X/Open Guide

Defining and Buying Secure Open Systems

X/Open Company Ltd.

X/Open Guide

Defining and Buying Secure Open Systems

ISBN: 1-872630-61-8
X/Open Document Number: G206

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

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@tog.org

Contents

List of Figures

List of Tables

Preface

The Open Group

In February 1996, X/Open and the Open Software Foundation (OSF) joined forces to become The Open Group, which represents one of the leading authorities in open systems. It is supported by most of the world's largest information systems suppliers, user organisations and software companies. By combining their complementary strengths — X/Open in providing specifications and its trade mark branding scheme, and OSF in facilitating collaboration among customers and system/software vendors toward the development of new open systems technologies — The Open Group is well positioned to assist vendors and users to develop and implement products which support adoption and proliferation of open systems.

Established in 1984, X/Open is an organisation dedicated to the identification, agreement and widescale adoption of Information Technology standards which provide compatibility (portability and interoperability) between software products, and so help users realise the business benefits of open information systems — lower costs, increased choice and greater flexibility. It achieves this by combining existing and emerging standards into an evolving set of integrated, high-value and usable open system specifications, which form a Common Applications Environment (CAE). It licenses a trade mark — called the X/Open Brand — for use on products which vendors have demonstrated are conformant to this CAE. The X/Open brand is recognised by users worldwide as the guarantee of compliance to open systems standards. X/Open is also responsible for the management of the UNIX trade mark on behalf of the industry.

Founded in 1988, the Open Software Foundation (OSF) delivers technology innovations in all areas of open systems, including interoperability, scalability, portability and usability. OSF is a worldwide coalition of vendors and customers in industry, government and academia, who work together to provide advanced open-systems technology solutions for use in a distributed computing environment. It runs programmes in collaborative research and development, to deliver vendor-neutral technology (software source code and supporting documentation) in key IT areas. These include OSF/1, Distributed Computing Environment (DCE), OSF/Motif and Common Desktop Environment (CDE).

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focused 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 industry standard. In addition, they can demonstrate product compliance through the X/Open brand trade mark. CAE specifications are published as soon as they are developed, so enabling vendors to proceed with development of conformant products without delay.

- *Preliminary Specifications*

  These specifications usually address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations. They are published for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification; rather, it is as stable as can be achieved, through applying X/Open's rigorous development and review procedures.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and developers are encouraged to develop products on the basis of them. However, experience through implementation work may result in significant (possibly upwardly incompatible) changes before its progression to becoming a CAE specification. While the intent is to progress Preliminary specifications to corresponding CAE specifications, the ability to do so depends on consensus among X/Open members.

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 on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world so as to stimulate discussion and activity in 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 a technical activity. Although at the time of its publication, there may be an intention to progress the activity towards development of a Specification, Guide or Technical Study, the ability to do so depends on consensus among X/Open members.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.
- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Corrigenda

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published on the World-Wide Web at **http://www.xopen.org** under Sales and Ordering.

Ordering Information

Full catalogue and ordering information on all Open Group publications is available on the World-Wide Web at **http://www.xopen.org** under Sales and Ordering.

This Document

This document is a Guide (see above). It has immediate relevance for Information Technology professionals who are building systems dealing with *sensitive data*. In practice, this is the requirement in the vast majority of both commercial and government use.

This guide is intended for anyone planning to purchase a secure open system. No special knowledge of security terminology is required, because a comprehensive glossary is provided.

This document gives clear guidance on the types of secure systems that are appropriate to specific circumstances, providing essential advice for procurement. In doing so, it helps to avoid the exaggerated levels of security that are sometimes specified when purpose and environment are not well defined.

The guide also presents solutions possible with open systems and how these solutions should be implemented, giving users guidance to the best practice in this area.

This guide is structured as follows:

- Chapter 1 provides an introduction to the basic principles of security and the issues involved. It assumes no specialist knowledge. The definitions it provides are an important starting point to the overall problem of procuring secure systems.
- Chapter 2 provides a procurement methodology for secure open systems.
- Chapter 3 describes five examples of environments that require secure systems. Each example is typical of particular kinds of security requirement.
- Chapter 4 identifies the specific threats to the environments described in Chapter 3.
- Chapter 5 discusses technical measures that are dependent on the corporate security policy; they can be applied to prevent these threats from becoming a reality.
- Chapter 6 introduces security classes for X/Open-compliant systems.
- Each of the next six chapters describes one of the security classes from Chapter 6 and explains its relevance in resolving the problems raised in the various examples.
- Chapter 13 provides recommendations for using the X/Open security classes.
- Appendix A is a proforma proprietary information protection policy.

There is a glossary and an index at the end of the guide.

Typographical Conventions

The following typographical conventions are used throughout this document:

- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition.
- User input in interactive examples is shown in `fixed width` font. **`Bold fixed width`** font denotes system output in interactive examples.

Acknowledgements

X/Open gratefully acknowledges:

- The **British Computer Society** (BCS) for permission to reproduce Appendix III of its copyrighted **BCS Guidelines on Good Security Practice**, edited by RAJ Middleton. Appendix III is entitled **PROFORMA PROPRIETARY INFORMATION PROTECTION POLICY** and appears on pages 50 to 53 inclusive of the BCS document.

Trade Marks

Kerberos™ is a trade mark of MIT.

UNIX® is a registered trade mark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open® is a registered trade mark, and the ‘‘X’’ device is a trade mark, of X/Open Company Limited.

Referenced Documents

The following documents are referenced in this guide:

Bellcore
: Bellcore Operations Systems Security Requirements, Technical Advisory, Issue 1, June 1991, Bell Communications Research.

CISR
: Commercial International Security Requirements, Final Draft, September 9, 1991, Ken Cutler & Fred Jones.

ECMA-138
: Security in Open Systems, Data Elements and Service Definitions, December 1989, European Computer Manufacturers Association.

ECMA TC 36
: Commercial oriented functionality class, Draft 4, A contribution to ECMA TC 36, work item 3, July 1992, European Computer Manufacturers Association.

ECMA TR/46
: Security in Open Systems, A Security Framework, July 1988, European Computer Manufacturers Association.

ITSEC
: Information Technology Security Evaluation Criteria, Provisional Harmonised Criteria, June 1991, Version 1.2, published by the Commission of the European Communities.

MSFR
: Minimum Security Functionality Requirements for Multi-User Operating Systems, DRAFT, Issue 1, January 16, 1992, National Institute of Standards and Technology. (Part of the Joint NIST-NSA Federal Criteria (FC) Project.)

OSI Reference Model
: ISO 7498-2:1989, Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.

POSIX
: Security Interface for the Operating System Interface for Computer Environments, POSIX P1003.6, Draft 12, September 1991, sponsored by the IEEE Computer Society.

TCSEC
: Trusted Computer System Evaluation Criteria. U.S. Department of Defense (DOD), 1985 DOD 5200.28-STD, National Computer Security Center, Fort Meade, Md. (also known as the Orange Book).

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

System Interfaces and Headers

XSH, Issue 4
: X/Open CAE Specification, July 1992, System Interfaces and Headers, Issue 4 (ISBN: 1-872630-47-2, C202).

System Interface Definitions

XBD, Issue 4
: X/Open CAE Specification, July 1992, System Interface Definitions, Issue 4 (ISBN: 1-872630-46-4, C204).

Security Guide
: X/Open Guide, 1990, Security Guide (Second Edition) (ISBN: 1-872630-07-3 G010 or XO/GUIDE/90/010).

XPG4

XPG4
: X/Open Systems and Branded Products: XPG4, July 1992 (ISBN: 1-872630-52-9, X924).

The following documents were used in the development of this guide, but they are not explicitly referenced:

- Glossary of Computer Security Terms, NCSC-TG-004, Version-1, 21 October 1988.
- Glossary of Information Technology Security Definitions, Initial Draft, SC27 WG1, October 1985.
- Simson Garfinkel, Gene Spafford: Practical UNIX Security, O'Reilly Associates Inc., Sebastopol CA, 1991. ISBN 0-937175-72-2.

Chapter 1

Introduction

This chapter outlines the implications of security in computer-controlled data processing and information systems.

1.1 Overview

Commercial companies and governments share a central desire — they want every system installed to be totally secure. Unfortunately, like every absolute, the closer they become to achieving it, the more they have to invest in making further gains. The law of diminishing returns applies; the possible consequences of over specifying security can be as costly as the reverse.

There is a common misconception that open systems are less secure than proprietary ones. However, this is not usually the case. Typically, an open system implementation is made available at the source-code level to the development community of system suppliers and consequently has greater exposure. Under these circumstances there is less likelihood of vulnerabilities or weaknesses going undetected.

In contrast, many proprietary systems are maintained and enhanced by a relatively small community. This may mean that certain areas of the system do not come under the fine scrutiny of examination that leads to enhanced quality. Also, in this less exposed environment there may be a greater opportunity for developers to install back doors into the software being developed.

In addition, many open systems are scrutinised by independent evaluators expressly checking that the security features have been implemented correctly and that the features are effective in meeting security goals.

The advice contained here is restricted specifically to the security capability provided by the operating systems of open systems. Applications running on these platforms should make use of the security capability provided (if it is suitable for their policy).

This guide does not attempt to address the issues raised by connection of systems to either homogeneous or heterogeneous networks. This can involve high risks, especially when connected to worldwide open networks, which are not under the user's control.

Security with a system of any kind is difficult to achieve. The difficulty lies in correctly assessing the nature of the threat to be guarded against. Security is defeated, not by openness, but by the incorrect analysis of the problem.

This guide presents the view of security in open systems that is current at the time of writing. It is not totally comprehensive, but covers the vast majority of situations in commercial and governmental systems that deal with sensitive information. This covers more than 95 per cent of all real-life situations.

1.2 Approach

This guide considers the application of X/Open-compliant systems to various situations where security is important. To do this, classes of X/Open-compliant systems are introduced based on:

- open systems that comply with the XPG4 Base profile defined in the referenced **XPG4** binder
- those that include the requirements detailed in the emerging POSIX .6 standard.

The functional classifications for the basic security features of open systems do not define new security functionality classes for the commercial area. Instead of defining requirements, they describe the capability available today in open systems.

The ability of systems to remain secure is expressed in terms of the generic headings of the referenced ITSEC specification. It is therefore possible to compare the description of available security technology for open systems given in this guide with requirements using the ITSEC structure.

If the functional requirements for a system have been clearly defined in the beginning of the procurement process, the classes described in Chapter 7 to Chapter 12 help to establish the extent to which these requirements are covered by open systems available today.

The aspects of assurance (effectiveness and correctness) are treated as less important, although there is information about the meaning and value of assurance classes in the evaluation criteria.

1.3 Security — Toward Definition

Like any other subject of continuing importance to the Information Technology (IT) industry, attitudes to security continue to evolve. Since the 1983 publication of the first evaluation criteria, IT security has become a major factor in the commercial world. It has also become evident that the threats to security are not principally those that concern military IT users.

In the military environment, the principal threats are seen to be the unauthorised disclosure of information, its unauthorised modification and the unauthorised withholding of information or resources.

In the commercial world however, the highest priority is usually *integrity*, followed by *availability* and *confidentiality*.

Additional concerns have also been brought into focus in recent years. The protection of applications against malicious and intentional attacks is balanced by the need for protection against accidental events that may disrupt the use of the system and cause damage.

IT security involves terms such as:

- security needs
- assets
- possible damage
- objects to be considered
- threats
- vulnerabilities
- risks
- measures
- residual risks.

IT security is the state of an IT system, in which the risks of the IT system's application due to relevant threats are reduced to an acceptable level by taking appropriate measures. IT security depends on:

- the purpose for which an IT system is used
- the IT system itself
- the environment in which it is used.

Figure 1-1 and Figure 1-2 on page 4 illustrate the relationship between the keywords of IT security.

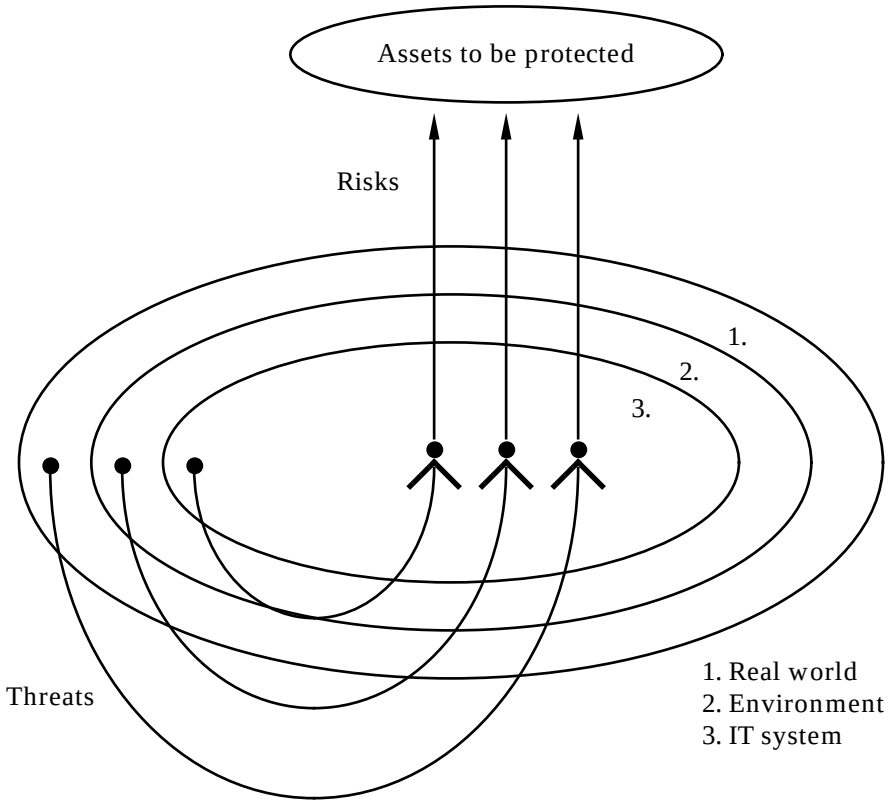

Figure 1-1 Keywords of IT Security 1

From the point of view of IT security, the world is divided into three parts:

- the *IT system*, encompassing hardware, software and other equipment
- the *environment*, including staff, organisation and infrastructure
- the *real world*.

The purpose of IT security is to protect *assets* against *threats*. Threats may originate from any of these three parts of the world. The possibility that the IT system is affected in an unwanted manner, combined with the damage caused to the assets, is called *risk* (see Figure 1-1).

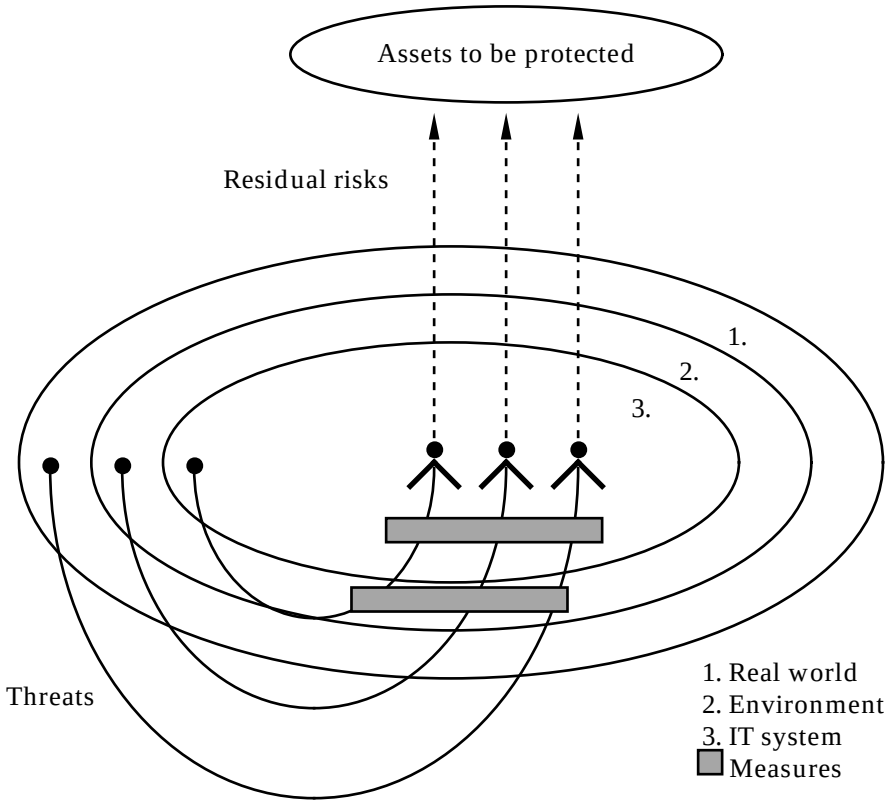

Figure 1-2 Keywords of IT Security 2

To prevent damage to the assets and to reduce the risks, *measures* are taken in the environment and the IT system. Measures in the IT system are represented by security functionality. The risks remaining in spite of the safeguards are called *residual risks* (see Figure 1-2).

These terms are explained in more detail below.

1.3.1 IT Security

The background to IT security is the commercial point of view that IT serves as a technical means for fulfilling business tasks under the rules of cost-effectiveness. The assets are represented by the application, which includes the information and its processing supported by an IT system, for example:

- text processing in an office
- communication systems
- scientific computations
- a bank accounting system
- an airline reservation system
- a software development environment.

Usually the values of the assets are much higher than the investment in the equipment. An organisation using IT is dependent on the reliability of the system and its correct functioning. The extent of dependency and the form it takes can only be determined individually by each organisation itself. To achieve this, the overall business goals of the organisation, the IT environment and the corporate security policy must be taken into account. To determine the

dependency means establishing the basic security needs in terms of integrity, availability and confidentiality which are fundamental for IT security.

1.3.2 Threats

There are many events or circumstances, called threats, that may affect the IT application and may cause damage in different areas, for example:

- personal injury
- damage to property
- social or political embarrassment
- violation of laws or internal regulations
- direct or indirect financial loss.

Examples of threats are:

- fire
- breakdown of the power supply
- hardware failure
- computer virus infection
- unauthorised access to, deletion of or change of data
- theft of storage media
- misuse.

Threats directly concern *objects* such as:

- personnel
- infrastructure
- software
- paperware
- storage media
- application data
- communication.

There are many reasons for the occurrence of threats, for example:

- human error
- malicious attack by insiders or outsiders
- technical failure.

Vulnerabilities are the properties of objects that make them susceptible to threats. The features of IT result in many vulnerabilities. The possible impact of a threat with respect to the application must be considered and assessed. This results in the risk associated with the threat. The risk is a measure of the importance or relevance of a threat in a particular environment. A measurement for the risk is often defined as the result of multiplying the probability of the occurrence by the cost of the potential damage.

1.3.3 Measures

To protect the assets, that is to reduce the risks and to counter the identified threats, measures must be chosen and introduced. Precautions must be taken for events that have never occurred but may cause severe damage if they do occur. Measures realised in hardware or software must be combined with measures concerning the personnel, the organisation and the infrastructure. Also, measures in communication technology, protection against compromising emanations, disaster provisions, and IT insurances may contribute to prevent unwanted effects. Every provision preventing the occurrence of a threat, reducing or limiting a possible damage, enabling or facilitating damage discovery or damage repair, is useful in this context.

After measures have been taken there still remains the residual risk. The residual risk must be reduced to an acceptable level. The measures must be chosen so that they are fit for purpose and provide a universal, continuous and balanced protection. The costs associated with measures are considerable and must be cost-effective: there must be a reasonable cost-benefit ratio for all measures.

A fundamental provision is that there is at least one person in the organisation assigned to IT security. IT security must be taken into account from the planning of a project to the implementation of the IT system. For example, IT security must be considered before procuring an IT system. In many cases this is cheaper than enhancing the security later and is more readily accepted by the users. The acceptance of IT security measures is fundamental and is also a matter of the awareness of all the people concerned with the IT application. For example, if people are aware that the connection of their system to a wide area network causes more and higher risks, they are likely to accept stronger protection and probably do not try to corrupt it.

Existing open systems provide a choice of security measures (see Chapter 5 on page 57).

Chapter 2

Procurement Methodology

This chapter first outlines how to procure a secure IT system. It then presents a minimal procedure which can be used with this guide to derive the requirements for security functionality.

2.1 Good Practice Procurement Methodology

The following stages should be incorporated in the procurement process:

1. The first stage is to specify the complete IT application or IT system and to determine the protection requirements from the user's point of view. This reflects the fact that security is a feature of the whole and cannot be achieved by only looking at the security of individual parts. For example, security cannot be achieved by just looking at the operating system.
2. The second stage is the *threat analysis* in which all relevant threats for this system must be identified. This is a more technical matter and presupposes facts about the hardware and software to be used and the environment in which it will be used.
3. The third stage is the *risk analysis*, by which the importance of the identified threats must be assessed.
4. In the fourth stage, a policy must be formulated for dealing with those threats and associated risks in a manner that is commensurate with management business considerations, goals and priorities. Based on relative cost, feasibility, availability of trained resources and so on, a high-level executive policy should be defined that distributes the responsibility to combat these threats between physical, personal, organisational and technological measures. IT security specialists, possibly with the assistance of professional consultants, must then take the technological section of the general policy and map it into detailed policy statements in terms of known topics such as identification and authorisation, access control, audit and integrity. Appendix A contains a proforma proprietary information protection policy.
5. The final stage is to define the hardware and software capability required to implement the detailed security policy. Success or failure of the capability would be determined by evaluating its ability to support the corporate security policy at the detailed and executive levels.

This gives the *system security functionality* that must be provided by the system being procured. The *assurance* that must be placed on the security functionality depends on the risks which must be reduced. The higher the need for reducing a risk, the higher the required confidence that the security functions work as claimed and cannot be compromised or deactivated.

This process consists of the following steps:

1. Specify the system or application under consideration, including boundary definitions.
2. Determine the protection goals concerning integrity, availability and confidentiality for the application and the information being processed. They may be considered as non-technical assets to be protected.
3. Assess the value of the assets.

