

Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Evaluation of the Security of Components in Distributed Information Systems

Examensarbete utfört i Informationsteori
av

Richard Andersson

LiTH-ISY-EX-3430-2003

Linköping 2003



TEKNISKA HÖGSKOLAN
LINKÖPINGS UNIVERSITET

Department of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden

Linköpings tekniska högskola
Institutionen för systemteknik
581 83 Linköping

www.FirstRanker.com

Evaluation of the Security of Components in Distributed Information Systems

Examensarbete utfört i Informationsteori
vid Linköpings tekniska högskola
av

Richard Andersson


LiTH-ISY-EX-3430-2003

Handledare: Jonas Hallberg, Amund Hunstad

Examinator: Viiveke Fåk

Linköping 2003-12-12

www.FirstRanker.com

 LINKÖPINGS UNIVERSITET	Avdelning, Institution Division, Department Institutionen för systemteknik 581 83 LINKÖPING	Datum Date 2003-12-12
--	---	------------------------------------

Språk Language Svenska/Swedish X Engelska/English	Rapporttyp Report category Licentiatavhandling X Examensarbete C-uppsats D-uppsats Övrig rapport _____	ISBN ISRN LITH-ISY-EX-3430-2003 Serietitel och serienummer ISSN Title of series, numbering _____
URL för elektronisk version http://www.ep.liu.se/exjobb/isy/2003/3430/		

Titel Title Författare Richard Andersson Author	Värdering av komponenters säkerhet i distribuerade informations system Evaluation of the Security of Components in Distributed Information Systems
--	---

Sammanfattning Abstract This thesis suggests a security evaluation framework for distributed information systems, responsible for generating a system modelling technique and an evaluation method. The framework is flexible and divides the problem space into smaller, more accomplishable subtasks with the means to focus on specific problems, aspects or system scopes. The information system is modelled by dividing it into increasingly smaller parts, evaluate the separate parts and then build up the system "bottom up" by combining the components. Evaluated components are stored as reusable instances in a component library. The evaluation method is focusing on technological components and is based on the Security Functional Requirements (SFR) of the Common Criteria. The method consists of the following steps: (1) define several security values with different aspects, to get variable evaluations (2) change and establish the set of SFR to fit the thesis, (3) interpret evaluated security functions, and possibly translate them to CIA or PDR, (4) map characteristics from system components to SFR and (5) combine evaluated components into an evaluated subsystem. An ontology is used to, in a versatile and dynamic way, structure the taxonomy and relations of the system components, the security functions, the security values and the risk handling. It is also a step towards defining a common terminology for IT security.

Nyckelord Keyword Security Evaluation, Modelling, Distributed Information Systems, Common Criteria, Ontology

www.FirstRanker.com

Abstract

This thesis suggests a security evaluation framework for distributed information systems, responsible for generating a system modelling technique and an evaluation method. The framework is flexible and divides the problem space into smaller, more accomplishable subtasks with the means to focus on specific problems, aspects or system scopes.

The information system is modelled by dividing it into increasingly smaller parts, evaluate the separate parts and then build up the system “bottom up” by combining the components. Evaluated components are stored as reusable instances in a component library.

The evaluation method is focusing on technological components and is based on the Security Functional Requirements (SFR) of the Common Criteria. The method consists of the following steps: (1) define several security values with different aspects, to get variable evaluations (2) change and establish the set of SFR to fit the thesis, (3) interpret evaluated security functions, and possibly translate them to CIA or PDR, (4) map characteristics from system components to SFR and (5) combine evaluated components into an evaluated subsystem.

An ontology is used to, in a versatile and dynamic way, structure the taxonomy and relations of the system components, the security functions, the security values and the risk handling. It is also a step towards defining a common terminology for IT security.

www.FirstRanker.com

www.FirstRanker.com

Table of contents

1. Introduction.....	1
1.1. Motivation	1
1.2. Problem Formulation	1
1.3. Contribution.....	2
1.4. Disposition	2
2. Background	3
2.1. IT Security.....	3
2.2. Important terms and definitions.....	5
2.3. Related Work.....	6
3. Preliminary approaches.....	17
3.1. From tree structure to ontology.....	17
3.2. Locality model and dependency algorithm.....	19
3.3. CC and component structure	23
3.4. Initial framework.....	23
4. Security Evaluation Framework	25
4.1. Modularity.....	27
4.2. Scope.....	28
4.3. Component/System Structure	29
4.4. Method of evaluation.....	32
4.5. Modelling and Implementation	33
4.6. Ontology.....	34
5. Evaluation method.....	39
5.1. Security values and metric	39
5.2. CC Security Functional Requirements	41
5.3. Security Evaluation of CC SFs.....	48
5.4. Map CC Security Functions to evaluated component characteristics	52
5.5. Evaluation of Systems made up of Components	54
6. Discussion	60
6.1. Security Evaluation Framework	60
6.2. Evaluation method.....	61
7. Conclusions.....	63
7.1. Summary.....	63
7.2. Future Work.....	64
7.3. Acknowledgements.....	66

References.....	67
Glossary	70
Appendix.....	74

List of Figures

Figure 1: The characteristics formed as a tree-structure.....	8
Figure 2: Sample class decomposition diagram.....	11
Figure 3: Sampled Securability	14
Figure 4: Restructuring of some elements from the characteristic tree	18
Figure 5: Example of a restricted building segmented in compartments	20
Figure 6: Graphs for physical segmentation and net access	20
Figure 7: The dependency algorithm.	22
Figure 8: Initial security evaluation framework.....	24
Figure 9: Overview of security evaluation framework.....	25
Figure 10: Security evaluation framework model.	26
Figure 11: Example of a component structure for a distributed information system.	30
Figure 12: Modelling and implementation	34
Figure 13: Ontology of Security Evaluation.	37
Figure 14: Security functions in a TOE of a system	42
Figure 15: Security functions in a TOE consisting of a distributed system.....	42
Figure 16: Simple node structure.....	55
Figure 17: Poisson distribution.....	65
Figure 18: SFR coloured according to CIA.....	78
Figure 19: SFR coloured according to PDR.....	79

List of Tables

Table 1: Security values for components and families of the FTP-class	52
Table 2: Lists all components of the FPT-class, in which papers they were found to be relevant for the smartcard-component and their estimated security values	54
Table 3: Security values for components and families of the FTP-class. The values are the security estimates from the smart card, the card reader and the resulting values from the combination of the two	59
Table 4: Revised CC structure including categorization of class, family or component.....	77

1. Introduction

“When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you can not measure it, when you can not express it in numbers, your knowledge is of a meagre and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to a stage of science.”

