

TNO Defence Research

TNO P
Labora

TD 92-3313

Phone +31 70 326 42 21

TNO-report

copy no.

title

EL-92-B365

2

Models and Building Blocks for
Secure Open Systems

①

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

SELECTED
APR 28 1993

TNORAP/9203374

AD-A263 374



TDCK RAPPORTENCENTRALE

Frederikkazerne, gebouw 140
v/d Burchlaan 31 MPC 16A
TEL. : 070-3166394/6395
FAX. : (31) 070-3166202
Postbus 90701
2509 LS Den Haag



TNO Defence Research

TNO Physics and Electronics Laboratory

Oude Waalsdorperweg 63
2597 AK The Hague
P.O. Box 90364
2509 JG The Hague
The Netherlands

Fax +31 70 328 09 61
Phone +31 70 326 42 21

1

TNO-report copy no. title
FEL-92-B365 2 Models and Building Blocks for Secure Open Systems

DISTRIBUTION STATEMENT A
Approved for public release
Distribution unlimited

APR 28 1993

author(s):
P.L. Overbeek

date:
November 1992

Accession For	
NEWS	<input checked="" type="checkbox"/>
DTIC	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	<input type="checkbox"/>

By
Distribution

Approved for Release

A-1

classification
title : unclassified
abstract : unclassified
report text : unclassified

All rights reserved.
No part of this publication may be reproduced and/or published by print, photocopy, microfilm or any other means without the previous written consent of TNO.

In case this report was drafted on instructions, the rights and obligations of contracting parties are subject to either the 'Standard Conditions for Research Instructions given to TNO', or the relevant agreement concluded between the contracting parties. Submitting the report for inspection to parties who have a direct interest is permitted.

TNO

no. of copies : 28
no. of pages : 117 (excl. RDP and distribution list)
no. of appendices : -

All information which is classified according to Dutch regulations shall be treated by the recipient in the same way as classified information of corresponding value in his own country. No part of this information will be disclosed to any party.

93

93-08993
XXXXXXXXXX

Netherlands organization for applied scientific research

TNO Defence Research consists of the TNO Physics and Electronics Laboratory, the TNO Chemistry, Materials Laboratory and the TNO Institute for Research



The Standard Conditions for Research Instructions between TNO and its clients at the Registry of the District Court and the Chamber of Commerce in The Hague shall apply to all instructions given to TNO

report no. : FEL-92-B365
title : Models and Building Blocks for Secure Open Systems

author(s) : P.L. Overbeek
institute : TNO Physics and Electronics Laboratory

date : November 1992
NDRO no :
no. in pow '92 : 709.2

Research supervised by : H.A.M. Luijff
Research carried out by : P.L. Overbeek

=====

ABSTRACT (unclassified)

This report investigates technical security in *open* systems. Security in open systems is a special problem since each element in an open system (hardware, networks, operating systems and applications) must be able to offer security in coordination with other elements. Concepts found in preceding studies are further developed and new concepts are introduced. The concepts of current international standards are generalised such that they are applicable to *open* systems. Furthermore, the different concepts are confronted with one another and unified using new models.

This study offers models and building blocks to create secure open elements that together will make up a secure open system. The study shows that it is possible to offer security services in open systems that are able to fulfil the majority of today's and tomorrow's requirements for security.

The study also shows that there still is a long way to go: standardised interfaces, protocols and data structures must still be defined. However, none of this is beyond the state-of-the-art. Furthermore, the study shows that there is a viable evolutionary path from today's practices and standards towards tomorrow's secure open systems.

rapport no : FEL-92-B365
titel : Modellen en bouwstenen voor beveiliging in 'Open Systemen'

auteur(s) : Ir. P.L. Overbeek
instituut : Fysisch en Elektronisch Laboratorium TNO

datum : november 1992
hdo-opdr no :
no. in iwp '92 : 709.2

Onderzoek uitgevoerd o.l.v. : Ir. H.A.M. Luijff
Onderzoek uitgevoerd door : Ir. P.L. Overbeek

=====

SAMENVATTING (ongecrubiceerd)

In dit rapport worden de mogelijkheden voor technische beveiliging in *open* systemen onderzocht. De beveiliging in open systemen is een bijzonder probleem omdat elementen van een open systeem (hardware, netwerk, besturingssysteem en applicaties) in staat moeten zijn om beveiliging te bieden in afstemming met andere elementen in het open systeem. In deze studie worden zowel modellen uit de voorgaande studies als nieuwe modellen gebruikt. De modellen die aangetroffen zijn in de huidige standaarden voor beveiliging worden gegeneraliseerd zodat ze toegepast kunnen worden in open systemen. Verder worden de verschillende mogelijke benaderingen met elkaar vergeleken en wordt, zo mogelijk, een gemeenschappelijke basis voor de verschillende mogelijkheden gezocht met gebruik van nieuwe modellen.

Deze studie biedt modellen en bouwstenen voor de beveiliging in en tussen de elementen van een open systeem, zodat deze elementen samen een veilig open systeem kunnen vormen. Aangetoond wordt dat het mogelijk is om beveiligingsdiensten in open systemen aan te bieden die in het overgrote deel van de beveiligingsbehoeften van vandaag en morgen kunnen voorzien. De studie laat ook zien dat de weg naar veilige open systemen nog lang is: standaardisatie van interfaces, protocollen en data structuren is noodzakelijk. Geen van deze zaken is echter over de

huidige technologische horizon. Ook wordt getoond dat er een begaanbaar groeipad is van de huidige gangbare praktijken en standaarden voor beveiliging, naar die volgens de in deze studie beschreven modellen voor beveiliging in open systemen.

CONTENTS

	ABSTRACT	2
	SAMENVATTING	3
	CONTENTS	5
1	INTRODUCTION	7
2	THE UNIFORM OPEN SYSTEMS MODEL	9
2.1	Introduction	9
2.2	The tense relationship between the operating system and the network	9
2.3	Is a network really so special?	10
2.4	The seven-layer OSI model	11
2.5	Generalisation of the OSI model: the unifying approach	13
2.6	Description of the layers	16
2.7	New peer-to-peer relationships	27
2.8	Wrap up	27
2.9	Using the Uniform Open Systems Model for security	28
3	A MATTER OF TRUST	32
3.1	Introduction	32
3.2	Dependencies within a system	32
3.3	Dependencies on the security in the environment	34
3.4	Trust assumptions and consequences	35
3.5	Conclusion	42
4	THE DOMAIN MODEL	44
4.1	Introduction	44
4.2	Security domains and security-domain relationships	44
4.3	Security functionality	49

5	VERTICAL SECURITY AND DISTRIBUTION OF TRUST	54
5.1	Introduction	54
5.2	Trusted entities	55
5.3	Use of subdomains	58
5.4	Vertical security services: contribution to security requirements	67
6	METHODS FOR PROVIDING SECURITY	68
6.1	Introduction	68
6.2	Centralised security functionality	69
6.3	Distributed security functionality	74
6.4	Integration of the two methods	78
7	THE SESSION MODEL	82
7.1	Introduction	82
7.2	The stages in a session	83
7.3	Sessions, security domains and horizontal and vertical security relationships	89
7.4	Conclusion	90
8	INTEGRATION OF RESULTS	91
8.1	Introduction	91
8.2	The system: what does it look like?	92
8.3	What is trusted?	93
8.4	Distribution, trust relationships and horizontal and vertical security	95
8.5	Domains in an open system	98
8.6	Sessions	100
8.7	Evaluation of requests and provision of security	102
8.8	What is achieved: potentials and limitations	105
9	CONCLUSION: TOWARDS SECURE OPEN SYSTEMS	107
9.1	Starting point	107
9.2	Future steps	109
9.3	Conclusion	110
10	ACRONYMS	112
11	REFERENCES	114

1 INTRODUCTION

This report offers models and building blocks for technical security in open systems. Security in open systems is a special problem since each element in an open system (hardware, networks, operating systems, database management systems and other applications) must be able to offer security in coordination with other elements.

This report takes the results of earlier studies as a starting point, most notably [40], [34] and [35]. The first study, the report of which is entitled 'Information Security: Past, Present and Future' [40], defines new requirements for security in today's and tomorrow's systems. Security requirements are studied from different angles: openness, organisational structures, the value of information and services, social structures and security of the system itself. The study shows the consequences of the developments in information technology for security. It is shown that these *developments imply shifts in the organisational security, an increasing impact of technical security and a decreasing effectiveness of physical and procedural security.*

Security has become more important and it should be a major design objective for tomorrow's IT products, whether they are open or proprietary. It will no longer be acceptable to develop new IT products first and add on security afterwards. However, one of the conclusions of that study is that information security did not advance as quickly as information technology. If information security does not keep up with the developments in information technology, the practical use of new technologies will be seriously hindered, if not stopped.

The second study is built upon the results of the first study. The main report of this study is entitled 'Secure Open Systems, an Investigation of Current Standardisation Efforts' [35]. The drive towards open systems requires standardisation in security as well. The second study investigates the current standardisation initiatives in the area of technical security in open systems.

Main conclusions of the study are:

- 1 None of the standardisation initiatives addresses all requirements for secure open systems.
- 2 None of the standardisation initiatives gives a solid basis for coordination of security between the elements of the open system.
- 3 All standardisation initiatives disregard the functionality that is needed in the information systems to represent normal organisational structures and responsibilities and the consequential tasks, rights and duties in the systems.

- 4 The standardisation initiatives ignore the needs for security that stem from society.
- 5 The basic security functionality in the systems, as described in the standardisation initiatives, is rather divergent and sometimes conflicting. Emphasis is on prevention, other security measures are neglected to a large extent, and, if addressed at all, lack structure.
- 6 There is a lack of integration between application security, operating-system security and network security. Therefore, an architecture for security functionality is needed that crosses the borders of applications, operating systems and networks.

This report focuses on models and building blocks for the technical security in open systems. It uses both concepts found in the previous studies and new concepts. The concepts of current standards are generalised such that they are applicable to open systems. Furthermore, the different concepts are confronted with one another and are unified in this report using new models.

In section 2 the first model is presented: applications, operating system, network process and hardware are brought together in one unifying model.

The section 3 discusses the implications of the underlying assumptions about security measures in effect, seen in many security standards. The fundamental question is: which elements of a system are assumed to be trustworthy and which are not? And what are the consequences of these assumptions?

Section 4 describes the security domain model. This model is a generalisation of existing models. In a security domain that what can be seen as a whole, seen from both a technical standpoint as well as the standpoint of the security policy, is grouped together. Interactions between the domains can take place. This depends on the specific domain relationships. These relationships can be horizontal (peer-to-peer) or vertical (hierarchical, nested). This gives rise to two kinds of security, namely horizontal and vertical security.

Section 5 elaborates on vertical security. In section 6 the two principal methods for the provision of technical security in a domain are discussed. The Session Model is given in section 7. This model places the provision of security services in the different stages during the life span of active entities and relationships between entities in a domain. In section 8 the models and building blocks of the previous sections are brought together. The resulting composite model both serves as a summary and as a demonstration of how these models fit together.

Section 9, entitled 'Conclusion: Towards Secure open Systems', gives conclusions and proposes a growth path starting from today's systems and models. The last two sections contain a list of acronyms and references to the literature.

2 THE UNIFORM OPEN SYSTEMS MODEL

2.1 Introduction

This section introduces a model to integrate operating system, network process and hardware. First it is shown that the already vanishing differences between communication and information technology must diminish even further. One reason for this is to avoid duplication of services. The distinction between the network process and the operating system is at least partly based on historical and subjective grounds.

This model uses the same layering principles as used in the OSI model. Functions of both the operating system and the network process are grouped together in layers, in the same way as is done in the OSI model. The adjacent communication and operating-system layers are integrated. The result is a uniform model in which both operating-system functions and network functions are integrated.

In the following sections this model will be used as a reasoning background in developing models and building blocks for security in open systems.

This model is based on earlier studies, published as [33, 29, 33]. The reader is assumed to have a general knowledge of the concepts behind the OSI model.

2.2 The tense relationship between the operating system and the network

The development of operating systems for computer systems started in the early sixties, the beginning of data processing by means of automated machines. The developments in data communications started only a little later, but was heavily based on technology for voice communication that was much older. In the next two decades, the developments in data communications (or: network communications) and in operating systems progressed more or less independently of each other without serious interference.

Currently, this separation gives increasingly rise to frictions [40]. Reasons for this are:

- Communication technology tends to become much more information technology orientated. Today's communication technology tends to put more emphasis on the software and less on the hardware.
- Information technology becomes aware of the network and its potentials as the support for distributed computing environments and distributed applications shows.

- As a consequence of the lack of integration between the operating system and the network, application builders have to put a considerable effort in handling network-specific activities and characteristics in the applications themselves.
- Moreover, within a single computer system two different types of services are offered with approximately the same functionality. One type works locally and may be offered by the operating system, the other type works in the network and is usually offered by an application (called network application). Examples of these duplicated services are: services for local and remote (via the network process) storage of information, services for local and remote management and use of files, services for local and remote process scheduling and services for local and remote handling of I/O.
- The management of a growing number of computer systems, each having its own particular type of network software is a growing pressure for the operational staff.
- The users of the computer systems in a network still have to know which resources and services are available locally, and which are located where on which computer system in the network.
- In the drive towards open systems, the relationship between applications, the operating system and the network process must be crystal-clear.

To summarise: the network is not yet truly transparent to users, applications, programmers and system management. The separation between the operating system and the network process causes a duplication of functions, tends to hinder the development of open systems and is a burden for the management.

2.3 Is a network really so special?

The network *is* special since it is treated that way. But is there a fundamental reason for the separation in, on the one hand the operating system and, on the other hand, the network process? We state that this is not the case, which will be shown in the following sections.

In this study it is assumed that the relationships between users and elements in a system are as follows. Human users have access to applications. Applications offer the services that present and give access to information for the users. The applications have access to the information via the operating system. Also, communication with other applications takes place via the operating system. A computer system can be connected to a network using network hardware and software. To activate the network and enable network services, the operating system starts some processes

that are responsible for listening for incoming requests for communication from the network. Similar processes are activated to service requests originating from local applications. In short, the total of these processes is referred to in this study as the *network process*.

The operating system hides configuration and manufacturer-specific characteristics from the applications. The abstraction level is different from that of the applications in the sense that the operating system only handles 'structured data' without knowing its meaning. The network is used when communication between computer systems is required. The network is one of the configuration and/or manufacturer-specific characteristics of a computer system. To make the network transparent to the applications, the configuration-specific characteristics of the network must be hidden by the operating system.

In the next sections a unifying view on applications, operating system, network process and hardware will be presented. This model is built on the following principles:

- The relationships between applications, operating system and network process must be clarified.
- Users do not directly access operating system or network functions but always make use of applications.
- All configuration and manufacturer-specific elements must be hidden for the applications.
- The network can be seen as a shared configuration-specific element of the connected operating systems.
- Duplication of services must be avoided.

To achieve the above characteristics, the model uses a layered structure that is similar to the seven-layer OSI model [7].

2.4 The seven-layer OSI model

In the last decade a lot of energy has been put in defining the network services in a standardised way. For this reason the ISO (International Organisation for Standardisation) developed the OSI model (Open Systems Interconnection model) [7]. The model was given standards at different levels. This model is chosen as a starting point as many of the aspects mentioned above have already been taken care off for the network part.

The seven-layer OSI model was created to allow network communication between computer systems with different architectures. The OSI model is layered since it seems natural to group related functions together.

The layers of the OSI model have been defined in the following way:

- Boundaries are created there where the description of services is both easy and logical and where the number of different interactions between the services is small.
- Similar functions are concentrated in the same layer.
- Functions that are manifestly different in the process of communication or in their technology are located in different layers. This creates layers with a different level of abstraction in the handling of data, for example: morphology, syntax, semantics.
- Changes within a layer do not affect other layers.
- For each layer, boundaries are created only between its direct upper layer and between its direct lower layer (if any).
- The number of layers is kept small.

The OSI model explicitly precludes any involvement with operating systems ([7, page 3]: "OSI is concerned with the exchange of information between open systems (and not the internal functioning of each individual system)" as well as "Any other aspects of systems which are not related to interconnection are outside the scope of OSI". The OSI model seems to regard operating systems, disks and other media as a cloud which interacts with OSI in a way not to be discussed. Nevertheless, OSI defines some standard applications not only to be used for networking. Also, the OSI management services must interact with matching management services of the open systems.

Should one investigate things any deeper, the actual borderline between data communication and other input/output is narrow. Some examples follow to illustrate this.

- 1 Information transport across a wire is called data communication whereas transport of the same information using a tape or a shared disk is not. In both cases one needs matching applications, protocols, device drivers and devices on both sides of the communication.
- 2 A database application accessing a database management system (DBMS) is treated, if the network is used, as a data communication process while the same application-to-application communication on a single computer system is treated as a completely local issue.
- 3 Within one computer system, more than one CPU, servicing multiple processes, may exist. These CPUs and their processes exchange information using a common bus or common memory. This form of data interchange is not considered being 'data communication'.

2.5 Generalisation of the OSI model: the unifying approach

Above, it was shown that in many cases duplicate functionality for local use and for use in the network resides in a single computer system. Since there are so many similarities, the OSI layers are generalised in such a way that their limitation to data communication will vanish. This is done in several steps, starting from the OSI model.

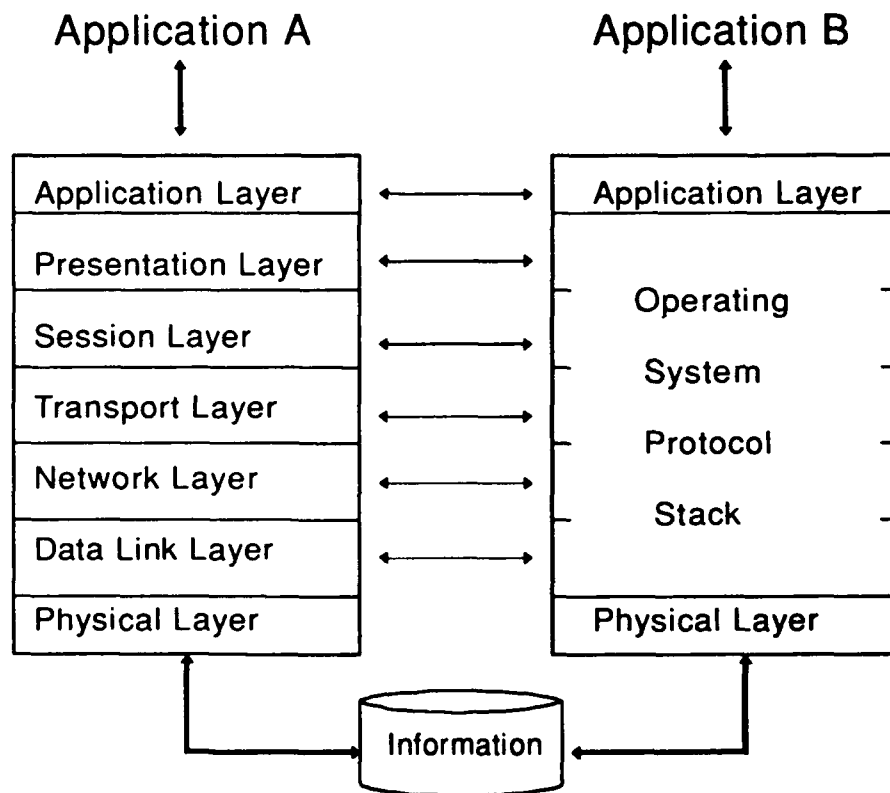


Figure 1: Disk sharing

In figure 1 it is shown how disk sharing between two applications (named 'A' and 'B') takes place in the operating-system layers below the applications (abbreviated as 'OS stack').

In the OS stack the application interfaces can be recognised at the top, and, going down, layers providing for instance encryption, database services, disk striping, device drivers and intelligent controllers, media and equipment.

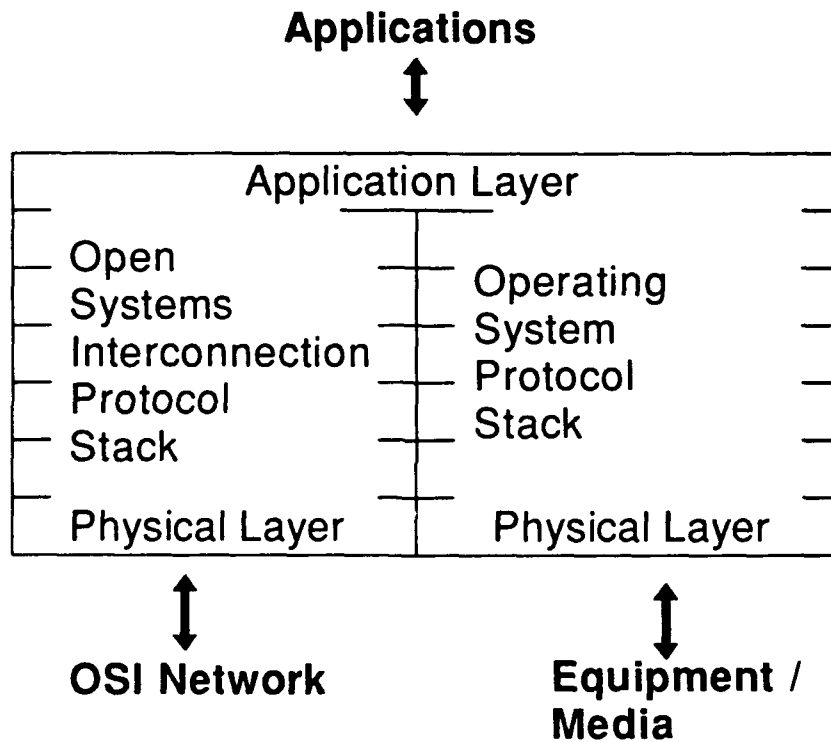


Figure 2: OSI and operating-system stacks united

Now another step is taken. Why should a distinction be made between the operating-system services and the OSI network services? Both types of services are offered to the applications at the top of Application Layer as shown in figure 2. Furthermore, both types of services are normally made available by means of a system-call interface. As layers with the same kind of functionality in both stacks can be recognised, it is obvious to join both stacks into a single new one (see figure 3).

This model is called the *TNO-FEL Uniform Open Systems Model* [33, 29]. *Uniform* because it integrates operating systems, open networks, media and all kinds of information structures in one single model.

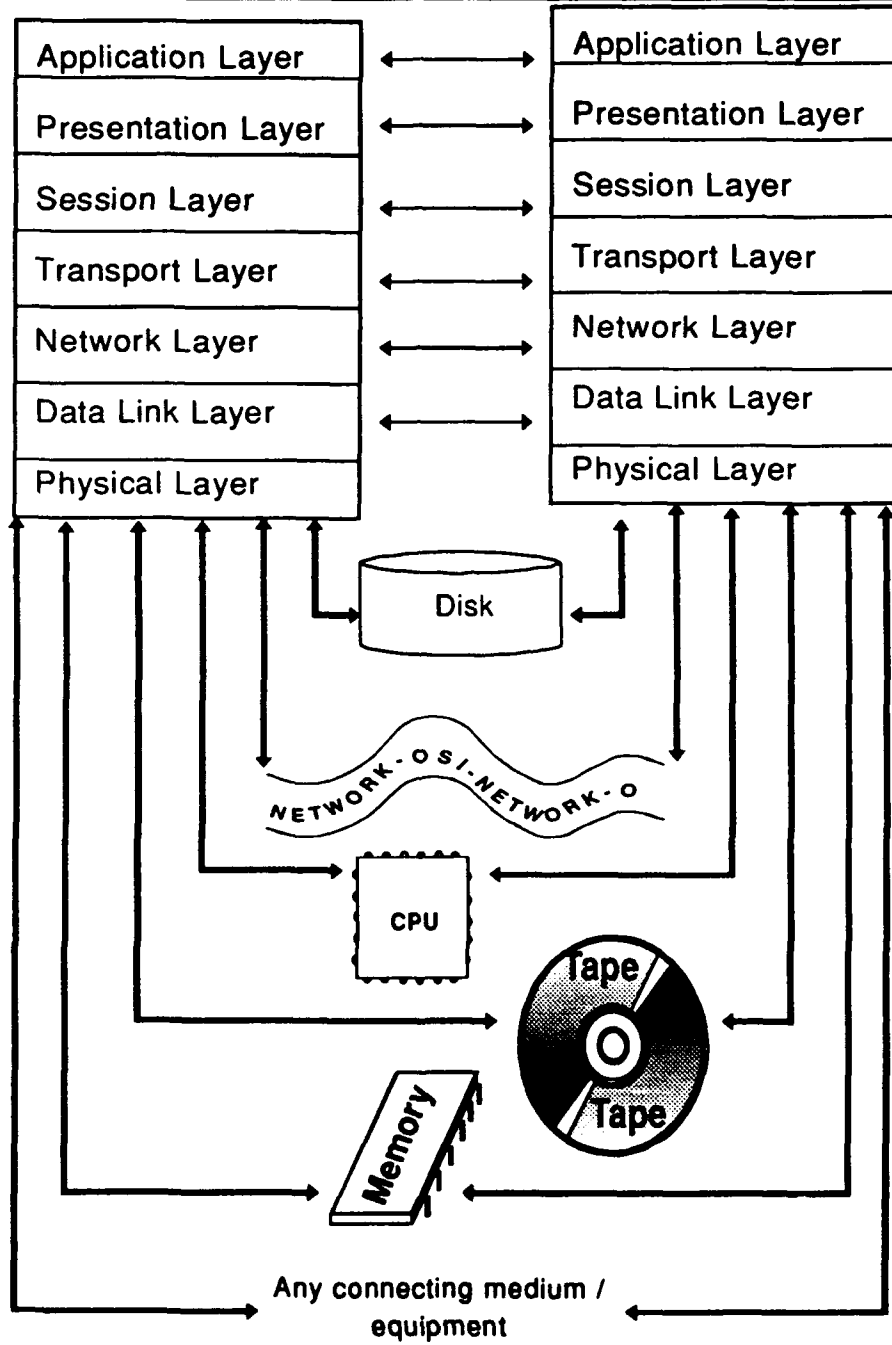


Figure 3: Integration of network and operating-system protocol stacks

2.6 Description of the layers

In this section a generalised description of the OSI layers to cover both the functions of today's operating systems and the functions of the network process will be presented. The following is described for each of the seven layers:

- An outline of the purpose of the layer.
- A description of the services offered by the layer to the layer above.
- A description of the functions provided in the layer and the use made of the services provided by the layer below.
- Examples.

2.6.1 Application Layer

2.6.1.1 Purpose

Being the highest layer, the Application Layer provides the sole means for applications to access information and use resources. The purpose of the Application Layer is to hide all configuration-specific elements, including the network, for applications.

2.6.1.2 Services provided to applications

Being the only layer that directly provides services to applications, the Application Layer necessarily provides all available services, either local or remote. These services may be available to the applications using a system-call interface. Generally, there are two types of service requests: requests for resources (e.g. access to information containers), and requests to access other applications (e.g. processing services and information services).

Examples of services offered by the Application Layer are:

- Handling of all service requests by applications regarding information, be it either access, manipulation, storage or other. The Application Layer, using lower-layer services, takes care of the selection or retrieval of the proper location of the information in the IT environment.
- Handling of all service requests regarding other resources, e.g. start a new process or use of specific processing facilities. The Application Layer, using lower-layer services, relieves the application of the selection of the most appropriate place for the execution of a request (e.g. where a process is started and which special purpose processing device may be used) or the selection of a specific resource out of a pool of similar resources. As stated before, the Application Layer hides the specifics concerning location and other configuration specifics from the application.

