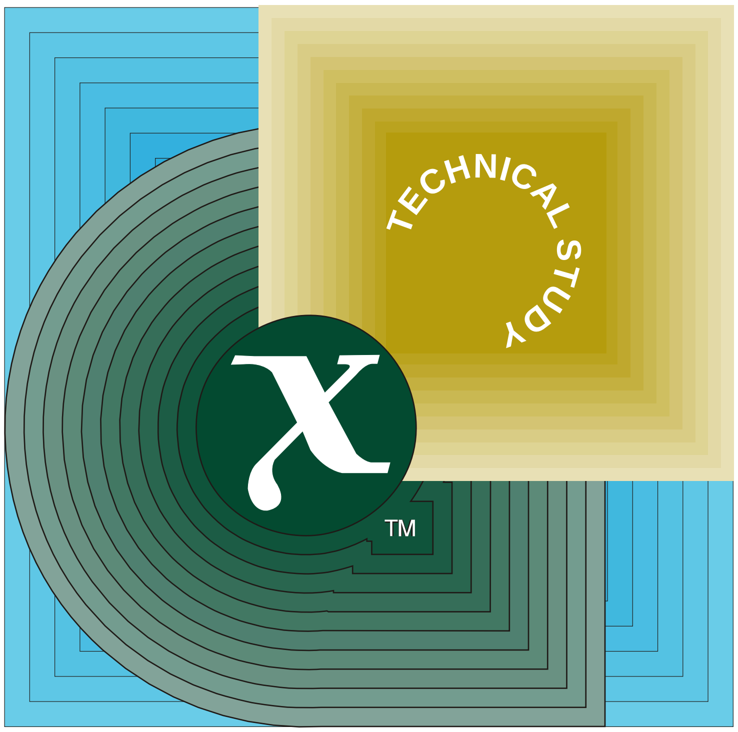
Technical Study

Desktop Security



THE Open GROUP

[This page intentionally left blank]

X/Open Technical Study

Desktop Security

X/Open Company Ltd.

X/Open Technical Study

Desktop Security

X/Open Document Number: E503

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.org

Contents

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is a Technical Study of security aspects of X/Open Common Desktop Environment (XCDE) specifications.

Structure

A brief description of the XCDE is given in Chapter 2.

Next, Chapter 3 gives an overview of the X/Open **XDSF** Guide. This framework has been developed by X/Open as a means for analysing and resolving distributed systems security issues. It is the framework within which XCDE security issues are discussed in this technical study.

All of the security threats that might apply, the countermeasures to them, and the extent to which countermeasures are described in the XCDE specifications, are discussed in Chapter 4.

Chapter 5 discusses the extent to which it would be appropriate to implement these countermeasures in the XCDE. This discussion forms the basis for some of the conclusions stated in Chapter 6. It has been separated from Chapter 4 because Chapter 4 is intended to be an objective statement of the threats that apply and of the countermeasures to them, whereas this chapter includes value judgements which, inevitably, are partly subjective.

Finally, Chapter 6 gives conclusions on the extent to which the XCDE specifications meet security requirements, and makes recommendations for action.

Trade Marks

Referenced Documents

The following X/Open documents are referenced in this study:

Motif Toolkit API
X/Open CAE Specification, April 1995, Motif Toolkit API (ISBN: 1-85912-024-5, C320).

Data Management: Security Issues
X/Open Technical Study, late 1995, Data Management: Security Issues (E507).

X11R5 File Formats
X/Open CAE Specification, May 1995, Window Management (X11R5): File Formats and Applications Conventions (ISBN: 1-85912-090-3, C510).

This comprises:

— Inter-Client Communications Conventions Manual (ICCCM)

— X Logical Font Description (XLFD)

— Compound Text

— Bitmap Distribution Format (BDF).

X11R5 Xlib
X/Open CAE Specification, May 1995, Window Management (X11R5): X Lib - C Language Binding (ISBN: 1-85912-088-1, C508).

X11R5 X Protocol
X/Open CAE Specification, May 1995, Window Management (X11R5): X Window System Protocol (ISBN: 1-85912-087-3, C507).

X11R5 X Toolkit
X/Open CAE Specification, May 1995, Window Management (X11R5): X Toolkit Intrinsics (ISBN: 1-85912-089-X, C509).

XCDE: Definitions and Infrastructure
X/Open CAE Specification, April 1995, X/Open Common Desktop Environment (XCDS): Definitions and Infrastructure (ISBN: 1-85912-070-9, C324).

XCDE Services and Applications
X/Open CAE Specification, April 1995, X/Open Common Desktop Environment (XCDS): Services and Applications (ISBN: 1-85912-074-1, C323).

XCS
X/Open CAE Specification, April 1995, Calendaring and Scheduling API (XCS) (ISBN: 1-85912-076-8, C321).

XDSF
X/Open Guide, December 1994, Distributed Security Framework (ISBN: 1-85912-071-7, G410).

XPG4, Version 2
The X/Open Branding Programme, How to Brand — What to Buy, February 1995 (ISBN: 1-85912-084-9, X951) (including X/Open CDE Branding Supplement, May 1995, X951S).

The following non-X/Open documents are referenced in this study:

ANSI X3.64-1979 C
: ANSI standard X3.64-1979: Additional Controls for Use with American Standard Code for Information Interchange.

ISO / IEC 6429
: ISO / IEC 6429: 1992, Information Technology — Control Functions for Coded Character Sets.

ISO / IEC 10736
: ISO / IEC 10736: 1995, Information Technology — Telecommunications and Information Exchange between Systems — Transport Layer Security Protocol.

RFC 822
: Internet RFC 822 — Standard for the Format of ARPA Internet Text Messages, Internet Architecture Board, 1982.

RFCs 1521 and 1522
: Internet RFCs 1521 and 1522 — MIME (Multipurpose Internet Mail Extensions), Parts 1 and 2, Internet Architecture Board, 1993.

Chapter 1

Introduction

1.1 Overview

The X/Open Common Desktop Environment (XCDE) specifications describe the use of ‘‘desktop’’ user interfaces within the X/Open Common Applications Environment (CAE).

The ‘‘desktop’’ style of user interface enables users of Information Technology systems to work much more effectively than traditional character-based user interfaces do. They are increasingly demanded by user enterprises. The XCDE specifications thus meet an important market need.

Quality of user interface is, however, not the only market requirement. There are many others, one of which is *security*. For many users, a good user interface is ‘‘nice to have’’, but security is essential, because a security breach could ruin them. Desktop-style user interfaces must be provided in a secure way.

The XCDE incorporates the X Window System windowing user interface infrastructure. Historically, this has usually been used over Local Area Networks (LANs). A LAN is often confined to a single building, or campus, and is relatively secure. Improvements in Wide Area Network (WAN) technology (such as ISDN and ATM) mean that the X Window System will increasingly be used over WANs. This will bring increased threats to security.

This technical study addresses security aspects of the XCDE specifications. It considers the security threats that apply, the possible countermeasures to them, the extent to which countermeasures are specified, and the extent to which they should be specified. It then states conclusions and presents recommendations for further action.

1.2 Interface Specifications and Components

X/Open publishes *interface specifications*, but brands products on the basis of their conformance to *component definitions* and *profile definitions*.

A *component* of the X/Open CAE is a functional unit whose interfaces are defined in specifications listed in the X/Open Portability Guide (the XPG). These specifications may be published by X/Open or by other bodies (for example, they may be international standards published by ISO).

A component as defined in the XPG can have one or more interfaces of the following kinds:

- human/computer interfaces used for interaction with human users
- portability interfaces used for interaction with other components and applications in the same computer (these interfaces are also referred to as ''programming interfaces'' or, when they are used by application programs, as ''application program interfaces'' or APIs).
- communications interfaces used for interaction across networks with other components and applications in a distributed system, and with other systems.

A profile is a collection of components that make up a logical set. Products that conform to a profile can be branded as such by X/Open.

In defining countermeasures to security threats, all of the interfaces of a component must be considered. For example, there may be little point in adding a ''confidentiality requested'' argument to a function in a programming interface specification for messaging unless the corresponding communications interface specification allows encryption.

Also, it is desirable to consider all the components in a system together, as well as considering each one singly. This avoids ''chinks in the armour'' where threats that affect several components are not completely countered by their combined security measures. It also avoids developing different mechanisms that do the same job, but for different components.

Security issues should therefore be addressed for components, rather than for interfaces, and should be addressed within the context of an overall system architecture or framework. The X/Open **XDSF** Guide provides the general context in which security aspects of every individual component should be addressed.

1.3 Scope of the Study

This technical study is concerned with components and profiles that have interfaces that conform to XCDE specifications.

