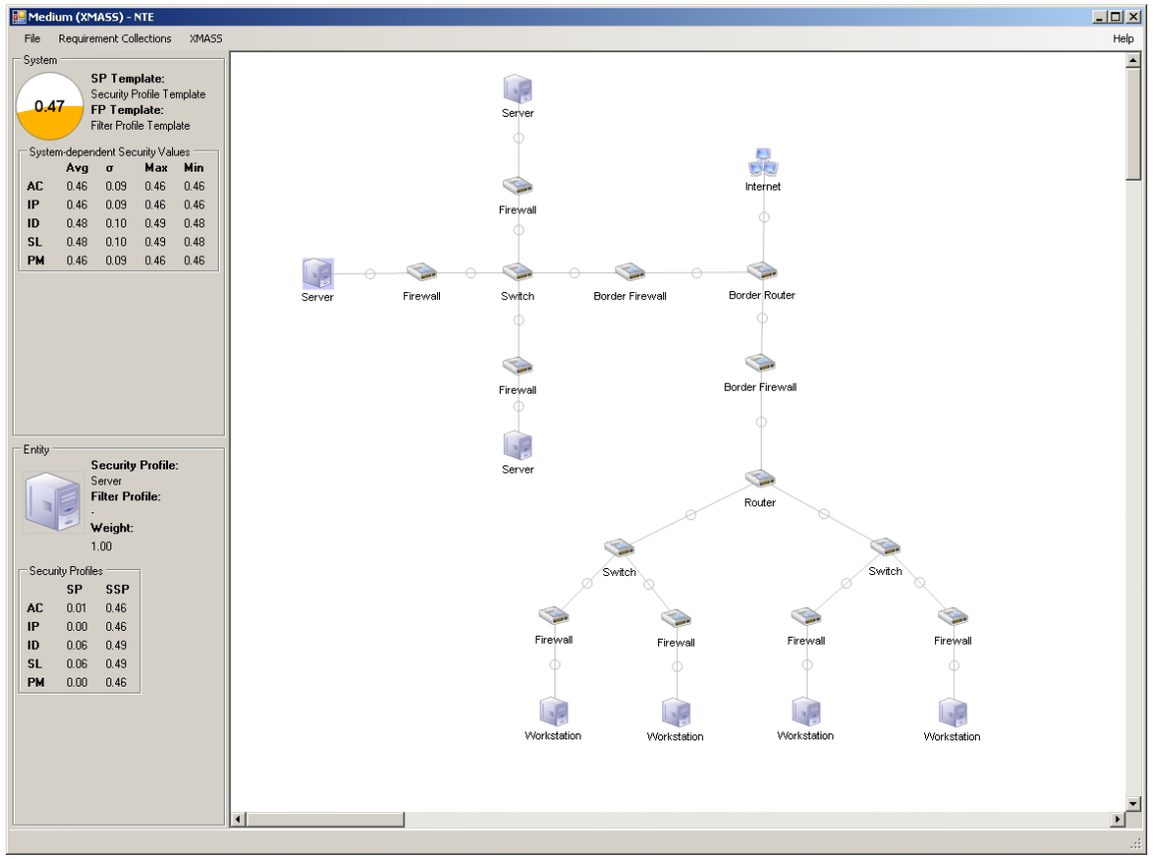
Refinement and realization of security assessment methods

JONAS HALLBERG, JOHAN BENGTSSON, RICHARD ANDERSSON



FOI, Swedish Defence Research Agency, is a mainly assignment-funded agency under the Ministry of Defence. The core activities are research, method and technology development, as well as studies conducted in the interests of Swedish defence and the safety and security of society. The organisation employs approximately 1000 personnel of whom about 800 are scientists. This makes FOI Sweden's largest research institute. FOI gives its customers access to leading-edge expertise in a large number of fields such as security policy studies, defence and security related analyses, the assessment of various types of threat, systems for control and management of crises, protection against and management of hazardous substances, IT security and the potential offered by new sensors.

FOI
Swedish Defence Research Agency
Command and Control Systems
P.O. Box 1165
SE-581 11 Linköping

Phone: +46 13 37 80 00
Fax: +46 13 37 81 00

www.foi.se

FOI-R--2387--SE
ISSN 1650-1942

Scientific report
December 2007

Command and Control Systems

Jonas Hallberg, Johan Bengtsson,
Richard Andersson

Refinement and realization of security assessment methods

Titel	Förfining och förverkligande av metod och process-modell för IT-säkerhetsvärdering
Title	Refinement and realization of security assessment method and process model
Rapportnr/Report no	FOI-R--2387--SE
Rapporttyp Report Type	Vetenskaplig rapport Scientific report
Utgivningsår/Year	2007
Antal sidor/Pages	96 p
ISSN	ISSN 1650-1942
Kund/Customer	Försvarsmakten
Forskningsområde Programme area	7. Ledning med MSI 7. C4I
Delområde Subcategory	71 Ledning 71 Command, Control, Communications, Computers, Intelligence
Projektnr/Project no	E7046
Godkänd av/Approved by	Martin Rantzer

Totalförsvarets Forskningsinstitut FOI

Avdelningen för Ledningssystem

Box 1165

581 11 Linköping

Abstract

There are risks associated with information technology, IT, that may substantially decrease the potential benefits. Thus, to maximize the utility of IT, possible security issues of information systems should be carefully considered and mitigated. To be able to keep security under control, its assessment is important. However, since security is an abstract, subjective, non-tangible property, properly assessing the security of non-trivial systems is hard and, currently, there are no methods for efficient, reliable, and valid security assessments. Thus, it is important to extend previous efforts in order to enable the design of efficacious methods.

The results presented in this report include:

- improvements and extensions of an existing method,
- a software environment for the implementation of methods,
- the implementation of a software tool for an existing method, and
- a novel method implementing a process model for security assessment.

Keywords: Networked information systems, IT security, security assessment, security metrics

Sammanfattning

Användning av informationsteknik, IT, medför risker som kan medföra att stora delar av dess fördelar elimineras. Därmed kräver en maximering av nyttan att eventuella säkerhetsproblem beaktas och hanteras. För att ha kontroll på IT-säkerheten är det av vikt att kunna värdera säkerhetsnivåer i system. Då säkerhet är en abstrakt, subjektiv, ogripbar egenskap är det svårt att värdera denna och för närvarande finns det inga effektiva metoder som ger pålitliga resultat. Därmed är det viktigt att bygga vidare på redan presenterade resultat för att möjliggöra utveckling av effektiva metoder.

Resultaten som presenteras i denna rapport inkluderar:

- förbättringar och utvidgningar av befintlig metod,
- en mjukvarumiljö för realisering av metoder,
- en realisering av en befintlig metod samt
- en ny metod som realiserar en processmodell för säkerhetsvärdering.

Nyckelord: Informationssystem, IT-säkerhet, säkerhetsvärdering, säkerhetsmetriker

Contents

1 Introduction

The use of information technology (IT) has the potential to increase the efficiency of individuals as well as organizations. However, there are risks associated with IT which may alleviate the benefits and even cause substantial damage. Thus, it is essential to be in control of all the security issues potentially originating from the use of information systems. This is a growing challenge because of the ever increasing complexity of information systems. The goal for individuals and organizations alike is adequate information security, but as the importance of IT in business increases, IT security becomes more important. This report aims at improving the ability to assess the IT security of information systems.

1.1 Motivation

Unfortunately, there are currently no methods for efficient, reliable, and valid security assessments (ACSAC, 2002; Vaughn et al, 2003; Seddigh et al, 2004; Geer, 2006; Hallberg et al, 2006). To a large extent this stems from the fact that security is an abstract, subjective, non-tangible property. This results in:

- difficulties to decide what is actually meant with security,
- the belief in *secure* as an achievable, ever-lasting property of information systems,
- security not being possible to measure, instead other system properties and effects have to be measured in order to enable estimations of security levels, and
- difficulties to interpret the meaning of measured security-relevant system properties and effects.

Consequently, security is difficult to assess. However, the dependence in information systems makes it important. Thus, the alleviation of these problems is the motivation for the work presented in this report.

1.2 Problem Formulation

To perform efficacious security assessments answering the needs of the system users, administrators, managers, and owners is difficult. Thus, it is important to:

- extend previous efforts regarding security assessment, in order to make academic results more concrete, and enable the design of efficacious methods and tools,
- implement proposed methods as software tools to enable their evaluation,

- provide environments for the evaluation and comparison of security assessment tools and examples illustrating how proposed methods can be used, and
- design novel methods addressing the needs of security assessment.

1.3 Contributions

The results presented in this report contribute to the area of security assessment through:

- improvements of the eXtended Method for Assessment of System Security, XMASS,
- the introduction of security and filter profiles based on the requirements on security functions (sw. Krav på säkerhetsfunktioner, KSF) formulated by the Swedish Armed Forces (2004),
- the introduction of an environment for the implementation of security assessment methods,
- the implementation of a software tool realizing the XMASS, and
- the use of Bayesian networks to illustrate the use of a process model for security assessment.

1.4 Report Layout

In chapter 2, the areas of IT security and security assessment are presented together with the process model for security assessment, the eXtended Method for Assessment of System Security (XMASS), and, finally, the concept of Bayesian networks. In chapter 3, improvements of XMASS are introduced. In chapter 4, security and filter profiles designed according to XMASS are introduced. In chapter 5, a software environment for security assessment tools and a tool implementing the XMASS are presented. In chapter 6, a security assessment method based on the process model for security assessment is introduced. In chapter 7, finally, the presented results are discussed.

2 Background

In this chapter, the use of the term IT security in this report is stated. Further, the area of IT security assessment is discussed. Thereafter, the process model for security assessment and the concept of Bayesian networks are presented.

2.1 IT Security

There are many excellent resources describing various aspects of IT security, for instance, (Anderson, 2001; Bishop, 2003; Gollmann, 2006). The term IT security, also referred to as computer security, is defined in many different ways depending on the context it appears in. It is therefore hard to give an explicit definition, which is suitable for all contexts. Referring to computer security, Gollmann (2006) states that there are several possible definitions, such as, "deals with the *prevention* and *detection* of *unauthorized* actions by users of a computer system."

In this report, the term security is used in the meaning of IT security, which consists of upholding the characteristics of confidentiality, integrity, and availability of IT systems and the data processed, transmitted, and stored in these systems.

2.2 IT Security Assessment

Security assessments are performed in order to establish how well systems meet specified security criteria. The aim of security assessments is to produce knowledge. This knowledge can be used to improve the security levels of the assessed systems. Perfect security is always the ultimate goal for a system, but it cannot be achieved. Security assessments can provide insight into the security posture of systems. However, it cannot guarantee any level of security, though it can provide a basis for confidence in the assessed system (Bishop, 2003).

Although IT security deals with technical elements, comprehensive security assessments need to consider other aspects of the assessed systems. Three such aspects are the organizational, human, and contextual aspects of systems. It is essential to point out that the inclusion of these aspects highlights the need to consider their influence on the IT security levels of systems. However, IT security assessment does not include the assessment of the security of organizations, persons, and contexts themselves.

Hallberg et al (2005) divide the task of performing a security assessment into the two subtasks securability assessment and security level assessment. Securability

is described as “a characteristic of the design of an information system, including technical, organizational, and individual aspects, aiming at an estimate of the level to which systems can be secured during operation. Thus, the securability is constant as long as the design is not changed.” On the contrary, the consideration of operational aspects of the system is required for deciding the security level. Thereby, “security levels change with the design, configuration, and state of systems and system entities.” Hallberg et al (2005) suggests security value as the comprehensive term including both securability and security level.

Gacic (2006) presents a structure of two main categories and five basic approaches to security assessment, illustrated in Figure 1. This structure is a development of the four classes of approaches to security assessment presented by Hallberg et al (2004). The first of the two main categories, *consequences*, consists of the two approaches *observation* and *test*. For these approaches, the system is viewed as a black box, with or without stimulation, when drawing conclusions based on the behavior of the system. The second main category, *characteristics*, consists of the three approaches *component*, *system-wide*, and *structural characteristics*. For these approaches it is assumed that knowledge of the security of systems can be gained by the knowledge of the internals of systems.

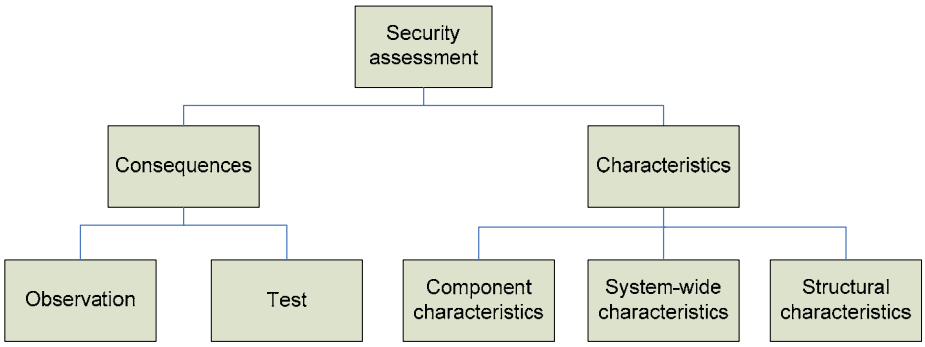


Figure 1: Basic approaches to security assessment.

2.2.1 Security Metrics

Security metrics is a central concept of security assessment. There is a multitude of different interpretations of the term security metrics. In this report, the definition by Hallberg et al (2004) is adopted:

> *A security metric contains three main parts: a magnitude, a scale and an interpretation. The security values of systems are measured according to a specified magnitude and related to a scale. The interpretation prescribes the meaning of obtained security values.*

The presence of magnitude and scale means there should be values that can be measured or computed. Moreover, if these values correspond to proper security metrics, they must be possible to interpret. However, the combination of measurability and computability on one hand and interpretability on the other hand is difficult task and a central issue to be solved for enabling efficacious security assessments.

2.2.2 Process Model for Security Assessment

Security is frequently assessed. For example, when a password is selected and, explicitly or implicitly, judged to provide adequate security, a security assessment has taken place. More extensive examples include assessments required to support the procurement and commissioning of IT systems or components, such as enterprise resource planning systems and firewalls.

Although frequently performed, most security assessment processes are performed without much regard to which steps are necessary and which steps are actually performed. Consequently, several of the steps are disregarded or performed implicitly and not documented. To address this problem Hallberg et al (2007) present a model for security assessment processes. The process model includes the six activities (Figure 2):

1. analyze needs regarding security assessment,
2. establish relevant security characteristics,
3. connect measurable security characteristics and effects to the relevant security characteristics,
4. measure selected security characteristics and effects,
5. compute security values, and
6. interpret security values.

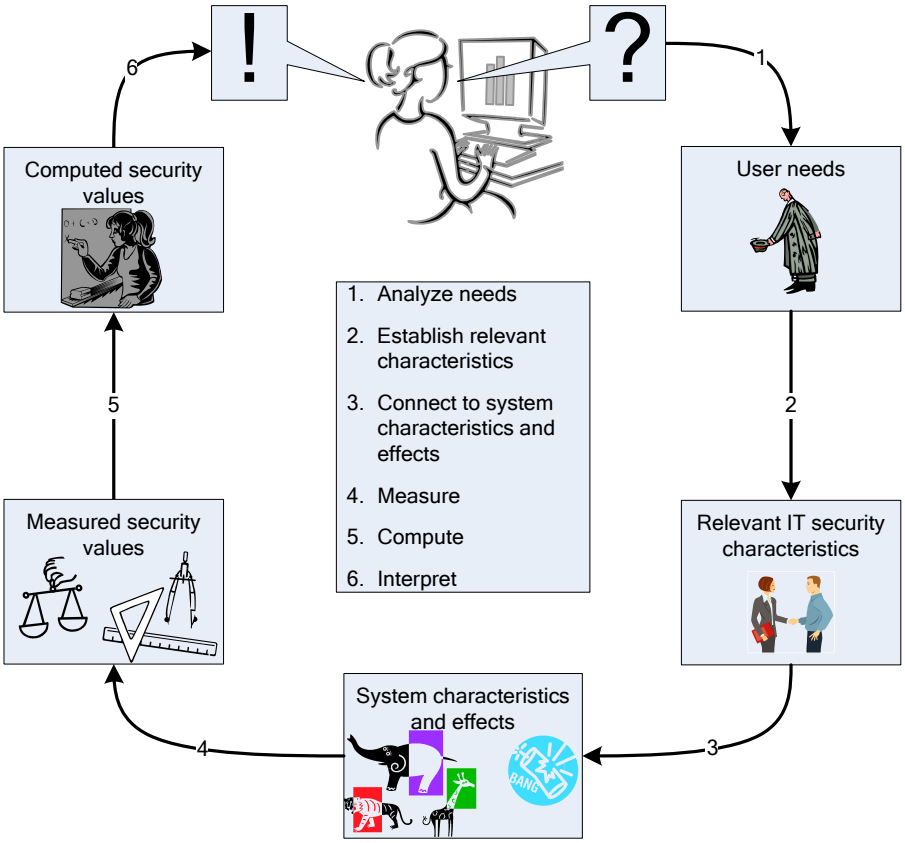


Figure 2: The process model for security assessment.

2.2.2.1 Analyze Needs Regarding Security Assessment

To be able to produce viable assessment results, the needs of the end users have to be established. To produce assessment results without a clear understanding of the connection to the end user needs is inefficient and often counterproductive. Thus, the needs motivating an assessment have to be identified, documented, and agreed on.

2.2.2.2 Establish Relevant Security Characteristics

In order to fulfill the identified needs, the characteristics of the system, whose assessment will support the end user, have to be defined. These characteristics are referred to as the relevant security characteristics, stressing their importance

for the end users. The definition of the relevant security characteristics results in a context-aware definition of IT security, since the characteristics specify what, considering the current system and specific situation of the end user, is important regarding security. Thus, the problems associated with the lack of common and, considering security assessment, useful definitions of IT security are alleviated.

The activity includes:

1. the evaluation of provided assessment needs, in order to assure their usefulness as a base for security assessments,
2. the specification of relevant security characteristics,
3. the specification of system scope, and
4. the specification of the relations between the security characteristics and the assessment needs.

2.2.2.3 Connect Measurable Security Characteristics and Effects to the Relevant Security Characteristics

To enable the actual assessment, the relevant IT security characteristics have to be measurable or broken down into measurable system characteristics and effects. When a relevant characteristic is not measurable, the definition of a computational model is required. The computational model describes how the measurable values can be aggregated into high-level security values corresponding to the relevant IT security characteristic. This will result in a set of security metrics corresponding to the relevant security characteristics.

This activity includes:

1. system modeling regarding system entities,
2. identification of system characteristics and effects,
3. system modeling regarding measurable system characteristics and effects, and
4. specification of the computations model.

2.2.2.4 Measure Selected Security Characteristics and Effects

When the set of measurable system characteristics and effects have been established, they have to be measured. This activity includes:

1. scrutinizing the system model in order to assure the presence of the information required for performing the measurement,
2. association of values to the measurable security characteristics and effects, and
3. validation of the accuracy of the measured values.

If the precision is insufficient, the measurement process has to be improved or the computations model has to be revised.