- Handling of all requests concerning application-to-application communication. The Application Layer, using lower-layer services, takes care of locating the requested application(s), selection of the most appropriate occurrence of the application (when more than one instance of a specific application is available), management and support of all concerns regarding the communicating applications.

It should be stressed that in this model, *no* applications exist that can only be used for local use or only for networked use.

2.6.1.3 Functions within the Application Layer

One of the major functions in the Application Layer is the hiding of the configuration-specific elements from the applications. The Application Layer functions also offer many of the necessary security functions. The Application Layer makes use of the lower layers to get information about the configuration and its status. As a consequence, the Application Layer is also responsible for locating, selecting and assigning the proper resources and *information containers* (storage capabilities for application-specific units of data).

Furthermore, system-management functions are located in the Application Layer. It is envisaged that specific functions will be located in the Application Layer to satisfy the particular needs of specific applications.

2.6.1.4 Examples

- 1 An application, on behalf of a user, may request that a subprocess is started that is known to require many matrix multiplications. Functions in the Application Layer, in turn assisted by services offered by the lower layers, will make an inventory of the availability of CPU capacity anywhere in the network and may decide to assign the new process either to special purpose processors or to general purpose CPUs using an additional software library. Note that functions within the Application Layer, possibly operating at different computer systems, have to cooperate. Like in the OSI model, these functions are called *peers*.
- 2 An application, on behalf of a user, may request that a new information container will be made accessible. Functions in the Application Layer search for free storage capacity of the requested type anywhere in the network and make the new information container available to the application. Of course, additional (security) checks have to be performed. Furthermore, it is likely that a name will be assigned in the application to the information container. That name must be unambiguous in the context of that specific user and that specific application.

To enable the retrieval of the information container, the functions in the Application Layer will probably assign an additional name that is unique in its IT environment and which remains internal to the Application Layer.

2.6.2 Presentation Layer

2.6.2.1 Purpose

The Presentation Layer takes care of the representation of information to the Application Layer in a suitable structure for the applications. The internal configuration and/or manufacturer-specific structure of the information may be quite different. Furthermore, when application-to-application communication is involved, translations of the syntax, e.g. that of the data structures, may be required.

The Presentation Layer is concerned only with the syntax, i.e. the representation of the data, and not with its semantics, i.e. their meaning to the Application Layer and the applications.

The Presentation Layer takes care of a suitable representation of the data to the applications. This relieves the applications of any concern with the problem of conversion. The applications can use any syntax. The Presentation Layer provides the transformation to the internal representation(s) in the IT environment.

2.6.2.2 Services provided to the Application Layer

The Presentation Layer provides the following services:

- Selection of syntax.
- Transformation of syntax.

Note that there are more possible syntactic versions of the same data: different syntaxes used by applications using the same data and the possible internal representations of the same data.

2.6.2.3 Functions within the Presentation Layer

Beside the functions to support selection and transformation of syntax, the Presentation Layer requires and passes on to the Application Layer diverse services from the Session Layer, most notably session management and data transfer services.

2.6.2.4 Examples

1 Compression

At some systems huge files that are infrequently used and do not require immediate access are stored in a compressed way. On UNIX systems, for example, the so-called *manual pages* (providing on-line help) are stored in this way. The Presentation Layer functions may offer transparent (de-)compression to the applications.

2 Data structures

Applications define their own data structures and build databases. A data structure as defined by an application, for example a linked list, may internally be represented quite differently, e.g. as an indexed file. Furthermore, a huge database, seen by the application as a whole, may be distributed over many data containers. The Presentation Layer functions take care of these syntax transformations.

3 Security services

The Presentation Layer functions offer services related to the confidentiality and integrity of data. In the Presentation Layer the data is available in the structure that is also used by the applications. This allows for security services at the highest level of granularity, up to the granularity of the applications' data structure.

Note that some of the security services, e.g. encryption, can efficiently be combined with the compression services.

2.6.3 Session Layer

2.6.3.1 Purpose

The purpose of the Session Layer is to provide an application with the means to organise and synchronise its activities concerning the use of resources (e.g. access to information and processing capabilities) and access to other applications. The Session Layer removes the differences between the specific needs of the application that issued a service request and the capabilities or appearance of the several configuration-specific resources in use. The Session Layer hides the characteristics and performance of the combined hardware in use from the application.

2.6.3.2 Services provided to the Presentation Layer

The Session Layer provides the following services to the upper layers:

Session establishment

The Session Layer services establish connections between resources and services as requested by the upper layers. Resources include information containers. Services include processing capabilities. Furthermore, Session Layer services start new processes (including the assignment of resources) and create connections for application-to-application communication.

Session management

Session Layer services combine the data units coming from the lower layers and present to the higher layers a smooth and error-free whole as requested by the upper layers. Services include synchronisation (including roll back, transaction isolation and rendez-vous points between applications), sequencing and timely delivery (in real-time environments).

Session release

The Session Layer also takes care of an orderly closure of the sessions: information containers are released, resources are withdrawn and returned to the pool, higher layers are properly informed about the termination.

In section 7 it will be shown which security services can be offered at the different stages of a session.

2.6.3.3 Functions within the Session Layer

The functions of the Session Layer, with the help of lower-layer services, establish, manage and release a session. Furthermore, the Session Layer functions require and transfer services from the Transport Layer, notably data transfer services, to the Presentation Layer.

2.6.3.4 Examples

1 File access

A specific service request from an application may require the opening of a file. The application may carry on and does not necessarily have to wait for the execution of the request. The Session Layer assigns resources (e.g. reserving additional storage capacity) and manages subsequent service requests (e.g. read, write, append data). Note that thanks to the 'hiding' services of the layers, the actual file data may be located anywhere in the environment, even divided among different types of storage capacity such as tapes and disks.

At release time the services of the Session Layer release the unused allocated storage capacity, close the file in an orderly way, apply security services as archiving and checking of the integrity, and informs the higher layers about the termination.

2 Process (login to logout)

In many computer systems, initialisation processes are active, only waiting for users to search for access to the computer system and its IT environment ('login'). After acceptable identification and authentication a new user session is created. Privileges/responsibilities (rights/duties) and resources are assigned to this session. At least one process is started acting on behalf of the user, usually a user interface or command interpreter. The user can initiate activities by passing commands, via his user process, to the Application Layer. As a result, new sessions may be started by functions in the Session Layer. When the user decides to discontinue his work on the system, he issues a command to release the session ('logout') and, as a consequence, Session Layer functions will terminate the user process and possibly other pending processes in nested sessions.

Note that sessions may be nested: a query of a database is a session within the user session. Similarly, the user session may be regarded as a session within the computer-system session spanning the period from boot until shut-down.

2.6.4 Transport Layer

2.6.4.1 Purpose

The Transport Layer provides transparent transfer of data. It relieves the higher layers from any concern regarding the detailed way in which higher-layer service requests are serviced with the available hardware.

The Transport Layer performs a transformation between the data-structure-orientated data units of the Session Layer and higher layers (logical structures in internal syntax) and the structures of the lower layers (the physical structures). The latter structures are determined by the available hardware.

The Transport Layer relieves the higher layers of concerns regarding the underlying hardware. Higher layers do not need the knowledge about what hardware is momentarily available in the IT environment. Higher layers only need to know which services can potentially be offered in this IT environment. The Transport Layer selects the most suitable hardware to service a request. Furthermore, the Transport Layer optimises the use of the available hardware, for example by splitting or duplexing data units to/from the same device and moving data between primary and secondary storage media.

2.6.4.2 Services provided to the Session Layer

Transport Layer services offer transparent (free of errors and in the right sequence) transport of data units to the Session Layer. In doing so, the Transport Layer services hide the characteristics of underlying hardware infrastructure. The services of the Transport Layer perform a transformation between the internal syntax chosen in the Application Layer which is used at the higher layers and the hardware-orientated data structures used at the lower layers. These services are offered to the Session Layer.

2.6.4.3 Functions within the Transport Layer

The Transport Layer has functions to create, manage and release connections between the requests concerning information containers or processing on the one hand and, on the other hand, the hardware devices that are used to service these requests. Furthermore, the Transport Layer has functions that perform transformations between the structures that the assigned hardware can handle and the structures in use by the higher layers (the internal syntax). Transport Layer functions offer transparency of specifics concerning the handling of the hardware to the higher layers. This implies that errors in data units are detected and corrected and that the higher layers are provided by orderly, sequenced and error-free responses to their service requests. Finally, the Transport Layer has functions to optimise the use of the underlying available hardware. One way to achieve this is to use statistical information to pre-fetch physical data structures in anticipation of a continued read request for higher-layer data structures. Note that the Transport Layer must know *which* hardware is available. However, the Transport Layer does not have to know *where* it is located.

2.6.4.4 Examples

1 Request for new file

The higher layers may request that a new information container, e.g. a file of records, is created. The Transport Layer assigns this file, depending on the availability of hardware and the requirements indicated in the request (e.g. regarding the required response times, and integrity of the information container), to one or more specific hardware devices. The Transport Layer translates the stream of records to the device-specific structures, e.g. tracks or memory blocks. Vice versa, the Transport Layer gets the device-specific data units, possibly combines, concatenates or splits up these units and offers a stream of records to the Session Layer. Note that the storage capacity may be located anywhere in the network. In this case, several translations between device-specific structures need to be made during the transport.

Neither the user, nor the application, nor the layers above the Transport Layer need to bother with the exact way a file is stored, e.g. on a local disk, a remote tape, in memory, etcetera.

2 Request for processing power

The Transport Layer will assign a request for a certain operation, e.g. a matrix multiplication, to a special purpose processor or to a set of parallel processors that are able to perform the requested operation in the most efficient way.

2.6.5 Network Layer

2.6.5.1 Purpose

The Network Layer provides to the Transport Layer independence from routing and relay considerations. This includes both local and remote transport of data. Therefore, the Network Layer has knowledge of the 'geographical' configuration of the IT environment, the location of the devices within that environment and the availability of routes that can be used to reach these locations. The Network Layer receives a data unit from the Transport Layer together with the specification of the device for which it is intended. The Network Layer transports this unit to the device. Vice versa, the Network Layer delivers data units to the Transport Layer. Or, to be more precise, to a specific place to that Transport Layer function (entity) that issued the request for the data unit (or otherwise caused the data unit to be produced). For simplicity, the term *entity* is used when referring to a *specific instance* of a function, service, data or other unit in the environment. Furthermore, a data unit may be addressed to Transport Layer entities (Transport Layer functions at a specific place) during the session-establishment stage. The set of these *interrupt* functions, intended to get attention and establish a relationship, is limited. The way in which they can be addressed is standardised. Note that at all layers some of these *interrupt* functions must be available in order to establish connections with higher-layer functions and, finally, applications.

2.6.5.2 Services provided to the Transport Layer

The basic service of the Network Layer is the transparent transport of data units between functions in the Transport Layer and the specific hardware devices anywhere in the IT environment. This service allows the structure and detailed content of the data units to be determined exclusively by layers above the Network Layer. The Network Layer services also mask the specifics that are caused by the physical, geographical configuration of the IT environment.

2.6.5.3 Functions within the Network Layer

The main function in the Network Layer is the transfer function. In most cases this function, offered as a combined effort of Network Layer entities in the IT environment, will take care of physical, geographical transport. In some cases, actual transport is not required since the destination is already reached. In that case, the data unit is transferred to the lower layers (actually delivered to a hardware device) or, vice versa, to the specific entities in the Transport Layer. The Network Layer has access to information about the actual geographical configuration of the IT environment. Functions in the Network Layer maintain information about the hardware devices and the alternative routes that can be used to reach these devices. Further, functions in the Network Layer also must maintain addressing information concerning the entities in the Transport Layer.

Furthermore, the Network Layer contains functions for maintaining a certain quality level, e.g. functions for segmentation, flow control, error detection and correction.

2.6.5.4 Examples

1 File access

The Transport Layer has received a request for a certain file operation, e.g. a write operation. Functions in the Transport Layer specify the exact device. The Transport Layer requests the Network Layer to deliver the data unit to the indicated hardware device. Functions in the Network Layer locate the device and select the most efficient route. Based on this information, the Transport Layer is informed about the most appropriate transformation of the data unit to device-specific structures. The Transport Layer entity at the destination where the physical device is situated, performs the final transformation to, for example, a disk structure. The Network Layer entity that finally delivered the data unit to the device may wait for the result of the action and report this result to the requester.

A request for a read operation is processed in a similar way: the entity that eventually delivers the read request to the device waits for the result of the read operation, which is a data unit in a device-specific way, and transports this result to the Transport Layer entity that issued the request.

2 Processing power

A processor may be regarded as a piece of hardware just like any other piece of hardware. Functions in the Transport Layer select a specific device, in this example a processing unit, to perform a certain operation or set of operations. Network Layer functions transport the implied data unit to the selected processor and await the results of the requested operation(s).

2.6.6 Data Link Layer

2.6.6.1 Purpose

The Data Link Layer provides the functional and procedural means to establish, maintain and release connections and transfer data with the hardware. Each Data Link Layer entity is connected with one specific piece of hardware.

2.6.6.2 Services provided to the Network Layer

The Data Link Layer services offer the Network Layer transparency with regard to the detailed way in which the hardware must be controlled. In this layer, one finds for instance all device drivers for networking, tape units, disk units, memory, printers, measuring equipment, robot arms, real-time devices and other equipment and media. Furthermore, services of the Data Link Layer enable the exchange of data units between entities in this layer at different locations.

2.6.6.3 Functions within the Data Link Layer

The Data Link Layer is responsible for a correct and error-free read/write operation per data unit. The structure of this data unit depends on the device used (frame, packet, sector, tape block, paper punch card, instruction). Standards that appear at this layer are, for instance, the IEEE 802.xx standards and the system interface part of the Small Computer Systems Interface (SCSI).

2.6.6.4 Examples

Generic device drivers are a good example of attempts to avoid duplication of efforts. A SCSI controller, for example, may service both disks and tapes. One SCSI controller is able to service more than one device. These devices are 'daisy chained', creating a striking resemblance to a network.

Another example of a generic device driver is the Postscript driver that is capable of controlling both displays and printers.

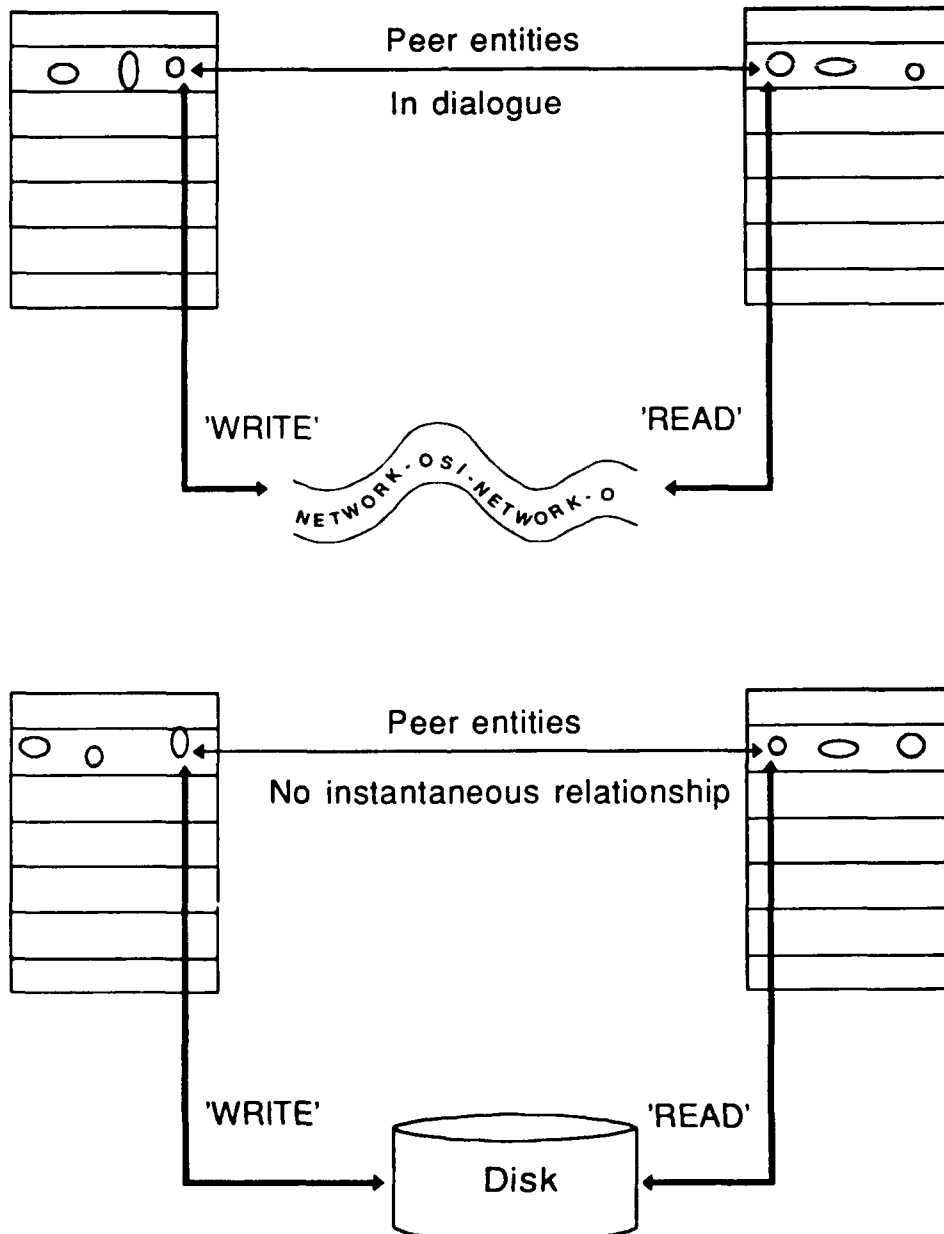
2.6.7 Physical Layer

2.6.7.1 Purpose

The Physical Layer provides mechanical, electrical and functional means to activate, maintain and de-activate the physical hardware. The scope of the Physical Layer relates to bit transfer to and from the hardware.

2.6.7.2 Services provided to the Data Link Layer

The main task of the Physical Layer is to offer a bit stream to the Data Link Layer and, vice versa, to write the bit stream to the hardware.



OSI peer entities often have an instantaneous relationship and are involved in a dialogue with each other. The relationships between the entities in the Uniform Open Systems model may also be diverted in time and do not have to be instantaneous.

Figure 4 New peer-to-peer relationships

2.6.7.3 Functions within the Physical Layer

The Physical Layer translates the bit stream into signals that represent the bit stream. These signals may, for instance, be electrical, magnetic or even mechanical (punch cards). Beside passive components (read many, write once), active components can be found, e.g. to *write* a bit stream to the network that will consequentially be *read* elsewhere. Another example of an active component is a processor moving data between input registers and output registers, while performing operations on this data.

At this layer both the media characteristics (e.g. disk size) and the physical characteristics (e.g. *how* to access the media) are defined.

2.6.7.4 Examples

At this layer, internationally accepted standards can be found as RS-232, X4, DTE/DCE, SCSI, EISA and other channel interfaces, flow control conventions, printer definition languages, coax- and fiber optics standards as well as media standards for tapes, floppies and disks.

2.7 New peer-to-peer relationships

In the OSI model, communication conceptually takes place between peer entities at the same layer. In reality, there is no direct communication between peers (except at the lowest layer). The real communication takes place using the lower layers. During the communication, a connection between the peer entities is maintained and the peer entities can be seen as being in a 'dialogue'. In the model described above, the relationship between peer entities is a little different. A 'write' operation, for example, does not necessarily require one specific 'read' operation at the same time. Or, to put it differently, the peer entities do not have to be involved in a dialogue.

The figure 4 illustrates the difference between these peer relationships.

Note that even in the OSI model, relationships are not necessarily of the peer-to-peer type. Entities of broadcasting protocols at the Data Link Layer, for example, are not involved in a dialogue with their 'listening' counterparts.

2.8 Wrap up

This model offers a unified view on computer and networked systems. The model shows how some of the main concepts of today's computer and communication technology can be brought together.

Although not excluded, it cannot be expected that this model will be the starting point for the design of a new generation of computer/communication systems soon. However, from

experience, it can be expected that this model opens new ways of thinking of how to use IT environments and will prove to be useful in the generation of creative and new solutions to old and new IT challenges.

2.9 Using the Uniform Open Systems Model for security

As a prelude to the following sections, a brief look at how the Uniform Open Systems Model can be used to model security will be given. This section focuses on the relationships between the operating system, the network process and the applications with regard to security. Note: some terms used in this section are explained in detail in later sections.

2.9.1 Today's situation

2.9.1.1 Security in operating systems

In [35] it is shown that for security in operating systems the TCSEC approach [3] is the most commonly used starting point. Although the TCSEC is a set of evaluation criteria for security, it is also used in practice as a set of design criteria. The security-enforcing functions are centralised in what is called the Trusted Computing Base (TCB). All security-relevant information is available in reference monitor databases.

2.9.1.2 Security in networks

For security in networks, following [35], the OSI Security Architecture is a promising starting point (although no full implementations are commercially available yet). All security-relevant information is available in a conceptual Security Management Information Base (SMIB).

2.9.1.3 Security in applications

For security in applications, two main directions currently exist: 'local' applications (not using a network) and networked applications (those that are aware of and/or use a network).

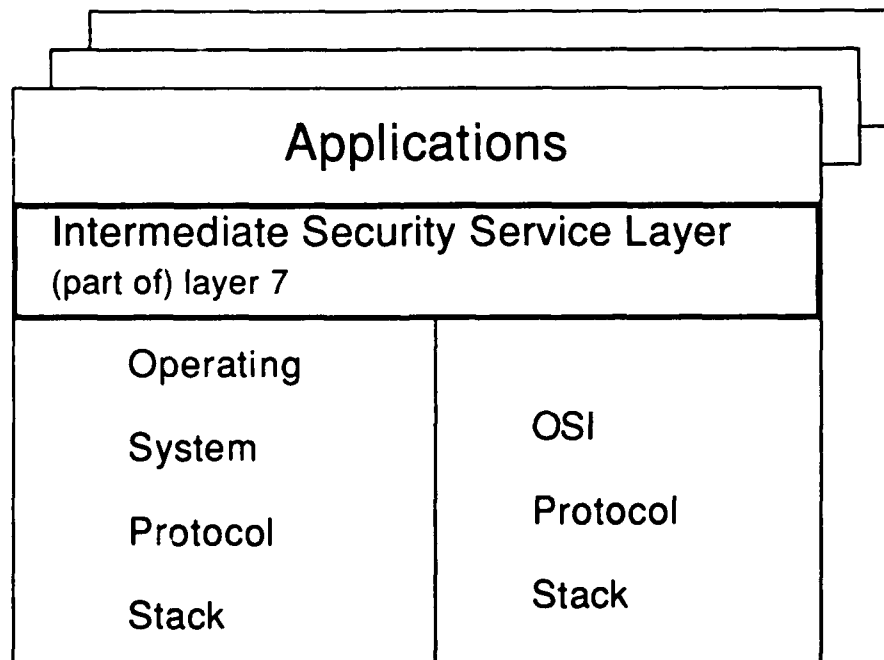
Local applications are under the control of the TCB. The TCB enforces the local security policy as far as possible within the scope of the granularity of the controlled elements (typically, the granularity is a 'file'). When a finer degree of granularity is needed, this must be provided by the application itself.

Networked applications are either not at all or only partly under the control of the TCB.

2.9.2 Applying the Uniform Open Systems Model

According to the Uniform Open Systems Model, the applications must be relieved of any concern with respect to the specifics of the IT environment. This 'hiding' of the specifics of the

configuration is offered by services at layer 7, the Application Layer. As a consequence, being also a benefit, many security services can be offered at the Application Layer as well. This enables a better combination of the 'traditional' operating-system TCB (with its Reference Monitor Databases) and the OSI Security Architecture (with its SMIB). The lower part of the operating-system stack and the OSI stack can remain unaltered. The security services at the Application Layer are concentrated in what is called the *Intermediate Security Service Layer*, see figure 5. This layer is called *intermediate* since it is situated between the applications on top and, below, the operating-system stack and the network process stack. Security functions in this layer mediate all requests stemming from the applications and the network.



The Intermediate Security Service Layer is situated between the applications on top and, below, the stacks of the operating system and the network process.

Figure 5: The Intermediate Security Service Layer

When an application issues a service request to the Intermediate Security Service Layer, information must be available to determine which security domains are involved.

Consequentially, it must be known which security policies have to be applied and which security policy dominates which. This is explained in more detail in section 4. The evaluation of the service request may depend on the user involved, application(s), operating system(s), network process(es) as well as the requested information or service (evaluation of requests and provision of security are discussed in more detail in section 6).

- 1 The user: it may be required to know on whose behalf the request has been issued and in which context. This context may include a persons role in the organisation and the currently adopted role as a user of the system (organisational context). This scope defines a user's specific rights and duties in the system in this role. Note that user authentication is not required when the request can be serviced anonymous, as is the case when the user possesses certain capabilities (e.g. a token or a pre-paid phone card for use of telephone services).
- 2 The application(s): it must be known which application(s) are involved and which security policies apply to these applications. Applications may have specific rights or duties to manage and access specific information. Furthermore, applications may be nested and process to process communication may take place. In these cases, there is more than one application security domain involved and the resulting security policy must be determined.
- 3 The operating system(s): the Intermediate Security Service Layer unites the stack of the operating system and the stack of the network process. Therefore, it must also encompass the traditional operating-system security functions, including the possibility to enforce 'local' TCB-like security policies. When a network is used, a combination of several of these 'local' policies may have to be taken into account.
- 4 The network(s): different security policies may be in effect for subsets within the network.
- 5 The requested information or service:
 - Each requested information item or service may have specific security demands, possibly depending on the location of the requester, e.g. from within a different security domain.
 - The evaluation of the request will not only be based on specific security rights and duties but also on the status and context of the requested service or information.

2.9.3 Extension of Clark-Wilson security policy model

In the Clark-Wilson security policy model [22], access control decisions are based on triplets:

USER/APPLICATION/INFORMATION.

Following the Clark-Wilson model, each evaluation of a request for access to information is based on these triplets.

In the discussion above, some of the concepts of ECMA [15, 16] and MOST [23, 24, 25] are used. In addition to the Clark-Wilson model, evaluation of a request is also based on the network(s) and the operating system(s) involved. Furthermore, the request does not have to be limited to information but may address all kinds of entities: when a request does not involve a relationship with specific information but one with another application or service, this request must be evaluated as well.

The placement of these concepts in the perspective of the ideas of Clark and Wilson leads to a natural extension of their model. Added in the evaluation are: operating system(s), network(s) and, when applicable, services. This leads to the "Extended Clark-Wilson security policy model". In the extended model, the security policy is modelled in the form of quintuples and related security information. The general form of a quintuple is:

USER/APPLICATION(S)/OPERATING_SYSTEM(S)/NETWORK(S)/ENTITY

Readers that are familiar with the Clark-Wilson model may read either:

USER/APPLICATION(S)/OPERATING_SYSTEM(S)/NETWORK(S)/INFORMATION

or

USER/APPLICATION(S)/OPERATING_SYSTEM(S)/NETWORK(S)/SERVICE

All evaluations will be based on these quintuples and the related security information. The scope of the Clark-Wilson model is the modelling of a policy for the control of access to information. This scope limits the usability of their model. Integrity and availability, for instance, can *not* be offered by means of access control only. Now that the scope of the model is broadened, the quintuples can be used for modelling of *all* of the security-relevant decisions, which is reflected in a quintuple's security information.

