DR. IR. PAUL L. OVERBEEK
TNO DEFENSIEONDERZOEK FEL
THE NETHERLANDS

TOWARDS SECURE OPEN SYSTEMS *C3*

Towards Secure Open Systems

An Architecture for Security in Open Systems

Author:
Dr. Paul L. Overbeek

TNO Physics and Electronics Laboratory
P.O. Box 96864
NL 2509 JG The Hague
The Netherlands

Phone: ++31 70 3264221
Fax: ++31 70 3280961
E-mail: overbeek@fel.tno.nl

Abstract

The rise of open systems offers new opportunities for information security. Today, we are in an excellent position to 'repair' some of the mistakes made in the past, and develop future-proof open information systems, offering affordable security in any respect.

This study offers an architecture and building blocks to create secure open elements that together make up a secure open system. The aim of this effort is to provide a reference framework for the discussion of security in an open systems environment. It is shown 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. Furthermore, the study shows that there is a viable evolutionary path starting from today's practices and standards.

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.

Towards Secure Open Systems

An Architecture for Security in Open Systems

Abstract

The rise of open systems offers new opportunities for information security. Today, we are in an excellent position to 'repair' some of the mistakes made in the past, and develop future-proof open information systems offering affordable security in any respect.
This study offers an architecture and building blocks to create secure open elements that together make up a secure open system. The aim of this effort is to provide a reference framework for the discussion of security in an open systems environment (OSE, but applicable as well for initiatives like GULS, POSIX, SIOs, security APIs and many more). It is shown 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. Furthermore, the study shows that there is a viable evolutionary path starting from today's practices and standards.
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.

1. Introduction

Currently, there is a drive towards *open systems*: systems composed of elements that are not proprietary with respect to one specific manufacturer. The elements of such an open system conform to properly defined interfaces, services and protocols. Following the literature, the term *open* is used in this study in combination with elements of a system like hardware, networks, operating systems, databases, and other applications.

Security

The elements of an open system must not only be able to coexist with one another but must also be able to benefit from one another and offer a concerted 'value added' effort. This also implies the coordination of security between the elements of an open system and consistency of security within an element.
It must be assumed, and this is not specific to *open* systems, that the information-technology infrastructure is shared with unreliable and unpredictable participants (computers, networks, users, and software). Currently, information flow is not restricted to one specific computer system or network and not even to any specific application. Information security must secure the information at all times and ubiquitously. Therefore all information flow also must be secure. In order to achieve this in an open-

systems environment, all elements of the open system must seamlessly fit together: standardisation is essential.

This paper is build upon earlier studies summarised in [8]. In section 2 some of the 'lessons learned' of these studies are briefly introduced. These viewpoints are used in section 3 to define an architecture for security in open systems. In section 4 an evolutionary path is given, starting from today's practices and standards towards tomorrow's secure open systems

2 Lessons learned...

The rise of open systems offers new opportunities for information security. Today, we are in an excellent position to build upon our experience and develop future-proof open information systems, offering affordable security in any respect. This will only succeed if we are prepared to learn from the past.

2.1 Why security requirements have changed and will continue to do so

Information Technology is changing

Developments in information technology lead to changing needs for information security: distribution, decentralisation, mobility and networking are some of the key words. Some of the consequences for security are: shifts in the organisational security, an increasing impact of technical security, and a decreasing effectiveness of physical and procedural security. However, the technology for information security has not advanced as quickly as information technology in general...

The usage of IT has changed

Not only the technology is changed. The usage of IT has changed as well. IT is used for quite different purposes than, say, twenty years ago. As a consequence of this new use of IT, new security requirements have emerged. In the specification of security requirements, the following viewpoints should be considered:

1. Requirements for security imposed by society:

All organisations are part of social structures and are influenced by external relations. This will result in consequential security requirements for the information systems. Examples are: anonymity, accountability, proof of a transaction, privacy, proof of proper functioning, and legal proof. Furthermore, special attention is required for security in information systems in safety-critical situations.

2. Security must fit an organisation:

It must be possible to represent organisational structures and the users' real-life tasks and responsibilities in the system. To be considered are: the representation of an employee in the system; the mapping of tasks to services by which information can be accessed and/or handled; the mapping of responsibilities to rights and duties in the system; the mapping of management tasks to services in the system.

3. Security requirements regarding the value of information:

Information and services are valuable and therefore need be secured. Their appraisal is determined by the properties confidentiality, integrity, and availability. The security measures must reflect the appraisal of the information. In the chain of security measures that address a possible security event we recognise measures that aim at prevention of an event, reduction of the consequential losses of an anticipated event, detection, repression and correction of a security event, and evaluation of security events.