2.2.2.5 Compute Security Values

The high-level security values corresponding to the relevant security characteristics are computed from the measured security values using the computational model. This activity includes:

1. the implementation of the computations model and
2. the actual computation of the security values corresponding to the relevant security characteristics.

2.2.2.6 Interpret Security Values

For the end users to gain any information from the security assessments, the computed high-level security values, corresponding to the relevant security characteristics, have to be interpreted. The success of this activity depends on well specified relations between the security values and the relevant security characteristics. This activity includes:

1. selection of schema for the interpretation of security values, depending on the correspondence between the security metrics and the needs of the end users and
2. the establishment of the interpretations of security values.

2.3 The eXtended Method for Assessment of System Security

The eXtended Method for Assessment of System Security, XMASS, was introduced by Hallberg et al (2006). The aim of XMASS is to support security assessments for networked information systems. The assessments are based on available knowledge of the security characteristics of the system entities and their relations. The system entities are divided into traffic generators and traffic mediators. Traffic generators can for example represent computers, while traffic mediators can represent hubs and firewalls. The important security aspects of system entities are described by security profiles. A security profile is vector with security values corresponding to security features of the system entities. Filter profiles are associated to the traffic mediators in order to capture their filtering capabilities. The main result of the method is a set of system-dependent security profiles (SSP:s), one for each traffic generator in the system.

The system modeling is supported by the possibility to create profiles for standardized system entities and their relations. There are no explicit limitations in the method disabling the inclusions of different system aspects. The computation of higher-level security values is controlled by the computations model, which can be specified by the user, but is tied to the structure of the system. Thus, the computation of aggregated security values, not just the input, depends on the system models. The assessment results are presented at various levels, for individual entities, for entities in a system context, and for the system as a whole. In the following subsections, the parts of the XMASS relevant for this report are presented. For a more detailed description of the XMASS, see (Hallberg et al, 2006). The central concepts of XMASS are illustrated in Figure 3.

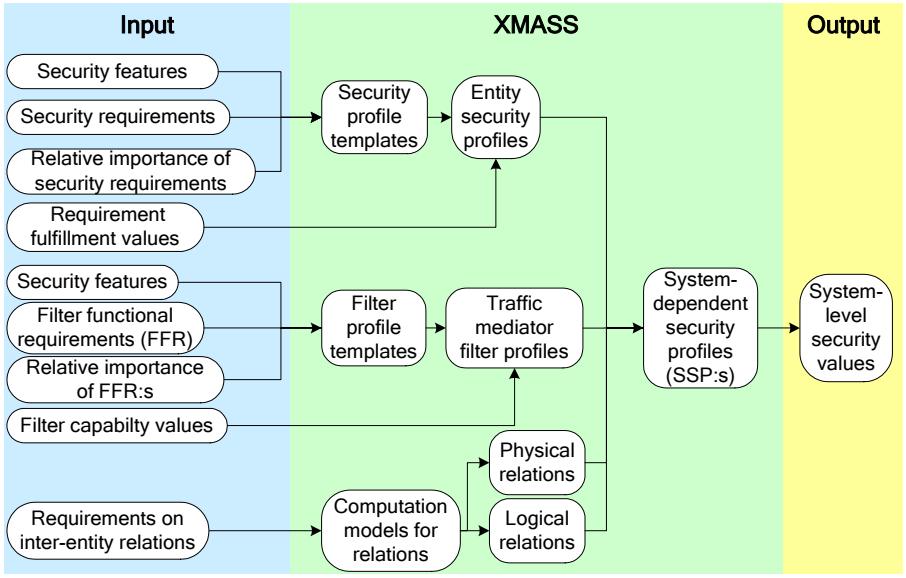


Figure 3: The central concepts of XMASS and their relations.

Table 1 includes the central terms used in the specification of the computations model of XMASS (Hallberg et al, 2006).

Table 1: Notation used for the presentation of security values computation.

Term	Description
SP^e	The security profile of entity *e*
N	The number of security features represented by the security profiles
SSP^e	The system-dependent security profile of entity *e*

Term	Description
$NSP^{e,nb}$	The neighbor-dependent security profile of entity *e* considering neighbor *nb*
f	The function used to calculate the effects on entities caused by neighbors
FP^{tm}	The filter profile of traffic mediator *tm*
EFP^{tm}	The effective filter profile of traffic mediator *tm*
ESP^{nb}	The equivalent security profile of neighbor *nb* considering the security profile, SP^{nb}, of *nb* and the effective filter profiles, EFPs, of intermediate traffic mediators
RSP^{nb}	The resulting security profile of the neighbor *nb* combining the effects of several equivalent security profiles, ESPs, resulting from alternative paths
$LSP^{e,lre}$	The logical security profile describing the effects of the logically related entity *lre* on the system-dependent security profile of entity *e*
$g^{e,lre}$	The function used to calculate the effects of a logical relation between the entities *e* and *lre*
h	The *system function* used to calculate the system-dependent security profile of entities based on the corresponding security profile, SP, the neighbor-dependent security profiles, NSPs, and logical security profiles, LSPs

2.3.1 Entity Security Profiles

While being an important intermediate result in the XMASS, the entity security profiles constitute a base for computation of the system dependent security profiles (SSP:s). The entity security profiles were taken as input in the MASS (Andersson & Hallberg, 2006), but in the XMASS they are computed through the following steps:

1. decide on a set of security features to be represented by the security profile,
2. for each security feature, decide on a set of entity security requirements, which describe what needs to be fulfilled in order to fulfill the security feature,
3. divide the entity security requirements, for each security feature, into the sets of fundamental and important security requirements,
4. prioritize the security requirements pair-wise in the sets of important security requirements based on their relative importance,

5. calculate the security profile template based on the data produced in step 1 to 4, and
6. calculate the entity security profiles by using the entity security profile template and the data on fulfillment of the entity security requirements.

As a result, the XMASS have to be provided with the following input:

1. a set of security features,
2. sets of fundamental and important entity security requirements for each security feature,
3. the relative importance among each pair of requirements in each of the sets of important security requirements, and
4. a parameter specifying the maximum influence of the important security requirements on the security values included in the security profiles.

The modification of the security profile template formulas presented in section 3.3.1 eliminates the need for the user to provide the parameter deciding the maximum influence of the important security requirements. The content of the input is not dictated by the XMASS, but it is vital for the reliability and validity of the assessment results.

2.3.1.1 Security Profile Templates

In the XMASS, a security profile template is to be seen as a set of formulas used to compute the security values in the security profiles. The template thereby consists of one formula per element in the security profile. The XMASS use the method for criteria weighting from the Analytic Hierarchy Process, AHP (Saaty, 1994), in order to decide the relative importance among the important security requirements. Entities having unfulfilled fundamental requirements should not be awarded a security value larger than zero, which is the reason for excluding the fundamental security requirements from the prioritization. To signify the weight of fundamental requirements, their fulfillment values are instead multiplied with the result of the prioritization of the important security requirements.

For each security feature, *k*, the set of security requirements is divided into one set of fundamental and one set of important security requirements, $\mathbf{RF}_k$ and $\mathbf{RI}_k$ respectively. For each security feature the relative importance between each pair in the set of important requirements, ri_i, $ri_j \in \mathbf{RI}_k$, is decided according to Table 2 by assigning weights, a_{ij}. The weights result in matrices, $A_k = \{a_{ij}\}$, which describes the relative importance of the pairs of important security requirements, $\mathbf{RI}_k$ of the security feature *k*.

A requirement, ri_i, considered more important than another requirement, ri_j, results in a corresponding weight, a_{ij}, with a value larger than 1. Similarly a weight with a value less than 1 express less importance of the former requirement compared to the latter, while a value of exactly 1 expresses equal importance between the pair of requirements. Values less than 1 are constructed by reversing the comparison, that is, comparing the latter with the former requirement, and using the reciprocal value, $a_{ij} = 1/ a_{ji}$.

The prioritization of the important requirements, $\mathbf{RI}_k$, of security feature k, is based on the specified weights and decided by calculating and scaling the eigenvector, $e_k = \{e_{ki}\}$, which corresponds to the largest eigenvalue, λ_{max}, of the matrix A_k. The eigenvector is scaled so that $\sum_i e_{ki} = 1$. For matrices with properties like those that A_k has, it can be shown that λ_{max} will be slightly larger than the dimension of the matrix and the rest of the eigenvalues will be close to zero (Forman & Selly, 2002).

The values of the security profiles intend to reflect the qualities of the entities regarding the corresponding security feature. The fundamental requirements of each security feature should be decisive for the respective security value, which is why the degree of fulfillment for each of these requirements are included as a factor (as in multiplication) in the security profile template. Fulfillment degrees for the important requirements are included as a weighted sum. A lowest possible value for the factor representing the important requirements is included in the template to avoid cases where no fulfilled important requirements result in a security value of 0. As a consequence, the weighted sum, which represents the important requirements, is scaled to limit the factor to a maximum value of 1. The template for the scalar security value, SP_k, corresponding to security feature, *k*, is presented as Eq. 2.1.1 in (Hallberg et al, 2006), that is,

$$SP_k = \left[\prod_{j=1}^{m} rfv(rf_j) \right] \cdot \left(v + (1-v) \cdot \left(\sum_{i=1}^{n} rfv(ri_i) \cdot e_{ki} \right) \right)$$

where rfv(x) returns the fulfillment value, [0, 1], of requirement x, and v, $0 < v \leq 1$, is a parameter deciding the influence of the important security requirements.

Table 2: The weights used when deciding the relative importance of requirements adapted from Saaty (1994)

Requirement weight	Meaning
1	Equal importance
3	Moderate importance
5	Strong importance
7	Very strong importance
9	Extreme importance

2.3.1.2 Calculation of Security Profile Values

After specifying the security profile template, the entity security profiles are calculated by inserting the fulfillment values for each included security requirement. The fulfillment values state how well each security requirement is met by the considered entity. A fulfillment value of 1 represents complete fulfillment of the corresponding requirement, whereas a value of 0 denotes non-compliance. Partial fulfillment is expressed by a fulfillment value between 0 and 1.

2.3.2 Traffic Mediator Filter Profiles

Filter profiles are needed during the computation of the system dependent security profiles (SSP:s) to assess how the filtering capabilities of the traffic mediators affect the security. In the XMASS, the filter profiles are computed using the following steps (Hallberg et al, 2006):

1. decide a set of requirements reflecting the needed filtering functionality,
2. for each security feature, prioritize each pair of filter functional requirements regarding their relative importance,
3. a filter profile template is calculated based on the data produced in step 1 and 2, and
4. calculate the filter profiles using the filter profile template and data on the fulfillment of filter functional requirements.

2.3.2.1 Filter Profile Templates

Filter profile templates are used to map the filtering capabilities of traffic mediators to the filtering profiles. The filtering profiles are used to compute the influence of traffic mediators during the computations of the system-dependent security profiles (SSP:s). To decide the filtering profile template, the filtering functionality of the network entities is characterized. Thereafter, the different catego-

ries of filtering functionality are prioritized regarding their relative importance by using the process for criteria weighting in the AHP (Saaty, 1994).

Traffic mediators are unfortunately unable to filter all malicious traffic. This inability is modeled by including an influence factor, $S_k \in [0, 1]$, in the filter profile template. Thereby the maximum value of each element, k, in the filter profile is S_k. The traffic mediators are characterized in order to assess their influence between traffic generators in a system. Basically this is done by creating a set of requirements on filtering functionality. The set of filter functional requirements, **FFR**, is used to assess the filtering capability of all traffic mediators.

The assessment of filter functionality can be based on traffic mediator types, specific traffic mediators, or specific traffic mediator configurations reflecting the amount of system data available. The result of the assessment is a vector with elements in [0, 1], which corresponds to each filter functional requirement, $ffr_i \in$ **FFR**. The elements in this vector are referred to as the filtering capability values, $fcv_i \in [0, 1]$.

The weights of the filter functional values in the filter profile template is decided using the process of criteria weighting from AHP in the same way as the security profile template, but with the exception that filter functional requirements are prioritized instead of security requirements. Moreover, the same set of filter functional requirements is used for all the filter profile values. Hence, for each filter profile value, FP_k, the filter functional requirements, ffr_i, $ffr_j \in$ **FFR**, are pair-wise prioritized according to their relative importance, resulting in weights, b_{ij}. The weights result in matrices, $B_k = \{b_{ij}\}$. Subsequently, the prioritization is decided by calculating and scaling the eigenvector, $e_k = \{e_{ki}\}$, which corresponds to the largest eigenvalue, λ_{max}, of B_k. The scaling of the eigenvector is done so that $\sum_i e_{ki} = 1$.

Filter profiles are vectors $[0, 1]^N$, where the values are calculated as

$$FP_k = S_k \cdot \sum_{i=1}^{N} \left(e_{ki} \cdot fcv_i \right)$$

where S_k is the filtering influence factor, e_k is a vector containing the weights of the filter functional requirements for the filter profile value k, and fcv_i are the filtering capability values of the traffic mediator.

2.3.2.2 Calculation of Filter Profile Values

The first step in calculating the filter profile values is to decide the filtering capability values, $fcv_i \in [0, 1]$ corresponding to each filter profile value. If there are N filter profile values and M filter functional requirements, there are $N \cdot M$ filtering capability values. The filtering capability values depend on the configuration as well as the functionality and operation of the assessed traffic mediator. However, in practice it may be challenging to acquire this data. Thus, the capability values of the traffic mediators have to be modeled. The second, and final, step of the filter profile computation is performed by inserting the filtering capability values into the filter profile template.

2.4 Bayesian networks

Bayesian networks, or belief networks, are used to represent knowledge in areas where uncertainty is present. Bayesian networks consist of *directed acyclic graphs*, DAGs, and can be described as probability models representing a set of variables and their causal influences. All nodes in the DAGs are affected by their parents only, that is, the nodes which are connected to the current node with a forward edge. When discrete random variables are used, tables can be used to specify the distribution of the values of a node for all combinations of the values of the parents of the node. The structuring of the DAGs is often an efficient mechanism to capture human knowledge, while measurement data is included in the specification of the parameters. (Ben-Gal, 2007)

Bayesian networks support the aggregation of all available knowledge and data related to a problem under study, even if parts of it are uncertain or a combination of objectively and subjectively measured probabilities. The models can straightforwardly be altered according to current observations. The possibility to include uncertain data and the transparency of the computations qualify Bayesian networks as a viable approach to security assessments, especially when starting from relevant security characteristics of systems, as will be explored in chapter 6.

3 Development of the XMASS

During 2007, the development of the XMASS has resulted in the addition of relation profiles and the entity quantity parameter (Bengtsson & Brinck, 2007) as well as improvements of the computations model.

3.1 Relation Profiles

The XMASS uses profiles to model both security values and filtering values of entities. A profile is a grouping of related values, which can be used by one or more entities. Thereby, an alteration of values in a profile affects all entities described with the specific profile. A structure like this facilitates the task of switching between different sets of values.

To enhance the consistency of the XMASS, two additional kinds of profiles are introduced: the physical relation profile and the logical relation profile (Bengtsson & Brinck, 2007). The introduction of these two kinds of profiles does not alter the method itself, but it results in a grouping of related values, which in turn results in a practical way to switch relation related values. This is illustrated in the software implementation of the XMASS, called SANTA, where the introduction of these profiles results in better possibilities to evaluate the method (see section 5.8).

Hallberg et al (2006) implicitly describes both physical and logical relation profiles, but never refer to them as profiles. The software implementation of the MASS, the predecessor of the XMASS, called ROME2 contains similar groupings for logical relations, but neither these are referred to as profiles.

A relation profile, whether physical or logical, consists of a matrix of weights and a matrix of functions. Both matrices are used in order to model the relation between entities and thereby capture how the relation affects the security values of the corresponding entities.

The physical relation profile is selected as a system-wide setting and, thereby, affects all physical relations in a system. The logical relation profile, on the other hand, is selected per relation, which means every logical relation in a system can use different profiles. Thereby, it is possible to model different kinds of logical relations, such as those resulting from Virtual Private Networks or the relation between a workstation and a central server managing the anti-virus software.

3.2 Network of Entities

Hallberg et al (2006) describes a traffic generator as an entity producing traffic, for example, a computer, a server, or a connected network. To facilitate the modeling of connected networks, Bengtsson & Brinck (2007) extended the representation of the traffic generators with a quantity value, which specifies how many equivalent entities the traffic generator represents. In this way, several equivalent entities can be represented by one traffic generator having the number of entities as quantity and the same security profile as if it would have been a single entity. Thereby, it is not necessary to add separate entities in the modeling phase to model a stack of equivalent entities or a public network.

There is a difference between subnets of the modeled systems and public networks connected to the modeled systems in that the former are included in the assessed system, while the latter are external and only their influence on the assessed system should be accounted for. To handle this difference, the possibility to set the weight of the traffic generators is used. That is, since the system-dependent security profiles, SSP:s, of subnets should be included in the computation of system-wide security values, the weight is set to a value larger than zero. On the other hand, when public networks are modeled, the weight of the entity should be set to zero. As a result of introducing a quantity value, all traffic generators having a quantity larger than one are considered to be a network of entities.

Besides changing the representation of a traffic generator, the extension results in changes in the calculations. When calculating the SSP, it has to be taken into consideration that a network of entities, having a quantity of n, represents n equivalent traffic generators. Thereby the network of entities should result in n equivalent neighbor-dependent security profiles (NSPs), which specifies the influence of neighbor entities, in the calculation of the SSP for a neighboring entity.

While calculating the SSP for a network of entities itself, internal effects have to be considered. A network of entities, having a quantity of n, is seen as n traffic generators connected through a hub, which have no filtering capabilities. Consequently, the security profile of the network of entities is added $n-1$ times into the calculation of the SSP of the network of entities in order to take internal effects into account. Thus, the SSP will be equivalent to the SSP:s of the traffic generators if the network is expanded into n traffic generators connected through a hub.

3.3 Improvements of Calculations

In this section, the computations model included in the XMASS is improved regarding the calculation of security profile templates and the combination of multiple paths between traffic generators.

3.3.1 Calculation of Security Profile Templates

The calculation of security values, that is, the structure of the security profile templates, has been updated with regard to the weight of the important security requirements. Previously, the formula for the calculation of the security values was defined as

$$SP_k = \left[\prod_{j=1}^{m} rfv(rf_j) \right] \cdot \left(v + (1-v) \cdot \left(\sum_{i=1}^{n} rfv(ri_i) \cdot e_{ki} \right) \right)$$

where the weight was specified as v, $0 < v \leq 1$, (Hallberg et al, 2006). v is included in the formula since no fulfilled important security requirements should not yield the result zero. However, there is a drawback of introducing this scheme. If there are no important security requirements, the maximum value of SP_k will be v, not 1. Moreover, v is global for the whole security profile template. Thus, it is not possible to specify different values of v for the specific security values, SP_k, of the security profiles.

