A Method for Assessment of System Security

Master’s thesis

in Information Theory

by

Rikard Andersson

LiTH-ISY-EX--05/3745--SE
Linköping, 2005
A Method for Assessment of System Security

Master's thesis
in Information Theory,
Department of Electrical Engineering,
at Linköpings universitet

by

Rikard Andersson

LiTH-ISY-EX--05/3745--SE
Linköping, 21 September 2005

Supervisors: Jonas Hallberg
Amund Hunstad
Swedish Defence Research Agency

Examiner: Associate professor Viiveke Fåk
Linköpings universitet
A Method for Assessment of System Security

Rikard Andersson

With the increasing use of extensive IT systems for sensitive or safety-critical applications, the matter of IT security is becoming more important. In order to be able to make sensible decisions about security there is a need for measures and metrics for computer security. There currently exist no established methods to assess the security of information systems.

This thesis presents a method for assessing the security of computer systems. The basis of the method is that security relevant characteristics of components are modelled by a set of security features and connections between components are modelled by special functions that capture the relations between the security features of the components. These modelled components and relations are used to assess the security of each component in the context of the system and the resulting system dependent security values are used to assess the overall security of the system as a whole.

A software tool that implements the method has been developed and used to demonstrate the method. The examples studied show that the method delivers reasonable results, but the exact interpretation of the results is not clear, due to the lack of security metrics.

Keywords
Computer security, Metrics, Assessment, Information systems, Modelling
Abstract

With the increasing use of extensive IT systems for sensitive or safety-critical applications, the matter of IT security is becoming more important. In order to be able to make sensible decisions about security there is a need for measures and metrics for computer security. There currently exist no established methods to assess the security of information systems.

This thesis presents a method for assessing the security of computer systems. The basis of the method is that security relevant characteristics of components are modelled by a set of security features and connections between components are modelled by special functions that capture the relations between the security features of the components. These modelled components and relations are used to assess the security of each component in the context of the system and the resulting system dependent security values are used to assess the overall security of the system as a whole.

A software tool that implements the method has been developed and used to demonstrate the method. The examples studied show that the method delivers reasonable results, but the exact interpretation of the results is not clear, due to the lack of security metrics.
Acknowledgements

I would like to thank my supervisors Jonas Hallberg and Amund Hunstad at the Swedish Defence Research Agency and my examiner Viiveke Fäk at the Linköping Institute of Technology for help, support and ideas during the work with this thesis. Finally, I would like to thank my opponent and fellow thesis worker Denis Gacic for valuable comments and discussions.

Rikard Andersson
Linköping, 21 September 2005
Contents

1. Introduction ................................................................................................................. 1
   1.1. Motivation .................................................................................................................. 1
   1.2. Problem formulation ................................................................................................. 1
   1.3. Limitations ................................................................................................................ 2
   1.4. Contributions ............................................................................................................ 2
   1.5. Disposition ................................................................................................................ 3

2. Background .................................................................................................................... 5
   2.1. IT Security ................................................................................................................ 5
   2.2. Distributed Systems ................................................................................................ 6
   2.3. Security Metrics ....................................................................................................... 6

3. Security assessment ...................................................................................................... 9
   3.1. System security assessment ..................................................................................... 9
       3.1.1. Security factors .................................................................................................. 9
       3.1.2. Attack modelling .............................................................................................. 10
       3.1.3. Penetration testing .......................................................................................... 11
   3.2. Common Criteria ..................................................................................................... 11
   3.3. The Heimdal Framework ......................................................................................... 12
   3.4. The CAESAR method ............................................................................................. 13

   4.1. System modelling ...................................................................................................... 17
       4.1.1. Components ....................................................................................................... 18
       4.1.2. Relations .......................................................................................................... 19
   4.2. Assessment ............................................................................................................. 20
       4.2.1. Neighbour-dependent security profile ............................................................. 22
       4.2.2. Logical security profile .................................................................................... 26
       4.2.3. System-dependent security profile .................................................................. 26
       4.2.4. Overall security value ...................................................................................... 27
   4.3. Discussion ................................................................................................................ 27
       4.3.1. Main developments in relation to CAESAR ..................................................... 27
       4.3.2. Parameters ........................................................................................................ 28
       4.3.3. Strengths ........................................................................................................... 28
       4.3.4. Weaknesses ..................................................................................................... 28

5. Examples ....................................................................................................................... 29
   5.1. Testing by example ................................................................................................... 29
   5.2. Parameters ............................................................................................................... 30
   5.3. Example 1: Bypassed firewall ............................................................................... 31
   5.4. Example 2: Partitioned network ............................................................................. 32
   5.5. Discussion ............................................................................................................... 34
6. Conclusions .................................................................35
   6.1. Discussion........................................................................35
       6.1.1. The set of security features ........................................35
   6.2. Future work......................................................................36
       6.2.1. Developing the set of security features .........................36
       6.2.2. Proper testing..............................................................36
       6.2.3. Hierarchical modelling.................................................36

References .............................................................................39

Appendix A: Common Criteria .................................................41
   Common Criteria...................................................................41

Appendix B: The ROME2 software ..........................................45
   System requirements.............................................................45
   The main window....................................................................45
   The main menu.....................................................................45
   The status bar.......................................................................46
   The system information panel.................................................46
   The component information panel.........................................47
   The workspace......................................................................47
   The component properties window........................................48
   The component evaluation window........................................48

Appendix C: Example relations .................................................51
   Example relations..................................................................51
Figures

Figure 1. Overview of the Heimdal framework.............................................................. 12
Figure 2. An example of a graphical representation of a CAESAR model. .......... 14
Figure 3. The CAESAR evaluation algorithm’s main concepts and their relations.................................................................................................................. 15
Figure 4. A MASS network model..................................................................................... 18
Figure 5. Overview of the concepts of MASS. ............................................................... 21
Figure 6. Example of connecting two traffic generators. ........................................... 22
Figure 7. Example of a traffic mediator in series with a traffic generator. ............ 24
Figure 8. An example of multiple paths........................................................................ 25
Figure 9. A simple network connected to the internet .............................................. 31
Figure 10. The simple network with the firewall bypassed.................................... 32
Figure 11. A simple interconnected network............................................................. 33
Figure 12. The overall security value (OSV) as a function of the security profile (SP) and filter profile (FP). ............................................................... 33
Figure 13. Class decomposition diagram (CC, 1999). ............................................. 41
Figure 14. Specifications of Protection Profile (PP). ................................................. 43
Figure 15. Specifications of a Security Target (ST). .................................................. 44
Figure 16. The main window of ROME2................................................................... 46
Figure 17. The workspace menu ................................................................................ 47
Figure 18. The component properties window........................................................... 48
Figure 19. The component evaluation window ......................................................... 49
## Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>The set of security features used in the examples.</td>
<td>30</td>
</tr>
<tr>
<td>Table 2</td>
<td>Example traffic generator profiles.</td>
<td>30</td>
</tr>
<tr>
<td>Table 3</td>
<td>Example traffic mediator profiles.</td>
<td>31</td>
</tr>
<tr>
<td>Table 4</td>
<td>The system-dependent security profiles.</td>
<td>32</td>
</tr>
<tr>
<td>Table 5</td>
<td>Example neighbour relation for traffic generators.</td>
<td>51</td>
</tr>
<tr>
<td>Table 6</td>
<td>Example system relation.</td>
<td>52</td>
</tr>
</tbody>
</table>
1. Introduction

“Security research is sometimes referred to as the ‘Humanities of Computer Science’ because, too frequently, ‘secure’ systems are built using equal measures of folklore and black arts.” (Greenwald et al, 2003)

Computer security is not currently a science (Greenwald et al, 2003). The main reason for this is a lack of reasonable measures and metrics. Without ways of measuring and quantifying computer security, it cannot evolve into a proper science. This thesis presents an effort to provide an assessment method that can be used to determine the security qualities of distributed information systems.

1.1. Motivation

With the increasing use of extensive IT systems for sensitive or safety-critical applications, the matter of IT security is becoming more important. IT systems must be adequately secured, but must also be effective and cost-efficient. An IT system must have an appropriate level of security; too weak security leaves the system vulnerable to attacks and too strong security is excessively expensive and restricts the usability of the system. In order to know what level of security is appropriate there is a need for a method for measuring or assessing the security of a system.

