, and extensions.

3.7 Denial of Service

Denial of service is incompletely addressed by all existing trusted X systems, nevertheless, most implementations attempt to minimize their vulnerabilities. The greatest denial of service threat in X is the use of "grabs." In X, a client may grab the server, the keyboard, or the mouse. In each case, the client is afforded exclusive use of the grabbed resource. Thus, for example, a client that grabs the keyboard is granted full and exclusive use of the server until it chooses to relinquish the grab. Similarly, a client that grabs the keyboard will receive all subsequent keyboard input until it releases its grab. Clients typically grab the keyboard to ensure receiving input wherever the mouse pointer may be, or grab the server to ensure adequate performance for certain operations (such as providing rubber-banding around windows when they are moved or resized). Further information on grabs can be found in [14].

At least one CMW has chosen not to constrain the use of grabs beyond the limitations imposed by MAC policy enforcement. Such an approach maximizes flexibility at the cost of increased vulnerability. Several implementations offer a compromise solution: grabbing the server requires a privilege, grabbing the mouse or keyboard is unrestricted. Because keyboard and mouse grabs are commonly used by standard libraries and applications (e.g., the Motif toolkit), implementations do not limit their use. Note that all CMWs provide a mechanism whereby the user may break existing grabs by using a secure attention key. This feature is provided so that users may access trusted path even when malicious clients are trying to prevent that by using grabs. Existing COTS applications are not written to deal with grabs being broken, because the notion of a spontaneously broken grab does not exist in the X specification. Thus, breaking a grab may cause applications to exhibit unexpected or incorrect behavior.

The TRW B3 X implementation, by virtue of polyinstantiating the server, is not susceptible to denial of service attacks based on grabs. At worst, a client operating at sensitivity level SL1 may perform a grab on the server at SL1 and thus deny service to all other clients running at SL1. However, that client cannot perform grabs on servers operating at different levels and is therefore unable to deny service to clients operating at different levels. The TRW implementation does not provide a mechanism to break grabs, other than killing the grabbing client. However, invoking the trusted path does not require breaking grabs, since the trusted path mechanism is outside the X protocol. That is, a client can continue to hold a grab on the X server while trusted path is serviced by a separate server which does not use the X protocol.

Denial of service attacks other than grabs are not addressed by existing systems. While there are no extensions associated with the control of grabs, differences among CMWs in terms of what grabs are permitted, under which circumstances, and with which privileges, are substantial.

3.8 Authentication Policies

Most current trusted X implementations are designed for standalone workstations. As such, all identification information (such as user ID and process ID) is authenticated at user login and is passed to the trusted X implementation via a trusted communication protocol exactly as described for privileges, above.

Trusted applications that dynamically alter their identification attributes must consider the different impacts this could have on different systems. In all existing X implementations, a client's authenticated identification attributes are established only at connection time and remain unchanged thereafter; subsequent changes in the identification attributes associated with a client are ignored by the window system.

The transition of trusted X implementations to networked environments will require the introduction of trusted network protocols. Most of these, such as DNSIX, MaxSix, and Kerberos (see [2, 8, 12]) provide the necessary identification and authentication information to the window system invisibly to the clients. As above, however, trusted clients that wish to modify these attributes must consider the different constraints various implementations may impose on such operations. At this time, there is insufficient experience in this area to determine which constraints implementations may enforce.
3.9 Cut and Paste Policies

The Inter-Client Communications Conventions Manual [7] specifies a complex protocol that inter-operable X clients are expected to follow when exchanging data for the purpose of cut and paste. The CMWREQS and CMWEC specify functionality that CMWs must provide as part of trusted cut and paste. The CMW functionality both enhances and constrains the normal cut and paste functionality (e.g., it specifies under what conditions a particular user may downgrade data and specifies that, when a user does downgrade data, the user must be able to view the data in its entirety prior to downgrading). Existing CMWs implement the CMWEC requirements differently (for details on the various implementations see [10]), but none have modified the ICCCM-specified protocol. From the client perspective, then, these modifications are invisible. Some implementations, however, do provide a privilege for use by trusted applications that wish to bypass the constraints imposed by the CMWEC-required security policies.

The TRW implementation does not provide the functionality called for by the CMWREQS or CMWEC, but it does allow one cut and paste operation that would otherwise fail due to MAC constraints: the ability to upgrade data. As with the CMW implementations, this feature is provided without modification to the ICCCM protocol, hence invisibly to clients. Unlike the CMWs, the TRW implementation does not provide privileges for use by trusted applications wishing to bypass security policy-related cut and paste constraints.