To address these two issues, the influence of the important security requirements should depend on the relation between the number of fundamental and important security requirements. Consequently, the weight of the important security requirements $(1 - v)$, corresponding to a specific security feature, is defined as

$$1 - v = \frac{n}{m+n}$$

where n is the number of important requirements and m is the number of fundamental requirements. Consequently, v becomes

$$v = 1 - \frac{n}{m+n} = \frac{m}{m+n}$$

Hence, if there are no important requirements, $n = 0$, their weight will become zero, since

$$1 - v = \frac{n}{m+n} = 0, \quad m > 0$$

On the other hand, if there are no fundamental requirements, $m = 0$, the important requirements get a weight of 1 in the calculations, since

$$1 - v = \frac{n}{m+n} = 1, \quad n > 0$$

Thus, the formulas for the calculation of the security values to be included in the security profiles, that is, the security profile template, becomes

$$SP_k = \left[\prod_{j=1}^{m_k} rfv\left(rf_{kj}\right) \right] \cdot \left(\frac{m_k}{m_k + n_k} + \frac{n_k}{m_k + n_k} \cdot \left(\sum_{i=1}^{n_k} rfv\left(ri_{ki}\right) \cdot e_{ki} \right) \right) \quad \text{(Eq. 1)}^1$$

where n_k is the number of important requirements and m_k is the number of fundamental requirements for security feature k. Furthermore, the formula has been updated to clarify that the sets of important and fundamental security requirements are specified for each security feature, for example rf_{kj} refers to the fundamental requirement j regarding security feature k.

3.3.2 Calculation of Filter Profile Templates

The equation of the filter profile template has been redefined to clarify that the filtering capability values are specified for each security feature, k. Thus, the filter profile template is defined as

$$FP_k = S_k \cdot \sum_{i=1}^{N} \left(e_{ki} \cdot fcv_k\left(ffr_i\right) \right)$$

[1] If there are no fundamental requirements, $m = 0$, the product of a series of terms in Eq. 1 is the empty (nullary) product, which has a value of 1.

where S_k is the filtering influence factor, e_k is a vector containing the weights of the filter functional requirements for the filter profile value k, and $fcv_k(x)$ returns the filtering capability value, [0, 1], of requirement x regarding security feature k.

3.3.3 Combination of Multiple Paths between Traffic Generators

The calculation of the resulting security profile, RSP, has been redefined. That is, how the presence of several paths between traffic mediators is regarded in the calculations of the SSP:s. Previously, it has been implicitly defined as

$$RSP_i^{nb} = \prod_{j=1}^{n} ESP_i^{nb,j}$$

where n is the number of paths between the two entities and $ESP^{nb,j}$ is the equivalent security profile of the neighbor nb via path j (Hallberg et al, 2006). By using this definition of the RSP^{nb}, two paths resulting in the same ESP^{nb}, as shown in Figure 4, would result in $RSP_i^{nb} = (ESP_i^{nb})^2$, which is not reasonable. Therefore the calculation of the RSP has been redefined to.

$$RSP_i^{nb} = \min\left(ESP_i^{nb,1},...,ESP_i^{nb,n}\right)$$

Thereby the resulting security profile, RSP^{nb}, is the element-wise minimum of the equivalent security profiles, ESPs, of the paths.

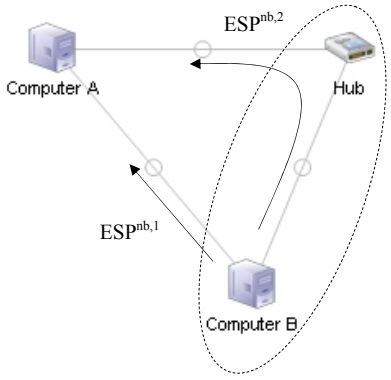


Figure 4: An example of multiple paths.

4 Creating Profile Templates

This chapter describes the design of security profile templates and filter profile templates for the XMASS. These tasks are vital in order to create an instantiation of the XMASS that can be used for system security assessment. The design processes are based on the methodology included in XMASS (Hallberg et al, 2006).

4.1 Creating a Security Profile Template

Security profile templates are created according to the first five of the six steps presented in section 0. The sixth step is to use the security profile template to calculate the entity security profile and is therefore disregarded in this section. The following design example of a security profile template use the security requirements specified by the Swedish Armed Forces (2004) for information systems, referred to as the KSF (Sw. Krav på SäkerhetsFunktioner). The steps 2 to 5, in section 0, are performed for one of the selected security features.

4.1.1 Step 1 – Decide on a Set of Security Features

The first step is to decide on the set of security features to use. The KSF defines a set of seven different security functions, but the functions addressing *compromising emanations* (CE) and *protection against unauthorized interception* are judged to be outside the scope of the XMASS and are therefore not included. The remaining five security functions are used as security features. Hence, the resulting set consists of five security features, which are specified in Table 3.

Table 3: The selected set of security features.

Security Features
Access Control
Security Logging
Protection against Intrusions
Intrusion Detection
Protection against Malware

The design of security profile templates based on the selected set of security features requires five formulas of the form presented in Eq. 1 at the end of section 3.3.1 to be designed. In order to decrease the extent of the following steps of this process, only the template for the security feature Intrusion Detection is created.

4.1.2 Step 2 – Decide Security Requirements

For each security feature specified in step 1, a set of security requirements has to be decided. These security requirements specify what needs to be assessed in order to decide the security value associated with the specific security feature. Thus, a set of requirements has to be selected regarding Intrusion Detection. In the KSF, the security requirements are grouped according to the clearance level of the system. For the levels restricted, confidential, and secret, the KSF defines twelve security requirements regarding intrusion detection, which are specified with a short description in Table 4. For a complete description of each security requirement, see Appendix A.

Table 4: The selected security requirements for Intrusion Detection.

Requirement ID	Short description
ID1	Maintain the security domain, which protects against intrusions and disturbances.
ID2	Have the possibility to provide for reliable time.
ID3	Only authorized administrators can maintain the security function.
ID4	Detection of already performed as well as ongoing intrusions.
ID5	Register time and date of each event.
ID6	All registered events can be presented interpretably for authorized persons and inspection of the registered events can be performed.
ID7	Enable tool-based inspection of registered events.
ID8	Trace misuse as well as attempts of misuse endangering security.
ID9	No registered events are erased, overwritten or destroyed.
ID10	Conclude, through automatic analysis, whether defined rules have been violated.
ID11	Ensure that registered events can be analyzed together with security relevant events.
ID12	The security function for intrusion detection shall resume at a defined secure state.

4.1.3 Step 3 – Divide the Security Requirements

The set of security requirements of each security feature needs to be divided into the sets of fundamental and important security requirements. If a security re-

quirement is judged to be of such importance that failure to fulfill it would ruin the security feature, that is, decrease its security value to zero, then it is a fundamental security requirement. The fundamental security requirements thereby represent all the security requirements that have to be fulfilled and where failure to fulfill one or more of them results in a security value of zero for that specific security feature. On the other hand, if the failure to fulfill a specific security requirement will not result in making the security feature worthless, then it is considered to be only an important security requirement.

A group of nine security experts at FOI was asked to divide the security requirements for intrusion detection into groups of fundamental and important security requirements. The aggregated result of the expert survey is presented in Table 5 and Table 6. The results of the voting are presented in Appendix B.

Table 5: Selected fundamental security requirements.

Fundamental Security Requirements
ID1 Maintain security domain
ID3 Authorized system administration
ID6 Interpretable presentation and possible inspection
ID12 Resume at secure state

Table 6: Selected important security requirements.

Important Security Requirements
ID2 Reliable time
ID4 Detection of intrusion
ID5 Registration of event time and date
ID7 Tool-based inspection
ID8 Trace misuse
ID9 Registered events secure
ID10 Automatic analysis
ID11 Registered and security-relevant event analysis

4.1.4 Step 4 – Prioritize the Security Requirements

The next step is to prioritize the important security requirements according to their pair-wise relative importance. Seven of the security experts at FOI were

asked to carry out this prioritization. The individual prioritizations were aggregated into the matrix A_k, presented in Table 7. The matrix was assembled by taking the median of the security expert's weights for each pair-wise comparison. The prioritization of each security expert is presented in Appendix B.

Table 7: Table of weights, A_k.

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11
ID2	**1**	1/3	3	1/3	1/3	1/5	1/3	1/3
ID4	3	**1**	3	1	1	1/3	1	2
ID5	1/3	1/3	**1**	1	1/3	1/5	1	1
ID7	3	1	1	**1**	3	1/3	1	1
ID8	3	1	3	1/3	**1**	1/3	1	3
ID9	5	3	5	3	3	**1**	3	3
ID10	3	1	1	1	1	1/3	**1**	3
ID11	3	1/2	1	1	1/3	1/3	1/3	**1**

4.1.5 Step 5 – Calculate the Security Profile Template

The fifth and final step of calculating the security profile template is to calculate the scaled values of the eigenvector corresponding to the maximum eigenvalue of the matrix A_k, defined in Table 7. The calculations result in the security profile template described in Table 8.

Table 8: The calculated weights of the important security requirements to be included in the security profile template for intrusion detection.

Security Requirement	Priority
ID2 Reliable time	0.05642708
ID4 Detection of intrusion	0.12638000
ID5 Registration of event time and date	0.06286048

Security Requirement	Priority
ID7 Tool-based inspection	0.13170758
ID8 Trace misuse	0.12523142
ID9 Registered events secure	0.29879861
ID10 Automatic analysis	0.12094164
ID11 Registered and security-relevant event analysis	0.07765318

To get a measurement of the consistency of the resulting priorities, a consistency ratio is calculated, as suggested by Saaty (2004). The consistency ratio, CR, is calculated as

$$CR = \frac{\left(\frac{\lambda_{\max} - n}{n-1}\right)}{RI}$$

where λ_{max} is the maximum eigenvalue, *n* is the dimension of the matrix, and RI is the random index for the specific matrix dimension. For this example, the dimension of the matrix is 8, which gives a random index of 1.40 (Saaty, 2004), and the maximum eigenvalue is 8.83865523. This results in a consistency ratio of 0.085577. Saaty (2004) recommends a CR not greater than 0.10 in order to have a consistent decision. The priorities, based on the aggregated matrix, thereby seem to represent a consistent decision.

The resulting security profile template is thereby

$$SP_{ID} = \left[\prod_{j=1}^{m} rfv\left(rf_j\right)\right] \cdot \left(\frac{1}{3} + \frac{2}{3}\begin{pmatrix} rfv(ID2) \\ rfv(ID4) \\ rfv(ID5) \\ rfv(ID7) \\ rfv(ID8) \\ rfv(ID9) \\ rfv(ID10) \\ rfv(ID11) \end{pmatrix}^{T} \cdot \begin{pmatrix} 0.05642708 \\ 0.12638000 \\ 0.06286048 \\ 0.13170758 \\ 0.12523142 \\ 0.29879861 \\ 0.12094164 \\ 0.07765318 \end{pmatrix}\right)$$

4.2 Creating a Filter Profile Template

Filter profile templates are created according to the first three steps specified in section 2.3.2.

4.2.1 Step 1 – Decide a Set of Requirements

The initial step is to decide a set of requirements regarding filter functionality. In the filter profile templates, the same set of filter functional requirements is used for all security features. For this example the set of filter functional requirements suggested by Hallberg et al (2006) is used. The suggested set of filter functional requirements along with definitions is provided in Table 9.

Table 9: Definitions of the filter functional requirements.

Filter Functional Requirements
Packet filtering (FF1) An adequate set of rules for the specific traffic mediator has to be defined. This set has to be applied to each IP packet, which is thereby forwarded or discarded. Both IP packets from and to the internal network have to be filtered.
Stateful-inspection (FF2) A state table has to be kept to keep track of each currently established connection. Incoming traffic to high-numbered ports should only be allowed for packets that map to an entry in the state table.
Application layer gateway (FF3) The application layer gateway (proxy server) has to support address and port translation for application-layer protocols. Moreover, users are authenticated before application-level connections are established.
Circuit level gateway (FF4) The circuit level gateway should not permit end-to-end TCP connections. Instead it should establish one connection each to the inside and outside hosts and relay the TCP segments between the connections.
Network address translation (FF5) For outgoing traffic, the addresses of the source hosts should be replaced by the address of the traffic mediator. Incoming traffic has to be forwarded to the correct destination.

4.2.2 Step 2 – Prioritize the Requirements

The filter functional requirements are prioritized according to their pair-wise relative importance regarding the specific security feature. Here, the calculations are done for a single security feature, the security feature for protection against intrusions. Consequently, the group of security experts at FOI was asked to judge the importance of these requirements regarding protection against intrusions. The aggregated result, the matrix B_k, is presented in Table 10. B_k was assembled by taking the median of the security experts' judgments for each pair-wise comparison. The prioritization of each security expert is presented in Appendix C.

Table 10: The median of the security expert's judgments for each pair-wise comparison, B_k.

	FF1	**FF2**	**FF3**	**FF4**	**FF5**
FF1	**1**	1/3	1/2	3	3
FF2	3	**1**	3	3	3
FF3	2	1/3	**1**	3	1
FF4	1/3	1/3	1/3	**1**	1/3
FF5	1/3	1/3	1	3	**1**

4.2.3 Step 3 – Calculate the Filter Profile Template

The final step of calculating a filter profile template is to calculate the scaled values of the eigenvector corresponding to the maximum eigenvalue for matrix B_k, specified in Table 10. The calculated weights of the filter functional values are presented in Table 11. The consistency ratio is calculated in the same way as for the prioritization of the important security requirements (section 4.1), but with a dimension, n, of 5 and a random index, RI, of 1.11. The result is a consistency ratio of 0.095487966. Thereby, it is reasonable to assume that the decision is consistent (Saaty, 2004).

Table 11: The calculated filter profile template.

Filter Functional Requirement	Priority
FF1	0.19290973
FF2	0.40409665
FF3	0.19610141
FF4	0.07006776
FF5	0.13682445

The resulting filter profile template is thereby

$$FP_{PI} = S_k \cdot \begin{pmatrix} 0.19290973 \\ 0.40409665 \\ 0.19610141 \\ 0.07006776 \\ 0.13682445 \end{pmatrix}^T \cdot \begin{pmatrix} fcv_{PI}(FF1) \\ fcv_{PI}(FF2) \\ fcv_{PI}(FF3) \\ fcv_{PI}(FF4) \\ fcv_{PI}(FF5) \end{pmatrix}$$

4.3 Reflections on Results

The creation of security and filter profile templates is vital in order to enable system security assessments based on XMASS. The processes for the creation of the templates were specified by Hallberg et al (2006).

The first two steps of the process of creating the security profile template were, in principle, resolved by the decision to use the security requirements, and their grouping, specified in the KSF. The fifth step was performed based on the results presented in section 3.3.1.

The third step, categorizing the security requirements as fundamental or important, was performed through a voting among security experts. The results are hardly unanimous, but for six out of the twelve requirements the decisions were strong (at least 7 to 2). In three cases, the difference was merely a single vote, making them important instead of fundamental. This indicates a need to more thoroughly define the requirements.

The fourth step, the prioritization of the important requirements, was performed individually by the security experts based on the AHP approach for criteria

weighting. It was experienced that AHP provides relevant mechanisms for the validation of the consistency of the results.

The first step of the process of creating the filter profile template was, in principle, resolved by the decision to use the filter functional requirements specified by Hallberg et al (2006). The third step was performed based on the results presented in section 3.3.2.

The second step, the prioritization of the filter functional requirements, was performed individually by the security experts based on the AHP approach for criteria weighting. It was experienced, among the security experts, as difficult to judge the relative importance of the requirements. This can be resolved through a more elaborated structuring of the requirements, for example, by clustering dependent requirements, before the prioritization.

5 Development of Assessment Tool Environment and Assessment Tool

The realization of assessment methods in software is a demanding process where time needs to be spent on implementing basic tasks like file handling as well as the actual method. Assuming a fixed amount of resources to spend on each implementation, time spent on the implementation of basic functionality leads to less thoroughly implemented assessment methods. On the other hand, if there is a development environment providing the basic functionality, the developers can spend more time on the implementation of the actual method and thereby produce software of higher quality and better extensibility.

To aid the development of tools based on the presented assessment methods, an assessment tool environment has been designed and implemented in the .NET framework. This work is presented in sections 5.1 to 5.7 of this chapter. Within this environment an implementation of the XMASS has been performed, as described in section 5.8. This chapter is based on the work presented by Bengtsson & Brinck (2007).

5.1 Design

The development in the area of security assessment is an ongoing process. Thus, methods are constantly refined requiring the implementations to be updated or rewritten from scratch. The research on novel methods is often branched into different approaches, which leads to branching in the software development resulting in a multitude of different versions.

Since the development of assessment methods still is in an early stage, the main group of users of tool implementations is researchers who want to evaluate and compare different assessment methods. Thereby, the user and the developer, implementing the tool, are in many cases the same person. Even so, the assessment tool environment, referred to as the NTE (New Tool Environment), is designed with both developers and users in mind to better identify relevant needs and requirements.

5.1.1 Developer Perspective

To facilitate the development of assessment tools, the NTE is designed to provide simple interfaces between software modules. By having interfaces which

only contain the functionality needed and nothing else, the interfaces become a form of guideline which assists the developer during the implementation. Moreover, well defined interfaces enable the developers to spend less time on figuring out the control flow of the program and instead concentrate on writing code for the methods defined by the interfaces.

A great deal of time is normally spent by the developer on implementing functionality to store data on and recover data from secondary storage. The required functionality includes conversion of data from the object based representation of the program to a suitable representation on disk. The amount of time spent on this kind of functionality is reduced by implementing a data access layer between the database and the tool implementation. Thereby the developer should be able to send objects directly to the data access layer, which handles the storing in the database. The layer should be able to rebuild the stored objects when requested.

The design of the graphical user interface is an important part of the implementation of a method and the NTE must therefore not limit the possibilities of the resulting tools, but rather assist the developer in the design. To be able to assist the developers with designing the graphical user interface, the NTE should provide a library of components which can be shared by tool implementations. This should streamline the design process and at the same time support conformity between the different tool implementations.

5.1.2 User Perspective

Two different tasks have been identified as the main tasks of a user:

- to evaluate methods by observing changes in the result upon changes in the input and
- to compare methods, or different versions of the same method, by comparing the results based on the same set of input.

These two tasks coincide with the task of security assessment, where different solutions may be compared through the variation of system data and different aspects of the same system may be assessed through the use of several assessment methods.

The generalized view of an assessment tool, presented in Figure 5, illustrates the steps from input to output. In the modeling phase, the tool is assumed to perform system modeling and, thereby, transform the input into systems and resources. The resources are components with associated security values, which are derived from the input, like for example a specific computer or firewall with security

values associated to the corresponding security characteristics of these components. A system is defined as a configuration of these resources, such as a model for how computers are connected in a network. The systems are grouped into projects to improve the usability and the possibilities to experiment with different configurations. Hence, a project is defined as a collection of related systems that are sharing some common resources. Moreover, the modeling phase includes the specification of the computations model used for the computing of security values. The final step of the assessment tool is to perform calculations where the security values and relations of the relevant resources are transformed into assessment results, which constitute the output of the tool.