There currently exists no established methods to assess the security of distributed information systems. Some approaches have been proposed and some research has been done, but most have focused on single components and on organisational or managerial aspects. Not much work has been done about technical aspects or about networked computer systems (ACSA, 2002).

1.2. Problem formulation

This thesis aims at developing a method for assessing the security of distributed information systems.
The problem can be divided into two distinct parts; finding a suitable metric for describing the security relevant characteristics and finding a method for assessing the security of the system according to that metric. The approach taken in this thesis is based on a structural approach to system security assessment. This approach is based on the hypothesis that:

“With the knowledge of:
1. the security values of all security-relevant system entities and
2. all security-relevant relations between system entities,
security values for the corresponding system may be decided.”

(Hallberg et al, 2004)

That is, the security values of a system depend on the security values of the components of the system and on the structure of the system.

The thesis is based on previous work carried out at the Swedish Defence Research Agency and Linköpings Universitet, primarily the master’s theses by Peterson (2004) and Bond & Pählsson (2004a).

1.3. Limitations

The most significant limitation of this thesis is that it does not explore the issue of suitable metrics. The presence of a good metric is assumed, and only very briefly discussed. This also means that the finer details of the representation of the security are not investigated. Some assumptions are made about the form and meaning of the security value of a component (see section 4.1.1) but the details are not elaborated upon.

Another significant limitation is that this thesis is not concerned with the assessment or measurement of the security of components. It is simply assumed that such assessment is possible and a method can be developed that produces viable results.

Further, this thesis is limited to the consideration of technical aspects. Organisational, operational or other non-technical aspects are not treated at all. The method that is presented may be applicable to those aspects as well, but that is not explored.

1.4. Contributions

The main contribution of the work described in this thesis is a method for assessing the security of distributed information systems, called MASS. MASS is a development of the CAESAR method (Peterson, 2004) and improves it primarily by introducing a vector representation of security characteristics and thus the ability to model more complex relations. This also enables a better modelling of physical connections. The actual physical connections are abstracted from, and instead they are modelled as sets of logical relations between the elementary security values of the components. MASS also significantly improves the modelling of logical relations, making it possible to model more specific effects of the relations.
Concerning security metrics, the method introduces a structure for the correlation of measurable security characteristics, thus achieving compound security values. However, how to specify these characteristics or how to interpret the resulting combined values is not treated.

1.5. Disposition

Chapter 2 presents a general background to the area of IT security and some of the general theory of the field. Relevant work in the area of assessment and evaluation is presented in chapter 3, which also gives in-depth descriptions of the methods that forms the basis of this thesis.

The developed method is presented in chapter 4, as well as design choices and a discussion of the strengths and weaknesses of the method. Some tests and their results are presented in chapter 5.

Finally, chapter 6 contains the conclusions of the thesis and suggestions for future work.

The appendices concern the Common Criteria (Appendix A), the ROME2 software that has been developed to demonstrate the method (Appendix B) and the data used to demonstrate the method (Appendix C).
2. Background

This chapter presents a general background to the thesis and covers the theoretical basics of the area.

2.1. IT Security

Security in general is about the protection of assets. In IT security, it is limited to the types of assets contained in information systems. The most obvious asset is the data in the system, but also services delivered by the systems or resources such as communications bandwidth or data storage space are included. There are many ways of defining what IT security is about; in this thesis, the following definition will be used:

“IT security deals with the prevention and detection of unauthorised actions by users of a computer system.” (Gollmann, 1999)

IT security is very often considered to consist of three different aspects: confidentiality, integrity and availability, commonly referred to as CIA (Gollmann, 1999; Bishop, 2003). Different authors have slightly different definitions, but for this thesis, the following is used:

- **Confidentiality** deals with preventing unauthorised disclosure of information. Unauthorised users should not be able to learn classified or sensitive information.

- **Integrity** deals with preventing unauthorised modification of information. It should not be possible for any user (even authorised ones) to corrupt, distort or falsify information in the system.

- **Availability** deals with the prevention of unauthorised withholding of information or resources. The assets of the IT system should be accessible and usable to authorised users when needed and without undue delay.
The CIA aspects specify what should be protected. Another view of IT security is to specify how to protect the assets. Three common aspects of this view are prevention, detection and reaction, commonly referred to as PDR (Gollmann, 1999). For this thesis, the following is meant:

- **Prevention** deals with measures that prevent the assets from damage.
- **Detection** deals with measures to detect damage to the assets or to detect attempts at damaging the assets.
- **Reaction** deals with measures to recover from damage to the assets or to stop a detected attempt at damaging the assets.

Survival is also sometimes mentioned as an additional function and it deals with the ability of the system to recover from failures or security breaches or to continue to function (possibly with limited capacity) despite failure or security breach.

The concept of security is also related to the concepts of reliability and dependability. For example, the availability of a system is clearly dependent on aspects such as fault tolerance, which traditionally is not considered a security issue. There is no perfectly clear boundary defining the area of IT security.

### 2.2. Distributed Information Systems

There are several possible ways to define a distributed system, focusing on different aspects such as physical location, information or services. For this thesis, the following definition is used:

> “...a distributed system is a computing system where the resources are spatially distributed and connected by some kind of network.” (Hallberg & Hunstad, 2002)

This is in contrast to a centralized system where the resources are gathered at a single location. Distributed information systems are often complex and dynamic, which makes security complicated.

### 2.3. Security Metrics

Security metrics are concerned with measuring the security of a system and of interpreting the measures. Thus, a metric is not just a collection of objective measurements, but also a framework for interpreting those measures. Due to the complex nature of system security, a number of different system properties have to be considered for a metric to be sufficiently complete. In ACSA (2002), it is observed that

> “No single IS [metric] will successfully quantify the assurance present in a system. Multiple measures will most certainly be applied and they will need to be refreshed frequently.”
The purpose of metrics is to support decision-making. Thus, metrics are derived from analysed measures and contributes to making meaningful decisions (Alger, 2001). They can also be considered as tools to improve performance and accountability (Swanson et al, 2003).

There exists several different proposed metrics (Swanson, 2001 and Schudel & Wood, 2001), aimed at different user needs and intended for different contexts. However, there is no standard or generally accepted metric for IT security, it is even so that:

“There is general agreement that no single system metric or any ‘one perfect’ set of information assurance metrics applies across all systems or audiences.” (Vaughn et al, 2002)

Many current approaches to measuring security and developing security metrics are focused on managerial or operational aspects of security. There has been little work on technical security metrics (ACSA, 2002).

Measuring security is complex. It is not possible to measure directly, so instead factors that contribute to security or consequences of security are measured (Peterson, 2004). Examples of factors influencing security are the number of users of a system or the presence of up-to-date anti-virus software. The number of detected intrusion attempts or downtime due to denial of service attacks are examples of measurable consequences of security.
3. Security assessment

Theory and related work in the area of security assessment is presented in this chapter. An overview of the Heimdal and CAESAR methods that are the bases for the current thesis is presented.

3.1. System security assessment

There are several different approaches to security assessment in general and to the assessment of system or network security. Most of the work done in the area of security assessment has been focused on management, organisation and policy. There has been considerably less work on technical aspects of security assessment (ACSA, 2002). The approaches that have been proposed include attempts to model factors influencing security, model attacks and their interaction with the system, and model vulnerabilities and exposure.

3.1.1. Security factors

One approach to assessment of system security is to look at factors that contribute to or hamper the security. Through analysis of these factors, a measure of the security of a system can be found.

Alves-Foss & Barbosa (1995) have developed a method to find a Security Vulnerability Index (SVI) that is a measure of how vulnerable the system is. The factors that contribute to system vulnerability can be classified into three types: System Characteristics, Potentially Neglectful Acts and Potentially Malevolent Acts. Factors that are found in the system are evaluated according to a set of SVI rules that assigns a Certainty Index (CI) that measures a level of certainty of system vulnerability. That is, if a certain factor is present in the system then there is CI certainty that the system is vulnerable. The SVI is then calculated from the CIs as if they were independent probabilities of system vulnerability. The SVI is not, however, a measure of the probability that the system will be compromised. The method is well suited for implementation in a rule-based expert system (Alves-Foss & Barbosa, 1995).
Another approach is that of Wang & Wulf (1997), who proposed a Security Measurement Framework. The approach assumes that a measure of the security of the components can be found and a model to find a Security Index (SI) for the system as a whole is developed. The modelling starts from high-level security attributes of the system, which are then successively refined in a tree structure down to the basic components. The security values of the lower levels are then aggregated up to the security attributes through different functions depending on the relations between them.