An older mechanism for supporting cut and paste in X uses cut buffers. Cut buffers are a data storage mechanism implemented as properties on the root window. Although they are obsolete, many existing applications still use them. Existing CMWs typically treat properties as labeled objects subject to the standard Bell-LaPadula restrictions. In some CMWs, cut and paste operations based on cut buffers are limited to the levels at which the cut buffers were created. In other CMWs, cut buffers are entirely disallowed. The transition from the original CMWREQs to the CMWEC eliminated the requirement that delete up be prohibited. In response, at least one CMW vendor is developing a protection scheme that attempts to allow full cut and paste via cut buffers. It is too early to tell whether or not this attempt will succeed. By contrast, TRW's implementation allows cut buffers, but they are only accessible to pasting clients at the sensitivity label of the cutting client.

No implementation provides any X extensions to modify or enhance the cut and paste functionality provided by the underlying system.

4 Architectures

This section describes the architectures for existing CMW implementations and TRW's prototype.

4.1 CMW Architectures

Today there is one CMW that has successfully completed joint DIA/NCSC evaluation as a B1/CMW system, and four that are currently undergoing the evaluation process. All five of these systems provide trusted X implementations with similar architectures. More detail about the Sun CMW architecture can be found in [5]; descriptions of other CMWs have not been published except in vendor documentation.

In each CMW implementation, the server and window manager are both part of the TCB. The server is trusted to enforce the "internal" security policies (i.e., those that do not directly involve the user). These policies include MAC, DAC, auditing, information labeling, etc. The server also assures the correct operation of the X functionality provided via the X protocol. The window manager enforces the "external" security policies (i.e., those that are visible to the user). These policies may include the visible labeling of windows, cut and paste mediation, trusted path, and so forth. This architecture is depicted in Figure 1.

Because the X servers are trusted, this architecture permits vendors to support extensions to the X protocol which can be used both by untrusted security-cognizant applications and trusted applications possessing privileges. Examples of such extensions were noted in the preceding section. In addition, because the policies enforced by the servers are typically applied in a fine-grained fashion on X objects rather than a more global scale (e.g., on a per-connection basis), the servers can support fine-grained privileges that allow trusted clients to adhere to the principle of least privilege when using privileges to bypass server security policies.

Inclusion of the window manager in the TCB permits these implementations to provide a wealth of user-oriented security features. Most importantly it permits the use of a single window manager to provide a consistent look and feel in a multi-level window environment. All CMWs, however, have capitalized on the window managers inclusion in the TCB to provide menu-driven security-related functionality to the user. Thus, for example, users may start new clients at user-specified sensitivity levels.

While the architecture described above offers many benefits, it also has its drawbacks. The single most significant drawback is that the server and window manager together represent a large and very complex body of code. While the NCSC has not expressed

Subject to user clearance and accreditation range constraints.
an opinion on the issue, we believe it highly unlikely that a system which includes both components in the TCB can successfully meet the TCSEC B3 minimality architecture requirements.

4.2 TRW Prototype Architecture

Over the past two years, a team consisting of TRW, Trusted Information Systems, and the University of North Carolina designed and developed a prototype for a B3 version of X, known as TX. The architecture of this system (more fully described in [4]) can be seen in Figure 2. The basic notion is replication of large untrusted processes. Summarized, the X server is moved entirely outside the TCB and replaced by simpler components which implement a strict MAC policy by virtualizing input (the Input Manager) and output (the Display Manager). Visible labeling of windows is also performed by the Display Manager. The system runs one X server per unique sensitivity label. Because the X servers are untrusted, there is no need for DAC on X resources; access is limited to a single user at a time on a given display.10

An untrusted window manager is also run for each X server. Cut and paste among the X servers is allowed by means of an intermediary (the Property Escalator) assisted by untrusted clients which translate the complex X selection protocol into requests to read or write cut data. Simple operations such as invoking a new security level or changing the user's password are provided via trusted path. Because of the replicated architecture, there are no privileges associated with any X protocol operations (i.e., all applications are untrusted with respect to the window system).

Following are some of the advantages of TX, as compared to existing CMW implementations:

- High assurance (targeted at B3)
- Conceptually simple security policy
- Small TCB - 5% of a typical CMW
- Can run any window manager (not tied to a particular look and feel)
- Can run many window systems (not limited to X)
- Updates to X server require no TCB change
- Labels on screen are trusted
- Free of architectural covert channels
- X protocol unchanged within a sensitivity label

In addition, TX also has significant disadvantages compared to existing CMW implementations:

- Lack of DAC restricts use to single-user-at-a-time
- No information labels
- No trusted graphics
- No sophisticated user interface for trusted path facilities
- No data downgrade facilities
- Greater performance overhead
- More difficult to do multi-level secure applications than on CMWs

Some of the disadvantages can be overcome using a balanced assurance approach. For example, the X servers and window manager could provide C2 or B1 DAC features and/or trusted graphics, hence requiring only moderate trust. Similarly, the X server and window manager could provide information labels with B1 trust, providing that the underlying operating system supports them.