This quote comes from Lord Kelvin in 1891 [1], and tells us about what he would have thought was needed in IT security, had he been alive today; namely a security value. In order to reach this value, a security metric has to be decided on and some kind of analysis or evaluation of the computer systems has to take place.

1.1. Motivation

IT security is steadily growing as a scientific field and needs therefore to mature. With its vast complexity and many different perspectives, it is a complex assignment for security scientists of different areas of expertise to cooperate and agree on methods, procedures and results. It is hard to identify the actual problems and unite on a common goal. This leads to that neither a security value, nor the evaluation scheme and metric to estimate this value have yet been agreed on among the scientists.

The development of more and more extensive information systems makes IT security an increasingly important factor. Most systems will very likely be exposed to hard tests regarding their ability to maintain the desired security level. That is why it is extremely important that this ability may be evaluated in the growing complexity of information systems. For this purpose, new methods and tools have to be developed.

1.2. Problem Formulation

This work aims at finding a method to evaluate the security of distributed information systems.

Since the evaluation process is such a vast and complex task, the first thing to consider is how the information systems and all other security-relevant issues should be modelled in order to enable a meaningful evaluation. The model has to handle distributed, extensive and dynamical systems, which put high demands on the model structure.

Ways to tackle the evaluation process need to be decided on. All aspects from finding a suitable metric and relevant system characteristics, to the translation of characteristics into a security value must be covered. The final goal of the evaluation process is to be able to estimate the security values of an information system with high accuracy. However, the evaluation method will probably not say with certainty that the system with security value x is better than the system with value y , slightly less than x . However, it might help in finding the vulnerabilities in a system. Moreover, the process for reaching the security measurement of a distributed information system might be more important than the measure itself.

1.3. Contribution

In order to handle the complex modelling of distributed information systems, a framework structure was developed, in which the evaluators can create and customize the evaluation process according to their needs. Using modularity concepts and a component library in addition to the framework, flexibility and dynamic abilities were introduced. In this way, prioritizing in different aspects of the evaluated system can be done, without losing the possibility to cover all aspects of the security evaluation, not only the most intuitive ones.

In order to get a relevant foundation for the evaluation, the security functional requirements of the Common Criteria were used, since they were regarded to have the most developed security functions of today. They enhance the different security mechanisms in systems that are needed in order to receive a good enough security.

The meaning of the security values and the metric that leads to them is analyzed. This is a necessary step in the process of comparing different systems. For the possibility of combining already evaluated components into subsystems, some mathematical functions were proposed as a starting point for more standardized methods.

Also, an ontology of the entire framework was introduced, as it gives more precise and foreseeable means to structure the distributed information system, its attributes and all its relations to the model and the evaluation.

The evaluation method was separated into five different steps based on the security functions and the component structure. Although the framework can handle all aspects of the information system, the evaluation method concentrates on the technological components as this is the first step in the process of evaluation.

1.4. Disposition

The second chapter explains IT security, defines some important terms and describes background information and important texts that were used in the research.

In the third chapter, the work process is described by explaining initial thoughts and ideas that lead to the current contents of the thesis.

The fourth chapter explains the security evaluation framework and the different parts of it.

The evaluation method of the framework is then described in the fifth chapter with some examples to help the understanding of the different steps of the method.

In the sixth chapter, the security evaluation framework and the evaluation method are discussed.

Finally in the seventh chapter, a summary of the thesis is given as well as some proposals for future work that might further improve the work presented in the thesis.

2. Background

This chapter covers the background information needed for this thesis. First, the concept of IT security is explained. Then some terms are defined that are considered important for this report. Lastly, related work is presented, categorized by the different methods of security evaluation that they use.

2.1. IT Security

IT security is about the protection of information assets and the services delivered by information systems. It suggests that risk analysis should be made to investigate the threats, and what protective measures should be undertaken in order to shield the assets. It could also be more specifically defined as the *“prevention and detection of unauthorised actions by users of a computer system”* [2].

One common way to describe IT security is to partition according to the ways information assets can be compromised. This categorization is often referred to as *“CIA”* [2].

- **Confidentiality:** prevention of illegal revelation of information.
- **Integrity:** prevention of illegal modification of information.
- **Availability:** prevention of illegal withholding of information or resources.

Arguments may be made that the list above is incomplete. Some would like to add authenticity to cover the aspects of confirming the identity of the user or entity on the other end of a communication line, or to validate origin and correctness of data. Others may think accountability is more important to establish responsibility in applications, like in electronic commerce.

Another way to categorize IT security is a rough classification of aspired abilities, sometimes abbreviated as *“PDR”* [2].

- **Prevention:** prevent your assets from being damaged.
- **Detection:** detect attempts to violate the security of a system.
- **Reaction:** block or minimize the damage caused by security violations.

Survival may also be introduced here to describe the ability of systems to survive and recover from failures and security breaches.

Another term often used in IT security is dependability. It unifies the concepts of security, reliability, integrity and availability, and acts as a system property that places justifiable reliance on the computer system and the services it delivers.

More on these fundamental aspects of IT security can be read in [2].

The current trend is that IT security problems steadily increase over time. If this trend continues, and there are no indications of the opposite, IT security will become more and more important. The reason for the growing problems with viruses, worms and trojans could, according to [3], mainly be traced to the following three circumstances.

- The Internet is growing at a rapid pace, and therefore the connectivity of the computers is increasing at the same rate. This makes the computers increasingly more vulnerable, since they are reachable by more and more attacks launched by hackers. Computers are also becoming more and more dependant on the Internet, as increasingly more functionality from distributed computers is needed.
- The philosophy in today's system is to make them extensible, easy to upgrade and evolve. The developers save a lot of money on this, since they will not have to perform quite as thorough testing; it will be possible to update to a corrected version later. Also the consumers gain a lot from extensible systems, because the systems will last longer as new functionality can be added later. However, it is very hard to keep malicious code or new vulnerabilities away from the extensible system.
- The size and complexity of the systems today are quickly growing. This is very troublesome, since security holes and vulnerabilities are in relation to size and complexity. It is especially apparent for operative systems, where for example MS Windows increases from 15 millions lines of code in Windows 95, dated 1997, to 40 millions in Windows XP from 2002. A guess is that the vulnerabilities in the system increase at least with the same rate as the number of lines of code. The percentage rise of new vulnerabilities will probably be much larger than the percentage rise of the number of lines of codes, since there will be more programmers working with the code.

Research in computer security was for long relatively insignificant compared to areas dealing with technology, performance and products with a greater functional significance. Later on, much effort has been put into researching computer security, as corporations, authorities and other groups have begun to realise the growing cost that products with poor security and assurance result in. However, both the handling of and the attitude towards IT security has to be changed in many ways. For example, the common fault to have security added as a final functionality into existing systems, instead of considering it during the entire development process. Another fault is the flawed over reliance on cryptography; which can not solve all problems in IT security.