Both of the methods described above consider the system as a whole and then tries to capture different security relevant aspects of it. However, neither of them considers the topology or the architecture of the system.

3.1.2. Attack modelling
The security of a system can also be modelled from its ability to resist attacks. Modelling the way an attack interacts with the system to cause compromise can result in a viable assessment of system security.

A method for modelling network attacks is presented by Clark et al (2004). The basis is a state description of the network. A possible attack is modelled from what state of the system it requires to exploit an exposure and the state of the system after the successful exploit. Exploit interactions can produce sequences of events, called attack chains. The attack chains can then be used to identify potential exploits and attack paths to study the way an attack can propagate through the system. If each exploit is assigned a probability, the probability of an attack reaching each step of the attack path can be easily calculated through probability theory. A significant problem with this approach is that the number of possible states and possible exploits are huge and it thus becomes difficult to do a sufficiently complete analysis.

Oman et al (2004) presents a method for analysing the security of Supervisory Control And Data Acquisition (SCADA) systems. SCADA systems are used for process control and are common in, for example, power generation systems and chemical industry. The idea is to model the system as a graph and then use methods and algorithms from graph theory to analyse it. The different components of the system are modelled as nodes in the graph and the connections between them as edges. The edges are assigned multidimensional properties representing, for example, the type of connection and use of encryption, which can then be assigned numerical weights. With the weighted graph in place graph theory can be used to find, for example, the most vulnerable access path to a certain node. The method seems to be primarily suited for finding attack paths and studying possible attack propagation in a network, but could possibly be suitable for other kinds of analysis as well. However, the method provides a way of modelling attacks at specific targets (nodes in the graph) but it is not clear if it can be used to assess the security of the system as a whole.
3.1.3. Penetration testing

Penetration testing is used to examine the vulnerabilities of a system. The testing is (usually) done from the outside of the system and tries to determine ways of penetrating the systems defences. Two main types of penetration testing can be distinguished: vulnerability scanners and Red Teams.

Vulnerability scanners are programs that scan a system for vulnerabilities. A number of commercial or open source scanners are available and widely used to assess the state of security of systems. The major drawback of vulnerability scanners is that they can only detect known vulnerabilities; they cannot identify previously unknown vulnerabilities or variants of known ones. There are also problems with false positives. Despite their problems, vulnerability scanners are an important tool for network administrators.

A Red Team (or Tiger Team) is a group specifically tasked to test the security of a system by trying to penetrate it. The effort required by the Red Team to breach the security of a system can be used as a measure of the security of the system (Schudel & Wood, 2001). The value arrived at will be quite non-repeatable, both in the respect of using other teams to attack the same system (because their skills differ from the first team) and in using the same team to attack the system again (since they learned from the first attack). Using the same Red Team to attack different systems (or different configurations of the same system) can still give some valuable information on the relative strengths of the systems. Red Teams also suffer from the fact that it is very difficult to find previously unknown types of vulnerabilities, although they arguably are better than vulnerability scanners.

Penetration testing can never show that a system is free from vulnerabilities, only that some specific vulnerabilities are not present. Although it is imperfect, penetration testing can offer valuable insights about the security of systems. Currently, it is in use as a valid information security metric in both the government and commercial sector (ACSA, 2002).

3.2. Common Criteria

Common Criteria (CC, 2004) is a widely spread and accepted standard for evaluating the security of products. It has been adopted as an ISO international standard (ISO 15408).

The basis of CC is an extensive set of Security Functional Requirements (SFR), which is used to describe the security functionality in products. To support the evaluation of a product category a Protection Profile (PP) with the requirements of the intended application is developed. To evaluate a product its security functionality is described in a Security Target (ST), which is based on applicable Protection Profiles, if any. Thus, Security Targets describe the security functionality of products, related to their intended application. The functionality of a specific product is then compared to its ST.
CC is supplemented with a standardised method, the Common Evaluation Methodology, to carry out a CC evaluation against a set of predefined Evaluation Assurance Levels (EAL0 to EAL7). The EAL is a measure of the level of trust that the product actually meets its requirements. There is an agreement between 21 countries (in May 2005) to recognise CC Certificates issued by recognised authorities.

A more detailed description of CC is given as Appendix A.

### 3.3. The Heimdal Framework

The Heimdal Framework described in Bond & Påhlsson (2004a) is:

> “A method to get a picture of the unconditional security properties of a product as well as finding a way to scale these properties based on a number of different factors, such as the requirements of a certain product type, and the environment in which the product is intended to operate, ...” (Bond & Påhlsson, 2004a)

The Heimdal framework is used to evaluate products and receive a measure of their security, possibly in relation to the intended use and environment. It is based on the Security Functional Requirements of the Common Criteria. Figure 1 is an overview of the framework.

The security functionality of a product is described in a Target of Evaluation Profile (TOE Profile or TP). The TP is a list of Security Features (SF) corresponding to the SFRs of CC. Each SF is assigned a value of 1 if it is implemented and 0 if it is not.

![Diagram](image-url)

**Figure 1.** Overview of the Heimdal framework. (Bond & Påhlsson, 2004a)

The requirements of the product are described in a corresponding Target Category Profile (TCP). The TCP is also a list of the SFs, with each one assigned a value of 0 or 1, depending on if it is considered relevant for the intended use of the product. To represent the environment of the product an Environment Profile (EP) is developed. The EP has the same form as the TCP and TP but with values in the range [0, A], where A is a positive real number. A value of 1 represents normal risk, a value less than 1 represents low risk and a value greater than 1 represents high risk. The value
assigned each SF in the EP represents the threat to that SF in the intended environment. The TCP and EP are multiplied to arrive at a Reference Profile (RP).

Finally, the TP is weighed against the RP to get an Evaluated Target Profile (ETP). From the ETP various more generalised attributes can be calculated, such as a single average security value or values corresponding to the CIA attributes.

### 3.4. The CAESAR method

The CAESAR method presented in Peterson (2004) is a method for evaluating the security level of a distributed system, if the security levels of the constituent components are known:

> “The purpose of CAESAR is to estimate the security level of an entire distributed information system, based on the security level of, and the relations between, its included components.” (Peterson, 2004)

The method assumes that a Security Level Estimate (SLE) is available (through some other method) for each component in the system. The SLE is a real number in the range [0, 1], but what exactly it represents is not specified. For network components, such as hubs or firewalls, a Traffic Control Estimate (TCE) is assumed. The TCE is “...based on [the component’s] ideal ability to conceal malicious network traffic” (Peterson, 2004).

A distributed system is modelled based on its physical layout. The components are modelled as traffic generators (computers and public networks) or traffic mediators (firewalls, routers, proxies and hubs). The connections between the components are primarily modelled as physical relations. There is also a provision for logical relations “to be able to regard components’ communicational premises and patterns and the implications of them”. Logical relations represent special trust relationships between components, such as the relation between a server and a client. Each logical relation is assigned a Logical Significance (LS), which is a real number. The description of the logical relations is somewhat sketchy and they are not implemented in the supporting software. An example of a network model is illustrated in figure 2.

The goal of the evaluation algorithm is to calculate the Overall Security Level (OSL) of the system. To be able to do that the System-dependent Security Level (SSL) of each traffic generator is calculated. The OSL is then given as the average of the SSLs of all the traffic generators in the system.

The SSL of a component is calculated from the SLE of the component and the contributions to its security level from neighbouring components and logically related components. Neighbouring traffic generators cause a Neighbouring Security Contribution (NSC) that depends on the SLE of the neighbouring traffic generator and the SLE and TCE of all traffic mediators between the traffic generators. The Logical Security Contribution (LSC) from logically related components is calculated from the SLE of the component and the LS of the relation. An overview of the algorithm is depicted in figure 3.
Figure 2. An example of a graphical representation of a CAESAR model (Peterson, 2004).
Figure 3. The CAESAR evaluation algorithm’s main concepts and their relations (Hallberg et al., 2004)
4. A Method for Assessment of System Security

This chapter presents a Method for Assessment of System Security (MASS) for modelling and assessing the security values of distributed information systems. MASS is a development of the CAESAR method presented by Peterson (2004).

The basis of the method is that the security relevant characteristics of a component can be described by a set of security features with associated elementary security values. This set is called the security profile of the component.