It is not restricted to those components and profiles that are currently defined in the XPG. It considers all possible components and profiles that have interfaces defined by XCDE specifications. Nevertheless, the XCDE profile and the XCDE components currently defined by X/Open[1] are clearly of particular interest. The study therefore addresses that profile and those components specifically.

A system that includes the XCDE will also include other components. In particular, it will also include an operating system. The XCDE is not necessarily the best place to implement all possible countermeasures. The fact that the XCDE specifications are silent about a particular countermeasure does not prevent that countermeasure from being implemented, either within the XCDE or in other components.

This technical study does not comment in any way upon the security of XCDE products. Although the countermeasures for some of the threats that it identifies are not specified, they are nevertheless often provided by implementations.

However, present and future products should provide countermeasures in a uniform way where they might affect applications portability and interoperability. The XCDE specifications should describe security countermeasures to an extent that is at least sufficient to ensure this.

For the purposes of this technical study, the term *security* covers provision against intentional acts by people, but not against accidents or natural disasters. This is in line with the definition of security in the X/Open **XDSF** Guide. This is not to belittle the importance of providing against accidents and natural disasters, it is simply a question of defining the scope of the study clearly.

1. See Section 2.11 on page 17.

Chapter 2

The X/Open Common Desktop Environment

2.1 Introduction

The X/Open Common Desktop Environment (XCDE) specifications extend the X/Open Common Applications Environment (CAE) to enable applications programs to interact with people in a manner that is modelled on the way that desktops are used.

On a desktop, information relating to several activities can be kept readily accessible and can be used in carrying out the activity on which work is being done at any particular moment. Any information-processing activity can be performed. There may be special provision, through diaries, memo pads, and so on, for performing common office tasks.

In an information-processing system, the desktop is modelled by a user interface that has ‘‘windows’’. A user can have one or more windows for each of his activities. Each window contains a visible representation of information which is displayed to the user and which he may be able to create or modify. Changes in the information held by the system cause changes to the displayed representations, and changes made by users to the displayed representations cause changes to the information held by the system. A user can interact with any information-processing application program in this way. There is special provision for applications such as diaries and message handling systems that support common office tasks.

This form of user interface is very powerful and flexible. From the point of view of the applications programmer, it requires a more sophisticated form of programming interface than a traditional character-based terminal interface does. In particular, the application program does not directly control all aspects of the user interface. Rather, there are autonomous programs (‘‘managers’’) that handle many aspects of the interface (such as refreshing the display when the user selects a new activity) and whose operation can be influenced by the application program and by the user.

2.2 Elements of the XCDE

A user of the XCDE interacts with his system via the X Window System display and window handling system and the Motif style of user interface. He can customise some aspects of the user interface. The applications that he can use include an icon editor, a file manager, a mathematical calculator, a simple text editor, a calendar and appointments diary and an electronic mail system. A character-based terminal emulator enables him to use non-desktop applications. Printer management services enable him to print files and control printer operation. ‘‘Help’’ on how to use the system and its applications is available to him.

These services to the user are supported by the following elements of the XCDE:

- the X Window System display and window handling system, including the X Protocol handler, the X Library and the X Toolkit
- the Motif toolkit
- a manager that handles ‘‘virtual screens’’ called workspaces
- a manager that handles sessions of interaction between users and the system
- a ‘‘front panel’’ that gives access to the basic facilities of the system
- a style manager that allows users to customise the system’s visual behaviour
- an icon editor application
- a file manager application
- a mathematical calculator application
- a simple text editor application
- a calendar and appointments diary application
- an electronic mail application
- a character-based terminal emulator
- a printer manager
- a ‘‘help’’ subsystem
- an inter-process messaging system (based on ‘‘ToolTalk’’) that enables applications to communicate with each other
- a ‘‘Drag and Drop’’ service that supports the transfer, under control of the user, of information between applications
- data typing services that enable applications to categorise information
- execution management services that enable applications that have not been designed for the XCDE to be executed within the XCDE.

APIs to these components are available to the applications programmer. An application program can interact with the XCDE through a C language interface or through a ‘‘scripting’’ interface that uses an extension of the shell command language. Applications written by different people must cooperate within the XCDE. The XCDE specifications define a number of conventions and common information formats that enable applications to cooperate effectively. The XCDE also contains services that enable applications programs to be created and integrated into the system, and services that enable applications and information files in the system to be associated with icons displayed at the user interface.

2.3 X Protocol

The X Protocol is the protocol that enables a computer on which an application is running (a *client*) to communicate with a computer that contains user interface hardware (a *server*) so that the application can interface to a user. It allows for the transfer of keyboard and other inputs from the server to the client, and of transfer from the client to the server of commands to display graphical shapes and text. It is described in the X/Open **X Window System Protocol** Specification.

2.4 X Library

Xlib, described in the X/Open **Xlib — C Language** Specification, is a software library providing a C language interface to the X Protocol. It also provides:

- some inter-client communications facilities that enable clients to communicate with window managers and session managers
- facilities to manage a database of resources associated with the X Protocol (the *Xlib Resource Manager*)
- utility functions for use by applications.

2.5 X Toolkit

The X/Open **X Toolkit Intrinsics** Specification defines a widgets management C language API.

A *widget* is instantiated by a window that enables a user to perform input or receive output or both. Each widget belongs to some *widget class*. The class that a widget belongs to largely determines the way that it behaves.

A widget class has associated with it a number of *actions* that its widgets can perform. These actions can be invoked in response to user-input events. This is done by *translation management*, which is table-driven. The *translation tables* can be extended and modified by application programmers.

As well as being invoked in response to user input, actions can be invoked directly by applications.

An instance of a widget has associated with it a number of stored information items called *resources* that it can use. Resources belong to *resource classes*. There is a mechanism for extending widget resource class records at run time.

2.6 Motif Toolkit

The XCDE Motif toolkit enables the application programmer to give a particular ‘‘look and feel’’ to the user interface. It includes:

- the Motif Window Manager (*mwm*) command-line utility that handles the windows on the user’s desktop
- a number of widgets and associated functions
- functions associated with the User Interface Language (UIL)

  Note: UIL files can be compiled with the UIL compiler, which can be invoked via one of these functions, to create User Interface Definition (UID) files that can be loaded at run time by the Motif resource manager library.

- a number of other functions that perform operations related to the user interface (‘‘Toolkit Functions’’).

The Motif window manager, the above functions, and associated data type definitions are specified in the X/Open **Motif Toolkit API** Specification. The X/Open **XCDE Services and Applications** Specification defines the XCDE window manager, which consists of the Motif window manager plus some extensions.

2.7 File Formats and Application Conventions

The X/Open **File Formats and Application Conventions** Specification contains:

- the Inter-Client Communications Conventions Manual (ICCCM)
- the X Logical Font Description (XLFD) specification
- the Compound Text Encoding specification
- the Bitmap Distribution Format (BDF) specification.

They are discussed in the following subsections.

2.7.1 Inter-Client Communications Conventions Manual

The Inter-Client Communications Conventions Manual (ICCCM) in the X/Open **File Formats and Application Conventions** Specification states the conventions that applications should follow in order to interwork with other applications. There are conventions for:

- selection mechanisms
- cut buffers
- window managers
- session managers
- manipulation of shared resources
- device colour characterisation.

2.7.2 X Logical Font Description

The X Logical Font Description (XLFD) specification in the X/Open **File Formats and Application Conventions** Specification identifies those aspects of character renderings that can change from one font to another. It specifies how different fonts should be identified, and how their properties should be represented, in a desktop environment.

2.7.3 Compound Text Encoding

The Compound Text Encoding specification in the X/Open **File Formats and Application Conventions** Specification defines how text that may include characters from multiple character sets should be encoded in a desktop environment.

2.7.4 Bitmap Distribution Format

The Bitmap Distribution Format specification in the X/Open **File Formats and Application Conventions** Specification defines a standard format for representing character fonts.

2.8 XCDE Definitions and Infrastructure

The X/Open **XCDE Definitions and Infrastructure** Specification contains an introduction, a glossary and descriptions of:

- XCDE Data Format Naming
- the relationship of XCDE to the X Window System and Motif
- the C interface to some miscellaneous XCDE services
- the XCDE message services (ToolTalk)
- extensions to the Motif drag-and-drop services
- a C language interface to data typing services
- a C language interface to execution management services.

These descriptions are contained in a number of self-contained chapters of the X/Open **XCDE Definitions and Infrastructure** Specification. Each of these chapters is discussed in a separate subsection below.