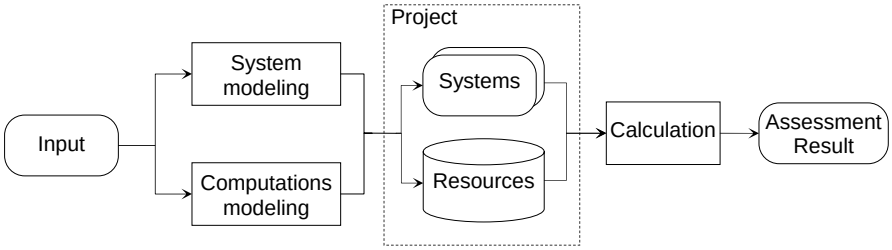


Figure 5: A generalized view of an assessment tool.

The introduced notion of systems facilitates the evaluation of methods by letting the user store different configurations of the resources, that is, study the effects of variations of the input. These configurations can be compared and conclusions may thereby be drawn from how they affect the results.

The grouping of systems into projects provides the user with the possibility to get a more organized view of the systems and, at the same time, enables the use of common resources.

As stated earlier, the user should be able to compare different assessment methods, which means the NTE has to be flexible enough to enable the use of different methods. By having the assessment methods implemented as plugins to the NTE, the user may create several projects using different assessment methods. Thereby all assessment tools can be gathered into one application, which eliminates the need to use a combination of different applications to reach an assessment result. Since the assessment tools can share the same input, the task of comparing different assessment methods is supported.

5.2 Structure Overview

An overview of the NTE structure is presented in Figure 6. The following sections describe the role of each of these main parts.

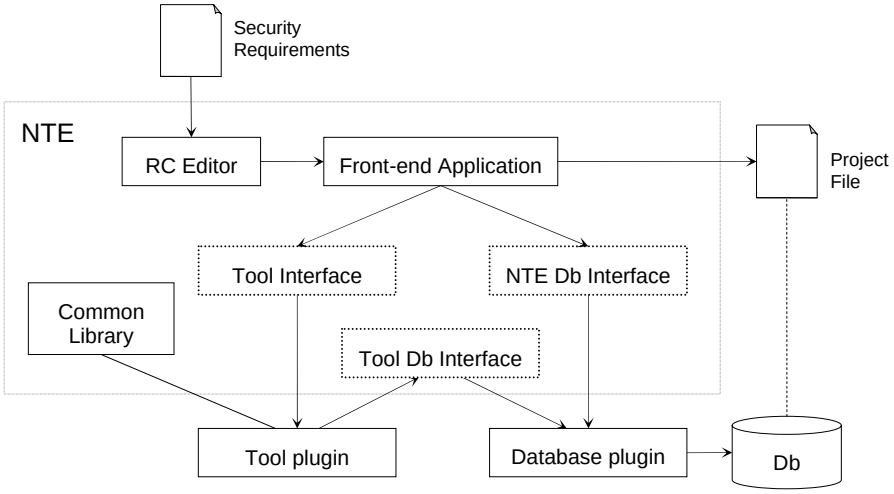


Figure 6: Schematic view of the structure of the environment.

5.2.1.1 Front-end Application

This is the part of NTE, along with the Requirement Collection Editor, which is visible to the user. Its task is to provide project handling and a workspace to be used by the tool plugins.

5.2.1.2 Plugins

As illustrated in Figure 6, there are two types of plugins which can be used by NTE: tool plugins and database plugins. A tool plugin is an implementation of an assessment method, while a database plugin contains the functionality needed to handle a database. The NTE is built to handle multiple plugins of each type in order to support the use of multiple tools and databases within different projects. Plugins are selected for each project, which results in a structure where all systems within a project use the same tool plugin.

5.2.1.3 Interfaces

To enable communication between the front-end application, the tool plugin and the database plugin, three different interfaces are needed. The communication between the front-end application and the tool plugin is handled through the tool interface. As illustrated in Figure 6, there are two different interfaces for communicating with the database plugin. Both the front-end application and the tool plugin need to communicate with the database plugin, but they have different needs when it comes to functionality. Each one of the two interfaces includes the functionality needed and nothing more. Hence both interfaces are as straightforward to use as possible.

5.2.1.4 The Common Library

The Common Library contains various components intended to be used by the tool plugin developers. Most components are GUI (Graphical User Interface) related and reduce the amount of time the tool plugin developers need to spend on GUI implementations.

5.2.1.5 Project File

The project file is basically an encapsulation of the project related data that needs to be stored. Information about the tool plugin and database plugin used in projects is needed by the NTE in order to be able to open the systems within the projects. Information about each system within a project is also needed in order to list the systems of a project without being forced to load the plugins. Hence each project file contains the following:

- information regarding the used tool and database plugins,
- information about all the systems within the project, and
- a database file containing the systems of the project.

5.2.1.6 Requirement Collection Editor

The Requirement Collection Editor is a built-in tool for the creation and alteration of requirement collections. To make sure requirement collections are accurately created and altered, the built-in editor is the only way to create and modify the requirement collection files used by NTE. The structure of a requirement collection is described in section 5.6.

5.3 Plugin Handling

In NTE, tool and database plugins are compiled as DLL:s, Dynamic-Link Libraries, and placed in the root folder of NTE. The files in the root folder are traversed when the NTE starts up and the assembly, that is, the partially compiled code library, from each DLL file is loaded and its classes are examined. In .NET, an assembly is a partially compiled code library for use in deployment, versioning and security. The assembly from DLL files having classes implementing either the tool interface or the database interface is stored in a dictionary indexed by the name and version of the plugin. When the user performs an operation that depends on a specific plugin, the correct assembly is looked up in the dictionary and an instance of the class is created. This instance is stored in a local variable as the active plugin of the corresponding type.

5.3.1 Tool Plugins

The structure of a tool plugin is not limited in any way other than that it should extend the UserControl class and implement the tool plugin interface described below. By extending the UserControl class, which is the base class for user defined graphical components, the plugin can be docked into the main workspace area of NTE. Tool plugins have access to the main menu and the status strip for further integration with NTE.

5.3.1.1 Tool Plugin Interface

The tool plugin interface, shown in Figure 7, is used for communication between the NTE and the tool plugin. Each tool plugin must implement the interface in order for the NTE to recognize it as a tool plugin. The interface specifies the functions called when the user performs an operation on a system, such as opening or saving.

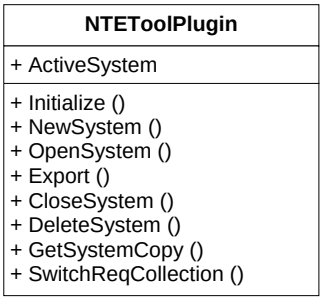


Figure 7: Class diagram of the tool plugin interface.

5.3.2 Database Plugins

In NTE, a database plugin is a data access layer operating on objects. To distinguish the differences in needs between the NTE and the tool plugins, the interface to the database plugin is divided into two parts, one to be used by the tool plugin and one to be used by the NTE.

The database plugin requires the existence of an interface to recognize classes defined by tool plugins. By introducing the empty interface NTEDbClass, these classes can be identified at the same time as it does not add any limitations for the tool plugin developers. The use of an empty interface can basically be seen as a flag for marking classes defined by tool plugins.

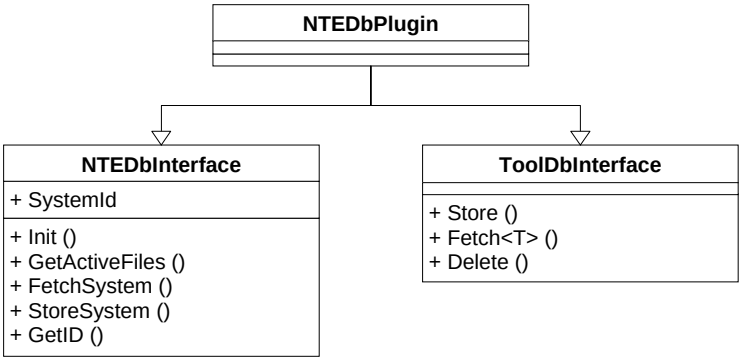


Figure 8: Class diagram of the database interface.

5.3.2.1 NTE Database Plugin Interface

The NTEDbPlugin interface, illustrated in Figure 8, is an empty interface, which implements both the tool database interface and the NTE database interface. A

database plugin must implement the NTEDbPlugin interface in order to be recognized by the NTE. Implementing this interface also assures that the database plugin contains the functionality needed by both the NTE and the tool plugin.

5.3.2.2 Tool Database Interface

The database interface used by tool plugins is designed with simplicity in mind and therefore only contains three methods which correspond to the basic operations on objects in a database. The three operations are: store, fetch and delete.

The method for storing is defined as a function taking an arbitrary object as a parameter. Objects of types defined by the tool plugin are required to implement the NTEDbClass interface in order to be stored in the database.

The method for fetching is a generic nullary function with a parameter T, where T is the type of the object which should be retrieved from the database. By having a generic parameter in the function call, the function can return a generic list with the same element type and, thereby, avoids having to cast the results into the correct type. A constraint is set in the interface to make sure that the type T implements the NTEDbClass interface, thus enabling type checking to be performed at compile time.

The method for deleting objects from the database takes the object to be removed as a parameter. The objects, and their relations to other objects, are permanently removed from the database and cannot be recovered.

5.3.2.3 NTE Database Interface

The NTE database interface is used by the NTE for communicating with the database plugin. The NTE takes care of the handling of systems and therefore needs to access methods operating on IDs. These methods, FetchSystem and StoreSystem, have the same functionality as the methods for fetching and storing in the tool database interface, except for the method for fetching taking an ID as a parameter and the method for storing returning an ID. The interface also defines a method for retrieving the ID of an arbitrary object and an attribute specifying the ID of the active system (GetID).

Furthermore, the initialization method of the database plugin is defined as a function taking three arguments: the path to the working directory of the plugin, a set of database files, and finally the ID of the system to work with (Init). The database plugin may create new files or remove existing ones during its execution,

which is why the interface also defines a method for retrieving the active set of files (GetActiveFiles).

5.4 Common Library

The Common Library contains a range of components that can assist tool plugin developers while designing user interfaces. To assist the developers as much as possible, the components in the common library need to be as accessible as the other components available in the .NET platform. Therefore, it is possible to use the components in the common library directly in the design view of Microsoft Visual Studio, which results in the possibility to edit the attributes of the components directly in the design view. The process of designing the user interface is also simplified by the possibility to place the components in the design view.

The common library contains a variety of components ranging from a textbox with functionality to handle numeric values, to a form for making AHP prioritizations. The common library can be extended with more components in the future as new needs arise.

5.5 Project Handling

The NTE defines a project as a file containing a collection of systems using the same tool plugin and the same database plugin. Requirement collections are shared between the different systems within a project and it is also possible to share information which is specific to the selected tool plugin. Thus, a project can be seen as a database of components where its systems represent different configurations of these components. The user can thereby experiment with different configurations of the same components.

5.5.1 Structure of the NTE File Format

As described in the structure overview in section 5.2, the NTE contains its own file format used for storing projects. The NTE file is an encapsulation of the project related data which needs to be stored. The structure and content of a database is only accessible by using a specific database plugin along with a specific tool plugin. Hence, the NTE would have to invoke the related plugins in order to gain knowledge about what is actually stored in a database. This means that in order for the NTE to keep track of the systems stored within each project it would need to call the related plugins. To avoid this, additional information is stored inside the NTE files.

The structure of the classes related to the NTE file is shown in Figure 9. There is one instance of the class SysInfo for each system in the project and it contains the name of the system, a description, the time of creation, the time of last modification, a screenshot and the ID of the requirement collection currently in use. Moreover, each instance of the class SysInfo contains an ID associated with the actual system specification stored in the database. This information is used for the listing of the systems of each project.

While listing available systems, the NTE has to verify that the needed plugins are available since it is not possible to open systems from a project without the right versions of plugins. Hence an instance of the PluginInfo class is stored for both the database plugin and for the tool plugin in order to keep track of the names and versions of the plugins.

The actual database files of a project are stored as instances of the class DbFile. Each DbFile contains the data members called filename and data. While opening a system, each database file of the project will be recreated in the working directory of the plugin as a file with the given filename containing the given data.

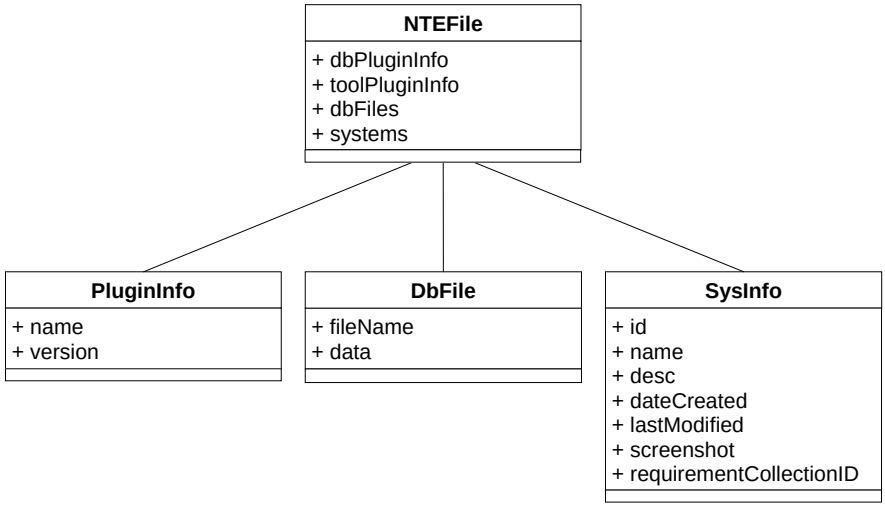


Figure 9: Class diagram of the NTE file classes.

5.6 Requirements Handling

Requirement collections are introduced in the NTE to organize the handling of the system characteristics and effects whose security values are used as input for the assessment. In NTE, these characteristics and effects are referred to as security requirements. Consequently, a requirement collection consists of a set of security requirements, where each requirement belongs to a security feature, as illustrated in Figure 10.

This straightforward two-level structure was designed to facilitate modeling of the relations between basic input and the initial aggregation of security values. For example, it reflects the structure of the security requirements specified for information systems, called requirements on security functionality (Swe. Krav på SäkerhetsFunktioner, KSF), and used by the Swedish Armed Forces (2004). In the KSF, the security properties and qualities of an IT system are described by a set of security features. For each security feature a set of security requirements is defined, describing what needs to be fulfilled in order to fulfill the security feature. Thus, the security features can be seen as corresponding to the security-relevant characteristics of the assessed systems, as specified in the process model for security assessment described in section 2.2.2. This is the case considering the KSF. However, tool plugins can treat the security features as intermediate results and perform further computations to produce their output. For example,

the XMASS tool plugin use the security features of requirement collections as the security profiles of system entities. These security profiles are used to compute the system-dependent security profiles, which in turn can be aggregated to produce the final output of the tool.

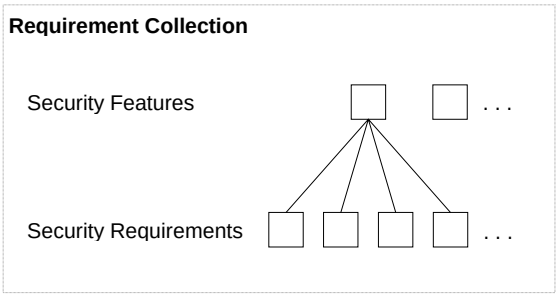


Figure 10: The structure of requirement collections.

In NTE, a requirement collection is implemented as illustrated in Figure 11. Each class has an ID since the objects in a requirement collection are compared at ID level instead of at object level. Thereby requirements used in different projects can be compared to avoid the creation of duplicates when, for example, exporting systems between projects. This special treatment requires extra functionality in the database layer, which is described in section 5.7.2.

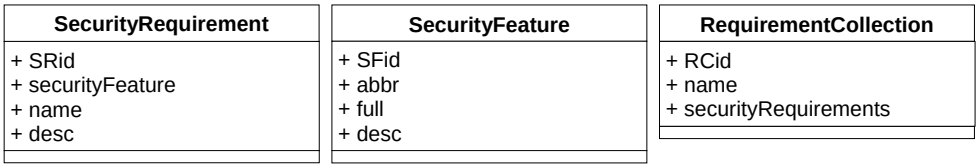


Figure 11: Class diagram of requirement collections.

Requirement collections are stored as files with the extension .rc. The only occasion these files will be used is when creating a new project or adding a new collection to an existing project. When this occurs, the content of the requirement collection file is stored as a new requirement collection in the database of the active project. In case a requirement collection with matching ID already exists in the database, the database is instead updated with the new version of the requirement collection from the file.

The NTE needs to ensure that requirement collections are handled in a proper way. Therefore, it contains a requirement collection editor where the user can

create and edit collections. Using this editor is the only proper way to create valid requirement collection files. Thus, the NTE has control of the handling of requirement collections. The need for this control is due to the importance of generating new IDs when a requirement collection has been edited and is not considered to be intact. A requirement collection is intact if no requirements or features have been added or removed. Thus, the user can alter names and descriptions and still have an intact collection, which is considered to be the same collection as earlier. Adding an altered, but intact, requirement collection to a project already containing the original collection will only update the names and descriptions instead of storing the collection as a new one. If a collection, on the other hand, is not intact, new IDs are generated for all security features, security requirements, and the collection itself. This results in the altered collection being seen as a completely new one.

The NTE provides functionality for switching between the requirement collections available in the project. All available collections appear in the main menu where the user can select which collection to be active. When the active collection is switched, the SwitchReqCollection method of the tool plugin is called to notify the tool plugin about the change.

The NTE also provides another type of requirements called filter functional requirements, which are used to describe the different kinds of functionality in network components filtering traffic. The set of filter functional requirements is fixed and is therefore statically implemented into NTE. The available filter functional requirements, listed in Table 12, were specified by Hallberg et al (2006).

Table 12: Filter functional requirements.

Filter Functional Requirements
Packet filtering
Stateful-inspection
Application layer gateway
Circuit level gateway
Network address translation

5.7 DbSQLite

As concluded in 5.1.1, the data access layer should operate on objects and thereby make it possible for the developers to effortlessly store and retrieve data. This does not add any requirements on the actual database engine. Hence it will

not limit the database plugin developer in the choice of a suitable database engine. For the database plugin described in this section a relational database is used. The database is powered by the SQLite database engine; a small C library implementing a self-contained and embeddable SQL database engine, which needs no setup or administration (SQLite, 2007).

The database plugin consists of three layers (Figure 12). The top layer, closest to the database interface, is called the object cache and contains functionality to ensure object consistency. The bottom layer is the actual database engine that performs the queries on the database. Between these two layers is the object-relational mapping layer, which performs the conversion between objects and SQL code.

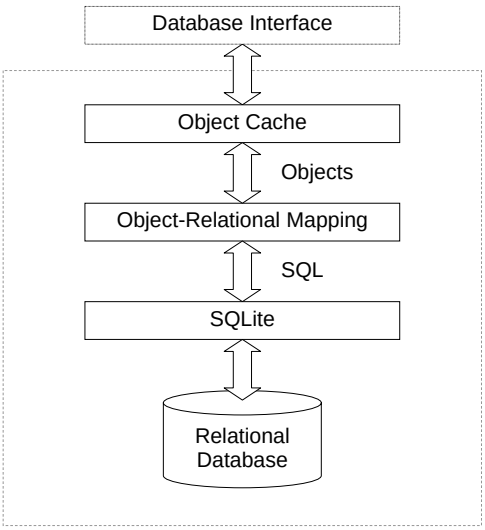


Figure 12: Schematic view of the structure of the database plugin.