4. Describe the objects involved in the application (technical assets), such as personnel, application data, communication, hardware, software, documentation, storage media and infrastructure.
5. Determine the relevant threats to which the objects described are exposed.
6. Determine the vulnerabilities of the system and the application.
7. Record the measures presupposed to be in the environment.
8. Assign values to the objects under consideration that are inherited from the values of the non-technical assets.
9. Estimate the frequency of threat occurrence.
10. Consider the risks of the threat as a combination of the frequency and the endangered values.
11. Write a corporate security policy at the general and detailed level that defines how these threats must be addressed and that can be used as a basis for evaluating the adequacy of the various measures taken.
12. Derive requirements for security functionality that are needed to support the corporate security policy.
13. Derive the needed assurance from the efficiency of the technical measures required to reduce the risks.
14. Choose a suitable system or a combination of IT products that fulfills both the operational requirements and the derived security requirements.
15. Check the cost-benefit ratio of the chosen system.
16. Check the acceptability of all the residual risks remaining after the measures are taken.

In practice, some parts of this procedure may need several iterations to reach the required goal.

This procedure should be followed wherever possible. Parts of it, especially the risk analysis, may be performed by external consultants who are specialists in this area. In some cases, however, it might be inappropriate to go through the complete procedure, for example, if there is a lack of resources. In this case, the minimal methodology described in Section 2.2 on page 9 should at least be used.

Note: Inadequate security functionality may compromise IT security.

2.2 Minimal Procurement Methodology

For the purpose of procuring secure open systems, the following is the minimal procedure that can be used to derive the requirements for security functionality.

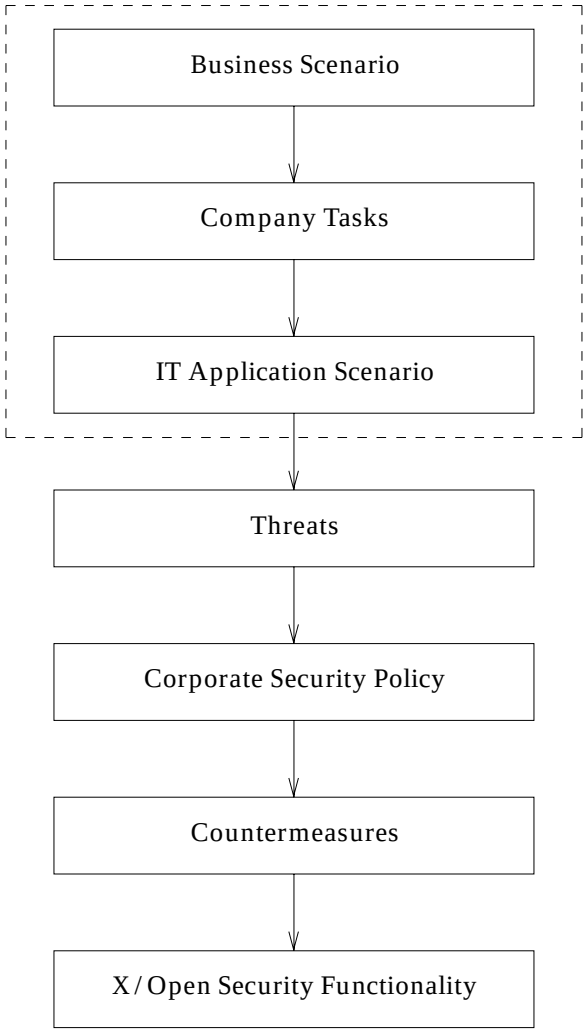

Figure 2-1 Minimal Procurement Procedure

The minimal procedure has five steps. Considering the business scenario and the company tasks supported by IT, a suitable IT application scenario must be identified from Chapter 3. The next step is to determine the relevant threats using Chapter 4. As soon as the security policy for the organisation is defined, the measures required in the IT system to address the threats can be identified, as described in Chapter 5. The final step is to choose the appropriate security functionality, described in terms of the X / Open security functionality classes of Chapter 6.

This minimal procedure is a reduced version of the complete procedure described in Section 2.1 on page 7. It does not take into account the following:

- the value of the assets to be protected
- which risks arise from the threats
- whether the complete set of countermeasures is appropriate
- whether the residual risk is acceptable.

However, it does ensure that all identified threats are countered by a measure.

Chapter 3

Scenarios

This chapter discusses five examples of environments that need secure open systems:

- office systems (see Section 3.1 on page 12)
- database applications (see Section 3.2 on page 18)
- batch applications (see Section 3.3 on page 23)
- software development (see Section 3.4 on page 28)
- control systems (see Section 3.5 on page 33).

They cover a wide range of applications in different business areas, such as:

- banking
- insurance
- general office work
- research
- retailing
- wholesaling
- industry
- medicine
- administration.

The scenarios may overlap because the same applications can be used in different business areas. For example, the execution of a makefile to produce software can run as a batch program. Also, a batch program can perform transactions on databases. Additional access to databases may be necessary during office work.

On the basis of the five scenarios, potential threats to security are identified and discussed in Chapter 4.

Each example description is in the following format:

- A brief overview is given so that you can assess its relevance.
- A typical task that uses IT is described.
- Boundary conditions and regulations are considered. These may originate from statutory or internal regulations. Obeying such regulations may affect the IT systems.
- Basic security requirements are identified with emphasis on:
  - integrity of information
  - availability of information, programs and resources
  - confidentiality of information.

To make the examples self-contained some information is repeated in each description. In all cases the examples are representative, rather than statements of mandatory procedures.

3.1 Office Systems

This scenario deals with work performed in industrial or governmental office organisations. Most of this work is *case handling*. Case handling has two different aspects:

- the *work flow* from one office worker to the next worker
- the *routine work* on the documents representing the case.

Normally office organisations are hierarchical. The exact structure depends on the goals and interests of the organisation. Many different tasks must be performed:

- central tasks, such as payroll accounting
- individual activities, such as simple calculations, looking up information or writing text
- case handling.

Usually the office work carried out by individual workers is part of other tasks. Office procedures specify the steps to be performed and by whom.

In the following section a typical office task is used as an example of case handling.

3.1.1 Office Task

Usually the work carried out by individual workers is part of an encompassing office task. Procedures specify the steps to be performed and by whom. Normally, the performer is named as an organisational unit or an office worker filling a post or having certain responsibilities; that is, playing a certain role. The stricter a procedure is specified, the more formalised the task. Figure 3-1 shows a simple sequential office task.

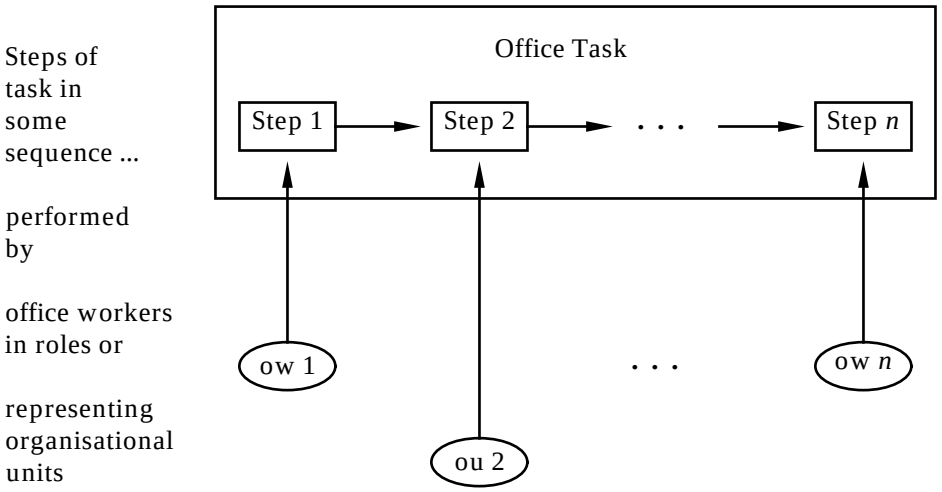

Figure 3-1 General Performance of Office Task

An approved and very flexible conventional tool for supporting the processing of office tasks is the *circulation folder* (CF) or distribution envelope. Its contents consist of arbitrary but task-related documents. They are worked on by performers who are specified on the cover of the CF by other office workers at their discretion but in accordance with the organisational procedures. An internal messenger service handles the transfer of a closed CF from an office worker's out tray to the in tray of the next worker. A clear distinction can be made between routine work on the contents of a CF and its transfer through an organisation (the flow of work). Parallelism occurs if copies of documents are put into folders of their own and these folders are forwarded independently.

The hypothetical example is a task that is performed with the help of a CF. The example deals with handling a request made by a member of the public to a public administration office. Handling a customer's order in a company would look very similar. The preparation of the answer is normally routine work.

One possible procedure is shown in Table 3-1.

Step	Organisational units and office workers					
	Internal postal service	**Registry**	**Head of department**	**Clerk in charge**	**Secretary**	**Another department**
1	X					
2		X				
3			X			
4				X		
5					X	
6				X		
7			X			
8		X				
9	X					
10		X				

Table 3-1 Overview of Office Task

The steps are numbered and named with the role. The documents of the case may be forwarded as the contents of a CF by the internal messenger service or electronically (by electronic mail, for example).

Individual Roles

Each step in the procedure concerns a particular role; possible IT support for parts of the task is described for each step:

1. Internal postal service
   The internal postal service receives a letter, places an entry stamp on it and names the department concerned in the appropriate place.

   IT support — Scanning converts a paper document into images in an electronic multi-media document. The entry stamp becomes a *document frame* of its own in the electronic document. Also, incoming electronic mail, telex and fax can automatically be put into an electronic document and stamped in the same way.

2. Registry
   In the registry the incoming letter is logged. If necessary, related records are added to the CF.

   IT support — The letter received is put into an electronic archive. This is in accordance with an electronic record plan maintained in a database. Also, it must be indexed for later information retrieval. Other documents that are already in the archive and are related to the case must be added to the archive.

3. Head of department
   The head of department evaluates the letter, makes remarks and gives instructions and delegates the preparation of an answer to the clerk in charge.

IT support — An advanced document processing system can be used for some instructions as written annotation or as voice annotation. It may be necessary for both the head of department and the clerk in charge to work on the document at the same time.

4. Clerk in charge
   After evaluating the letter, the instructions and remarks, the clerk in charge starts the routine work. This includes looking up case-related information (such as encompassing records, legal books, external databases, internal lists), looking for similar cases, performing calculations and phoning.

   IT support — The clerk in charge needs the most varied types of support, for example:

   - text and document processing
   - access to personal, internal or external databases for obtaining relevant information
   - personal processing facilities such as pocket calculators, calendars, spreadsheets, business graphics
   - the facility of retrieving similar cases from the archive.

   The result is a draft of the reply.

5. Secretary
   The secretary types the letter.

   IT support — In the case of an office supported by IT the secretary may not be involved further. Nevertheless, the secretary needs similar IT support to that required by a clerk in charge.

6. Clerk in charge
   The clerk in charge checks that the letter is correct.

   IT support — If the answer is developed iteratively, that is, the case goes back several times to the head of department, a modification history may be required.

7. Head of department
   The original answer is approved by the head of department who signs the letter and gives instructions for further processing.

   IT support — The final result is signed using a signature handling method which cannot be forged.

8. Registry
   If necessary, additional copies of the answer are made in the registry and forwarded to other departments.

   IT support — Copies must be made electronically and forwarded to other departments, according to the instructions of the head of department.

9. Internal postal service
   The internal postal service confirms the mailing of the final version on the copy.

   IT support — The results of the case handling are printed and mailed to the outside world.

10. Registry
    The copy is handled in the registry according to the instructions, so that it may be resubmitted at a later point in time.

Practical Considerations

In real life there are many reasons for deviations from this procedure, some of which are mentioned here. An office worker may send the documents back to a previous worker in the chain for clarification. If the case is not simple, a superior or other departments may have to be involved in the signature procedure. Also, it may be necessary to postpone the case and resubmit it when missing information is available.

The case considered is initiated in the world outside the office organisation. There are, however, many office tasks that are initiated inside, such as a requisition order, travel handling (booking, expenses, allowances) or application for leave. In the performance of these tasks the internal postal service, handling mail from or to the outside world, is not involved. Despite that, they are handled in a similar way.

To support the routine work and the transfer of work, an office system that replaces the CF by an electronic counterpart is desirable. This needs to offer the same flexibility while overcoming the drawbacks of the CF through additional and novel possibilities. Electronic CFs are much more than electronic mail, as they have the following features:

- They automatically migrate to office workers in organisationally-defined roles.
- They are automatically resubmitted.
- They provide many kinds of exception handling.
- They allow their whereabouts to be deduced.
- They keep histories of their migration.
- They allow the integration of arbitrary application programs for supporting the routine work on the folder's content.

Systems offering these features are just emerging.

The many different aspects of office work are reflected in many different forms of IT support. In addition to the facilities already mentioned the following are available:

- standardised office document architectures
- order handling
- desk top publishing
- expert systems
- hypertext
- forms handling
- planning tools.

Hardware architectures for running an office system such as the one outlined can consist of workstations or powerful PCs connected by a local area network. An alternative is a client-server configuration with, for example, X-terminals. Adequate hardware for the archive is an optical disk.

3.1.2 Boundary Conditions and Regulations

Work in offices can be restricted by limiting conditions. Also, it is controlled by different regulations that are either set up by the government or internally by the company or organisation itself. The following sections contain some examples of regulations that may affect office work.

External Regulations

Examples of statutory regulations are:

- privacy legislation
- Value Added Tax regulations
- recommendations concerning the use of standards in governmental offices
- legal requirements on the elapsed time permitted before a response must be made by a government office.

Internal Regulations

Examples of internal regulations are:

- rules for handling office tasks as they are laid down in an organisational handbook
- instructions on how to deal with documents, for example how to store them in accordance with a record plan
- instructions on how to deal with confidential material
- instructions on how to deal with deadlines in contracts
- directions on software standards to be employed
- directions on the internal development of applications
- administrative regulations covering procurement of hardware and software, system administration (including the installation of hardware and software, and maintenance), user administration (including the introduction of users), and so on.

3.1.3 Basic Security Policy Requirements

Integrity, availability and confidentiality of information are considered in greater detail with respect to office systems in the following sections.

Integrity

- Documents must not be modified by unauthorised office workers or outsiders.
- In an open system application, programs can be integrated for supporting the routine work on subtasks. They must behave in accordance with the organisation's security policy.
- Accountability must be provided for certain types of cases dependent on the organisation's policy.
- Histories of the routine work on documents may have to be kept.
- If an office procedure has to be cancelled, work done so far may have to be undone or compensating changes made.

Availability

- Cases and their documents must not be mislaid during their transfer from one station to the next one.
- At any point in time a document's location may be requested. Cases and their documents must not be mislaid by workers.
- To perform the routine work on an office procedure's document, the office worker may need to have access to support that goes beyond text processing. The support required might include calculators, spreadsheets, business graphics, calendars or information kept in personal, internal or external databases.

Confidentiality

- Confidential information may be read by authorised office workers only and must not be disclosed to outsiders.
- Confidential information must not be copied into files accessible by unauthorised users. This is a problem especially in office systems offering advanced user interfaces, where this may happen by cut and paste operations between windows.

3.2 Database Application

A database is a collection of stored operational data.

Most organisations have to maintain operational data that is kept in a database system offering *transaction processing*. Access to the data is carried out inside *transactions*. A transaction is a complete unit of work that is atomic. A transaction makes changes that always leave the database in a consistent state. Either all the necessary changes are made (the transaction is committed) or none are made (the transaction is rolled back). A transaction is either interactively triggered or executed as part of a *transaction program*. Some organisations use many different transaction programs performing a large number of transactions per day.

Manufacturing companies, banks, hospitals or government departments have to maintain a lot of operational data. Examples are product data, account data, bookkeeping data, patient data or planning data.

Normally, a database management system (DBMS) is employed to keep and provide access to this operational data. To that end the DBMS offers functions for data definition and for data manipulations such as insertion, deletion, update and query. Both data definitions and manipulations are performed inside transactions. There are different reasons why a transaction may fail:

- hardware failure
- software error
- internal inconsistency in the database.

Banks are organisations that run many transactions each day (more than a million for the large banks). The scenario considered leads to the execution of transactions.

The performance of transactions is governed by organisational rules. Also, statutory regulations must be taken into account. Finally, requirements can be recognised that are concerned with safety as well as security.

3.2.1 Cash Withdrawal

The example deals with a customer withdrawing cash from his account at a bank's branch. This process starts at the branch and triggers an appropriate transaction in the DBMS at the bank's central mainframe. Similar sequences of activities can also be found in other organisations. Transactions in a DBMS are often triggered by events in the outside world.

Several IT components are involved:

- a terminal (often an intelligent one or a PC) at a branch of the bank
- a telecommunication line
- the mainframe at the central bank office.

This example is illustrated in Figure 3-2 on page 19 and is considered in more detail in this section.

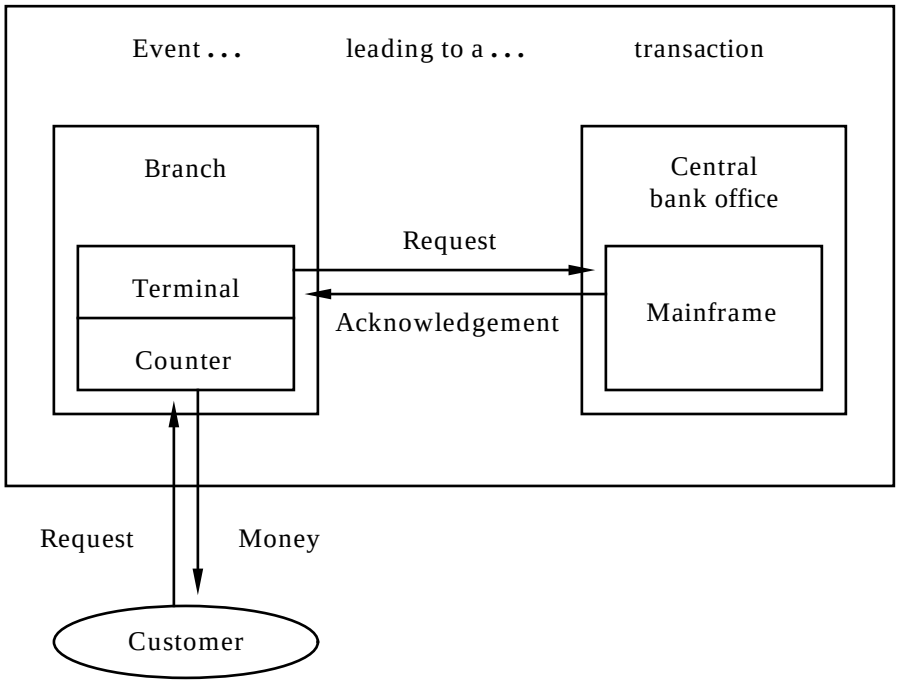

Figure 3-2 Bank Event Triggering a Transaction

Another example is a customer's bank transfer order which follows a similar pattern. However, instead of money the customer gets a receipt, and the bookkeeping on the mainframe is performed by another appropriate transaction program. Also, the results of bank transfer orders may have to be forwarded to the target bank by an intervening central bank. This is done by means of tapes or, if possible, by direct communication lines.

One possible course of processing the cash withdrawal example is shown in Table 3-2.

Step	Customer	Cashier	Terminal	Mainframe
1	X			
2		X		
3			X	
4				X
5			X	
6		X		
7	X			

Table 3-2 Overview of Cash Withdrawal

Sometimes an additional clerk is involved who performs some of the cashier's work. Some steps are performed by people and some by IT equipment:

1. A customer wanting to withdraw some cash fills in, signs and hands in a request form, possibly guided by the cashier.
2. The cashier must first check the customer's account number and signature.

3. At the terminal, the cashier then selects an appropriate service that provides guidance for entering the account number as well as the amount of money asked for. Finalising the request triggers a transaction at the mainframe.
4. The mainframe performs double-entry accounting.
5. An acknowledgement is displayed at the terminal indicating a successful result of the transaction.
6. The cashier stores the request form.
7. Finally, the cashier hands out the money to the customer.

Practical Considerations

In practice, there may be many reasons for deviations from this regular flow of handling cash withdrawal, for example:

- terminal is down
- communication line is down
- mainframe is down
- transaction reports that there is not enough money available on the account
- an irregularity, for example a deficit of the branch's cash at hand, is detected during the daily balancing.

Disorder of the mainframe is disastrous for the bank. However, many precautions are taken to ensure that this is a very rare event. Disorder of a terminal is not serious; often, there is another available. Problems with the communication line occur from time to time. Regardless of the bank's problem, the customer expects to receive the money.

If there is not enough money in the account, the cashier can discuss the matter with the customer. The customer may claim that another transfer of money to the account has already been made but has obviously not yet been credited. If the customer is known to the cashier, it is easier to make a decision; in some cases a credit has to be discussed.

An error for which the bank is responsible must be cleared as soon as possible by the branch's employees.

Example System

An example of a secure system to meet a bank's requirements is as follows:

- The terminal is connected to the mainframe by means of an intelligent control unit. For accountability purposes this unit logs all information from the terminal. The logs are the basis for the daily balance. Also, they are exploited to overcome a temporary disconnection. When the communication line is again operational, the log information on the transfers intended to have taken place in the meantime is forwarded to the mainframe.
- The control unit must also be able to perform checks such as the syntactical correctness of account or bank numbers.
- The database applications of a bank have to cover about 1500 different situations. In the example, only two different transaction programs are involved:
  - — One is for the normal course of processing that checks the request with respect to account number and availability of money and returns either accepted or rejected.

— The other executes the same bookkeeping activities, but without checking, because it must accept the cashier's decision.