In their model, Clark and Wilson seem to focus on the non-networked computer systems only. Using the extended model, the networked environment, with distributed evaluation and provision of security, can be modelled as well.

3 A MATTER OF TRUST

3.1 Introduction

Why does an organisation entrust its continued existence to information systems? Because the organisation trusts the systems to behave in an acceptable manner. What do we trust in a system? What *is* trust anyway and where can we find it in a system? A definition of trust will include non-absolute terms as 'belief', 'assume', 'confidence', 'expectation' and even 'faith' or 'hope'. On the other hand, many people define trust in much stronger terms as 'assurance' and 'certainty'.

In [35] it is shown that many of the security models and architectures trust that some level of protection already has been provided. The ECMA model [15], for instance, bases its security on asymmetric cryptography. Asymmetric cryptography requires additional security functionality to maintain the integrity of both the public and the secret key as well as functionality to maintain the confidentiality of the secret key. This, however, lies outside the scope of the ECMA model. Many models are built upon assumptions about the security that is in effect to protect the environment of the model. It is vital that *all* these trust assumptions are made explicit, to make the users of the model aware of the consequences of these trust assumptions. With a better understanding of what can be trusted in a system, a more balanced set of security measures can be applied, tailored to fit the requirements of an organisation and tailored to counteract the threats in the environment of the system.

First in this section, it will be shown how the elements of a system depend on each other for their security. Beside these dependencies in the system, the security of the system depends on threats in the environment. Next the consequences of the trust assumptions, about what can be trusted in the environment and which elements in the system can be trusted, are discussed. All this to show the impact of underlying trust assumptions on the security of the models in their practical use.

3.2 Dependencies within a system

In this subsection, the dependencies between elements in a system for the elements' security are shown. This also has an impact on the security of the information, which will be shown in a

number of cases. This subsection also illustrates how one's perspective on the composition of a system may influence the modelling of security.

3.2.1 Case 1: One view on a stand-alone system

When a stand-alone system is taken to exist of hardware, firmware and software, the following dependencies exist:

- The integrity and availability of the hardware depends on physical, procedural and organisational security measures.
- The security of the firmware depends on the security of the hardware and possibly on additional technical security functions embedded in the firmware (e.g. an integrity check). Since firmware requires little management effort of the users (it is seldom changed) procedural and organisational security are of less importance.
- The security of the software depends on the security of the firmware as well as the technical security functions in the software itself. Note that technical security measures require accompanying procedural and organisational security measures.
- It is clear that the security of the information in the system depends on the security of the combination of hardware, the firmware and the software. Of course, the security of the information also depends on the hostility of the system's environment.

3.2.2 Case 2: Another view on a stand-alone system

Another way to look at a stand-alone system is as seeing it as being a composition of applications, operating system and hardware.

- As in case 1, the integrity and availability of the hardware depends on the physical, procedural and organisational security measures in place.
- The security of the operating system depends on the security of the hardware and possibly additional technical security functions embedded in the operating system itself (e.g. a security kernel that protects access to the operating system itself). The security of the operating system also depends on proper security management, thus on procedural and organisational security measures.
- The security of the applications depends on the security offered by the operating system and the technical security measures in the applications themselves. The applications' security management depends on procedural and organisational security measures.
- The security of the information depends on the security of the hardware, the operating system and the applications in use to handle the information.

Note: cases 1 and 2 both describe the same system but in a different manner. This leads to different views on the dependencies in the system.

3.2.3 Case 3: A networked system

When a networked system is taken to exist of several computer systems (composed of applications, a network process, operating system and hardware) and physical communication lines, the following dependencies exist:

- The security of all physical communication lines depends on the physical, procedural and organisational security measures in place.
- The security of the network process depends on the security of the computer system in which the process is active. Furthermore, the security of the network process depends on internal technical security measures in the network process itself. It may also depend on the security of the network processes at the other computer systems (the peers with which communication may take place). The last implies a dependency on the whole of security measures at *all* computer systems with which communication is possible.
- For the security of the hardware, operating system and applications: see case 2.
- Information: in addition to the dependencies identified in case 2, the security of the information in a networked system also depends on the security of the communication lines and of the network processes. The dependencies identified in case 2 are extended to all computer systems in the network since computer systems in a network may depend on each other for their security.

3.3 Dependencies on the security in the environment

Whether a system can be trusted to handle our information or not does not only depend on the security in the system itself but also on the threats in the environment of the system. Since this environment is changing, the threats are changing. In the cases above, the hostility of the environment is touched upon as an important factor in the security of a system. In general, threats in the environment can be identified as falling into one of the following categories [40, 39, 41, 32]:

Generic threats:

- Acts of God
- Malicious human activities
 - By authorised users
 - By unauthorised users / outsiders

- Human errors; mistakes
 - By authorised users
 - By (unauthorised) outsiders
- Software inadequacies
- Hardware malfunctioning
- Malfunctioning of communication lines

These threats are caused by the following *risk bearers*: factors causing threats that can be managed to reduce risks:

- Environment
- Human beings
- Applications
- Operating system
- Network process
- Hardware
- Connecting media (cabling)

The hostility of the environment is of great importance to the security of the information in the organisation. Most of the models for security described in [35] take a worst case scenario with regard to the users of the system: users cannot be trusted. On the other hand, most of these models ignore threats stemming from, for example, environmental issues, disruptions in power supply or communications, application errors and regular mistakes of user and outsiders.

This leads to an unbalanced and unrealistic threat scenario. For example, in many environments it may be perfectly eligible to work with an operating system like MS DOS. MS DOS does not offer any confidentiality, but in many cases that is not a problem. However, loss of integrity, e.g. by a virus, may cause a serious problem indeed.

With a better understanding of what can be trusted in the system and in the system's environment a more balanced set of security measures can be applied. For example, when the regular users can be trusted, less priority can be given to preventive security measures (less attempts for malicious security breaches) and more emphasis can be put on detection and corrective measures.

3.4 Trust assumptions and consequences

In the previous subsections, it was shown how diverse the assumptions are about what can be trusted. In this subsection, the consequences of assumptions about trust will be illustrated, without

attempting to be exhaustive. The trust relationships are discussed from the standpoint of the owners of the system and the information as well as the users thereof.

The following list depicts the spectrum of possible trust assumptions:

- Owners/users of a system do not trust anything: nothing in the system can be trusted.
- Owners/users trust the integrity of the hardware.
- Owners and users trust the hardware, local boot, operating system and terminal lines. The owners may, or may not trust the users.
- Owners and users trust the hardware, boot, operating system, terminal lines and applications. The owners may, or may not trust the users.
- Owners and users trust the local hardware, boot, local network hardware, local network process and network file server. The owners may, or may not trust the users.
- Users trust their 'private' computer system, and some specific computer systems in the network. The users do not trust the network. The owners trust some computer systems in the network. It is an untrusted network connecting both trusted and untrusted computer systems.
- Owners trust the whole environment. The users distrust the owners and the IT environment.

This list is just an indication of the spectrum of what can be trusted. It is certainly not absolute: a specific system can be trusted to handle information depending on the relative importance of the information. So, in our evaluation of whether we *trust* our assets to be handled in a specific information system, appraisal of the combination of environmental issues, the system (composition, manageability) and the relative importance of specific information or intended activities is of utmost importance.

Some of the cases in the spectrum of trust assumptions are further elaborated upon, this to investigate the consequences of the trust assumptions and the relationships between owner, user and system. It will be investigated if and how the user verifies whether he can trust the system, if and how the system trusts the user and if the owner trusts the users and the system.

Case 1: Users cannot trust the system.

Example: a system that is not managed and placed in a hostile environment, e.g. a campus. The owners of the system regard the system as of less importance.

Trust relationships and establishment thereof:**- User verifies system:**

A (new) user will convince himself of the integrity of the hardware by visual inspection, perhaps bringing his own boot-'dongle' (a dongle is a piece of hardware with some logic stored in ROM) and using special equipment or software to test the system. After convincing himself of the integrity of the hardware, the user will use his own operating system (local boot, e.g. from floppy disk), verify the integrity of the software that is available in the system or use his own applications from floppy disk.

The user of the system is the owner of the information and the results of the activities at the system. After use, the user will not leave anything of value behind.

- System:

The system has no means to control the user.

- Owner of the system:

The owner of the system is not really interested in its use.

Case 2: Owner and users trust a computer system: hardware, local boot, operating system and possibly terminal lines. Some applications can be trusted, others can not.

Example: a properly managed computer system placed in a protected area (e.g. computer room).

The communication between users and system (e.g. terminal lines) is assumed to be safe. All users are employees of the organisation that owns the system and the information.

Establishment of trust relationships:**- System controls the user:**

Two different situations exist, depending on whether the users can be trusted or not.

When the users can *not* be trusted, the operating system and the applications need to protect themselves from malicious users. A user will only have access to information, applications and services of the operating system after positive identification and authentication. The operating system protects itself and the information by a set of security measures, with emphasis on prevention. In addition, the applications may apply a refined security policy and put restraints on the users.

When the users *can* be trusted, the threat of malicious users is low. In comparison with above, security measures aiming at prevention need less emphasis and other priorities are set in favour of security measures for reduction, detection and correction. There is a shift in the security measures from the earlier to the later stages in the event cycle.

- **Trust relationships inside the system:**

At boot time, the integrity and availability of the hardware is verified as far as possible. It is also possible to verify the integrity of the operating system before loading by the boot process. The operating system verifies the integrity of the applications. Applications depend on the operating system for their security.

The operating system must anticipate that malicious applications intend to provoke the system. Therefore, the operating system has to control all irreversible (security critical) activities and apply a balanced set of additional security measures to enable reduction, detection and correction.
- **User trusts the system:**

In the case that the operating system is able to protect itself, it should be possible to have the system provide the user with some evidence of the user's activities in the system. This evidence can take the form of a *voucher* or *receipt*, or a digital form thereof, possibly transferred to the user's smart card. When needed, this evidence can be used by the user as proof of his activities at a later stage. This makes it easier for the user to trust the system, since his vulnerability will be limited.

Such evidence may also be useful in the case that the users are customers of the owner. The evidence should not be limited to activities at the operating-system level only. It may also be useful in many applications.

In the case that the operating system is *not* able to protect itself, the users may choose to check the integrity of the operating system and the applications themselves. The user verifies the integrity of the applications or have the operating system do so. The user may also decide to use his own applications (which he trusts) carried with him e.g. on a floppy disk.
- **Owner does or does not trust the users:**

When the owner does not trust the users, the behaviour of the users must somehow be verified. The owner can do this either by non-technical security measures as procedures and inspection or by technical security measures in the operating system and applications. In the latter case, the owner must be able to trust the system (see: system controls the user).

Examples of technical security measures to verify the users are audit, security monitoring and logging of the users' activities.

Note that even when the owner does trust the users, security measures still are required to protect against normal mistakes and errors.

Case 3: Local computer system, local network and network file server are trusted by the owner and the users.

Example: This is a networked workstation or PC of which the hardware is protected by the user and/or by physical security measures. The operating system and applications are loaded from a file server in the network. Information is stored at this network file server. The network file server is placed in a protected area, e.g. the computer room, and is properly managed on behalf of the owner. The local network can be trusted.

Seen from the standpoint of the user, the whole of local workstation/PC, network and file server create the system.

Establishment of trust relationships:

- System controls the user:

When the users cannot be trusted, the users must be authenticated and controlled. The authentication takes place as a joint effort of both the local workstation and the network file server. The local workstation collects authentication information from the user on behalf of the network file server. The authentication decision takes place at the file server. All requests of a user are transferred to the server where they are evaluated.

Observations:

- Note that the security measures at the local workstation/PC do not have to be as exhaustive as in the previous cases since many of the security measures and services can be offered centrally at the file server.
- Furthermore, note that from a security point of view this situation is comparable with a system consisting of a computer system and connected dumb terminals.
- Trust relationships inside the system:
At boot time, the local workstation/PC may perform an integrity check at the binary image of the operating system, loaded from the network file server. The operating system, once loaded, may perform the integrity checks on the applications retrieved from the file server. Additional security measures may be applied to protect the availability in the system, e.g. of the communication between the local workstation and the network file server.
- User trusts the system:
The user does not have the means to verify the system as a whole. But, since the server is able to protect itself, the file server may provide the user with some 'evidence' so that the user can show proof of his actions at a later stage, as explained at page 38.

- Owner does not trust the users:
When the owner does not trust its users, technical measures at the network file server, e.g. audit, can assist in the controlling of the behaviour of the users.

Case 4: An untrusted network with both trustworthy and untrustworthy computer systems.

Example: The trustworthy computer systems are properly managed and protected, e.g. by its users/owners. These trusted computer systems are connected to a network that is not protected to the same extent. Other users are able to connect (hostile) systems to the network and the network traffic can be monitored, modified or tampered otherwise.

Establishment of trust relationships:

- Trust relationships between computer systems:
In comparison with the previous case, the most important additional problem is the establishment of a trust relationship between the trustworthy computer systems in the network. If it is known which computer systems can be trusted, the computer systems may authenticate each other and exchange credentials (in the previous case, this would imply mutual authentication between network file server and local workstation/PC). If it is *not* known which computer systems can be trusted, preventive measures alone will not be sufficient. More emphasis must be put on security measures for reduction of losses, detection of security breaches and correction of consequential damage.
Note that authentication and exchange of security-relevant information takes place at several peer levels: between the network processes residing in the computer systems, between the communicating operating systems, and, possibly, between communicating applications. After the trust relationship is successfully established, it is also possible to use user-authentication information stemming from one trusted computer system at another. In this way, a user only has to go through the authentication procedure at one computer system.
- System controls the user:
Once a trust relationship between trustworthy computer systems is established, possibly implying secure communications, these systems must work as one (super) system to control the user.
- User trusts the system(s):
The user trusts the (local) computer system he is connected to. He trusts this computer system to establish trusted peer-to-peer relationships with other trustworthy systems. In general, the user has no means to verify the proper behaviour of the system(s).

As in the previous case, one could imagine that the computer system(s) provides for evidence of the user's activities. This, of course, becomes more complex when several computer systems are involved.

- Owner does not trust the users:
On behalf of the owner security measures *per* computer can be applied to verify the behaviour of the users. This requires much labour and does not offer a proper view on the network-wide activities of the users. Therefore, a network-wide auditing facility could be advantageous.

Case 5: The owner trusts the whole IT environment; users do not trust the owner.

Example: An Automatic Teller Machine (ATM) network: a network with a central host and many ATM terminals, owned by a bank.

Establishment of trust relationships:

- Trust relationships in the ATM network:
The ATM terminal is protected against third-party attacks by physical measures, and against insider attacks by a mixture of security measures. The central host is considered to be secure, being protected by a balanced set of security measures. The ATM terminals contain trusted hardware. Authentication of the ATM terminals to the central host, as well as securing the communication lines, is based on cryptographic functions held in the trusted hardware. There is only one application in the ATM terminal which is trusted by the owner.
- The owner does not trust the users:
In order to establish a trust relationship, the owner applies a mixture of security measures. The owner has the ATM terminal demand a PIN code to authenticate the user. The user's credit limit may be checked as well as the user's withdrawal history. Despite all the security measures, the bank still does not trust the users. Therefore, the bank transfers most of the financial risks involving use of the ATM network to its users.
- ATM network controls the user:
As indicated above, the user has to authenticate himself with his magnetic stripe card and the matching PIN code.
- User trusts the ATM network:
The user has no means to verify the proper behaviour of the ATM network on beforehand. Nevertheless, the user does trust the ATM network to some extent (otherwise he would not

use it). Only an alert user is able to detect malfunctioning of the ATM network in a later stage, but he hardly has any means to initiate corrective actions or prove a false withdrawal (phantom or fraud). The ATM terminal can give out a voucher but this voucher serves at best as a means to convince the bank of an apparent failure.

3.5 Conclusion

In the previous paragraphs, the impact of the conceptions about trust was shown. It was shown that:

- Some physical security is always needed.
- Technical security measures require support of procedural and organisational security measures.
- A 'germ' of trust can be used to build upon. This offers the possibility of extension of what can be trusted in a system. Examples are:
 - The operating system depends on the security of the hardware. When the hardware is sufficiently protected, it is technically possible to maintain the integrity of the operating system throughout the system boot process. This offers a basis for a trustworthy operating system (requiring, amongst other, proper technical measures in the operating system as well as procedural and organisational security measures to offer appropriate security management).
 - Applications depend on the operating system for their security. When the operating system can be trusted, it is possible to have the operating system perform integrity checks and protect the working space of the applications in such a way that the integrity of the applications can be trusted (additional security measures in and surrounding the application are required).
 - Trustworthy computer systems in a network are able to create a secure session between themselves, even in a hostile environment.
- The more technically 'advanced' an information system is, the more security measures can be applied. A seemingly contradiction is, that users have *less* possibilities to verify the proper behaviour of the system, while the owners have *more* possibilities to audit the users' activities.

- With a better understanding of what can be trusted in a system a more balanced set of security measures can be applied, tailored to fit the requirements of an organisation and the threats in the environment of the system.
- Many security models are built upon assumptions about the security that is in effect to protect the environment of the model. It was shown that the security of the whole system depends on these trust assumptions, since parts in the system depend on other parts for their security. Therefore, *all* assumptions about the underlying security must be made explicit, such that the users of a model are aware of the consequences of these assumptions.

4 THE DOMAIN MODEL

4.1 Introduction

In this section the concept of the security domain is elaborated upon further. In the foregoing study, see [35], this concept is already briefly discussed. The concept of the security domain is chiefly used by the ECMA [15], but can also be found in the TDI [18] and in the TCSEC [3]. These documents all use a different definition for the notion of security domain. However, the underlying idea is the same: within a security domain one single security policy is applied to a bounded group of objects and subjects. Other terms are *entities* for both objects and subjects (object: an entity in a passive role; subject: an entity in an active role) and *resources* for objects. The bounded group of entities is referred to as the (*security*) *domain set*.

In this section, the two most important domain relationships are described in more detail as well as the impact of these relationships on both horizontal and vertical security, see [34].

4.2 Security domains and security-domain relationships

The security domain is a managerial concept that limits the scope of a particular security policy. A domain covers a bounded group of entities. Entities are often referred to as subjects (entities in an active role, as users, applications, processes, etc.) and objects (entities in a passive role, such as resources, files, media). When a more detailed security policy is to be applied to a group of entities within a domain, a subdomain is created. The entities in a subdomain form a subset of those in the encompassing domain. The security policy of the domain also applies to all its subdomains. A subdomain policy is a refinement thereof, possibly having a finer degree of granularity. For practical reasons, the domains are often chosen in such a way that the entities in the domain set have more or less the same level of abstraction and form a whole from a technical point of view. Examples of possible security domains are:

- An application with all application data.
- A computer system including operating system, applications and information.
- A local network with connected end systems, operating systems, network processes and applications.

In this report, the following definition of a security domain is used: a *security domain* is a bounded set of entities to which a specific security policy applies.

The set of entities consists of:

- Passive entities, also called resources or objects.
- Active entities, also called activities or subjects.

This definition of security domain is derived from both the definition of ECMA in [15] and the definition of the TDI in [18]. The security policy for a domain is implemented by a set of basic security functions and based on domain-specific security information. A specific set of security functions may serve more than one domain, using domain-specific security information per domain.

Two fundamentally different relationships between security domains exist: a peer-to-peer relationships and a hierarchical relationship.

4.2.1 Hierarchical domains

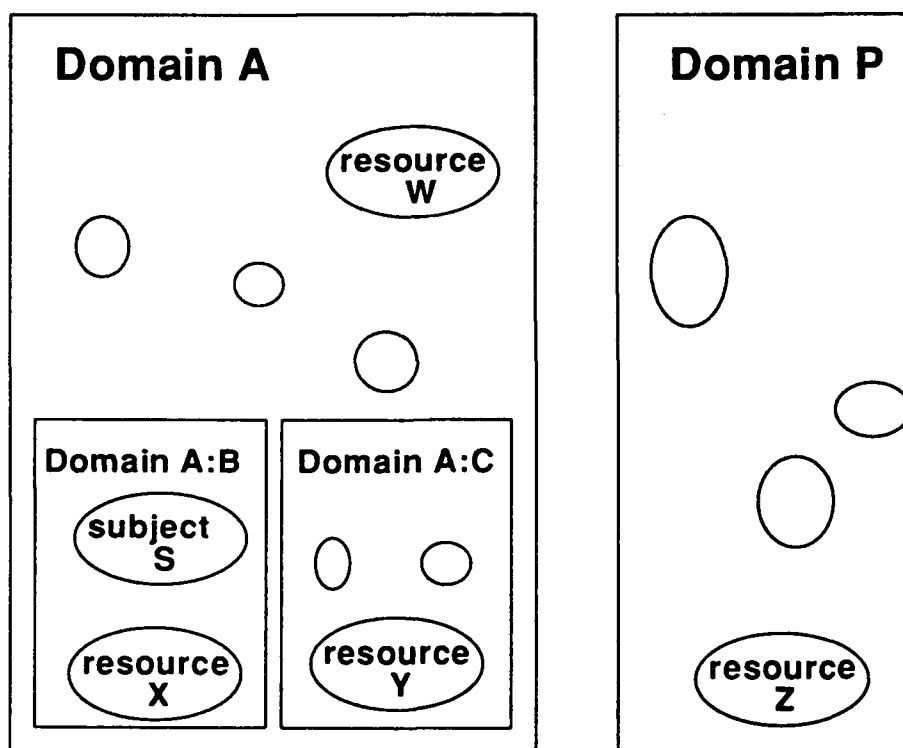
A domain is called a *subdomain* of another domain if the subdomain's set of entities is created from a subset of that other domain's set. The relationship between domains A and A:B in figure 6 is one of domain to subdomain. The set of entities in subdomain A:B is created from a subset of the set of domain A. Following [18], the domain is referred to as the *more primitive* domain of its subdomain(s) (e.g domain A is the more primitive domain of subdomains A:B and A:C). The subdomain is a *less primitive* (sub)domain of the domain it is created from. A domain that is not a subdomain of any other domain is called a *most primitive* domain.

The mapping between the policy of domain A and its subdomains is important. Domain A delegates responsibility for security decisions and provision of security to its subdomains. A subdomain acts autonomously with respect to all policy matters delegated to that subdomain. The security policy of a subdomain should not be in conflict with its domain. It should be a refinement of the domain's security policy with respect to the entities in the subdomain set.

Thus, when subject S in figure 6 requests to establish a relationship with resource W, the domains A and A:B are involved. The subdomain security policy of A:B determines whether subject S may be permitted to establish such a relationships at all. Then, the security policy of A determines whether the requested relationship between resource W and subject S can be established.

Note that within domain A the granularity may be such that not all entities within the subdomain are visible. This is, for example, the case with records in a file: the records may be managed enti-

ties within the subdomain A:B; the file in which these records are contained is a managed entity within domain A.



The notation A:B reflects that domain A:B is a subdomain of domain A. The ovals represent entities. Entity S is in a subject role, regarding some other entities as possible resources.

Activities of subject S will always be controlled by the security policy of subdomain A:B, since subject S is in domain set of A:B. Depending on the intended activity, other domain policies may apply:

- Use of resource X by subject S is controlled by the security policy of domain A:B since both entities are in the set of domain A:B.
- Use of resource W by subject S is controlled by both the security policy of domain A and the more refined security policy of subdomain A:B since W is in the domain set of A and S is in domain set of A:B which is a subset of the set of A.
- Use of resource Y by subject S is controlled by the security policy of the domain that is the least extensive superdomain embracing both domains A:B and A:C, which is domain A, and the more refined security policies of both domains A:B and A:C.
- Since no joint superdomain exists, use of resource Z by subject S will be based on negotiations between the peer domains A and P and on the security policy of subdomain A:B.

Figure 6: Security domains and relationships

Example:

Given:

- A Domain A consists of a computer system, including an operating system which implements the security policy within domain A, several applications, users (subjects, active entities) and files (objects, passive entities).
- B Domain A:B consists of a DBMS which implements the security policy within domain A:B, database users (subjects) and a database composed of many records (the objects).

Observations:

- 1 All policy matters with regard to the security of the records are delegated to subdomain A:B. The record objects are not even visible in domain A. Thus, subdomain A:B is autonomous with respect to the implementation of the security policy concerning the records.
- 2 Should it be possible to access a database file as a whole from within subdomain A:B, this would be subject to the policy of domain A. The subdomain A:B does not have any authority with respect to entities that are not in its domain set.
- 3 Database users may be given rights and duties that are specific to the DBMS and the databases, and do not have any meaning outside the DBMS. Therefore, management of the rights and duties of the database users can be done autonomously from within the DBMS. Since a database user is also a system user, the creation of a new database user implies the creation of a new system user. Therefore, for the evaluation of a request to create a new database user both the security policy of domain A and that of domain A:B will be applied.

To summarise: the two main characteristics of domains with a hierarchical relationship are:

- 1 The set of entities of the subdomain is created from a subset of the domain set.
- 2 Responsibilities are delegated by the domain to the subdomain. The subdomain acts autonomously with respect to the inherited responsibilities. The security policy of the subdomain is a refinement of the domain's security policy.

4.2.2 Peer domains

Another type of relationship between two domains is the peer-to-peer relationship. As examples, see domain A and P in figure 6. In a peer-to-peer relationship, each domain is considered autonomous with respect to the other. The set of entities of domain P and the set of entities in domain A are disjoint, this is: no entity exists in both sets. The entities in both the domain sets have a comparable granularity. Each domain has its own security policy.

Interaction between peer domains can be based on either one of the following:

- A joint superdomain exists.

When an inter-domain relationship has to be established, the policy of a joint superdomain is applied (e.g. domain A in figure 6 for a relationship between an entity in domain A:B and an entity in domain A:C).

- No joint superdomain exists.

When no joint superdomain exists, negotiation between the peer domains has to take place (e.g. for communication between domain A and domain P in figure 6).

The security policy that is applicable to the inter-domain relationship does not bypass the security policies of the subdomains but comes in addition to these policies.

Examples of peer domains

- 1 Two independent computer systems may be peer domains of each other. Each computer system has its own security policy and forms an autonomous domain. When an entity in one of the domains requests to establish a relationship with an entity in the other domain, an inter-domain relationship will be established based on negotiation between the two domains. The means of communication influences the negotiation process. The means may be considered trustworthy or untrustworthy for a certain use. This affects the need for additional security services between the two peers, e.g. encryption of the session.
- 2 Two applications may be peers of another, for example, the database management system (DBMS) and the application of a database user. The database user's application takes care of the communication with the user and the preparation of queries. The DBMS translates the queries in retrieval actions on records in the database files. A relationship between these two applications is established in the same way as in the first example. The communication between the two applications may require the involvement of several domains. For example, when the two applications are entities at different computer systems, the communication requires the involvement of two operating-system domains and the network (either none, one or several domains, maybe trusted, maybe not). The application domains are *depending domains*, since they directly depend on another domain for their security, in this case, on the operating-system domain. It does not make sense for the applications to distrust the operating system. So an apparent option for the application domains would be to request the operating-system domain(s) to offer the required security for their communication. This, of course, in addition to the security offered in the application domains themselves.

To summarise, the two main characteristics of a peer-to-peer relationship between domains are:

- 1 The domain sets are disjunct.
- 2 The security policies in each domain are independent of each other.

4.3 Security functionality

From the previous section it follows that the following types of security functionality may be present in a security domain:

- Functionality providing security within a domain.
- Functionality providing security between peer domains.
- Functionality providing security between domain and subdomains.