2.8.1 XCDE Data Format Naming

The section on Data Format Naming forms the entire contents of the chapter entitled ‘‘General Definitions and Requirements’’. It contains a description of the names that are used to identify data formats. Standard names are defined for a number of formats, such as encapsulated postscript, RFC 822 message format, and the X PixMap format. A set of conventions is defined for the names of other data formats.

2.8.2 X Window System and Motif

The X/Open **XCDE Definitions and Infrastructure** Specification contains a description of the relationship between the XCDE and X Window System and Motif. It includes specifications of three additional widgets and of the C interfaces to some widget convenience functions for those widgets.

2.8.3 Miscellaneous Desktop Services

This chapter of the X/Open **XCDE Definitions and Infrastructure** Specification contains C language definitions of functions that initialise the desktop library and of a header file that includes a number of public constant definitions.

2.8.4 Message Services

The XCDE message services are based on the Sun Microsystems ToolTalk service. They support the creation, sending, reception and processing of messages between applications. Message reception is based on pattern matching criteria. The X/Open **XCDE Definitions and Infrastructure** Specification contains:

- a specification of the C interface to message services
- specifications of command line interfaces to message-related utilities
- specifications of standard messages.

2.8.5 Drag and Drop

The Motif drag-and-drop services are described in the X/Open **Motif Toolkit API** Specification. They allow for transfer of information between applications when the user ‘‘drags’’ an icon representing the information from one window to another, and ‘‘drops’’ it there. The X/Open **XCDE Definitions and Infrastructure** Specification defines extensions, including convenience APIs, to the Motif drag-and-drop services.

2.8.6 Data Typing

The XCDE data typing services provide capabilities that enable the attributes of a file or other data item to be determined from its name, in accordance with criteria held in a database. The X/Open **XCDE Definitions and Infrastructure** Specification defines:

- a C language API to XCDE data typing services
- the location and format of the files from which the database is created.

2.8.7 Execution Management

The XCDE execution management services support the handling of *XCDE actions*. These are applications and utilities that can be invoked from an application (via *DtActionInvoke*( )) or from the command line (via the *dtaction* utility).

Each system in the XCDE has access to an actions database (which may be distributed across several systems) that defines the relationship of actions to files associated with application programs and utilities. If commands to invoke them are placed in the action database (for example, by an application via the execution management API or by a user via XCDE action creation services), then XCDE actions can be invoked by the user from the desktop.

Note that XCDE actions are not the same as widget actions (although a widget action may be the means of executing an XCDE action).

The X/Open **XCDE Definitions and Infrastructure** Specification defines:

- a C language API through which applications can load the database, query the database, and invoke XCDE actions (via *DtActionInvoke*( ))
- the *dtaction* utility that can be invoked from the command line and that in turn invokes an XCDE action
- the format of files from which the database is loaded.

2.9 XCDE Services and Applications

The X/Open **XCDE Services and Applications** Specification defines:

- APIs to the standard services and applications that are available to the user in the XCDE
- tools that are available to the programmer writing applications for the XCDE and to the system integrator building a system that includes the XCDE
- standard conventions that should be used by application designers so that the user interface will have the appropriate ‘‘look and feel’’.

These definitions are contained in a number of self-contained chapters of the X/Open **XCDE Services and Applications** Specification. With some exceptions, each of these chapters is discussed in a separate subsection below. The exceptions are:

- the chapter on window Management Services, which is discussed together with the Window Management part of the X/Open **Motif Toolkit API** Specification in Section 2.6 on page 8
- the chapter on Calendar and Appointment Services, which is discussed together with the X/Open **Calendaring and Scheduling API** Specification in Section 2.10 on page 16.

2.9.1 Window Management Services

The XCDE window manager handles the windows on the user’s ‘‘desktop’’. It is a superset of the X/Open Motif *mwm* window manager. It is discussed together with the Window Management part of the X/Open **Motif Toolkit API** Specification in Section 2.6 on page 8.

2.9.2 Workspace Management Services

The XCDE workspace manager provides support for multiple ‘‘virtual screens’’ known as *workspaces*. The X/Open **XCDE Services and Applications** Specification defines a C language API to the workspace manager. The API enables an application to query and control the state of a workspace, to add windows to workspaces, and to remove windows from workspaces.

2.9.3 Session Management Services

A session is the collection of applications, settings and resources that are present on a user’s desktop from the time that he logs in to the time that he logs out. The XCDE session management services are described in the ICCCM sections of the X/Open **File Formats and Application Conventions** Specification. The X/Open **XCDE Services and Applications** Specification specifies the session management capabilities that must be provided, and defines:

- a C-language API to them
- XCDE actions for session management.

2.9.4 Help Services

The XCDE help services provide information to the user about the system, the desktop environment and the applications that he can use. They are provided to the user through a number of widgets on the desktop. The X/Open **XCDE Services and Applications** Specification describes the help service capabilities that must be provided, and defines:

- the widget classes of the help service widgets
- an API to help services
- XCDE actions for help services

- a markup language (*HelpTag*) that can be used for writing help information.

2.9.5 Calendar and Appointment Services

The chapter on Calendar and Appointment Services is discussed together with the X/Open **Calendaring and Scheduling API** Specification in Section 2.10 on page 16.

2.9.6 Mail Services

The XCDE Mail Services allow the user to send, receive and manipulate electronic mail messages. The message format defined in RFC 822 is assumed, and the messages can be MIME-encoded and have attachments, as described in RFCs 1521 and 1522. The X/Open **XCDE Services and Applications** Specification describes the mail capabilities that must be provided, and defines:

- XCDE actions for mail services
- inter-process (ToolTalk) messages for mail services.

2.9.7 File Management Services

The XCDE file management services provide a user interface for manipulation of objects (including files) and folders, and for application execution. The X/Open **XCDE Services and Applications** Specification specifies the file management capabilities that must be provided, and defines:

- XCDE actions for file management
- inter-process (ToolTalk) messages for file management.

2.9.8 Front Panel Services

A front panel appears in every XCDE workspace. It is a window that provides the user with access to the most commonly used desktop facilities. The X/Open **XCDE Services and Applications** Specification specifies the front panel capabilities that must be provided by an implementation of the XCDE, and defines the format of the configuration files that give the values of the front panel parameters in installed systems.

2.9.9 Text Editing Services

The XCDE text editing services enable the user to create and edit short documents, using the *DtEditor* widget. The X/Open **XCDE Services and Applications** Specification specifies the text editing capabilities that must be provided, and defines:

- the *DtEditor* widget
- a C-language API to the XCDE text editing services
- the *dtpad* utility that can be invoked from a command line
- XCDE actions for text editing
- inter-application (ToolTalk) messages for text editing.

2.9.10 Icon Editing Services

The XCDE icon editing services enable users to create and modify icons. The X/Open **XCDE Services and Applications** Specification specifies the icon editing capabilities that must be provided, and defines:

- XCDE actions for icon editing
- inter-application (ToolTalk) messages for icon editing.

2.9.11 GUI Scripting Services

The XCDE GUI scripting services provide extensions to the shell programming language defined in X/Open **XPG4** CAE Specifications that allow access to the XCDE services. The X/Open **XCDE Services and Applications** Specification defines the *dtksh* utility that can be invoked from the command line and that provides these services.

2.9.12 Terminal Emulation Services

The XCDE terminal emulation services provide a window for applications written for character-based terminals. The services are available in two forms: a stand-alone client and a widget. The terminal emulation provided is partly based on the VT220 terminal, and is compatible with ANSI X3.64-1979 and ISO/IEC 6429. The X/Open **XCDE Services and Applications** Specification specifies the terminal emulation capabilities that must be provided, and defines:

- a C-language API to XCDE terminal emulation services
- the *DtTerm* terminal emulation widget
- the *dtterm* terminal emulation utility that can be invoked from the command line
- XCDE actions for terminal emulation
- the format of the data stream (including escape sequences) recognised by the terminal emulation.

2.9.13 Style Management Services

The XCDE style management services enable users to customise the visual elements and system behaviour of the XCDE. The X/Open **XCDE Services and Applications** Specification defines the capabilities that an implementation of XCDE style management services must provide, and the style management XCDE actions.

2.9.14 Application Building Services

The XCDE application building services provide the application developer with aid in assembling graphical objects into the user interface and with generation of appropriate calls to XCDE API routines. The X/Open **XCDE Services and Applications** Specification specifies the capabilities that an implementation of the XCDE application building services must support, and defines:

- the *dtcodegen* utility that can be invoked from the command line and that provides XCDE application building services
- XCDE actions for application building.