10By contrast to ordinary X or a CMW, where user 'sally' could bring up a window on the display where 'joe' is logged in.
5 Interoperability Issues

Interoperability in X means the ability to run an X client (application) on a brand A host and have its output displayed using a brand B X server (graphics terminal or workstation). In the multi-level secure world, we divide X interoperability into four classes:

1. Untrusted well-behaved clients
2. Untrusted, but security cognizant clients
3. Trusted application clients
4. Trusted system clients

The first class includes existing Commercial Off The Shelf (COTS) applications, while the second class includes applications which are untrusted but aware that they are running on a multi-level secure system. The third class includes trusted applications that are not part of the windowing system, while the fourth class includes trusted applications which are part of the windowing system, such as window managers. Each of these classes of applications has different interoperability requirements and expectations, as described in the following subsections.

Besides these classes of interoperability, we also discuss an alternate approach which holds promise for allowing separation of security policies from system implementation.

A related issue, not addressed here, is human interoperability, sometimes known as “drivability”. This refers to the ability of a person to move from one system to another, understanding the human interface and security policy of the system being used. Issues here include the trusted path interface, how windows are labeled, and downgrading usage. Another aspect of drivability in TRW’s system is the impact of changes being local, not global. For example, changing the mouse movement rates affects only the sensitivity label (SL) at which the request occurred. Thus, a user switching between SLs would find that the mouse moves at different rates, depending on the current SL. This class of problems is left for user interface designers.
5.1 Untrusted Well-Behaved Clients

Every trusted X vendor claims that well behaved untrusted applications will run unchanged on their system. There are two key questions: what does "well behaved" mean and what privileges must the application be granted, if any.

The trivial (but useless) definition of well-behaved is those applications which run unchanged. A more useful first step is those applications which use only those resources (such as windows) which they created. While this is a useful starting point, it both rules out a large class of applications (e.g., screen image dumps) and is insufficient, as additional restrictions are required.

While each implementation allows virtually the entire X protocol, different implementations impose different privilege requirements on protocol operations. One solution to running COTS applications is to assign them privileges to override those X security policies which prevent their operation. However, this is counter-productive: operations which are sensitive enough to require a privilege should not be freely handed out to COTS applications whose functioning is unknown.

As previously described, X includes the notion of a "grab", which provides exclusive access to a device or the entire server. All trusted X systems except the TRW prototype restrict grabs in some way. Given that grabs are a fact of life among normal X applications, the inconsistent restrictions placed by the various implementations severely hampers interoperability.

The Trusted Systems Interoperability Group (TSIG) X subgroup has made several attempts to define the common set of interoperable protocol operations. One such effort enumerated those protocol operations for which no existing implementation imposed restrictions, with the assumption that all resources being affected were created by the client making the request. Even with that limitation, the common set was too small to be useful.

Among the operations which were restricted by one or more of the existing implementations were requests to:

- create a window (only in certain limited cases)
- change certain window characteristics
- set grabs
- warp the pointer
- manipulate the default colormap
- get atom names (global symbols)
- create or change certain types of graphics contexts

In addition, the group did not consider other difficult issues such as limitations on interoperable cut-and-paste and how asynchronous events are handled in a secure heterogeneous system.

While there continues to be significant hope in TSIG that untrusted well-behaved applications can be made interoperable, we have grave doubts that such interoperability can occur any time soon. The gaps among the various security policies are too broad and will remain broad until there is significant experience with what works well in the real world.

5.2 Untrusted Security Cognizant Clients

Besides the restrictions placed on the protocol as described above, untrusted security cognizant applications need the ability to get the security attributes of X resources (e.g., their DAC and MAC attributes). While there has been some progress in TSIG in this area, interoperability is hampered because the implementations disagree on what X resources need to have security attributes, what the attributes are, and how to represent the attributes.

Most X implementations agree on the basic resources which need security attributes, including both objects created by the applications, such as windows, and global resources, such as the font search path. However, as one examines the protocol in detail, disagreements arise when trying to determine whether some data items are even resources (e.g., the current cursor position). To be interoperable for a security cognizant client, trusted X systems must agree on what data items have security attributes, and how to represent the security attributes to clients.

Even for untrusted applications, some of the security attributes may be changeable, such as the DAC attributes. Thus, to have untrusted interoperable security cognizant applications, there must be some level of agreement on what the security policy is for changing those attributes. Should client A1 be allowed to change the DAC attributes for client A2 if they have the same user ID? While this is typically acceptable in an operating system, it may not be in a windowing system.