These different types of security functions are described in more detail in the following subsections.

4.3.1 Security functionality within a domain

All security matters that arise between entities in the domain set are settled within a domain. There are several ways to achieve this. One way is by mediating all security-relevant actions in the domain. This concept is used in the TCSEC [3] and known as the Reference Monitor. Another method, focussing less on access control and being less biased to prevention, is using a set of cooperating but more or less independent security functions, each function offering its service when implicitly or explicitly requested by the entities in the domain.

More about methods for the provision of security can be found in section 6.

4.3.2 Horizontal security: security functionality between peer domains

Security based on peer-to-peer relations is called *horizontal security*, since the peers are equal to one another and entities in both the domain sets are of the same level of abstraction. When peer domains are subdomains of the same domain, all interactions between the peer domains are subject to that superdomain's security policy (interactions between entities in subdomains A:B and A:C will be subject to domain A's policy). When the domains are fully autonomous, e.g. when no joint superdomain exists, special measures are required that are subject to the security policies of both domains involved. Decisions on requests will be taken in both domains and will be based on *negotiation* between the domains. Therefore, security information must be exchanged between the domains in a structure that is recognised by both domains. This requires standardisation. Some initiatives in this area are already taking place. In ISO SC27, for example, a work item is defined on Security Information Objects (SIOs) [1, 2]. Two well-known examples of

SIOs are the Privilege Attribute Certificate (PAC) [16, 14] and the X.509 certificates for authentication [8, 17, 5]. Another initiative is taking place in the Internet community where a 'Generic Security Service API' is defined [28].

One important aspect about horizontal security between peer domains is that the interconnection itself may or may not be considered trustworthy. This may be the case when two computer systems want to establish a peer-to-peer relationship using an insecure network. On the other hand, two applications at the same computer system may wish to communicate with each other using the services of 'their' trustworthy operating system.

In the establishment of a horizontal relationship between peer entities, the following steps must be taken:

- Establishment of a secure dialogue:
The communication path between the peers must be cleared from unwanted interference, i.e. the communication must be made invulnerable to tampering, tapping or other attacks. This may require services for confidentiality, integrity and availability. In many cases, these services will require the use of cryptographic techniques. The techniques in use must anticipate and resist attacks by forgeries since positive identification has not taken place yet.
- Authentication of the peers:
The peers verify each other's identity. This requires authentication services.
- Negotiations between the peers:
Exchange of service requests between the peers takes place as well as the possible exchange of security information. Both peers evaluate the request individually based on both local security information and security information provided by the other peer. Depending on the type of request, negotiation between the peers may be required. Eventually, as a result of this joint effort, both peers decide autonomously about granting or denying the request.

Note that the decision about whether an autonomous peer domain can be trusted or not is a management issue and not a technical one. Information about which domains can be trusted and to what extent must be available in each domain. It may be necessary to exchange this information between the domains involved by other than technical means. Furthermore, contracts or other legal means, regulating the agreements about trust between domains, may be required in cases where the domains involve different organisations.

Use of peer domains and horizontal security may contribute in offering the following security requirements, as identified in [34, 35].

- Security requirements that stem from organisational considerations:
The domains may be organised in such a way that they map to organisational structures and relationships. For example, computer systems of different organisations, able to communicate through some public network, may be defined as peer domains. Furthermore, users may be grouped in domains according to the users' real-life tasks and responsibilities.
- Security requirements for the security of the system:
The domains are capable of maintaining their own internal security to some extent. In a relationship between peer domains there should be an agreed level of trust. The security in the communication between peer domains is offered as a joint effort of the peers, and is based on horizontal security. The granularity of protection in the system depends on that of the domains.
- Security requirements that regard the value of information:
 - Confidentiality, integrity and availability can be offered within a domain, in sub/super-domain relationship and in a peer-domain relationship.
 - In a domain, all sorts of security measures may be offered (prevention, reduction, detection, repression, correction and evaluation). These security measures may also be offered as a joint effort between peer domains.
 - It is required that an organisation can trust the system as a whole in the way it handles the organisation's valuable information. The domains do not directly influence this trust, but, using the domain concept, the management may become easier (the domains should be chosen in such a way that the entities in the domain set group together in a practical 'natural' way).
- Requirements for security imposed by society:
Anonymity can easily be offered between peer domains. The peers simply do not exchange information about their users (instead, the peers may exchange credentials).
Accountability within a domain does not differ from the way it is done today. In a peer-domain relationship, accountability requires identification and authentication of (either human or logical) users of services by the service-providing domain. Furthermore, a charging method must be agreed upon (probably contractually). Another way of implementing accountability in an anonymous way between peer domains is by using pre-paid 'credit' holders, equivalent to a phone card or digital version thereof.

Different kinds of *proof*, such as proof of a transaction (non-repudiation), proof of proper functioning and legal evidence, can be offered between peer domains. Such a proof may take the form of an officially recognised certificate, either in paper or digital format (see page 38).

In the case of *safety-critical* systems the domain concept can be used to isolate one domain from the other and thus prevent unwanted interference.

Privacy: containers of privacy sensitive information may be separated from one another in different peer domains. Firstly, the security policy of these domains should inhibit the coupling with other domains containing privacy sensitive data and control the flow of sensitive information to other domains. Secondly, special security functions may be in effect in the domain to provide lawful handling of privacy sensitive information. For example, the first time information about a person is registered a letter of notification can automatically be generated.

- Security requirements regarding openness:

An open system may consist of many elements. These open elements must be able to offer security in coordination with other open elements. Use of the domain concept can help here too. By the nature of open elements, the boundaries are clearly defined and their interfaces are standardised. This gives a solid basis to create a domain. An open element may map directly into a domain, or, to put it different, the open element creates a domain. General domain interfaces can be defined such that exchange of security information and the provision of security services between domain and subdomains as well as between peer domains will be possible.

The table 1 (page 53) summarises for which of the security requirements, as identified in [35], use of horizontal security services can be helpful.

4.3.3 Vertical security: security functionality between hierarchical domains

The security that is needed to secure the communication between hierarchical domains is called *vertical security*.

One interesting aspect of nested domains is the possibility to delegate responsibility from the domain to the subdomain. In the same line, the subdomain may inherit some of its security policy from the higher domain. This alleviates the management of these subdomains.

 Table 1: Security requirements which can benefit from horizontal security services

Real-life tasks --> Services	Yes
Responsibilities	Yes
Duties/obligations	Yes
Exclusions	Yes
Control of use of services	Yes, from peer domains
Control access to information	Yes, from peer domains
Authentication	Yes, between domains
Security information	Common security language needed
Security management	Common security management needed
Confidentiality	Yes
Integrity	Yes
Availability	Yes
Prevention	Yes
Reduction	Yes
Detection	Yes
Repression	Yes
Correction	Yes
Evaluation	Yes
Mutual trust	Exchange of 'evidence' between peers
Requirements from society	Yes
Other characteristics:	
Horizontal security is aware of	Some or all peer domains
Distribution of trust	Yes, between peer domains
Trust relations	Yes, with peer domains

Another important aspect of the hierarchic structure between domains is that it enables a domain to arbitrate interactions between entities in different subdomains. This is the case, for instance, for interactions between entities in subdomain A:B and subdomain A:C in figure 6. In addition to the subdomain policies these interactions will be subject to the security policy of domain A. Domain A can provide means to subdomains A:B and A:C for secure interaction between these subdomains. To some extent the subdomains are depending on the domain. Therefore, it is required that the domain provides protection (and other security services) to its subdomains. Section 5 discusses the subject of vertical security relationships in more detail.

5 VERTICAL SECURITY AND DISTRIBUTION OF TRUST

5.1 Introduction

In [35] it is shown that the creation of open systems as a composition of (open) elements requires that the elements of the open system must be able to offer the required security together.

Therefore, there must be some method for the elements to exchange information about the available security functions. Three types of security functions were identified earlier (see section 4), corresponding with as many trust relations:

- *Internal* security functions. These are security functions that are *internal* to the open element. These functions address resources (information containers and active entities) that are internal to and managed by that open element.
- *Horizontal* security functions. These functions address peer-to-peer security between an open element (or an entity) and a peer. These are, for instance, security functions for OSI peer entities.
- *Vertical* security functions. These functions address security of and between open elements that are in a hierarchical relationship with one another. These are, for instance, security functions between an application and the operating system.

The internal and horizontal security functions were addressed in section 4. In this section, we concentrate on vertical security.

In open systems, elements may depend on one another for their security. For example, both the integrity of applications and the integrity of the network process depend on the protection of the operating system. Furthermore, the security of the resources, in use by applications and network services, depends on the protection offered by the operating system. On the other hand, the granularity of the security services of the operating system is limited. The operating system cannot 'see' the refined information structure in, for example, a file.

This shows once again that *one* element of an open system is not able to offer all the required security but that cooperation between the open elements is needed. Furthermore, it shows that open elements may depend on one another for their security.

In this section a model is presented for the delegation of security responsibilities from a domain to its subdomains. The conditions that must be fulfilled and the implications, both for the domain and the subdomain, are discussed as well. This section elaborates on section 4.

5.2 Trusted entities

In [18] the concept of trusted applications is introduced. In this section, this concept is generalised to trusted entities that cover for vertical security.

The concept of trusted entities is simple but needs some introduction:

- 1 Given is the fact that in a system some basic security functionality exists.
- 2 This basic security functionality is used to protect other entities.
- 3 Some entities may be given the predicate *trusted*. A trusted entity is assigned the responsibility to constitute a security policy with regard to a bounded set of entities.

These three items are discussed in more detail below.

- 1 Given is the fact that in a system some basic security functionality exists.
The basic security functionality implements the security policy in its security domain. This basic security functionality can be characterised as follows (see figure 7):
 - The security functions that are provided (either enforced or on request).
 - The resources it is able to protect and the granularity of this protection.
 - The activities it is able to control and the granularity of control.The basic security functionality creates a security domain (see page 45). The security domain consists of a set of entities, encompassing a set of resources and a set of activities:
 - The set of resources (passive entities and active entities) that are protected by a single set of basic security functions.
 - The set of activities (active entities) that are controlled by the same single set of basic security functions.

Note: The set of resources is not limited to passive entities only: one active entity may access another active entity. So, a process, giving access to some service, can be accessed by an active entity as a resource.

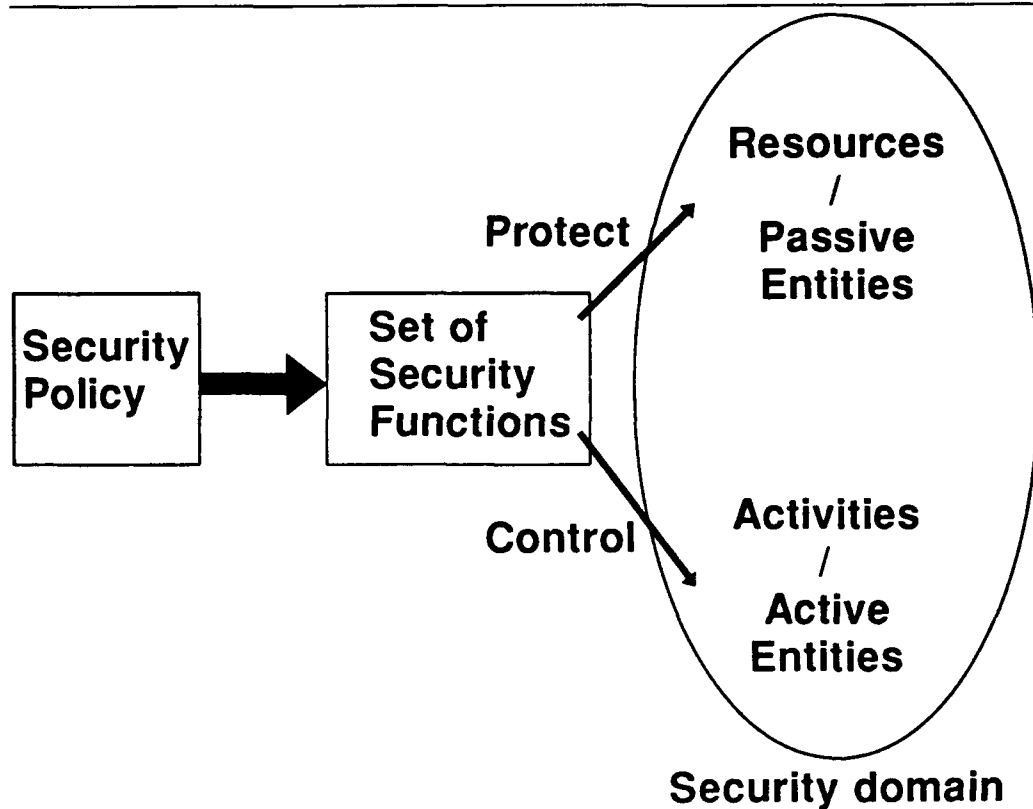
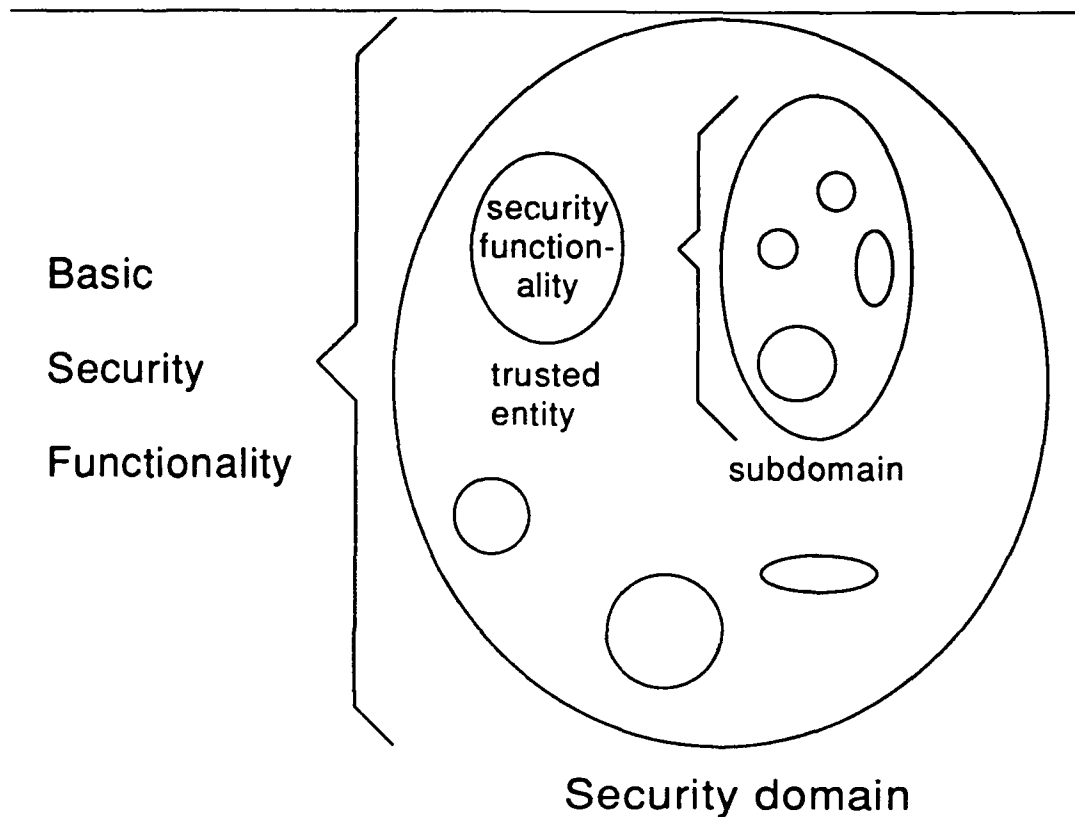


Figure 7: Security policy, functionality and domain

- 2 The basic security functionality is used to protect other entities.
This follows directly from the definition of a security domain: both passive and active entities are protected. For passive entities the protection implies, for example, access control, integrity control and availability control. For active entities the protection includes the protection of the working space and the assigned resources.
- 3 A trusted entity is given the responsibility to constitute a security policy with regard to a bounded set of entities.
A *trusted entity* is an active entity that is trusted to perform and control security-relevant activities to a bounded group of (passive and/or active) entities within a domain (see figure 8). This bounded group of entities forms a subset within the domain. A trusted entity is implementing a security policy with regard to this subset and thus a security subdomain is created. The trusted entity offers all the required security functions and controls *all* the



An entity within a security domain is given the predicate *trusted entity*. This trusted entity implements a security policy with regard to a set of entities, being a subset of the domain set. Thus a security subdomain is created.

Figure 8: Security domain, trusted entity and security subdomain

security-relevant activities within the subset that are *delegated* by the domain. The trusted entity is being resourced by the security domain of which it is part.

Note that the granularity of the entities in the subdomain may be more refined than the granularity of the entities in the domain. This is because a trusted entity may subdivide entities of the domain that are in its own subset into entities having a finer degree of granularity. As a result, the granularity of protection in the subdomain can be more refined.

Summary: Active entities may be given the predicate 'trusted entity'. A trusted entity is given the responsibility for the security within a bounded set of resources. These resources may be regarded by the trusted entity as being composed of smaller entities.

5.2.1 Conditions for creating subdomains

The conditions for creating security subdomains are:

- 1 The subdomains must be properly identified.
- 2 The security policy of the subdomain does not conflict with the policy of the domain.
- 3 The trusted entity and the passive and active entities in the subdomain are and remain part of the same domain.
- 4 A newly created entity will be a member of the domain set it is created in. The domain sets are *closed environments*: import and export of entities is controlled by the basic security-providing functions. For a subdomain, this control is performed by the trusted entity.
- 5 Subdomains do not overlap.
- 6 The domain protects the subdomain.
- 7 The domain does not interfere with the subdomain with regard to the responsibilities that are delegated by the domain to its subdomain.

These conditions are derived from the conditions in the TDI [18].

Note: A security domain that is not protected by a superdomain is a most primitive domain (see section 4.2.1). Such a domain uses only hardware resources that have to be protected by non-technical security measures, e.g. physical security.

5.3 Use of subdomains

The practical use of subdomains is illustrated by two different relationships between (open) elements: the operating system (OS) and a Database Management System (DBMS); the operating system and the network process.

5.3.1 The operating system and the database management system

Assume a Database Management System (DBMS) is an application using database files in one specific directory. The DBMS is installed as a trusted entity in the domain set of the operating system (OS). It is trusted with regard to all security-relevant activities concerning the database files in that specific directory as well as that directory itself. So, the responsibility for the security of the database files and the contents of the directory are delegated by the operating-system domain to the DBMS subdomain.

First, the internal security functionality in both the DBMS subdomain and the operating-system domain are discussed (as far as it is relevant to this specific relationship). Next, the vertical secu-

urity functions offered as a joint effort of security services in the DBMS subdomain and the OS domain are discussed. Note that horizontal security is not applicable in a hierarchical situation.

5.3.1.1 Internal security functions

5.3.1.1.1 Responsibilities of the security functions of the operating system are:

- 1 To protect the integrity of the DBMS binary as a passive entity. The DBMS binary is a file which is a normal passive entity within the security domain of the OS.
- 2 To protect the DBMS running as a process. The security functions of the OS protect the processing environment of the DBMS and the assigned resources (the DBMS's working space). The DBMS process is a normal active entity within the security domain of the OS.
- 3 To prevent any actions that are *not* initiated by the DBMS with regard to the database directory and the files therein. All actions regarding these files and that directory must be controlled by the DBMS.

Further, when requested by the DBMS, the operating system will provide resources to the DBMS (of course, such a request has to be evaluated in the OS security domain as well). These resources will be controlled by the security functions of the DBMS. Until released by the DBMS, they will not be controlled by the security functions of the OS security domain.

5.3.1.1.2 Responsibilities of the security functions of the DBMS are:

- 1 To protect the database directory and the files therein.
- 2 To protect entities having a finer granularity contained in the database files. The database files may contain tables, records, links, pointers or other structures. It is possible that these more refined entities not even have any meaning outside the context of the DBMS.
- 3 To control all activities within the DBMS domain.
- 4 To protect and manage its own internal resources.

5.3.1.2 Vertical security functions

It was shown how the DBMS depends on the security provided by the operating system. Among other things, the DBMS depends on the security of the operating system to maintain the integrity of the DBMS binaries and to protect the resources in use by the DBMS. Thus, the DBMS depends on the proper functioning of the internal security functions of the operating system. On the other hand, the fulfilment of the security policy that must be implemented by the operating system partly depends on the proper functioning of the internal security functions of the DBMS. For, although the DBMS is a trusted entity, the DBMS can easily frustrate the policy of the security domain of the OS, for example by giving illegitimate access to entities in the DBMS subdomain.

So, the proper functioning of internal security functions not only affects the security within a domain but also affects the security of both higher and lower domains (super- and subdomains). Many security services can be implemented in either the operating system or the DBMS. However, some services must be offered in conjunction and for some services it would be highly impractical or lead to duplication when offered by one of the entities alone. Starting from the list of general security requirements defined in [35], examples of vertical security services offered as a joint effort by both the operating system and the DBMS are given below.

Information labelling and flow control:

Services for information labelling and flow control may support both the confidentiality and the integrity of information.

The operating system and the DBMS should understand each other's way of labelling of information containers that are transferred between these two (or even better: use the same labelling scheme). Of course, the treatment of an information container with a specific label should be equivalent. An SQL request, for instance, may be sensitive. It should be treated with the same care by both the DBMS and the operating system. The same holds for the results of a query.

Furthermore, when an information container is transferred by the DBMS to the operating system, the operating system must take over the control of the information flow.

Identification and authentication:

It is highly impractical if each application has to implement services for identification and authentication. This does not even take into account the fact that the overall security should hardly benefit from duplication of these security services in each of the application subdomains: these subdomains depend on the operating-system domain for their protection anyway.

Anonymity:

A user may be identified at the operating-system level but his identity does not necessarily have to be passed on to the DBMS. In this way the user remains an anonymous wanderer through the information accessible by using the DBMS (accounting of an anonymous DBMS user can be performed as a service of the operating system on a lump sum basis).

Audit:

A separate audit function in each application would be highly impractical.

Non-repudiation and notarisation in applications:

This service can only be offered with the support of the operating system. Firstly, authentication, which is required for non-repudiation services, is not a service of each application but either a central service or a service of the operating system itself. Secondly, the results of the non-repudiation service must be protected. This requires either a trusted third party or local protection of the results. In both cases the services of the operating system are needed.

Accounting:

A separate accounting function in each application is not always needed.

Availability:

Services to support availability in applications can only be offered in coordination with corresponding services offered by the operating system.

Integrity:

Coordination of integrity services in applications and the operating system can avoid duplication of efforts.

Confidentiality:

This also holds for confidentiality services: coordination of these services in applications and the operating system can avoid duplication of efforts. As an example: encryption and key management services can be made available to all applications using the operating system.

5.3.2 The operating system and the network process

The network process is an active entity within the domain set of the operating system. Resources are assigned by the operating system to the network process. The network process itself is responsible for the management and security of these resources.

In this example it will be investigated to what extent the network process and its resources can be regarded as a security subdomain (the network subdomain) within the operating-system security domain, following the conditions in section 5.2.1 derived from the TDI [18]. Then, the network process is a trusted entity within the set of the operating-system security domain. The responsibility for the security of all network activities is delegated to the network subdomain. First the internal security functionality in both the network subdomain and the operating-system domain are discussed (as far as relevant in their relationship). Next the vertical security functions, offered as a joint effort of security services of the network process and the operating system are discussed (horizontal security is not an issue in a hierarchical domain relationship).

5.3.2.1 Internal security functions

5.3.2.1.1 Responsibilities of the security functions of the operating system are:

- 1 To protect the integrity of the network software as a passive entity. The network software is stored in files being normal passive entities within the security domain of the OS.
- 2 To protect the active network process and its working space. The security functions of the OS protect the processing environment of the network process, being a normal active entity within the security domain of the OS.
- 3 To prevent any activities that are *not* initiated by the network process with regard to the resources that are assigned to the network process. All activities regarding these resources must be controlled by the network process.

Furthermore, when requested and after proper evaluation of such a request, the operating system will provide resources to the network process. These resources will be controlled by the security functions of the network process. Until released by the network process, they will not be controlled by the security functions of the OS security domain.

5.3.2.1.2 Responsibilities of the security functions of the network process are:

- 1 To protect and manage the resources assigned by the operating system.
- 2 To protect and manage the entities created by the network process. For instance, memory blocks (assigned by the OS as a resource) can be subdivided by the network process in several buffers containing information in frames, packets, etcetera, each having a different sensitivity level. These entities may have a finer degree of granularity than that in the OS domain. These refined structures do not necessarily have any meaning outside the context of the network process.
- 3 To control active entities within the network subdomain, addressing other active entities or the passive entities mentioned under 1 and 2.

5.3.2.2 Vertical security functions

The network software, and thus the network process, depends on the security of the operating-system domain for its integrity. The security of the resources that are assigned to the network security subdomain depends on the security offered by the operating system as well. Thus, the security of the network security subdomain depends on the protection offered by operating-system security domain. On the other hand, the realisation of the security policy of the operating-system domain partly depends on the network subdomain. The network process can, for example, spill resources and thus undermine an availability service, or can frustrate an information labelling policy and thus undermine the information flow control. This demonstrates that the

proper functioning of the internal security functions in a domain influences the security in the subdomains and *vice versa* that of a subdomain affects the security of the domain.

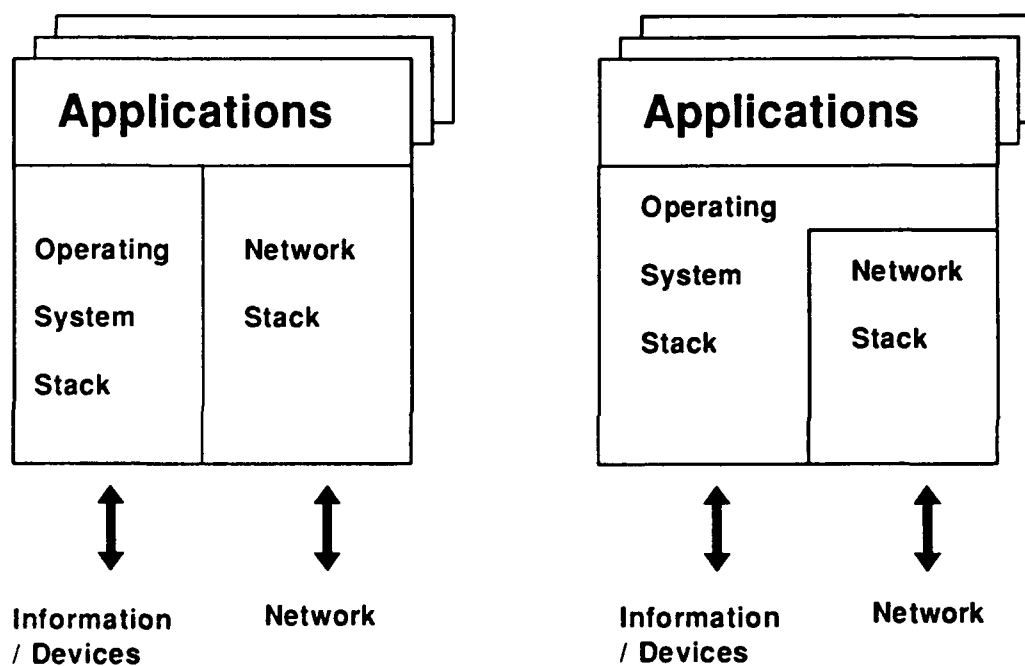


Figure 9: Two views on network use

Most security requirements cannot be offered as an exclusive effort of security services within the network process. To investigate vertical security in the relationships between the network subdomain and the operating-system domain it is relevant to distinguish two views on the relationships between applications, operating system and network (see figure 9).