The requirements for information security have changed and will continue to do so. If information security does not keep up with these developments, the practical use of new technologies will be seriously hindered, if not stopped.

2.2 Standardisation efforts

There is a tremendous amount of standardisation effort going on in the area of security [6]. However, the many different standardisation initiatives are insufficiently coordinated. The consequence is that the initiatives are fragmented and that they hardly fit together (yet, there are some positive exceptions). Furthermore, the set of security requirements that is addressed by the standards is limited and none of the standardisation initiatives addresses all security requirements for security in open systems.
A lack of integration has been found between application security, operating-system security, and network security. For the purpose of integration of security, a unifying architecture for security functionality and interfaces is needed that crosses the borders of applications, operating systems, and networks.

2.3 Trust assumptions

Many of the common security models are built upon non-explicit assumptions about the security that is in effect to protect the environment in which the model operates. It is vital that all 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 as well as counteract the threats in the environment of the system.

2.4 Security domains

Those entities which can be regarded as a whole, seen from both a technical standpoint as well as the standpoint of the security policy, are grouped together in a security domain. 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 types of security, namely horizontal en vertical security.

2.5 Sessions as a bases for security management

A session is the life span of an event that can be managed individually. Different stages in a session are identified as session establishment, management during a session, and session release. Security services can be alloted to the different stages in order to fulfil the security requirements identified earlier. Furthermore, the use of sessions as a security-management concept offers a basis for a wider range of security requirements than the requirements that can be fulfilled using access control only.

2.6 Methods for providing security

Only two fundamentally different methods for providing technical security in a domain exist. The first method is based on centralised security functionality. The second method uses decentralised (distributed) security functionality. These two methods do not exclude one another but can be used in conjunction. The centralised method for evaluation of requests and provision of security can be used within one domain as well as between a domain and its subdomains. The distributed method can be used between peer domains. The distributed method can be built upon the centralised method. In this way, within the superdomain, seen as the composition of all entities in the peer domain sets, 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.

3. An architecture for security in open systems

3.1 Introduction

In this section an integrated model is presented which serves to illustrate how the viewpoints of section 2 can be fitted together to create a unifying 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 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 this unifying architecture are discussed.

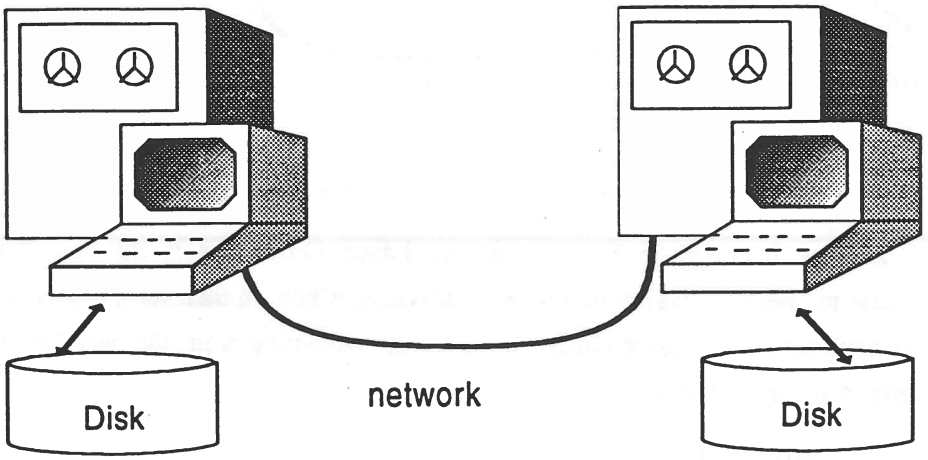

Figure 1: Two computer systems in a network: physical view

3.2 The system: what does it look like?

Figure 1 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 2 (next page) shows the logical view on the same systems in the network. The figure shows applications, operating systems, network processes and information. 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 to 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 2 shows an intermediate layer between the applications and the network process. This intermediate layer may be independent or added to the operating system. It may offer many services, but essential with respect to this section is that this layer assists in providing security and