The interaction between a component and its neighbouring components affects the effective security values of all the involved components. The resulting security values can be described by system-dependent security profiles. That is, to evaluate the security of a system the system-dependent security of each component is evaluated. These system dependent values are then used to assess the security value of the system as a whole.

4.1. System modelling

In MASS system models consists of components and relations between them, in basically the same way as in CAESAR. However, the MASS method can be applied on different levels of abstraction. For example, a component could represent a software product, connected to other products in the “system” of a single computer or it could represent a complete local area network connected to other computer networks in a widely distributed system of networks. In this thesis, all examples will assume that a component is a piece of hardware (computer, firewall, etc) connected in a local area network.

An example of a network modelled by the MASS method is illustrated in figure 4.
Figure 4. A MASS network model. The security profiles (SP) are made up of Audit (Aud), Access control (Acc) and Authentication (Aut) security features.

4.1.1. Components

There are two types of components: traffic generators and traffic mediators (Peterson, 2004). Traffic generators are for example computers or public networks, traffic mediators are for example hubs, firewalls and routers. The difference between the types is that a traffic generator generates traffic but cannot mediate any traffic. A traffic mediator cannot generate any traffic itself but mediates traffic between traffic generators.

The security level of a component is modelled by a security profile. A security profile is a set of security features (Bond & Pählsson, 2004a) with corresponding elementary security values. A security feature describes an aspect of security relevant functionality of the component. Its security value is a measure of the strength of the implementation of the feature in the component. The security values are in the range of [0, 1], where 0 means that the security feature is not present in the component and 1 means that it is perfect. Values in between may be qualitatively described as weak or strong, but exactly what values a security feature would require to be considered strong (or weak) is dependent on the set of security features used and on the context. The security profiles are expressed as vectors, where the value of each element in a vector corresponds to the elementary security value of one of the security features.

The security profile should describe the actual security level of the component. For instance, if a component has strong features considering audit by itself but weak features considering authentication, the result should be low security values for the audit features too. The reason for this is that the audit features depend on the authentication to identify users and the weakness of the authentication therefore degrades the audit features. These kinds of dependencies should thus be resolved before the specification of the security profile.
The security profiles of traffic mediators represents how secure they are in themselves, i.e. how hard they are to hack. Their ability to filter malicious traffic is a different property, not directly tied to their own security. For example, a hub is extremely resistant to attacks, since no configuration of it can be done remotely, but it has no effect on the traffic passing through it. Another example could be a firewall that has very strong filtering, but may be protected by a weak password and therefore susceptible to attack.

The ability of a traffic mediator to stop malicious traffic is modelled by a filter profile. The filter profile is a vector of the same security features that a security profile. Each element is a value in the range [0, 1] that represents its ability to filter traffic affecting the corresponding security features. A value of 0 means that it does no filtering and a value of 1 that no malicious traffic gets through.

In CAESAR, the security level of a component was modelled through the Security Level Estimate. Correspondingly, the security value of a component can be calculated from the security profile of MASS. It is calculated as the, possibly weighted, average of the elementary security values of all the elements of the security profile. If a weighted average were used, the weights would represent the relative importance of each of the security features for that specific component.

The method proposed here is not tied to a specific set of security features. Any set that is considered adequate can be used. Large sets can provide more detail, but are much more difficult to use while small sets are easier to use but may be to general to provide useful results. How to find an appropriate set is not explored in detail in this thesis.

Each component has to have the security values of its security features decided and all traffic mediators must have their filter profiles decided. Methods for doing so are outside the scope of this thesis. The methods will depend on the set of security features chosen. An example of a method is the Heimdal framework (see 3.3), but other methods, with other sets of security features, are possible.

4.1.2. Relations
Components are connected through relations, either physical relations or logical relations, as in CAESAR. The relations capture the security relevant effects of connecting different components.

Physical relations
The physical relations model actual connections between components through physical means, typically a cable but possibly through other media such as radio or infrared. The physical relation is bi-directional and symmetric. That is, the relation from component A physically connected to component B is the same as that from component B to component A.

The physical relations are modelled according to the physical layout and topology of the modelled system. They contain no other modelled properties except the components they are connected to.
It is difficult to model the impact of a real physical connection, since it is very much dependent on the nature of the interfaces, the services running on the components and other factors. What is relevant to the security of components is the interaction between them that a physical connection enables. In MASS, this interaction is modelled as a set of relations between the security features of the components. Thus, the actual physical connection is abstracted from, and replaced with a set of logical relations between the elementary security values of the components.

**Logical relations**

Logical relations represent special patterns of trust and communication. An example would be a Virtual Private Network, where the traffic over the VPN channel is not affected by any firewalls or routers between the end computers. Another example could be the relation between a workstation computer and the central server managing the anti-virus software.

The logical relations are uni-directional. If the real connection to be modelled goes both ways, it is modelled as a pair of logical relations, one for each direction. A logical relation is associated with a function that describes the effect the relation has on the security value of the connected component, see section 4.2.2.

The physical relations could be seen as a special case of the logical relations. They could then be considered as a pair of logical relations (one in each direction) with a specified function.

### 4.2. Assessment

Assessment of a modelled system is done in two main steps. The first step is to evaluate each traffic generator in the context of the system. The result of that evaluation is a system-dependent security profile for each traffic generator. In the second step, the calculated system-dependent security profiles of all the components are used to determine system-wide security characteristics, such as an overall security value.

The first step of evaluating the traffic generators is divided into three steps; evaluating the physical relations, evaluating the logical relations and combining those results to get the system-dependent security profile of the component. The evaluation of the physical relations also takes into account the effects of traffic mediators and multiple paths. An overview of the concepts of MASS is presented in figure 5.

The system-dependent security value of a traffic mediator is not of any interest, since it does not contain or generate any information. Therefore, only traffic generators are considered for the calculations of system-dependent security values. Two traffic generators are *neighbours* if they are connected directly or through one or more traffic mediators. They are not neighbours if all the connections between them go through other traffic generators. The neighbour concept is identical to that in CAESAR.
Figure 5. Overview of the concepts of MASS.
4.2.1. Neighbour-dependent security profile

The effect of connecting two components will be a change in the effective security value of the components. Exactly what that effect will mean is dependent on what the security features of the component mean, that is, it will depend on the set of security features chosen.

If the security features are considered as describing security functionality of the components, the functionality of the neighbouring component will have an effect. The functionality of the neighbour dominates or influences the functionality of the evaluated component. If, for example, the neighbour has weak authentication it will allow weakly authenticated users to access itself and the evaluated component it is connected to, thereby weakening the effective authentication of the evaluated component.

![Figure 6. Example of connecting two traffic generators.](image)

However, the effect of connecting two components is not as simple as a direct influence of the respective security features on each other. Consider the example in figure 6. The vectors represent the security profiles of the components. Component A has a strong audit feature (the second value), but component B has a very weak audit feature. This does not have any effect on component A, because the audits are done independently between the components. However, the weak authentication (the fourth value) of component B will be detrimental to the audit of component A. This is because the audit feature is dependent on authentication of users, and through component B, weakly authenticated users will get access to component A. The neighbour-dependent security value for the audit feature in component A would therefore be significantly lower than the independent value.

The examples indicate that the effective elementary security values of a component will depend on its own (independent) elementary security values and on some (or even all) the elementary security values of any other components it is connected to. This effective security can be represented by a neighbour-dependent security profile.

The neighbour-dependent security profile is on the same form as the (independent) security profile and represents the security level of a component in the context of a
specific system. The neighbour-dependent security profile, with \( m \) security features, is calculated through a function, \( f: [0, 1]^m \to [0, 1]^m \), of the security profiles of the connected components:

\[
\text{NSP} = f(\text{SP}, \text{SP}^1, \ldots, \text{SP}^n),
\]

where \( \text{NSP} \) is the neighbour-dependent security profile of the component being evaluated, \( \text{SP} \) is the (independent) security profile of the component, \( \text{SP}^k \) is the security profile of neighbouring component \( k \) and \( n \) is the total number of neighbours.

For a single security feature \( i \) of the component the relation becomes

\[
\text{NSP}_i = f_i (\text{SP}_i, \text{SP}_i^1, \ldots, \text{SP}_i^n)
\]

That is, the security value of each security feature in the NSP is a result of some function of that security feature in the SP of the component and of all security features in the SPs of the neighbours.