- 1 Applications communicate with the network services (e.g. services at layer 7 of the OSI model) autonomously. The distribution and information flow takes place outside the control of the operating system (left part of figure 9).
- 2 Applications use the operating system to activate and use network services (right part of figure 9).

In the left view of figure 9, applications may communicate directly with the network (these applications are referred to as *network applications*). Like all applications, these network applications use resources and information assigned by and available through the operating system. A conflict with the security policy in the operating-system domain can only be avoided when network applications are both trustworthy and constitute the same (or: a non-conflicting) security policy as in the operating-system domain. This implies that all applications using network services must be trusted entities within the operating-system security domain. Moreover, all trusted network applications must offer network security services in accordance with comparable services in the operating-system domain (e.g. authentication). This leads to duplication of services and risks of non-conformity.

Following this view, the management of security will become very complex and a lot of the responsibilities for security will be put on the user. For situations requiring more than a minimum level of security this does not seem a very realistic approach to reach the required security in an efficient and effective way.

In the right view of figure 9, network applications use the operating-system to communicate with the network. Network applications do not have to be trusted entities since they are under the control of the security functions of the operating system. However, the security functions of the operating system must be extended to address specific network security functionality. Most of the network security services at the level of applications and the Application Layer are already available as a *local* variant of the security service (see section 2). The local and the network variant of a security service can be integrated into one service. In this way the difference between 'local' and 'remote' will diminish.

Most security services in networks are only useful in conjunction with matching security services of the operating system(s), thus, as vertical security services. As an example, the security services as defined in the OSI Security Architecture (OSI SA) [6] will be used to discuss which of these services require a matching service in the operating system. Or, to put things differently: which of the security services must be implemented as vertical security services and be offered as a joint effort by both the operating system and the network process.

Authentication services

The authentication service of the OSI SA validates a claimed identity of (communicating) peer entities and/or the claimed network-addresses (sources) of information in the network subdomain. The authentication services of the operating-system domain have to add authentication of the (remote) processes, users and passive entities to give the whole authentication process any value.

Access control services

The access control service of OSI SA provides protection against unauthorised use of OSI resources, which are entities in the network subdomain. The operating system must control the access of applications and users to local entities, including the local network process.

Confidentiality services

OSI SA defines several services for confidentiality. These services are only of value when the required confidentiality is also offered at the end systems. Furthermore, the classification of information is determined at the end systems. Therefore, the confidentiality services in the operating-system domain and the network subdomain must use the same or an invertible labelling scheme.

Integrity services

The same discussion holds for integrity services: OSI SA defines several services for integrity. As with confidentiality services, the integrity services are only of any value when the demanded integrity is also offered at the end systems.

Non-repudiation and notarisation services

The OSI SA non-repudiation services provide proof of delivery and proof of origin (the source) of a data unit. A non-repudiation service in the network can only prove that a specific transport of data has occurred between two identified partners. Such a proof does neither say anything about the person involved nor about the concerning applications.

Without the possibility for proper interpretation such proof hardly has any value. The operating system, possibly together with the concerned application(s), must provide non-repudiation services as well, to give a network non-repudiation service any value.

The non-repudiation services in the operating-system domain, the network subdomain and the application subdomain(s) must jointly gather non-repudiation information. For example: information that serves as evidence of a certain transaction (e.g. transfer of an amount of money) issued by an identified individual (e.g. a specific bank employee), from one account to another account and one bank to another.

Security Management services

In [35] it is shown that the management of security in the network process and that of the operating system must be coordinated to be able to constitute non-conflicting security policies in the operating-system domain and the network subdomain.

The conclusion is that almost all OSI SA security services in the network subdomain require accompanying services in the operating-system domain. Thus, it can be observed that almost all these services are vertical security services, jointly offered by the operating system, the network process and, possibly, (trusted) applications.

5.3.2.3 Limitation

Many vertical security relationships exist between the operating system and the network process. So, returning to the question at the beginning of section 5.3.2, "to what extent can the network process and its resources be regarded as a security subdomain (the network subdomain) within the operating-system security domain?", the following can be concluded. Yes, in many respects the network process can be seen as a security subdomain of the operating-system domain. However, there is one crucial limitation: the network security subdomain is not a closed subdomain (see section 5.2.1). Entities enter the network subdomain from other parts of the network, not controlled by the (local) operating-system domain. Furthermore, entities leave the network domain where they are outside the control of the hierarchical domains. So, this approach conflicts with conditions 2, 3 and 4 as stated in section 5.2.1 (see page 58).

The conclusion is that, although there are many aspects to it, the network process can *not* be seen as a security subdomain of the operating system, following the conditions referred to above. In sections 4.2.2 and 4.3.2 horizontal security functions have been introduced. In sections 6 and 8 it is shown that the above conditions can be relaxed, using a combination of vertical and horizontal security.

5.4 Vertical security services: contribution to security requirements

The table 2 summarises for which of the security requirements as identified in [35] use of vertical security services can be helpful.

Table 2: Security requirements which can benefit from vertical security services

Real-life tasks --> Services	Yes
Responsibilities	Yes
Duties/obligations	Yes
Exclusions	Yes
Control of use of services	Yes
Control access to information	Yes
Authentication	Yes
Security information	Common security interfaces needed
Security management	Common security management needed
Confidentiality	Yes
Integrity	Yes
Availability	Yes
Prevention	Yes
Reduction	Yes
Detection	Yes
Repression	Yes
Correction	Yes
Evaluation	Yes
Mutual trust	Not directly
Requirements from society	Yes
Other characteristics:	
Vertical security is aware of	Subdomains, possibly superdomain
Distribution of trust	Yes, between domain and subdomains
Trust relations	Yes, between domain and subdomains

6 METHODS FOR PROVIDING SECURITY

6.1 Introduction

From [35] it can be concluded that only two fundamentally different methods for providing security in a domain exist. The first method is based on centralised security functionality and comes from the TCSEC [3]. The second method uses decentralised (distributed) security functionality and is used by the ECMA Security Framework [15].

One of the conclusions in [35] is that there is a lack of integration between security in applications, security in operating systems and security in networks. To enable the integration of these different types of security, one of the problems to be solved is that of cooperation between the two different methods for providing security.

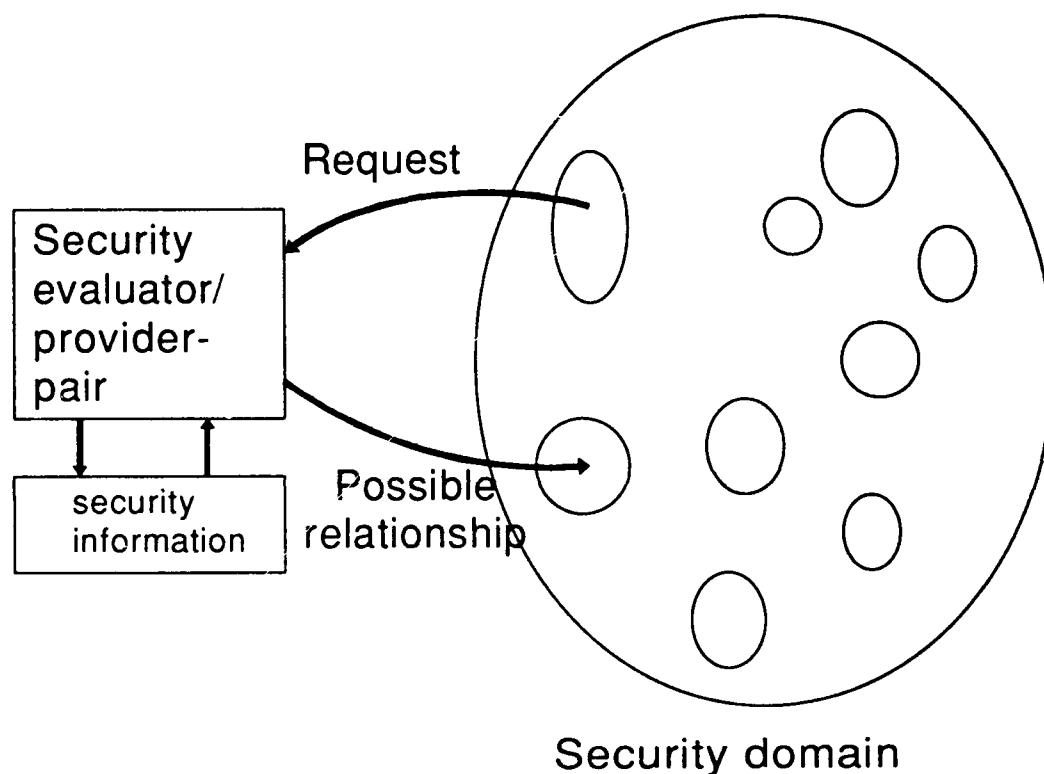
First in this section, a generalised description of the two methods for providing security is presented. Next, a model is proposed in which the two methods are integrated. It is also shown how the two methods relate to vertical and horizontal security.

In the generalised description the following method of modelling of activities in a system is used.

- 1 An entity issues a *request* for a (change of a) relationship with one or more other entities.
- 2 This request is evaluated by (one or more) *security evaluator(s)*. A typical outcome of an evaluation is that the requested relationship is either permissible or not.
- 3 Based on the result of the evaluation, one or more security functions are activated, providing the required security. The activated security functions are called the *security providers*. A typical action of a security provider is the enforcement of the outcome of the evaluation, which results in either the establishment or the denial of the requested relationship.

In section 7, modelling the life span of active entities and relationships between entities (these life spans are called *sessions*), it will be shown that a request may result in the establishment, change, or release of a relationship between entities. Within the scope of this section it suffices to focus exclusively on *new* relationships.

6.2 Centralised security functionality



All entities (represented by the smaller ovals) within the security domain (big oval) are controlled and protected by one security evaluator/provider-pair

Figure 10: Centralised security functionality

6.2.1 Description

Figure 10 shows the centralised security functionality. This figure is derived from the evaluation and enforcement by the TCB as known from the TCSEC.

An active entity, for example a process, issues a request for a relationship with one or more other entities. In the TCSEC, such an entity is said to be in its *subject* role. The addressed entities may be passive (e.g. files) or active (e.g. other processes). In the TCSEC, the addressed entities are called the *objects*. The request is evaluated by the security evaluator/provider-pair that also carries out the decision and provides the required security. Each relationship between entities is established via the evaluator/provider-pair.

The security domain is formed by the set of entities that is controlled by *one* security evaluator/provider-pair. Essential to centralised security functionality is that from the point of view of all entities within the domain set, there is exactly one evaluator/provider-pair that acts as one whole. Thus, from the point of view of the entities, there is no distinction between evaluator and provider. The security-provider part is designed in such a way that *all* requests will be mediated by the evaluator part. The outcome of *all* requests (the decision) is enforced by the security-provider part. All security information is under the control of the security evaluator/provider-pair.

The entities do not offer any security functionality of their own, at least not any security functionality that lies within the scope of the security evaluator/provider-pair.

The entities in the domain are either passive or active. Since passive entities cannot protect themselves, they will always be dependent on one or more other (trusted) entities for their protection. In the centralised method, there is precisely one trusted entity per domain, being the security evaluator/provider-pair.

Note: it is perfectly legitimate to regard the security evaluator/provider-pair itself as one of the entities in its security domain. In that case, the evaluator/provider-pair must be capable of offering the security it requires for itself.

To summarise: the security evaluator/provider-pair controls all security information, mediates all requests and provides all the security required.

6.2.2 Vertical security and centralised security functionality

In this subsection it will be shown how centralised security functionality can be used to provide vertical security. The primary characteristic of vertical security is that security is offered between domain and subdomain.

The security evaluator/provider-pair takes care of all security-relevant activities within the domain. The creation of a security subdomain within a given domain is a typical example of vertical security. The creation of a subdomain requires delegation of trust from the domain to the subdomain. A request for the creation of a security subdomain is evaluated in the normal way by the security evaluator. The issuer of such a request must be a trusted entity that implements a security evaluator/provider-pair with respect to the activities and entities in the future subdomain.

In addition to the normal evaluations the security evaluator/provider-pair verifies the following:

- Has the requester been allowed to create a subset?
- Is the requester a trusted entity with respect to the activities and the entities addressed in the request?

The subdomain as a whole is protected by the security-providing functionality of the domain of which it is a part.

After the creation of a security subdomain, the new situation is as shown in figure 11. All requests stemming from entities in the set of the security subdomain are mediated by the trusted entity.

The trusted entity offers centralised security functionality and implements an evaluator/provider-pair. Requests stemming from entities in the domain are handled in the normal way by the evaluator/provider-pair of the domain.

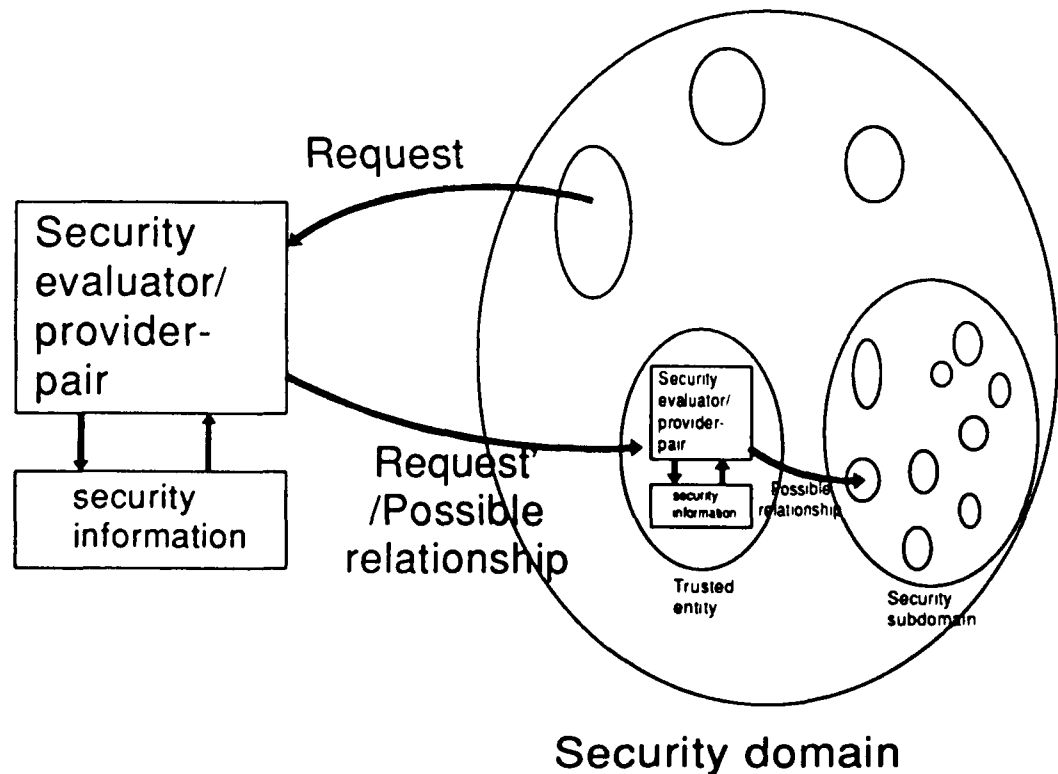


Figure 11: Nesting of centralised security evaluators and providers

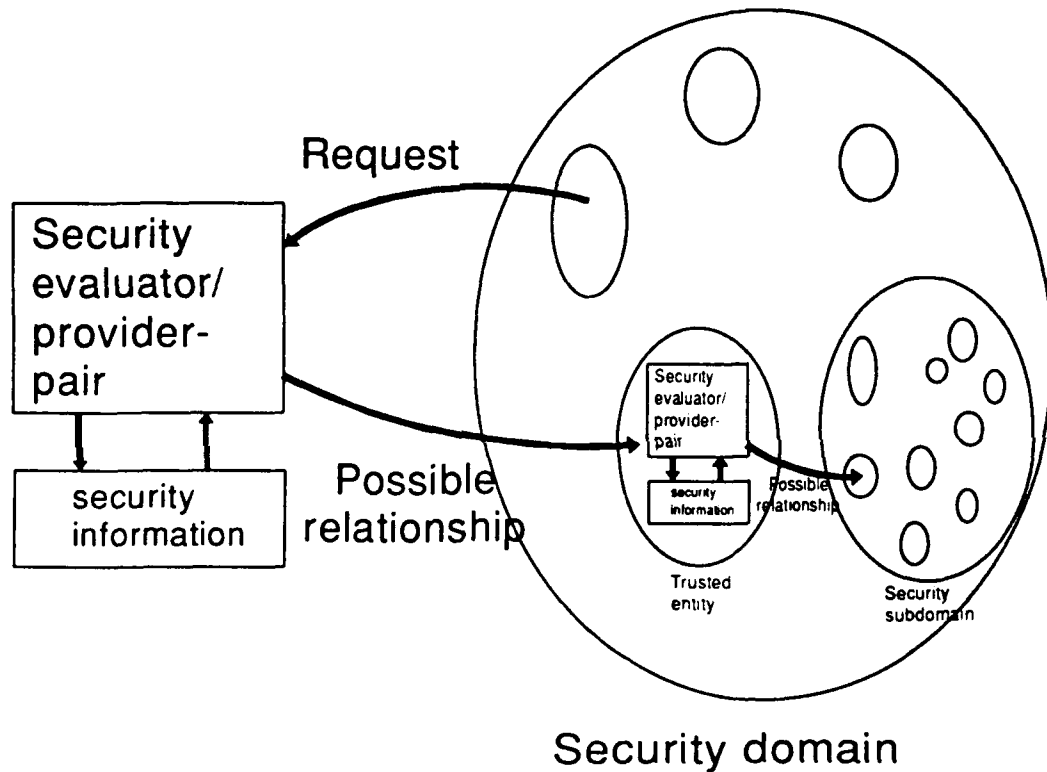


Figure 12: Mediation of requests crossing the domain/subdomain border: request from an entity in the domain for a relationship with an entity in the subdomain

Figures 12 and 13 show the mediation of requests involving an entity in the domain and an entity in the subdomain.

If an entity within the domain requests a relationship with an entity in the subdomain (figure 12), the first mediation takes place by the evaluator/provider-pair of the domain. If the result of this mediation does not exclude the permission of the requested relationship, the request (possibly in an adapted structure) is transferred to the trusted entity of the subdomain, implementing the evaluator/provider-pair. The trusted entity, in its turn, evaluates the request. When both evaluations are positive, the establishment of the requested relationship is granted and the required security is provided as a joint effort of both the security evaluator/provider-pairs.

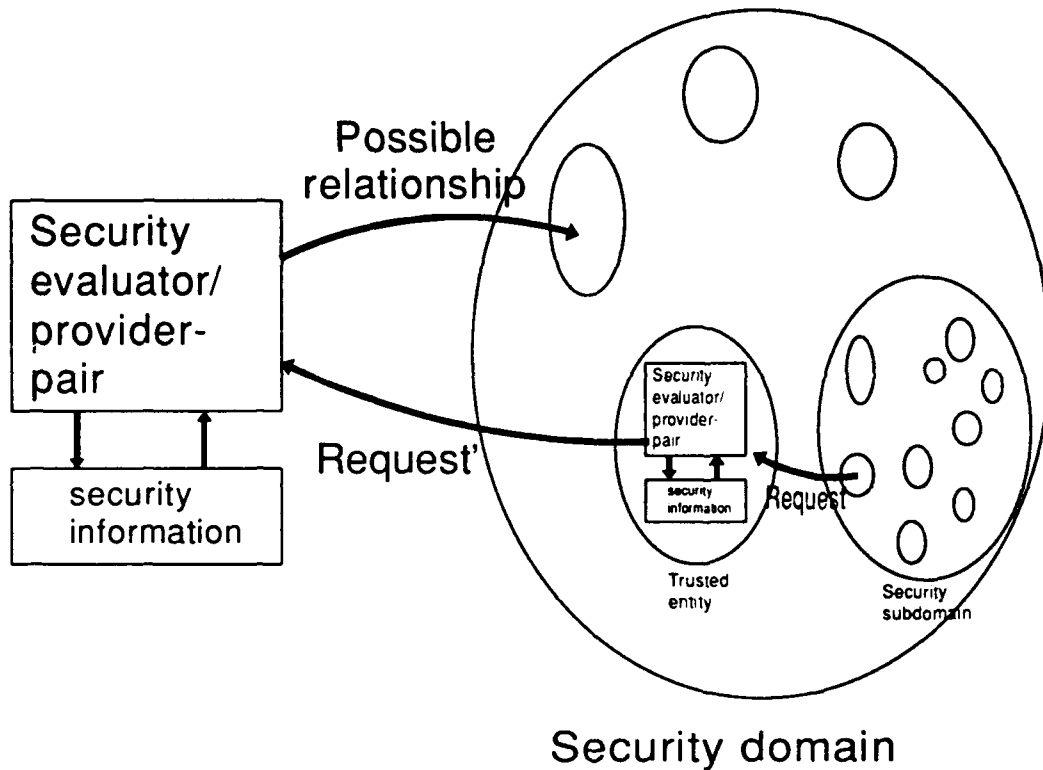


Figure 13: Mediation of requests crossing the domain/subdomain border: request from an entity in the subdomain for a relationship with an entity in domain

The other case (figure 13), an entity within the subdomain issuing a request involving an entity that is not in the subdomain set, is similar: the request is evaluated by both the security evaluator/provider-pairs but in reverse order. The required security is, again, offered as a joint effort of both pairs.

Finally, figure 14 shows the case in which the trusted entity alone is not able to offer a required security service. The trusted entity requests the assistance of the evaluator/provider pair of the domain. If the request is granted, the required security service is offered as a joint effort of both evaluator/provider-pairs. Note that the requester does not notice the fact that there is more than one evaluator/provider-pair involved.

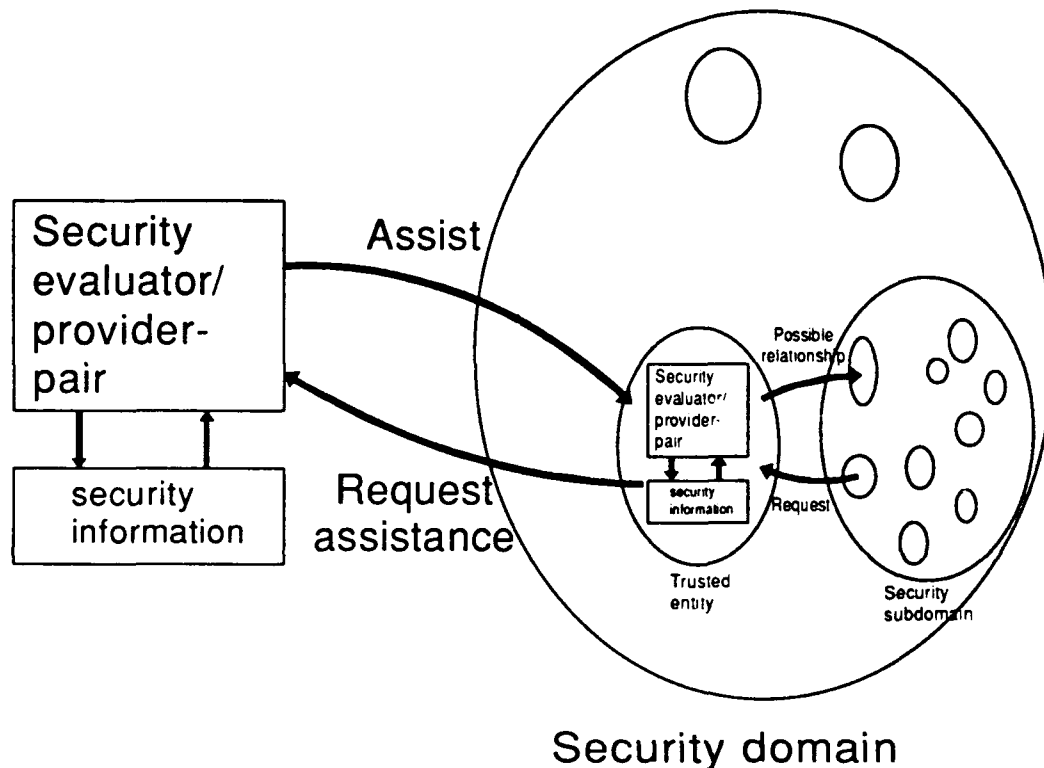


Figure 14: Trusted entity requires assistance of evaluator/provider pair of the domain

In sections 4 and 5 examples of vertical security services were given. In the above it was shown how the centralised method can be used to offer a joint effort between domain and subdomain, which is the primary characteristic of vertical security. The centralised security functionality method therefore is a suitable candidate for providing vertical security.

6.3 Distributed security functionality

6.3.1 Description

Figure 15 shows distributed evaluation of requests and provision of security. This method is partly inspired by the ECMA Framework [15] and some early results of MITRE work [20, 19]. An active entity, for example a process, issues a request for a relationship with one or more other active entities. In the upper part of figure 15 the requester sends the request directly to the entity involved. The addressed entity evaluates the request, based on security information in its

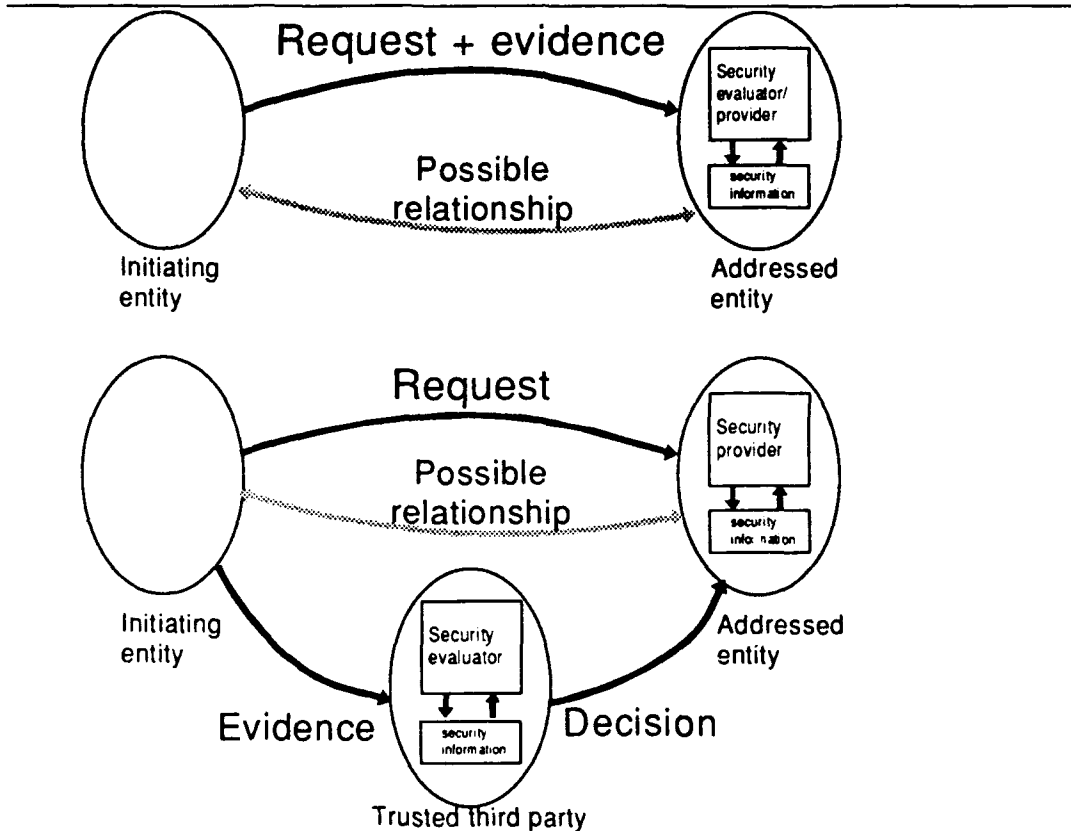


Figure 15: Distributed evaluation and provision of security

possession and security information ('evidence') provided by the requester. The evaluation results in a decision. The decision is carried out by the security-providing functionality of the addressed entity. Thus, following this method, the security information is distributed and both the evaluation of requests and the security-providing functions are centralised functions of the addressed entity. The lower part of figure 15 shows how two entities make use of a third entity that is trusted by both entities (a *trusted third party*). The requester issues a request. The addressed entity asks the assistance of the trusted third party to evaluate the request. The trusted third party performs the evaluation based on security information in its possession and/or additional security information ('evidence') provided by the requester. The outcome of the evaluation (the decision) is given to the addressed entity. Again, the execution of the decision is the task of the security-providing functions of the addressed entity.