2.9.15 Application Integration Services

The XCDE application integration services enable application developers and system administrators to integrate applications into the XCDE. The X/Open **XCDE Services and Applications** Specification defines:

- the *dtappintegrate* utility that can be invoked from the command line and that provides XCDE application integration services
- XCDE actions for application integration.

2.9.16 Action Creation Services

The XCDE action creation services enable users to create and modify action and data type definition files. Action definition files provide the ability to associate XCDE actions with icons on the desktop. Datatype definition files provide the ability to specify sets of files that contain data of particular types, and to associate icons with them.

The XCDE action creation services are provided through a set of predefined XCDE actions that are specified in the X/Open **XCDE Services and Applications** Specification. The X/Open **XCDE Services and Applications** Specification also defines the capabilities that an implementation of XCDE action creation services must support.

2.9.17 Print Job Services

The XCDE print job services provide information to users about printers and print jobs. The X/Open **XCDE Services and Applications** Specification defines the capabilities that an implementation of the XCDE print job services must support, and defines the print job service XCDE actions that can be invoked.

2.9.18 Calculator Services

The XCDE calculator services provide basic computation capabilities to users. The X/Open **XCDE Services and Applications** Specification defines the capabilities that an implementation of the XCDE calculator services must support, and defines the calculator service XCDE actions that can be invoked.

2.9.19 Application Conventions

The ‘‘Application Conventions’’ chapter of the XCDE describes font and icon conventions that applications should follow.

2.9.20 Application Style Checklist

The ‘‘Application Style Checklist’’ chapter of the XCDE states the style requirements for XCDE applications.

2.10 Calendar and Appointment Services

The XCDE calendar and appointment services enable users to maintain appointment diaries, and to browse and update their own and other users' diaries in order to set up group meetings.

A C language API to the XCDE calendar and appointment services is defined in the X/Open **XCDE Services and Applications** Specification and the X/Open **Calendaring and Scheduling API** Specification. The X/Open **XCDE Services and Applications** Specification also specifies the capabilities that an implementation of XCDE calendar and appointment services must support, and defines:

- command-line interfaces to calendar and appointment services
- XCDE actions for calendar and appointment services
- actions that can be invoked from the desktop or by applications and that provide calendar and appointment services
- inter-application (ToolTalk) messages used by calendar and appointment services
- the data formats of calendar archive files and calendar entries.

2.11 Components and Profiles

Two X Window System components are currently defined in the X/Open **XPG4** CAE Specifications. A further three components and a profile are currently being defined for XCDE.

The existing XPG4 components are as follows:

XPG4 X Window System Display
This provides a local display for remote applications. Conforming products must support the X Protocol (as servers) and the File Formats and Applications Conventions.

XPG4 X Window System Application Interface
This provides X Window services to applications to enable them to drive graphical displays. Conforming products must support the X Protocol (as clients), the Xlib API, the X Toolkit Intrinsics API and the File Formats and Applications Conventions.

The three XCDE components currently being defined are as follows:

XPG4 X Window System Application Interface V2
This is similar to the XPG4 X Window System Application Interface component, but its definition is aligned with Revision 5 of the X.11 specifications (so that conforming products must implement the referenced X/Open **X Window System Protocol** Specification).

XPG4 Motif Toolkit
This provides APIs equivalent to those of Motif Release 1.2 to a Graphical User Interface toolkit supported by an underlying X Window System at X11R5 level (conforming products must implement the referenced X/Open **Motif Toolkit API** Specification).

XPG4 Calendaring and Scheduling
This provides a high-level calendaring and scheduling API that can be supported by calendaring and scheduling services (conforming products must implement the referenced X/Open **Calendaring and Scheduling API** Specification).

The XCDE profile currently being defined is the **XPG4 Common Desktop Environment**. It provides a standard set of functional capabilities and supporting infrastructure, and the associated standard APIs, command line actions, data interchange formats and protocols for implementing a desktop workstation. It provides standard forms of the facilities normally found in a graphical user interface environment, including windowing and window management, session management, file management, electronic mail, text editing, calendar and appointments management, calculator, application building and integration services, print services and a help service. Conforming products must include the XPG4 X Window System Application Interface V2 component, the XPG4 Motif Toolkit component, and the XPG4 Calendaring and Scheduling component, and must also implement the X/Open **XCDE Services and Applications** Specification and the X/Open **XCDE Definitions and Infrastructure** Specification.

Chapter 3

The X/Open Distributed Security Framework

3.1 Overview of the X/Open Security Framework

This overview is necessarily considerably simplified. The reader is referred to the X/Open **XDSF** Guide for a full description.

3.1.1 Scope and Purpose of the Framework

The X/Open **XDSF** Guide applies to computer systems that can:

- interact with people, with other computer systems, or with electronic or mechanical devices (for example, in process-control applications)
- store information
- perform operations on information that is input to them or that they have in store.

It is particularly concerned with systems that contain components of the X/Open Common Applications Environment (CAE).

Computer systems are vulnerable to damage to, or unauthorised use of, information or resources. A possibility of such damage or unauthorised use is a security threat. The X/Open **XDSF** Guide is a framework within which system implementors, application writers and end users can cooperate to prevent security threats from being realised.

3.1.2 Security Domains and Policies

A *security domain* is a computer system, a part of a system, or a collection of systems and parts of systems, that is considered as a unit for security purposes. Security of a domain is achieved by analysing the threats that apply to it, considering the possible countermeasures, creating a *security policy* for the application of the appropriate countermeasures, and following that policy.

An operational system typically contains a number of security domains. They may be nested (for example, the whole system could form a domain and the disc subsystem could be a subdomain of it). They may overlap. (For example, a database management system and application could form one domain, and the disc subsystem could form another. These domains overlap because they both contain the part of the disc store that is used by the database management system. However, neither is a subdomain of the other.)

3.1.3 Principals

A *principal* is an entity that is identified within a system, that has access to the information held by the system, and on behalf of which the system can perform operations. Often (but not always), the principals of a system are the people who interact with it.

The restriction of particular information and operations to particular principals is an important aspect of most security policies.

3.1.4 Threats

The X/Open **XDSF** Guide identifies the threats that could apply to any security domain. They are:

- unauthorised modification
- unauthorised disclosure
- unauthorised use of resources
- denial of service
- repudiation.

3.1.5 Countermeasures

The countermeasures identified in the X/Open **XDSF** Guide are as follows:

Domain Segregation
This ensures that inputs to the domain can only be made, and outputs from the domain can only be received, through the identified interfaces of the domain.

Information Verification
This ensures that the sources of inputs and destinations of outputs are authentic, that the inputs are received from them correctly, and that the outputs are received by them correctly.

Service Mediation
This ensures that inputs and outputs can only be made when the principals on whose behalf they are made are authorised to request them.

Resilience and Recovery
This ensures that the services of the domain are available when needed.

Activity Recording
This ensures that a record is kept of activity relevant to security.

3.1.6 Security Services

The X/Open **XDSF** Guide identifies a number of *security services* that can be used in implementing countermeasures. They are classified as follows:

Separation Mechanisms
Responsible for the segregation of resources, and can be used to ensure that the information and resources of one domain are not visible or accessible from outside it. An example is the service provided by many operating systems of ensuring that the memory allocated to one process cannot be accessed by another.

Basic Security Facilities
Used within a domain to enforce security policy requirements for that domain. An example is the *authentication service* that verifies the claimed identity of a principal.

Domain Interaction Services
Support interaction between security domains. An example is the *secure association* service that provides a secure communications channel.

Separation mechanisms are typically provided by the hardware, operating system and communications services (the *platform*) rather than by data management components or applications. Basic Security Facilities and Domain Interaction Services could be provided by security components (possibly in conjunction with the platform), and could be provided via programmatic interfaces. Although these services are identified in the X/Open **XDSF** Guide, programmatic interfaces for them are, in most cases, not currently specified by X/Open.

3.1.7 Security Awareness

The security services could be used directly by an application. Such an application is *security-aware*. Alternatively, they could be used by the system components that are used by the application, without the application having to invoke them directly. Such an application is *security-unaware*.

3.2 Application of the Framework to the Desktop

The XCDE may be implemented on a single computer, but is typically implemented on one or more clients and one or more servers, connected by a network.

In an operational system, a desktop component typically intersects with or forms part of several overlapping security domains. The component must be able to support part of the countermeasures required by the security policies of the domains that include it.

A desktop component:

- could form a single security domain, or part of a single security domain
- could alternatively form several security domains, or parts of several security domains, each domain being associated with one or more clients or servers.

A domain that contains a desktop server is also likely to contain the hardware and operating system (the *platform*) on which the server runs. The security policy for the domain must take into account what security services are provided by the desktop component (or part of it) that is in the domain. This may partly depend on what security services are provided to the desktop component by the platform.