Additionally, computer security tends to be discussed on either a high level of abstraction or on the concrete system component level with little or no abstraction. A way of closing the gap between specifications and implementation has to be found [4].

For more and extended information regarding this, study [3, 5].

2.2. Important terms and definitions

The following definitions are considered central for the thesis. Some of them were defined for the purpose of this thesis, while other terms already did exist but have been defined here as well to clarify the exact meaning.

Distributed Information System

A Distributed Information System (DIS) is used to emphasize the distribution of information in the system and the fact that users and organizations are considered to be part of the system [6]. The two main architectures for distributed systems are client-server and peer-to-peer. Often transparency is wanted in a system, to hide the fact that it is divided into smaller subsystems.

Ontology

A way to formalize a specific conceptualization of a problem area, i.e. to share a common understanding of a domain. All objects in the ontology, called concepts, may contain slots of attributes and relations to other concepts. Instances for concepts may be created to change the ontology into a knowledge base. More information about ontologies is found in 4.6.

Risk Level

The security value for a system in use that has been put into its contextual environment, i.e. threats and assets are included in the model. After reaching this value, risk management could be performed on the model.

Securability

Securability is a term used to point out that because of technical vulnerabilities and the human factor, there is nothing like complete security in the case of IT systems. The design of IT products should therefore strive to be “*secure enough*” – designed for securability [7]. The goal of securability is that systems can be secured to an aspired level during operation [6].

This term is in the thesis used to describe the security value for a physical system that is not in use but has its organizational and individual aspects included in the model.

Security Indicators

Security Indicators (SI) is a term used in the text to exemplify unspecific security values that somehow estimates the security present in a system.

Security Level

The security value for a system that is in use and, thus, have its operational aspects included in the model.

Security Metric

Security metrics are the methods measuring security and specifying, if not the meaning of the security value, at least the different values that can be assigned.

Security Value

Security value is a common term used to denote one of the three more specific terms; securability, security level and risk level.

2.3.Related Work

Here follows background information about system modelling, security evaluation and other topics that were found to be useful. Those texts that have special relevance for the thesis are also mentioned.

2.3.1. System modelling approaches

When modelling computer systems, in order to evaluate them, there is a need to describe the mechanisms, components and how they interact with other components. There are several different ways that a system could be modelled. These methods of system modelling can be partitioned into the five following categories [7].

- **Policy modelling**
Implements models of security policy.
- **Attack modelling**
Categorizes attacks, and where in the system they could be stopped. The attacks are being analyzed, but not the system themselves.
- **Structural modelling**
Tries to model the information systems and their structure as a whole. The model will often become quite abstract.
- **Layer-based modelling**
Information systems are made up of components that could easily be connected with each other.
- **Security indicators modelling**
The complexity of designing and choosing security characteristics is reflected by the past attempts in this category only being partially successful.

These system modelling techniques were investigated in order to find the model to use for the security evaluation. Structural modelling was chosen in this work, although attack modelling was deemed interesting for specific approaches to the evaluation.

2.3.2. Identification of Security Relevant Characteristics in Distributed Information Systems

This is the title of the master thesis report [8] that constitutes the foundation for the work presented in this thesis. It suggests a set of characteristics to be used as a first step towards finding a technique for modelling, building and evaluation of distributed

information systems. It contributes with the following three fundamental steps as a necessary action towards finding measurable characteristics in a DIS.

- A definition of basic physical system components.
- A set of security relevant system characteristics for the components.
- Categorization of the system characteristics into a structure.

These three steps form the tree-structure shown in Figure 1. The structure commences with Confidentiality, Integrity and Availability (CIA) as root nodes. The children nodes are anything from security methods and policies to security products and the leaves are attributes that can be assigned a security value. The evaluation will then be performed by traversing the values from the leaves upwards in the tree, yielding new security values in each step.

For the purpose of this thesis, the CIA-tree has been regarded a good starting-point, but with structural flaws that are very hard to circumvent. The problem is that when mixing all these nodes of different kinds into the same tree-structure, the intent and purpose of the tree will become confused. There is no possibility in the tree to distinguish nodes that signify security objectives and functions from nodes that denote system components and their characteristics. Relations between the nodes will be very hard to model, and there is no way to tell the different relations apart, because the only relations that can exist are “children to” or “parent of”. There will also arise many doubles of the nodes since many components help to attend to several different kinds of protections (e.g. cryptology is used for accountability, protection against interception, non-repudiation and authentication). A more stringent and dynamic structure is needed. This is where an ontology is desired, instead of the advocated tree-structure.