- For accountability purposes all updates to the database are logged. Often, the queries are also logged.
- One possible approach is to run the transactions on a daily and reduced working copy of the database. During the night the updates are made on the main database.
- Banks obey a basic integrity constraint: in each bookkeeping activity two accounts are involved. These are the credit account and a debit account. In the example they are the customer's account and the branch's cash account. The constraint is that the sum of the two accounts is the same before and after bookkeeping.
- Normally, the main software components employed for bookkeeping and local support are commercial products, for example, database systems or office software. However, customised software can also be included in the system, which may originate from third party or in-house development.
- The central mainframe also has many connections to terminals or PCs to provide staff with information or other support for their clerical work.

3.2.2 Boundary Conditions and Regulations

Executing database applications is governed by different regulations that are set up either by the government or internally by the company or organisation itself. Examples are given in the following sections.

External Regulations

There are statutory regulations covering privacy, civil law and commercial law. Also, there is the bankers' discretion or the bankers' duty of secrecy. This means that accounts and transfers must be kept secret.

Internal Regulations

Examples of internal regulations are:

- As the continuous operation of the mainframe is vital, it is common to protect it by guarded entrances and safety zones.
- There are rules governing the installation of software for local support on the office workers' PCs. Only approved software should be executed.
- Accountability is vital for a bank. Thus, many regulations focus on achieving accountability. Also, there are rules on audits.
- Many regulations exist which deal with correct bookkeeping.
- Development of software is governed by rules concerning methods to be applied.

3.2.3 Basic Security Policy Requirements

Integrity, availability and confidentiality of information are considered in greater detail with respect to database applications in the following sections.

Integrity

The transaction concept is necessary for observing the integrity constraints of the database. It is, however, not sufficient for protecting against modification which is unauthorised. Other requirements might be:

- Only authorised modification of operational data (account data in the example) should take place. This means that the transaction programs performing modification must be approved. There must not be a way around these approved programs.
- Only approved software should be executed on PCs.
- For accountability, certain activities must be logged so that audit trails can be made.
- A consequence is that unauthorised modification of logging data must be made impossible.

Availability

- The ultimate goal is that a customer can be served at any time and under any circumstances. That presupposes the availability of the DBMS and of the transaction programs. That, in turn, presupposes the availability of the terminal, the communication line and the mainframe. Appropriate measures must be provided to overcome failures such as broken lines.
- A general and central requirement is that a bookkeeping activity must not be lost.
- To perform office work, bank clerks may need to have access to different supporting functions (such as those provided by integrated office packages) or information kept in internal databases.

Confidentiality

Confidential information may only be read by authorised clerks and must not be disclosed to outsiders. Examples of confidential information are personal customer data, account balances, bank transfers, and so on.

3.3 Batch Applications

Processing large amounts of data without the need for interaction is usually done by batch programs. Often, they are embedded in encompassing tasks.

In the past, a typical batch program was tangible: a set of punched cards keeping both program code and data. Nowadays these are permanently stored. The main characteristic of a batch program is that, once started, its execution cannot be influenced interactively. In general, a batch program reads input from possibly different sources and produces output into possibly different targets. Examples of batch programs are compilations, executions of makefiles, large calculations, payroll accounting or applications accessing databases. Batch programs are used on all kinds of computers ranging from PCs to mainframes. On mainframes, batch programs often deal with a large quantity of data. One of the goals is to optimise the use of resources and to maximise the throughput. Batch programs can run as background or foreground processes. Also, batch programs can be started at a remote station to which they send back their output.

In general, a batch application is more than just a single batch program. It may consist of several batch programs which are controlled by shell scripts or procedures written in a job control language. A task may involve not only batch applications but also interactive applications and manual office work. A simple example is presented in Figure 3-3.

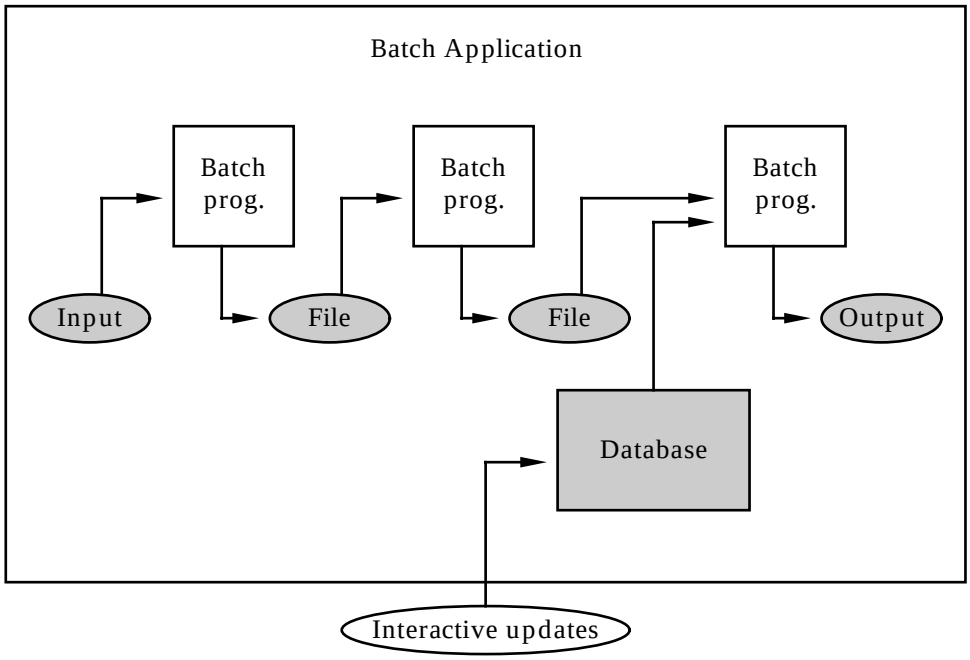

Figure 3-3 An Example Batch Application

The example considered is monthly accounting based on an organisation's administrative data. It consists of payroll and project accounting and is a conglomerate in which batch applications make up the essential parts. Monthly accounting is a sophisticated application which, in principle, is continuously in operation. To explain the individual batch applications, they are considered in their context.

There are limiting conditions for administrative data processing: statutory and internal regulations govern processing and operation. Also, requirements can be recognised that are concerned with both safety and security.

3.3.1 Monthly Accounting

The term *monthly* does not mean that this task is performed only once a month. Instead it is a continuous execution of subtasks and activities. They all serve the goals of both payroll and project accounting. To that end, information on the allocation of work originating from employees has to be entered by means of a document reader, checked by clerks and processed on a mainframe. In the example this takes place every ten-day period. In addition, after each month the results of its three ten-day periods must be merged and further processed, resulting in the monthly closing. Not only salary payment depends on these results, but also the organisation's cost accounting and financial accounting. Due to its complexity the monthly payroll and project accounting cannot be considered in detail. However, the sequence of its main subtasks shows how batch processing is embedded in this task. Table 3-3 shows a simplified flow of work in the example task. Its subtasks are outlined in the following paragraphs.

Step	Document reader	Batch	Interacting clerk at terminal	Office worker at desk
after the end of each ten-day period				
1	X			
2		X		
3				X
4			X	
5		X		
after the end of each month				
6		X		
7			X	
8		X		

Table 3-3 Overview of Monthly Accounting

Flow of Work

The steps identified in Table 3-3 are as follows:

1. Entering basic data
   The basic data consists of the employees' *on premises* times which are kept on clock cards and the allocations of manpower to projects entered into time sheets by the employees. This data is read by means of a special device *document reader* and stored on intermediate storage.

2. Generation of reports
   The basic data is read from the intermediate storage and entered into a database. Reports are generated on the allocation of the working hours of the employees and of other accountable work. In the case of obvious inconsistencies, such as non-existent project numbers, additional error messages are issued.

3. Checking reports
   The computed allocations must be checked in each department and, if necessary, annotated with correction requests. In particular, errors reported must be taken into account.

4. Corrections
   Members of the wage and salary office have to enter the corrections into the database interactively.

5. Balance of ten-day period
   Different reports and statistics must be generated. They are forwarded to the administrative departments, specialist departments and project managers. Also, the data is prepared for storing on microfilm for accountability purposes. The ten-day period data must be stored on tape for the monthly balance.

   Steps 1 to 5 are performed after each ten-day period. After the third ten-day period three tapes with ten-day period data are available. They must be merged for the further processing of monthly data.

6. Data for the month
   The result of the monthly data, the flexitime data for the preceding month, and the calculations of overtime must be combined into the balance for the month. Also, the data is prepared for storage on microfilm for accountability purposes.

7. Update on employee data
   Data of certain employees in the database must be updated if changes have taken place during the current month. For example, a changed name, another account number or bank, or an increase of salary may have to be processed.

8. Calculation of wages and salaries
   The monthly wages and salaries must be calculated on the basis of the actual monthly data and the updated employee data. Also, the payments must be triggered. The normal results are:

   - For each employee a salary slip and an individual record of monthly working hours are produced.
   - Tapes for the bank are produced with one record for each employee; the bank processes this tape and takes care of the transfer of money to the employees' accounts.
   - Tapes are also produced for social security authorities, insurance institutions and health insurance funds.

Payroll accounting is not the only task performed by administrative data processing. About 150 single batch programs from the various batch applications can be in use in a medium-sized organisation.

Practical Considerations

An organisation often has a department of its own which is responsible for administrative data processing. The members of this department deal mainly with operations like running the payroll accounting, analysis of new regulations or new technologies, programming as part of further development, and maintenance.

The main hardware components used in the example are a mainframe, some terminals in the administrative department, a document reader, check-in clocks and PCs for local support. Replacing the old-fashioned clock cards and check-in clocks by electronic badges would lead to modification of the accounting procedures. Further modification would be required if all departments were equipped with terminals connected to the mainframe.

Usually, major software components employed for administrative data processing are commercial products, for example the database system. However, there might be other components that have to be customised. This can be undertaken in house, by software houses or

in a combined effort.

As well as developing technologies there are rules, laws and regulations to be obeyed, which are also changing. Thus, a continuous maintenance effort is necessary for adjusting those software components that must take into account these regulations.

3.3.2 Boundary Conditions and Regulations

For a batch application like the accounting procedure, many different regulations govern the application's capability and operation. Examples are given in the following sections.

External Regulations

There are statutory regulations covering privacy, hours of employment, protection of youth, holidays with pay and so on. In addition to these regulations, pay agreements exist and there are special agreements dealing with certain occupational categories. Also, there are agreements concerning payments over and above the agreed scale.

Internal Regulations

Possible internal regulations influencing the salary calculations are agreement between the company's management and work council, flexitime agreement, travelling expenses guidelines, overtime agreement, shift regulations for certain occupational categories, and individual regulations for part-time employees.

Examples of common internal regulations concerning the operation of administrative data processing are:

- Access to administrative data has to take place from rooms with access control.
- The passwords used must not be simple and must be changed regularly.
- A modified four-eyes principle must be applied on certain updates. For example, salary increases must be updated and checked by different members of the staff.

From another point of view, these could be considered as measures.

3.3.3 Basic Security Policy Requirements

Integrity, availability and confidentiality of information are considered in greater detail with respect to batch applications in the following sections.

Integrity

Unauthorised modification of operational data (payroll data in the example) must not take place. Thus, the batch programs performing modification must be approved.

Availability

The application programs as well as the different data (operational data and log data) must be available when required. That presupposes the availability of the database system employed and the availability of the hardware involved (computer, the different storage media, communication lines, document reader and so on).

Confidentiality

In general, personal data is confidential. In many organisations, wages, salaries and awards are considered confidential. Confidential information, including the logging data on microfilm, may only be read and used by authorised office workers and must not be disclosed to outsiders. To that end it is necessary that the batch programs performing read access must be approved. Also, access privileges for individuals must be determined.

3.4 Software Development Environment

This scenario deals with the development and maintenance of software. This is usually undertaken by a team whose members have different responsibilities. Their cooperation as well as the development process are governed by certain rules.

A Software Development Environment (SDE) consists of hardware and software. It is used by software engineers, programmers, project managers and quality assurance staff. In the case of security-relevant software, evaluators may scrutinise whether the software fulfills the security requirements.

Software development is a procedure which is inherently supported by IT. Software development is governed by organisational rules. In particular, the cooperation of the team members is governed by certain rules such as life cycle methodologies and standards. Also statutory regulations or requirements from a customer must be taken into account. Finally, requirements concerning safety as well as security can be recognised.

3.4.1 Development Procedure

Software development consists of several steps, as illustrated in Figure 3-4.

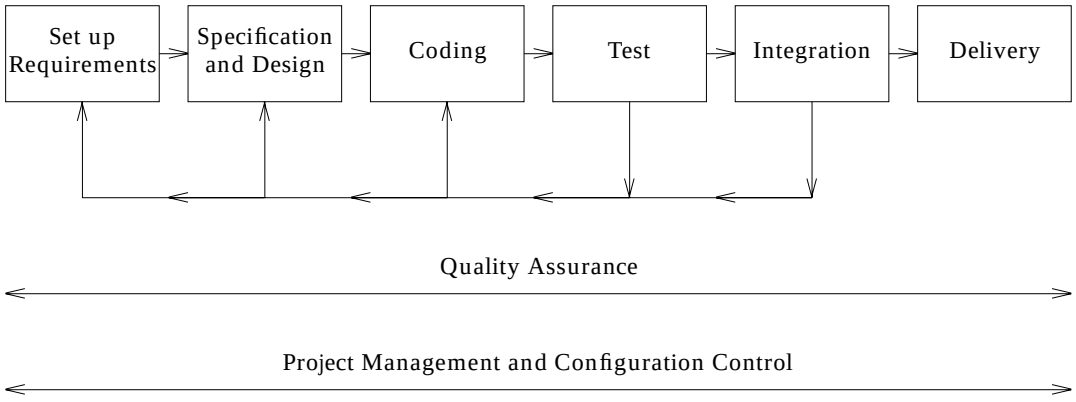

Figure 3-4 Steps During Development Procedure

Each step takes the output of the preceding step as its input. There is feedback at each step and there may be several iterations. Within a step several tasks must be performed. Some may be done in parallel whereas others may have to be performed in a certain sequence.

The software development process usually starts with the customer setting up a requirements specification for the software system to be developed. If the software is safety-critical there is special emphasis on certain safety-related activities. If a new product is planned for the software market, the requirements are determined by market analysis of the target group's needs.

The requirements specification is the basis for the functional specification and design of the system. The design specification is implemented in a suitable programming language. The source code is either interpreted or compiled to produce an executable code. Most designs consist of several modules, which are linked together. The executable module may be linked with other existing modules from a library and loaded for debugging and testing. After the successful testing of all the software system's modules, integration and integration testing are performed. In parallel with the development activities internal and external documentation is written, quality assurance and configuration control activities are carried out, and project management is performed.

Eventually, the finished software system is shipped to the customer or put on the market. Maintenance activities then take place. Also, decisions concerning further development must be made.

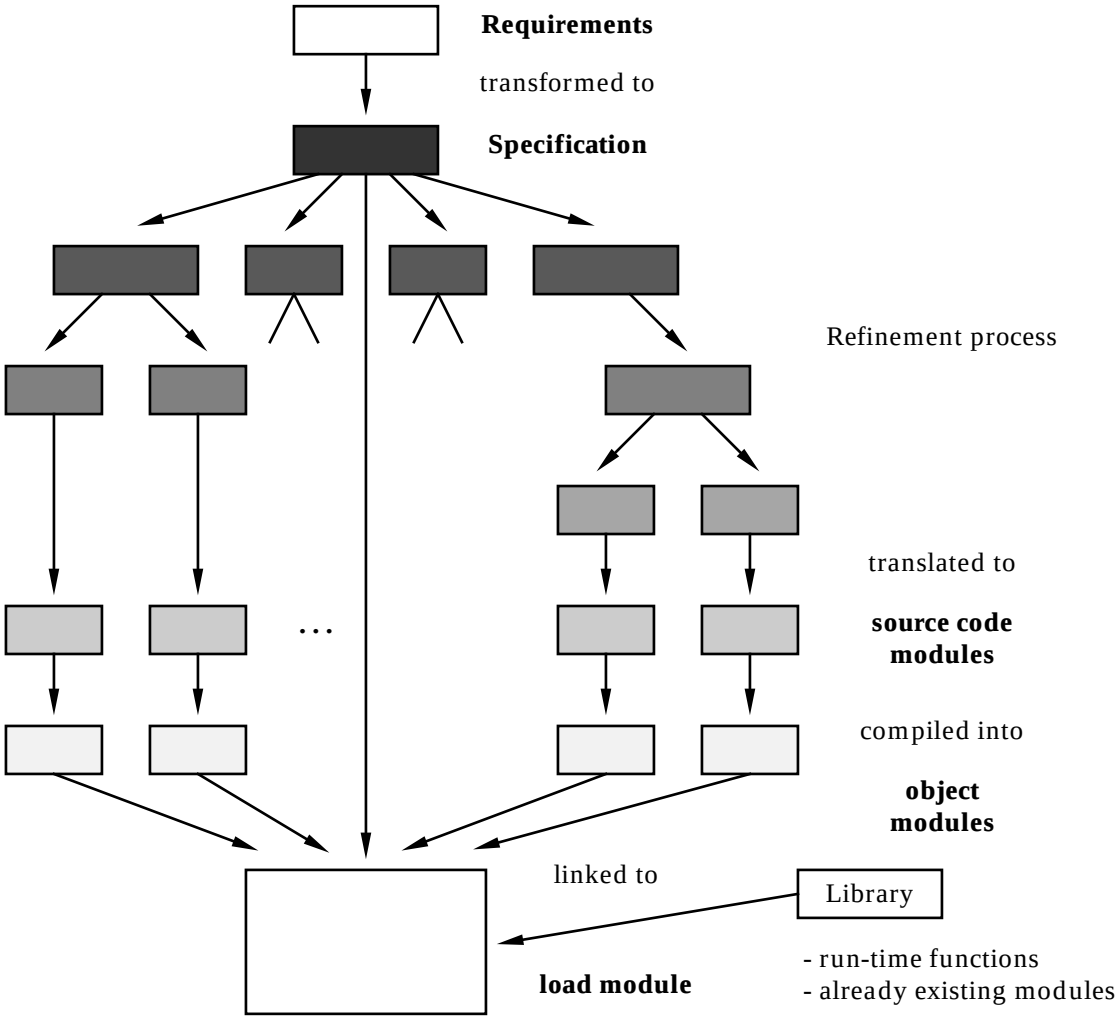

Figure 3-5 Data Handled during Software Development

Data Handled

The data handled in an SDE is mainly text files of different types such as requirement documents, system specifications, documentation, source code, makefiles, source code control system files and different administrative data. Data can also consist of files with binary code such as compiled units or load modules. Each data item (see Figure 3-5) is not independent from other data items but is linked by different relationships. These relationships are generated during the development process. Data is either components of some other data generated by a decomposition or refinement process, or is produced by applying certain transformation rules to get another representation of the same object.

Infrastructure

An SDE contains the following hardware components:

- computing resources, such as mainframes, workstations, personal computers, all located within one or more buildings
- devices for data storage, such as disks or tapes
- I/O devices such as terminals, printers and communication lines.

An SDE may consist of a single computing resource, which is either a multi-user system (such as a mainframe to which several terminals are connected allowing users to work in parallel) or a single-user system (such as a personal computer). Alternatively, an SDE may consist of several computing resources which are either stand-alone or which may be networked together. If they are connected by a network, the data may be stored on a single device or on several devices that are local to the computing resource. The computing resources may be distributed over different rooms or buildings in the company area and connected by a local area network, or located in geographically different sites, cities, countries or even continents. Data may be exchanged electronically via some form of public communication system or by mailing data storage media such as tapes or disks.

The use of an SDE may be either uncontrolled or controlled by different means. The SDE may be located in a restricted area where guards make sure that only people belonging to the company may enter the building. Within the building several other access control methods may be available to prevent unauthorised people from accessing the hardware, for example doors with special locks or number codes. For people who are allowed to work in an SDE, special measures are taken to prevent unauthorised access to hardware or software, for example passwords may be required.

Architecture

The SDE may be a centralised system or a distributed system connected by a network. Accordingly, the data stored may be distributed or located on a single device. Often backup media are used in case of system failure and for archiving data for long-term storage or for maintenance.

A data management system is necessary. This may be either a conventional file system of an operating system or a database. An interface builder may be necessary to build up a graphical interface. Other tools are also needed to support the development process. Some tools have to be available over the whole life cycle, for example, an editor, configuration and version management and documentation facilities. Other tools are only necessary to perform a certain task within the development procedure, such as the compiler, linker, debugger and loader. Libraries are necessary to store run-time system components, predefined or reusable software components (such as mathematical functions or sort algorithms), or to store the several independently-developed software modules (which are linked together during the integration phase to build the system required).

Organisation

Several software systems or products may be developed in parallel within an SDE. There may be any number of people involved in a software project dependent on its size.

The different tasks of the development procedure are performed by people in different roles. Each role has defined activities to be performed during the life cycle of the software. The operations that may be applied to specific objects are defined. The relationship between roles and people is not necessarily 1:1. The roles and the tasks associated with them are as follows:

- customer: setting up requirements that must be fulfilled by the software system to be developed
- project manager: monitoring the progress of the project, milestones, deadlines or similar tools; keeping track of the different activities; reporting to customer
- administrator: monitoring the budget; assigning people to roles
- system administrator: creating and deleting user accounts; hardware and software installation; backup and recovery activities
- auditor: specifying events which must be audited; analysis of audit trails; administration of audit system
- software engineer: specifying and designing the software system functions
- configuration controller: controlling versions of all components
- toolsmith: recognising common functions or basic utilities during the refinement procedure and realising them so they can be used by others
- programmer: coding the design; debugging of programs
- quality assurance controller: monitoring each activity of the development procedure; refining the design; monitoring state transitions during the development procedure; analysing for completeness and consistency in every state; integrating separately-developed modules; testing of implementation; formally verifying implementation against specification.

The assignment of roles to people may be handled differently by different companies. Nevertheless, separation of roles plays an important part in the software development procedure.

3.4.2 Boundary Conditions and Regulations

Certain regulations must be obeyed during the software development procedure. These may be either external or internal regulations. Examples are given in the following sections.