A domain that contains a desktop client may also contain the platform on which the client runs. Client domain security policies must also have regard to platform security services.

Chapter 4

Threats and Countermeasures

This chapter discusses the specific threats that can arise in connection with the XCDE, the countermeasures that are appropriate, and the extent to which countermeasures are specified for the XCDE.

Note that this is simply intended to be a list of threats and of appropriate countermeasures. There is no suggestion that any of these countermeasures should be implemented within the XCDE. In some cases (for example, physical countermeasures), there is no possibility of implementing them within the XCDE. Thus, a statement that no countermeasures for a certain threat are specified for the XCDE does not imply any shortcomings, either in the XCDE specifications or in conforming products. The question of which countermeasures should be implemented within the XCDE is addressed in Chapter 5.

4.1 User Interface Threats

These threats apply to:

- the front panel
- application user interfaces implemented using Motif (with XCDE extensions), the X Toolkit, XCDE help services, XCDE text editing services and XCDE terminal emulation services
- utilities that can be invoked from the command line (*mwm*, ToolTalk utilities, *dtaction*, *dtcodegen*, *dtappintegrate*, *dtksh*, *dtterm* and calendar and appointment utilities)
- XCDE actions, because they can also be invoked from the command line (and, indirectly, from the desktop).

The threats are given below.

4.1.1 Masquerade of User

One user can pretend to be another user (who perhaps has more privileges).

The normal countermeasure to this is authentication. The XCDE does not specify any user authentication (the X Protocol and Xlib support authentication of the X client, but not of the server). User authentication provided by the operating system (for example, login) provides some protection.

4.1.2 Session Hijack

A user can take over a session established by another user (for example, if the second user leaves his terminal unattended).

This could be countered by requiring re-authentication of the user whenever an action is requested. Such re-authentication would however be a serious inconvenience.

XCDE session management provides session lock and screen saver facilities for the front panel and for application user interfaces. The session lock facility enables a user to prevent his session from being taken over while he is away from his terminal, but has to be invoked deliberately. A screen saver facility causes the user's screen to be overwritten under certain conditions; it can be implemented such that further input is then disabled until the user re-authenticates himself.

These facilities do not provide complete protection in themselves. Users must adopt secure working practices (for example, in activating the session lock facility when leaving terminals unattended).

The session manager is required to invoke a screen saver on user request. Systems may provide a user option to invoke a screen saver on X server timeout. According to the session management specification, they are not required to do this but, according to the style management specification, they are required to allow the user to enable or disable the screen saver and to set the screen saver timeout interval.

There is no protection for utilities or actions invoked from the command line.

4.1.3 Arrogation of Authority

A user can claim privileges to which he is not entitled, or attempt to access information that he is not authorised to access. This can have the same results as masquerade.

This is normally countered by imposing *access control* and rejecting requests for unauthorised actions.

The XCDE does not specify any user access control. Access control implemented by the operating system (for example, read/write/exec access control on files) provides some protection.

4.1.4 Resource Overload

A user can overload a resource and thereby make services unavailable to other users (for example, on a low-bandwidth network, a user might, by continually moving his mouse, take up so much network bandwidth that other users had very poor response times).

This can be countered to some extent by resource allocation limits imposed by the operating system. For example, a time-slice scheduling algorithm makes it impossible for a single process to use all of the CPU time when other processes are running. This usually does not apply to all resources. For example, many operating systems do not limit the amount of filestore space that a user can have.

No countermeasures are specified for the XCDE.

4.1.5 Back-door Access to Displayed Data

It is sometimes possible to monitor, or even modify, information displayed on a user's screen, by methods such as:

- tapping into the video display hardware or into cables that connect it
- monitoring electro-magnetic radiation emitted by the display (for example, in a detector van parked outside the building where the user works)
- installing a software process that monitors the contents of video memory.

The normal countermeasures to this are physical security and electro-magnetic screening.

No countermeasures are specified for the XCDE.

4.2 API Threats

These threats apply to the following APIs:

- Xlib
- the X Toolkit (Xt)
- the Motif toolkit (with XCDE extensions)
- XCDI miscellaneous services [*DtInitialize*( )]
- data typing
- execution management
- messaging (ToolTalk)
- drag and drop
- workspace management
- session management
- help services
- text editing
- terminal emulation
- calendaring and scheduling.

They also apply to all XCDE actions, since these can be invoked through execution management.

The threats are given below.

4.2.1 Masquerade of Application Program

A program can masquerade at a programmatic interface by presenting a user identity that does not belong to its *principal* (normally, the user who invoked it). This can lead to the same results as masquerade at a human/computer interface.

Masquerade at a programmatic interface can be countered by relying on the authentication obtained by the operating system when the user logs on. For example, on an X/Open-compliant system, the implementation of the API can use *getuid*( ) to check his identity. However, this counter is not always effective. For example, an application with super-user privileges could use *setuid*( ) to give itself any user identity. Further authentication is therefore sometimes obtained via the programmatic interface itself.

The authentication features of the X Protocol and Xlib are countermeasures, but:

1. the authentication features of the X Protocol are incompletely specified and require bilateral agreement if they are to work
2. the authentication features of the X Protocol are not accessible through Xlib
3. the authentication feature of Xlib is not very good — it relies on host identity rather than application instance identity
4. these countermeasures only apply to functions that require X communication (so, for example, they provide no protection for execution services).

The calendaring and scheduling API provides for password authentication of the user.

Other than these, no countermeasures are specified for XCDE.

4.2.2 Session Hijack

One application can take over a session established by another. For example, if it could obtain the session handle returned by *csa_logon*( ) to another application, a rogue application could read or modify the calendar entries for the user who had invoked the other application.

No countermeasures to this are specified for the XCDE.

The operating system may provide some protection (for example, by ensuring that applications can not access each others' memory, and hence can not obtain session handles).

4.2.3 Arrogation of Authority

An application program may arrogate authority by attempting, under the identity of its own principal, to use services that he is not authorised to use, or attempting to access information that he is not authorised to access. This can have the same results as masquerade.

The normal countermeasure to this is access control.

In some situations, this can be countered by the access control imposed by the operating system. In others, further access control may be applied via the programmatic interface.

Access control is applied by calendar and appointment services. Apart from this, no access control is specified for XCDE APIs.

The operating system may impose access control on some resources accessed via the APIs.

4.2.4 Shared Memory Access

An application can sometimes access memory used by the API implementation or by other applications. For example, suppose that CSA session records are kept in user memory space and that session handles are pointers to session records. An application that has obtained one session handle can search the memory near where it points to, and so obtain other session handles.

This can result in unauthorised modification or disclosure. A modification of the implementation's data structures could have further results, such as denial of service to other users.

No countermeasures to this are specified for the XCDE. Some platforms and API implementations may provide protection against this threat.

4.2.5 Resource Overload

An application can overload a resource and thereby make services unavailable to other applications.

As with excessive use of a human/computer interface, this can be countered by a combination of resource allocation limits imposed by the operating system and further resource rationing imposed by the implementation of the interface.

No countermeasures to this are specified for the XCDE.

4.3 Communications Threats

These threats apply to:

- communications that use the X Protocol
- ToolTalk messages (even where these are sent from one process to another within a single system).

The threats apply to inter-client communications, including those invoked via Xlib. Such communications take place via servers and the X Protocol. In this study, inter-client communications are not considered separately from other X Protocol communications.

The threats are given below.

4.3.1 Masquerade of Communications Partner

One process can pretend to be another process, or can pretend to be in a system other than the one in which it really is.

In the context of the X Protocol:

- a masquerading server could provide access to applications and information
- a masquerading client could obtain users' authorisation credentials and these could later be used to gain access to the system (such a masquerader is sometimes called a *password grabber*).

This threat is normally countered by authentication.

The X Protocol and Xlib provide some authentication, but it is limited (see Section 4.2.1 on page 25).

A further consideration with X Protocol authentication is that it is not obvious how the client should identify and authenticate itself. Such authentication is necessary to counter the threat of a password grabber. A simple password scheme would not be adequate, because a rogue server could immediately learn the password. Schemes could be devised (for example, the client could send a password that has been encrypted using a ''one-time pad'' algorithm that produces a different result each time it is used), but they would necessarily be complex. The form of authentication allowed for in the X Protocol — a simple exchange of information — significantly restricts the range of schemes that could be used.

The ToolTalk message server (*ttsession*) can provide authentication on two levels:

unix	The sender and receiver must have the same user identity.
des	The underlying RPC calls use AUTH_DES.

In a distributed system, it is unsafe to rely on user identities, since different users can have the same identity on different computers.