The authors are even more pessimistic that this class of applications can be made interoperable.

5.3 Trusted Application Clients

This class of clients includes applications which are part of the TCB and have privileges to override certain window system security policies, but are not part of the window system. For example, a paragraph editor might fit this category, as might a front end for a multi-level database which labels individual data items on the screen.

Besides the ability to query security attributes and change them, this class of applications needs the ability to assess and enforce the system security policy. A major issue, not currently addressed in the trusted X environment, is what policy should be enforced in a distributed heterogeneous environment. That is, if a
trusted application is running on host A, but displaying on host B, which security policy should the trusted application enforce? One answer is that it enforces A's policy with respect to A's objects (e.g., files), and B's policy with respect to B's objects (e.g., windows). Another possible answer is that it enforces the least restrictive policy which is at least as restrictive as both A's and B's policies. This is clearly a general problem in distributed systems, not just X, but one for which the trusted X community lacks a full understanding.

At this point, we perceive relatively little hope in the TSIG community that this class of applications can be made interoperable.

5.4 Trusted System Clients

The final class of applications is those applications that are part of the trusted windowing system. This typically includes a window manager and may also include special clients for performing cut-and-paste operations. Interoperability here means that a brand A trusted window manager could run with a brand B trusted server and yield useful results. This is conceptually appealing, since it might allow a third-party to develop a window manager (or desktop manager) which provides a better user interface. However, each implementation splits responsibility for carrying out the security policy differently. For example, the role of mediating cut-and-paste could be performed by the window manager or the server. If brands are mixed, the result could be that neither or both pieces would try to perform mediation.

This level of interoperability is not considered as an attainable goal, although everyone agrees that it is desirable.

5.5 An Alternate Approach

A different approach, first proposed by Mark Smith in [1], is to assume that implementations will be able to agree on what is interoperable. Rather, a Policy Defining Client (PDC) is postulated which acts as the reference monitor to the X server. Each time the X server has an operation to perform on behalf of a client, it sends a message to the PDC explaining what it has been asked to do, and asking for permission to proceed. The X server has no understanding of what the actual policy is, only that it must ask the PDC for permission. Thus, the X server must be trusted to ask the PDC before each operation, and to follow the PDC's instructions.

Smith also proposes a set of protocol extensions which would allow clients to ask the X server to make a query on their behalf to the PDC. These extensions would be available to untrusted security cognizant applications as well as trusted applications. By using such extensions, trusted applications could follow the security policy of whatever system they are using, which should increase their value.

Smith's approach is currently being considered by TSIG as the most viable solution to interoperability problems. Considering the difficulties TSIG members have encountered in defining a common security policy, this approach has several advantages. First, it allows trusted applications to run portable. Second, it allows for third-party security policy definitions, rather than embedding them directly in the X server (e.g., non Bell-LaPadula[1] models could be inserted without changing the X server). Third, it allows for progress in interoperability without getting stuck in continuing policy discussions, which have been a drain on the TSIG efforts.

However, there are also some serious difficulties with this approach. First and most importantly, it still doesn't address the most crucial class of interoperability: what can untrusted applications do on any X system. Second, it requires that clients be written to account for any possible policy, since the PDC could implement any policy. Third, there are potential performance problems given the number of interactions required between the X server and PDC, although these might be minimized by appropriate caching or by structuring the PDC as part of the X server rather than as a separate process. Fourth, it still requires defining how security data is represented (e.g., how an ACL is stored by the PDC), and facilities to manipulate the security data.

While Smith's approach is appealing, it is not yet clear whether it will yield useful results.

6 Conclusions

Security policies among the trusted X implementations differ widely. While most CMW systems have similar security policies and architectures, their implementations are not identical. The TRW prototype has a radically different security policy and architecture from the CMWs. The result is that there is currently relatively little interoperability among implementations.

Our experience participating in interoperability working groups (such as TSIG) indicates that the hurdles that must be overcome to achieve full interoperability are sufficiently substantial that little is likely to be achieved in the short term. However, due to the following factors, some optimism is warranted: (1) there is general agreement throughout the community that the issue is both critical and difficult, (2) vendors continue to support interoperability working groups, and (3) trusted X implementations are new and therefore have not yet created a large market of third party software dependent on any particular implementation. Taken together, these factors favor a slow transition towards a common base that supports interoperability.
Acknowledgements

The authors gratefully acknowledge the discussions of TSIG members including Glenn Faden (Sun), Ed Cande (DEC), Jeff Glass (DEC), and Mark Smith (AT&T). We also appreciate the help of Rita Pascale (TRW) who critiqued numerous drafts of this paper.

References