Evaluation of the Security of Components in Distributed Information Systems

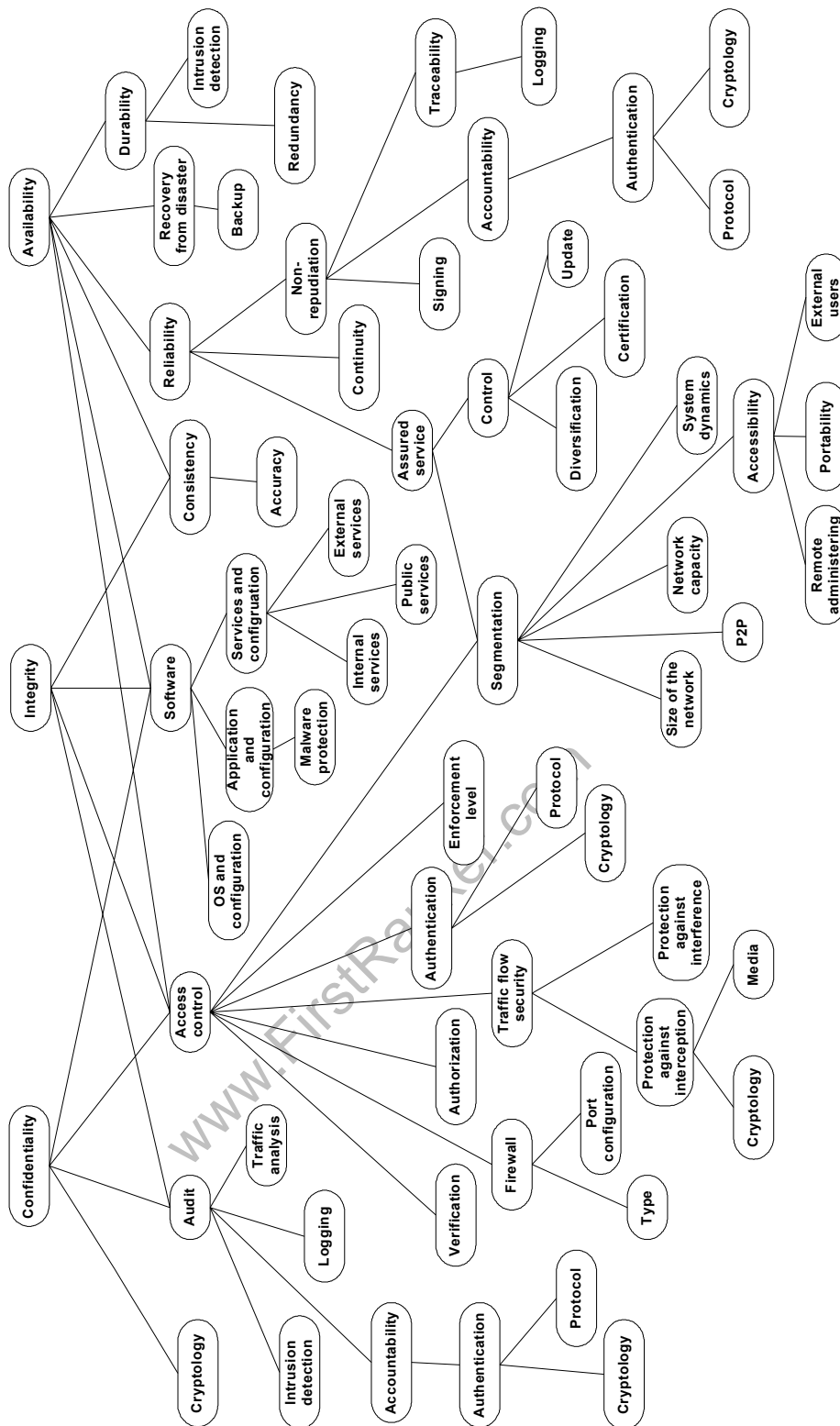


Figure 1: The characteristics formed as a tree-structure [8].

2.3.3. Certification methods

The use of certified products and systems provides a high-level of confidence that the claims being made about security functionality have been independently verified and tested.

A large variety of different certification methods exist; most of them have different usage and concerns. A distinction can be made between technical and organizational certifications.

Technical certifications

Trusted Computer System Evaluation Criteria (TCSEC), often referred to as the orange book, was started in 1967 in the U. S. A. [2]. It was developed mainly to provide a metric to evaluate a degree of trust for computer systems, guidance to manufactures and a basis for specifying security requirements.

In 1990, France, Germany, the Netherlands and the United Kingdom published the Information Technology Security Evaluation Criteria (ITSEC) based on existing work in their respective countries [2]. ITSEC is a structured set of criteria for evaluating computer security within products and systems.

These both certification standards lead to the joint effort of the Common Criteria for Information Technology Security Evaluation (CC), which will be explained at greater detail later.

Organizational certifications

The most recognized organizational certification is ISO/IEC 17799 (formerly the British Standard BS7799, published 1995). The standard identifies a number of “*critical success factors*” that an organization must achieve if it is to be successful implementing information security. It addresses most of the physical, procedural, personnel and management issues not addressed by the certification methods concentrating on technological aspects [9].

There are also other certifications for individual and organizational evaluations, like the Information Security Awareness Certification from Information Technology Association of America [10]. Here individuals make web-based tests and have to pass the tests with a minimum score in order to receive the personal certification. Furthermore, 90% of the staff of an organization must pass the individual test in order for the organization to receive a certification.

Eight different topics are covered:

- computer “*best practices*”
- computer ethics & misuse
- internet “*best practices*”

- malicious software
- passwords
- physical security
- sensitive information
- social engineering.

2.3.4. Common Criteria

The Common Criteria for Information Technology Security Evaluation (CC) was introduced in 1993 and represents the outcome of international efforts to align and develop the existing European (ITSEC) and North American (TCSEC) criteria towards a common standard for carrying out security evaluations. By establishing a common base, the results of an IT security evaluation are more meaningful to a wider community.

CC has a catalogue of standard Security Functional Requirements (SFR) which holds a set of functional components used to express functional requirements of products and systems. CC also has a catalogue of Standard Assessment Requirements (SAR) that is applied to verify that the functional capabilities are implemented correctly. The Security Functional Requirements can be used to develop a Protection Profile (PP) and as a means for developing a Security Target (ST). A PP specifies a profile of the implementation-independent requirements for a class of products or systems that meet specific customer needs. An ST specifies the implementation-dependent "*as-to-be-built*" or "*as-built*" requirements that are to be used as a basis for a particular product or system.

An IT product that is the subject of an evaluation is called the Target of Evaluation (TOE). The security of the TOE is controlled by the TOE Security Functions (TSF), which can be compared to the concept of Trusted Computing Base (TCB), a more common definition in the world of computer security [11]. The Security Functions that the TSF consist of are later referred to as the SFs.

A CC evaluation is carried out against a set of predefined assurance levels, called the Evaluation Assurance Levels (EAL0 to EAL7). This scale represents the ascending levels of confidence that can be placed in the TOEs security functions. It covers more the system development process than the system itself.

The SFR of CC is divided into eleven different classes. Each class contains several families, which each consists of one or more components. The components are also made up of one or several elements. An element is a specific description of a single security task. This structure can be seen in Figure 2 where the class contains three families. Each family contains several numbered components.

The components are hierarchically grouped, which can be interpreted as that the component which has the lower hierarchical order is a subset of the component with a higher order. For example, in Family 1 (Figure 2), the first component is a subset of the

second, and both the first and second components are subsets of the third. Components may also depend on other components, i.e. some components do not function properly if the component they are depending on is neglected.

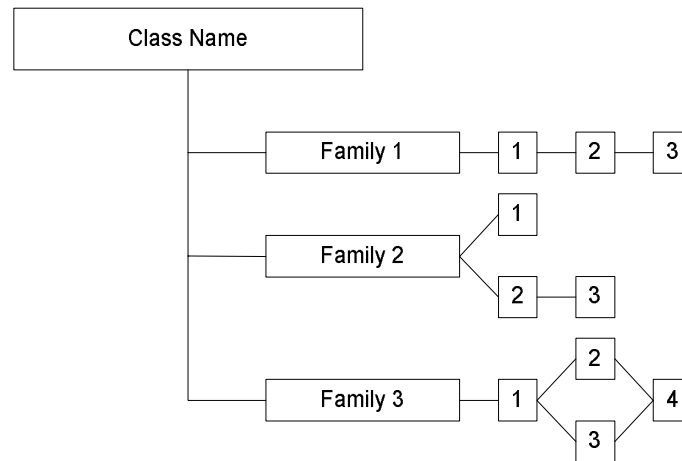


Figure 2: Sample class decomposition diagram [12].

More information on CC and its SFR can be found in [12, 13, 14, 15].