External Regulations

External regulations may be set up by the customer, by governmental guidelines or by law. Examples of external regulations are existing standards for the development procedure. These may be imposed by the customer on the development of the system. Special requirements may be set up if the system contains safety-critical software. There may be, for example, a requirement from the customer that the test environment delivered by him must be kept confidential. For instance, the data of the test scenario can be classified. Rules can exist that govern the mode of access a user with a certain clearance level has to the different forms of classified data.

Internal Regulations

Internal regulations are those set up by the company. They may specify the kind and sequence of activities to be performed and the roles for the different activities. There may be certain instructions and procedures for the delivery of software.

3.4.3 Basic Security Policy Requirements

The purpose of software development is the automation of a procedure or task by a piece of software. Examples are the substitution of files and records in folders by a database, the support of office work by office automation, controlling of production lines, controlling of power plants and navigation of aircraft.

Integrity, availability and confidentiality of information are considered in greater detail with respect to software development in the following sections.

Integrity

The most important problem is the loss of integrity of software. Loss of integrity may arise by malicious or accidental code modification. The same applies to test data, test results or documentation. Other integrity problems result from incorrect configuration by using the wrong versions from a library. Integrity can also be violated by loss of consistency. This can happen by incomplete modification due to human error or system failure. An incomplete refinement by decomposition procedure may also result in an inconsistent state. Loss of integrity can result in loss of availability (if the code is overwritten or destroyed) or in loss of confidentiality (if confidential information is corrupted).

Availability

Different factors may cause loss of availability of either hardware or software. The computer may break down, may be destroyed or may be stolen. The communication lines of a network may be cut. The storage media with important data may be stolen or destroyed. Even if the hardware is available, the information accessed using it may not be available. Thus, programs or data on the different storage media, including the archive, may no longer be available for reuse or maintenance.

Confidentiality

Requirements concerning the confidentiality of some information are important for nearly all software manufacturers. In particular, when a new product is being developed, some design aspects and the source code are handled in a restricted manner and must be protected against unauthorised access by competitors.

3.5 Control Systems

With the development of the microprocessor, control systems have evolved from simple electro-mechanical devices to sophisticated computerised systems. Control systems are now part of the IT environment. A control system supports such tasks as centralised control, monitoring and decision making for other computing entities. Examples of controlled entities are:

- network elements that switch or route voice, data or other communication traffic
- user workstations under a single administrative or management centre
- medical devices monitored during medical procedures.

Two common characteristics of control systems are that they often have real-time constraints and they are highly networked.

The operational aspects of control systems are characterised by the tasks the control system performs. For example, control systems may receive information from entities (*monitor*) then analyse the data for problems or respond to requests (*make decisions*) and finally, provide direction to the controlled entities (*command*). The particular tasks performed by individual control systems vary according to the entities under control and the sophistication of the control system. Control system security concerns are directly related to their sophistication because advanced control systems have greater decision making and command capabilities, making the controlled entities more vulnerable to security breaches.

The example control system scenario in this chapter is based on typical control systems for communication networks. The operation and performance of communication-related control systems are governed by rules. For example, who may operate the control system and what procedures must be followed in the event of system failure. These rules are contained in handbooks and written practices and are often implemented in the control system software. Control systems for entities utilised for public service or safety must implement and be run according to various statutory regulations. Finally, requirements concerning safety as well as security can be recognised.

3.5.1 Example System

Control systems are usually designed in a one-to-one-to-many hierarchy to maximise the capabilities of the system and the operator. One operator or technician usually staffs the control system (the one-to-one relationship) while the control system itself controls many entities (the one-to-many relationship). The procedures to be followed given events or requests from the controlled entities are encoded in the control system and understood by the operator, but at different levels of abstraction. Figure 3-6 on page 34 shows a typical control system.

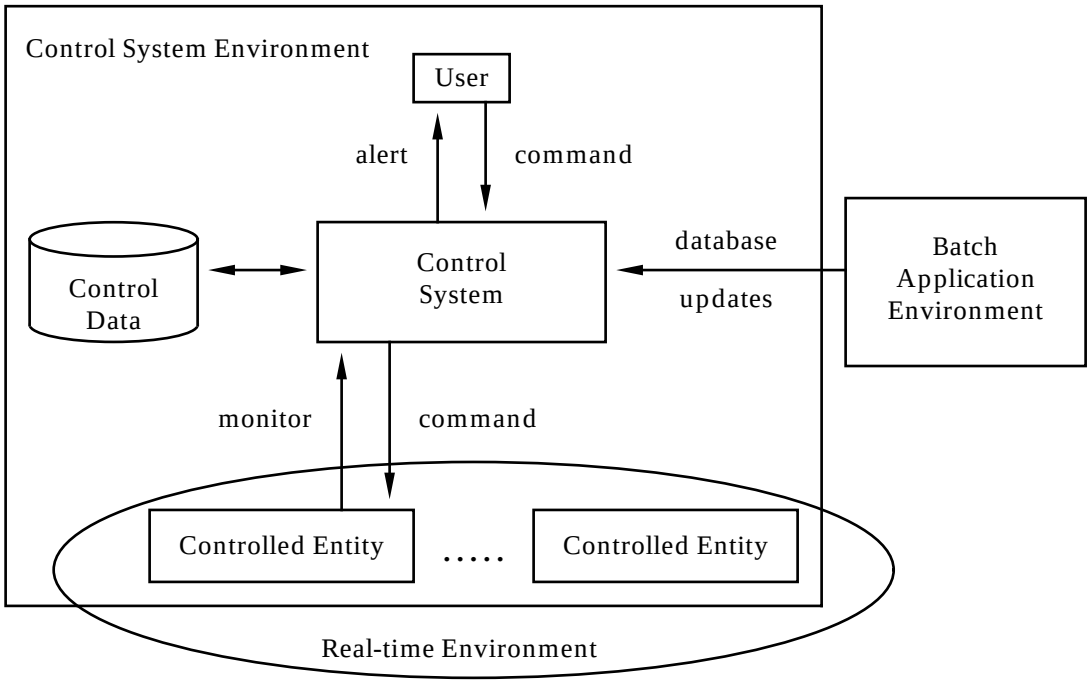

Figure 3-6 Typical Control System

Control activity in the control system is most often triggered by information or requests originating from the controlled entities. A typical event in a communications network is that one of the network elements, a router perhaps, sends status information about network flow to a network management control system. Many IT components are involved:

- the network element
- the data network facility connecting the network element to the control system
- control software running in the control system
- a supporting database
- an operator's console
- workstation and communication facilities to the control system
- perhaps one or more batch-oriented applications supporting the control system, for example, to download network configuration information.

Processing Network Flow Information

A typical course for processing network flow information from controlled entities is shown in Table 3-4 on page 35. The network flow information is transmitted to the control system from one or more controlled entities (step 1). The nature of the control system requires the control system automatically to process the information (step 2), (for example, to determine new routing tables) and potentially return control commands (such as updated routing) back to the controlled entities (step 3). Sometimes human response is necessary (for example, when network congestion is critical) so the control system interacts with the operator (step 4), who instructs the control system on how to resolve the conflict (for example, restrict traffic from non-critical hosts) (step 5). The control system takes the operator's instructions and translates them into specific

commands for the controlled network entities (step 6).

Step	Controlled Entity	Control System	Operator
1	X		
2		X	
3	X		
4			X
5		X	
6	X		

Table 3-4 Overview of Flow Information

A clear distinction can be made between routine work on the information from a controlled entity and the transfer of the information through the systems (the flow of work). Parallelism can occur if the control system analyses data in parallel with an operator.

The following subtasks (steps) can be identified:

1. Status data transmission
   The controlled entity transmits status and alarm information to the control system on demand or periodically. Critical information is transmitted immediately, while general operating data is usually accumulated and transmitted less frequently. Sometimes the most critical alarms, for example, *system failing*, can only be transmitted once (and then the entity goes down), requiring fault-free communication and reception.
2. Data analysis
   The control system uses data from the controlled entities to reach control decisions. Typical decision software is made up of expert systems combined with static analysis. Decision making is frequently done without operator intervention. Operators are appraised of decision results on a direct or summary basis, but usually after commands are applied.
3. Automatic command execution
   The control system determines what action must be taken and sends commands to its subordinate entities to implement the decision. The control system often has privileged access to the entities.
4. Operator intervention or request
   Control systems may request operator intervention when unusual situations are detected or when drastic commands (possibly those that affect service) must be sent to the controlled entity and confirmation is requested. The operator may also request the control system to effect some change to the entities that was not a result of information sent to the control system.
5. Control system processing of operator command
   Processing operator requests results in either a one-to-one or a one-to-many situation. The operator command may affect one or several controlled entities. The control system interprets the command, determines the number and selection of affected entities, encodes the request in an entity-specific form and transmits.

Modern Systems

As controlled entities become more complex, and service requirements dictate that faults be handled in a shorter time frame, intervention by an operator becomes less desirable. Modern communication control systems handle almost every situation presented by the controlled entities. Because of the real-time considerations, deviations and exceptions require special case handlers to be built into the control system itself and not to be part of the operator procedures. For example, network control systems are often used to reconfigure the network in real time based upon status information or alerts from network entities, using its supporting database to maintain current and planning views of the network. Inability to communicate with a network entity requires the control system to command other entities to reconfigure while the control system works to restore communications.

Loss of communication with the operator or supporting batch applications may not have immediate serious consequences, unless drastic commands that must be sent to controlled entities need operator confirmation. However, the control system can often continue to handle problems, reporting them to the operator when communication is reestablished.

3.5.2 Boundary Conditions and Regulations

Control systems are often implemented as a mechanism to improve adherence to boundary conditions and regulations. For example, complex telecommunication networks cannot be maintained without the support of automated decision making and control. Use and responsibility of control systems are governed by different regulations that are either set up by the government or internally by the company or organisation itself. The following sections give examples of regulations that might affect control systems.

External Regulations

Examples of statutory regulations are:

- privacy acts protecting voice or data transmission over networks
- recommendations concerning standards for reliability and quality of entity or service performance
- legal regulations on time spans within which service affecting events may impact the community

Internal Regulations

Examples of internal regulations are:

- instructions on handling control system requests
- rules on managing confidential material
- guidelines on control system security.

3.5.3 Basic Security Policy Requirements

Integrity, availability and confidentiality of information and function are basic security requirements for the control system environment. However, the precedence of these basic requirements varies according to circumstance. The prevailing interest with respect to control systems is service continuity of the controlled entities. In addition, operation of the control system must be in accordance with the organisation's security policy. Because the control system makes control decisions which may affect security, the entity security policy must be encoded into the decision software.

Integrity

Maintenance of the integrity of the control system's databases, decision software and communication mechanisms are important to ensure reliable operation. Requirements include:

- Access to the control system is restricted to prevent unauthorised access to the controlled entities.
- Strong authentication between the control system, controlled entities and support systems is necessary to prevent intruders from masquerading as the control system to controlled entities.
- Encryption and timestamping between entities is necessary to prevent replay of sensitive commands.
- Integrity of the software and operational databases must be maintained to ensure correct decisions are made and proper commands transmitted.
- Accountability may be problematic since control system decisions are often difficult to trace back to a single person. Thus, changes to the database or decision rules need to be logged as a way to determine accountability.

Availability

The availability of the services provided by the controlled entities (but not the control system) usually overrides other concerns. Indeed, the control system itself is deployed to help ensure service availability. Many requirements are established solely to maintain availability:

- Reliable protocols and networks are used to guarantee message delivery.
- Strong software engineering techniques are used to reduce software faults which may cause incorrect decisions to be made or incorrect commands to be transmitted.
- Fault-tolerant hardware and software are used.

Confidentiality

Most control systems do not require broad measures of confidentiality to maintain operational or control security. However, particular aspects of the control system may require very strong confidentiality controls. Thus, requirements analysis must be performed carefully to identify information confidentiality needs of data. Specific requirements are:

- Information used by the control system to authenticate itself to controlled entities must be protected.
- Information used to make control decisions based on client-specific needs must be protected. For example, a company that provides communication services to various clients may need to make control decisions based on client-specific information. This data must be protected to prevent competitors from gaining access.

- If the control system monitors security events, alarm thresholds must be secret.
- Encryption keys must be protected.

Chapter 4

Specific Threats

This chapter deals with threats against the integrity of information and programs, against the availability of data, programs and resources, and against the confidentiality of information. The threats are considered for each of the scenarios presented in Chapter 3.

Not all threats relevant for a particular application and in a special environment are taken into account. The importance of threats is not assessed, as this is only possible with the complete knowledge of the application and the environment (see Chapter 2). Rather it is the intention to focus on those threats that are typical and relevant for the scenarios. The first section deals with threats that, due to their generality, can occur in all scenarios.

A threat may lead to *consequences* such as dissatisfied customers, loss of money or wasted work. These consequences must be evaluated in order to recognise the importance of countering the specific threats. Threats can have different *causes*. For example, the threat of *loss of data* can be caused by inadvertent removal of a medium or by a hardware malfunction. Thus, knowledge about the causes of threats is a prerequisite for choosing and employing suitable countermeasures. The three basic threats are:

- *loss of availability*
- *loss of integrity*
- *loss of confidentiality*

This sequence approximately reflects the priority in the commercial world. The basic threats might be used to group threats according to their effect. Alternatively, they can be used to outline security problems that might arise in the use of an IT system. Authenticity and anonymity in the communication context, and obligation in a legal area might also be worth considering to determine all relevant threats. Examples of events and circumstances that might cause harm to the IT system are as follows:

software bug	hardware failure or breakdown	fire
trojan horse	breakdown of air conditioning	program manipulation
incorrect operation	illegal copies	hacker
worm	unauthorised access (logical and physical)	electromagnetic radiation
virus	missing security awareness	wire tapping
earthquake	acts of sabotage	power failure
demotivated staff	erroneous data input	terrorist attack
static charging	use of private hardware	theft of media
	inappropriate disposal of printouts	

4.1 Common Threats

This section represents typical threats that are relevant to all the scenarios. The lists in each group are examples and do not preclude other threats. In general they are countered by safeguards in the physical and organisational area (see Chapter 5).

The harm that could be caused by a natural disaster should be considered. Examples of such disasters are:

- thunderstorm
- hurricane
- flash of lightning
- earthquake
- explosion
- fire
- flood.

Potential damage might be restricted by infrastructural or physical measures such as choice of site, construction of building and rooms, fire protection, automatic control and detection. All kinds of disaster provisions help to limit the damage of harmful events that cannot be prevented.

Examples for another class of common threats are:

- failure of IT components
- disruption of power supply
- failure of air conditioning.

The consequence is generally the loss of availability, which means a partial or complete disruption of services. This may cause severe damage if the organisation is totally dependent on the functioning of the IT system. When there is no alternative to the use of the IT system, provisions must be made to ensure that the service is available again within a certain amount of time. Maintenance as suggested by the supplier, some redundancy in the power supply, automatic control and warning of air conditioning failure are possible safeguards to help prevent this type of event.

The following threats may affect the use of the IT system:

- incorrect operating
- theft of parts of the equipment
- vandalism against the IT system.

Safeguards such as restricted access to the equipment, and education and motivation of the staff should be considered to prevent this type of event.

Nowadays many IT systems are connected to a local area network or a wide area network. In the future, networking will be the rule. In many cases the network is and will be heterogeneous: different types of IT systems from different vendors are connected and interoperate. In these cases it is even more important to prevent unauthorised access by measures such as identification and authentication and access control (see Chapter 5).

Other crucial problems are:

- loss of integrity and confidentiality of the exchanged data
- incorrect routing
- uncertainty of the identity of sender and receiver.

In this context measures like encryption, message authentication code, digital signature, along with reasonable key management, are of interest. The availability of network services should be

in accordance with the requirements of the users and the application. Technical measures concerning network and communication security are not supplied in this guide (see Chapter 6).

The following are possible problems of data integrity, availability and confidentiality:

- inadvertent misuse
- incorrect operation.

These are countered by training, motivation and control of the people using the IT system. A fundamental safeguard is the *four-eyes principle*. To a certain extent, control can be effected by plausibility checks in the software. Also in this context, the human-machine interface is of considerable relevance: the easier it is to use, the less likely are errors to occur.

Incorrect handling of removable storage media can affect the integrity and confidentiality of its data. Normally this can only be prevented by a combination of measures in software and hardware, and organisational measures in the environment of the IT system.

In special cases where confidentiality of data is very important, electromagnetic radiation emanating from equipment such as terminals and cables can be a security problem. This is known as *compromising emanation*; it can be controlled by special measures known as *tempest* measures.

4.2 Office Systems

In case handling, different kinds of threats are possible, resulting in the following damage:

- delay of office work through denial of service
- loss of information through unauthorised access
- repudiation of informaton sources
- unauthorised modification of data and essential utilities
- disclosure of confidential information.

Threats of these kinds are considered in more detail in the following sections.

4.2.1 Delay of Office Work

Delay of case handling through denial of service can lead to different consequences. For many cases deadlines exist. If these are not met the result can be one of the following:

- loss of money if an offer with a discount is not answered in time
- loss of market position if reactions are not fast enough
- wasted work if a tender is not finished in time
- a dissatisfied customer if promised papers are not delivered in time, with the further consequence that the company's reputation suffers
- a dissatisfied member of the public if a response to a request takes too long, with the further consequence that the organisation's image suffers
- change of legal conditions if legal dates are not obeyed.

4.2.2 Loss of Information

If documents of a complete case or a case itself get lost there is an infinite delay. Many consequences are similar to a normal delay:

- loss of money if an offer with a discount is not answered
- wasted work
- dissatisfied customer if promised papers are not delivered, with the further consequence that the company's reputation suffers
- dissatisfied member of the public if a request is not serviced, with the further consequence that the organisation's image suffers
- change of legal conditions if legal dates are not obeyed.

A further consequence is that additional work is necessary to restart the missing case or to rewrite the lost documents. Care is needed to ensure that activities such as incrementing values in a database or booking flights are not performed a second time.

There are many different reasons for loss of information. One possibility is technical problems, such as different breakdowns. Another possibility is that either accidentally or deliberately an office worker:

- removes a document
- cancels a case
- assigns an incorrect record number to a document.

4.2.3 Problems with Flow of Work

Such problems can occur when a case is forwarded to an office worker who claims not to be responsible. Another example is a case that is mislaid. In all cases time and effort has to be spent to clarify the situation. Another consequence is an unnecessary delay (see Section 4.2.1 on page 42).

There are different reasons for problems with the work flow. One possibility is technical problems, such as different breakdowns. Other possibilities are as follows:

- unknown changes of responsibilities; this can cause a case to be forwarded to an office worker who should not be involved
- changes of the organisational structure; these can influence the routing of case handling
- the application of organisational procedures that are incomplete or erroneous
- loss of information on the location of a case.

4.2.4 Repudiation

An office worker may claim not to have been involved in a certain case. If the work in question is an approval or an important decision, it may take some time to clarify the situation. There are two explanations for the office worker's claim: it may be true or he cannot remember the case. If he has not been involved, someone else has worked on the document, possibly forging a signature.

4.2.5 Unauthorised Modifications of Software and Data

Unauthorised modifications of software and data may be difficult to detect, yet have catastrophic consequences. As a result, tools, utilities or applications that once ran correctly, now disrupt user activity, produce incorrect results, or even corrupt or destroy existing documents, databases or backup media.

Users who are unaware of the use of unauthorised software (such as public domain software) or who ignore corporate policies against its use, may introduce viruses into the system or experience undesired side effects.

A faulty software development process may allow programmers to insert unreviewed changes (which may be malicious or destructive) into source code.

Inappropriate access controls, may allow documents to be altered. For example, a refusal may be changed to an approval, resulting in financial loss, employee confusion and dissatisfaction, or customer discontent. Similarly, a change to routing information may result in misrouting and delay.

Data entry applications that allow authorised users inadvertently to enter incorrect data into forms, may introduce errors into databases.

Incorrect results, altered documents or corrupted databases may lead to incorrect decisions. For example, if a calculator or spreadsheet program has been altered to produce incorrect results, the user may be misled into making the wrong business decision. That wrong decision may result in missed market opportunities, inventory shortages or excesses, financial losses and so on.

Even if the modifications are detected early, efforts to locate and fix the problem are likely to be time-consuming and disruptive.

4.2.6 Disclosure of Confidential Information

The proportion of confidential material handled in an office organisation depends on the type of office organisation. In all organisations personal data is considered confidential. Also, trade secrets, contracts, company strategies, information on customers or partners are usually considered confidential.

An office worker to whom confidential information is disclosed may be not aware of its sensitive nature. The potential damage depends on how the office worker exploits the information.

Causes for the disclosure of confidential information can be:

- unknown changes of responsibilities; this can cause a case to be forwarded to an office worker who is not authorised
- the application of organisational procedures that are incomplete or erroneous
- changes of the organisational structure that can influence the routing of case handling
- allowing an unauthorised person to look at the work of an authorised office worker
- copying confidential data into a file accessible by unauthorised users, for example by cut and paste operations within a window system
- granting unnecessary access rights because of an insufficient granularity.

4.3 Database Applications

Different threats against operational data, its integrity, availability and confidentiality are possible as follows:

- delay of transactions
- loss of information
- incorrect results
- unauthorised modification of data
- disclosure of confidential information.

In the following sections, examples of these threats are considered in more detail.

4.3.1 Delay of Transactions

Due to methods for stating the value, no principal problems need be expected with possible delays in bookkeepinging or bank transfers. However, withdrawing cash may be difficult if a preceding bank transfer has not been committed in time.

The main consequence of unavailable services is dissatisfaction of customers. Possible reasons for unavailable services are as follows:

- breakdown of a terminal at the branch
- breakdown of the communication line between the branch and the central office
- breakdown of the main computer in the central office
- data in the database cannot be accessed because of missing permissions
- transaction programs are not available
- transaction programs do not proceed because of deadlock
- transaction programs not proceed because other programs use up the resources.