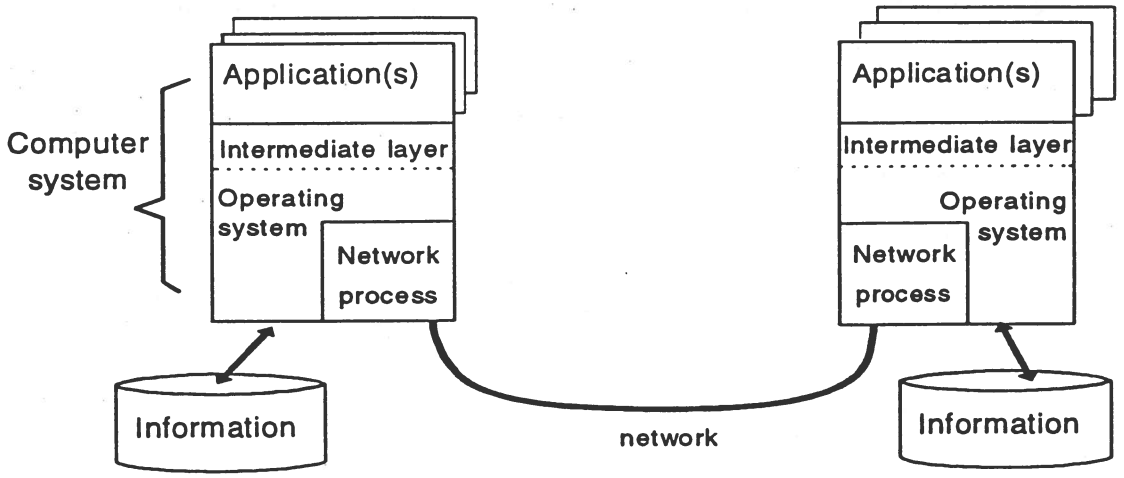

Figure 2: Two computer systems in a network: logical view

therefore is called here 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.

3.3 What is trusted?

In section 2.3 it was stated that with a better understanding of what can be trusted in a system, a more balanced set of security measures can be applied. It is important to note the dependencies within the system:

- The trustworthiness of the operating system also depends on that of the hardware and the environment in which it resides.
- The trustworthiness of applications also depends on that of the operating system and, as a consequence, on that of the hardware and environment.
- The trustworthiness of network processes at each computer system also depends on that of the operating system and that of the hardware and environment.

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

3.4 Distribution, trust relationships, and horizontal and vertical security

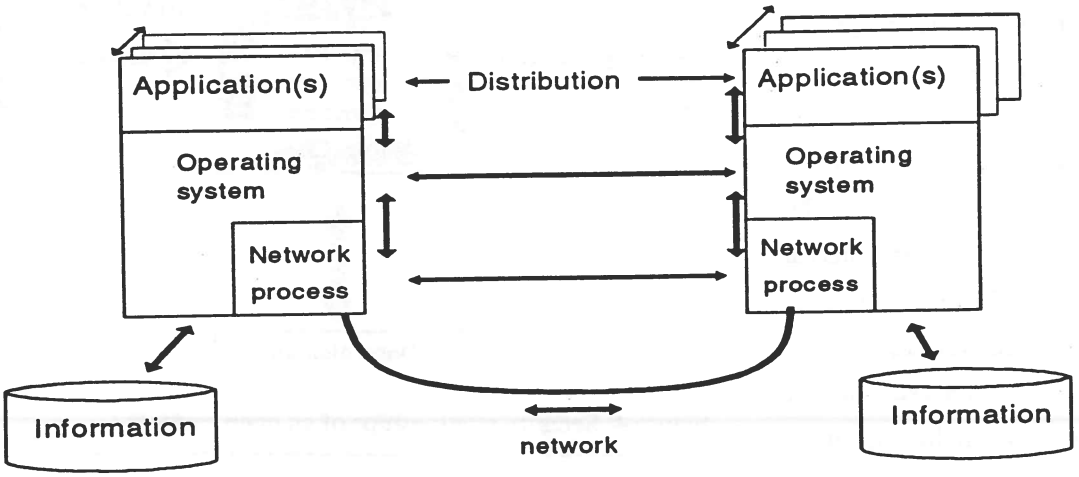

Figure 3: Horizontal and vertical flow

In figure 3 the flows of information are added to the view presented earlier in figure 2. 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 at the peer computer system goes in the upward direction, from (possibly) the network process, the operating system to the application.