Further, it is assumed that the effects of the respective security features are independent of each other. That is, the effect from security feature \( j \) in the neighbour is independent of the effect of security feature \( i \neq j \). This is consistent with the assumption that the security features of the component themselves are independent (see 4.1.1). The resulting function would then be

\[
\text{NSP}_i = f_i (\text{SP}_i, \text{SP}_i^1, \ldots, \text{SP}_i^n) = \sum_{i=1}^{m} W_i f_i (\text{SP}_i, \text{SP}_i^1, \ldots, \text{SP}_i^n),
\]

where \( W_i \) is the relative weight of the function \( f_i \) (that is, \( \sum W_i = 1, j=1 \ldots m \)) and \( m \) is the total number of security features.

The functions \( f_i \) can be any function \( f_i: [0, 1] \to [0, 1] \). Examples of usable functions are the \textit{Min}, \textit{Max} and \textit{Average} functions.

Finding the correct function for each \( f_i \) and the correct value for each \( W_i \) is essential to produce valid results. These will depend on the set of security features chosen to represent the security of the components. The method used to find theses functions and values will also depend on the set of security features and it is probably not possible to develop a general method for it. Using common sense combined with testing against cases where the expected result is known beforehand is a viable ad-hoc method.

**Traffic Mediators**

A traffic mediator sits between traffic generators and is partly transparent to the traffic generators, see figure 7. A traffic mediator in itself has no effect on the security value of a traffic generator, but it does affect the security value if placed between traffic generators.
The ability of a traffic mediator to filter malicious traffic is represented by its filter profile. However, a traffic mediator could be attacked directly and have its filtering capabilities subverted or even turned off. Thus, the effective filtering ability of a traffic mediator is dependent on both its filtering capabilities and its own security. This is represented with an effective filter profile. The effective security profile is calculated as the filtering profile multiplied with the security value of the traffic mediator, that is:

\[ EFP = FP \cdot SV \]

where EFP is the effective filter profile, FP is the filter profile and SV is the security value of the traffic mediator (see 4.1.1).

Relevant for the security value of a traffic generator is the communicational pattern in its connections. It makes no difference if that pattern is generated by a traffic generator and then filtered by one or more traffic mediators or if it is generated directly by a traffic generator. Thus, it should be possible to model the effect of a traffic generator connected through a traffic mediator as a single equivalent traffic generator. The security profile of the equivalent traffic generator would be calculated from the security profiles of the traffic generator and the effective filter profile of the traffic mediator.

The equivalent security profile is calculated as

\[ ESP = SP + (1 - SP) \cdot EFP \]

where ESP is the equivalent security profile, SP is the security profile of the traffic generator and EFP is the effective filter profile of the traffic mediator. This calculation can be carried out iteratively for any number of traffic mediators connected in series with a traffic generator.

A path with no traffic mediator in it (i.e. a direct connection between two traffic generators) has an equivalent security profile that is identical to the security profile of the neighbouring traffic generator, that is \( ESP = SP \).
Multiple paths
A neighbouring component may be connected to the component of evaluation through more than one path (for example via a direct connection and via a connection through a traffic mediator, see figure 8). It is not obvious how this affects the security of the component of evaluation.

Figure 8. An example of multiple paths.

There are three possible ways to solve the problem: (1) apply the effects of all paths, (2) select one of the paths or (3) calculate an aggregated result. The first method seems inappropriate since it would give that component a disproportionately large effect on the security. Using the second method would mean that one of the paths are completely ignored, which does not seem reasonable since it should have some effect.

Consequently, the third approach is used in MASS. A new resulting security profile is calculated as the element-wise product of the equivalent security profiles (ESPs) of the two paths:

\[ \text{RSP}_i = \text{ESP}^1_i \cdot \text{ESP}^2_i , \]

where RSP is the resulting security profile and ESP\(^1\) and ESP\(^2\) are the profiles of the respective paths. This calculation can be applied iteratively for any number of paths.

The method is chosen since it captures the fact that two possible paths with different vulnerabilities will expose all of them, not just the vulnerabilities of one of the paths. The resulting security value should therefore be lower than either of the paths. It also has the benefit of being simple to use and understand. There could be other methods and the choice of method could depend on the context and on the set of security features used.
4.2.2. Logical security profile

Logical relations between components will have an effect on the effective security value of the components, in much the same way as the physical relations. This effect can be captured by a logical security profile. The logical security profile is calculated as:

\[ \text{LSP} = g^l(\text{SP}, \text{SP}^l) , \]

where LSP is the logical security profile for the relation, SP is the security profile of the component of evaluation and SP\(^l\) is the security profile of the logically related component. The function \( g^l \) represents the logical relation and is on the same form as the function \( f \) for calculating the physical security profile. However, the function \( g^l \) differs from the function \( f \) in that where the logical relation has no effect on the component of evaluation the corresponding value of the LSP is \( \text{Nil} \). \( \text{Nil} \) values are ignored in all subsequent calculations.

The logical security profile represents the effect of the logical relation on the security value of the component of evaluation. There is an important difference to the physical security profile in that the physical security profile represents a resulting security value, but the logical security profile represents an effect on the security value. The logical security profiles can be thought of as modifiers to the physical security profiles and the result of applying them is the system-dependent security profile.

Since most logical relations are limited in scope, but important within that scope, it is probable that most logical security profiles will have a large proportion of \( \text{Nil} \) values. For example, a central server administering the anti-virus software of workstations will only affect those security features that relate to virus protection, but will have a large impact on those features.

4.2.3. System-dependent security profile

When the neighbour-dependent security profile and all logical security profiles of a component have been calculated they are used to calculate the system-dependent security profile. The system-dependent security profile represents the total security level of the component in the context of the system. It is calculated as:

\[ \text{SSP} = h(\text{NSP}, \text{LSP}^1, \ldots, \text{LSP}^l) , \]

where SSP is the system-dependent security profile, NSP is the neighbour-dependent security profile of the component, and LSP\(^1\) to LSP\(^l\) are the logical security profiles from the /logical relations of the component and \( h \) is a function on the same form as the functions used in the calculations of the NSP and LSPs.

The function \( h \) handles the \( \text{Nil} \) values of the logical security profiles by ignoring them. If there are no logical relations to a component, its system-dependent security profile should be the same as its neighbour-dependent security profile. That is, the function \( h \) must be such that \( h(\text{NSP}) = \text{NSP} \).
The system-dependent security profile is the major result of the MASS method. It is a description of the effective security value of a component in the context of the system and can be analysed by itself or used to analyse the system as a whole.

4.2.4. Overall security value
When the method has been applied to all the components in a modelled system, the result is a system-dependent security profile for each component. These profiles can then be analysed and more conclusions about the components and the system as a whole may be drawn.

A system-dependent security value can be calculated from the system-dependent security profile in the same way as the security value of the component is calculated from the (independent) security profile, i.e. as a, possibly weighted, average. This corresponds to the system-dependent security level of CAESAR.

The system-dependent security values of the components can be used to calculate an overall security value for the system as a whole. This can be calculated as a weighted average of the system-dependent security values of the components. The weights represent the importance of each component; an important server would have a high weight and a peripheral client would have a low weight. Apart from the use of weights, it is the same calculation as in CAESAR.

Other results could also be calculated from the system-dependent security profiles, depending on the context and intended use of the assessed system. For example, it would be possible to categorize all components as having low, medium or high security values and present a summary of the number of components in each category. It would also be possible to identify the weakest component or the weakest security feature to indicate where more security measures are needed. The security relevant characteristics of the system are contained in the system-dependent security profiles. How to present this information should be decided by the specific needs of the user.

4.3. Discussion
The MASS method has been comprehensively described above, but a few points merit more discussion.

4.3.1. Main developments in relation to CAESAR
The MASS method is a development of CAESAR in several respects, but retains much of the same basic ideas. MASS keeps the fundamental idea of CAESAR that the security level of each component should be assessed in the context of the system and these results then used for assessment of the system as a whole.

The most important development is the way that the security characteristics are modelled. In CAESAR, it was done by a scalar value, the Security Level Estimate, but in MASS, it is done by a vector, the security profile. This gives a more complex
representation of the security level of components and enables modelling of more complex relations.

The modelling of logical relations is also a major difference between CAESAR and MASS. In CAESAR, the logical relations were very general and basically the same for all logical relations, except for the weight. In MASS, they can be thoroughly specified and can be different for each logical relation. They can thus capture very specific effects.