5.7.1 Object-Relational Mapping

The database interface of the NTE requires the database plugin to be able to operate on objects. Hence, the database plugin must be able to translate the classes of the object model into tables in the relational model, and back again when they are fetched from the database. This problem was described by Peak & Heudecker (2006) and is referred to as the Object/Relational impedance mismatch.

The solution to the translation problem is often referred to as Object-Relational Mapping, ORM. The ORM forms a layer between the object cache and the database engine and takes care of the conversion of objects to and from the relational format. To support the developer, the mapping between the object model and the relational model is transparent. This is achieved by using data type reflection, which is a feature of the .NET framework granting access to class structure information at runtime. Thereby both the type and data of each member of an object can be extracted without knowing its representation in advance.

At compile-time no information about the classes in the different tool plugins is known by the database plugin. Therefore the ORM layer needs to handle the creation of tables in the database dynamically at runtime. A description of how this dynamical runtime creation of tables works is provided by Bengtsson & Brinck (2007).

5.7.2 Object Caching

The database interfaces that are designed to only pass instances of objects between the tool plugins and the database plugins result in two issues to be resolved, as pointed out by Peak & Heudecker (2006). These two issues could also be seen as a single translation issue seen from two different angles.

Firstly, each fetch operation on a specific entry in the database always has to return the same reference to the object representing the entry. Otherwise, the consistency of the object structure will be broken since the tool plugin will receive clones of the object. For example, assume there are two instances of the class A, a_1 and a_2, where both instances hold a reference to the same instance of the class B, b_1. If changes are applied to the b_1 member of a_1, these changes must also affect the b_1 member of a_2. Similarly, a comparison of these two b_1 members must evaluate to true.

Secondly, each store operation of a specific object must operate on the same entry in the database. Hence, if an object that is already present in the database is stored, the present entry should be updated with the data of the object instead of inserted as a new entry.

Both problems are solved by introducing the object cache whose main objective is to maintain a table of previously handled objects. Each element in the table is a triple of Type × ID × Object, where Type is the type of the object, ID is the identity of the entry in the database and Object is the actual object. The object cache

contains functionality to look up an object, that is, Type × ID → Object, and to retrieve the ID of an object, that is, Object → ID.

When a fetch operation occurs, the cache will be checked whether it contains a reference to an instance of the given type and ID. If there is a match, the cached reference will be returned instead of a reference to a new instance. Thereby, the consistency between object relations is maintained and the fetch operations are sped up, since re-fetches of objects from the database are avoided.

During a store operation the cache will be checked whether it contains a reference to the object. If a reference is found, its cached ID will be used for updating the entry in the database instead of storing the data as a new entry. If, on the other hand, no reference is found in the object cache, a new entry will be created.

In section 5.6 the need for special treatment of requirement collections is pointed out. All objects that belong to a requirement collection should be compared by ID in order to distinguish between them. This is handled by checking the type of the object of each operation to allow comparison at ID level for types matching any of the three defined classes. All objects of other types are compared at object level.

5.7.3 Database Operations

As described in section 5.3.2, the tool plugin developer can use three different operations on the database through the tool database interface: fetch, store and delete.

5.7.3.1 Fetch

When a fetch operation occurs, the first step is to look up the table structure for the requested class. Table structures are generated based on the data types of the public data members of a class. To avoid regeneration of this information and thereby speed up the database operations, all discovered table structures are stored in a dictionary, which works as a cache. A table structure is thereby looked up by first inquiring the dictionary and thereafter generating the table structure if it is not available in the dictionary.

In the next step, SQL queries are constructed to select the relevant data from the database. For classes with a simple structure the queries are trivial, but for more complex classes the queries contain join clauses in order to resolve any associations. The queries to the database result in a set of matching rows, where the ID

of each row is checked against the object cache to determine the existence of any instance of the given class and ID. In case of no match in the object cache, the constructor of the class will be called with the values retrieved from the database, which results in a new instance of the class. Both the newly created and the cached instances are merged into a list which is returned to the calling tool plugin. All new instances are added to the cache for future use.

5.7.3.2 Store

The first step of the store operation is to look up the table structure for the class in the dictionary. If it is not available in the dictionary, it will be generated. Subsequently SQL queries are created based on the table structure of the class. Depending on the existence of the object in the object cache, the query will be either an insert or an update statement.

Related lists and dictionaries are stored using a recursive function call, which means that the related data first is stored and then the ID of the data is stored in the relating table. While updating a relation, all previous relations must first be deleted since the content of a list or dictionary can, and most likely will, have changed since it previously was stored. This is done with the help of a recursive function call traversing the table information tree and deleting all related data from the database.

5.7.3.3 Delete

The delete operation is used to remove entries in the database by passing along a reference to the object to delete as an argument. The object cache is used to look up the table ID of the given object. If the object is not present in the object cache the call to the delete function will be ignored, since an object that has not been previously fetched or stored can not exist in the database. On the other hand, both the entry in the database and the object cache will be deleted if the object exists in the object cache.

5.8 The XMASS Tool Plugin

Utilizing the support provided by NTE, a tool plugin implementing the eXtended Method for Assessment of System Security (XMASS), referred to as the Security AssessmeNT Application (SANTA), has been created. Firstly, the SANTA supports the use and validation of the XMASS. Secondly, the SANTA enables the functionality of the NTE to be verified. The implementation is based on the

XMASS as specified by Hallberg et al (2006) along with the improvements described in chapter 3.

Software implementations of security assessment methods are important for the evaluation of the corresponding methods. Since no previous implementation of the XMASS exists, the implementation is useful for evaluation purposes of the method, while it at the same time verifies the functionality of NTE. Therefore the design focus for this tool plugin has been on creating an implementation where values and settings easily can be varied. By implementing a graphical user interface that fulfils this design focus, it should be easier to see how alterations of the input affect the assessment result. With knowledge of how a certain variation should affect the security, it may be possible to draw conclusions from the assessment result about the soundness of the implemented assessment method.

How to use the SANTA in combination with the NTE is further described in (Bengtsson & Brinck, 2007b).

5.8.1 Graphical User Interface

A graphical user interface (GUI) should be built upon the existing knowledge of the users. By using familiar concepts in the GUI, the users more quickly learn how to use the application. This could for example be achieved by using language and expressions that are familiar to the users (Galitz, 2007). The main users of this assessment tool implementation are researchers who want to evaluate or compare methods. Hence the target group for SANTA can be assumed to be familiar with both the XMASS and the area of security assessment in general and thereby recognize most of the used notations.

Designing the GUI in a way similar to other applications of the same genre as the user previously has been working with improves the level of familiarity and thereby leads to easier orientation. To achieve this familiarity for SANTA, the design is based on the design used in ROME2, which is the software implementation of the predecessor of the XMASS called the MASS. In reality, this means that the different parts of the GUI are placed in a similar way as in ROME2.

Compared to ROME2, there are more parameters in the SANTA that need to be set in order to perform an assessment. Therefore it is important to find a logical way to group these settings so that it, to some extent, is obvious for the user where to look. By placing all settings regarding a specific topic, for example, system-wide parameters, in the same dialog, it becomes more straightforward to

locate the specific controls. At the same time it gives a clear overview of all possible parameter settings. Another example is the profile manager. Even though it is possible to manage each specific type of profile from the dialog where that specific profile is selected, there is a profile manager where all types of profiles can be managed. The profile manager results in an overview of the available profiles. Thus, the users will not have to, for example, create a traffic mediator in order to be able to see and manage the available filter profiles.

The workspace of the XMASS plugin is designed to allow faster and easier system modeling compared to ROME2. This is mainly achieved by using different mouse click events. An entity is for example created by a double click on the left mouse button while a relation is created by doing a drag-and-drop between two entities using the right mouse button.

5.8.2 Presentation of Results

The support for evaluations of the XMASS is enhanced by proper presentation of assessment results. How to present the results is not included in the specification of the XMASS. Hallberg et al (2006) give examples of possible approaches, but state that it all depends on the specific needs of the users. Thus, the results presentation in SANTA is based on the identified needs of the users. Consequently, SANTA enables the presentation of multiple assessments of the same system model but with varying input security values.

The XMASS tool plugin provides automatic assessment of the modeled system when all needed settings have been made. The left side of the workspace is used for presenting the assessment results, which consists of results regarding both the whole system as well as the currently selected entity. If not all parameters have been set, a list of absent settings will be shown instead of the assessment results.

Taking the evaluation possibilities of the XMASS one step further, the plugin contains functionality to aggregate over the results from assessments, that is, to compare the results of different assessments using varied inputs, as illustrated in Figure 13. By specifying start and end states for the security values in the system, a series of intermediate states using linear progression from the start state to the end state is created. Each state is assessed as normal, but the result is stored in a vector instead of displayed in the left side of the workspace. When all states have been assessed, the vector is sent to a presentation view which displays the result as graphs for the different components of the system. This solution enables the user to perform evaluations on the system without altering the original state.

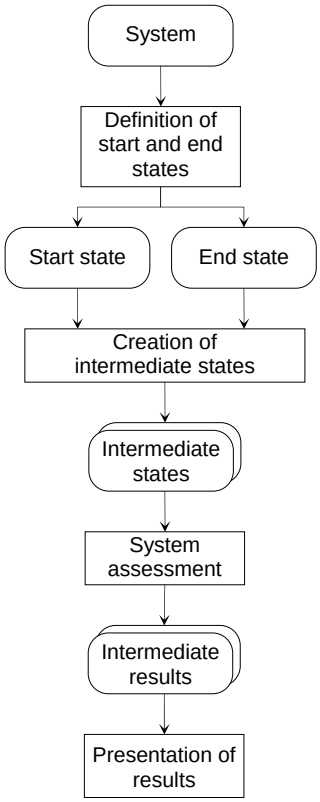


Figure 13: Result aggregation and presentation in the XMASS.

6 Implementation of the Process Model for Security Assessment

In this chapter, an approach to security assessment based on the process model for security assessment and Bayesian networks is introduced. The process model and Bayesian networks are briefly described in chapter 2. Moreover, an example with an assessment based on the KSF is presented.

6.1 Security Assessment Method

The use of Bayesian networks as a base for the proposed security assessment method allows different security functions to be evaluated, independently of the degree of knowledge about the different functions. The evaluation of the security functions could be based on facts, statistics, or assumptions. First a good-enough assessment should be established, containing all important aspects of the system influencing the security characteristic to be assessed. If that evaluation is based, to some degree, on assumptions rather than facts, an assessment increasing the knowledge about the security status of the system can still be made. In most cases, this would be impossible if validated facts about all security functionality of the system were required. Another attractive property of Bayesian networks is that more details can be added later on, and the effects of adding these facts can be observed.

Another advantage of using Bayesian networks is that the security values will actually mean something; they are probabilistic values of the modeled states of studied nodes. A hypothesis is created and the security assessment indicates whether this hypothesis is likely to be true or not. This implies that the dependencies between functions in the system model need to be distinguishable, and it should be possible to see how these dependencies affect the system as a whole. Furthermore, by using Bayesian networks in security assessment, already developed methodology and models, from other areas of research, can be utilized.

In the following subsections, the proposed method is described following the steps of the process model (section 2.2.2). Considering the integrated approach of the proposed method, some of the subtasks of the process model are not relevant for describing the method and have, consequently, not been regarded.

6.1.1 Analyze Needs

Methods encompassing the whole assessment process have to consider the analysis of needs. All security assessments should be based on these specified needs of the users, owners of the system, or other stakeholders that in some way benefit from the results. To find the needs, these stakeholders must first be identified. They may consist of individuals, organizational units, business processes, or software. Then, the actual needs of the identified stakeholders have to be identified. This can be achieved using formalized methods, such as the method for needs analysis regarding security assessment referred to as MedBeVIS (Hallberg et al, 2005), or less stringent methods.

Independently of which approach is used to formalize the needs, it is important that the results are documented in order to establish the purpose of the assessment. Moreover, the identified needs should be ratified by the stakeholders. The result from this first step of the assessment is a set of, possibly structured, security assessment needs.

6.1.2 Define Relevant Security Characteristics

The purpose of this step is to decide the relevant IT security characteristics of the system. The assessment of these characteristics should produce the information desired by the receiver of the results. In other words, when the security assessment needs have been established, the relevant security characteristics that these needs are covering should be found. Moreover, this activity has to define the full extent of the system, its boundaries to other systems, and the phases of the system lifecycle that have to be considered.

Useful input to this activity is relevant sets of IT security characteristics, as well as the needs for assessment defined during the previous activity. To transform the needs into the desired characteristics, quality-based requirements engineering methodology (Hallberg, 1999) can be used.

It can be hard, especially initially, to completely model how the characteristics fulfill the needs. The use of Bayesian networks as a structure for the modeling enables assumptions. These can be changed when the knowledge of the system increases and perhaps other characteristics are found to better fulfill the needs. Since Bayesian Networks handles probabilities, the IT security characteristics should be formulated like probability hypotheses, for example, the probability that an intruder will acquire sensitive information from a computer on the studied network.

The results of this activity are:

- a set of relevant IT security characteristics,
- specifications of the relations between the decided characteristics and the needs, and
- a specification of the system to be assessed.

6.1.3 Connect to System Characteristics and Effects

The purpose of this step is to determine how to decide the security values corresponding to the relevant security characteristics. Thus, security effects and characteristics that can be measured or computed should be structured and modeled in such a way that they can be used to compute security values for the relevant security characteristics to be assessed.

When the assessment is based on statistics and user know-how, the step *mapping onto system* will be closely related to the two following steps *measure* and *compute security values*. This is because the security values are modeled rather than computed or measured.

If the security characteristics to be assessed are too complex, they have to be broken down into less complex, measurable characteristics. This requires understanding of how the security functions work in reality, how the system characteristics affect each other, what is needed in the system in order to increase security levels, and what might result in security breaches. It has to be analyzed how all these less complex characteristics affect the assessment of the relevant security characteristics. Since the structure of Bayesian networks consist of directed acyclic graphs (DAG), the nodes representing the measurable system characteristics have to refer, directly or indirectly, to the relevant security characteristic to be assessed. The node representing this security characteristic is the final node in the DAG, which all the other nodes are pointing towards. If there is more than one relevant security characteristic to assess, they could be modeled in the same graph. Then the nodes representing the measurable security characteristics can lead to either one of the relevant (final) nodes. However, to decrease the complexity of the models each relevant security characteristic can be modeled separately, even though many of the supporting characteristics would be identical in the different models.

When specifying the DAG, it is most straightforward to start from the targeted relevant security characteristics and a set of lower-level characteristics. Thereaf-

ter, a computations model, which links them together and describes how the security functions relate to each other, is constructed. This work is best performed with a sandwich approach working both bottom-up and top-down to build a consistent structure, with intermediate characteristics inserted as needed to connect the other nodes. There is no need to use all available low-level characteristics and effects as long as the relevant security characteristics are adequately assessed.

6.1.3.1 System Modeling Regarding Entities

In order to model the system entities, it should be specified exactly what entities the system consists of. That is, the granularity and type of content of the system model is decided. The model should adhere to the system borderlines, include the system aspects, and cover the system phases specified in the previous step (section 6.1.2). However, the proposed method does not prescribe any structural modeling of the assessed system. The system entities consist of the security characteristics as well as the system characteristics and effects included in the Bayesian network. For example, system model entities could be security logging mechanisms and requirements rather than network components and organizational units. The proposed method does not presently compute entity security values prior to the final system security assessment; rather the entire system is evaluated as a whole.

The proposed method starts by looking at the entire system at an abstract level in order to create a comprehensive system model providing an overview with less detail. Thus, initially, the model should include the relevant security characteristics identified earlier. The model is specified in more detail later on, when the system model is populated with system characteristics and effects covering the relevant aspects. When the main structure of the model has been decided, it becomes more straightforward to analyze and decide on the details to be included in the model.

6.1.3.2 Identification of Measurable System Characteristics and Effects

It is up to the assessors to identify measurable system characteristics and effects in the proposed method. Support to map the wanted security characteristics to measurable system characteristics and effects can be found in proposed sets of security requirements for security functionality. Examples, at the level of computer and network components, are the Common Criteria (CC, 2004) and the

KSF (Swedish Armed Forces, 2004). These could be used as a starting point, or as a way of understanding the details of the security functionality.

6.1.3.3 System Modeling Regarding Characteristics and Effects

When the measurable system characteristics and effects to be included in the assessment have been identified, the system model has to show how these attributes relate to each other. That is, the relevant security characteristics included in the Bayesian network should be broken down to specific, measurable nodes. In the proposed method, the systems and computations models are tightly integrated. As the computations model takes shape, it is possible to assign values to each measurable node and to observe the overall effect the assigned values have on the assessment of the system. This makes it possible to adjust the system model so the nodes affect the assessment appropriately.

6.1.3.4 Specification of Computations Model

As mentioned earlier, the computations model for Bayesian networks has the structure of a directed acyclic graph (DAG). Starting with the system model as a network of nodes, the computations model has to be specified in order to decide how these nodes should influence each other.

Every relation from a parent node to a child node is analyzed in a top-down manner to decide how much the parent node should influence the child node. Thereby, the relevant security characteristics are the only non-parents in the DAG and the measurable system characteristics and effects are the non-children. At this stage, it does not matter whether the analysis is based on statistics, on the subjective experience of the users, or basic evaluations and measurements. The activity should result in a combined system and computations model consisting of a DAG, where every node has a reasonable influence on the end nodes, that is, the relevant security characteristics to be assessed.

For every node in the model, two or more states are defined, for example *secure transmission* and *insecure transmission*. These states should be disjunct and cover every possible state of the node. Thus, the node should always be in exactly one of these defined states.

After the states have been defined, the probability for each state in the node is estimated or evaluated. For nodes that lack parents and, consequently, do not depend on other nodes, this step is more straightforward since only the probabilities of the defined states of the nodes need to be estimated. For nodes that depend

on other nodes (i.e. have parent nodes), the probability estimation of each state need to be made for each combination of all the possible states that the parent nodes can be in. This implies that the probability for a specific node to reach a certain state strongly depends on the parent nodes. For example, assume that node A has the states a_1 and a_2 and node B has the states b_1 and b_2. If both A and B affect node C, then all the states of node C should be estimated for every combination of the states of node A and B, that is, a_1^b_1, a_1^b_2, a_2^b_1, and a_2^b_2. This means that if A has n_a states, B has n_b states, and C has n_c states, then $n_a \cdot n_b \cdot n_c$ estimations have to be made.

The result from this step is a DAG, where each node represents a security characteristic or effect whose security value can be measured or computed from the values of other nodes. A relation from a parent node to a child node reflects that the security value of the parent node will affect the value of the child node. The security characteristics should only depend on the characteristics represented by the modeled parent nodes. This can be difficult to achieve, since there are often hidden dependencies in the characteristics, and capturing all the system characteristics affecting the node may require extensive work.

The result of this activity is a complete Bayesian network.

6.1.4 Measure Security Values

This step handles the measuring of the security values that constitutes the input to the assessment method. This may be accomplished through statistical evaluation or decisions based on experience.