It is not clear from the X/Open **XCDE Definitions and Infrastructure** Specification what the AUTH_DES scheme of the underlying RPC is, or what protection it gives to ToolTalk users.

4.3.2 Tampering with Communications Media

This includes passive monitoring. It can be used to disclose information in transit, to modify information in transit, or to obtain unauthorised use of resources. (Within a single system, ToolTalk messages could be monitored or modified on disc or in memory.)

The normal countermeasures are integrity checking and encryption of information in transit.

No countermeasures are specified for the XCDE.

Lower-level communications protocols such as the OSI secure network and transport protocols or (in the case of ToolTalk messages) secure RPC can provide some protection, but they are not very widely used.

It should be noted that encryption and integrity checking can degrade performance. They should therefore be used only where there is a real security need. This particularly applies to interactive applications such as those of the XCDE.

4.3.3 Security Breach in Remote System

A security breach in a communications partner can lead to disclosure or modification of information held by a system, or to unauthorised use of its resources. For example, if someone could monitor the screen of an X server, they could obtain information held by client applications and displayed on the screen.

No countermeasures against this are specified for the XCDE.

4.3.4 Traffic Analysis

Analysis of traffic on communications networks can sometimes provide useful information. For example, heavy traffic on a military network could be a prelude to an impending attack.

No countermeasures against this are specified for the XCDE.

4.3.5 Arrogation of Authority

A communications partner can claim privileges to which it is not entitled.

Xlib applies access control on the basis of host identity. The X Protocol provides for more general authentication, which could be used as a basis for access control.

The ToolTalk message server (*ttsession*) can apply access control on the basis of its authentication (see Section 4.3.1 on page 27).

4.3.6 Network Damage or Congestion

This can deny service to users (for example, if a network is congested, X communications could be so slow as to make some X servers effectively unusable).

No countermeasures against this are specified for the XCDE.

4.3.7 Repudiation

The originator of a message can deny having sent it.

This is normally countered by digital signatures.

No countermeasures against this are specified for the XCDE.

4.4 Stored Program Threats

There are a number of threats associated with stored programs in general that are discussed in the X/Open **XDSF** Guide. They are based on the possibility that someone (for example, a programmer, a service engineer, a system manager, a user or an intruder gaining access to the system, possibly remotely via a network) can modify a program or substitute a different version, and can thereby cause harm. Modified or substituted programs can contain, in addition to their visible useful functions, the following features:

- hidden functions that give unauthorised access to resources (such programs are called *trojan horses*)
- hidden mechanisms that permit protection mechanisms to be circumvented and that are activated in some non-apparent manner, such as by a seemingly random piece of input data (such mechanisms are called *trap doors*)
- mechanisms that perform destructive functions when a particular input condition is detected or on some particular date (such programs are called *logic bombs*)
- self-replicating code that attaches itself to other programs when executed (such code is called a *virus*).

In some cases, an unintentional defect in a program can have the same effect as a trap door that is introduced deliberately. For example, it may result in a utility that has obtained super-user privileges returning control to its caller while those privileges are still in force. A malicious user who knew of the defect could use the utility to give himself super-user privileges.

Stored program threats apply to *dtksh*, *dtcodegen*, *dtappintegrate* and *uil*( ). They also apply to certain XCDE databases (Xt translation tables, Xt resource tables, the data types database, the actions database, the front panel database, and Xlib resources) because these databases affect program execution.

They could be introduced into dtksh, and such like, programs. Alternatively, a database could be changed in such a way as to cause a substitute program to execute in place of the real one. For example, a change to the Xt translation tables could cause an attacker's program to be executed when a particular key is pressed.

Threats to the databases could be realised through the means described in Section 4.5 on page 31.

A particular issue for the XCDE is that it provides the user interface to applications. If the substitute program were, for example, a password grabber, it could enable the attacker to breach an application's defences.

The threats apply to the development utilities themselves, as well as to the applications developed using them. For example, a rogue version of *dtcodegen* might be introduced into a system, and might insert logic bombs, and so on, into *all* of the applications developed using it.

Program threats can be countered by:

- quality assurance during program development
- restricting access to files containing programs.

No countermeasures are specified for the XCDE.

4.5 Data Store Threats

The threats associated with the data stores of the XCDE are similar to those discussed in the X/Open **Data Management: Security Issues** Technical Study. They include the threats listed in the X/Open **XDSF** Guide that apply to filestores.

They apply to:

- X server resource databases
- Xt translation tables
- Xt resource tables
- the data types database
- the actions database
- help files
- UIL and UID files
- the front panel database
- calendar archive files.

The threats are given below.

4.5.1 Back-door Interfaces

Operating system file handling services, backup/restore facilities and low-level disc access utilities, for example, can often be used to read or modify data held on disc.

This can be countered by applying encryption or integrity checking to information held in store.

No countermeasures against this are specified for the XCDE.

4.5.2 Logical Dependencies

A piece of private information can sometimes be deduced from logically related public information. Or modification of a piece of public information can cause a program to change a piece of logically related private information.

This can only be countered by good application design and system management.

A possible instance in the context of the XCDE would be the placing of fake information (for example, for the XCDE resource database) in a publicly accessible directory that is searched before the privately accessible directory containing the real information.

This can only be countered by good application design and system management.

No countermeasures against this are specified for the XCDE.

4.5.3 Re-use of Media

Information can be disclosed if the memory in which it is stored is re-used for other purposes without the information having been erased. For example, a disc might be replaced by a larger one, and the replaced disc drive might be sold with the disc contents still readable.

This can be countered by ensuring that the contents of all storage are erased before the storage is re-used.

No countermeasures are specified for the XCDE.

4.5.4 Utilisation Analysis

In a similar way to communications traffic analysis, analysis of disc or memory utilisation can sometimes disclose useful information.

No countermeasures against this are specified for the XCDE.

4.5.5 Physical Damage

Data can be destroyed when the physical media on which it is stored are damaged. This can be achieved by damaging the data store, by physical means (for example, scratching the surface of a disc), or by electromagnetic means (for example, by exposing a disc to a strong magnetic field).

It can be countered by ensuring that the equipment is kept physically secure (for example, in a locked computer room).

No countermeasures are specified for the XCDE.

4.5.6 Resource Overload

If a disc or other memory store is full, it may not be able to store XCDE data. This could result in XCDE services and applications being unavailable to users. It is likely to affect other subsystems and applications as well as the desktop environment.

It can be countered by resource allocation imposed by the operating system, or by the desktop implementation reserving sufficient space for its needs when it is started up.

No countermeasures are specified for the XCDE.

4.5.7 Repudiation

Someone could store something in an XCDE data store and then deny having done so.

This can be countered by digital signature.

No countermeasures against this are specified for the XCDE.

4.6 Printers

The principal threats associated with printers are as follows:

- Information can be disclosed to unauthorised people if they are able to see printed output; for example, if a printer is installed in a publicly accessible place.

  This threat can be countered by keeping printers in physically secure locations.

- A printer can be used by unauthorised people if print services are publicly accessible on a system. (Although high-quality printers are expensive resources, this is not considered a serious threat in many enterprises. It can be classed as a form of petty theft.)

  This can be countered by applying access control to print services.

- One person making excessive use of a printer can render it unavailable to other users.

  This threat can be countered by resource allocation. If necessary, a system manager can terminate a lengthy print job that is holding up other work.

These threats apply to all printer services invoked via the XCDE, including those controlled by Print Job Services.

No countermeasures are specified for the XCDE.

4.7 Generic Countermeasures

The following countermeasures do not relate to individual threats, but each provides some protection against many of the threats discussed in this section.

None of these countermeasures are specified for the XCDE.

4.7.1 Event Detection

This is the detection of events relevant to security. Relevant events can include attempted security violations and normal events such as users logging on and off.

Event detection may lead to taking action; for example, by generating alarms, changing encryption keys or logging the event as part of an audit trail. This does not prevent breaches of security. It may prevent an attacker from capitalising on them by catching him red-handed.

4.7.2 Audit Trail

An audit trail is created by recording all the significant events that take place in a transaction. While this does not in itself prevent breaches of security, it does make it easier to detect the people responsible, and therefore can act as a deterrent. It also helps a systems manager, when there has been a breach of security, to assess the damage that has been done and prevent it from being repeated.

4.7.3 Security Recovery

This is the taking of appropriate action to recover from actual or attempted security violations detected by functions such as event handling or systems management. Actions similar to those described for Event Detection (in which violations are detected by the XCDE services) can be taken.