This gives two flows:

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

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 4 shows the three security dimensions, focussing on open elements. Each open element may have the following three types of security functionality:

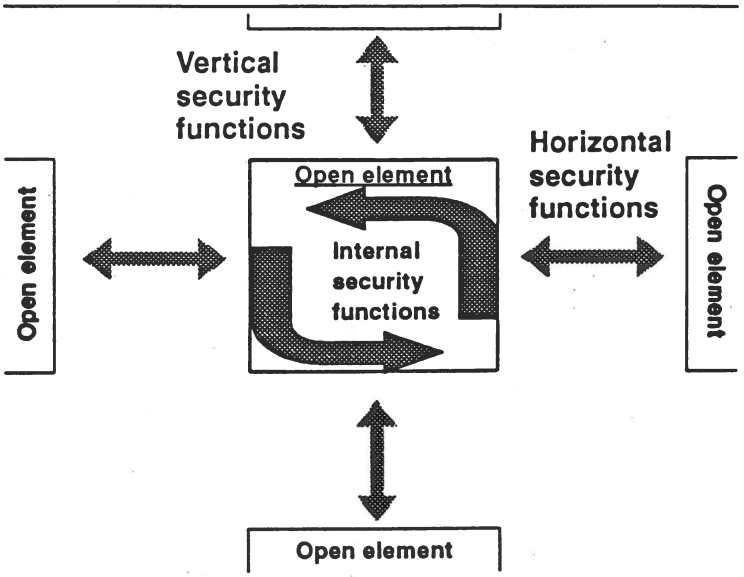

Figure 4: Security relationships of an open element

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.

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. in 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.

In figure 5 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) entirely depends on non-technical security measures such as organisational, physical, and procedural security measures.

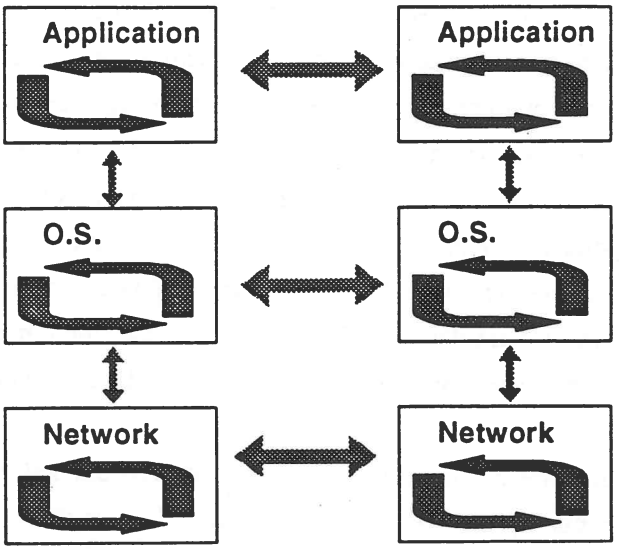

Figure 5: Horizontal and vertical security relationships between elements of an open system

3.5 Domains in an open system

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. Consequently, one application may provide the security for none, one, or several security domains.
- The operating-system security domain: each operating system corresponds to one operating-system security domain. In most cases, there is only one operating-system security domain per computer system.
- The network security domain: each network process corresponds to one network security domain. In most cases, there is only one network security domain per computer system.

3.5.1 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 one another's equals: the peer domain relationship. The first relationship requires vertical security, the second horizontal security.

3.5.1.1 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 'application' 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. 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 degree of 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. Vertical security includes delegation and inheritance. Both take place when the subdomain is created (in the session-establishment stage, see later on).

- 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.

3.5.1.2 The peer domain relationship and horizontal security

The security between peer domains is based on horizontal security. 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.

3.6 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. Three stages in a session can be identified: 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. This 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. 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 with 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.

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.

3.7 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.

3.7.1 Two methods for offering security

In section 2.6, the two principal methods to offer security were introduced. The following types of functionality are 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).

These types of functionality must be available in each security domain.

Centralised security functionality

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 security evaluator/provider acting as one whole. The security provider is designed in such a way that all requests will be mediated by the evaluator. The outcome of the evaluation (the decision) is enforced by the security provider. All security information on which an evaluation can be based is under the control of the security evaluator/provider. 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. The entities in the domain are either passive or active. In the centralised method, there is one trusted entity, being the security evaluator/provider.

To summarise: in the centralised method there is one security evaluator/provider that controls all security information, mediates all requests and provides all the required security. The centralised method can be used to provide vertical security.

Distributed security functionality

The second method for offering security uses distributed evaluation of requests and distributed security functionality. Essential in the distributed evaluation of requests and provision of 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. The distributed method can be used to provide horizontal security.

The two methods can be combined to offer security across all possible domain boundaries.

3.7.2 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 consultation of the security provider, that sufficient resources are available to offer the required security services before a request is granted.

3.7.2.1 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.

3.7.3 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. 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.