There are several obvious variations on this scheme:

- 1 A refinement may be the involvement of the trusted third party in the execution of the decision. One way to achieve this is to have the trusted third party issue vital information as, for example, the cryptographic keys. In this case, the trusted third party serves not only as an evaluator but also as a security provider.
- 2 The establishment of a relationship may start by sending a request to the trusted third party first instead of to the addressed entity.
- 3 The evaluation procedure may include several protocol steps. Numerous examples of these 'handshaking' schemes exist, for example in Kerberos (see [27, 12, 11 and 21]), in Sesame [42], and in OSI [9, 4].

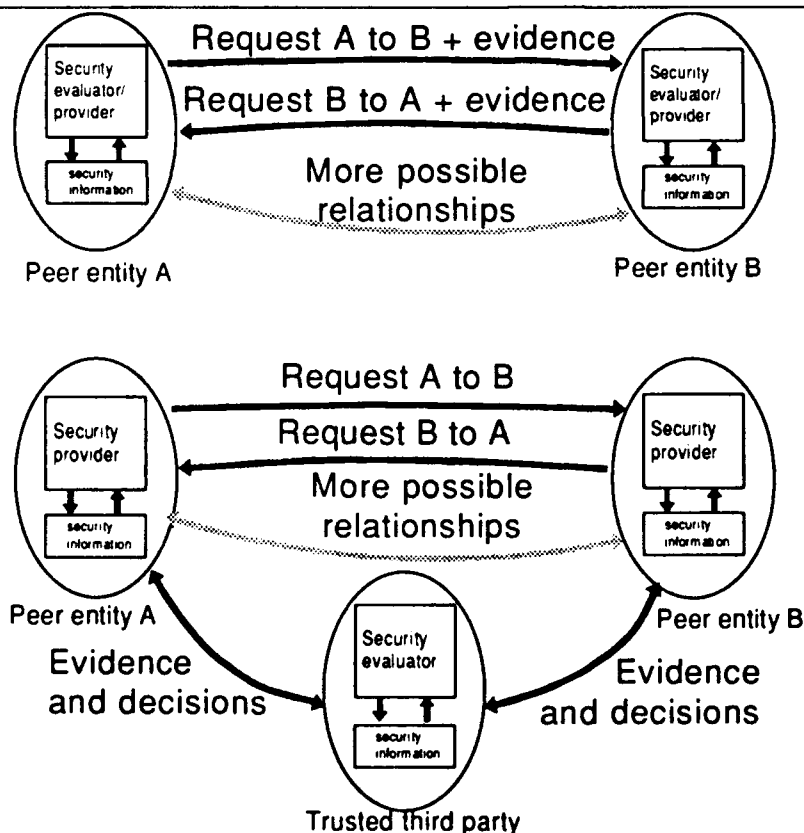


Figure 16: Distributed evaluation and provision of security in a peer-to-peer situation

The starting point in the above is a unidirectional establishment of a relationship: one entity wants something from or with another entity. A peer-to-peer situation that often occurs is one in which

two or more entities establish a more symmetric relationship involving distribution of activities and exchange of information between the participants in the session. In this case, security information, evaluation and security-providing functionality must be present in more entities. This situation is shown in figure 16.

Essential in the distributed evaluation of requests and providing security is that the entity addressed in the request is an *active* entity. This is elementary to perform its part of the evaluation and provision of security. Protection of an entity in a passive role (e.g. being accessed as 'a file' or accessed directly in its working space) cannot be achieved by using the distributed method.

Summary: the security information is distributed among the entities and, possibly, a trusted third party. There may be several security evaluators, possibly including a trusted third party. If several evaluators exist, the evaluation of a request is distributed among them. Also, several security providers may exist, again, possibly including a trusted third party. The execution of the decision will be distributed among the security providers.

6.3.2 Horizontal security and distributed security functionality

The primary characteristic of horizontal security is that security is offered between peer domains. Distributed security functionality operates at the level of peer entities. These peer entities are independent of one another. They are more or less autonomous in their evaluation of requests and do not depend on one another for their own security. If the involved entities each constitute a security domain, the primary characteristic for horizontal security is satisfied.

The creation of a new relationship may imply the chaining of relationships, see figure 17. The chained relationships together create one, more complex, relationship. Since many security issues will involve all entities in the relationship, horizontal security must not only be able to address the security requirements between two peers but must be able to address the aggregated security requirements evolving from a composite horizontal relationship. In this way, a peer-to-peer relationship can be created among three or more domains.

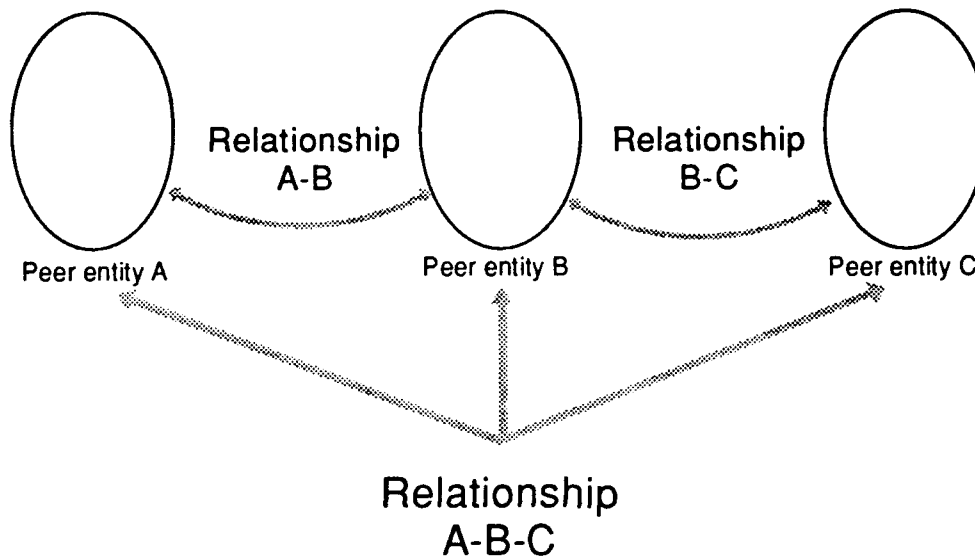


Figure 17: Chaining of relationships

6.4 Integration of the two methods

Two methods for security evaluation and provision of security have been presented earlier in this section. Now it will be shown how the two methods can be combined. First the main characteristics of the two methods are summarised.

The method based on central security evaluation and provision of security is suitable for providing vertical security between domain and subdomain. These domains are nested. The less primitive subdomain depends on the more primitive domain for its security. The centralised provision of security is only possible when all requests are funnelled through the evaluator that is central to the domain. In doing so, the security-providing functions are able to protect the evaluator and themselves (to the extent possible within the scope and granularity of the domain). All security information that is relevant (only) within the domain is in possession of and protected

by the evaluator/provider-pair. Central evaluation and provision of security can be used to protect both passive and active entities. The central evaluator/provider-pair of a subdomain determines whether or not a request requires the involvement of the more primitive domain. If the involvement of the more primitive domain is required, the central evaluator/provider-pair of the subdomain is to pass the request to the domain's central evaluator/provider-pair (possibly in an adapted format).

The method using distributed evaluation and provision of security is a suitable method for providing horizontal security between peer domains. This method requires a mutual protocol, including agreement on structure and contents of the requests. The entities involved in distributed evaluation and provision of security are not able to protect themselves against other attacks and possible security breaches than those possible using the protocol. So, the entities involved in distributed security are, in general, not able to fully protect themselves. The distributed method can only be used by active entities.

Both methods have their own particular advantages and disadvantages. The advantages of the two methods can be combined, reducing the disadvantages. In the combined use of the two methods, the following must be noted:

- 1 At the lowest level of the most primitive domain a request may address a *passive* entity. Passive entities are not able to participate in the evaluation of requests and are not able to provide for their own security. Therefore, the distributed method (requiring active entities) cannot be used at the level of the most primitive domain. The centralised method can be used at the level of the most primitive domain.
- 2 Nested domains, each less primitive domain created from a subset of the entities in the more primitive domain, can be protected using the centralised method since it can be enforced that all requests are mediated by one or more of the nested evaluators.
- 3 Security between active peer entities can be achieved using the distributed method. If the active entities are entities in different, disjunct domain sets centralised evaluation of a request is not possible. So, when peer domains are involved, the distributed method is the only possible method.
- 4 Finally, it is clear that a central evaluator/provider-pair is an active entity.

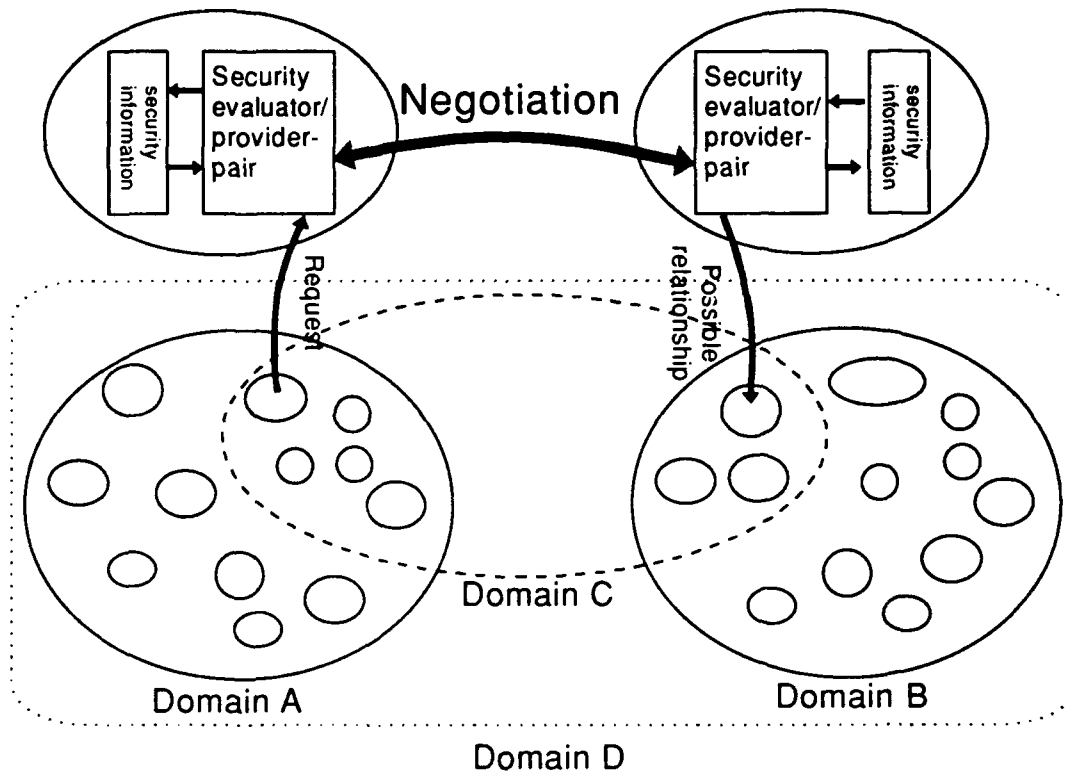


Figure 18: Combination of central and distributed evaluation and provision methods, offering both horizontal and vertical security

Figure 18 shows the combination of the two methods. The figure shows two domains with central evaluation and provision of security (domains A and B). The security between these two domains is based on distributed evaluation and provision of security. This creates the third domain (domain C). The set of entities of domain C is created from subsets of (one or both) domains A and B. In the domains using the centralised method (domains A and B) all requests are mediated by the central evaluator/provider-pair. When a request involves only entities and activities within one domain the evaluation and provision of security takes place in the normal way by the central evaluator/provider-pair of that domain. The security information of the entities concerned must indicate whether the local policy, based on centralised evaluation, must be applied or whether another policy, in this example the distributed policy, must be applied. Note that this security information is under the control of the central evaluator/provider-pair.

When a request involves activities and/or entities in another domain, or the security information indicates that the request is relevant to another domain, the distributed method is applied. Note

that the *decision* about whether the centralised or the distributed method must be applied is made by the centralised evaluator that mediates the request in first instance. So, the decentralised method is, so to speak, built on top of the centralised method: the functionality for distributed evaluation and provision of horizontal security is an integrated part of the centralised evaluator/provider-pair.

Each evaluator/provider-pair implements a local security policy within its domain. The distributed method is implemented by functionality in the two evaluator/provider-pairs. The two evaluator/provider-pairs constitute a joint policy with regard to a joint subset consisting of activities and entities that are of interest to both domains. These two local policies and the joint policy together enforce one policy that governs all activities and entities within the two domains and the joint subdomain. This creates a superdomain (domain *D* in figure 18).

Now, looking at the superdomain only and neglecting its composition, we see that all requests are evaluated within the superdomain, all security functionality is provided in the domain, and all security information is in possession of and protected by the joint security functions of the superdomain. Thus, it has all the characteristics of the centralised method. The composition of the evaluator(s) and security provider(s) is of no importance when the whole composition can be regarded as a single security evaluator/provider-pair. The result is that horizontal and vertical security are integrated and offer a combined effort.

To summarise:

- 1 The centralised method for evaluation of requests and provision of security can be used within one domain as well as between domain and subdomains.
- 2 The distributed method can be used between peer domains. The distributed method can be built upon the centralised method.
- 3 Within the superdomain, seen as the composition of all entities in the two peer domain sets (and thus including the joint subdomain set), the security evaluators and security-providing functions *seen as a whole* implement the centralised method: horizontal and vertical security are integrated and offer a combined effort.

7 THE SESSION MODEL

7.1 Introduction

In section 2.6.3 the concept of a *session* was introduced. A *session* is the life span of an event that can be managed individually. A session is either the life span of an entity in an active role or the life span of a relationship between two or more entities. Different stages in a session are identified. Stages are: session establishment, management during a session and session release.

In the same section 2.6.3 the following examples of a session are given:

- 1 The session during which a relationship exists between an application and a file. The session begins when the application issues a request for access to the file. First the relationship between the application and the file is established (session-establishment stage) followed by use of the file by the application (session-management stage) and finally the file is released by the application (session-release stage).
- 2 The session spanning the life span of a process. This session, during which a process exists as an active entity, is a nested session (a subsession) of the operating system.
- 3 The session of a user spanning the period of time in which a person uses a specific application.
- 4 The session between processes: a relationship exist between processes involved in process-to-process (or: application-to-application) communication. Such a session starts at the initiation of the communication, and ends at the 'disconnect'. Similar peer-to-peer sessions exist between the peer entities at the OSI layers.

Sessions may be nested: a query of a database is a session *within* a user's session. Similarly, the user session may be regarded as a session within an application's session, which in turn is nested within the operating-system's session spanning the course of time from boot until shut-down.

The number of possible relationships in a session is not limited. In the case that more than two applications have a relationship, or in the case that a specific file is being accessed by more than one application, it may be more practical to regard these more complex relationships as one session.

This section is inspired by results of the MOST project [23, 24, 30, 25]. The concept of a session is used in this section to investigate where the different security services can be placed in order to fulfil the security requirements as identified in [35]. Furthermore, in [35] and [34] it is shown that many of today's security policies address access control only. In this section it will be demonstrated that the use of a session model, spanning the duration of a manageable event, offers a basis for a wider range of security requirements than the requirements that can be fulfilled using access control only.

First in this section, the different stages in a session will be discussed. For each stage the range of possible security services is discussed. Furthermore, it will be shown how the session model can be applied to security domains as well as to horizontal and vertical security relationships.

7.2 The stages in a session

The following stages in a session are identified:

- Session-establishment stage.
- Session-management stage.
- Session-release stage.

In this subsection the different stages in a session are further refined and the range of possible security services in each stage is described. The description is independent of the method in which the security functions are provided (see section 6). In this section, the terms *security evaluator* and *security provider* are used as defined in section 6. The description below is a functional description and does not refer to possible implementations. For example, in implementations the initiating and addressed entities in a relationship may implement security evaluators and providers themselves.

7.2.1 Session-establishment stage

7.2.1.1 Description

The session-establishment stage is entered when an entity issues an implicit or explicit request to the security evaluator. An entity that issues a request is called a *requester*. A request is either a request for the creation of an (independent) active entity or a request for a relationship with one or more other entities. The request must be evaluated by the security evaluator(s) involved, and, if

the request is accepted, sufficient security providers must be activated in order to fulfil the security requirements for this relationship.

7.2.1.2 Activities

The activities in the session-establishment stage start when a request is received by the security evaluator. The activities described below do not necessarily have to take place in the listed order and may have to be performed iteratively (particularly the first three steps).

1 Identification

The request is identified by the security evaluator(s). If required for this type of request, the requester and/or addressed entities are identified as well.

2 Competence: determination of security domains involved

Each security evaluator involved verifies that the request and the entities addressed therein lie within its competence. This is, the entities are within its security domain and the security evaluator is authorised to handle this specific request (note that the security evaluator is a trusted entity).

Three possibilities exist:

- If the request is completely outside the evaluator's domain, but inside a superdomain, the request is redirected to the evaluator of the superdomain.
- If the request is partly inside the evaluator's domain and partly inside a superdomain, the request is evaluated jointly by both security evaluators.
- If the request is partly outside the evaluator's domain, and no joint superdomain exists in which all involved entities are nested, the request will have to be evaluated based on negotiation between the security evaluators of the peer domains in which the requester and the addressed entities remain. These negotiations require a session between the security evaluators involved.

3 Authentication and provision of anonymity

The requester and the addressed entities are authenticated to the extent required for this request. The authentication may be based on evidence provided by the entities involved and/or on security information in possession of the security evaluator(s).

Anonymity and authentication are closely related. In some cases, anonymity can be seen as the absence of identification and authentication. Further, in many cases an entity's identity, *once authenticated*, is no longer of importance during the remainder of the session.

4 Evaluation of duties and privileges required

Can the request be allowed? The request is evaluated based on security information provided by the entities involved (credentials, evidence, restrictions, duties, etcetera), and/or security information in possession of the security evaluator(s). In section 2.9.3 it is proposed to effectively base the evaluation of the request on the quintuple:

USER/APPLICATION(S)/OPERATING_SYSTEM(S)/NETWORK(S)/ENTITY

5 Evaluation of current state

Can the request be permitted in the current state of the system? The outcome of the evaluation will also be based on:

- State of the entities involved (e.g. entities may temporarily be unavailable because they are claimed for exclusive use).
- Global domain state (e.g. a back-up is running).

6 Evaluation of required security providers

Can the request be serviced? The required security providers (security services, mechanisms, resources, etc.) are based on the specific demands and capabilities of the entities involved and, of course, on the security services demanded in the request. The set of required security providers can be offered by a combination of security mechanisms and resources (e.g. services for privacy, confidentiality and integrity may be based on the same set of cryptographic mechanisms). The availability of these resources will effect the outcome of the evaluation.

7 Activation and consultation of supportive security providers:

If required, supportive security providers may be used, for example:

- Services for reduction of consequential losses in case of a security event. Reductive measures may require corresponding corrective measures. These security measures may, for example, address the requirements for high availability as in safety-critical applications.
- Consultation of a security event detection service (e.g. a real-time audit facility or an intrusion-detection service). When an event is detected, repressive measures may be taken.
- Services to provide proof of activities in a later state.

These services collect and register information during a session to provide for (legal or other) proof of activities. This may include, for example, proof of transaction by means of a non-repudiation service. Another example is a service that collects information in a

way comparable to the so-called *black box* used in airplanes. This black box-service may be useful in safety-critical applications.

- Privacy services:

Applicable law may require registration and reporting of specific privacy-relevant activities. Furthermore, the legal use of privacy-relevant information is limited. For example, the combination of privacy-relevant information from different sources (*matching*) may be prohibited. Privacy services may be used during a session to limit and/or register the use of privacy-relevant information.

8 Activation of required security providers

The security providers (services, mechanisms, resources, etc.) required in this session are activated (this is done in conjunction with the set of security providers that is already in effect to ensure the secure operation of the new set of security providers).

9 Authorisation of the request

This is the last step in the session-establishment stage. The requester is assigned all that is needed to facilitate the request. This may imply the transition of an entity from a passive to an active state or the establishment of a relationship in which access to passive entities or communication with other active entities is enabled. Note that if the requester is or (perhaps indirectly) represents a real user, the assignment of rights and duties in the system must reflect this person's tasks and responsibilities in the real organisation (see [35, 40]).

The outcome of each of the evaluation steps in the session-establishment stage may be one of the following:

- Yes: the conditions for allowing the request are sufficiently met (as far as within the scope of the step).
- No: the conditions are insufficiently met, so the request will be denied.
- Don't know: there is insufficient information available for a definite decision. In this case, the security policy will decide what happens. Possibilities are:

- Collect additional information to reduce the risks or reduce possible consequential losses.

For example, if the person (the user) that issues a request from some remote location cannot be properly authenticated, the request can be allowed if the remote system or some other user takes over responsibility for the consequences.

The underlying idea is that security events cannot be excluded in all cases, no matter what preventive measures are in effect. In many cases a combination of detective

measures and reductive measures, such as non-repudiation (together increasing the *blame-ability* or providing legal proof), may be sufficient (or, make it someone else's problem).

- Take additional security measures to enable the possibility to undo the activities in this session in a later stage (e.g., to remove the new entities that are created during this session). Or, comparable, the session takes place normally but any change will only be in effect after additional authorisation (e.g. an electronic mail message is prepared in the normal way, but, since the clearance of the requester is insufficient, it will only be sent outside the organisation's domain after clearance by a superior officer).
- Negotiate about change of the request. This may include change of the requested security levels or services.

This non-deterministic approach, accepting the uncertainty in decisions, improves the flexibility and conforms more closely to real world practices.

7.2.2 Management stage

In the management stage activities take place to provide for security and some events need attention. The most important events and activities are:

- Provision of security

A set of security providers is activated during the session-establishment stage and performs its job during the management stage. In this set two subsets are identified. The first subset aims at *protection* of the session and is assigned in order to offer the required security for this session. The second subset aims at *control* of the session. The security providers in this subset see to it that the session remains within its assigned working space, resources, entities, etcetera.
- Event handling, this includes:
 - Standard management functions such as synchronisation, concurrency and control.
 - Managing change requests from within the session: changes of the working space, assigned resources, etcetera, as a consequence of changing needs within the session. These changes are evaluated in a similar way as in the session-establishment stage.
 - Handling of attempts to address resources and/or entities outside the assigned set (outside the working space). Possible reactions are:
 - Regarding this event as an implicit change request.
 - Regarding the event as a possible security breach, in which case additional security services are activated, for instance services for repression and correction.

- Responding to events in the environment of the session. Events outside the scope of control of the session may influence the session and even require the termination of the session. Examples of these events are: technical failures, human errors, precedence of entities and activities with a higher priority. When services for high availability are in effect, alternate resources or even other computer systems may be activated in case of failures.
- Request for the creation of (nested) subsessions:
The session-establishment stage is entered again, taking into account all the security-relevant specifics of the session.
- Request the termination of this session:
The session-release stage is entered.

7.2.3 Release stage

The closure of a session may either be demanded by the issuer of the original request (the requester), demanded by one or more of the involved entities, or initiated by security providers due to events outside the session (e.g. events in a more primitive domain).

A release request is evaluated first. The request does not necessarily have to be granted. Other entities involved in the session may be depending on continuation. Nested subsessions within a less primitive domain may depend on the session too. The entity that issued the release request may be unaware of these dependencies. Many of today's problems in availability seem to be caused by lack of insight in these dependencies [36, 38, 37]. Availability can benefit highly from security measures during the session-release stage. This is of importance since availability is of growing concern in most of today's information systems. For example: it should not be possible to turn off the power supply of a computer system when there are still active entities using the system. The same holds for bringing down communication equipment or shutting down an application or a system while there are still active depending entities.

If the termination request is granted, the security providers that were initiated during session establishment and were active during the session will be stopped as well. Typical activities include (not necessarily in chronological order):

- 1 Integrity control: exchange and registration of information to check the integrity of the session's information and activities. When the integrity services detect a problem, corrective measures may be taken.

- 2 Availability control: as indicated above, termination of a session may influence other sessions.
- 3 Proof: different kinds of proof of the session and its contents may be provided for. This proof may be given in the form of a receipt or signed certificate, possibly with the assistance of a non-repudiation service and a trusted third party.
- 4 Output to security management, registration, etcetera, serving, among others, as input for evaluation of the system-wide security, accounting and 'retrospective' detective measures such as audit.
- 5 Standard activities such as withdrawal of the set of assigned resources, working space, rights/duties, etcetera, and giving the set back to the pool. Security measures must remain in effect to protect possible residual information.

If the termination request is *not* granted, the requester will be informed of this decision and the session-management stage is re-entered.

7.3 Sessions, security domains and horizontal and vertical security relationships

The life span in which a security domain exists can be regarded as a session. This subsection briefly discusses how horizontal and vertical security services can be placed in a session of a security domain.

7.3.1 Vertical security between sub- and super-sessions of security domains

Only a session that creates a most primitive domain does not have a containing supersession. The session of the most primitive domain remains in the physical hardware and depends for its security on non-technical security, e.g. physical protection. In all other cases, the session of a security subdomain is created as a nested subsession within a session of the security domain. In section 5 many vertical security services between domain and subdomain have been described. Some of these services can be placed at specific stages of the session of the subdomain, generally presented in the following.

During the establishment stage of the subsession, the security evaluators of the domain are heavily involved in the creation of the subdomain. The subsession of the security subdomain is depending on the session of the domain, since, during the subsession-establishment stage, it is assigned a set of resources, entities, rights/duties and a working space by the session. This set is a subset of the domain's set. In the subdomain, this subset may be subdivided and thus offer a finer

granularity of protection. Security providers in both the domain and the subdomain are activated to offer those vertical security services that can only be offered as a joint effort between domain and subdomain. Furthermore, during the establishment stage of a security subdomain, assignment of security providers of the security domain both for protection as well as for control of the subsession takes place. These security providers are active during the management stage of the subsession. During the management stage of the subsession, security evaluators and providers of the security subdomain take care of all security issues within the subdomain, possibly in cooperation with the security providers of the encompassing domain.

Finally, during the release stage of the subsession, all security services are wrapped up and the subdomain is closed down in such a way that re-activation can be accomplished in a secure way. In all stages of the subsession exchange of security information between session and subsession takes place.