Before any security values are measured, the system model has to be checked in order to verify that all the data required is either possible to deduce from the system model or its source is clearly identified, and thus possible to retrieve. Since the computations and system models are tightly integrated, the proposed method inherently supports this.

The system characteristics and effects are measured or judged to decide the corresponding security values. It is possible to choose the assessment method for each characteristic independently of the other characteristics. Some entities can be assigning security values subjectively, while other values are objectively measured. The possibility to choose measurement methods freely is one of the strengths of the proposed method, since objective measurements sometimes can

be difficult to achieve. However, objective measurements should be included in assessments whenever possible.

The result of this activity is that all the non-children nodes of the DAG have been assigned values.

6.1.5 Compute Security Values

Since the computations model is described as a Bayesian network, there are several tools available that can be used for the actual computations. In the example below, GeNIe from Decision Systems Laboratory (2007), University of Pittsburgh is used to implement the Bayesian network. Since the DAG and the probabilities of the states of the non-children, that is, the measurable nodes, have been defined, the values of all the children nodes can be computed.

The result of this step is a graph, where each node consists of security states with associated probabilities.

6.1.6 Interpret Security Values

One of the major advantages of using Bayesian networks for security evaluations is that meaningful metrics are inherited directly. The metrics consist of the probabilities of specified states, and the results of the assessment will thus specify these probabilities.

6.2 Example

In this section, an example of how the proposed method can be used is presented. The example is placed within the framework of an intuitive scenario.

6.2.1 Analyze Needs

A government agency has recently identified a legal requirement demanding the capability to account for every possible security breach concerning the information systems of the agency. Especially breaches that result in information leaks to non-authorized parties are of importance. This demand leads to that an assessment process is initiated at the agency to establish exactly in what way the system has to be updated to fulfill the requirement. Although this example constitutes a fairly clear case concerning the needs for the security assessment, a needs analysis is initiated to formalize the needs.

The stakeholders are identified as the management of the agency. They are asked to contribute with relevant data, which in this example consist of the legal texts in question. Based on these texts and interviews with the management, relevant needs are identified. The identified core need is the ability to be able to assess the strength of the security logging. Several other needs are probably also important in order to satisfy the legal demand mentioned in the scenario, for example, the ability to show that actions have been taken to secure the data needed for forensic investigations. However, for this example, the single need of evaluating the logging functionality of the computer system of the agency was deemed sufficient.

The identified need is documented together with references to the underlying data and the stakeholders. This is important in order to maintain traceability from the need to its origin. Thereafter, the identified need is presented to the stakeholders in order to acquire their ratification of the need.

The outcome of this activity is a need that the organization deems important enough to justify the continuation of the assessment process.

6.2.2 Define Relevant Security Characteristics

When the need has been established, the relevant security characteristics that correspond to this need have to be found. If these security characteristics are assessed, the identified assessment need is fulfilled.

In this example, the KSF collection of security requirements (Swedish Armed Forces, 2004) is used as a mean to identify the relevant security characteristics. The KSF has been defined by the Swedish Armed Forces to support the certification of the security of information systems. By studying the KSF, useful facts about the security logging functionality can be found. These facts can be used to model the characteristics. An alternative to the KSF is the Common Criteria (CC, 2004), where the logging function is explained in great detail and broken down into less complex functions.

Thus, security logging is defined as the relevant security characteristic whose assessment will fulfill the need of the stakeholders. Thereafter, the relations between the relevant security characteristic and the assessment need should be specified. In this example, there is a direct mapping between the relevant security characteristic (security logging) and the desired need (ability to establish the strength of the security logging).

Finally, during this activity, the system scope has to be specified. The system to be assessed in this scenario encompasses the whole information system of the agency. However, the assessment is limited to the technical system aspect, not because the non-technical areas are unimportant, but rather because they require additional efforts, which will not be covered in this example. Considering the technical aspects is assumed to be an adequate starting point for more comprehensive assessment.

6.2.3 Connect to System Characteristics and Effects

In this example, the steps System modeling regarding entities, Identification of measurable system characteristics and effects, and System modeling regarding characteristics and effects are integrated into one since the system entities are comprised by the security and system characteristics. Again, information from the KSF is used in order to find the necessary security and system characteristics and how they relate to each other. Moreover, other functions, which the logging functions depend on but not belong to, are analyzed and described. An example of this is the ability to discover misuse and attacks in the system.

The KSF lists twelve different security requirements regarding security logging that should all be fulfilled for the logging functionality to be regarded secure enough (Appendix A, Table 17). In the KSF, there are three separate lists of security requirements corresponding to systems where the highest level of classification for the information handled by the system is restricted, confidential, and secret respectively. Keywords have been selected from the specification of the KSF requirements to summarize their descriptions (Table 13).

Table 13: List of security requirements for security logging based on the KSF.

Req. id	Description
SL1	Maintain its own security domain
SL2	Provide reliable time
SL3	Only authorized administrators can maintain the security function
SL4	Register events that are of relevance
SL5	Register date and time of events and the identity of the user or subject
SL6	Tracking of misuse
SL7	Security log can be presented in readable format
SL8	Tool-based inspection
SL9	Back up of the security log
SL10	No registered events are erased, overwritten, or in other ways destroyed

Req. id	Description
SL11	Maintain a defined secure state
SL12	No user activity takes place in the system if the security logging is inactive

Originally, the assessment of systems based on the KSF is supposed to state whether the requirements are fulfilled or not. However, there are several reasons for allowing intermediate values, resulting in more nuanced security values which can be used for judging the adequateness of the security of systems. Firstly, there is the possibility to grade the possible implementations of security functionality differently. Secondly, systems have to be considered as interacting entities. Thus, the fulfillment of the requirements may vary throughout the system, resulting in more complex values for the overall system.

At this point, the relevant security characteristic and some of the system characteristics to be measured have been decided. Now, the relations between these characteristics have to be defined, that is, how they all fit together in the model. For this purpose, the sandwich approach of the proposed method is used. Thus, the relevant security characteristic is broken down into smaller, less complex, intermediate characteristics. Thereafter, the system characteristics are associated with these intermediate characteristics, in order to connect all the nodes of the model. Some of the intermediate characteristics are additional characteristics, identified as complement to the list extracted from the KSF. The additional characteristics identified are listed in Table 14.

In order for security logging to work, it is dependent on the following security characteristics:

1. detection of security-relevant events (SC1),
2. secure storage of logs (SC2),
3. analyze logs to detect misuse, faults and attacks (SC3), and
4. maintain security domain (SL1).

SC1, SC2, and SC3 are created to establish nodes in the model that cover all aspects of secure logging. The four characteristics above are all at the same level of detail. If these four abstract characteristics are fulfilled, it could be said that the security logging has the possibility to function correctly and effectively (Figure 14).

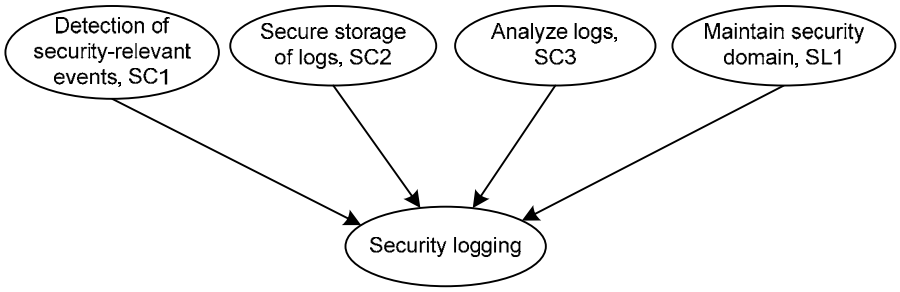


Figure 14: Initial Bayesian network model.

In the next step the intermediate characteristics are analyzed further in order to relate them to the rest of the characteristics. SL6, *tracking of misuse*, is mapped to the *detection of security-relevant events* (SC1). SL7, *security log can be presented in readable format*, and SL8, *tool-based inspection*, are placed under *analyze logs* (SC3). SL3, *authorized administrators*, is connected to SL1, *maintain security domain*.

The other characteristics are harder to place, even though most of them deal with similar tasks; the physical data storage, or lack of storage. In order to be able to add these characteristics to the model, other characteristics need to be added. Thus, the characteristics *actions when security logging is inactive* (SC4), *physical creation of logs* (SC5) and *physical storage of logs* (SC6) are added to the model (Figure 15). These will work as a layer between the characteristics that have been included in the model, and the ones remaining. Without them, the difference between the levels of abstraction of the characteristics would be too large, making it hard to analyze how the characteristics affect each other.

Table 14: The additional security characteristics identified.

Req. id	Description
SC1	Detection of security-relevant events
SC2	Secure storage of logs
SC3	Analyze logs to detect misuse, faults and attacks
SC4	Actions when security logging is inactive
SC5	Physical creation of logs
SC6	Physical storage of logs

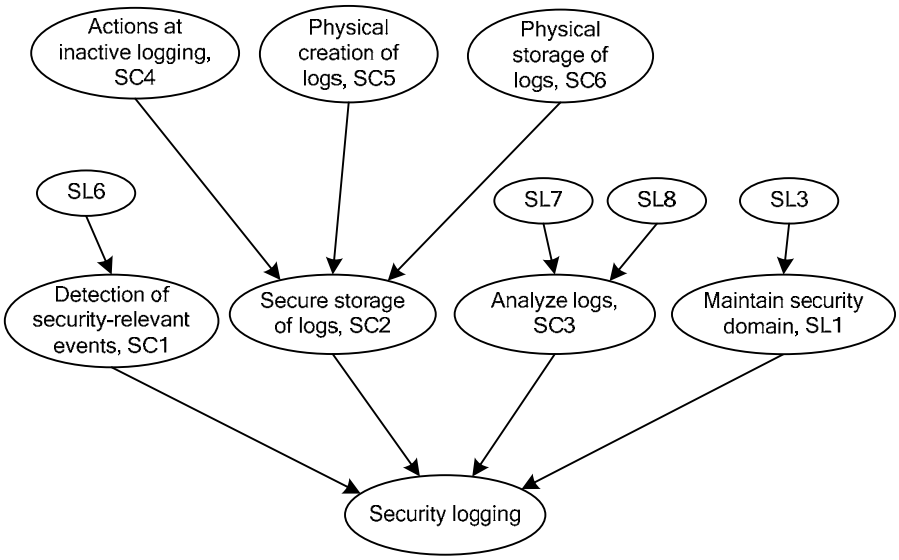


Figure 15: Intermediate Bayesian network model.

Further characteristics are now mapped to the intermediate nodes SC4, SC5, and SC6. *Maintain a defined secure state* (SL11) and *No user activity takes place in the system if the security logging is inactive* (SL12) are connected to *Actions when security logging is inactive* (SC4). *Back up of the security log* (SL9) and *No registered events are erased, overwritten, or in other ways destroyed* (SL10) belong to *Physical storage of logs* (SC6). *Provide reliable time* (SL2), *Register events that are of relevance* (SL4) and *Register date and time for the event and the identity of the user or subject* (SL5) all belong to *Physical creation of log* (SC5). However, they depend on each other. This is because SL5 depend on relevant events being registered (SL4) with accurate time (SL2). Moreover, *Register events that are of relevance* (SL4) affects *Analyze logs* (SC3).

The result is security characteristics structured as a DAG (Figure 16), where every characteristic points to the final node, i.e. *Security logging*. That means that all characteristics from the KSF, as well as the ones created by the assessor, are structured in a system model in form of a Bayesian network, where it is shown how they all affect each other and the final node.

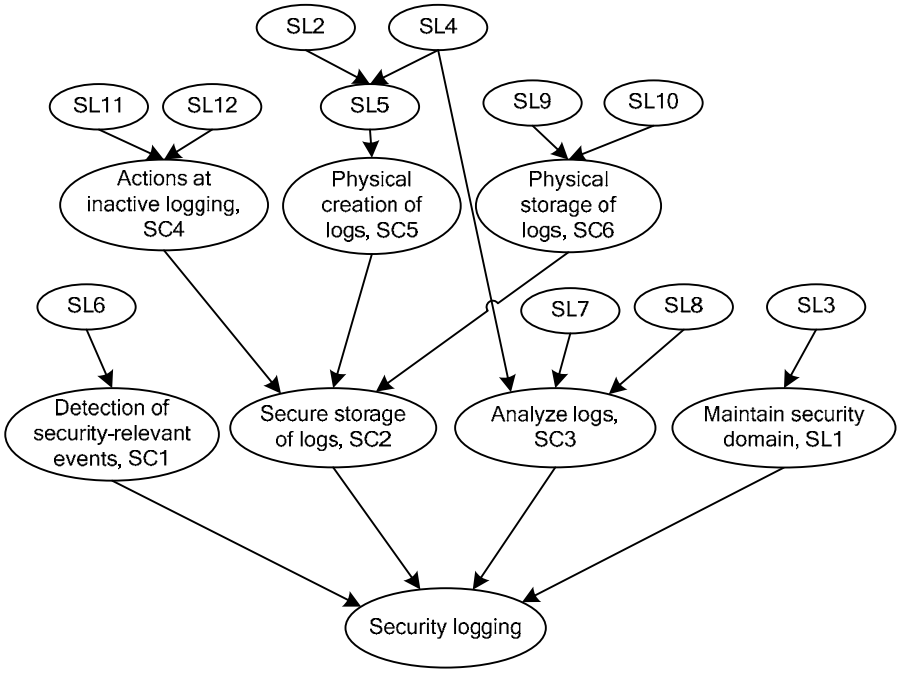


Figure 16: Bayesian network model resulting from the system modeling.

When the system model has been completed, it is time to specify the computations model, that is, to specify how the nodes affect each other in the Bayesian network. Since the assessor in the example has no possibility to measure the security values or use any form of statistics, the analysis is based entirely on the expertise and judgments of the assessor as a security expert. In this text, only the part of the Bayesian network consisting of the nodes SL2, SL4, and SL5 is treated. The rest of the nodes are calculated in the same manner, until a security value for the final node; *security logging*, is reached.

The states of all the nodes are to be defined. Thereafter, the states are to be estimated according to their probabilities. Regarding the node representing accurate time (SL2), two states are possible: the time is either reliable or unreliable.

It is more complex to estimate how SL4, *register events that are of relevance*, affects SL5, since it affects SC3, *analyze logs*, as well. Thus, the occurrences of both relevant and irrelevant events have to be considered. Since this makes SL4, and its influence on SL5 and SC3, hard to model, it is divided into the two separate nodes SL4-1 and SL4-2, modeling the occurrence of relevant and irrelevant

events respectively. Thus, SL4-1 concerns the successful or unsuccessful storage of relevant events. Stored irrelevant events will make it harder later on to search for relevant events in the logs. This is handled by SL4-2. The Bayesian network model has to be updated to reflect these changes (Figure 17).

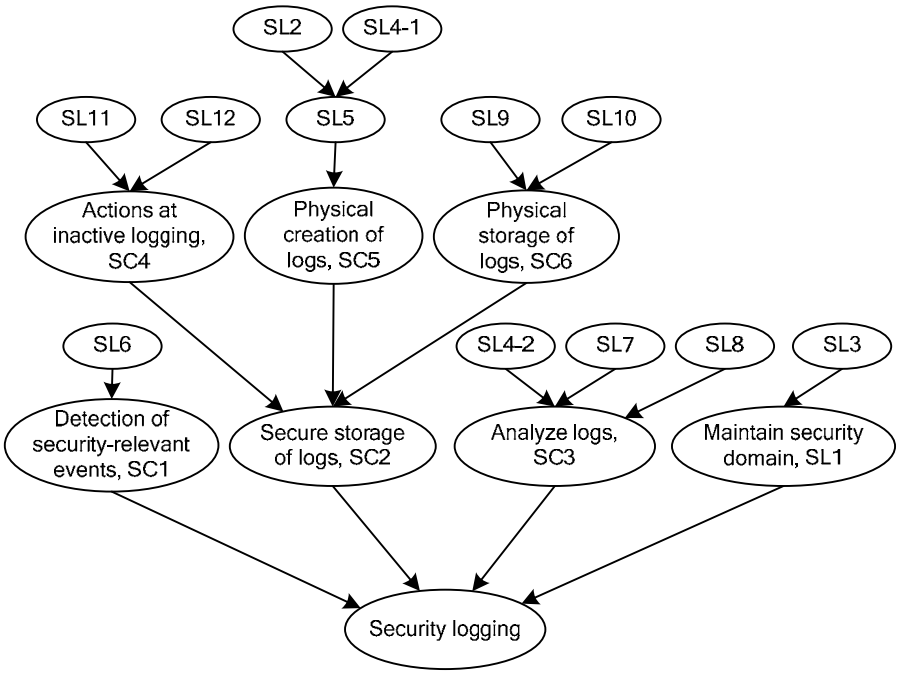


Figure 17: Bayesian network model resulting from the alterations during the computations modeling.

When analyzing SL5, three different states were found: *relevant events logged with reliable time*, *relevant events logged (without reliable time)* and *relevant events missed*. These states of SL5 should be defined based on SL4 and SL2. Both SL4-1 and SL2 have two states that both are relevant for the estimation of SL5. As there are three new states that should be defined for two times two dependable states, there are twelve (3·2·2) probabilities to be estimated (Table 15).

Table 15: Estimations of the probabilities of the three states of SL5 for each relevant combination of the states of SL2 and SL4-1.

SL4-1: Store relevant events	Relevant event stored		Relevant event not stored	
SL2: Provide reliable time	Reliable time	Unreliable time	Reliable time	Unreliable time
Relevant events logged with reliable time	0.9	0.1	0	0
Relevant events logged	0.1	0.9	0	0
Relevant events missed	0	0	1	1

6.2.4 Measure Security Values

The measurement activity should supply probability values for the states of the nodes which are not depending on other nodes. As for the specification of the computations model, the assessor in the example has no possibility to measure the security values or use any form of statistics. Thus, the measurements are based entirely on the expertise and judgments of the assessor as a security expert.

Previously, the states for SL2 were defined as either reliable or unreliable time. It is now estimated that accurate time is supplied in 99% and inaccurate time in 1% of the cases. Then, the probability prior estimations of the two states of SL4-1 *relevant event stored* and *relevant event not stored* are specified as 90% and 10%.

6.2.5 Compute Security Values

The computations model is implemented in the GeNIe tool (Decision Systems Laboratory, 2007). Using the security values from the previous section, the security values corresponding to SL5 are computed (Figure 18). The computations result in calculated probability values for the dependable states of SL5. The probability of a relevant event logged with reliable time is 80%, the probability of a relevant event missed is 10%, and the probability of a relevant event logged (without reliable time) is 10%.