4.3.2 Loss of Information

Loss of information occurs if a program or data in the database is not available. The main consequence of this loss of information is that extraordinary effort has to be spent for recovery or rework. This contributes to avoidable delay (see Section 4.3.1).

There are many different reasons for loss of information. In addition to technical problems (such as disk failures) losses can occur because:

- a developer (accidentally or deliberately) removes a data or program file
- storage media (disks, tapes or optical discs) are stolen.

4.3.3 Incorrect Results

Results may be incorrect because of incorrect input data or erroneous programs. The main consequences are:

- customers are displeased if their accounts are incorrect
- wrong business decisions of any kind can be made.

Input data can originate from the database or can be entered incorrectly by a cashier at the terminal. Data kept in the database can become incorrect because of modification by unauthorised persons or programs (see Section 4.3.4 on page 46).

4.3.4 Unauthorised Modification of Data

The consequences of unauthorised modification vary as shown by the following examples:

- Modification of customers' account data may lead to disastrous effects. In particular, the bank's reputation may seriously suffer.
- Modification of administrative data, such as salary or personnel data, may cause serious problems for the bank's administrative data processing.
- Modification of programs may paralyse data processing in general.
- Access to log data may make it useless for accountability purposes. In particular, it can make it impossible to update the main database during the night.
- Access to program files, for example removing them, can make the processing of the different transactions impossible.

It may take some time to detect that data has been modified. The more time it takes the more serious are the consequences and the more difficult it is to find the cause. In addition, much effort may be required to establish what has happened and to undo or compensate unauthorised modification.

Unauthorised modification of data can, for example, be performed by:

- new applications under development
- a virus which somehow has been implanted
- the accidental or deliberate invocation of programs doing another task
- programs of malicious developers (such as a trojan horse)
- data manipulation statements interactively performed on parts of the database.

The execution of these programs or statements written in data manipulation language can be enabled because access rights are of an insufficient granularity.

4.3.5 Disclosure of Confidential Information

Data on customers (their accounts, transfers and so on), the log data (containing similar data for accounting purposes) and personal data of the bank's employees are confidential.

The consequence of a disclosure of confidential data depends on its kind, to whom it is disclosed and what is done with the knowledge gained in this way.

The disclosure of customer data is a violation of the banker duty of secrecy. Customers may become dissatisfied and even go to another bank. If a disclosure of personal data is detected, the employees concerned are likely to complain.

Causes for the disclosure of confidential information can be:

- Access rights of inappropriate granularity have been granted to clerks.
- Unauthorised persons were permitted to look at the results of query programs.
- A clerk may derive confidential information from non-confidential results of statistical queries or from access to the log file.
- The communication lines between branches and the central office may be tapped.
- A malicious developer may have realised programs that look up the confidential data. Alternatively, the developer may have produced a program in such a way that it stores interesting data in a file accessible to unauthorised persons. The existence of such programs can, in turn, be caused by missing regulations governing the installation of programs or by not obeying such regulations.

- A developer may have secretly copied interesting data onto a tape archive.
- Storage media, like printouts, defective tapes, or hard disks, are not correctly destroyed.
- Output is directed to a device that is accessible to unauthorised persons.

4.4 Batch Applications

A characteristic of the monthly accounting example is that most of the operational data involved is personal data and hence is confidential. Different threats against this data, its integrity, availability and confidentiality are possible as follows:

- delay
- loss of information
- incorrect results
- unauthorised modifications of data
- disclosure of confidential information.

In the following sections, examples of these threats are considered in more detail.

4.4.1 Delay

There are deadlines for the different steps in the administrative application. Unexpected delays can cause these deadlines to be missed with the following consequences:

- employees are dissatisfied with belated payments
- project managers may have to base decisions on data that is not up-to-date
- customers are dissatisfied if data on a project's progress is not on time.

In addition to technical problems (such as different types of breakdowns or failures) delays can be caused by:

- loss of information (see Section 4.4.2) such as missing input data or programs
- concurrent programs that use up the system's resources, for example, a program under development and still misbehaving
- data is not accessible because its current location is not known or because of inappropriate permissions.

4.4.2 Loss of Information

Loss of information occurs if a program or data files are not available when requested. In this scenario, the disks containing the clock card data, the microfilms containing the accountability data or the tapes with the payment records for the bank may be not available.

The main consequence of these losses of information is that extraordinary effort has to be spent for recovery or rework. This contributes to avoidable delays (see Section 4.4.1).

There are many different reasons for loss of information. In addition to technical problems (such as disk failures) losses can occur because:

- somebody (accidentally or deliberately) removes a data or program file
- storage media, for example disks, tapes or optical discs, are stolen.

4.4.3 Incorrect Results

Results can become incorrect because of incorrect input data or erroneous programs. The main consequences are:

- employees are dissatisfied with incorrect payments
- project managers may make incorrect decisions based on erroneous project data
- customers are displeased if data on a project's progress is incorrect.

Input data originates from the database, from temporary files, from a pipe, or (in the example considered) from a disk used as intermediate storage (into which the document reader writes clock card data). Data kept in the database can be modified by unauthorised persons or programs (see Section 4.4.4). Data in files or data transferred by means of a pipe can become incorrect if the program producing it is erroneous. Data on disks can be corrupted by a magnet. In addition, disks can be changed by mistake.

The existence of erroneous programs can be caused by missing regulations governing the installation of programs or by not obeying such regulations. Probably, an erroneous program does not misbehave frequently; if it did it would be easily identified. It is more likely that it produces incorrect results under rare circumstances only.

4.4.4 Unauthorised Modification of Data

The consequences of unauthorised modification are illustrated by the following examples:

- Access to the tape with the payment records for the employees can make its further processing impossible or can modify administrative data. The goal could be to increase certain payments.
- Access to the disks with the data on the hours worked can make its further processing impossible or could modify project data. The goal could be to increase certain calculations of overtime.
- Access to data or program files, for example removing them, can make the processing of the application impossible.

It may take some time to detect that data has been modified. The more time it takes the more serious are the consequences and the more difficult it is to find the cause. In addition, much effort may be required to establish what has happened and to undo or compensate unauthorised modification.

Unauthorised modification of data can be performed by:

- new programs under development accessing real data
- the accidental or deliberate invocation of programs doing another task
- programs of a malicious developer.

The execution of these programs can be enabled by unnecessary access rights due to insufficient granularity.

4.4.5 Disclosure of Confidential Information

The consequences of a disclosure of confidential information depend on to whom it is disclosed and what is done with the knowledge gained in this way, for example:

- In general, if a disclosure of personal data is detected, the employees concerned are annoyed.
- Conclusions about the performance, efficiency and frequency of sickness of employees can be drawn and exploited in the handling of internal applications.

Causes for the disclosure of confidential information can be:

- Unnecessary access rights are granted because of an improper granularity.
- Unauthorised persons, for example visitors of the administrative department, are allowed to look at the results of batch programs.
- A malicious developer may have produced programs that look up the confidential data.
- The developer may have produced a program in such a way that it stores interesting data in a file.
- A dissatisfied developer may secretly have copied interesting data onto an archive.

4.5 Software Development Environment

Different threats against the data produced in an SDE, its integrity, availability and confidentiality are possible as follows:

- delay of development
- loss of information
- unauthorised modification of data
- disclosure of confidential information.

Usually, a software product is further developed and has to be maintained. However, error reports can be sent in from the locations using the software product (on possibly different platforms). These error reports must be processed. This leads to a further threat typical for an SDE: delay in handling error reports.

In the following subsections, examples of these threats are considered in more detail.

4.5.1 Delay of Development

Normally, the development of software is scheduled so that it meets deadlines. An unexpected delay is likely to cause the deadlines to be missed. The main consequences are:

- customers' dissatisfaction (the company's reputation suffers)
- loss of money
- delay in launching other projects.

In addition to inadequate control of the development process and technical problems (such as different types of breakdowns or failures), a delay can occur because of:

- unavailability of development tools; this may be caused by inappropriate permissions or incorrect file links
- misbehaving tools that are erroneous; the existence of such programs can, in turn, be caused by missing regulations governing the installation of programs or by not obeying such regulations
- developers making mistakes; this can be provoked by inadequate user interfaces in the support programs
- support program showing undesired side effects; the existence of such programs can be caused by missing regulations governing the installation of programs or by not obeying such regulations
- lack of understanding of sophisticated tools such as make, source code control system, debugger, interface builders or requirements tools.

4.5.2 Loss of Information

Loss of information occurs if documents or source code files are deleted or cannot be found. If no recent backups are available this is likely to become a serious problem.

The normal consequence is that extraordinary effort has to be spent for recovery. This contributes to avoidable delays (see Section 4.5.1).

There are many different reasons for loss of information. In addition to technical problems (such as disk failures) losses can occur because:

- a developer (accidentally or deliberately) removes a document
- a careless developer does not obey organisational rules for storing
- no effective back-up procedures exist.

4.5.3 Unauthorised Modification of Data

The unauthorised change of program sources can be particularly disastrous because a malicious attacker can do almost everything, including the installation of programs such as viruses and trojan horses. The consequences of unauthorised modification are illustrated in the following examples:

- Modifying text, for example erasing a ‘‘not’’ in a requirements paper, specification or error report, reverses its meaning.
- Changing dependencies in a makefile may cause libraries and the final programs to go out of date.
- Changing environment variables may mislead the *make* utility or the execution of programs.
- Incorrect setting of the clock may mislead utilities that depend on timesettings of various files, for example the utility *make*.
- Modification of entries in initialisation scripts may lead to unwanted capabilities.

It may take some time to detect that data has been modified. The more time it takes the more serious are the consequences and the more difficult it is to find the cause. In addition, much effort may be required to establish what has happened and to undo or compensate unauthorised modification.

Unauthorised modification of data is made possible by:

- the application of organisational procedures that are incomplete or erroneous
- allowing an unauthorised person to take over the work of an authorised developer
- a virus which somehow has been implanted
- granting of unnecessary access rights because of an insufficient granularity
- testing on operational master data.

4.5.4 Disclosure of Confidential Information

Information kept in an SDE may be confidential because of a customer requirement. In many cases, source code as well as development documentation is considered confidential. Sometimes the company has decided that the software developed should be put on the market, and that the competitors should not know of these plans.

The consequences of disclosures of this confidential information are:

- A disclosure usually angers customers, with the further consequence that the company’s reputation suffers. Further damage can occur if secret test data is disclosed.
- The competitive advantage may vanish if sources, conceptual papers or other kinds of documentation are disclosed to competitors. Ultimately the company may become bankrupt.

Causes for the disclosure of confidential information can be:

- forwarding documentation to an employee who should not be involved
- granting of unnecessary access rights because of an improper granularity
- allowing unauthorised persons to look at the work of developers
- copying confidential data into a file accessible by unauthorised users, for example by cut and paste operations within a window system
- a dissatisfied employee secretly copying interesting sources and documents onto some movable storage medium

- incorrectly destroying storage media like printouts, defective tapes or hard disks
- directing output to a device that is accessible to unauthorised persons.

4.5.5 Delay in Handling Error Reports

Error reports subitted by customers should be investigated as soon as possible. It should be verified that there really is an error and, if so, the error should be corrected in the different versions of the software product. The customer should be sent a response and eventually a bug fix.

A delay can make customers dissatisfied with the consequence that the company's reputation suffers. Furthermore, delays can hinder an efficient recognition of related errors and, thus, cost money. Handling error reports may be considered as a special form of case handling.

There are different reasons for delay in handling error reports. As well as technical problems (such as different types of breakdown or failures) a delay can be caused by:

- unknown changes of responsibilities that cause an error report to be forwarded to a developer who is not in charge
- applied organisational procedures for dealing with error reports that are incomplete or erroneous
- changes of the organisational structure that can influence the routing of ongoing error reports
- incorrect information on responsibilities for different versions or target systems
- loss of information on an error report's location
- forwarding an error report to an employee who is on leave instead of his substitute.

4.6 Control Systems

This scenario represents applications that provide command and control support to other complex computing entities. Command and control scenarios exist in almost every computing environment where human control is either too costly, or more importantly, too slow to provide proper and timely control decisions. The principal characteristic is the reliance on the control system to maintain availability of some service. Threats to availability are important, however additional threats to integrity and confidentiality must also be considered. Threats of varying kinds are possible, which almost always lead to loss of service:

- delay of control actions
- incorrect or unauthorised control decisions
- disclosure of confidential control information
- repudiation of control commands.

These threats are considered in more detail in the following sections.

4.6.1 Delay of Control Actions

Delay, in this case, can mean delay in making, transmitting or receiving control actions. Where real-time constraints are paramount, and dependent on the control environment, any delay can potentially result in loss of service or worse (for example, in a medical system). Other consequences are:

- issuing or implementation of control commands after the actual event is over
- inability to counter a *domino effect* where more and more controlled entities fail due to delay in conveyance of commands
- extremely long periods when the system is not available as the control environment is restored, entity by entity
- loss of customer confidence.

Delays can be caused by:

- errors in software design
- performance problems in the network connecting control systems to controlled entities
- unauthorised use of the control system or network inhibits processing or transmission of control requests
- improper operator response procedures
- faulty security mechanisms preventing access to controlled entities.

4.6.2 Incorrect or Unauthorised Control Decisions

Reliability and smooth operation of the entire control system environment depend upon correct decisions being made for each control scenario. Improper decisions may be accidental or deliberate. For example, unauthorised control decisions could permit criminals to have commands issued by network control systems to cover tracks of illegal use or reroute traffic through devices capable of eavesdropping.

In addition to those related to delay, the consequences of incorrect or unauthorised control decisions are:

- loss of service or revenue
- misdirection of service (servicing by the wrong provider)
- extra work to determine and repair the root cause of the erroneous control decision.

Causes of incorrect or unauthorised control decisions are:

- accidental or deliberate modification of control rules, data or transmission of control commands due to poor security
- improper or poorly-defined operator procedures
- incorrect thresholds for responding to events
- software that incorrectly models responses to control scenarios.

4.6.3 Disclosure of Confidential Control Information

Control systems may be designed so that particular decisions are based on confidential parameters. For example, a public network offering may utilise customer-set parameters to route traffic or bill for services, or may use patented algorithms to make decisions. Competitors who could access this information may be able to learn enough to steal business or, perhaps, intrude on the customer environment.

The direct consequence of disclosure of confidential control information is loss to the service provider or the customer. Disclosure may be performed by:

- eavesdropping on data sent from the controlled entities or commands from the control system
- inference analysis of the causal effect of various responses by the control system as a result of data sent by the entities
- unauthorised access to control system databases, or upstream batch applications
- dissatisfied employees
- perusal of improperly discarded printouts, magnetic media and so on.

4.6.4 Repudiation of Control Commands

Customers of services managed by control systems may try to place the blame on the owners of control systems for losses due to service failure. When legal action is threatened, it may be up to the owners of the control system to show that certain actions were indeed taken to ensure service continuity. This is especially true when control systems directly control entities owned or leased by a customer, for example in outsourcing situations. Repudiation of claims that commands or status information were sent will invariably lead to protracted and expensive legal actions. The causes for repudiation may be:

- lack of required information, for example, timestamps in audit records or loss of audit records
- inability to correlate commands issued by the control system to data sent by the controlled entities
- failure of the operator to recall his response to particular situations
- failure of the command protocol to ensure commit requests and responses are provided and recorded
- issuing commands by unauthorised entities masquerading as the control system; issuing status information by devices masquerading as controlled entities
- operators or intruders directly accessing controlled entities and invoking commands or setting parameters.

Chapter 5

Technical Measures

This chapter describes measures that are dependent on the corporate security policy. Special features of measures realised in hardware and software are discussed in terms of capability and assurance. Guidelines for the choice of secure products or systems are presented.

5.1 Measures in General

Measures must be chosen to counter relevant threats for an IT application or an IT system. The choice of appropriate safeguards depends on the system, on the application and on the chosen corporate security policy (see Chapter 2). Measures may be taken in several areas such as infrastructure, organisation, personnel and IT. Communication technology is considered to be part of IT. Special kinds of measures are tempest equipment, disaster provisions and IT insurances. Infrastructural or physical measures deal with the composition of the environment, including buildings, floors, rooms, walls, windows, doors and boxes. Organisational measures are all regulations concerning procedures, tasks and responsibilities for IT security. The fundamental organisational measure is that there is at least one person assigned to the security of each system or application. Examples of personnel measures are information, education, motivation and control. The measures realised in hardware and software are considered separately below.

The choice of measures depends on the kind of threats to be countered and on the approach the organisation has chosen for dealing with them as formulated in their security policy. Usually you must look at the causes of a threat to select appropriate measures. Often a threat has many causes. Therefore several measures must be introduced to counter that threat completely. However, one measure may affect several threats. The general principle for the composition of measures is that a chain is only as strong as its weakest link. Therefore measures have to be combined so that the effect is universal and balanced. Special attention must be paid to the possible interference of different measures and to measures whose effects are based on the effects of other measures (*interdependency*).

The choice of measures must be based on the corporate security policy and the operational requirements, both of which must be in line with corporate goals and tasks.

Besides the aspect of being fit for purpose, there is the aspect of the effectiveness of a countermeasure. This reflects how much a measure reduces a risk associated with a threat. The effectiveness of personnel and organisational measures depends to a great extent on how willing and capable people are of accepting regulations. The effectiveness of physical measures is roughly determined by the condition and the quality of the material applied.

In a commercial environment, the efficiency of measures alone and in combination has to be ensured. This means that a reasonable relationship between the cost of a measure and the protection it supplies must be achieved.

5.2 Security Policy Realised in Hardware and Software

When procuring IT systems, the security policy realised in hardware and software is of special importance. In this context generic headings, functionality classes and assurance levels are the fundamental keywords.

The technical measures realised in hardware and software may be grouped under the generic headings of the ITSEC specification:

- identification and authentication
- access control
- accountability
- audit
- object reuse
- accuracy
- reliability of service
- data exchange.

5.2.1 Generic Components of Corporate Security Policy

The generic headings point out in which way technical measures are effective.

Identification and Authentication

Identification is the act that enables the recognition of a subject like a user or another computer already known to the system by means of unique names. Authentication is the act of verifying a claimed identity. For example, the identity of a user contacting the system is often claimed by entering a login name and verified by entering a password.

Access Control

Access control is a measure of limiting access to objects (like files or devices) to authorised subjects. This includes the administration (the granting and revoking) of access rights and their verification. A subject becomes authorised by being granted the necessary access rights.

Accountability and Audit

These measures deal with the collection, protection and analysis of information on actions performed by subjects and on routine events or exceptional events that might affect security. The goal of accountability is that the actions of a subject can be uniquely traced back to it. The goal of audit is to be able to determine whether security violations have actually occurred, and if so which information or resources were compromised.

Object Reuse

Object reuse means that certain resources, such as main or peripheral storage, are consecutively used by different subjects or objects. Before reuse these resources must be prepared in such a way that no conclusions can be drawn concerning their former content and use.

Accuracy

Accuracy deals with the correct handling of data. In particular, it ensures that data passed between subjects and objects has not been modified in an unauthorised manner, and that relationships between data are accurately maintained.

Reliability of Service

Reliability of service is the property that objects are accessible and usable on demand by authorised subjects. In particular, this includes the guarantee that time-critical functions are performed when necessary. It requires measures such as error detection and error recovery, which are intended to restrict the impact of errors and, therefore, to minimise disruption or loss of service. In addition to the design aspect, reliability is an important factor in the quality of hardware and software to be used.

Data Exchange

This area covers the security of data during transmission over communication channels between different computing systems. All aspects above concerning a single computing system must be dealt with again. In particular, these are identification and authentication, access control, accountability and audit, accuracy and reliability of service.

According to the OSI reference model the following items are under the subject of data exchange:

- peer entity and data origin authentication
- access control
- data and traffic flow confidentiality
- data integrity
- non-repudiation.

5.2.2 Functionality Classes

Functionality classes combine a set of security features for special purposes. Functionality classes are defined in different sources (for example, the ITSEC specification, the CISR specification, the TCSEC specification, the Bellcore specification). Examples of functionality classes are given by the six classes C1, C2, B1, B2, B3, A1 in the TCSEC specification and ten classes F-C1, F-C2, F-B1, F-B2, F-B3, F-IN, F-AV, F-DI, F-DC and F-DX in the ITSEC specification.

There is still a need for functionality classes to be defined according to the needs of business and user communities. They should serve as a means for communication between users, procurement offices and vendors. The use of functionality classes is recommended for the procurement of IT systems, because it makes comparisons between systems easier. Functionality classes should be the base for a common understanding of security needs.

Mechanisms are used to realise security functionality. Examples for mechanisms are the use of passwords for authentication of users and the use of digital signatures to achieve authenticity of information. Sometimes the use of a special mechanism, such as the data encryption standard (DES) algorithm, is compulsory because of boundary conditions for the IT application. The strength of the applied mechanism strongly impacts the effectiveness of a security function. This is considered to be a matter of assurance.

5.2.3 Assurance Levels

One aspect of security functionality is fitness for purpose. The other aspect is the assurance that the IT security functionality cannot be deactivated, bypassed, corrupted or circumvented. Assurance is composed of *effectiveness* and *correctness* in the evaluation criteria. This corresponds to the fact that a security function may be compromised either due to an implementation error or due to a fundamental weakness in the design of a security mechanism or function.

Effectiveness is assessed by considering:

- the constructive aspects:
  - — suitability of functionality to counter identified threats
  - — binding of functionality
  - — strength of mechanisms
  - — construction vulnerability assessment
- the operational aspects:
  - — ease of use
  - — operational vulnerability assessment.

The assessment of correctness implies the consideration of:

- the development phases:
  - — requirements
  - — architectural design
  - — detailed design
  - — implementation including test
- the development environment aspects:
  - — configuration control
  - — programming languages and compilers
  - — developers' security
- the operational documentation for users and administration
- the operational environment aspects:
  - — delivery and configuration
  - — start-up and operation.