4.3.2. Parameters
Before the MASS method as presented above can be used for assessing system security the different parameters of the method has to be decided. The most important is to choose a good set of security features. What set to use or how to find a good set is not explored in this thesis, but an example is presented in chapter 5.

The set of security features determines all the other parameters, namely the function for calculating the neighbour-dependent security profiles (\( f \) in section 4.2.1) and the function for calculating the system-dependent security profiles (\( b \) in section 4.2.3).

Finding good parameters is necessary for the method to be useful and to produce valid results.

4.3.3. Strengths
One of the major strengths of the MASS method is its flexibility and adaptability. Choosing different parameters the method may be applied on different levels of abstraction and in varying contexts. There is great potential to adapt the method to fit specific user needs. However, the heavy dependence of the method on the parameters could also be seen as a weakness (see section 4.3.4).

Another strength of MASS is the modelling of logical relations. These provide a possibility to model patterns of trust and communication that cannot be captured with the physical relations. Since they can be tailored to each specific relation they should be able to the model the many different kinds of relations that exists in real systems.

4.3.4. Weaknesses
The MASS method is very dependent on the set of security features chosen. The set chosen will determine how difficult it is to find the correct functions for calculating the different security profiles and how good the results of the method will be. It is a significant effort to find a proper set of security features and the correct parameters for the method.
5. Examples

In order to illustrate the MASS method, two examples are presented in this chapter. Moreover, the examples are also used for preliminary tests of whether the method produces reasonable results. The examples were created with the ROME2 software developed to demonstrate the MASS method. For more information about ROME2, see appendix B.

5.1. Testing by example

Proper and thorough testing of the MASS method is outside the scope of this thesis. However, the examples used to demonstrate the method are also used to illustrate the qualities of the method.

The idea is to model simple networks and then make changes to the networks that have known effects. The most obvious example is that used as example 1: bypassing a firewall (see section 5.3). The calculated security values before and after the changes are compared to see if they show the expected results. If the results were contrary to expectations, it would indicate that the method is unsound or that the used parameters (relations and profiles) are faulty. Correspondingly, if the results were the expected, it would indicate that the method might be usable, at least in that specific context.

Since there is no metric corresponding to the security values produced by MASS, it is not clear how the security values should be interpreted. That is, it is not known what security value a component or system ought to have to be considered strong or weak. However, it is clear that a higher security value means that the system is more secure than if it had a lower security value. Therefore the absolute security values are not discussed, only the changes in security values resulting from changes to the model.
5.2. Parameters

The examples have been done with a set of security features corresponding to the classes of the Security Functional Requirements of the Common Criteria (CC, 2004). The set consists of eleven features, see table 1.

Table 1. The set of security features used in the examples.

<table>
<thead>
<tr>
<th>ID</th>
<th>Descriptive name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAU</td>
<td>Security audit</td>
</tr>
<tr>
<td>FCO</td>
<td>Communication</td>
</tr>
<tr>
<td>FCS</td>
<td>Cryptographic support</td>
</tr>
<tr>
<td>FDP</td>
<td>User data protection</td>
</tr>
<tr>
<td>FIA</td>
<td>Identification and authentication</td>
</tr>
<tr>
<td>FMT</td>
<td>Security management</td>
</tr>
<tr>
<td>FPR</td>
<td>Privacy</td>
</tr>
<tr>
<td>FPT</td>
<td>Protection of the TOE Security Functions</td>
</tr>
<tr>
<td>FRU</td>
<td>Resource utilisation</td>
</tr>
<tr>
<td>FTA</td>
<td>TOE access</td>
</tr>
<tr>
<td>FTP</td>
<td>Trusted path/channels</td>
</tr>
</tbody>
</table>

The relations used were developed for the test using intuition and common sense and they are merely intended as examples and to illustrate the method. The relations are given in appendix C.

The security profiles used in the examples were taken from three sources. The “Linux” and “Windows” profiles were taken from Bond & Pählsson (2004b) and the “Firewall” and “Router” profiles from Wesslén (2005). They were evaluated with the Heimald Framework (see section 3.3) to arrive at the security values for the classes, which were then used as the elementary security values for the security features. The filter profiles for the traffic mediators were set to the same values as the security profiles. The trivial “Hub”, “Internet” and “Perfect” profiles were developed specifically for use in the examples. The profiles are presented in table 2 and table 3.

Table 2. Example traffic generator profiles.

<table>
<thead>
<tr>
<th>SF</th>
<th>Linux</th>
<th>Windows</th>
<th>Perfect</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAU</td>
<td>0.50</td>
<td>0.42</td>
<td>1.00</td>
</tr>
<tr>
<td>FCO</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>FCS</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>FDP</td>
<td>0.23</td>
<td>0.23</td>
<td>1.00</td>
</tr>
<tr>
<td>FIA</td>
<td>0.63</td>
<td>0.80</td>
<td>1.00</td>
</tr>
<tr>
<td>FMT</td>
<td>0.44</td>
<td>0.89</td>
<td>1.00</td>
</tr>
<tr>
<td>FPR</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>FPT</td>
<td>0.21</td>
<td>0.15</td>
<td>1.00</td>
</tr>
<tr>
<td>FRU</td>
<td>0.00</td>
<td>0.33</td>
<td>1.00</td>
</tr>
<tr>
<td>FTA</td>
<td>0.00</td>
<td>0.28</td>
<td>1.00</td>
</tr>
<tr>
<td>FTP</td>
<td>0.00</td>
<td>0.50</td>
<td>1.00</td>
</tr>
</tbody>
</table>
### Table 3. Example traffic mediator profiles.

<table>
<thead>
<tr>
<th>SF</th>
<th>Hub</th>
<th>Router</th>
<th>Firewall</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.62</td>
</tr>
<tr>
<td>FCO</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FCS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.75</td>
</tr>
<tr>
<td>FDP</td>
<td>0.00</td>
<td>0.15</td>
<td>0.28</td>
</tr>
<tr>
<td>FIA</td>
<td>0.00</td>
<td>0.30</td>
<td>0.82</td>
</tr>
<tr>
<td>FMT</td>
<td>0.00</td>
<td>0.11</td>
<td>0.94</td>
</tr>
<tr>
<td>FPR</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FPT</td>
<td>0.00</td>
<td>0.02</td>
<td>0.15</td>
</tr>
<tr>
<td>FRU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.33</td>
</tr>
<tr>
<td>FTA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.28</td>
</tr>
<tr>
<td>FTP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
</tbody>
</table>

### 5.3. Example 1: Bypassed firewall

A simple network is modelled as in figure 9. It consists of three computers (two Windows computers with security value 0.37 and one Linux computer with security value 0.23) connected to each other via a generic hub and connected to the internet through a generic firewall. The computers are not very secure and the resulting overall security value calculated by MASS is just 0.21.

![Figure 9. A simple network connected to the internet](image)

The model is then changed by bypassing the firewall and connecting the internet directly to the hub, see figure 10. The expected result is a significant decrease in the overall security value of the system, as the firewall shields some vulnerabilities of the network from exposure to the hostile internet. The MASS model gives a large drop in the overall security value, down to a mere 0.07. It is not possible to say if the size of the drop accurately represents the real effect, but it is at least qualitatively correct.
The system-dependent security profiles before and after the change are shown in table 4. Both profiles have converged to identical values for all security features except FRU, FTA and FTP, where the “Windows” computers have retained their independent values. When the firewall is bypassed, the “Internet” has a huge impact and reduces all the elementary security values to 0, except FRU, FTA and FTP.

Table 4. The system-dependent security profiles.