Advantages of CC

CC is one of the most commonly used security evaluation standards of today. Many researchers and scientists have spent lots of time developing it, making functions to cover the most important aspects of computer security. They are still making improvements and new ideas are adopted as computer security evolves. Although the purpose of CC differ from the purpose of this thesis report, the completeness and usefulness of the security functions still makes them an ideal choice as a bedrock for the evaluation of technological components.

The major beneficial functionality of the whole Common Criteria plan is that those who write Protection Profiles, often done with the interests of the customers in mind, will be able to drive the market. Thus the information security can be seen as a market driven industry. The role of CC is that of a meta-standard, providing a framework for spawning more specific standards. This reasoning about what drives the development of CC is explained at greater detail in [16, 17].

Disadvantages of CC

Here follows a list of some of the most serious drawbacks of CC. There are also some notes about how the drawbacks can be circumvented, prevented or reduced in this thesis, and references to where it can be found.

- CC is an evaluation of design methods, not an evaluation of security functionality. It is the system development process that is being evaluated, not the system itself. This means that the given EAL only states whether a large enough pile of paperwork over the design process exists or not. The correctness and importance of those papers does not even have to be verified and examined.

This is not the case for the approach presented in this thesis, as the SFR of CC is being used as a base for the system evaluation, not the evaluation process (see 5.2). Also security values are used rather than the EAL (see 5.1).

- There is a strong emphasis on the “*all or nothing*” nature of an evaluation. A product either meets the profile or it does not. The lack of official feedback to the profile writers leaves them guessing as to what requirements to relax or remove [16].

The evaluation of this thesis accomplishes the opposite, as it is the feedback of where the vulnerabilities of the system might appear that is one of the main tasks of the entire evaluation process (as explained in 5.1 and 5.4).

- Another complication is that even a slight change of the configuration renders the evaluation completely unusable.

As the evaluation process in this thesis combines components into subsystems, an idea for a solution to the above problem is to re-evaluate the changed component and then recombining the components (see 5.5).

- CC assumes a static set of threats for the environment. This means that the number of threats and attacks that will endanger the component are presumed and the evaluation is performed under the influence of this presumption. This environmental assumption, as observed in [18], does not coincide with the usual view; that computer security deals with the worst case scenarios when dealing with risk analysis (while the rest of computer science deals with the average case). If a non-hostile environment is assumed, and the evaluated product instead ends up in a hostile environment, the evaluation becomes useless [19].

This problem is, in this thesis, handled in the framework model as the environment is a module in the system model that can be either ignored or taken into consideration depending on the intent and resources of the evaluator (see 4).

- Questions about the objectivity of the evaluators may arise, since it is up to their judgment to rule whether a product is to pass or fail a CC evaluation.

The evaluation method described in this thesis is not intended to be secret as is the case in CC.

2.3.5. Work using CC as a foundation

The following work has been founded on the Common Criteria.

A method for designing secure solutions

Objections may arise to whether CC is inconvenient for the evaluation of large IT-systems or not [20]. Arguments are being made that the CC is more a standard of

evaluation of security functionality focusing on *products*, thus giving them limitations in describing end-to-end security since their use in complex IT solutions is not intuitive. To be able to use CC as a tool for designing secure solutions, additional work has to be done.

- A system model that is representative of the functional aspects of security within complex solutions.
- A systematic approach for creating security architectures based on the Common Criteria requirements taxonomy and the corresponding security system model.

The SFR of CC is then categorized into five strictly operational categories instead of the original eleven:

- credentials/identity
- audit
- integrity
- access control
- flow control.

With these alterations of CC, the SFs are defined, modelled and documented in order to facilitate greater trust in the operation of resulting IT solutions.

This work shows the possibility to use CC for the evaluation of ISs, and even if the work identifies problems in doing so, it also presents steps of how to accommodate for these problems.

A Common Criteria framework for the evaluation of Information Technology system security

A method of evaluation with three steps is suggested. The method uses CC as a basis to define all security functions [21].

- In the first step, a list of all functions that could have an effect on the defined security objectives is produced.
- In the second step, this list is shortened as the most effective functions are singled out.
- The last step deals with comparing the list with the functionality of the existing TOE.

This work suggests using the SFR of CC as a basis for security evaluation, thus supporting the idea of this thesis.

2.3.6. Quantifications

As quoted from Lord Kelvin in the introduction of this paper, you have to be able to measure and quantify the objects of interest in order to reach true scientific methods. Here follows some texts that try to introduce measurement in IT security.

Modeling of Distributed Systems Focusing on IT Security Aspects

Ideas for reaching compound values for securability are proposed [7]. Sampled values of security measurements of different aspects are combined into one value, possibly a vector, as seen in Figure 3.

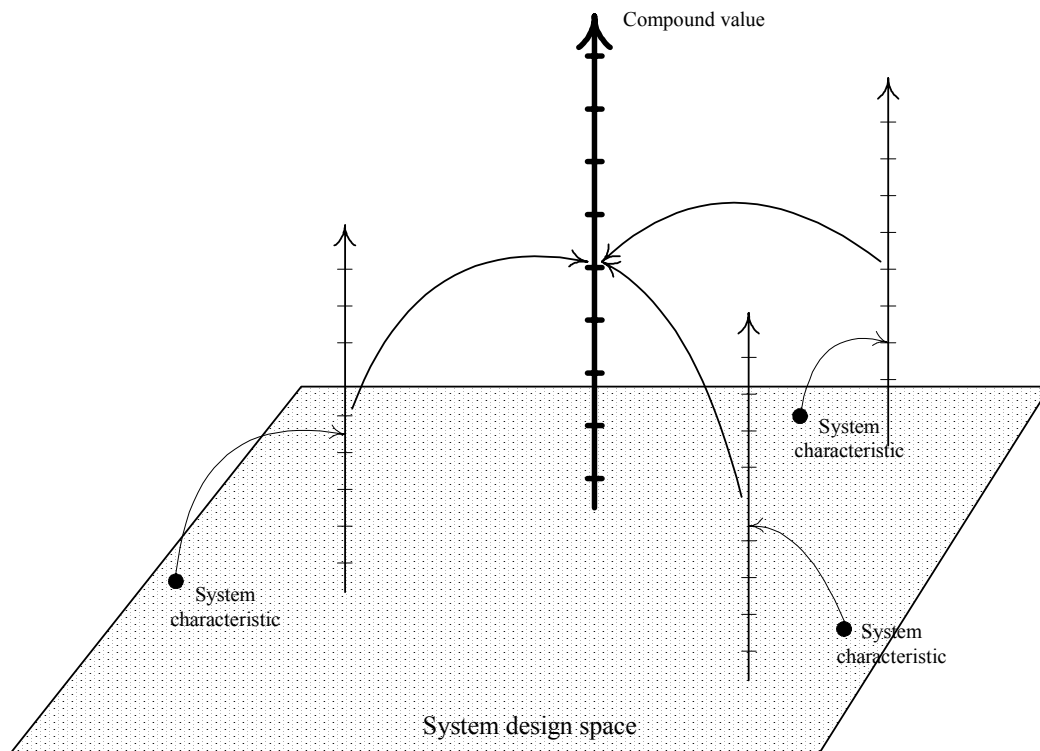


Figure 3: Sampled Securability [7].