The lower assurance levels of the TCSEC specification and the ITSEC specification aim at basic confidence in security functionality. They are intended for protection against inadvertent acts. It is not appropriate for security or safety-critical applications. There is a high probability that security functions may be circumvented or deactivated by determined and knowledgeable intruders. In fact, it is expected that most commercial users would be satisfied with the assurance that the security functions are working as claimed by the vendor and proven through demonstrated successful execution of a full security function test suite. This is a level of assurance somewhat less than ITSEC E1. This concept could be further extended by having a third party perform the tests. With an increasing assurance level, confidence grows that the protection can withstand hard attacks under many conditions. The effort necessary for the construction and the evaluation process increases with growing assurance level. The main interest for a broad commercial community lies in the lower assurance levels. When high values are at risk, a careful analysis must be performed to decide whether a certain assurance level is sufficient.

5.2.4 Relationship between Functionality and Assurance

The separation between the aspects of functionality and assurance is crucial in adopting the technical requirements to the needs of many different scenarios. The functionality corresponds to the threats and the assurance corresponds to the risks associated with the threats. In different scenarios many different combinations of security functions are requested. The higher the risk to be reduced by technical measures the higher is the demand for confidence that the security functions work as claimed. In special cases different assurance levels for different security functions might be desirable.

Figure 5-1 illustrates the fundamental correspondence between threats, risks, functionality and assurance.

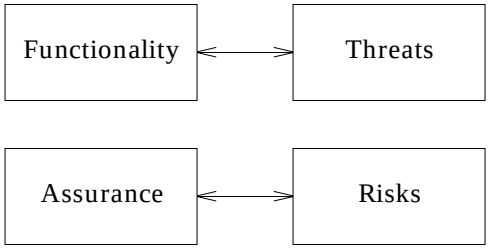

Figure 5-1 Fundamental Correspondence

5.2.5 Evaluation and Certification

Trustworthiness is the feature of an IT system that the users can rely upon the security functionality to work as claimed. Trustworthiness may be achieved through an independent evaluation and certification. Evaluation is the process of checking an IT system or product against certain criteria. Evaluation activities cannot be completely covered by usual quality assurance means. A certificate for an IT system or product shall confirm the evaluation result. It shall give the assurance:

- that the specified security functionality is present and working as claimed
- that the security functions have a certain effectiveness
- that this has been checked with a certain amount of effort and accuracy according to the claimed assurance level.

A certified system that complies with a high assurance level might still be faulty. However, a low assurance level of an evaluated system does not imply the existence of many bugs or a poor design. However, the higher the assurance level the lower is the probability of security-relevant errors.

For scalability, it seems desirable to support different assurance levels. A yardstick for different assurance levels must be defined. The assurance levels in the ITSEC specification are an attempt to do this.

Another matter is who may perform evaluation or certification (first party, the vendor, third party or an independent evaluation facility). To gain confidence in the evaluation results and the certificate, the evaluation facility and the certification body have to be impartial and objective.

5.3 Choice of Secure Products or Systems

In the procurement process the appropriate security functionality and the needed level of trust have to be fixed according to the organisation's security policy. This is achieved by extracting technical security requirements on the IT system by means of a risk analysis and management process (see Chapter 2). The number of security functions is not important; instead the fitness of the security features to counter the relevant threats to the IT application is important. The security functions should comply with the intended operational procedures. Very often the implications of a reasonable use of security functions are not considered and not known to all the people concerned. Almost any security functionality requires increased effort from users and administrators. Any security functionality whose application is not accepted and compromised by the people concerned is less efficient or even worthless. Information and education should be assisted by a manual (see the CISR specification, Exhibit A - Matrix, V. B.).

The following examples show that security functionality cannot be considered out of the context.

In the case of encryption being required to maintain confidentiality or authenticity of information exchange, there may be a problem if not all peers in the international network are capable of encrypting and decrypting in the same way (possibly because of national laws).

Mandatory access control (MAC) requires a considerable amount of administration effort (for example, for downgrading). Auditing usually impacts system performance (dependent on the degree of auditing performed) and requires a considerable amount of administration effort (checking audit trails).

For economic reasons, procurement offices prefer commercial products available on the market. On the basis of the technical security requirements and a market survey, a system should be composed of components or systems commercially available. It is easier to compare components and systems if the security functionality is expressed in terms of commonly-agreed functionality classes. If there is no suitable product available, an appropriate product has to be ordered from a vendor or supplier. In this case the technical security requirements must be communicated to the supplier. They should be expressed in terms of commonly-agreed functionality classes.

People in procurement offices should be aware of whether they only communicate pure security requirements (for example, problems like identification and authentication of users) or additionally prescribe special mechanisms to fulfill certain requirements (for example, solutions such as login with passwords, or smart cards).

If applications with different security needs cannot be separated and have to run on one system, it is necessary to check whether it is possible to satisfy all requirements. Usually if there are no conflicting requirements concerning functionality, a system that satisfies the more demanding assurance requirements is chosen. Normally compromises must be made, as commercial products available on the market do not exactly suit all purposes. Lack of security functionality can often be compensated by organisational, physical and personnel measures in the environment.

For example, if the access control in the system is not sufficient, it may be supported by measures controlling the physical access to the system. Alternatively a powerful accountability and audit functionality may deter people from unauthorised access, as they are then accountable for their actions. If there are no technical means of object reuse for storage media, they might be substituted by rules concerning the handling of storage media. If the identification and authentication features are not sufficient for a system connected to a wide area network, the access to the system might be restricted by time conditions, dial back procedures or interactive control of data exchange.

The required assurance level is associated with the risk that must be reduced by the security functionality. The assurance level must be high when the security of the application crucially relies on the correct functioning of the claimed security functionality. In this context it is necessary to fix the appropriate level of trust; that is, whether evaluated and certified systems should be applied and which kind of evaluation. For example, first party or third party, or anything similar, is desirable. Lists of evaluated products may then be helpful. They are available at national certification bodies.

Chapter 6

X/Open Security Classes

In this and subsequent chapters, the X/Open security classes are presented and, in the context of the five application scenarios, recommendations are given for their use. Security concepts are introduced and the basic concepts of open systems are outlined.

The classes reflect security features of the operating system of open systems. There is no special concern for security in networks, for security in heterogeneous environments, and for the security of applications running on open systems. Nevertheless it is a good idea to base the security functionality of an application on the fundamental security features supplied by open systems.

6.1 Security Concepts

In an IT system and its environment two kinds of entities can be identified: active entities and passive entities, called *subjects* and *objects* respectively. Typical examples of subjects are users or programs in execution; typical examples of objects are data files or devices. A subject can act on or access an object by reading, writing, changing or deleting it. There may be objects that can be executed on behalf of a subject. Such an object temporarily becomes a subject. An example is a data file containing an executable program or commands. It is possible that new abstract objects are realised by means of more elementary objects. An example is a database system realising relations by means of files or realising views by means of relations.

In order to control the activities of subjects in an IT system, rules must be set up which regulate the accesses of subjects to objects. A rule defines the access rights a certain subject is allowed to exercise on a certain object under certain conditions. A user may act in different roles dependent on tasks and responsibilities. Examples of roles are system administrator and normal users such as a member of a task force, software developer, cashier or office worker. The same user can have different rights dependent on the role he is adopting. Accordingly, programs working on behalf of a user may temporarily inherit different sets of access rights.

A fundamental security requirement is that only those activities covered by existing rights can take place in an IT system. If this cannot be ensured the following basic threats to security can occur:

- loss of integrity, where integrity is the property that information is protected against unauthorised modification or destruction
- loss of availability, where availability is the property of being accessible and usable upon demand by an authorised entity
- loss of confidentiality, where confidentiality is the property that information is disclosed to unauthorised entities.

These threats must be countered by a combination of environmental and technical measures. The technical measures can be grouped under the generic headings of the ITSEC specification, which are considered in Chapter 5.

6.2 Basic Concepts of Open Systems

The basic concepts of open systems are centred around *file*, *process* and *shell*. With respect to security two further notions, *user* and *group* (or *user group*), are of importance. A user is a person interacting with the system. A group is a set of users all of whom need to use a given set of objects. Every user is a member of at least one group. Users and processes are subjects whereas files are objects. Processes act on behalf of users.

The file system allows files to be created and removed. Files can be considered as named containers for data. They are objects that data can be written to or read from. There are four types of file as follows:

- A regular file contains data and is organised as a sequence of bytes. Files may be read and written in a random-access fashion. Normally, text or executable programs are stored in a regular file.
- A directory is a file containing directory entries. A directory entry associates a filename with a file of any type. Different names might be associated with the same file. Within one directory all names must be different. As directories may contain the names of other directories, the file system is hierarchically ordered. The user may read the data in a directory as though it were a regular file to determine the names of the files it contains. However, directory files are protected such that only the system can write to them.
- A device is a special file representing an I/O device. The interface to devices is the same as the interface to regular files. However, the information going to or coming from a device is not stored but directly processed by the I/O device. Dependent on the buffering method used in the device driver, block special files and character special files are used.
- A FIFO special file reads data on a first-in-first-out basis. It is normally used to pass data between unrelated processes.

Each file belongs to one owner. The file system stores information concerning ownership, protection and usage with each file.

In each system a set of processes runs. Each process is largely independent of other processes, having its own protection domain, address space, timers and an independent set of references to system- or user-implemented objects. User requests directed to the system are performed by processes. A new process is created by making a logical duplicate of an existing process. The new process becomes the child of the creating parent process. The parent process is able to wait for the child's finalisation. A process can overlay itself with the memory image of another process, passing the newly-created process a set of parameters. An example is the process managing the login procedure at a terminal. If a user has successfully performed a login, the process creates a child which overlays itself by a shell process.

The shell is an interactive interface between the user and the system and acts on behalf of the user. The shell performs commands read from the user's terminal device file or from a regular file. New files containing commands (shell scripts) can be generated; this allows users or groups of users to build up commands of their own. In addition, the shell provides an easy-to-use pipe mechanism which transforms the output of one process into the input of another.

6.3 Security Classes

Documents such as the emerging MSFR standard or the ECMA TC 36 standard state basic security requirements for operating systems such as those used in X/Open-compliant systems. At the time of writing this document, most X/Open-compliant systems do not fully meet the requirements expressed in the emerging MSFR standard and ECMA TC 36 standard. Typically however, a common base of security functions is provided, upon which other functions can be added. This document group sets of commonly-available capability into categories referred to as classes. The basic class is X-BASE representing the security functionality typically supplied by most open systems that are compliant with the XPG4 Base profile defined in the **XPG4** binder (see Chapter 7). X-BASE capability is not required by XPG4. Instead it is capability that is typically found in XPG4 systems. Indeed, none of the security classes defined in Chapter 7 to Chapter 12 inclusive represent requirements for security functionality. The intent is solely to categorise commonly-available capability or capability that is typically packaged as a security feature.

There are five additional classes, which describe sets of features that are becoming more common in XPG4 systems but are not yet typically available in most systems. These additional classes describe the security functionality typically available when a vendor offers discretionary access control, mandatory access control, auditing, privileges and special functions for distributed systems:

- Discretionary access control provides a finer grained control in specifying user or group access to objects. The features commonly available with discretionary access control are described by the X-DAC class (see Chapter 8).
- Mandatory access control provides a means of restricting access to objects which is based on the sensitivity of the information contained in the objects and the formal authorisation of subjects to access information of that sensitivity. The features commonly available with mandatory access control are described by the X-MAC class (see Chapter 9).
- Security audit systems provide the property that the actions of a subject can be uniquely traced back to the subject. The features commonly available with security audit systems are described by the X-AUDIT class (see Chapter 10).
- Privilege mechanisms provide the capability to limit subjects to execute only the set functions and utilities needed for that subject to perform its authorised task. The features commonly available with a privilege mechanism are described by the X-PRIV class (see Chapter 11).
- Some systems provide additional security functionality for distributed applications whose components reside on networked X/Open-compliant systems. The typical security features available for distributed systems are described by the X-DIST class (see Chapter 12).

These additional classes are presented under the generic headings of the technical measures described in Chapter 5.

The capability is related to the requirements of functionality classes in the ITSEC specification and to the requirements of the CISR specification.

Figure 6-1 on page 68 shows the scope of the security classes.

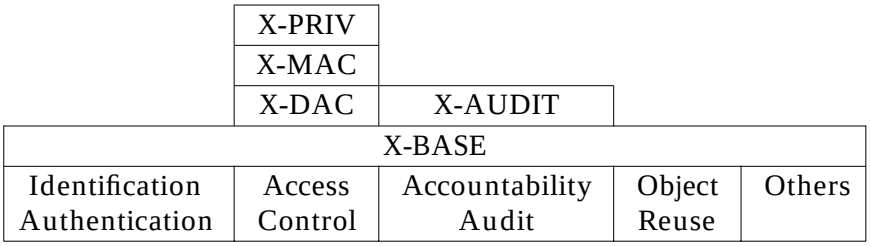

Figure 6-1 Scope of Security Classes

6.3.1 Classes and Systems

Today, systems in the market are usually oriented towards the classifications given in the TCSEC specification, and the equivalent classes provided in the ITSEC specification. Systems are often advertised to be ''designed to meet the requirements of TCSEC class C2'', or ''according to ITSEC F-B1''. The list below gives a mapping between the X/Open security classes and the security functionality of the TCSEC specification or the ITSEC specification.

X-BASE All systems, independent of the claimed security level.

X-DAC Systems providing Access Control Lists (ACLs), often at TCSEC level B1 or higher; all systems reaching ITSEC F-C2.

X-MAC Systems designed to meet TCSEC B1 or ITSEC F-B1.

X-AUDIT Systems designed to meet TCSEC C2 or ITSEC F-C2.

X-PRIV Systems designed to meet TCSEC B2 or ITSEC F-B2.

X-DIST The security functionality is outside the scope of TCSEC. Systems providing security functionality for distributed systems, such as Kerberos or OSF/DCE, usually meet these requirements.

Note: The security functionality you need may be found in other systems not making such a claim, as many systems provide only partial implementations of the TCSEC functionality classes. If a system claims to reach a certain class, you can use the list to find out which X/Open classes are covered.

Chapter 7

X-BASE

This chapter describes the capability typically available in X-BASE. It includes Identification and Authentication, Basic Access Control, Accountability and Audit, and Object Reuse.

7.1 Identification and Authentication

Identification is the function that enables the system to recognise a user or entity as being authorised to access the system. Authentication is the function that proves beyond reasonable doubt that the identified user or entity is as claimed. No useful work should be possible on a system until the user has been identified and authenticated.

Identification and Authentication are fundamental security requirements. A fully identified and authorised user is assumed and required for most other security functions to be effective, especially Access Control, Accountability and Audit.

7.1.1 What Identification and Authentication Provide

A unique identification code is provided for every user of a system. The system maintains, under tight security, a file or directory of these codes. When a user logs on to the system the identification code he submits is compared with the system's stored value. If a successful comparison is made, the user is asked to authenticate himself. Many different mechanisms may be provided for authentication, offering different levels of resistance to abuse or misuse. The most commonly-available in X/Open-compliant systems is a password mechanism. Dependent upon the security policy selected for an installation, a number of rules relative to password usage must be observed. Typically the alphanumeric content and length of a password are controlled, as is its period of use before replacement. In general, rules are selected to minimise the chances of a password being guessed by an unauthorised user.

7.1.2 Using Identification and Authentication

Careful consideration must be given to the formulation of password content and change rules. Opinions vary; each organisation must decide its own policy. Some organisations favour rules that enforce complex passwords (even system-generated passwords) that must be changed frequently to thwart attempts to gain unauthorised access. Others feel that the more complex a password and the more often it is changed, the more likely it is that the user cannot remember it without writing it down, thereby jeopardising security. A common belief is that adequate security against password-cracking attacks is achieved by ensuring that:

- The Identification and Authentication system file is only accessible by a privileged program and never readable by a human.
- It is used in conjunction with some simple password content and change rules.

However, if passwords are transmitted over networks where eavesdropping is possible, password complexity and expiration mechanisms do not provide adequate protection. In that case, stronger methods are needed (see Chapter 12 on page 89). For user to host authentication, token authenticators can be used to protect against eavesdropping and remove some of the risks associated with unreliable user practices for protecting their passwords, as follows:

- Terminals connected directly to an open system always require identification and authentication established by the system itself. With network connections, many systems allow remote logins that trust the authentication of the originating host. Use of this feature over untrusted networks, or listing hosts that cannot be trusted in the trusted hosts database, may defeat security and should only be used with great care.

  One particular problem in the networking area is the trivial file transfer protocol (*tftp*) program, which is usually used for specific administrative functions, for example, downloading boot code and font files for X terminals. Most versions of X/Open-compliant systems are able to restrict the use of *tftp* to this purpose.

  Since *tftp* requires no identification and authentication, it should never be configured in a way that allows users to download files interactively. The File Transfer Protocol (*FTP*), which provides identification and authentication, should be used for that purpose.

- Individually assigned, unique user IDs are necessary for accountability. Although every open system allows administrators to choose unique user IDs, they do not necessarily enforce this.

- Temporarily disabling unused accounts increases security. Such accounts are very attractive to attackers because their presence may not be noted for quite a while. The **Security Guide** gives advice on how to disable accounts by editing the password file.

- One well-known trojan horse attack on systems is to leave a terminal with a program that mimics the standard login program. If a user tries to log in, this fake login program captures the user's password and thus compromises his account. This type of attack can be prevented with a *trusted path* between the machine and the terminal. With this mechanism, the user enters a secure attention key (SAK), which makes sure that all processes running for this terminal are killed and their real login program is invoked. However, SAKs are not provided by many open system vendors.

7.1.3 Features Commonly Available

The following features are commonly available on open systems:

- A mechanism is provided to establish and verify the claimed identity of a user.

- A means of associating collections of related users into groups and assigning group privileges is provided. Users may belong to more than one group.

- A mechanism is provided for forcing every user to prove his authenticity before any work can be carried out on the system. This is supported with customisable warnings of the consequences of unauthorised use.

- At a minimum, a password-based authentication mechanism with the following features is provided:

  — A password is one-way encrypted before storing.

  — A password is never echoed when typed in by a user.

  - — A password is user-changeable. Only the user or an administrator with appropriate privilege can change the password. A user can only change his password after reauthentication.
  - — The administrator can inhibit the possibility of changing passwords on a per-account basis.
  - — A mechanism is provided for dealing with access attempt failures without providing useful information to an unauthorised malicious user.
- A means is provided by which only an authorised and highly privileged user may manage the contents of the user identification file and, for example, temporarily disable users.
- An Application Programming Interface (API) is provided to access the authentication mechanism from applications.

7.1.4 Features Sometimes Offered

The following features are sometimes available on open systems:

- Administrative procedures may be provided to introduce new users and enforce unique user IDs for the purpose of accountability.
- The mechanism dynamically to change user IDs may be restricted for users that are allowed to change to a privileged user ID.
- The user database may contain additional, customer-defined identifying information, such as user name, affiliation and telephone number.
- Authentication information, such as the encrypted password, may be stored in an authentication database accessible only by authorised users.
- The system may enforce a minimum password length and a minimal password complexity in terms of a mix of alphabetic, numeric and special characters.
- The parameters for password complexity may be modifiable by the administrator at site.
- Checking for easy-to-guess passwords may be provided, for example, password identical to user ID, certain combinations of user ID, or dictionary words.
- Password generators may be provided; they produce pronounceable passwords that are hard to guess, but easy to remember.
- System applications to support password management may be provided, such as the configuration of default policies and other parameters, on a system-wide or per-account basis.
- A password ageing mechanism, on a per-account basis, may be provided that allows the administrator to specify a minimum change time and an expiration time of a password in weeks.
- Additional password ageing parameters may be provided, such as a warning to the user before the password expiration date. Such parameters would require the appropriate administrative tools to manipulate them.
- A history mechanism may be provided, on a per-account basis, to check reuse of old passwords.
- Tools to check the consistency of the user and authentication databases may be provided.
- The capability of using alternative authentication methods may be provided. This could include the possibility of branching securely during authentication to code not supplied by

the vendor, or the possibility of specifying the authentication methods to use for specific users and groups.

- A system-defined Secure Attention Key may be available.
- It may be possible for an authorised user temporarily to disable all logins of non-privileged users to the system.
- It may be possible to restrict access to the system to specific times (time of day, weekday, date).
- It may be possible to restrict system access for privileged accounts to certain terminals or hosts.
- It may be possible to block accounts after a certain number of consecutive, unsuccessful login attempts.
- It may be possible to block login devices or remote hosts after a certain number of unsuccessful login attempts.
- It may be necessary for a certain interval of time to elapse after an unsuccessful login attempt, before the next attempt can be made.
- After successful login, the date, time and location of the last successful and unsuccessful login by that user may be displayed.
- The system may provide the capability for automatic disconnection or reauthentication of users after a specifiable period of inactivity.
- The system may provide a mechanism for user-initiated locking of his terminal's input device. Unlocking requires user authentication.

7.1.5 Applicable Standards

Standard requirements for Identification and Authentication are being defined in the emerging MSFR standard and ECMA TC 36 standard.

Standard operating system interfaces for these functions are being defined in the emerging POSIX .6 standard.

Many *de facto* standard tools for password management are available to supplement base operating system capability, for example, from the Computer Emergency Response Team (CERT) at the Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, U.S.A.

7.2 Access Control

Once a user has been identified and authenticated, the system mediates his access rights to the system's resources. Access control deals with the right of subjects to access objects. Subjects are users and processes acting on their behalf. Objects are files, devices and interprocess communication (IPC) objects.

System access control mechanisms permit implementation of organisation access control policies on computer systems. With proper configuration, unauthorised access to objects can be prevented.

7.2.1 What Access Control Provides

Open systems conformant to current X/Open specifications provide a standard access control mechanism based on permission bits. There are three classes of users: the owner, the users belonging to the group associated with the object and all other users. For each class of user, there are three bits: read access, write access and execute/search.

7.2.2 Using Access Control

Each object has both an owner and a group associated with it. The owner of an object determines who may access it. When an object is created, the permission bits are initialised according to a configurable default.