<table>
<thead>
<tr>
<th>SF</th>
<th>Linux Before</th>
<th>Linux After</th>
<th>Windows Before</th>
<th>Windows After</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAU</td>
<td>0.35</td>
<td>0.00</td>
<td>0.35</td>
<td>0.00</td>
</tr>
<tr>
<td>FCO</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FCS</td>
<td>0.32</td>
<td>0.00</td>
<td>0.32</td>
<td>0.00</td>
</tr>
<tr>
<td>FDP</td>
<td>0.17</td>
<td>0.00</td>
<td>0.17</td>
<td>0.00</td>
</tr>
<tr>
<td>FIA</td>
<td>0.23</td>
<td>0.00</td>
<td>0.23</td>
<td>0.00</td>
</tr>
<tr>
<td>FMT</td>
<td>0.40</td>
<td>0.00</td>
<td>0.40</td>
<td>0.00</td>
</tr>
<tr>
<td>FPR</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FPT</td>
<td>0.06</td>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
</tr>
<tr>
<td>FRU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>FTA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.28</td>
<td>0.28</td>
</tr>
<tr>
<td>FTP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
</tr>
</tbody>
</table>

5.4. Example 2: Partitioned network

A simple network consisting of two types of computers is modelled as in figure 11. The computers are Windows computers (with a security value of 0.37) and Perfect computers (with a security value of 1.00) and they are connected to each other through hubs and a firewall. The firewall is “Perfect”, that is, all elementary security values of both the filter and security profiles are 1. The resulting overall security value is 0.67.
The elementary values of the filter and security profiles of the firewall are then varied between 1 and 0. That results in the overall security value dropping from 0.67, for all elementary values of the profiles being 1, to 0.48 for all values being 0. The result is presented in figure 12.

**Figure 11.** A simple interconnected network.

**Figure 12.** The overall security value (OSV) as a function of the security profile (SP) and filter profile (FP) of the firewall.
As can be seen from the figure the overall security value of the system varies linearly with the values of the filter and security profiles (the apparent unlinearities in the figure are caused by rounding errors). Thus, the results are qualitatively as expected. That is, the overall security value decreases when the efficiency of the firewall drops. However, as with the previous example, it is not possible to determine if the size of the change is correct.

5.5. Discussion
As seen in the examples the MASS method delivers reasonable results, at least in those cases. That could indicate that the method is capable of modelling real networks with reasonable accuracy. However, it is definitely no proof that MASS is a valid and useful method. More comprehensive testing is necessary to reach such a conclusion.

An important aspect of the method is the set of security features used. The method is very dependent on that set and it could make the difference between the method producing useful or useless results.

The set used for the examples has the strength of being small and thus relatively easy to analyse. It is also based on a well-known foundation, which makes it easier to understand. However, the security functional requirements of the Common Criteria were not meant to be used in this way and it is difficult to determine how they relate to each other. Finding the correct weights and functions describing the relations is a difficult problem. That problem is even more obvious when trying to use the full set of the Common Criteria security functional requirements as the set of security features in MASS.

A different set could probably significantly improve the performance and usability of the method. Developing such a set is however outside the scope of this thesis.
6. Conclusions

This chapter presents some further discussion on the work presented in the preceding two chapters and also includes some suggestions for future development of the method.

6.1. Discussion

The previous chapters have presented a method for assessment of system security, called MASS. The basis of the method is that the security of components can be modelled by a set of security features, which represent the security relevant characteristics of the component. Connections between components are modelled as special functions, which capture the effect of the connections on the respective components. From these modelled components and relations, the system dependent security of each component is calculated. These system-dependent values are then used to assess the overall security value for the system as a whole.

The major contribution of MASS stems from the representation of the security of components as a vector of security features. The more complex representation of component security enables more complex models of the relations, which hopefully better captures reality. MASS also captures system structure, both the physical layout (through the physical relations) and the logical structure (through the logical relations).

6.1.1. The set of security features

As mentioned before (see section 4.3.4 and section 5.5), the set of security features used is very important for the usefulness of the method. A good set of features is easy to use and produces sufficiently correct results, while a bad set either produces bad results or are difficult to use. The difficulties can be either in understanding the results, or in finding the correct relations and profiles.

The set used for the examples (see chapter 5) works, but it is not perfectly clear what the results actually mean. The elementary security values of the security features are
not based on any proper metric and their meaning is therefore not defined. Consequently, the resulting security values are not supported by a metric either, and the interpretation is not clear. The lack of a proper metric also makes it difficult to understand what the relations really mean and makes it difficult to find the correct relations (the functions $f$ and $h$ in section 4.2.1 and 4.2.3 respectively).

It is probable that a better set of security features can be found (see section 6.2.1).

### 6.2. Future work

Although MASS is a working method for assessing the security of IT systems, several developments or further investigations are possible.

#### 6.2.1. Developing the set of security features

As mentioned in the previous section there probably exists better sets of security features to use with MASS than the set used in the examples (chapter 5). It is also possible that different sets are suitable in different contexts so it could be valuable to have several different sets to use for different applications. Finding such sets is an important future development of MASS.

A possible approach to developing such sets is to look at (a subset of) one of the taxonomies proposed for security metrics, for example Sedigh et al (2004), Vaughn et al (2003) or Swassson (2001) and Swassson et al (2003). These taxonomies do not present complete sets of attributes that could be directly used as security features, but provides a framework under which to look for a good set of security features.

#### 6.2.2. Proper testing

The MASS method does not rest on any scientifically strict proofs, since there exist no formal theory or proper metrics for security assessment. Thus, the only way to get assurance that it works is through testing. The testing done in this thesis (see chapter 5) only shows that it works for two specific examples, which is inadequate. In order to get assurance that it works in other cases proper testing is needed.

Testing is also needed to develop the set of security features to use and to find the correct functions for the relations.

#### 6.2.3. Hierarchical modelling

The MASS method can be used to model systems at several different levels of abstraction. A significant improvement of the method would be to develop hierarchical modelling, to enable the method to handle several different levels of abstraction at the same time. For example, a small local area network could be modelled and that model then used as a component of a larger network. Such a development would greatly enhance the usefulness of the method for modelling large or complex systems.

It would probably be possible to represent a sub-system as a single component within a larger system. Such a component would have a security profile that would be
an aggregated result of the security profiles of the components of the system it would represent. The structure of the MASS method is such that it seems likely that it would be possible to develop a method for calculating such an aggregated profile.
References


Bond, A. & Pählsson, N. (2004b) *HEIMDAL Security Evaluator 3000 .NET Distribution CD v. 2.0 beta*, FOI Memo, ISSN 1650-1942


REFERENCES


Appendix A: Common Criteria

This appendix is taken in its entirety from Hallberg et al (2004). Minor typographical changes have been made.

Common Criteria

Common Criteria (CC, 1999) is a widely spread and accepted evaluation method, originating from its predecessors TCSEC (1983) and ITSEC (1991). It is based on a set of standardized Security Functional Requirements (SFR) that can be expressed in Protection Profiles (PP) and Security Targets (ST). The product, which the latter describes the behaviour of, is referred to as the Target of Evaluation (TOE). CC is divided into three parts. Part 1 includes a general introduction and overview of CC. Part 2 provides a language for functional descriptions of security requirements and TOE security functions. Part 3 describes the assurance requirements of CC, that is, what is required to achieve a certain Evaluation Assurance Level (EAL).

The SFRs are divided into eleven classes, each describing different security aspects. These classes are further divided into families, which in turn consist of components. The components can be made up of one or more elements. In figure 13, the first family contains three hierarchical components, where component 2 and component 3 can both be used to satisfy dependencies on component 1. Component 3 is hierarchical to component 2 and can also be used to satisfy dependencies on component 2. (CC, 1999)

![Class decomposition diagram](image)

**Figure 13.** Class decomposition diagram (CC, 1999).

In the second family there are three components, of which not all are hierarchical. Components 1 and 2 are hierarchical to no other components. Component 3 is
hierarchical to component 2, and can be used to satisfy dependencies on component 2, but not to satisfy dependencies on component 1.

In the third family, components 2, 3, and 4 are hierarchical to component 1. Components 2 and 3 are both hierarchical to component 1, but non-comparable. Component 4 is hierarchical to both component 2 and component 3.

The SFRs are divided into the following eleven classes (CC, 1999):

**FAU – Security Audit**
Security auditing involves recognising, recording, storing, and analysing information related to security relevant activities

**FCO – Communication**
The FCO class provides two families specifically concerned with assuring the identity of a party participating in a data exchange. These families ensure that an originator cannot deny having sent the message, nor can the recipient deny having received it.

**FCS – Cryptographic Support**
The TOE Security Functions may employ cryptographic functionality to help satisfy several high-level security objectives. This class is used when the TOE implements cryptographic functions, the implementation of which could be in hardware, firmware and/or software.

**FDP – User Data Protection**
The FDP class contains families specifying requirements for TOE security functions and TOE security function policies related to protecting user data. FDP is split into four groups of families that address user data within a TOE, during import, export, and storage as well as security attributes directly related to user data.

**FIA – Identification and Authentication**
The families in the FIA class deal with determining and verifying the claimed identity of users, determining their authority to interact with the TOE, and with the correct association of security attributes for each authorised user.