A framework for security measurement

A vector of numbers, each number being a function of some aspect of computer security, was suggested to accommodate for the fact that security is a multi-dimensional attribute [22]. Further observations states that an estimation method must be made for the security measurement, since direct measurements of security properties are made impossible due to the scopes and structures of modern computing systems.

The decomposition of the system to be evaluated is then accomplished by setting the system as a root node, and then dividing the system into increasingly more specific components as children to the root node. The leaves of this tree-structure are assumed to be measurable. While traversing these leaves with estimated security values, a value for the root node will eventually be reached.

Different mathematical solutions are also proposed for combining nodes at the same level, depending on the meaning of the component that the nodes are representing. For example, when two nodes are both equally vital for the success of the parent node; the node with the least security value is selected as the value for the parent node. Moreover, weighting, priority and sensitivity methods are suggested in the paper.

This work showed the need for relevant security metric and values, and also argued for the strength in letting the security value range from 0 to 1, to be able to use potent mathematical functions with a focus on probability.

A Common Criteria framework for the evaluation of Information Technology system security

In this work by Kruger and Eloff [21] (also mentioned under 2.3.5), numbers are appointed to the security functions, referred to as “*strength of association*” (SOA). This number determines the effect the security function will have on the objective. The functions are structured in a tree together with the objectives, and the nodes have been given SOA-values. Then the impact of every function on the objective is calculated to see their respective effects on the objective.

This work gave some ideas to the graph structuring of the components in a DIS when combining the components, which is explained in 5.5.

2.3.7. Attack modelling

Attack-models often use graphs, where the states of the system are the nodes and the attacks, failures or actions that lead to deterioration are the nodes. Some value of situation-specific significance could be placed at each node.

On the Functional Relation between Security and Dependability Impairments

This paper is an attempt to establish defined concepts to clearly describe the different steps when dealing with attacks, and how they affect the integrity and correctness of computer systems [23]. The work aims only to unify the terminology and not to give a correct reflection of the reality.

The system is partitioned into several parts depending on its placement. Nodes are used to represent the internal states, and links represent the action that will change the internal state of the system (e.g. the action *failure* will change the state of the system from correct state to failed state).

A Graph-based system for network-vulnerability analysis

Phillips and Painton Swiler introduces an attack graph, where each node in the graph represents a possible attack state, and links represent a change of state caused by a single action taken by the attacker [24]. Each link has a weight representing a success probability, average time to succeed, or a cost/effort measure for the attacker. Using time as a metric gives a more obvious meaning to intrusion detection.

The attack graph can be used to simulate various attacks, or to identify attack paths that are most likely to succeed. The attack graph could also be used to identify undesirable activities attackers could perform once they enter the network. The major advantage is that the method considers the physical network topology in conjunction with the set of attackers.

Listing of vulnerabilities and security exposures

In order to learn relevant attacks so that they could be covered in the model, it might deserve mentioning that Mitre, a non-profitable organization working for the interest of the public as a national resource in the U.S.A., aims to standardize names for all known vulnerabilities and other information security exposures. This list should be considered as a dictionary, not a database [25].

2.3.8. Vulnerability assessment and penetration testing

Penetration testing could best be described as breaking into networks to identify vulnerabilities and to secure them. Vulnerabilities could for example be configuration problems or known software bugs. It could be viewed as hacking into the system and repairing the security holes in the system before a malevolent hacker finds those holes and takes advantage of them [26].

There are possibilities to measure security from the results achieved from penetration testing [27]. A security measure could be an estimation of the total effort required by attackers to penetrate the system.

3. Preliminary approaches

Here the work process is being explained, together with some background reasoning and ideas that helped in reaching the results.

3.1. From tree structure to ontology

Since the foundation of the thesis is based on a previous report [8] (see 2.3.2), the characteristic tree (Figure 1) became the starting point for the research. The main goal was initially to develop this tree further to get measurable system characteristics that could be used to estimate the security of a system. However, when investigating the problem area, some critical drawbacks about the characteristic tree were found.

- The characteristic tree from Figure 1 is not a tree structure. The definition of a tree is a connected graph without a cycle, and the number of nodes being one more than the number of links. As can be seen in Figure 4, the graph is not a tree because it contains cycles, for example confidentiality – access control – authentication – cryptology – confidentiality. Another structure than a tree should be used to represent such relations.
- Although confidentiality, integrity and availability are fundamental terms in IT security, the use of them as top-nodes in a categorization is not recommendable. As seen in Figure 4, both software and access control derives from all three of the top-nodes, and this will be the case for many other elements as well. Thus, categories that will yield better partitions are recommendable.
- The exact meaning of the relations is not specified. Sometimes the relation is “*method that introduces a security characteristic*”, as in the case with non-repudiation and accountability in Figure 4. Between some elements, the relation can instead be interpreted as “*physical component that implements security function*”, as it is between access control and firewall in Figure 4. The exact meaning of the relations should be obvious in the figure and the structure should be able to handle a more stringent way to partition the objects, like in a taxonomy.

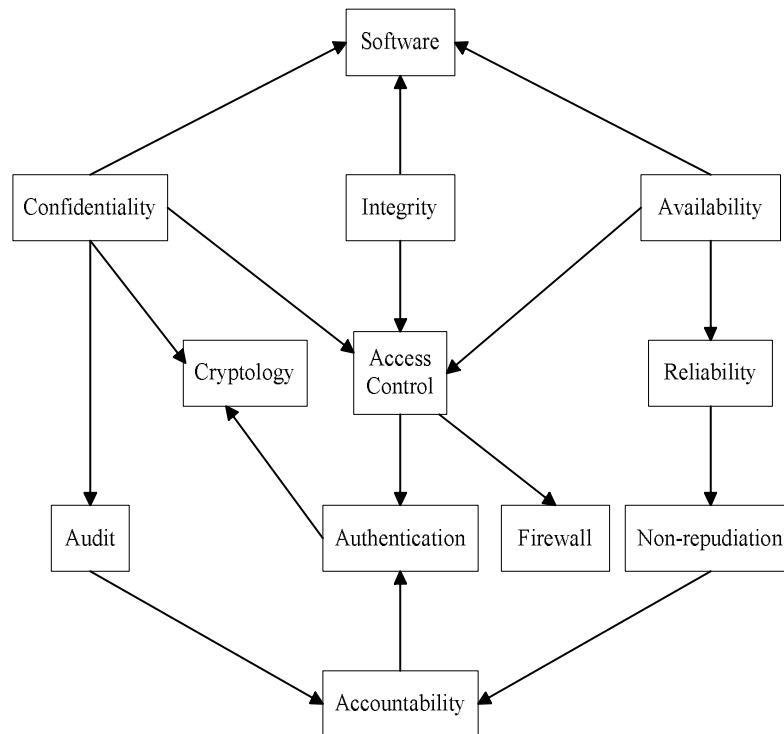


Figure 4: Restructuring of some elements from the characteristic tree.