7.3.2 Horizontal security *within* a session between peer domains

In section 6 the two methods for the evaluation and provision of security have been introduced. If all entities in a session and all activities are within one domain, the centralised method is applied to provide security. If this is not the case, i.e., when several domains are involved, the session is controlled by several security evaluators and, possibly, several security providers. Horizontal security relationships exist between these peer domains. Exchange of security information takes place between the peer security evaluators during all stages.

7.4 Conclusion

In [35] and [34] it is shown that many of the security policies are access control policies. In the same studies it is shown that the current standards in the field of technical security put emphasis on preventive security measures for (mostly) confidentiality and (recently) integrity, neglecting other security needs and other security measures. The model presented in this section stresses the fact that many security services can *not* be offered by means of access control only. This model intends to assist in the definition and placement of security services addressing a wider range of security requirements.

8 INTEGRATION OF RESULTS

8.1 Introduction

In this section the models and building blocks as presented in the previous sections are brought together. The resulting integrated model serves both as a summary of the building blocks and models presented previously, as well as a demonstration of how these models can be fitted together to create a single security architecture.

First in this section, it is shown how an open system can be modelled as a composition of several open elements. Secondly, the concept of trust in a system is reviewed. Next, the different types of security functionality are discussed as well as their relationships with one another. Three types of security domains are defined, matching the boundaries of the open elements. In the security

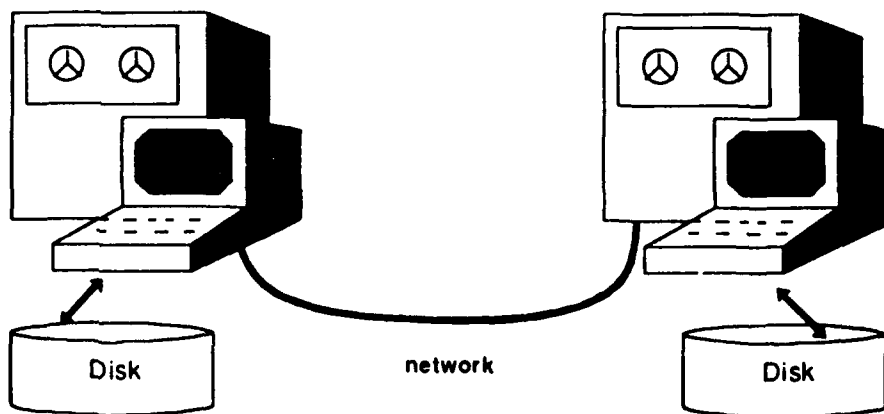


Figure 19: Two computer systems in a network: physical view

domain relationships the different types of security functionality can be applied.

Next, the concept of a session is applied to security domains. Specific security measures can be applied during the different stages of a session. There are two principal methods for the evaluation of requests and the provision of security. These two methods can be combined to offer the required security in and between security domains. Finally, the potentials and limitations of the integrated model are discussed.

8.2 The system: what does it look like?

Figure 19 shows two computer systems connected to a network. This figure shows the hardware, or physical view on the system. It shows the computer hardware, terminals, physical media as well as the physical communication links.

Figure 20 shows the logical view on the same systems in the network. The figure shows applications, operating systems, network processes and information.

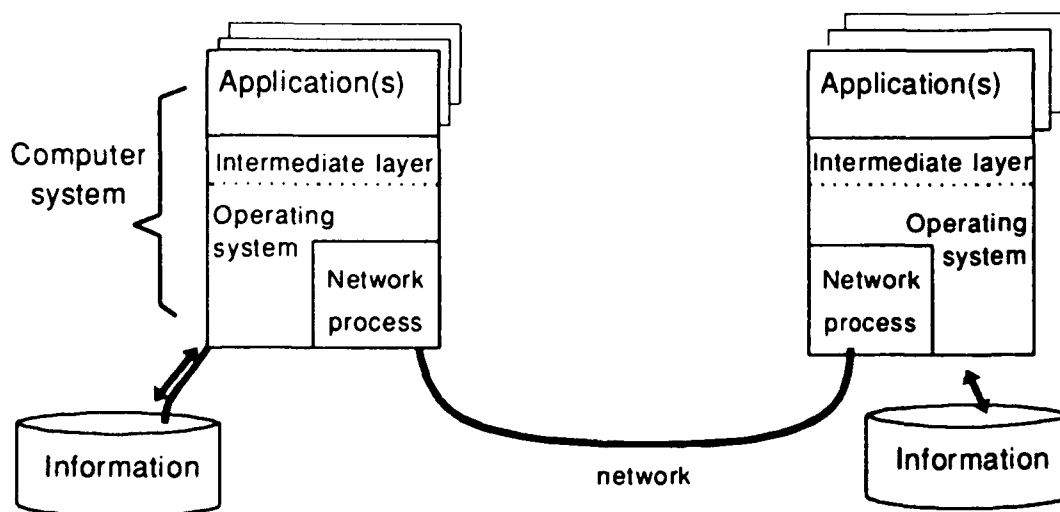


Figure 20: Two computer systems in a network: logical view

Together, the applications, the operating system and the network process make up an open system. These elements are therefore called open elements.

The users use applications. The applications give access to information and other applications, using the services of the operating system. The operating system hides the specifics of the configuration from the applications. The network is seen as one of the configuration-specific elements. The network process takes care of the connection with other systems.

Figure 20 shows an intermediate layer between the applications and the network process. The figure shows a simplification of the Uniform Open Systems Model, introduced in section 2. The operating system may be a 'conventional' operating system, so it may or may not be layered. Added to the operating system is an intermediate layer. This layer may offer many services (see section 2), but essential with respect to this section is that this layer assists in providing security (and therefore is also called the Intermediate Security Service Layer). Only through this intermediary, applications have access to operating system and network services, which, in their turn, offer access to information or other applications. Correspondingly, all activities stemming from the network will be mediated by the intermediate layer as well.

8.3 What is trusted?

Many models are built upon assumptions about the security that is in effect to protect the environment of the model. It is vital that *all* these trust assumptions are made explicit, so that the users of the model are aware of the consequences of these trust assumptions.

With a better understanding of what can be trusted in a system, a more balanced set of security measures can be applied, tailored to fit the requirements of an organisation and the specific threats in the environment of the system.

The organisation(s) in which information systems are in use must decide on the following trust assumptions:

- 1 Which hardware/environments can be trusted.
- 2 Which operating systems can be trusted.
- 3 Which applications can be trusted.
- 4 Which network processes can be trusted.
- 5 Which communication links can be trusted.
- 6 Which persons can be trusted.

And, to what extent and for which uses and activities these can be trusted.

In section 3 it was shown that:

- The trustworthiness of the operating system also depends on that of the hardware and the environment in which it resides (trust assumption 2 necessitates trust assumption 1).
- The trustworthiness of applications also depends on that of the operating system and, as a consequence, that of the hardware and environment (trust assumption 3 necessitates trust assumptions 1 and 2).
- The trustworthiness of network processes at each computer system also depends on that of the operating system and that of the hardware and environment (trust assumption 4 necessitates trust assumptions 1 and 2).

The trusted elements of the system will require access to information about the trustworthiness of other parts of the system.

When several organisations are involved, mutual trust assumptions are registered, e.g. by contract.

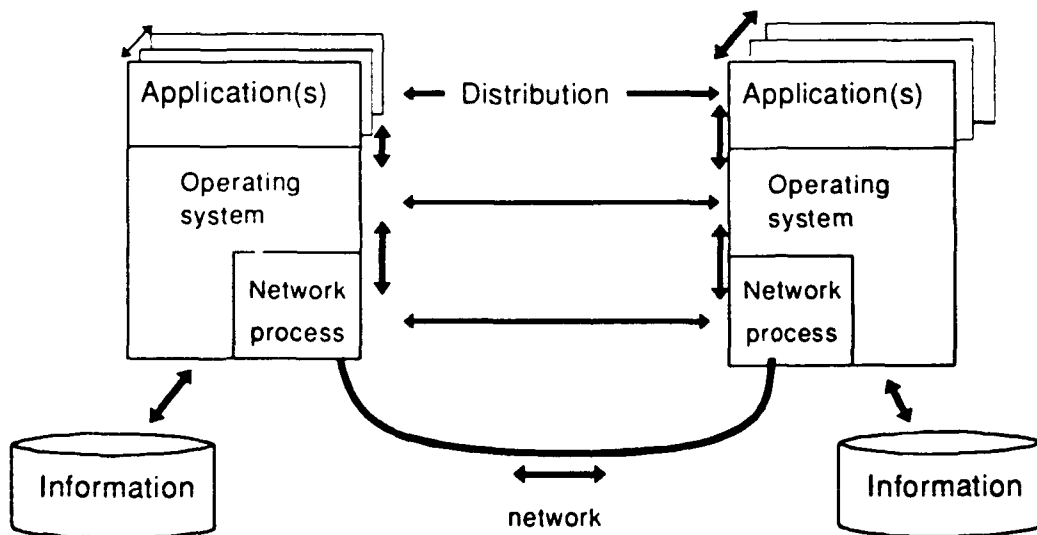


Figure 21: Horizontal and vertical flow

8.4 Distribution, trust relationships and horizontal and vertical security

In figure 21 the flows of information are added to the view presented earlier in figure 20. The information flows are bidirectional. We like to think about communication or information exchange as being horizontal: one application talking to another application, being at the same level of abstraction either locally or at a remote computer system. The other horizontal flows are those between communicating operating systems and between communicating network processes. In reality, the communication is not horizontal, except at the lowest, physical, level; the communication is based on a vertical flow: an application uses the operating system, which may or may not use the network process in order to communicate with a remote computer system. When application-to-application communication takes place, the vertical flow goes at the peer computer system in the upward direction, from (possibly) the network process, the operating system to the application (also see figure 4 in section 2).

This gives two flows:

- The conceptual *horizontal* flow between peer elements.
- The real *vertical* flow between application, operating system and network process (at the lowest level, the physical level, the real flow is in the horizontal direction).

Since security must be present at all times and all places, this gives us two security dimensions:

- *Horizontal security*: security between peer elements.
- *Vertical security*: security between elements that use services and those that provide services.

Both need one another, one does not work without the other.

Beside these two security dimensions there is obviously a third one:

- *Internal security*, which provides the security *in* an open element.

Figure 22 shows the three security dimensions, focussing on open elements. Each open element may have the following security functionality:

Internal security functionality:

Functionality to protect the managed entities that are internal to the open element and
functionality to control and create, manage and release relationships (sessions) between
these entities.

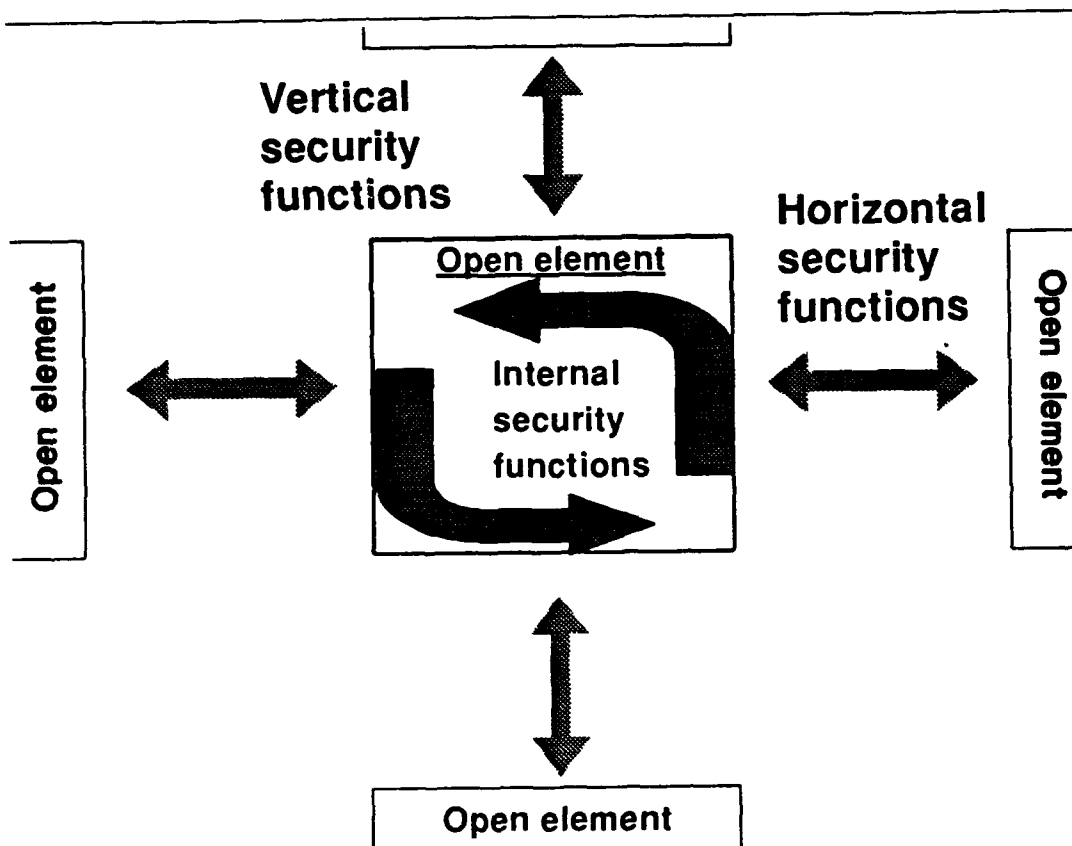


Figure 22: Security relationships of an open element

Horizontal security functionality:

Security functionality to create, manage and release secure sessions between peer elements.

This requires functionality to:

- 1 Establish a secure dialogue in the case that an insecure communication means is used (e.g. a network).
- 2 Authenticate the peer element(s).
- 3 Negotiate requested activities.
- 4 Provide the security services requested/required for this session.
- 5 Manage the session, react to change requests and other events, offer protection and control.
- 6 Release the session in a secure way.

Vertical security functionality:

Vertical security has three aspects:

- Which security functionality is offered to the higher open elements.
- Which security functionality is required from the lower open elements.
- Which security functionality is offered in cooperation with a lower or higher open element. This requires functionality to exchange security information and to offer security crossing the boundary of the open element.

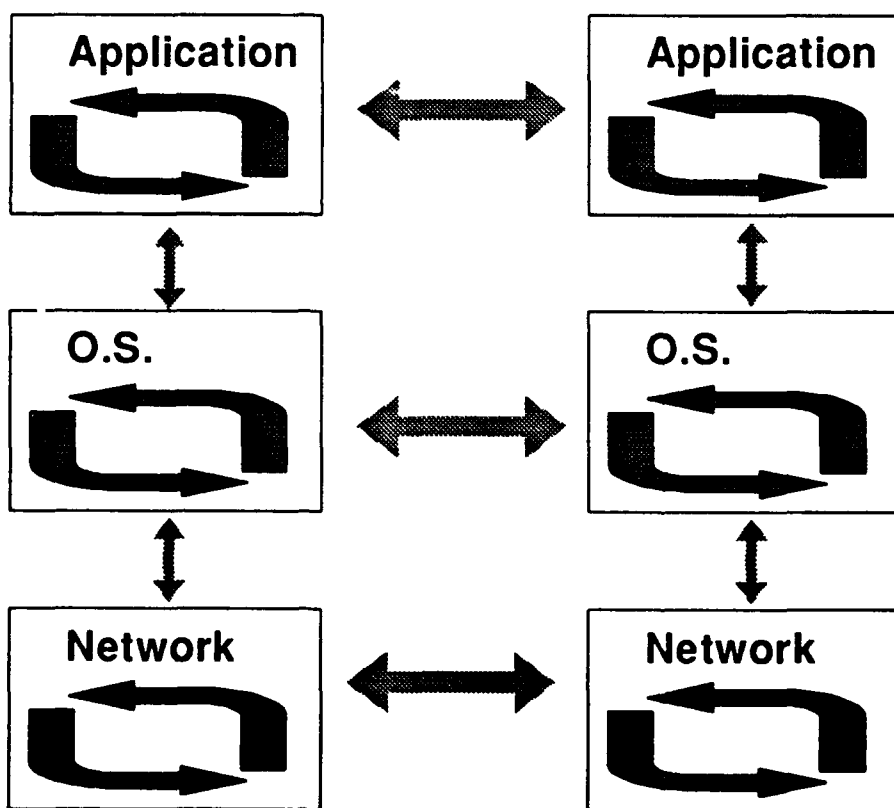


Figure 23: Horizontal and vertical security relationships between elements of an open system

In figure 23 all security relationships between the elements of an open system are shown. Horizontal security and vertical security need one another, one does not work without the other. Both depend on internal security. Finally, and returning to the beginning of this section, technical security (also called logical security) depends on non-technical security measures such as organisational, physical and procedural security measures as well.

8.5 Domains in an open system

The concept of security domains was elaborated upon in section 4. Within a security domain one single security policy is applied to a bounded group of entities. This group of entities forms the domain set.

The security domains are chosen such that the domain sets are either disjunct or subsets of one another, and, for practical reasons, match the boundaries of the open elements. This implies the following security domains:

- The application security domain: an application may constitute either none, a single one or several security policies, depending on the number of disjunct subsets under its control. Thus, one application may provide the security for none, one or several security domains.
- The operating-system security domain: each operating system equals an operating-system security domain. Mostly, there is only one operating-system security domain per computer system.
- The network security domain: each network process equals a network security domain. Mostly, there is only one network security domain per computer system.

8.5.1 Domain relationships

There are two types of domain relationships: the relationship in which one domain depends on the other, as is the case in a domain versus subdomain relationship, and the relationship between domains that are another's equals: the peer domain relationship. The first relationship requires vertical security, the second horizontal security.

8.5.1.1 The domain-subdomain relationship and vertical security

Within the operating-system domain trusted entities may exist. The network process, seen as a whole, is a trusted entity with regard to the security and management of network entities.

Applications may either be trusted or not. Untrusted applications typically do not constitute a security domain. A trusted application has its own set of entities, being a subset of the set of the operating-system domain. A trusted application implements a security policy (based on internal security functionality) with respect to its set of entities. The security policy of an application domain is a refinement of the security policy of the operating-system domain. The set of the application domain may be further subdivided in disjunct subsets. In that case, several security policies may be constituted by one application, one policy for each subset.

Both open elements 'applications' and 'network process' are given resources and are protected in their working space by the operating system. Therefore, if these elements are able to constitute a

security domain, they are subdomains of the operating-system domain (see section 5). The operating-system domain is the most primitive domain, residing directly in the hardware. The relationship between the operating-system domain and the subdomains is one of more primitive, versus less primitive domains.

The subdomain set of an application security subdomain is created from a subset of the operating-system security domain. The same holds for the network security subdomain. Note that the granularity within the application domain may be much finer than in the operating-system domain. The managed entities within the application domain may not even have to be visible within the operating-system domain. This becomes even more clear in the network domain: many of the buffers, frames, and other entities only and/or only temporarily exist within the network domain and are undefined and inaccessible outside the network domain.

Between the operating-system domain and each of its subdomains (the application domains and the network domain) vertical security relationships exist. These are the security services that can only be offered as a joint effort between security functions in the subdomain and the domain. Examples of vertical security services were given in section 5. Vertical security includes delegation and inheritance. Both take place when the subdomain is created (in the session-establishment stage, see later on in this section).

- Delegation

A domain may delegate responsibilities to its subdomains. This is visible in the definition of the subdomain's set of entities and the set of activities that may be performed under the control of the subdomain with respect to these entities. For an application subdomain this may involve a set of files and a set of activities with respect to these files. All responsibility for the handling of these files may be delegated to this one application. For the network subdomain, delegation is likely to involve the handling of all network entities.

- Inheritance

Subdomains are created as sessions within a domain. A subdomain inherits some of its security policy from the more primitive domain. This alleviates the management of the subdomain: when no delegation has taken place, a less primitive subdomain simply cannot compromise the policy of a more primitive subdomain since it does not inherit the possibility to do this.

8.5.1.2 The peer domain relationship and horizontal security

The security between peer domains is based on horizontal security. In section 4.3.2 examples of horizontal security services are given. The following three types of peer domains exist:

Application peer domains:

Two or more applications may be peers of one another. When no vertical or other horizontal security matters prevent this, the peers together are able to constitute a joint security policy.

The set of entities that is governed by this joint security policy forms a joint security domain.

Operating-system peer domains:

Two or more operating systems may be peers of one another. Again, the peers may be able to constitute a joint security policy, creating a joint domain. Since the operating systems are the most primitive domains, the operating systems will 'master' the negotiations when applications (creating a less primitive domain) want to establish a peer-to-peer relationship.

Network process peer domains:

Network processes, active in different computer systems, may be peers of one another.

Governed by the operating-system domain policy, the network domains may establish a joint network domain.

8.6 Sessions

A *session* is the life span of an event that can be managed individually. A session is either the life span of an active entity or the life span of a relationship between two or more entities. In the Session Model, described in section 7, the concept of the session seen from a security point of view was discussed. Different stages in a session are identified. Stages are: session establishment, management during a session and session release. At each stage specific security requirements have to be addressed and specific security measures can be applied. The Session Model stresses the fact that many security services can *not* be offered by means of access control only. This model intends to assist in the definition and placement of security services, addressing a wider range of security requirements.

Three types of sessions are identified:

- 1 Sessions in which only a single active entity takes part.
- 2 Sessions in which passive entities are involved, beside at least one active entity.
- 3 Sessions between two or more active entities.

Having modelled an open system as a composition of applications, an operating system and a network process as open elements, this implies three types of sessions involving open elements.

Type 1: Sessions in which only a single active entity takes part.

- **The operating-system security-domain session:**

The session that exists in the time span from 'boot' of the computer system, via 'running' of the operating system as an active entity, until shutdown of the computer system. Within the lifetime of this session, the operating-system domain is active. This session resides in the physical hardware and depends on non-technical security measures only (outside the scope of these models). This session creates the most primitive domain at a computer system.

- **An application session:**

The session of an application, from initiation, running as a process, until termination. If created by the operating system, this session is a nested session within the session of the operating system. If created by another application, this session is a nested session in that application's session.

If the application is a trusted entity, the application session may create a security subdomain.

This is a subdomain of either the operating-system domain or another application domain (which, in that case, must be trusted as well).

- **The network-process security-subdomain session:**

The session of a network process, from installation and activation, active period, until de-installation. This creates the network domain, which is a subdomain of the operating-system domain.

Type 2: Sessions in which passive entities are involved:

- All active entities, either applications, the operating system or the network process, are able to initiate a session with passive entities. Some examples of passive entities are: records, files, buffers. A session of an active entity with a passive entity starts at the request for the relationship, followed by activities on the passive entity, and ends with releasing the passive entity. An active entity may be involved in several sessions with passive and/or active entities simultaneously. Furthermore, several active entities may simultaneously have sessions with the same passive entity.

When the activities take place in an application-security subdomain (which implies that the application is a trusted entity within the operating-system domain and the passive entity concerned is in the set of the application's security domain), this is a nested session within the session of the trusted application. In turn, this session is a session within the operating-system session.

Type 3 Sessions between two or more active entities.

- The sessions of this type are sessions between peers. These peers are either peer applications, peer operating systems or peer network processes.

8.7 Evaluation of requests and provision of security

The two methods for offering security are summarised first in this subsection. Next, the use of the two methods to provide for the different types of security functionality in an open element (internal security, vertical security, horizontal security) are discussed, under the assumption that the domain boundaries are chosen to match the boundaries of the open elements.

8.7.1 Two methods for offering security

In section 6, the two principal methods to offer security were discussed. The following functionality is needed:

- Security-evaluating functionality, mediating each request for a security-relevant activity (called: evaluator).
- Security-providing functionality, to control and protect each session to an acceptable level (called: provider).

This functionality must be available in each security domain.

The first method for offering security in a security domain is based on centralised security functionality. Essential to centralised security functionality is that from the point of view of all entities in the domain set, there is precisely one evaluator/provider-pair acting as one whole. The security-provider part is designed in such a way that all requests will be mediated by the evaluator part. The outcome of the evaluation (the decision) is enforced by the security-provider part. All security information on which an evaluation can be based is under the control of the security evaluator/provider-pair. The entities do not offer any security functionality of their own, at least not any security functionality that lies within the scope of the security evaluator/provider-pair. The entities in the domain are either passive or active. In the centralised method, there is one trusted entity, being the security evaluator/provider-pair.

To summarise: in the centralised method there is one security evaluator/provider-pair that controls all security information, mediates all requests and provides all the required security.

In section 6 it was shown how the centralised method can be used to provide vertical security.

The second method for offering security uses distributed evaluation of requests and distributed security functionality. Essential in the distributed evaluation of requests and providing security is that the entity addressed in a request takes an active part in the evaluation and/or providing its own security. Therefore, it must be an active entity. The security information is distributed among the entities and, possibly, a trusted third party. There may be several security evaluators, possibly including a trusted third party. If several evaluators exist, the evaluation of a request is distributed among them. Also, several security providers may exist, again, possibly including a trusted third party. The execution of the decision will be distributed among the security providers.

In section 6 it was shown how the distributed method can be used to provide horizontal security.

As shown in section 6, the two methods can be combined to offer security across all possible domain boundaries.

8.7.2 Internal security

The functionality within a security domain must be sufficient to offer all the internal security autonomously. So, the evaluator must be able to evaluate and decide on service requests stemming from an entity within the domain, addressing another entity in the same domain. Since no domain boundaries are crossed, the centralised method for evaluation and provision of security can be used within a domain. Using the centralised method, the security-providing functions are able to carry out each decision of the security evaluator.

Note that the security evaluator does not only decide on the validity of a request, but must also ensure, by soliciting the security provider, that sufficient resources are available to offer the required security services before a request is granted.

8.7.2.1 Distribution

The security evaluator in a domain must also be capable of deciding whether a request is partly or completely outside its jurisdiction. This is, the entities involved are not (all) in the domain set and/or the requested actions are not within the responsibilities of the evaluator. When a request falls totally outside the domain set and set of responsibilities, the request is either delegated to a less primitive subdomain (the request has to be delegated to a subdomain) or to a more primitive domain (the request has to be delegated to a superdomain).

If the request falls partly outside the evaluator's responsibilities and inside those of the evaluator of a sub- or superdomain, a vertical relationship between the evaluators involved is established.

Another possibility is that the request falls partly within the evaluator's responsibilities, but also within the responsibilities of the evaluator of a peer domain. The security evaluator must be able to establish a peer-to-peer relationship (possibly in cooperation with a more primitive domain). A peer-to-peer session will have to be established. The two security evaluators in the peer domains negotiate the request and reach a joint decision.

8.7.3 Vertical security

Evaluation and provision of security in a domain can be based on the centralised method. When a security evaluator in a less or more primitive domain forwards a request to the evaluator of the sub- or superdomain, this request can be handled by the evaluator concerned as a local request. So, the cooperation between security evaluators that are in a vertical relationship with each other is not very complex.

When a security provider offers a specific security service, this usually requires the assistance of security providers of the more primitive domains, since security functions/mechanisms of different granularity must be combined (shown in section 5). The security provider therefore issues a request (for assistance in providing a security service) to the evaluator of the more primitive domain. This request is evaluated in the normal way by the security evaluator of the more primitive domain. The requester will be informed about the outcome of the evaluation. Thus, the outcome of a request on the level of a more primitive domain may impact the outcome of an evaluation on the level of a less primitive domain.

Of course, the more primitive domain protects the less primitive domain in its working space, its files, etcetera. The more primitive domain does all that is necessary to protect the less primitive subdomain. This does not require communication between these vertical domains.