**FMT – Security Management**
The FMT class is intended to specify the management of several aspects of the TSF: security attributes, TSF data, and functions. The different management roles and their interaction, such as separation of capability, can be specified.

**FPR – Privacy**
The FPR class contains privacy requirements. These requirements provide a user protection against discovery and misuse of identity by other users.

**FPT – Protection of the TSF**
The FPT class contains families of functional requirements that relate to the integrity and management of the mechanisms that provide the TOE security functions and to the integrity of its data.
**FRU – Resource Utilisation**

The FRU class support the availability of required resources such as processing capability and/or storage capacity.

**FTA – TOE Access**

The FTA class specifies functional requirements for controlling the establishment of a user’s session.

**FTP – Trusted Path/Channels**

Families in this class provide requirements for a trusted communication path between users and the TOE security functions, and for a trusted communication channel between the TOE security functions and other trusted IT products. A PP specifies a profile of the implementation-independent requirements for a category of products or systems that meet specific customer needs, whereas a ST specifies the implementation-dependent security functionality used as a basis for a particular product or system. In figure 14 and figure 15, the structure of the PP and ST documents are presented respectively.

![Diagram of Protection Profile structure](image)

**Figure 14. Specifications of Protection Profile (PP) (CC, 1999).**

PPs can be developed using the methods of CC. To simplify this process, CC has a catalogue of standard SFRs which holds a set of functional components used to express functional requirements of products and systems.

A CC evaluation is carried out against a set of predefined assurance levels, Evaluation Assurance Levels (EAL1 to EAL7). These levels represent the ascending level of trust that can be placed in the implementation of the security functionality of the TOE.
Figure 15. Specifications of a Security Target (ST) (CC, 1999).
Appendix B: The ROME2 software

The ROME2 software has been developed to demonstrate the MASS method. It is a development of the ROME software (Peterson, 2004) and has also incorporated some parts from the Heimdal software (Bond & Pahlsson, 2004b).

The main purpose of the tool is to demonstrate the MASS method and to enable testing of the method. Since the modelling and evaluation of IT networks with the MASS method requires a substantial amount of calculations it is infeasible to test the method by hand.

The current version of the tool is designed to use a set of security features corresponding to the Security Functional Requirements of the Common Criteria as they are used in the Heimdal Security Evaluator, or a subset thereof. The implementation of the method is flexible and can use other sets of security features, but the graphic presentation is tied to that set. All relations and profiles are read from text files, so it is very easy for the user to change those.

System requirements
The ROME2 software is developed for Microsoft Windows XP and requires the Microsoft .NET Framework to run. The latest version of the .NET Framework can be downloaded from Microsoft’s website.

The main window
Figure 16 depicts the main window of ROME2. It consists of three main parts; the system information panel to the upper left, the component information panel to the lower left and the workspace to the right. It also has the main menu at the top and the status bar at the bottom.

The main menu
The main menu contains the “File”, “View”, “System” and “Help” menus.

The file menu
The file menu contains choices for creating a new system, opening a saved system, saving a system, saving a system with a new file name and for exiting ROME2.

The view menu
The view menu contains options for how the modelled system is displayed in the workspace. Choosing “Security value” or “System-dependent security value” under “Node colour” sets the colour of the text from blue (for a value of 1) to red (for a value of 0).
APPENDIX B: THE ROME2 SOFTWARE

Figure 16. The main window of ROME2.

The system menu
“Re-evaluate now!” forces the program to re-evaluate the model and update all properties. The “Auto evaluate” turns automatic evaluation on or off. If the automatic evaluation is on the system is reevaluated at least every time something is changed. For large systems, performance may suffer noticeably if the automatic evaluation is turned on.

The help menu
The “User manual” choice shows the user manual, the “ROME2 on the web” choice opens the system internet browser and sends it to http://www.itsecurity.foi.se/dfs/ and the “About ROME2” choice opens the splash window that is also shown at start-up.

The status bar
The status bar displays messages and information about the system and user actions.

The system information panel
The system information panel displays information about the modelled system. The modelled properties are the number of traffic generators and the number of traffic mediators and the calculated properties are the overall security value.
The component information panel
The component information panel displays information about the currently selected component. If no component is selected the component information panel is blank. The properties shown depend on the component; different things are shown for traffic mediators and traffic generators.

*Note: If automatic evaluation is turned off, the information shown may not be correct! To re-evaluate the system, press “F5”.*

The workspace
The workspace is the area where the actual modelling of the system is done. Right-clicking in the workspace brings up the menu shown in figure 17. Menu items that are not applicable are not shown, i.e. “Properties” is only shown when a component is selected.

![Figure 17. The workspace menu.](image)

Using “Add”, components and relations can be added to the model and using “Remove”, they can be removed. Selecting “Properties” opens the component properties window and “Evaluation” opens the evaluation window.

When adding logical relations, the first component clicked will influence the second component. That is, the second component will be dependent on the first. The logical relation is added to the second component. The direction of the influence is indicated by an arrow on the relation.

The “Influence” choice evaluates the influence of the chosen component. The security and filter profiles of the component are multiplied by modifiers between 0 and 1, in steps of 0.1. For each step, the overall security value is calculated and the result is saved in a .xls file in the “Evaluations” directory.

The OSV meter
The OSV meter shows the overall security value, as percent and with a pie chart. The pie-chart changes colour from blue to red as the overall security value decreases.

Tool tips
Some information about a component is shown if the mouse is held stationary over a component.
**Note:** If automatic evaluation is turned off, the information shown is probably not correct. To re-evaluate the system, press “F5”.

**The component properties window**

The component properties window is shown in figure 18. The window allows the user to enter or select the name, weight and profile of the component. The ID, (independent) security value and system-dependent security values are also shown. Since “Weight” is not applicable to traffic mediators it is set to 1 and locked for traffic mediators.

![Component properties window](image)

**Figure 18.** The component properties window.

**The component evaluation window**

The evaluation window shows the evaluation details of the selected component. It is shown in figure 19. The “Overview” tab shows the general properties of the component, including the independent and system dependent security values and the corresponding confidentiality, integrity and availability values. It also shows a bar graph of the security values of each of the classes.

The “Details” tab shows a detailed report of the security and CIA values for all the security features of the component, either as a table or as raw xml data. The “Treeview” tab provides the same information as the “Details” tab, but as a tree of security features, families and classes.

The “File” menu enables the export of evaluation data as an xml, Microsoft Excel or plain text file.
**Figure 19.** The component evaluation window; the overview tab
Appendix C: Example relations

This appendix contains the relations used in the examples of the thesis.

Example relations

The neighbour relation for traffic generators, i.e. the function \( f \) in section 4.2.1, is presented in table 5 and the system relation, i.e. the function \( b \) in section 4.2.3 is presented in table 6. Since no logical relations are used in the examples, no logical relations are presented here.

Table 5. Example neighbour relation.

<table>
<thead>
<tr>
<th></th>
<th>FAU</th>
<th>FCO</th>
<th>FCS</th>
<th>FDP</th>
<th>FIA</th>
<th>FMT</th>
<th>FPR</th>
<th>FPT</th>
<th>FRU</th>
<th>FTA</th>
<th>FTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FCO</td>
<td>0.00</td>
<td>0.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FCS</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FDP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FIA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.30</td>
<td>0.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.20</td>
</tr>
<tr>
<td>FMT</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FPR</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FPT</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FRU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FTA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FTP</td>
<td>0.00</td>
<td>0.20</td>
<td>0.00</td>
<td>0.00</td>
<td>0.30</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
</tbody>
</table>
Table 6. Example system relation.

<table>
<thead>
<tr>
<th></th>
<th>FAU</th>
<th>FCO</th>
<th>FCS</th>
<th>FDP</th>
<th>FIA</th>
<th>FMT</th>
<th>FPR</th>
<th>FPT</th>
<th>FRU</th>
<th>FTA</th>
<th>FTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
</tr>
<tr>
<td>FCO</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FCS</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
</tr>
<tr>
<td>FDP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FIA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FMT</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FPR</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FPT</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FRU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FTA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>FTP</td>
<td>0.00</td>
<td>0.20</td>
<td>0.30</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Nil</td>
<td>Min</td>
<td>Min</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Nil</td>
<td>Min</td>
</tr>
</tbody>
</table>
Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© 2005, Rikard Andersson