Chapter 5

Implementation of Countermeasures

This chapter discusses how the threats identified in Chapter 4 should be addressed. First, it lists the threats that, for a variety of reasons, should be addressed outside the XCDE. It then discusses how the remaining threats can appropriately be addressed within the XCDE.

5.1 Threats that Should be Addressed Outside the XCDE

It is not appropriate for the XCDE to be concerned with countermeasures to all of the threats identified in Chapter 4. This section lists the threats that should be dealt with by applications, by the platform, by physical and electronic security, and by secure working practices, and for which countermeasures should not be implemented as part of the XCDE.

5.1.1 Threats to be Addressed by Applications

The following threats arise only rarely in the context of the XCDE. There may be some applications for which they present problems. However, these applications are few, and the overhead that would be imposed on all applications by countermeasures to them is not justified. It should therefore be left to those applications that are at risk to address these threats:

- traffic analysis of communications
- repudiation of communications messages
- utilisation analysis of data stores
- repudiation of data store information.

5.1.2 Threats to be Addressed by the Platform

The following threats can be more appropriately addressed by the hardware, operating system and communications platform than by the XCDE. In some cases, it is also advisable to apply physical or electronic security countermeasures:

- resource overload via user interface
- back-door access to displayed data
- session hijack at API
- shared memory access
- resource overload via API
- network damage or congestion
- resource overload in data store
- printer threats.

5.1.3 Threats to be Addressed outside the IT Systems

The following threats can most appropriately be addressed entirely by physical security, electronic security and secure working practices:

- re-use of media
- physical damage to data stores.

5.1.4 Security Breaches in Remote Systems

This threat should be addressed by ensuring that appropriate countermeasures are applied in all communications partners. This is a matter of general security policy, rather than of specific measures.

5.2 Threats Requiring XCDE Countermeasures

This section discusses the remaining threats. For these threats, it is appropriate to implement countermeasures as part of the XCDE. (It may also be appropriate to implement countermeasures elsewhere; for example, in the platform.)

In most cases, the identified countermeasures affect the XCDE profile but do not affect any of the three XCDE components that are currently being specified. Exceptions to this are noted in the following sections.

5.2.1 Masquerade of User

Front Panel Services

Masquerade of a user to Front Panel Services can most appropriately be countered by requiring the user to authenticate himself. This authentication could be performed by Front Panel Services or by some other part of the XCDE such as the Session Manager. Alternatively, it could be performed outside of the specified parts of the XCDE, provided that the Front Panel services can rely on it having taken place, and can determine the authenticated user identity.

In view of the fact that many applications and utilities rely on the user identification mechanism of the operating system, the authentication scheme should be linked with the identification and authentication scheme of the X/Open **XPG4** CAE Specifications, and should establish principal identities as understood by that scheme.

XCDE Applications

Each application could provide its own authentication scheme, as Calendaring and Appointment Services do. However, for user convenience, it is better that applications rely on authentication provided by the operating system or the XCDE, than that they force users who have already authenticated themselves to do so again.

It is likely that some applications will rely, either explicitly or implicitly, on the user identification mechanism of the operating system. For example, an application might write data to a file, and rely on operating system file access permissions to prevent different users from accessing each others' files.

The needs of most applications for user authentication can be met by an authorisation scheme that is enforced by XCDE session management and linked to the user identification mechanism of the X/Open **XPG4** CAE Specifications.

Utilities

A utility invoked from the command line could enforce its own user authentication scheme, but it would be more convenient if it could rely on the operating system to authenticate users.

Similarly, a utility invoked from the desktop should be able to rely on the operating system or the XCDE to authenticate users.

The needs of most utilities for user authentication can be met by the operating system authentication scheme plus a linked authorisation scheme that is enforced by XCDE session management.

XCDE Actions

Similar considerations apply to those discussed for utilities, above.

5.2.2 Session Hijack at the User Interface

Desktop

For the front panel and XCDE applications, and for utilities and XCDE actions invoked from the desktop, a session lock facility and a timeout-based screen saver facility are good, practical countermeasures to this threat.

The X/Open **XCDE Services and Applications** Specification requires implementation of a password-based session lock facility, and of a screen saver. It does not, however, specify that the screen saver should (even optionally) require the user to authenticate himself before the screen is restored. Nor does the X/Open **XCDE Services and Applications** Specification make it clear whether a system must provide the capability for the screen saver to be timeout-based.

Command Line

It is not appropriate that the XCDE should provide countermeasures to this threat for utilities and actions invoked from the command line. It could, however, be appropriate to provide a means of preventing particular actions from being invoked other than from the desktop, so that they can rely on being protected from this threat.

5.2.3 Arrogation of Authority at the User Interface

Front Panel

The front panel gives access, amongst other things, to the home directory of ‘‘the user’’. This access presumably includes the ability to delete files, using the *trash can* control. However, the X/Open **XCDE Services and Applications** Specification does not indicate how ‘‘the user’’ is identified, and does not state any requirements for control of access to resources.

It would be appropriate for the front panel services to apply access control on the same basis as the operating system. This would be a natural extension of the authentication and authorisation capabilities discussed in Section 5.2.1 on page 37.

Access control should be applied:

1. to the front panel services themselves — it should be possible to deny access to the front panel to a user who has not authenticated himself
2. to resources that can be accessed via front panel services.

XCDE Applications

XCDE Applications that have desktop user interfaces include Calendar, Electronic Mail, File Management, Text Editing, Icon Editing, Terminal Emulation, Style Management, Application Building and Integration, Action Creation, Print Job Management, Calculator and user-specific applications. Most of these provide access to resources. Between them, they can provide access to all of the resources of a system.[2]

Other than for Calendar and Appointment services, the X/Open **XCDE Services and Applications** Specification and other XCDE specifications do not require access control to be applied.

It would be appropriate for XCDE applications to apply access control on the same basis as the operating system. This would be a natural extension of the authentication and authorisation capabilities discussed in Section 5.2.1 on page 37.

Access control should be applied to the applications themselves and to the resources to which they provide access. Applications are likely to be invoked from the command line as utilities or from the desktop as XCDE actions. The application of access control in these circumstances is discussed below. Application of access control to resources will be enforced by the operating system, provided that applications execute with the correct principal identities (*uid*s). This means, for example, that they should not be allowed to execute with ‘‘super-user’’ privileges.

Utilities

Access to a utility invoked from the command line is generally controlled by the operating system. Within the X/Open CAE, the user must have execute permission on the file containing the executable utility program.

Access to resources via a utility invoked from the command line is also generally controlled by the operating system when the utility uses operating system services to access the resources.

XCDE Actions

Access to resources via XCDE actions could be controlled by the operating system when the XCDE actions use operating system services to access the resources. This requires that the actions must execute with the correct principal identity.

The *dtaction* utility enforces execution with the correct principal identity. (A user can select a particular entity via the **-user** option, but is required to enter a password if he selects an identity other than his current one.) The *DtActionInvoke*( ) function is not explicitly required to enforce it.

Access to command actions, and to map actions that ultimately map onto command actions, can be controlled by the operating system, in the same way as access to utilities from the command line (since such an action has associated with it an execution string that is presumably passed to a shell by *dtaction* or *DtActionInvoke*( )). Again, the user identity used for access control is that assumed by *dtaction* or *DtActionInvoke*( ); this must be the correct user identity if the access control is to be effective.

Access to ToolTalk message actions, and to map actions that ultimately map onto them, could be controlled. This would require extensions to the X/Open **XCDE Definitions and Infrastructure** Specification.

2. In most cases, the resources are not ‘‘owned’’ by the applications, but by the operating system. The calendar is an exception to this.

5.2.4 Masquerade of Application Program

Each application program should, when executing, do so under the identity of a principal. It should be possible for API implementations to rely on that identity.

The identity of the principal is normally established at the user interface. For an API implementation to be able to rely on this identity, the operating system and the XCDE environment must provide authentication, and must make the correct identity available to the API implementation.

As discussed in Section 5.2.1 on page 37, it would be appropriate for the XCDE to use an identification scheme that is linked with the identification scheme of the operating system, and to ensure that identities are authenticated at the user interface.

There are problems with this in distributed systems, however. An application may include processes that execute on different computers. It is undesirable for a user to have to go through an authentication procedure for each computer. However, the operating system security features described in the X/Open **XPG4** CAE Specifications do not provide for an identity that is established on one computer to be *reliably* known in other computers; the only way of making it known in other computers is for the application to communicate it, and a rogue application might not do so correctly.

There is also the problem that a user might have different identities on different computers. This is essentially a management issue rather than a security issue, but it does make the provision of security more difficult.

In the absence of a distributed identification and authentication scheme, security can only be achieved by requiring separate authentication on each computer in a distributed system.