8.7.4 Horizontal security

When a request requires the establishment of a peer-to-peer relationship, the following actions take place:

- 1 A peer-to-peer session is established (see earlier in this section and section 7). When the means for exchange between the peers is not considered trustworthy enough, authentication and establishment of secure communication must take place. The authentication process takes place between the two security evaluators, using their respective security providers. The establishment of a secure communication means, if required, is arranged by the security providers.

- 2 The two security evaluators negotiate about the request (consulting their security providers). When the request is granted, the set of entities within both domains may be subdivided. Each domain may be split in two subdomains. One subdomain will be governed based on the security policy between the peers. The other subdomain is governed in the same way as before. The security evaluators and security providers of the two peer domains now constitute a joint security policy. Since this policy is applied to a bounded set of entities, we can speak of the establishment of a joint security domain.

To summarise:

- Internal security is offered by centralised evaluation and providing security within a security domain (created by an open element).
Internal security offered by centralised evaluation and providing security may be present in applications, in an operating system and in a network process.
- Vertical security between open elements is offered by a combination of the security provider of a less primitive domain and both the security evaluator and provider of the more primitive domain. As a whole, these functions act as one centralised evaluator and provider.
Vertical security may be offered between application and operating system, as well as between network process and operating system.
- Horizontal security between open elements is offered by the two security evaluators involved and, when security services are required, the two security providers involved.
Horizontal security is offered between peer applications, between peer operating systems and between peer network processes.

8.8 What is achieved: potentials and limitations

A security architecture is defined crossing the borders of applications, operating systems and network. Some of the models presented in this report require the development of standardised interfaces, protocols and data structures. When these become available it is possible to create secure open systems that are capable of:

- Fulfilling *all* security requirements.
- Ensuring the security of information, applications, operating system and network process.
- Offering security between applications, between operating systems and between network processes.
- This security can be offered in a computer system but also between computer systems, even using an insecure network.

Open elements designed according to this model will be able to offer security in coordination with other open elements, thus creating a secure open system.

Furthermore, the models provide a reference and assist in the development of open software products addressing a wider range of security requirements.

Assumptions:

Supportive, non-technical security measures are required:

- The integrity of the hardware of the computer systems must be ensured.
- The integrity of the operating system, the network software and the applications at installation time must be ensured.
- The trustworthiness of the diverse elements in an open system must be determined. Information about trustworthiness must be made available to the trusted elements in the system (possibly in the form of credentials and certificates). When several organisations or independent organisational units are involved, the trust assumptions must be agreed upon by the parties involved.

Limitations

- Changes in the configuration, e.g. rebuilding an open system to comprise another set of open elements, may require re-evaluation of the trustworthiness of the system as a whole. Information to support integrity measures and authentication may have to be updated.
- As indicated above, trust relationships between peers can only be established between peers that know one another to be trustworthy and can prove their identity (directly or via a trusted third party), e.g. by demonstrating their credentials. This requires the exchange of validation information (e.g. credentials) which lies outside the scope of this model. This model does not offer the possibility to establish trust relationships between new, still unknown partners (unless a trusted third party exists).
- Vertical security can be enforced by using only preventive security measures (although it is not recommendable to rely on one type of security measures only). In horizontal security, prevention is not always possible. Horizontal security remains an issue of trust to a certain extent. Therefore, in offering horizontal security more emphasis must be put on reduction, detection and repression. The risks in a commercial environment can further be reduced by using trusted third parties and a mutual agreement on limitation of the liability.

9 CONCLUSION: TOWARDS SECURE OPEN SYSTEMS

This section formulates an evolutionary approach from today's practises and standards to those of tomorrow in an open systems' environment. Choices are made in such a way that a realistic (which is not the same as easy) evolution from today's open and proprietary systems is possible.

9.1 Starting point

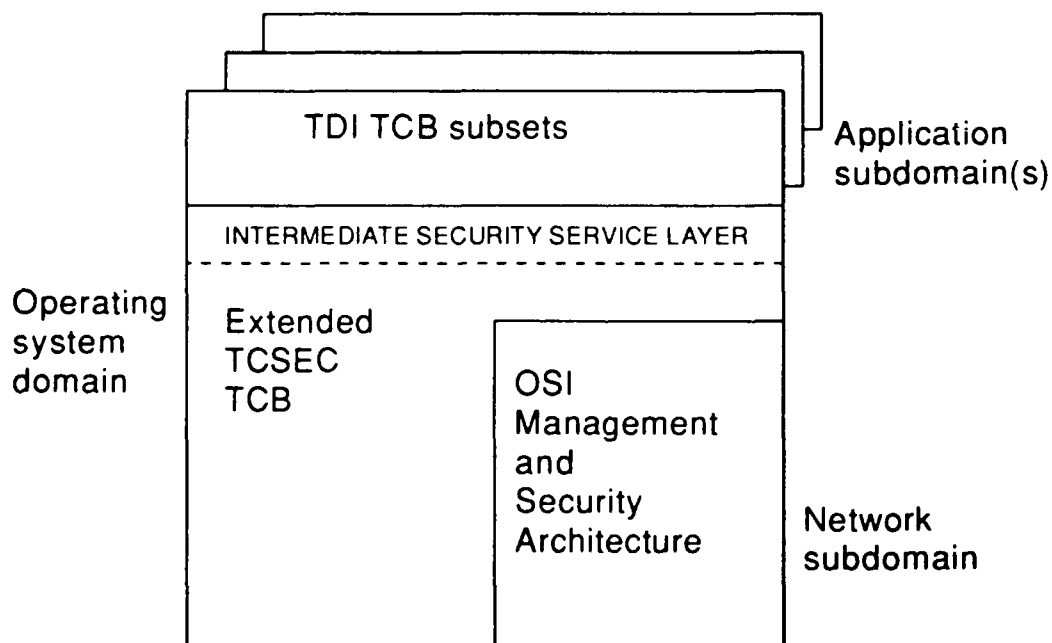


Figure 24: Security standards per computer system: first step

Figure 24 shows the starting point for a growth path and the use of security standards therein. The *operating system* will implement the most primitive domain for a computer system. The security of the operating system is based on the TCSEC TCB [3] with the extensions as proposed in the TDI [18]. The kernel of security in the operating system will be the TCB in which security

functionality is centralised and enforced by the reference monitor. Added to the TCSEC approach are: the concept of trusted entities and the possibility to create subdomains.

The operating system has two types of subdomains: the first type is the network-process subdomain (in most cases *one* per computer system); the second type is the application subdomain. The operating system will protect these subdomains and provide resources. The operating system will be able to delegate security tasks to its subdomains.

Also added to the TCB is the *Intermediate Security Service Layer*. This layer, through the reference monitor, will mediate all requests for operating system and network services, stemming from the applications, and mediate all request for access to applications stemming from the network. Thus, all use of network services and all activities stemming from the network will be guarded and protected by the operating-system domain (and, of course, the network subdomain). Today, the boundary of what should be treated as an application and what should be treated as part of the network process is rather vague. Using this model, the granularity of the information represented in the managed entities is a proper yardstick: applications work with entities with information of the same or more refined granularity than that of the operating system; the network subdomain works with entities that are less information and more communication orientated. Entities in the network domain do not have any meaning outside this domain. This implies, for example, that electronic mail, FTAM, and EDI are regarded as applications.

Starting point for security in applications is the TDI. *Applications* that are trusted entities constitute a security policy with respect to the entities in their domain set, which is assigned by the operating system. The trusted applications therefore constitute a security evaluator/provider-pair as a TCB subset, possibly in the form of a reference-monitor-like mechanism, thus creating a Trusted Application Base (TAB).

Within the *network subdomain*, the internal security is based on OSI management. The horizontal security between network subdomains is based on the OSI Security Architecture.

9.1.1 What does it add to today's situation

The concepts of the TCSEC, the TDI and the OSI Security Architecture are integrated. All network interactions will be brought under the control of the operating-system TCB, implementing one or more security policies in one or more security domains and subdomains. This starting point should be achievable for many manufacturers today.

9.2 Future steps

The following steps should follow, in equal priority.

9.2.1 Horizontal security

Protocols and interfaces for horizontal security between applications and between operating systems must be developed, based on the ECMA approach for communication between peer domains (note that the communication between sub/superdomains is based on the TDI approach, which may be seen as a specific application of the ECMA approach; the horizontal security between the network subdomains is based on the OSI Security Architecture, which was one of the sources of inspiration in the development of the ECMA model). Supportive security standards, notably X.509 and standards like Kerberos or Sesame, can be embedded.

The development of protocols and interfaces for horizontal security is neither beyond the state of the art, nor very complex. This is shown from the experience of the development of protocols conform the OSI Security Architecture, the development of Kerberos and X.509. First priorities are:

- Peer-to-peer authentication (similar to the ISO Authentication Framework [9]).
- A common security language to exchange security information, starting with standardised data structures for security information (also see [1, 2], starting to define standardised data structures for security information, the so-called Security Information Objects).

Next, horizontal security services must be standardised, requiring protocols and interfaces. See section 4.3.2 for a description of possible horizontal services.

9.2.2 Vertical security

Subdomains and their domains must 'learn' to communicate. Two issues must be solved:

- Communication between the operating-system domain and its network subdomain. The easiest way to achieve this is to define a security interface between OSI Management (organising the security in the network subdomain) and the security evaluator of the operating system. This is possible in the form of a bi-directional interface between the reference monitor and OSI Management.
- Communication between the operating-system domain and a *generalised* application subdomain (one fits all).

This requires the definition of a bi-directional security interface between the operating system and the applications.

Also, vertical security services must be standardised, requiring protocols and interfaces for the security providers in the different domains. See section 5 for a description of possible vertical services.

9.2.3 Internal security

Different from today's practices will be the fact that the internal security-evaluation functions will have to be able to differentiate between requests that require local evaluation only and requests that cross the domain boundary. In the latter case, communication with subdomains, superdomains or peer domains must be established.

9.2.4 Enrichment of security services

Security standards should be enriched to address a wider range of security requirements, as identified in [35] and [40]. The list of security requirements so far poorly addressed includes the following (see [35] for details):

- Requirements for information security that stem from organisational considerations include:
 - Mapping of real-life tasks and responsibilities to rights and duties in the information system.
 - Mapping of organisational structures and relationships to the system.
- Demands from society for information security include: privacy, anonymity, security for safety-critical systems and legal or other proof of use and correct functionality of an information system.
- Requirements for security functionality in the system include: authentication, management, integrity and availability. Further, a more balanced set of security measures is needed, putting less emphasis on prevention and more on reduction, detection, correction and system-wide evaluation of security.

9.3 Conclusion

The studies [35] and [40] concluded that there is a lack of integration between application security, operating-system security and network security and that, for the purpose of integration of security, an architecture for security functionality is needed that crosses the borders of applications, operating systems and networks.

This study offers models and building blocks to create secure open elements that together will make up a secure open system. The study shows that it is possible to offer security services in

open systems that are able to fulfil the majority of today's and tomorrow's requirements for security.

The study also shows that there still is a long way to go: standardised interfaces, protocols, data structures must still be defined. However, none of this is beyond state of the art. Furthermore, the study shows that there is a viable evolutionary path from today's practices and standards, so there is no need to start all over again from scratch.

10 ACRONYMS

The following acronyms are used in this document.

ANSI	American National Standards Institute
API	Application Programming Interface
ASE	Application Service Element
ATM	Automatic Teller Machine
BC	Behavioural Component
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CEC	Commission of the European Communities
DAF	Support Framework for Distributed Applications
DBMS	Database management system
DoD	USA Department of Defense
EC	European Commission
ECMA	European Computer Manufacturers Association
EDI	Electronic Data Interchange
ETSI	European Telecommunications Standards Institute
EWOS	European Workshop for Open Systems
FIPS PUB	Federal Information Processing Standard Publication
FTAM	File Transfer, Access and Management
IEEE	Institute of Electrical and Electronics Engineers
IEPG	Independent European Programme Group
IPSE	Integrated Project Support Environment
ISO	International Organisation for Standardisation
IT	Information Technology
ITAEGV	Information Technology Advisory Expert Group for Information Security
ITSEC	Information Technology Security Evaluation Criteria
ITSEM	Information Technology Security Evaluation Manual
JTC	Joint Technical Committee
LAN	Local Area Network
MHS	Message Handling System
MIT	Massachusetts Institute of Technology

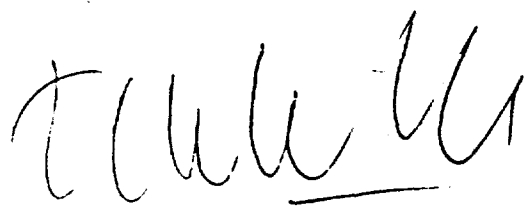
NATO	North Atlantic Treaty Organisation
NCSC	USA National Computer Security Center
NIST	USA National Institute of Standards and Technology
NOSA	NATO OSI Security Architecture
NTCB	Network Trusted Computing Base
ODA	Office Document Architecture
OIW	Open Implementors Workshop
OS	Operating System
OSF	Open Software Foundation
OSI	Open Systems Interconnection
OSI SA	OSI Security Architecture
PAC	Privilege Attribute Certificate
PCTE	Portable Common Tool Environment
POSIX	Portable Operating System Interface for Computer Environments
SC	Security Component
SC	Subcommittee
SCSI	Small Computer Systems Interface
SILS	Standard for Interoperable Local Area Network Security
SIO	Security Information Object
SMIB	Security Management Information Base
SQL	Structured Query Language
SSA	Supportive Security Application
STANAG	NATO Standard Agreement
TAB	Trusted Application Base
TC	Technical Committee
TCB	Trusted Computing Base
TCSEC	Trusted Computer System Evaluation Criteria (Orange Book)
TDI	Trusted Database Management System Interpretation of the TCSEC (Grey Book)
TIU	Trusted Interface Unit
TNB	Trusted Networking Base
TNI	Trusted Network Interpretation of the TCSEC (Red Book)
TOE	Target Of Evaluation
TOS	Text and Office Systems
UIN	User-Identification Number

11 REFERENCES

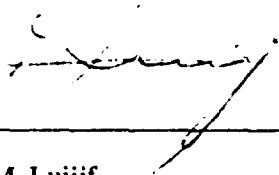
- 1 "2ND WD: Security Information Objects (SIOs), Part 1: Methods and Guidelines for Definition and Registration", ISO/IEC/JTC1/SC27 N422, 1992-04-08
- 2 "2ND WD: Security Information Objects (SIOs), Part 2: Element and Generic SIO class specification", ISO/IEC/JTC1/SC27 N423, 1992-04-08
- 3 "Department of Defense Trusted Computer System Evaluation Criteria", DoD 5200.28 STD, (also known as TCSEC or Orange Book), December 1985
- 4 "Entity authentication mechanisms, part 2: Entity authentication using symmetric techniques", CD 9798-2, Source: ISO/IEC JTC 1/SC27 N489 (WG2), 1992-06-09
- 5 "ETG - Directory Security Architecture", EWOS/EGDIR/91/144 (EWOS/EGSEC/92/001), 18th October 1991
- 6 "Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture", ISO 7498-2:1988
- 7 "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", ISO 7498:1984
- 8 "Information Processing Systems - Open Systems Interconnection - The Directory - Part 8: Authentication Framework", ISO/IEC 9594-8:1990
- 9 "Information Technology - Open Systems Interconnection - Security Frameworks for Open Systems - Part 2: Authentication Framework", ISO/IEC DIS 10181-2, 1991-07-18
- 10 "Information Technology - Open Systems Interconnection - The Directory - Proposed Draft Addendum to ISO 9594 (parts 2 to 4) on Access Control", ISO/IEC 9594-1 to 3 /PDAD 1, also available as ISO/IEC JTC1/SC21 N4041, N4042, N4043, December 1989
- 11 "Kerberos version 5 RFC", available from krb-protocol@athena.mit.edu, December 1990
- 12 "Kerberos: An authentication Service for Open Network Systems", MIT Project Athena, Cambridge MA 02139, January 1988
- 13 "Secure association service and management", ECMA/TC32-TG9/92/xx (draft, 1992)
- 14 "Security Application - Authentication and Privilege Attribute", ECMA/TC32-TG9/91/26 (7th draft, April 1991)
- 15 "Security in Open Systems - A Security Framework", ECMA TR/46, July 1988
- 16 "Security in Open Systems - Data Elements and Service Definitions", ECMA-138, December 1989

- 17 "The Directory - Part 8: Authentication Framework", CCITT X.509, 1990
- 18 "Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria", (also known as TDI or Grey Book), USA National Computer Security Center, NCSC-TG-021, library no. S235.625, April 1991
- 19 Abrams M.D, "Rule-based trusted access control" IFIP SEC'92, 8th International Information Security Conference, Singapore, pp 443-454, 27-29 May 1992
- 20 Abrams M.D, Eggers K.W, LaPadula L.J, Olson I.W, "A Generalized Framework for Access Control: An Informal Description", MITRE Corporation MP-90W00043, August 1990
- 21 Bellovin S.M, Merrit M, "Limitations of the Kerberos Authentication System", Conference Proceedings, USENIX - Winter '91 - Dallas TX, 1991
- 22 Clark D.D, Wilson D.R, "A Comparison of Commercial and Military Computer Security Policies", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, April 1987
- 23 Franken P.W.M, Meulemans P.A.C.M, Overbeek P.L, v. Rijt Q.G.N, Groen J. (PTL), Jimmink R. (PTL), Vonk P.: "Basisontwerp voor de beveiliging van telematicasystemen", FEL-90-C029, in Dutch, 1990
- 24 Franken P.W.M, Meulemans P.A.C.M, Overbeek P.L, v. Rijt Q.G.N, Groen J. (PTL), Jimmink R. (PTL), Vonk P. (PTL); "Een model voor de beveiliging van telematicasystemen en -diensten", FEL-90-C386, in Dutch, 1990
- 25 Groen J, Meulemans P.A.C.M, "Security architecture for computer and communications systems", in SECURICOM 91 - 9th Worldwide Congress on Computer and Communications Security and Protection, pp 55-71, CNIT, Paris la défense, France. Groupe Blenheim, Levallois Perret Cedex/France, March 20-22 1991
- 26 I'Anson C, Mitchell C, "Security Defects in CCITT Recommendation X.509", Computer Communication Review, Vol 20 (1990) no.2, pp30-34, 1990
- 27 Kohl John T, "The evolution of the Kerberos Authentication Service", Project Athena, EurOpen '91, pp 295-313, 2-24 May 1991
- 28 Linn John, "Generic Security Service Application Program Interface (GSS-API)", Internet Draft, Common Authentication Technology WG, doc.expiration.date: 15 December 1992, June 1992
- 29 Luijff H, Overbeek P, "The FEL-TNO uniform open systems model", pp. 201-208. In: Towards an Open World: Proceedings 1989 DECUS Europe Symposium, The Hague, Holland - DECUS, Switzerland, September 18-22, 1989

- 30 Meulemans P, Overbeek P, Groen J, Jimmink R, "Beveiliging op maat - PTT Research en TNO slaan handen ineen", *Telecommagazine* 5(10):35-41, in Dutch, 1990.
- 31 Overbeek P, Luijff H, "UNIFORM OPEN SYSTEMS model: A network wide view on applications and operating systems", pp. 1/112 - 1/134. In ECODU, (Ed.), *Conference Proceedings ECODU 47, European Control Data Users, Z.pl., April 17-21 1989.*
- 32 Overbeek P, Sipman W, "INFORMATIEBEVEILIGING - een praktische gids voor de bescherming van uw gegevens", ISBN 90-72194-32-4, uitgeverij Tutein Nolthenius - Amsterdam, in Dutch, September 1992
- 33 Overbeek P, "Beveiligings - en capaciteitsaspecten van local area networks", *FEL IR* 1986-17 (2 parts), in Dutch, 1986
- 34 Overbeek P, "Secure Open Systems - An Investigation of current Standardisation Efforts for Security in Open Systems", *IFIP SEC'92, 8th International Information Security Conference, Singapore*, pp 89-102, 27-29 May 1992, also available as FEL-92-B225, June 1992
- 35 Overbeek P, "Secure Open Systems, An Investigation", *FEL-91-B293*, December 1991
- 36 Overbeek P, "Stijging computercriminaliteit zet door", *NGI Magazine* 6(9):3-5, in Dutch, 1991
- 37 Overbeek P, "Verliezen bij automatisering moeilijk te schatten", *NGI Magazine* 5(10):12-14, in Dutch, 1990
- 38 Overbeek P, "Verliezen bij informatica-gebruik waarschijnlijk gigantisch - Gebrekkig cijfermateriaal bij bestrijding", *Informatie Management *VOL*(11):31-35*, in Dutch, 1990
- 39 Overbeek P, "Informatiebeveiliging: verleden, heden en toekomst - De invloed van veranderingen in de informatie technologie", pp. 19-30., *Efficiency Beurs - Data Security Seminar, RAI Congrescentrum*, in Dutch, 4-5 oktober 1990
- 40 Overbeek P, "Information Security: Past, Present and Future - Impact of Developments in Information Technology on Security", in *SECURICOM 91 - 9th Worldwide Congress on Computer and Communications Security and Protection*, pp. 9-26, CNIT, Paris la défense, France. Groupe Blenheim, Levallois Perret Cedex/France, March 20-22 1991, also available as TNO Report FEL-91-B100, 1991
- 41 Overbeek P, "Verleden, Heden en Toekomst beveiliging in Informatie Technologie", pp. 1-15. *Colloquium computer- en communicatiebeveiliging - Themadag 1991, Philips Security Office, Evoluon Eindhoven. Philips, Eindhoven*, in Dutch, 1 oktober 1991
- 42 Parker, T.A, "Secure European System for Applications in a Multi-vendor Environment (The SESAME Project)", *DoD Security Conference*, November 1991



D.W. Fikkert
(project manager)



ir. H.A.M. Luijff
(supervisor)



ir. P.L. Overbeek
(author)

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

(MOD-NL)

1. DEFENSE REPORT NUMBER (MOD-NL) 2. RECIPIENT'S ACCESSION NUMBER 3. PERFORMING ORGANIZATION REPORT NUMBER
TD92-3313 FEL-92-B365

4. PROJECT/TASK/WORK UNIT NO. 5. CONTRACT NUMBER 6. REPORT DATE
20555 NOVEMBER 1992

7. NUMBER OF PAGES 8. NUMBER OF REFERENCES 9. TYPE OF REPORT AND DATES COVERED
117 (EXCL RDP & DISTR.LIST) 42 FINAL

10. TITLE AND SUBTITLE
MODELS AND BUILDING BLOCKS FOR
SECURE OPEN SYSTEMS

11. AUTHOR(S)
OVERBEEK P.L.

12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
TNO PHYSICS AND ELECTRONICS LABORATORY,
P.O. BOX 96864, 2509 JG THE HAGUE, THE NETHERLANDS
OUDE WAALSDORPERWEG 63, 2597 AK THE HAGUE, THE NETHERLANDS

13. SPONSORING/MONITORING AGENCY NAME(S)
TNO PHYSICS AND ELECTRONICS LABORATORY, THE HAGUE, THE NETHERLANDS

14. SUPPLEMENTARY NOTES

15. ABSTRACT (MAXIMUM 200 WORDS, 1044 POSITIONS)
THIS REPORT INVESTIGATES TECHNICAL SECURITY IN OPEN SYSTEMS. SECURITY IN OPEN SYSTEMS IS A SPECIAL PROBLEM SINCE EACH ELEMENT IN AN OPEN SYSTEM (HARDWARE, NETWORKS, OPERATING SYSTEMS AND APPLICATIONS) MUST BE ABLE TO OFFER SECURITY IN COORDINATION WITH OTHER ELEMENTS. CONCEPTS FOUND IN PRECEDING STUDIES ARE FURTHER DEVELOPED AND NEW CONCEPTS ARE INTRODUCED. THE CONCEPTS OF CURRENT INTERNATIONAL STANDARDS ARE GENERALISED SUCH THAT THEY ARE APPLICABLE TO OPEN SYSTEMS. FURTHERMORE, THE DIFFERENT CONCEPTS ARE CONFRONTED WITH ONE ANOTHER AND UNIFIED USING NEW MODELS.

THIS STUDY OFFERS MODELS AND BUILDING BLOCKS TO CREATE SECURE OPEN ELEMENTS THAT TOGETHER WILL MAKE UP A SECURE OPEN SYSTEM. THE STUDY SHOWS THAT IT IS POSSIBLE TO OFFER SECURITY SERVICES IN OPEN SYSTEMS THAT ARE ABLE TO FULFIL THE MAJORITY OF TODAY'S AND TOMORROW'S REQUIREMENTS FOR SECURITY. THE STUDY ALSO SHOWS THAT THERE STILL IS A LONG WAY TO GO: INTERFACES, PROTOCOLS AND STANDARDISED DATA STRUCTURES MUST STILL BE DEFINED. HOWEVER, NONE OF THIS IS BEYOND THE STATE-OF-THE-ART. FURTHERMORE, THE STUDY SHOWS THAT THERE IS A VIABLE EVOLUTIONARY TRACK FROM TODAY'S PRACTICES AND STANDARDS, TOWARDS TOMORROW'S SECURE OPEN SYSTEMS.

16. DESCRIPTORS
INFORMATION SYSTEMS
SECURITY
DISTRIBUTED NETWORKS
STANDARDISATION

IDENTIFIERS

17a. SECURITY CLASSIFICATION
(OF REPORT)
UNCLASSIFIED

17b. SECURITY CLASSIFICATION
(OF PAGE)
UNCLASSIFIED

17c. SECURITY CLASSIFICATION
(OF ABSTRACT)
UNCLASSIFIED

18. DISTRIBUTION/AVAILABILITY STATEMENT
UNLIMITED

17d. SECURITY CLASSIFICATION
(OF TITLES)
UNCLASSIFIED

UNCLASSIFIED

Distributielijst

- 1 Hoofddirecteur TNO-Defensieonderzoek
- 2 Directeur Wetenschappelijk Onderzoek en Ontwikkeling
- 3 HWO-KL
- 4 HWO-KLu
- 5 HWO-KM
- 6 Technische Universiteit Delft, t.a.v. Prof. Dr. I.S. Herschberg
- 7 Technische Universiteit Delft, t.a.v. Drs. J.W.J. Heijnsdijk
- 8 - 10 TDCK
- 11 Directie FEL-TNO, t.a.v. Ir. P. Spohr
- 12 Directie FEL-TNO, t.a.v. Dr. J.W. Maas, daarna reserve
- 13 Archief FEL-TNO, in bruikleen aan Ir. F.G.J. van Aken
- 14 Archief FEL-TNO, in bruikleen aan Ir. J. Bruin
- 15 Archief FEL-TNO, in bruikleen aan D.W. Fikkert
- 16 Archief FEL-TNO, in bruikleen aan Ir. G.H. Heebels
- 17 Archief FEL-TNO, in bruikleen aan Ir. H.A.M. Luijff
- 18 Archief FEL-TNO, in bruikleen aan Ir. P.A.C.M. Meulemans
- 19 Archief FEL-TNO, in bruikleen aan Ir. P.L. Overbeek
- 20 Documentatie FEL-TNO
- 21 - 28 Reserves

Indien binnen de krijgsmacht extra exemplaren van dit rapport worden gewenst door personen of instanties die niet op de verzendlijst voorkomen, dan dienen deze aangevraagd te worden bij het betreffende Hoofd Wetenschappelijk Onderzoek of, indien het een K-opdracht betreft, bij de Directeur Wetenschappelijk Onderzoek en Ontwikkeling.