The administrator typically defines the membership of each group. Groups can be used to restrict access to an arbitrary number of users. A user can simultaneously be a member of several groups.

7.2.3 Features Commonly Available

The following features are commonly available on open systems:

- The system controls access to all files, devices and IPC objects.
- Control of access to resource objects is based on user identification.
- Access rights recognised by the system are read, write and execute. For directories, the execute access right controls searching.
- Access rights are specified for the file owner class, file group class and file other class of every file. Every subject trying to access an object falls into exactly one of these classes:
  - — file owner class: if the effective user ID of the process matches the user ID of the file
  - — file group class: if the process is not in the file owner class and its effective group ID (or any of its supplementary group IDs, if the implementation supports them) matches the group ID of the file
  - — file other class: if it is neither in the file owner nor in the file group class.
- Access rights of the file owner class take precedence over the access rights of the file group and other classes. Access rights of the file group class take precedence over the access rights of the file other class.
- Only processes with appropriate privilege can override access control checks. Commonly-called *set-user-ID* programs (processes that can override access control checks) must be carefully designed and implemented to limit the override to specific objects and purposes. Set-user-ID programs are frequently written carelessly, thus permitting an attacker to misuse the program to bypass system controls.

- If a file mode is not specified during creation of a new object, the default access rights are determined by a pre-defined file-creation mode mask associated with the creating process.
- When creating a new object, its user and group IDs are set to the effective user and group IDs of the creating process.
- Access rights for a file, device or an IPC object can only be changed by a process being in the file owner class or having appropriate privilege.
- Access to a file, device or IPC object is checked during the ''open'' of the resource. If an object's access rights change after it is opened, subjects can continue to use it until the object is ''closed''.

7.2.4 Features Sometimes Offered

The following features are sometimes available on open systems:

- Particular system implementations may provide additional or alternate file access mechanisms, for example, the one described in Chapter 8 on page 79, or both.
- Additional file access mechanisms may further restrict the access permissions defined by the file permission bits.
- In directories having the *sticky bit* set in their file mode, unprivileged processes can only remove entries if their effective user ID matches the user ID of the file pointed to by that entry.

  Note: Remember that the ability to remove a file from a directory depends on the write access to the directory itself, and not on the permission bits of the file. When a directory has the sticky bit set, only the file owner can remove a file from that directory. This is useful for world-writable directories, for example **/tmp**, because it prevents other users from removing files they do not own.

7.2.5 Applicable Standards

All the capability described in Section 7.2.3 on page 73 is fully specified in the X/Open specifications **System Interface Definitions** and **System Interfaces and Headers**. Application programming interfaces for Section 7.2.3 are provided in the emerging POSIX .6 standard.

7.3 Accountability

The goal of accountability is to provide traceability for security-relevant actions. This provides a means of tracing activities back to individuals who can then be held responsible for their actions. Current, basic X/Open-compliant systems typically do not provide exhaustive audit functions expressly for user accountability. However, typical system accounting functions, designed primarily to monitor system performance, can provide some measure of audit that can be used for security purposes.

This section covers the use of system accounting functions for security audit. In Chapter 10 on page 85, additional audit functions specifically for security monitoring are described.

Quite often, organisations need to determine what actions were taken by specific users. This analysis may be performed following a system break-in, system security policy violation or as a matter of routine. Sometimes, after system failures, it is required that the system be reset to its previous state. Additionally, applications such as run-time analysis and intrusion detection are dependent on information describing every system action and the user responsible for initiating the action.

All of these activities require information detailing the sequence of events which took place on a system, as well as the information describing the user initiating the actions.

7.3.1 What System Accounting for Security Provides

Achieving user accountability requires a detailed trail of events that occur on a computer system. This information in turn provides the data required for the investigation of events such as compromise of system security, break-ins and system security policy violations. To provide this information the system must record data sufficient to determine the action that occurred, the object or objects affected and the user initiating the action. Administrative utilities, to initialise, initiate collection of and report the data must also be provided.

On most X/Open-compliant systems, this capability is partially provided by the system's process accounting facility. In particular, system processing accounting records processes invoked and the invoking users, but does not record objects accessed. Reporting utilities are also geared more towards system performance issues rather than security.

7.3.2 Using Process Accounting for Security

In order to ensure that a detailed trail of events is recorded, system process accounting results in the recording of significant amounts of data. The amount of data varies based on the size of the system, number of users and system activity. In some cases, the amount of data recorded by process accounting may become excessive. In this case accounting data may be stored on remote media, compressed or purged. Because the integrity of the accounting data is a critical element in providing traceability, protection of the accounting data is a primary concern. If the accounting data can be modified, violations may be hidden and false or misleading data may be introduced.

7.3.3 Features Currently Available

Most X/Open-compliant systems provide a process accounting facility which provides logging information, principally for accounting purposes. These functions also provide logging of certain security-relevant events:

- Major system events are recorded, such as:
  - time and date of system boots and shutdowns
  - changes to the system's clock
  - login and logout of users (the system records date, time, device and ID).
- If per-process accounting is enabled, a record is generated for each process that exits, containing:
  - command name
  - exit status
  - accounting user ID
  - accounting group ID
  - controlling terminal
  - start time
  - user, system and elapsed time
  - average memory usage
  - characters transferred to terminals
  - blocks read/written to disks.
- Utilities are available to interrogate the accounting log files and display information such as the last login of every user, all users who have logged in since last shutdown, and so on.
- Accounting files may be searched and printed using a variety of selection criteria, including:
  - terminal line
  - user ID
  - group ID
  - start/end time
  - command name
  - resource usage.
- Most X/Open-compliant systems provide the capability to log system messages, usually printed to the system console, into log files. Also many system daemons keep log files on their activities or use a central logging daemon to do this.

7.3.4 Features Sometimes Offered

Tools that provide real-time analysis, intrusion detection and disaster recovery based on the accounting data as well as high-level accounting report facilities are frequently available. A derivative of process accounting, auditing expressly for security purposes, is also available on some X/Open-compliant systems (see Chapter 10).

7.3.5 Applicable Standards

The emerging MSFR standard and ECMA TC 36 standard describe baseline requirements for auditing for security purposes. The emerging POSIX .6 standard provides standard interfaces for an audit facility that can be used by the operating system and applications.

7.4 Object Reuse

The goal of object reuse is to prevent unauthorised, unintentional disclosure of information. This may occur, for example, if the disk blocks allocated to a deleted file are not purged before they are reallocated to a new file. In this case, the user who creates the new file may have access to information previously deleted by another user. Object reuse controls ensure that data contained in deleted files cannot be reaccessed.

Most open systems provide some form of access control, providing the ability to restrict access to user and system data. However, if data believed to be deleted is made available when data blocks are reallocated, a user can easily circumvent system access controls.

7.4.1 Considerations For Object Reuse

No standard implementation specification is used for object reuse. Most X/Open-compliant systems do not clear data blocks when files are deleted and the blocks placed on the free list. Instead, cleansing is done by reallocation. Access to the blocks on the free list, where valuable data may still be available, requires direct access to the disk or memory. On X/Open-compliant systems, access to disk or memory is controlled by the system access control functions.

7.4.2 Features Commonly Available

Open systems protect against object reuse in varying ways. Some systems clear data blocks upon reallocation, while others restrict the viewing of data beyond the logical end-of-file (EOF).

7.4.3 Features Sometimes Offered

Almost every open system provides the object reuse capability described in Section 7.4.

7.4.4 Applicable Standards

Object reuse is a requirement in the C2 class of the TCSEC specification and the F-C2 class of the ITSEC specification; it is included in the emerging MSFR standard and the ECMA TC 36 standard.

Chapter 8

X-DAC

This chapter describes security capability that provides discretionary access control; it enhances the access control capability of X-BASE to provide a finer granularity in access control than is possible with the permission bit mechanism of X-BASE. The ACL is used as an additional access control mechanism, not as an alternative mechanism to X-BASE.

The ACL supports the capability to define access rights for:

- multiple (named) users
- the owner
- multiple groups
- everybody else (other).

This also makes it possible to specify that a particular user or group does *not* have any access to a resource, by explicitly excluding that user or group.

8.1 What Discretionary Access Control Provides

For each resource on the system, an ACL is maintained with entries for selected (named) users, selected groups and other. Access rights (read, write and execute/search) are defined for each specified user, for each group (that is every user who is a member of that group) and for all other users of the system.

The X-BASE permission bit mechanism can be considered as a subset or a special case of the ACL mechanism, with an ACL defining exactly three entries for the file owner class, file group class and file other class. Thus, in some environments the term ACL is used for both mechanisms.

An example ACL could yield the following external representation:

```
#file   : draft.txt
#owner  : mary
#group  : xopen
user    :           : rw-
group   :           : r--
mask    :           : r--
user    : fred      : rw-   #effective: r--
user    : john      : ---
group   : xotgsafe  : rw-   #effective: r--
group   : xotechman : r--
other   :           : ---
```

The output of the command:

```
ls -l
```

would show this file as follows:

```
-rw-r-----+  1 mary     xopen 9150 30 Mar 13:06 draft.txt
```

Note the following points about the ACL above:

1. Some entries have an empty second field. These are the three entries for the user (file owner class), group (file group class) and other (file other class), and the entry for the

mask. The empty user entry refers to the owner of the file (mary); the empty group entry refers to the file group (xopen). If the owner or file group changes, this entry needs no update, as it implicitly references the user ID and group ID associated with the file.

2. The file group class is enhanced and now contains the file's group, and all users and groups listed in qualified entries of the ACL (that is, all user and group entries having a name associated with them). The mask entry is used to note the maximum access rights granted to any of the file group's members. This has the advantage that programs using *chmod*( ) can still work together with ACLs, as they do not affect the qualified user and group entries.

3. The ACL allows read access to the file for the file owner class, which is currently user mary, the file group class, xopen, the user fred and the user groups xotgsafe and xotechman. To specify these access rights in a system without ACLs, you would need a group encompassing all the members of the groups xopen, xotgsafe and xotechman, as well as the users mary and fred. You would have to make sure that user john is not a member of this group, as you want to deny access to him in any case.

4. Although the user fred and the group xotgsafe also have the right to write the file, they would currently not be allowed to do so, as the mask does not give write access to the members of the file group class. The output denotes this with the #effective flag that shows the currently effective access rights for that entry. If the mask entry is changed, for example, with the command:

   ```
   chmod g+w draft.txt
   ```

   then user fred and group xotgsafe have write access. However, the group xotechman is still only able to read the file. All other users are not allowed to access the file at all. This situation (access for read and write permission for one group, read permission for another group, and no access for the rest) cannot be modelled with the traditional permission bit mechanism.

5. The user john will not get access to the file, even if the file permission bits allow everybody else to access the file, for example if the command:

   ```
   chmod ugo+rw draft.txt
   ```

   is used to give access to file owner class, file group class and file other class.

6. The output from *ls* shows a + sign after the permissions to indicate that an ACL exists for this file.

8.2 Using Discretionary Access Control

The major drawback of ACLs is the increased complexity of setting access controls during file creation, changing them for existing files and so on. This complexity impacts both application programmers and users setting access controls directly.

The emerging POSIX .6 standard ACL mechanism supplements and enhances the permission bit mechanism, but does not replace it. This is important because the mechanism is designed to coexist with the permission bit mechanism; most applications continue to work when used in an environment using ACLs.

8.3 Features Commonly Available

Most ACL mechanisms can control access to all resource objects at the granularity of a single user. For each resource object, it is possible to specify a list of users and groups with their specific access rights to this object. These access rights may specify that *no* access is allowed; in other words, it is possible to specify that particular users or groups do *not* have access to the object.

8.4 Features Sometimes Offered

Access control to resource objects may be further refined by ACLs supporting restrictions such as:

- method or location of the user, for example, a user may have access to a particular file from a specified terminal
- time, date, day of week
- program used to access the resource object.

8.5 Applicable Standards

Discretionary access control is one of the areas addressed by the emerging POSIX .6 standard by defining an ACL API.

Chapter 9

X-MAC

This chapter describes security capability that provides support for mandatory security policies. A mandatory security policy is a system-enforced access control policy as opposed to an access control policy left to the discretion of the user.

Discretionary access control relies on users to share information safely through judicious and careful use of optional system facilities. If the appropriate user control and care is not properly exercised, undesirable disclosure of information may occur.

If your system contains particularly sensitive information that must never be accidentally disclosed to unauthorised users or if the system is supporting different groups of users that should never be able to access each other's information, the facilities of mandatory access control should be considered.

9.1 What Mandatory Access Control Provides

All users (subjects) and all information (objects) have an associated security label. This label consists of a hierarchical classification level and a set of categories or groups. The subject's label defines what levels and categories of information the user may access. The object's label defines the security level of the information.

Whenever a user or subject attempts to reference an object the system compares their labels and determines whether access should be allowed. The site must set up the system so that all subjects and objects carry the appropriate labels. To the user, the enforcement of mandatory access control rules does not require any action.

As an example, the security policy may require that all information on the system be labelled as *open*, *company confidential* or *company secret*. Typically, users carry only one of these labels during a logon session. As long as the user's classification is greater than or equal to that of the information, the system allows access, subject to discretionary access controls if they exist.

In addition, a security policy may define categories such as: **development**, **marketing**, **personnel** or **finance**. Users cleared for access to marketing information would be automatically excluded from referencing information in the other categories, independent of their classification level. Even if classified information is not stored on the system, the category feature is useful for separating departmental information.

Strict rules are enforced for the upgrading and downgrading of labels. For example, a user logged on at the company confidential level may not create **open** data but may read **open** data and may create **company secret** data.

An X-MAC system is equipped with a mechanism or procedure for adding appropriate labels to unlabelled information entering the system. All information leaving the system, such as output to a printer, carries the classification labels, which must be displayed as appropriate for the output media.

9.2 Using Mandatory Access Control

The main drawback of mandatory access control is the increased complexity introduced by the use of mandatory access control facilities.

9.3 Features Commonly Available

Mandatory access control is normally implemented to comply with the TCSEC specification, levels B1 and above.

The functions are as follows:

- Support is provided for user and object labelling including classification and category.
- Automatic enforcement of access control is provided by the system based on labels.
- Automatic assignment of labels to newly-created or introduced objects is provided.
- A mechanism is provided for preventing a user from detecting the presence of an object to which he has no access rights.

9.4 Features Sometimes Offered

No implementations offering additional mandatory access control capability are known at the time of writing.

9.5 Applicable Standards

X-MAC capability is specified in the TCSEC specification, levels B1 and above. The emerging POSIX .6 standard describes portable APIs for the setting and manipulation of mandatory access control labels.

Chapter 10

X-AUDIT

This chapter describes auditing features that are additional to those in X-BASE. The goal of auditing is to provide accountability for security-relevant actions. Accountability is the property that the activities on a system can be traced back to individuals who can then be held responsible for their actions.

Audit mechanisms provide a powerful detection capability to complement preventive measures. However, they have an overhead associated with administration and resource. Auditing is a passive measure that records user access to resources. As such, it can permit detection of security policy violation or security mechanism breach. In addition, potential attackers may be deterred by the fact that they can be identified and held responsible for their actions.

10.1 What Auditing Provides

Auditing generates an information trail containing security-relevant events. The exact contents of the audit trail are determined by the security policy.

Audit events can be recorded by system commands, system calls or applications.

10.2 Using Auditing

Protection of the audit trail is important; attackers may hide their actions by manipulating the audit trail. Users cannot be held responsible for their actions if they can claim that somebody else has modified the audit trail in order to trap them.

The amount of data generated by the audit system can be huge. The capability of selecting the events to be audited on a per-user basis can substantially reduce the amount of data being written to the audit trail.

If the audit system fills up the available disk space, it denies service to users until the administrator takes the appropriate action. The corporate security policy may require that auditing be switched off or reduced, or the machine may be shut down.

Every site needs to investigate carefully the tradeoffs between the goals of accountability and availability. You need to find the optimum balance between the desire to audit as much as possible and the necessity to use only a minimum of system resources. A risk analysis can provide the necessary information on the types of audit information that would have to be collected at a given site.

Auditing presupposes identification and authentication. Only if users can be identified, can they be held responsible for their actions. Typically, X/Open-compliant systems can be set up so that every user has a unique user ID, which can then be used by the audit system.

It is important to note that it is extremely hard to protect an audit system against intruders that gain privileged status.

10.3 Functionality Commonly Available

X/Open-compliant systems typically offer basic system accounting capability (see X-BASE). X-AUDIT enhances this capability to provide a more detailed audit trail which can record user activities to a very fine granularity. The following features are commonly available on open systems:

- The capability is provided to enable or disable dynamically the types of events recorded based on user ID, port or other criteria.
- As a minimum the following are recorded: authentication, access or change to resource objects, modification of audit configuration and changes to the system security parameters.
- For each event, the following information is recorded: event type, a time stamp, user ID, resource involved, and indication of success or failure of the user's operation.

10.4 Features Sometimes Offered

In practice few systems provide the complete set of auditing capabilities. Useful additional capability provided by some vendors include: tools that can produce exception reports, summary reports and detailed reports on specific users; resource objects; time periods; or communication facilities.

In addition, administration capability is sometimes provided to support automated backup and deletion of audit trails to prevent loss of data due to exhaustion of storage facilities.

10.5 Applicable Standards

The emerging POSIX .6 standard defines interfaces for the recording and analysis of audit records; the emerging MSFR standard and the ECMA TC 36 standard require a number of tools to consult, summarise and interpret audit databases.

The emerging POSIX .6 standard describes portable APIs for single-system audit capability. The emerging POSIX .6 standard currently has not defined a portable audit data format, making the provision of a heterogeneous network-auditing tool difficult to implement. POSIX has promised to investigate this issue.

The emerging MSFR standard and the ECMA TC 36 standard provide the requirements for the audit system for operating systems. These include the minimal set of events that should be audited and requirements for administration of the audit subsystem.

Chapter 11

X-PRIV

A privilege is the ability to exercise a controlled or restrictive service. Privilege mechanisms provide a finer granularity of access to services requiring appropriate privilege. Each subject authorised to perform a security-relevant action in the system is granted only the most restrictive set of privileges necessary to accomplish this task. This chapter describes the features of a privilege mechanism (X-PRIV).

In systems without a privilege mechanism such as X-PRIV, all privileges are combined into a single account. This account can perform every security-relevant action, which may lead to security problems. The problems become more evident as systems grow larger and administrative power is split into several roles, such as operator, user administrator, audit administrator and so on.

A privilege mechanism defines distinct privileges that can be assigned to separate accounts so that those accounts are only empowered to accomplish specific tasks.

11.1 What Privileges Provide

Features provided by a typical privilege mechanism allow fine-grained system control and the composition of administrative roles from sets of privileges. This has important advantages:

- If an intruder gains a privileged status, for example, by breaking into an administrative account, he does not get every privilege existing in the system. This limits the possible damage.
- Administrative tasks may be separated into different roles, so that several administrators can be assigned. For example, backups, user administration and auditing could be three tasks performed by three different administrators.
- The damage caused by errors made by administrators is limited.

Assigning privileges to programs provides a mechanism similar to the set-user-ID bit mechanism in standard systems.

The reasons for implementing both mechanisms are as follows:

- Privileges assigned to programs rather than to users put the execution of a privileged program under the control of a program that is trusted to do only the actions it is supposed to do. This can help to avoid disastrous errors that could occur when a user has a certain privilege and makes an error, for example, typing:

  ```
  rm *
  ```

  in the wrong directory. Trusted programs usually do some sanity checking before carrying out sensitive actions.
- Even unprivileged users need to perform privileged tasks. One example is changing a password. The utility that updates the password database needs the privilege to access it.

The advantage of the privilege mechanism over the set-user-ID approach is the flexibility provided by the privilege combinations, which may be important for complex applications.

11.2 Using Privileges

There is an additional administrative overhead in specifying the proper set of privileges for each administrative task, for example, backup and restore. Existing applications may no longer function correctly.

In a system with privileges the set-user-ID or set-group-ID mechanism remains useful, as this approach helps to build *protected subsystems*. Protected subsystems are a combination of data files and programs, where access and modification of the data files is only possible via the accompanying programs. However, often a combination of both mechanisms is desirable, especially when protected subsystems want to use other privileged system services.

Giving all privileges to a single account is usually not a good idea.

11.3 Features Commonly Available

The following features are provided on open systems:

- Most systems provide fine-grained privilege mechanisms. On such systems, typically the following privileges are provided:
  - — The discretionary access control mechanisms can be overridden.
  - — Signals can be sent to arbitrary processes.
  - — The real user ID or real group ID of a process can be set.
  - — Restrictions for setting set-user-ID or set-group-ID bits on a file can be overridden.
  - — Privileges can be assigned to a file.
  - — The system can be halted and rebooted.
  - — Process limits and file system quotas can be overridden.
  - — Device special files can be created.
  - — File systems can be mounted.
  - — Set-user-ID programs can be executed.
  - — Data can be imported and exported.
- Actions requiring appropriate privilege by the user can only be carried out if the user holds the corresponding privilege.
- Privileges are associated with individual users and programs.

11.4 Applicable Standards

The emerging POSIX.6 standard and the emerging MSFR standard are expected to cover restricted privileges.

Chapter 12

X-DIST

This chapter describes security capability that can be used to improve the security of distributed applications whose components reside on one or more networked X/Open-compliant systems. For the most part, X-DIST is really the extension or use of X-BASE capability to provide security for distributed applications. In some cases, new capability is needed in X-DIST over X-BASE, while in others, X-BASE capability can be used without change. When X-BASE capability is used, typically, additional administration is necessary to ensure consistency of security information among all components of the distributed environment.

In order to avoid repeating capability covered in X-BASE, X-DAC and so on, it is assumed that all capability in those areas can be applied to distributed systems, except where noted.

In a distributed environment a company may interconnect different organisations with their own security policies, authentication methods, user security attributes and machine evaluation ratings. The administrators of such a network may want to provide an integrated system, in which users only have to log in once to access resources anywhere in the network. In addition, the administrators might want a single point of control for system administration.