3.7.4 *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. 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 provision of security within a security domain (created by an open element).
  Internal security offered by centralised evaluation and provision of 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.

3.8 What is achieved: potentials and limitations

A security architecture is defined crossing the borders of applications, operating systems, and networks. Following this architecture it is possible to create secure open systems that are capable of:

- Fulfilling a wider range of 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.

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 various elements in an open system must be determined. Information about trustworthiness must be made available to the trusted elements in the system. 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 should be updated.
- As indicated above, trust relationships between peers can only be established between peers that know one another to be trustworthy and that can prove their identity (directly or via a trusted third party), e.g. by demonstrating their credentials.
- 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.

4. Conclusion: Towards Secure Open Systems

This section formulates an evolutionary approach from today's practices 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.

4.1 Starting point

Figure 6 shows the starting point for a growth path and the use of security standards therein.

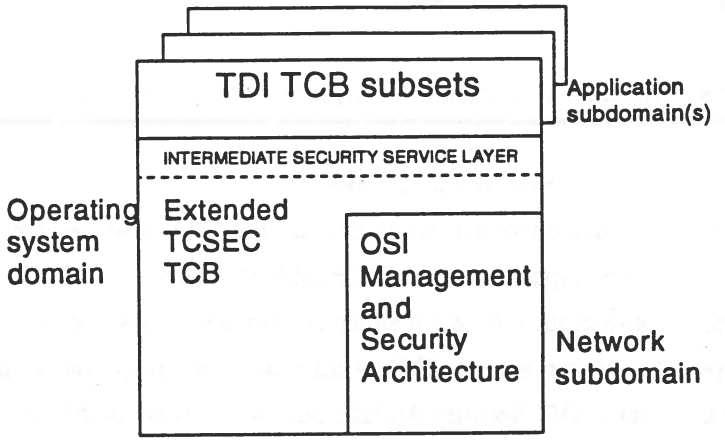

Figure 6: Security standards per computer system: first step

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 [2] with the extensions as proposed in the TDI [3]. 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 delegate security tasks to its subdomains.

Also added to the TCB is the *Intermediate Security Service Layer*. Through the reference monitor, this layer will mediate all requests for operating-system and network services stemming from the applications, and mediate all requests 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 by the network subdomain).

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 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 may be based on OSI management (although some additional functionality is required). The horizontal security between network subdomains is based on the OSI Security Architecture.

4.1.1 *What does this 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.

4.2 Future steps

The following steps should follow, each having equal priority.

4.2.1 *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 special 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 [4]).
- A common security language to exchange security information, starting with standardised data structures for security information (also see [5], drafts of standardised data structures for security information, the so-called Security Information Objects).

Next, horizontal security services must be standardised, requiring protocols and interfaces.

4.2.2 *Vertical security*

Domains and their subdomains 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.

4.2.3 *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.

4.2.4 *Enrichment of security services*

The security standards mentioned in the previous subsections address only part of the security requirements as identified in section 2. These (and other) security standards should be enriched to address a wider range of security requirements.

4.3 Conclusion

The preceding studies [6] and [7] 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 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 the state of the art. Furthermore, the study indicates that there is a viable evolutionary path starting from today's practices and standards, so there is no need to start all over again from scratch.

Acknowledgments

The author would like to thank Prof. Dr. I.S. Herschberg, J.W.J. Heijnsdijk (Delft - University of Technology), and H.A.M. Luijf (TNO Physics and Electronics Laboratory) for their valuable support during the studies and in the preparation of this paper.

5. References

1 *Information Processing Systems - Open Systems Interconnection - Basic Reference Model*, ISO 7498:1984

2 *DoD Trusted Computer System Evaluation Criteria*, DoD 5200.28 STD, (also known as TCSEC or Orange Book), December 1985

3 *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, (also known as TDI), USA NCSC-TG-021, lib.no. S235.625, April 1991

4 *Information Technology - Open Systems Interconnection - Security Frameworks for Open Systems - Part 2: Authentication Framework*, ISO/IEC DIS 10181-2, 1991-07-18

5 *Security Information Objects (SIOs)* ISO/IEC/JTC1/SC27

6 Overbeek P.L, *Secure Open Systems, An Investigation*, FEL-91-B293, December 1991

7 Overbeek P.L, *Information Security: Past, Present, and Future - Impact of Developments in Information Technology on Security*, pp. 9-26. In SECURICOM 91 March 20-22 1991, also available as TNO Report FEL-91-B100

8 Overbeek P.L, *Towards Secure Open Systems*, ISBN 90-9005824-9, 2nd ed, July 1993