5.2.5 Arrogation of Authority by Application Program

Except for Calendaring and Scheduling, which has its own access control scheme, it would be appropriate for XCDE programmatic interfaces to apply access control on the same basis as the operating system (and to do so by relying on the operating system access control facilities).

For example, the XCDE execution management services should not enable command actions to be invoked by users who do not have execution permission for the command executable files concerned. They can enforce this by ensuring that the executables are executed by the operating system under the correct principal identity, so that if the principal does not have execution permission then the operating system returns an error.

Currently, the XCDE specifications do not require the components of the XCDE to assume the correct principal identity when applications are executed. (There is nothing, for example, to prevent the execution manager from executing actions as super-user.)

5.2.6 Masquerade of Communications Partner

Some of the problems of reliably identifying principals in distributed systems are discussed in Section 5.2.4. The need to prevent masquerade at communications interfaces is another aspect of this.

It would be appropriate for the XCDE to provide authentication for X Protocol communications and for ToolTalk messages.

As an alternative to changing the X Protocol, it would be possible to require that secure lower-level protocols (such as that defined in ISO/IEC 10736) are used. This would, however, still require some changes to the XCDE specifications, to describe how the lower-level protocol facilities should be used.

The above countermeasures would affect the XPG4 X Window System Application Interface or the XPG4 X Window System Application Interface V2 component.

Similarly, it is possible to use authenticated RPC to support ToolTalk. The X/Open **XCDE Definitions and Infrastructure** Specification refers to this possibility, but does not describe it in detail.

5.2.7 Tampering with Communications Media

Again, it would be appropriate for the XCDE to provide integrity checking and encryption for X Protocol communications and for ToolTalk messages. Again, this could be done through the use of secure lower-level protocols and secure RPC, but even this would require some additions to the XCDE specifications.

These countermeasures would affect the XPG4 X Window System Application Interface or the XPG4 X Window System Application Interface V2 component.

5.2.8 Arrogation of Authority by Communications Partner

It would be appropriate for the XCDE to apply access control to X Protocol communications and to ToolTalk messages.

Application of access control to X Protocol communications would affect the XPG4 X Window System Application Interface or the XPG4 X Window System Application Interface V2 component.

Access control can be applied on the basis of authenticated identities. Some issues associated with this are discussed in Section 5.2.4 on page 40.

5.2.9 Stored Program Threats

The powerful facilities of the XCDE make it easy to develop a rogue program that mimics a real one at the user interface. The complexity of the mechanism by which executables are selected and their behaviour is controlled through resource database files gives an attacker good possibilities for introducing such a program into a system.

Some protection can be provided by the operating system access control facilities, which apply to resource data base files. However, the XCDE specifications do not in general state explicitly that this access control applies.

The principal countermeasures are good application development and system management practices, based on operating system access control facilities. It could be appropriate for the XCDE specifications to include advice on these practices.

5.2.10 Logical Dependencies in Data

The considerations discussed in Section 5.2.9 apply to the only threat of this type that it is appropriate to address within the XCDE. This is the threat that fake information could be placed in a publicly accessible directory that is searched before a private directory containing the real information.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The threats that can arise in connection with the XCDE and the countermeasures that can be taken against them are discussed in Chapter 4.

Not all of those countermeasures need necessarily be implemented within the XCDE. The countermeasures that it would be appropriate to implement within the XCDE, to help prevent the threats from being realised, are identified in Chapter 5 on page 35. They are:

- authentication of;
  - users (alternatively, this could be implemented outside the XCDE provided that elements of the XCDE can rely on it and can obtain the authenticated user identities)
  - X communications partners
  - ToolTalk message senders and recipients
- session lock
- screen saver
- access control over access to:
  - the front panel
  - XCDE actions
  - X Protocol communications
  - ToolTalk messages
  - system resources (via the operating system)
- encryption and integrity checking of:
  - X Protocol communications
  - ToolTalk messages.

Access control, integrity checking and encryption applied to X Protocol communications would affect the the XPG4 X Window System Application Interface or the XPG4 X Window System Application Interface V2 component. The other countermeasures listed above apply to the XCDE profile, but not to any of the XCDE components that are currently defined or are currently being defined.

The XCDE specifications cover:

- authentication of X communications partners (but this is partial and unsatisfactory)
- authentication of ToolTalk message senders and recipients (but this is incompletely described)
- session lock
- screen saver (but this is incompletely described)

- access control for X Protocol communications (but this is partial and unsatisfactory)
- access control for ToolTalk messages
- authentication and access control for calendaring and appointment services.

As can be seen from a comparison between the above lists of appropriate facilities and specified facilities, the XCDE specifications do not mandate all of the facilities that are appropriate.

It should also be noted that provision of authentication by Calendaring and Appointment services might be considered inappropriate if authentication is provided elsewhere within the XCDE, or if it is provided outside the XCDE but in such a way that the XCDE can rely on it.

This is not to say that systems currently supplied by vendors are insecure. Some facilities that are missing from the XCDE specifications are provided as a matter of course by most system vendors.

Unless the XCDE specifications require an adequate level of security, however, it will be hard for users to tell whether conformant systems meet their needs. Also, people may come to think that XCDE-compliant systems are insecure. This could damage open systems generally in the marketplace.

6.2 Recommendations

The following recommendations are made in the light of the conclusions in Section 6.1 on page 43.

1. The XCDE specifications should be enhanced to describe how the functionality of the XDSF security model applies to the XCDE.

   The remaining recommendations cover specific aspects of this.

2. It should be possible to prevent users who have not been authenticated from obtaining sessions from the session manager.[3]

3. It should be possible to restrict access to the front panel to authenticated users.

4. In an XPG4 environment, user identities in the XCDE are the user identities of the X/Open **XPG4** CAE Specifications. The XCDE specifications should refer to user identities, and the XCDE profile definition should require those identities to be the same as the *uid*s of the operating system when the operating system has an X/Open brand.[4]

5. The XCDE specifications should state that, when an XCDE component accesses resources via the operating system, it does so under the user identity that was authenticated by the user, except where a particular section of the specifications explicitly states otherwise.

6. Consideration should be given to whether Calendar and Appointment services should be able to apply access control on the basis of authenticated user identities used by other parts of the XCDE, rather than applying authentication itself.

7. The XCDE specifications should explicitly state that access to XCDE command actions is controlled through operating system file access permissions on the executable files for the actions.

8. Consideration should be given to how access to ToolTalk Message actions and to the ToolTalk Message API can be controlled.

9. The XCDE specifications should explicitly state that, when an X Toolkit action or an XCDE action is executed, it starts executing under the user identity that was authenticated to the session manager by the user. They should further state that no action described in any XCDE specification shall change its user identity unless the specification explicitly says that it may do so.

10. Consideration should be given to how authentication of the X client and server can be improved and made mutual. It might require an extension to the X Protocol, or use of a secure underlying protocol. This authentication procedure, and its relation to session manager authentication, should be specified as a user option.

11. Access to X Protocol communications should be controlled on the basis of the above authentication.

12. Provision for encryption and integrity checking of X Protocol communications should be specified as a user option. It could be achieved through an extension to the X Protocol, or through an underlying protocol. (Requirements to use particular underlying protocols should be stated in the XCDE profile definition rather than in the X/Open **X Window**

3. This does not imply either that the session manager must perform the authentication or that the session manager must apply access control. Other parts of the system can authenticate the user and control access to the session manager.

4. The XCDE profile does not require the operating system platform to have an X/Open brand.

System Protocol Specification.)

13. The facilities for authenticating ToolTalk messages[5] should be completely described in the X/Open **XCDE Definitions and Infrastructure** Specification, or a reference to a complete description should be given.
14. One authentication option should be for ToolTalk to use an authenticated RPC. (Requirements to use particular underlying RPC mechanisms should be stated in the XCDE profile definition rather than in the X/Open **X Window System Protocol** Specification.)
15. Access to ToolTalk messages should be controlled on the basis of the above authentication.
16. Consideration should be given to specifying requirements for privacy and integrity assurance for communications, such as those that support ToolTalk, that underly the XCDE.
17. Support for a timeout-based screen saver that requires a password before the screen is restored should be specified.
18. The XCDE specifications should explicitly state that access to resource database files is controlled by operating system file access permissions. They should provide guidance on how these permissions should be used to ensure that the correct files are accessed.
19. The application to the XCDE of the generic countermeasures discussed in Section 4.7 on page 34 should be considered.

5. Authentication may be provided by the underlying transport services rather than by the implementation of ToolTalk itself. Nevertheless, the authentication facilities available through the ToolTalk interface should be described.

Index