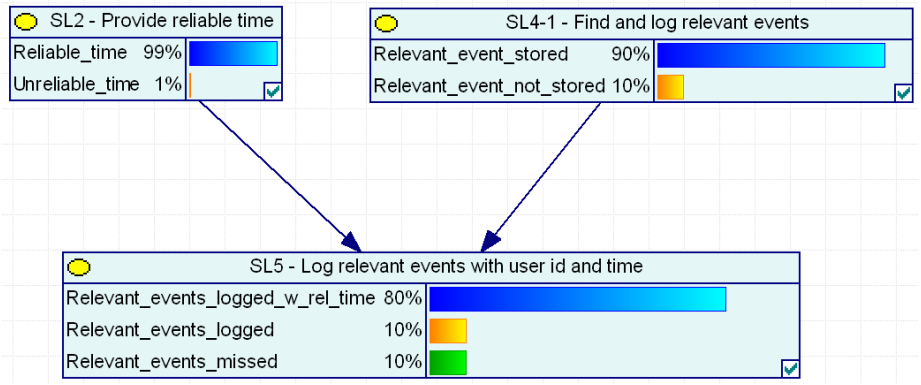


Figure 18: Example of computed security characteristics from the GeNIe tool.

6.2.6 Interpret Security Values

The result of the assessment is a variety of states with computed probabilities for each state. The method of Bayesian networks suggests that a hypothesis should be stated for the final node. When a hypothesis has been formulated, the interpretation of the resulting security value is straightforward. One hypothesis, useful in the context of this example, could be "there should be less than 1% probability of the state *relevant events missed*".

7 Conclusions

Assessing the security of networked information systems is difficult, but nevertheless important. Two of the main reasons for the importance of efficacious security assessment methods are:

1. the ongoing integration of systems, which makes it impossible to comprehend the resulting systems and the security effects caused by all actions affecting the system without the aid of proper security assessment methods and
2. the need to incorporate security mechanism and thinking in all processes relating to these systems, which results in security assessment needs relating to, for example, systems requirement engineering and configuration management.

Currently, there is a lack of methods addressing all the steps necessary for security assessments and all the relevant system aspects. A remaining issue is the aggregation of measured values into meaningful high-level security values.

In this report, two approaches to security assessment are taken. The first approach supported by the eXtended Method for Assessment of System Security, XMASS, (Hallberg et al, 2006) starts from detailed knowledge about the entities (components) of systems and aggregates this knowledge into system security values. Here, the XMASS is improved, extended, and implemented as a software tool.

The second approach is illustrated by a method realizing the process model for security assessment (Hallberg et al, 2007). The method is based on Bayesian networks and starts by extracting the needs for security assessment. Thereafter, the relevant security characteristics, whose assessment will answer the identified needs, are decided. These relevant security characteristics are connected to measurable system characteristics and effects, possibly via intermediate computable characteristics. The measurable system characteristics and effects are assigned values and thereafter the higher-level security values, including those corresponding to the relevant security characteristics, are computed. Finally, the assessment is completed by the interpretation of the security values associated to the relevant security characteristics. This method illustrates how security can be methodically assessed, without access to all the details of complex systems, considering all the activities necessary for complete assessments. As the assessments

evolve, the method supports the inclusion of additional details. Thus, evolutionary security assessment is supported.

Moreover, the report describes the design of an environment for the implementation of security assessment software tools. The environment, referred to as NTE, alleviates the need to implement method specific databases for the storage of system and computational models as well as assessment results. Furthermore, the specification of system characteristics and results constituting the input to assessments is supported.

There are numerous tasks that should be undertaken in order to further support the development of security assessment methods and tools, as indicated by the following short list.

- A set of profiles for the XMASS should be assembled.
- A study of the underlying reasons for the assessment results provided by the XMASS should be undertaken.
- Alternative methods for the selection of priorities for the important security requirements and the filter functional requirements used in the XMASS should be studied.
- Real-word assessment should be performed with XMASS as well as the proposed method.
- Bayesian network assessments can be performed for all system entities included in system-wide security assessments. Thus, the proposed method based on Bayesian network could be combined with a structural method, such as the XMASS.

Bibliography

ACSA (2002), *Proc. Workshop on Information Security System Scoring and Ranking,* Applied Computer Security Associates, http://www.acsac.org/measurement/proceedings/wisssr1-proceedings.pdf

Anderson, R. (2001). *Security engineering: A guide to building dependable distributed systems*, Wiley.

Andersson, R. & Hallberg, J. (2006). *System security assessment – a concept demonstrator*, FOI Memo 1798, Linköping, Sweden.

Ben-Gal, I. (2007). Bayesian Networks, in Ruggeri, F., Kenett, R., & Faltin, F. (editors), *Encyclopedia of Statistics in Quality and Reliability*, John Wiley & Sons.
http://www.eng.tau.ac.il/~bengal/BN.pdf

Bengtsson, J. & Brinck, P. (2007). *Design and Implementation of an Environment to Support Development of Methods for Security Assessment*, Master's Thesis, University of Linköping, LiTH-ISY-EX--07/4022--SE.

Bengtsson, J. & Brinck, P. (2007b). *Using NTE with XMASS*, FOI Memo 2255, Linköping, Sweden.
http://itsecurity.foi.se/dfs/FOI-Memo-2255.pdf

Bishop, M. (2003). *Computer Security – Art and Science*, Addison-Wesley, ISBN 0-201-44099-7.

CC. (2004). *Common Criteria for Information Technology Security Evaluation,* Part 1: Introduction and general model, Part 2: Security functional requirements, Part 3: Security assurance requirements. Version 2.2, January 2004.

Decision Systems Laboratory. (2007). GeNIe Homepage. University of Pittsburgh. (Accessed 2007-12-06).
http://genie.sis.pitt.edu/

Forman, E. & Selly, M. (2002). *Decisions by Objective - How to Convince Others That You are Right*, World Scientific Publishing Company, ISBN 978-9810241438.

Gacic, D. (2006). *FSA – Framework for Security Assessment of Distributed Information Systems*, Master's thesis, Royal Institute of Technology, Stockholm, Sweden.

Galitz, W. (2007). *The Essential Guide To User Interface Design – An Introduction to GUI Design Principles and Techniques*, Wiley Publishing, ISBN 978-0-470-05342-3.

Geer, D. (2006). *Measuring Security*, Lecture Notes, Training program M3. 15th USENIX Security Symposium, Vancouver, Canada. July 31-August 4, 2006.

Gollmann, D. (2006). *Computer Security*, 2nd ed, John Wiley & Sons, ISBN 978-0470862933.

Hallberg, J., Hunstad, A., Bond, A., Peterson, M., & Påhlsson, N. (2004). *System IT Security Assessment*, FOI-R—1468—SE, Defence Research Establishment, Linköping, Sweden.

Hallberg, J., Hunstad, A., & Peterson, M. (2005). *A Framework for System Security Assessment,* Proceedings of the 2005 IEEE Workshop on Information Assurance, West Point, NY, June 2005.

Hallberg, J., Hallberg, N., & Hunstad, A. (2006). *Crossroads and XMASS: Framework and Method for System IT Security Assessment*, Scientific report. FOI-R--2154—SE, FOI, Linköping, Sweden.

Hallberg, J., Hunstad, A., & Hallberg, N. (2007). *Handbok för IT-säkerhetsvärdering (in Swedish)*, FOI Memo 2099, Linköping, Sweden. http://itsecurity.foi.se/dfs/FOI-Memo-2099.pdf

Hallberg, N. (1999). *Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach*, Dissertation No. 596, Linköping Studies in Science and Technology.

Peak, P. & Heudecker, N. (2006). *Hibernate Quickly*, Manning, ISBN 1-932394-41-9.

Saaty, T. (1994). *Fundamentals of Decision Making and Priority Theory – with the Analytic Hierarchy Process*, Vol. VI, RWS Publications, Pittsburgh, USA.

Saaty, L. (2004). Decision Making - The Analytic Hierarchy and Network Processes (AHP/ANP), *Journal of Systems Science and Systems Engineering*, vol. 13, No. 1, pp1-35, March, 2004.

Seddigh, N., Pieda, P., Matrawy, A., Nandy, B., Lambadaris, J., & Hatfield, A. (2004). *Current Trends and Advances in Information Assurance Metrics*, Second Annual Conference on Privacy, Security and Trust, October 13-15, 2004. http://dev.hil.unb.ca/Texts/PST/pdf/seddigh.pdf

SQLite. (2007). SQLite home page (accessed 27 November 2007). http://www.sqlite.org/

Swedish Standards Institute. (2004). *Terminologi för informationssäkerhet*, SIS HB 550, utgåva 2, ISBN 91-7162-576-3.

Swedish Armed Forces. (2004). *Krav på säkerhetsfunktioner – Grunder*, 10 750:78976, 2004-12-20.

Vaughn, R., Henning, R., & Siraj, A. (2003). *Information Assurance Measures and Metrics – State of Practice and Proposed Taxonomy*, Proceedings of the Hawaii International Conference on System Sciences (HICSS-36), Waikoloa, Hawaii, January 6-9, 2003.

Appendix A – The KSF

The KSF documentation (Swedish Armed Forces, 2004) specifies the requirements of the different security features for each classification level. Here, the requirements on systems at the different classification levels have been merged into one list per security feature. In Table 16 to Table 20 these lists of security requirements, compiled from the KSF documentation, are specified. The lists have been translated by FOI from their original formulation in Swedish. The original identification tags are kept for reference and are referred to as KSF id.

Table 16: Security requirements for access control.

KSF id	Req. id	Description
HRG-5-1 HCG-5-1 HSG-5-1	AC1	The security function shall, together with the other security functions in the IT system, maintain its own security domain which protects against manipulation and disturbances, both from subjects and users that belong to or do not belong to this domain.
HRG-5-2 HCG-5-2 HSG-5-2	AC2	The security function shall, together with the other security functions in the IT system, have the possibility to provide for reliable time.
HRG-5-3 HCG-5-3 HSG-5-3	AC3	The security function shall make sure that only authorized administrators can maintain the security function and handle its security settings.
HRBK-4-1 HCBK-4-1 HSBK-4-1	AC4	The security function for access control shall prevent the access to the IT system's subjects and objects of users as well as subjects that are not authorized nor have access rights to the IT system.
HRBK-4-2 HCBK-4-2 HSBK-4-2	AC5	The security function for access control shall uniquely identify and authenticate a user before access to any functionality or provision of access rights is allowed to take place in the IT system which is protected by the security function.

KSF id	Req. id	Description
HRBK-4-3 HRBK-4-3 HSBK-4-3	AC6	The security function for access control shall authenticate a user when: • logging in, • canceling temporary access protection, • changing security attributes for authentication, and • the time for time-limited use of the IT system's resources has expired.
HRBK-4-4	AC7	The security function for access control shall ensure a certain quality of the security attribute, if it is a password, used for authentication, by making sure that the security attribute is provided with: • a minimum period of validity, • a minimum number of approved characters that are used for creating the security attribute, and • a maximum period of validity.
HRBK-4-5 HCBK-4-4 HSBK-4-11	AC8	The security function for access control shall ensure that all users can be made individually responsible (that is non-repudiation) for their actions in the IT system.
HRBK-4-6	AC9	The security function for access control shall use security attributes of users, subjects, and objects as a control mechanism when regulating access.
HRBK-4-7	AC10	The security function for access control shall use password or equivalent as control mechanism and security attribute for authentication.
HCBK-4-5 HSBK-4-5		The security function for access control shall fulfill the requirements for strong authentication in compliance with the HKV MUST ITSA and TSA requirements for signal protection systems.
HRBK-4-8 HCBK-4-6 HSBK-4-5	AC11	The security function for access control shall be able to take automatic precautions in case of failed authentications. Such precautions shall embrace denial of access to the IT system and locking of the affected user account for a certain period of time.
HRBK-4-9 HCBK-4-7 HSBK-4-6	AC12	The security function for access control shall support different specified roles.

KSF id	Req. id	Description
HRBK-4-10 HCBK-4-8 HSBK-4-10	AC13	The security function for access control shall ensure locking of security attributes considered to be revealed for users or subjects that are not authorized nor access rights to the IT system. The locking can be initiated directly or at next log in.
HCBK-5-4 HSBK-5-1	AC14	The security function for access control shall be able to maintain a defined secure state when parts of or the entire functionality containing data relating to; • assigned rights for roles, • users belonging to a role, or • the relations and restrictions of roles are corrupt or inaccessible.
HSBK-4-7	AC15	The security function for access control shall ensure that there is no role, user, or subject, which has access to all subjects and objects that are available in the IT system the security function is meant to protect.
HSBK-4-8	AC16	The security function for access control shall ensure that authorized administrators, whose task is to handle the security in the IT system the security function is meant to protect, by no means have authority or access to the security logs in the same IT system.
HSBK-4-9	AC17	The security function for access control shall ensure that authorized administrators, whose task is to handle and inspect the security logs in the IT system the security function is meant to protect, by no means have the same authority or access as the authorized administrators who handle the security in the same IT system.
HSBK-5-2	AC18	The security function for access control shall be able to maintain a defined state of security when the security attributes used for authentication and access decisions are corrupt or inaccessible.
HSBK-5-5	AC19	The security function for access control shall be able to provide defined and agreed on administrative roles with the possibility to verify the correctness of the executable code that involves the security function.

Table 17: Security requirements for security logging.

KSF id	Req. id	Description
HRG-5-1 HCG-5-1 HSG-5-1	SL1	The security function shall, together with the other security functions in the IT system, maintain its own security domain which protects against manipulation and disturbances, both from subjects and users that belong to or do not belong to this domain.
HRG-5-2 HCG-5-2 HSG-5-2	SL2	The security function shall, together with the other security functions in the IT system, have the possibility to provide reliable time.
HRG-5-3 HCG-5-3 HSG-5-3	SL3	The security function shall make sure that only authorized administrators can maintain the security function and handle its security settings.
HRSL-4-1 HCSL-4-1	SL4	The security function for security logging shall, in a security log, register events that are of relevance for the security of the IT system.
HSSL-4-1		The security function for security logging shall, in a security log, register events that are of relevance for the security of the IT system, including: • the use of control mechanisms for authentication, • access to subjects and objects, and • changes to access control lists.
HRSL-4-2 HCSL-4-2 HSSL-4-2	SL5	The security function for security logging shall, together with each registered event, also register date and time for the event and the identity of the user or subject.
HRSL-4-3 HCSL-4-3	SL6	The security function for security logging shall ensure that tracking of misuse, and attempts to misuse, of the IT system can be performed.
HSSL-4-3		The security function for security logging shall ensure that the tracking of misuse, attempts to misuse, and potential misconfigurations of the IT system endangering the security can be performed.
HRSL-4-4 HCSL-4-4	SL7	The security function for security logging shall ensure that all events registered in the security log can be presented in readable format.

KSF id	Req. id	Description
HSSL-4-4		The security function for security logging shall ensure that all events registered in the security log can be presented in readable format and that the inspection of the registered events can be performed.
HRSL-4-5 HCSL-4-5	SL8	The security function for security logging shall enable tool-based inspection of the events registered in the security log.
HSSL-4-5		The security function for security logging shall enable tool-based inspection of the events registered in the security log. The inspection shall be based on the possibility to sort and seek registered events.
HRSL-4-6 HCSL-4-6	SL9	The security function for security logging shall enable back up of the security log.
HSSL-4-6		The security function for security logging shall enable back up of the security log. Back ups should be based on printouts or copying to other storage medias.
HRSL-4-7 HCSL-4-7 HSSL-4-7	SL10	The security function for security logging shall ensure that no registered events are erased, overwritten, or in other ways destroyed as a consequence of flaws in the security function or the security log being full.
HSSL-5-1	SL11	The security function for security logging shall be able to maintain a defined secure state when events cannot be logged.
HSSL-5-2	SL12	The security function for security logging shall ensure that no user activity takes place in the IT system if the security log is inactive.

Table 18: Security requirements for intrusion prevention.

KSF id	Req. id	Description
HRIS-5-1 HCIS-5-1 HSIS-5-1	IP1	The security function for intrusion prevention shall, through self-inspection, perform controls of integrity: • at start-up, • when authorized administrators so calls for, and • when resuming ordinary operations from a secure state in order to demonstrate correct functionality of the underlying solution.

KSF id	Req. id	Description
HRIS-5-2 HCIS-5-2 HSIS-5-2	IP2	The security function for intrusion prevention shall be able to maintain a defined state of security when the entire or parts of the functionality, which restricts the information allowed to be transferred through the security function, is corrupt or inaccessible.
HRG-5-1 HCG-5-1 HSG-5-1	IP3	The security function shall, together with the other security functions in the IT system, maintain its own security domain which protects against manipulation and disturbances, both from subjects and users that belong to or do not belong to this domain.
HRG-5-2 HCG-5-2 HSG-5-2	IP4	The security function shall, together with the other security functions in the IT system, have the possibility to provide for reliable time.
HRG-5-3 HCG-5-3 HSG-5-3	IP5	The security function shall make sure that only authorized administrators can maintain the security function and handle its security settings.
HRIS-4-1 HCIS-4-1 HSIS-4-1	IP6	The security function for intrusion prevention shall hinder all access to the subjects and objects of the IT system for those subjects that do not have access rights to the IT system.
HRIS-4-2 HCIS-4-2 HSIS-4-4	IP7	The security function for intrusion prevention shall restrict the information allowed to be transferred through the security function by controlling both incoming and outgoing flows of information.
HRIS-4-3 HCIS-4-3 HSIS-4-5	IP8	The security function for intrusion prevention shall ensure that no information is transferred without using the configured filters of the security function.
HRIS-4-4 HCIS-4-7	IP9	The security function for intrusion prevention shall enable configurations allowing information to flow only in one direction through the security function.
HSIS-4-7		The security function for intrusion prevention shall be constructed so that the transferring of information through the security function takes place with separate interfaces for incoming and outgoing flows of information.
HSIS-4-8		The security function for intrusion prevention shall be constructed so that each interface ensures that information can only flow in one direction through the interface.
HRIS-4-5	IP10	The security function for intrusion prevention shall ensure that information classified as RESTRICTED is not transferred to other IT systems than those which can handle information classified as RESTRICTED or higher.

KSF id	Req. id	Description
HCIS-4-8		The security function for intrusion prevention shall ensure that information classified as CONFIDENTIAL is not transferred to other IT systems than those which can handle information classified as CONFIDENTIAL or higher.
HSIS-4-9		The security function for intrusion prevention shall ensure that information classified as SECRET is not transferred to other IT systems than those which can handle information classified as SECRET or higher.
HRIS-4-6 HSIS-5-8	IP11	The security function for intrusion prevention shall prevent not identified subjects from using, influencing, or in other ways manipulating the security function.
HCIS-4-6 HSIS-4-3		The security function for intrusion prevention shall in case of identification and authentication errors deny access to, and use of the security function.
HRIS-4-7 HCIS-4-9 HSIS-4-10	IP12	The security function for intrusion prevention shall ensure that no disallowed network traffic is transferred through the security function.
HRIS-4-8 HCIS-4-10 HSIS-4-11	IP13	The security function for intrusion prevention shall limit the information a user or a subject receives as response when denied access to the security function.
HCIS-4-4	IP14	The security function for intrusion prevention shall be constructed so that those filters used in the security function are equivalent to the application protocol level, and that there exist a filter for each protocol.
HSIS-4-6		The security function for intrusion prevention shall be constructed so that those filters that are used in the security function are equivalent to the application protocol level, and that there exist a filter for respective protocol. Further, additional restrictions shall be able to be performed at protocol level, e.g., only certain types of instructions can be transferred.
HCIS-4-5	IP15	The security function for intrusion prevention shall identify and authenticate subjects that transfer information through the security function when establishing such communication. The authentication shall involve two different security attributes.