12.1 What Distributed Security Provides

In a distributed environment, the user sees consistent security across heterogeneous systems which may consist of multiple security domains with:

- multiple security policies
- multiple authentication methods
- multiple user security attributes
- multiple machine evaluation ratings.

If network eavesdropping is a threat, authentication mechanisms using passwords that may be compromised via network eavesdropping can be replaced with strong authentication.

Strong authentication is such that the authentication information transmitted to the service is sufficient to prove the identity of the claimant, but does not provide information that can be used by the service or an eavesdropper to masquerade as the claimant.

12.2 Using Distributed Security

A user logs in and authenticates once to the distributed system, and then can access resources and services throughout the network. This user is accountable for all his security-related actions across the distributed system. Identification with distributed systems is difficult without a global name space.

Implementation of strong authentication typically requires cryptographic mechanisms. Open systems usually provide cryptographic functions, or support for cryptographic software, that can be used by applications to implement standard or proprietary, strong authentication mechanisms. Therefore designers of international networks have to consider national regulations on the export and use of encryption. It is possible that the encryption algorithm and the portions of data encrypted vary across the network.

Although the user sees consistent security, an administrator faces an additional burden of managing multiple security policies and different environments.

Centralised security services may introduce potential vulnerabilities such as single point of failure.

Some machines in the distributed system such as PCs may not have any security features. Other machines may have different sets of security features (as represented by their evaluation levels or configurations). Some of the systems may be collected into security domains with unique security policies.

Some security attributes such as ACLs may be inconsistent in different portions of the network. Other attributes such as mandatory access control labels and privileges may not be supported in a distributed environment.

A variety of protocols (for example, TCP/IP, OSI and SNA) may have options securely to transmit security attributes.

Audit formats may vary on different systems and may not be compatible.

Computers can be told to lie about their network address almost as easily as humans can lie about their claimed identity. Thus, falsification of identity is no more a problem for distributed systems than it is for monolithic systems if proper authentication is performed.

Network services for clock synchronisation, that is, a time service, are generally available. Time services are a basic requirement for distributed computing and have applications beyond security. Time services that distrust large time change deltas should be implemented to prevent attackers from resetting clocks in support of replay attacks.

12.3 Features Commonly Available

This technology is currently not commonly available.

12.4 Features Sometimes Offered

Strong mechanisms for distributed application security are beginning to appear in the marketplace and these mechanisms can be integrated into open systems with varying degrees of difficulty. OSF/DCE Security and the Kerberos System from MIT are examples of available mechanisms.

However, the fact that such mechanisms are not widely distributed with the base system does not mean that the currently known techniques are immature or inherently weak. In fact, most open systems are equipped with tools that can be immediately used to improve the security of distributed applications. The major issue is that frequently, today such mechanisms must be developed from more primitive tools and integrated into applications, rather than being integral to the underlying system. In most cases, use of existing mechanisms is more of an administrative burden than one truly related to distributed security.

12.4.1 Identification and Authentication

The following identification and authentication features may be provided:

- A unique identity may be established for all *principals*, that is, users and services, within the distributed environment.
- Distributed authentication mechanisms (which cannot be bypassed) may verify the identify of users to services and of services to services prior to service connection and communication.
- An authentication mechanism may be provided that does not expose authentication secrets, for example, keys or passwords, to other entities on the network.

  Kerberos is an example of a strong authentication and key distribution system. Kerberos is available from MIT and is integrated into DCE to provide a secure remote procedure call mechanism.
- The system may be able to identify the originator of any information received over communications channels, or at least be able to determine that no identity is associated with data received. In the latter case, the information may have to be quarantined.
- The system may limit user access to network facilities.
- The system may provide a mechanism for message confidentiality and message integrity. Message confidentiality prevents network eavesdroppers from listening to transmitted messages. Message integrity prevents active attackers from changing and resending messages. Implementing confidentiality and integrity requires the use of cryptographic functions, subject to export and use laws.

12.4.2 Access Control

The following access control features may be provided:

- The host-based access control mechanisms may be extended to principals accessing the system from the network.
- The access rights of a principal may be obtained from a network service, for example, a privileged server, and returned securely to the system that needs to make access control decisions.

  Maintaining semantic integrity of access control or privileged information across platforms is difficult without truly distributed ACLs. OSF/DCE provides a distributed ACL mechanism. Protecting access control information in a distributed environment requires cryptography.

- A capability may be provided where an entity can delegate another entity or service the right to use his verified identity to access yet another service. Access is then granted based on the identity of the original entity, not on the intermediate server. This mechanism is called *proxy functionality*, and is provided by Kerberos Version 5.

12.4.3 Audit and Accountability

The following audit and accountability features may be provided:

- On each system in the distributed environment, the capability of X-AUDIT may be extended to the operations and functions of the distributed components.

  The key issue with audit in a distributed environment is end-to-end accountability. That is, the ability to account for a transaction or other action from one end of the distributed environment (for example, a human user at a workstation) all the way to the final application. If global name spaces are used and clock skews accounted for, end-to-end audit is generally a matter of central collection and analysis.

- Audit records can be transmitted securely over the network to a central audit collection point.

12.5 Applicable Standards

ECMA TR/46 provides a security framework and ECMA-138 specifies data elements and service definitions for security in open distributed systems.

The Internet Engineering Task Force (IETF) is in the process of defining a generic security services API (GSS-API).

No international standards currently exist. When these standards appear, they will probably be based on MIT's Kerberos system, DCE and DME.

Chapter 13

Recommendations

This chapter recommends the use of the X/Open security classes. This does not imply that the recommendations are sufficient for any particular IT environment. The security functionality supplied by open systems is a building block to achieve IT security in combination with other measures. Chapter 2 explains how this can be handled in the procurement process.

13.1 Goal

The essential goal is to achieve security for the IT system and the information processed. The security functionality required in the IT system can be based on the security functionality of the underlying operating system.

The recommendations are presented in a matrix (see Table 13-1 on page 94), where the threats identified for the scenarios in Chapter 3 are related to the functionality classes. Each entry in the matrix refers to a paragraph in Section 13.2 on page 94 which describes how the corresponding functionality class contributes to counter the corresponding threat. If a listed threat is relevant for an application, appropriate security functionality to counter the threat should be chosen.

13.2 Threats and Security Functionality

Threat	Functionality Class					
	X-BASE	**X-DAC**	**X-MAC**	**X-AUDIT**	**X-PRIV**	**X-DIST**
Delay	1			9		
Loss of Information	2	7	8	9	10	11
Work Flow Problems					10	
Repudiation				9		11
Intrinsic Work Problems	3	7	8		10	13
Unauthorised Modification	4	7	8	9	10	11
Disclosure of Information	5	7	8	9	10	14
Incorrect Results	6	7		9	10	13

Table 13-1 Recommendation Matrix

1. X-BASE prevents delay caused by a user, by separating users and confining the effects of actions of one user to that user. The main aspect of preventing disruptions is a matter of the quality of hardware and software, the appropriate use, and the application under appropriate conditions. As far as delay is caused by unavailability of data and programs, X-BASE is capable of avoiding such problems (see 2).

2. The access control supplied with X-BASE may prevent the destruction of files (either by accident or on purpose) if it is used in a reasonable manner. Reasonable means that access rights are restricted (as far as possible) to users or groups, to the access mode, and to the time period that is necessary.

   Access control and audit rely on identification and authentication of users when accessing the system by login. It is the first barrier to prevent unauthorised access to the system and its data. The system keeps the identification of a user during the whole session until logout so that any access to a file or any attempt to do so can be traced back to the originator. The system checks the access rights of a user before he is allowed access to a file in any way.

3. X-BASE helps to avoid problems with the support of routine work by ensuring the integrity and availability of tools supplied in the system.

4. X-BASE provides protection against unauthorised modification of data and tools by restricting access to the system, and provides a basic access control for files in the system. The login with password can be a powerful safeguard, if passwords are used by all users, if good passwords are used, and if the passwords are changed from time to time.

5. X-BASE helps to keep information confidential by basic capability concerning identification, authentication and access control that can permit read access only. A supporting effect is provided by the object reuse capabilities of X-BASE.

6. X-BASE may prevent incorrect results caused by the unauthorised modification of programs. The protection of data stored in the system contributes to the integrity of the result.

7. X-DAC provides the means for granting finer grained access rights than X-BASE. Access rights for files can be granted and revoked on the basis of single users and user groups. X-DAC represents a more flexible and powerful instrument than the discretionary access control included in X-BASE.

   X-DAC is capable of avoiding the loss of information (see 2), of preventing problems with the support of routine work (see 3), of supporting the integrity of tools and data (see 4), of countering unwanted disclosure of information (see 5), and of restricting the possibilities

of incorrect results (see 6).

8. X-MAC provides means for an obligatory rule-based access control. Contrary to the discretionary access control, mandatory access control cannot be deactivated by users. Dependent on the type of policy implemented in X-MAC, a certain type of security policy is enforced by the system, like information flow control or integrity policies. For instance, one policy is characterised by *no writing down* and *no reading up* within a hierarchy of documents and users. X-MAC is useful if the users of a system represent different levels of trustworthiness and the separation between different levels is to be achieved by system security functionality. X-MAC is also capable of separating different organisational units sharing the same IT system.

   X-MAC may be employed to protect information against deliberate and accidental destruction. In this context it contributes to avoiding problems with routine work. It may be employed to protect information in the system against unauthorised updates. However, the main objective of X-MAC is to prevent disclosure of information and to restrict the information flow according to the access policy implemented.

9. X-AUDIT provides means for the logging of user actions and of access to the system and files. The logging of actions of users (such as access to a certain file) enables the auditors to reconstruct the history of events. The detection of unauthorised actions is facilitated. Users can be made accountable for their actions. Proof of illegal activities can be produced. If this is known, it may deter users from illegal activities.

   Automated tools should support the evaluation of audit data to alleviate the task of the auditor. Adjusting audit features according to operational needs is of great importance.

   Audit functionality relies on identification and authentication of users when accessing the system by login (see Chapter 7 on page 69). The system keeps the identification of a user during the whole session until logout so that any actions and attempted actions can be traced back to the user.

   Illegal or inadvertent actions use up resources such as CPU, main storage or peripheral storage, causing unnecessary delay for other users. Audit may deter malicious users from the deliberate destruction of information. The logging data may support the retrieval of lost information, for example, by determination of time and date of destruction, which helps to identify the correct backup. If the security functionality of X-AUDIT is active, evidence of a user's actions is available. This can be used to trace unauthorised activities.

   The deterrent effect of X-AUDIT capabilities supports the integrity of tools and data against unauthorised modification and problems with the support of routine office work. X-AUDIT can make it possible to prosecute the culprit, if sensitive information is disclosed. X-AUDIT can help to explain how incorrect results came from processing.

10. X-PRIV provides means for realising the principle of least privilege. X-PRIV supplies standard privileges which support the distribution and flexible allocation of administrative tasks among a set of users. The goal is to avoid the problem of one privileged superuser.

    When used in a reasonable manner, X-PRIV prevents misuse and abuse of privileges by users, regardless of whether it occurs by accident or on purpose. It limits the damage arising from unwanted use of global privileges. In this sense it is effective against threats like trojan horses, viruses and bugs in programs. In this way it may avoid loss of information, unauthorised modification and disclosure of information. In an office system environment, it represents a reliable basis for the protection of the software and system data which reduces the problems with the work flow and the support of the routine work.

11. X-DIST prevents loss or destruction of information that is in transit from one system to another through the use of error correcting codes and message integrity measures.

    Distributed access control mechanisms can help implement a single access control policy on many hosts from a single access control server. With this capability, policy changes can be applied faster and with broader effect. Thus, information loss due to accidental or deliberate destruction can be more consistently prevented.

12. X-DIST can support third party non-repudiation services that can provide proof that one system sent another system some data. Protocols for non-repudiation systems exist; however, in practice, implementations are limited.

    X-DIST functionality that includes audit mechanisms that can forward audit data to a central trusted repository can provide rudimentary non-repudiation facilities.

13. X-DIST provides for secure flow of information through a distributed application. X-DIST functionality provides message integrity, semantic integrity of access control information, and a method of linking each message with a proof of authenticity. This permits downstream systems to process messages quickly, that is, without complex procedures to determine if the message was authentic or correct, thus preserving the routine work function and helping to ensure correct results are obtained.

14. X-DIST confidentiality capabilities, based on message encryption, prevent unauthorised disclosure of security data or other sensitive information from occurring due to network eavesdropping.

    Use of strong authentication mechanisms prevents theft of passwords which could subsequently be used to gain unauthorised access to systems.

Appendix A

BCS Guidelines

This appendix contains text which is reprinted with the permission of the British Computer Society (BCS).

Note: Because this is an extract from a BCS document, the style of this appendix does not match the style of the rest of this guide.

PROFORMA PROPRIETARY INFORMATION PROTECTION POLICY

1. Philosophy

The ABC Company recognises that the protection of its proprietary information is critical to its survival and competitive market position. Adequate protection must also be given to personal information about the employees of ABC, and to proprietary or personal information that other organisations and individuals entrust to ABC Company.

2. Policy

It is Company policy that proprietary information of all types will be controlled and protected as a vital business resource. This will apply to information owned by, or in the care of the Company.

Information must be classified according to its sensitivity, value and confidentiality. Appropriate protection will be provided for each level of classification against accidental or deliberate loss, theft, misuse or damage.

The only exceptions to this policy are information belonging to others which the Company has agreed contractually to protect in a different manner, Government classified or controlled information which must be handled according to Government regulations, or information which is unclassified and requires no special protection.

3. Scope

The policy applies to ABC Company worldwide, where there is no conflict with local laws. It applies to information stored, processed or communicated in all forms, whether by way of paper, microfilm, microfiche, or in any electronic or magnetic medium. The policy also controls proprietary information not committed to paper or other media, such as ideas, proposals and decisions.

All ABC Company proprietary information as well as other information held in the custody of the Company is covered by this policy.

4. Classifications

There are four categories of proprietary information which are recognised by ABC Company. Information is classified according to its level of importance and the damage which could be caused to the Company through loss or unauthorised or inadvertent disclosure.

4.1 *"ABC Secret"*

The company's **MOST IMPORTANT** information.

Disclosure, loss or alteration may result in **VERY SERIOUS DAMAGE** to the Company.

This category includes the most sensitive or confidential plans, ideas, financial data, research and development and activity and similar information.

The use and distribution of any document or record so classified must be severely restricted to only a few people within the Company who have an absolute *need-to-know*.

4.2 *"ABC Confidential"*

The company's **BUSINESS-SENSITIVE** information.

Disclosure, loss or alteration may result in **SERIOUS DAMAGE** to the Company. This category includes information which is sensitive to the ABC Company and is normally associated with a particular process, project or function. Information so classified should only be distributed on a limited NEED-TO-KNOW basis within the company.

4.3 *"ABC Internal Use Only"*

Company-wide information.

Disclosure, loss or alteration may result in **DAMAGE** to the Company.

This category includes information which can be distributed to any company employee but should not be given to other persons or organisations without the designated owner's specific authorisation.

4.4 *"ABC Personal"*

Information about Company employees and other individuals.

Disclosure, loss or alteration may result in **PERSONAL DAMAGE**, **EMBARRASSMENT** or **BREACH OF LAW**.

This category includes personal data about individuals which may be protected by law, of a descriptive or subjective nature, or of such a nature that a reasonable individual might not want it disclosed, or the designated owner wishes to limit its disclosure.

4.5 *Unclassified Information*

Information will be unclassified if it is publicly available or its disclosure, loss or alteration will cause no damage to the Company.

5. Information Control and Protection

5.1 *Protection*

ABC Company proprietary information is to be used only for authorised business purposes of the Company. Information must be protected appropriate to its assigned classification by all persons who handle, use, or have access to such information.

5.2 *Use*

Company classified documents and other media are to be used only in accordance with this policy, including such activities as their marketing, handling, distribution, electronic transmission, discussion, copying, storage and destruction.

5.3 *Disclosure of Information to Non-Company Parties*

Disclosure of ABC Company proprietary information to non-company parties will be made only with the prior approval of the appropriate Company management representative. Disclosure will be made in accordance with this policy and other related Company policies and standards.

Unauthorised disclosure of the Company's classified proprietary information may result in disciplinary proceedings.

5.4 *Declassification*

The classification of information may be reduced in level or declassified only by the designated owner and, only if the sensitivity, value and confidentiality of the information has been correspondingly reduced since its original classification.

6. Responsibilities Under This Policy

6.1 *Originators of ABC Company Classified Information*

Those who create Company proprietary information but who may not necessarily be designated owners. Responsible for assigning appropriate classifications to information in accordance with owners' requirements.

6.2 *Owners of ABC Company Classified Information*

Designated responsible for ensuring that Company proprietary information will be appropriately classified and declassified, and that appropriate protection will be provided for the information.

6.3 *Custodians*

Those who hold classified information at any stage of its handling and use. Responsible for ensuring the necessary protection of Company proprietary information and/or information classified by third parties entrusted to the Company.

6.4 *Function Managers*

Responsible for adherence to, and support of, this policy and related standards.

6.5 *Employees*

Personally responsible for protecting the Company's classified proprietary information as a regular part of their business processes and work assignments.

6.6 *Company Security Staff*

Responsible for specialist support in the implementation, promotion and compliance monitoring of this policy.

Glossary

ACL
Access Control List — A list used to control access to a file or resource. The list contains the user IDs and group IDs that are allowed access to the file or resource.

API
Application Programming Interface — Function definitions for programmers.

asset
Represents the commercial or any other value of the IT application which includes the information and its processing supported by an IT system.

assurance
The confidence that may be held in the security provided by an IT system.

authentication
The act of verifying the claimed identity of an individual, station or originator.

availability
The prevention of the unauthorised withholding of information or resources.

binding of functionality
An aspect of the assessment of the effectiveness of an IT system, namely the ability of its security enforcing functions and mechanisms to work together in a way which is mutually supportive and provides an integrated and effective whole.

biometrics
Access control technology for positive personal identification. Techniques include recognition of eye blood vessel patterns, hand geometry, palm prints, voice recognition and signature analysis.

BSD
Berkeley Software Distribution.

certification
The issue of a formal statement confirming the results of an evaluation and that the evaluation criteria used were correctly applied.

CF
Circulation folder.

compromising emanation
Unintentional data-related or intelligence-bearing signals that if intercepted and analysed, disclose the information transmission received, handled or otherwise processed by the information processing equipment.

confidentiality
The prevention of the unauthorised disclosure of information.

correctness
A property of an IT system such that it accurately reflects the stated security target for that system or product.

DAC
Discretionary Access Control.

DBMS
Database Management System.

DES
Data Encryption Standard.

effectiveness
A property of an IT system representing how well it provides security in the context of its proposed operational use.

EOF
End-of-file.

evaluation
The assessment of an IT system or product against defined evaluation criteria.

four-eyes principle
Also known as the two-man rule. The task must be performed by two people. This should catch any errors.

functionality class
A predefined set of complementary security functions capable of being implemented in an IT system.

granularity
Relative fineness or coarseness by which a mechanism can be adjusted. In the TCSEC specification, access control to the *granularity of a single user* means that the access control mechanism can be adjusted to include or exclude any single user.

IETF
Internet Engineering Task Force
The protocol, enginering, development and standardisation arm of the Internet Architecture Board (IAB).

The IAB of the Internet Society has become the public domain network standardisation body. It performs a similar role to the ISO Council or the CCITT Plenary.

integrity
The prevention of the unauthorised modification of information.

IPC
Interprocess communication.

IT
Information Technology.

IT security
The state of an IT system, in which the risks existing in the application of the IT system because of relevant threats, are reduced to an acceptable level by means of appropriate measures.

key
A sequence of symbols that controls ciphering and deciphering.

key management
The generation, storage, distribution, deletion, archiving and application of keys in accordance with a security policy.

least privilege principle
Principle that each subject in a system be granted only the most restrictive set of privileges

(or lowest clearance) necessary to perform authorised tasks. Application of this principle limits the damage that can result from accident, error or unauthorised use.

MAC
Mandatory Access Control.

measure
Any provision to counter threats or their consequences.

MIT
Massachusetts Institute of Technology.

object
A passive entity that contains or receives information.

OSF/DCE
OSF's Distributed Computing Environment.

outage
A period of system non-function due to a power supply failure.

outsourcing
The practice of procuring from external sources rather than producing within an organisation.

PC
Personal Computer.

residual risk
The risk that always remains after measures have been implemented.

risk
A measure for the importance or relevance of a threat in a particular environment.

risk analysis
The process by which the importance of identified threats is assessed.

SAK
Secure Attention Key.

SDE
Software Development Environment.

security functionality
The functionality of an IT system which is to counter identified threats.

SGID
Attribute of a regular file indicating that on execution of the program contained in this file the effective group ID will be set to the group ID associated with the file.

strength of mechanism
An aspect of the assessment of the effectiveness of an IT system, namely the ability of its security mechanisms to withstand direct attack against weaknesses in their underlying algorithms, principles and properties.

subject
An active entity, generally in the form of a person, process or device.

SUID
Attribute of a regular file indicating that on execution of the program contained in this file the effective user ID will be set to the user ID of the file owner.

tempest
The study and control of spurious electronic signals emitted by electrical equipment.

threat
An action or event that might prejudice security.

threat analysis
The process in which all relevant threats for an IT system and its application are identified.

trojan horse
A program that apparently performs a useful task but carries out other actions covertly, for example, using the authorisations or rights of the caller to circumvent the security policy.

TCB
Trusted Computing Base.

virus
An addition to a program; the addition has replicating capability. A typical virus has the ability to damage objects, as well as the ability to hide a partial or full copy of itself in one or more unrelated programs. After penetration by a virus, restored assurance of a system's integrity may require analysis or recompilation of all programs changed since the penetration, because they could contain copies of the virus.

vulnerability
A security weakness in an IT system or its application. It may be due to failures in analysis, design, implementation or operation.

Index