A first approach to solve the problems of unspecified relations was to break up the tree into several trees; one physical tree containing the system components, and one defence-tree that holds the security functions and policies. Then a mapping can be made between these two trees to show what security attributes specific system components contributes with in the DIS. Additionally an attack tree can be created to list all attacks that exists and thus making it possible to examine what parts of the system in the physical tree or security attributes in the defence-tree that might be vulnerable to these attacks. This makes it easier to discover weak spots in the system. The development of an attack tree was considered unimportant at the present stage of the thesis. However, it may be useful later.

For the physical tree, partitions made in earlier discussions [5] were found suitable. It partitioned a system into technological, organizational, operational, environmental and individual components. Of course, the first partition is more intuitive as all physical system components belong here, and this is what the initial research should be concentrated on. However the other categories are also very important in order to evaluate a real system in use (or a contextual real system in use according to Figure 12).

In order to categorize all the parts of a distributed information system (DIS), a structure consisting of nodes representing physical objects is needed. The objects on the lower levels of the tree are dynamical and will change noticeable over time as new system parts are invented and developed. However, on the higher levels of abstraction, where the categories of the system parts are more general, the objects will be more static. For the

objects on the lowest abstraction levels, there must somehow be possible to perform an evaluation or estimation of the security attributes.

This way of fitting relations between an attack-tree, a defence-tree and a physical tree is similar to ontologies. Thus, ontologies were studied and found to be suitable and beneficial for the modelling of all of the objects from these three trees, their individual relations and other useful information. Ontologies do not have the weakness of unspecified relations, as they are clearly stated. The structure is also dynamic, distinguishing between heritage relations, adding taxonomy-like partitions, and other relations.

3.2. Locality model and dependency algorithm

When a structure for the system components and security defences had been developed, a way to divide the system and its components for the model had to be introduced.

As a first system model, a sort of physical scope was made. This divided physical rooms into spaces, sectors, that were delimited by doors. Furthermore, which persons who are granted access into which rooms were also important. The sectors were named compartments and were linked together by doors with some kind of access control. Doors without any protection did not divide two compartments as they did not restrain access to that compartment.

An example of a segmented area with different compartments is shown in Figure 5. There are two rooms (1 and 2) each containing a single computer and a locked door that restricts access. A bio-scanner controls access to the corridor (3) leading to these two rooms. The room (4) that leads to the corridor also leads to a room (5), with a locked door containing a firewall. The building is protected by a fence and a gate guarded by a watchman. Note that the area just inside the fence and the main entrance belong to the same compartment since the main door does not have any access control.

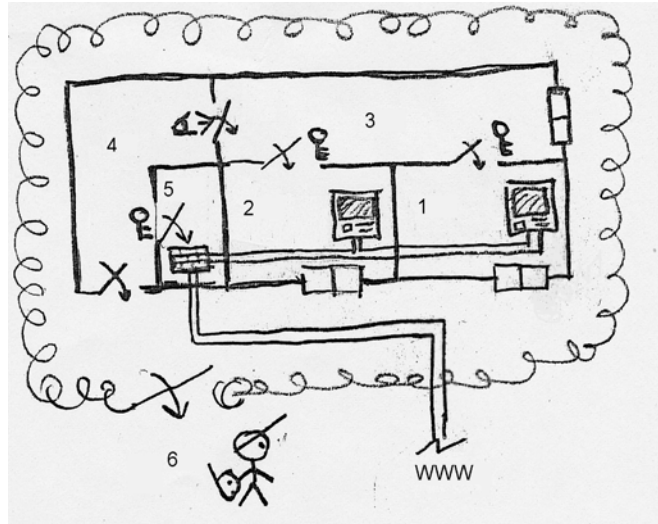


Figure 5: Example of a restricted building segmented in compartments according to physical clearance and net layout.

The compartments can now be seen as nodes in a graph, where every door and passageway between the compartments represents the links in the graph. Also the structure of the network can be represented in a graph. How these two different graphs look like (if drawn from the example in Figure 5) can be viewed in Figure 6. The number of each node represents the number of the corresponding compartment.