KSF id	Req. id	Description
HSIS-4-2		The security function for intrusion prevention shall identify and authenticate those subjects that transfer information through the security function when establishing such communication The authentication shall involve three different security attributes of whom one is a cryptographic function.
HCIS-5-3 HSIS-5-3	IP16	Following an incorrect behavior of the security function or an interruption for maintenance, the security function for intrusion prevention shall resume at a defined secure state.
HSIS-5-6	IP17	The security function for intrusion prevention shall be able to provide defined and decided on administrative roles with the possibility to verify the correctness of the executable code of the security function.

Table 19: Security requirements for intrusion detection.

KSF id	Req. id	Description
HCG-5-1 HSG-5-1	ID1	The security function shall, together with the other security functions in the IT system, maintain its own security domain which protects against manipulation and disturbances, both from subjects and users that belong to or do not belong to this domain.
HCG-5-2 HSG-5-2	ID2	The security function shall, together with the other security functions in the IT system, have the possibility to provide for reliable time.
HCG-5-3 HSG-5-3	ID3	The security function shall make sure that only authorized administrators can maintain the security function and handle its security settings.
HCID-4-1 HSID-4-1	ID4	The security function for intrusion detection shall enable detection of already performed intrusions as well as ongoing intrusions.
HCID-4-2 HSID-4-2	ID5	The security function for intrusion detection shall, together with each separate registered event, also register time and date for the event as well as the identity of the user or subject.
HCID-4-3	ID6	The security function for intrusion detection shall ensure that all registered events can be presented in a form that is interpretable for authorized persons.

KSF id	Req. id	Description
HSID-4-3		The security function for intrusion detection shall ensure that all registered events can be presented in a form that is interpretable for authorized persons and that inspection of the registered events can be performed.
HCID-4-4 HSID-4-4	ID7	The security function for intrusion detection shall enable tool-based inspection of registered events. The inspection shall be based on the possibility to sort and seek registered events.
HCID-4-5 HSID-4-6	ID8	The security function for intrusion detection shall ensure that tracing of misuse as well as attempts to misuse that could endanger the security of the IT system can be performed.
HCID-4-6 HSID-4-8	ID9	The security function for intrusion detection shall ensure that no registered events are erased, overwritten or in other ways destroyed as a consequence of flaws in the security function or the event log being full.
HSID-4-5	ID10	The security function for intrusion detection shall, through automatic analysis, be able to conclude whether defined rules have been violated. The defined rules shall include such events that are known to represent misuse of or intrusion in IT systems.
HSID-4-7	ID11	The security function for intrusion detection shall ensure that registered events can be analyzed together with security relevant events registered by the security function for security logging.
HSID-5-1	ID12	Following an incorrect behavior of the security function or an interruption for maintenance, the security function for intrusion detection shall resume at a defined secure state.

Table 20: Security requirements for protection against malware.

KSF id	Req. id	Description
HRG-5-1 HCG-5-1 HSG-5-1	PM1	The security function shall, together with the other security functions in the IT system, maintain its own security domain which protects against manipulation and disturbances, both from subjects and users that belong to or do not belong to this domain.
HRG-5-2 HCG-5-2 HSG-5-2	PM2	The security function shall, together with the other security functions in the IT system, have the possibility to provide for reliable time.
HRG-5-3 HCG-5-3 HSG-5-3	PM3	The security function shall make sure that only authorized administrators can maintain the security function and handle its security settings.

KSF id	Req. id	Description
HRSK-5-4 HCSK-5-1 HSSK-5-1	PM4	The security function for protection against malware shall, through self-inspection, perform controls of integrity at start-up and when authorized administrators so calls for, in order to demonstrate correct functionality of the underlying solution.
HRSK-4-1 HCSK-4-1	PM5	The security function for protection against malware shall hinder all access to the resources of the IT system by objects containing malware.
HRSK-4-2 HCSK-4-2	PM6	The security function for protection against malware shall, through the control mechanism, ensure that no malware can: • change, • destroy, or • in other ways manipulate the objects in the IT systems protected by the security function.
HRSK-4-3 HCSK-4-4 HSSK-4-4	PM7	The security function for protection against malware shall ensure detection of malware by controlling both incoming and outgoing information flows.
HRSK-4-4 HCSK-4-5 HSSK-4-5	PM8	The security function for protection against malware shall ensure that no information is transferred to or from the IT system without the control mechanism of the security function being in use.
HRSK-4-5 HCSK-4-7 HSSK-4-7	PM9	The security function for protection against malware shall, if malware is detected, be able to automatically take measures. Such measures shall include the placement of infected subjects or objects in quarantine as well as warning authorized administrators and the affected user.
HRSK-4-6 HCSK-4-3	PM10	The security function for protection against malware shall use a definition file as control mechanism for the objects in the IT system protected by the security function.
HRSK-4-7 HCSK-4-8 HSSK-4-8	PM11	The security function for protection against malware shall perform controls of subjects and objects: • during operation, • at start-up, and • when authorized administrators call for it.
HRSK-4-8 HCSK-4-9 HSSK-4-9	PM12	The security function for protection against malware shall be able to automatically update the protection against malware in a secure manner.

KSF id	Req. id	Description
HCSK-4-6 HSSK-4-6	PM13	The security function for protection against malware shall, through automatic analysis, be able to detect potential malware. Such analysis shall include comparison to the definition file for the objects protected by the security function.
HSSK-4-3	PM14	The security function for protection against malware shall use two from each other independent control mechanisms for protection against malware for the objects in the IT system protected by the security function. The first control mechanism shall be control against the definition file and the second shall be configuration control.
HSSK-5-2	PM15	The security function for protection against malware shall only accept verified and validated objects for use as control mechanisms.
HSSK-5-3	PM16	The security function for protection against malware shall be able to maintain a defined secure state when the entire or parts of the functionality that detects malware, is corrupt, inaccessible, or out of date.

Appendix B – Security Profile Template Data

Nine security experts at FOI were asked to classify a set of security requirements regarding intrusion detection into the groups of fundamental and important security requirements. The results are presented in Table 21.

Table 21: Results from the classification of security requirements.

Req. id	Description	Fund.	Imp.
ID1	The security function shall, together with the other security functions in the IT system, maintain its own security domain which protects against manipulation and disturbances, both from subjects and users that belong to or do not belong to this domain.	7	2
ID2	The security function shall, together with the other security functions in the IT system, have the possibility to provide for reliable time.	1	8
ID3	The security function shall make sure that only authorized administrators can maintain the security function and handle its security settings.	8	1
ID4	The security function for intrusion detection shall enable detection of already performed intrusions as well as ongoing intrusions.	3	6
ID5	The security function for intrusion detection shall, together with each separate registered event, also register time and date for the event as well as the identity of the user or subject.	4	5
ID6	The security function for intrusion detection shall ensure that all registered events can be presented in a form that is interpretable for authorized persons and that inspection of the registered events can be performed.	6	3
ID7	The security function for intrusion detection shall enable tool-based inspection of registered events. The inspection shall be based on the possibility to sort and seek registered events.	1	8
ID8	The security function for intrusion detection shall ensure that tracing of misuse as well as attempts to misuse that could endanger the security of the IT system can be performed.	2	7

Req. id	Description	Fund.	Imp.
ID9	The security function for intrusion detection shall ensure that no registered events are erased, overwritten or in other ways destroyed as a consequence of flaws in the security function or the event log being full.	3	6
ID10	The security function for intrusion detection shall, through automatic analysis, be able to conclude whether defined rules have been violated. The defined rules shall include such events that are known to represent misuse of or intrusion in IT systems.	4	5
ID11	The security function for intrusion detection shall ensure that registered events can be analyzed together with security relevant events registered by the security function for security logging.	4	5
ID12	Following an incorrect behavior of the security function or an interruption for maintenance, the security function for intrusion detection shall resume at a defined secure state.	7	1

In the next step of calculating the security profile template, seven of the security experts were asked to do pair-wise prioritizations of the important security requirements regarding their relative importance for intrusion detection. To calculate the consistency ratio a random index of 1.40 was used (Saaty, 2004). The results are presented for each security expert below.

Security expert #1

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11	Eigenvector
ID2	**1.00**	0.20	3.00	5.00	0.33	0.20	0.20	0.20	0.05412162
ID4	5.00	**1.00**	5.00	7.00	1.00	3.00	3.00	5.00	0.27244935
ID5	0.33	0.20	**1.00**	5.00	0.20	0.20	0.20	3.00	0.05554643
ID7	0.20	0.14	0.20	**1.00**	0.20	0.14	0.20	0.33	0.02041516
ID8	3.00	1.00	5.00	5.00	**1.00**	0.33	1.00	3.00	0.14360530
ID9	5.00	0.33	5.00	7.00	3.00	**1.00**	5.00	5.00	0.25730182
ID10	5.00	0.33	5.00	5.00	1.00	0.20	**1.00**	3.00	0.13230863
ID11	5.00	0.20	0.33	3.00	0.33	0.20	0.33	**1.00**	0.06425169

The resulting λ_{max} is 8.83865523, which results in a CR of 0.085577064.

Security expert #2

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11	Eigenvector
ID2	**1.00**	0.20	3.00	3.00	0.33	0.20	0.33	3.00	0.07637572
ID4	5.00	**1.00**	5.00	7.00	3.00	1.00	1.00	3.00	0.23640149
ID5	0.33	0.20	**1.00**	5.00	0.33	0.20	0.33	1.00	0.05212524
ID7	0.33	0.14	0.20	**1.00**	0.20	0.14	0.20	0.33	0.02418577
ID8	3.00	0.33	3.00	5.00	**1.00**	0.33	1.00	3.00	0.12972791
ID9	5.00	1.00	5.00	7.00	3.00	**1.00**	3.00	5.00	0.28284958
ID10	3.00	1.00	3.00	5.00	1.00	0.33	**1.00**	3.00	0.14818227
ID11	0.33	0.33	1.00	3.00	0.33	0.20	0.33	**1.00**	0.05015202

The resulting λ_{max} is 8.540040748, which results in a CR of 0.055106199.

Security expert #3

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11	Eigenvector
ID2	**1.00**	0.50	4.00	0.50	1.00	0.33	1.00	2.00	0.11886305
ID4	2.00	**1.00**	2.00	1.00	2.00	0.33	0.50	2.00	0.12663438
ID5	0.25	0.50	**1.00**	0.50	3.00	0.50	1.00	3.00	0.10288044
ID7	2.00	1.00	2.00	**1.00**	3.00	3.00	1.00	3.00	0.19293375
ID8	1.00	0.50	0.33	0.33	**1.00**	2.00	0.50	1.00	0.08602272
ID9	3.00	3.00	2.00	0.33	0.50	**1.00**	0.33	2.00	0.14634867
ID10	1.00	2.00	1.00	1.00	2.00	3.00	**1.00**	3.00	0.17343511
ID11	0.50	0.50	0.33	0.33	1.00	0.50	0.33	**1.00**	0.05288189

The resulting λ_{max} is 9.289458362, which results in a CR of 0.131577384.

Security expert #4

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11	Eigenvector
ID2	**1.00**	0.33	0.14	0.20	3.00	0.20	0.14	0.20	0.03219647
ID4	3.00	**1.00**	0.20	1.00	3.00	0.33	0.14	0.33	0.05719631
ID5	7.00	5.00	**1.00**	5.00	7.00	3.00	1.00	3.00	0.28742586
ID7	5.00	1.00	0.20	**1.00**	3.00	0.33	0.20	0.33	0.06632414
ID8	0.33	0.33	0.14	0.33	**1.00**	0.20	0.14	0.20	0.02485662
ID9	5.00	3.00	0.33	3.00	5.00	**1.00**	1.00	1.00	0.15114770
ID10	7.00	7.00	1.00	5.00	7.00	1.00	**1.00**	1.00	0.22970520
ID11	5.00	3.00	0.33	3.00	5.00	1.00	1.00	**1.00**	0.15114770

The resulting λ_{max} is 8.492594079, which results in a CR of 0.050264702.

Security expert #5

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11	Eigenvector
ID2	**1.00**	0.14	0.20	0.33	0.33	0.20	3.00	0.33	0.03953189
ID4	7.00	**1.00**	3.00	5.00	1.00	3.00	5.00	5.00	0.27472469
ID5	5.00	0.33	**1.00**	0.33	0.14	0.20	1.00	0.20	0.05361184
ID7	3.00	0.20	3.00	**1.00**	0.20	0.33	3.00	1.00	0.07796354
ID8	3.00	1.00	7.00	5.00	**1.00**	3.00	5.00	3.00	0.25893413
ID9	5.00	0.33	5.00	3.00	0.33	**1.00**	5.00	3.00	0.16393436
ID10	0.33	0.20	1.00	0.33	0.20	0.20	**1.00**	0.20	0.03102417
ID11	3.00	0.20	5.00	1.00	0.33	0.33	5.00	**1.00**	0.10027539

The resulting λ_{max} is 9.134003055, which results in a CR of 0.115714597.

Security expert #6

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11	Eigenvector
ID2	**1.00**	1.00	1.00	0.11	0.11	0.11	0.11	0.11	0.02775270
ID4	1.00	**1.00**	1.00	1.00	1.00	0.33	1.00	1.00	0.07474926
ID5	1.00	1.00	**1.00**	1.00	1.00	0.11	1.00	1.00	0.06395837
ID7	9.00	1.00	1.00	**1.00**	3.00	0.33	1.00	1.00	0.11556264
ID8	9.00	1.00	1.00	0.33	**1.00**	0.11	1.00	1.00	0.07948889
ID9	9.00	3.00	9.00	3.00	9.00	**1.00**	9.00	9.00	0.45330587
ID10	9.00	1.00	1.00	1.00	1.00	0.11	**1.00**	3.00	0.10493437
ID11	9.00	1.00	1.00	1.00	1.00	0.11	0.33	**1.00**	0.08024790

The resulting λ_{max} is 9.33515756, which results in a CR of 0.136240567.

Security expert #7

	ID2	ID4	ID5	ID7	ID8	ID9	ID10	ID11	Eigenvector
ID2	**1.00**	3.00	3.00	0.20	1.00	0.20	0.33	0.33	0.06988432
ID4	0.33	**1.00**	3.00	0.20	0.20	0.20	0.33	0.20	0.03835145
ID5	0.33	0.33	**1.00**	0.33	0.33	0.33	0.33	0.33	0.03802031
ID7	5.00	5.00	3.00	**1.00**	3.00	3.00	3.00	3.00	0.28130986
ID8	1.00	5.00	3.00	0.33	**1.00**	3.00	3.00	3.00	0.19125345
ID9	5.00	5.00	3.00	0.33	0.33	**1.00**	0.33	0.33	0.10719688
ID10	3.00	3.00	3.00	0.33	0.33	3.00	**1.00**	0.33	0.11511120
ID11	3.00	5.00	3.00	0.33	0.33	3.00	3.00	**1.00**	0.15887252

The resulting λ_{max} is 9.433909563, which results in a CR of 0.146317302.

Appendix C – Filter Profile Template Data

A group of seven security experts at FOI was asked to do pair-wise prioritizations of the filter functional requirements regarding their relative importance for protection against intrusions. To calculate the consistency ratio a random index of 1.11 was used (Saaty, 2004). The results are presented for each security expert below.

Security expert #1

	FF1	FF2	FF3	FF4	FF5	Eigenvector
FF1	**1.00**	3.00	5.00	5.00	3.00	0.43624387
FF2	0.33	**1.00**	3.00	3.00	3.00	0.24432562
FF3	0.20	0.33	**1.00**	0.33	0.33	0.05672748
FF4	0.20	0.33	3.00	**1.00**	0.20	0.08297309
FF5	0.33	0.33	3.00	5.00	**1.00**	0.17972994

The resulting λ_{max} is 5.517357452, which results in a CR of 0.116521949.

Security expert #2

	FF1	FF2	FF3	FF4	FF5	Eigenvector
FF1	**1.00**	3.00	5.00	3.00	3.00	0.42185435
FF2	0.33	**1.00**	5.00	3.00	3.00	0.26891255
FF3	0.20	0.20	**1.00**	0.33	0.33	0.05144932
FF4	0.33	0.33	3.00	**1.00**	0.33	0.10035410
FF5	0.33	0.33	3.00	3.00	**1.00**	0.15742968

The resulting λ_{max} is 5.355379053, which results in a CR of 0.080040327.

Security expert #3

	FF1	FF2	FF3	FF4	FF5	Eigenvector
FF1	**1.00**	0.33	0.50	1.00	0.33	0.09554934
FF2	3.00	**1.00**	2.00	3.00	0.50	0.27703497
FF3	2.00	0.50	**1.00**	3.00	1.00	0.21992337
FF4	1.00	0.33	0.33	**1.00**	0.33	0.08844028
FF5	3.00	2.00	1.00	3.00	**1.00**	0.31905204

The resulting λ_{max} is 5.155943287, which results in a CR of 0.035122362.

Security expert #4

	FF1	FF2	FF3	FF4	FF5	Eigenvector
FF1	**1.00**	3.00	5.00	7.00	7.00	0.54099374
FF2	0.33	**1.00**	3.00	5.00	1.00	0.17983441
FF3	0.20	0.33	**1.00**	3.00	0.33	0.07895714
FF4	0.14	0.20	0.33	**1.00**	0.20	0.03975870
FF5	0.14	1.00	3.00	5.00	**1.00**	0.16045602

The resulting λ_{max} is 5.317595167, which results in a CR of 0.071530443.

Security expert #5

	FF1	FF2	FF3	FF4	FF5	Eigenvector
FF1	**1.00**	0.20	0.33	3.00	1.00	0.10188673
FF2	5.00	**1.00**	3.00	5.00	7.00	0.51005210
FF3	3.00	0.33	**1.00**	5.00	3.00	0.24070564
FF4	0.33	0.20	0.20	**1.00**	0.33	0.05103443
FF5	1.00	0.14	0.33	3.00	**1.00**	0.09632111

The resulting λ_{max} is 5.236764074, which results in a CR of 0.053325242.

Security expert #6

	FF1	FF2	FF3	FF4	FF5	Eigenvector
FF1	**1.00**	0.11	0.11	5.00	3.00	0.10420067
FF2	9.00	**1.00**	1.00	9.00	7.00	0.44006347
FF3	9.00	1.00	**1.00**	3.00	3.00	0.35082887
FF4	0.20	0.11	0.33	**1.00**	1.00	0.05006158
FF5	0.33	0.14	0.33	1.00	**1.00**	0.05484542

The resulting λ_{max} is 5.824547286, which results in a CR of 0.185708848.

Security expert #7

	FF1	FF2	FF3	FF4	FF5	Eigenvector
FF1	**1.00**	0.33	0.20	0.33	0.33	0.06226527
FF2	3.00	**1.00**	0.33	0.33	0.33	0.10803614
FF3	5.00	3.00	**1.00**	3.00	1.00	0.34144010
FF4	3.00	3.00	0.33	**1.00**	0.33	0.17036380
FF5	3.00	3.00	1.00	3.00	**1.00**	0.31789469

The resulting λ_{max} is 5.288953999, which results in a CR of 0.06507973.