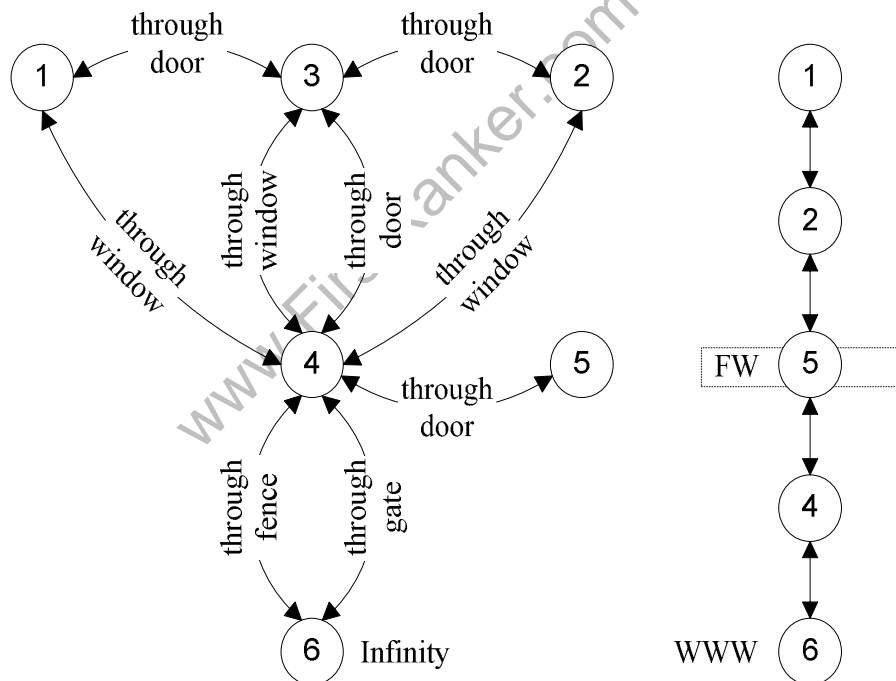
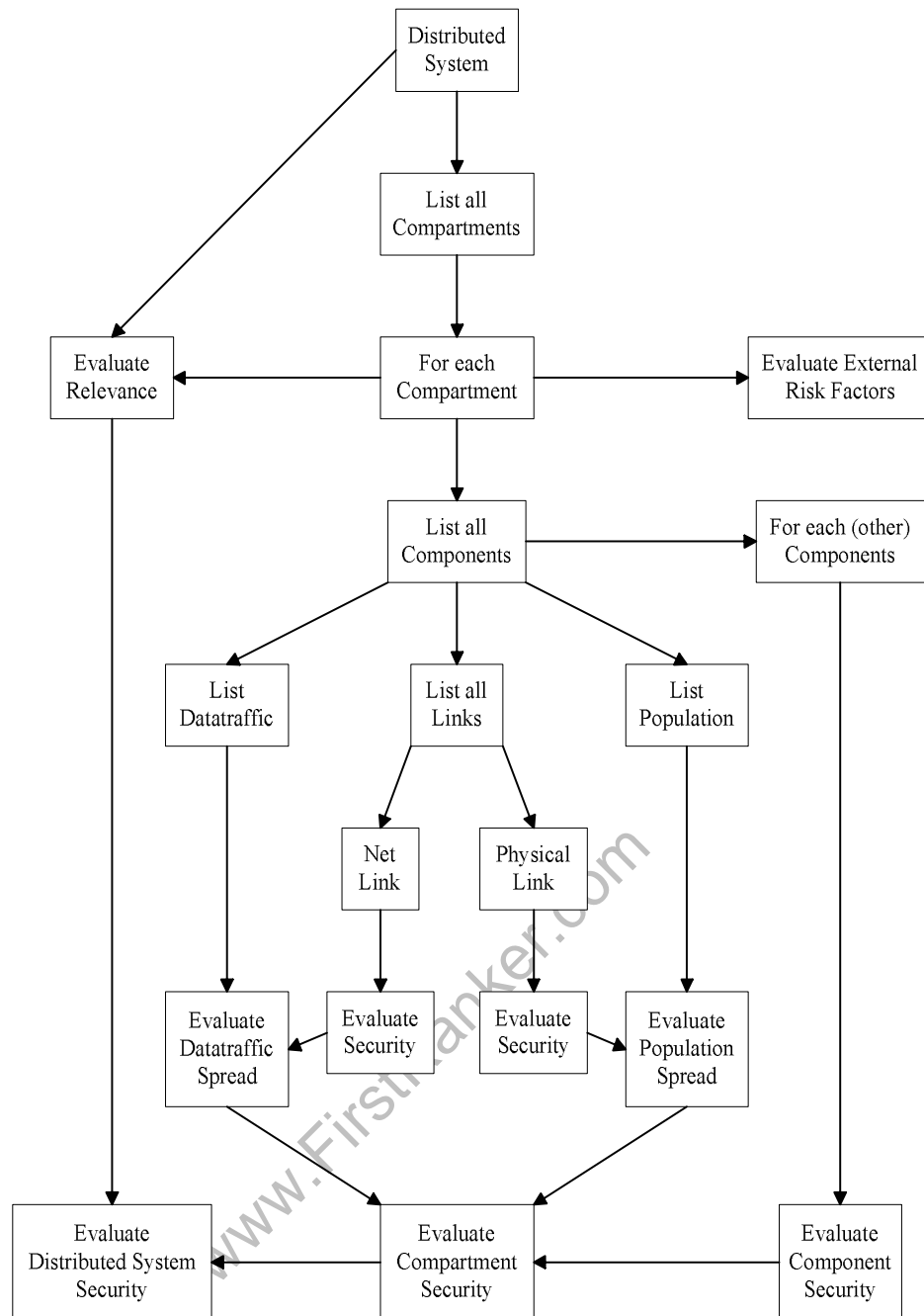


Figure 6: Graphs for physical segmentation and net access, both derived from the segmented building shown in Figure 5.

Each node is then assigned a list of users that are to be granted access to the corresponding compartment. The outermost segment (representing compartment six in Figure 5) is, as seen in Figure 6, assigned the infinity (or WWW) since no control is possible here and the entire population of that area is an equally possible threat. Now each link is assigned a probability value reflecting the effectiveness (i.e. how effective the access control protecting a connection between two compartments is at keeping unauthorized people out). The compartment model resulted in the creation of the “*dependency algorithm*” shown in Figure 7 below.

In the dependency algorithm, the different compartments are evaluated one at a time. Every connection is analyzed and evaluated. Also other components in the compartment have their security evaluated, and their authorised users and authorised data traffic stated and estimated. Now, with the help of the evaluated links, the estimated spread of unauthorised users and data traffic can be calculated to show where this spread is most likely to occur. When finally all compartments have been evaluated this way, their result could be added up to represent the security evaluation of the whole distributed information system.

www.FirstRanker.com

Evaluation of the Security of Components in Distributed Information Systems**Figure 7: The dependency algorithm.**

Even though the model works well to illustrate physical security, like intrusions, thefts, fires and access control, it has flaws in the modelling and evaluation of system components, mainly because of the focus on the sectors which have no real meaning in most aspects of IT security. However, the model should work well as a complement to another model, focusing on IT security.

This model is what later became one of the scopes in 4.2; the locality model.

3.3. CC and component structure

Since the previous locality model was found to be more focused on physical security than of IT security, another way of partitioning the DIS had to be found. The most common method, and also that of the previous thesis [8], is to divide the system into system components, evaluate them separately and then combine them into a common evaluation.

This method was found to fit well with the ontology structure, where components are hierarchically ordered concepts of the ontology. Evaluated components will become instances in the ontology while creating a knowledge base. Ontologies are further described in 4.6.

When pondering over what the defence tree from above should contain, the security functions of the Common Criteria were found suitable for this task. The main reason is that CC is commonly used; it has been thoroughly developed, and is perhaps the most important standard for computer security of today. The way STs and PPs map system component to security functions is also very similar to the idea of this thesis. The methods and terminology of CC that are suitable were introduced in the model. It should in theory be possible to calculate the security values of system components, which is further discussed in chapter 5.

3.4. Initial framework

To deal with the fact that necessary restrictions in the system model should not hinder it from evolving, there was a need for a dynamical framework that covers all system aspects, not only those needed for this thesis, but also for all possible aspects that might be relevant in the future. In order to achieve this, a framework that is general in its structure was developed. Most aspects could now be fitted into the framework, and it could also easily be changed.

The initial model, seen in Figure 8 below, highlights the differences between the system model and the actual system. This was made to stress the differences between the system and its model, making it clearer which aspects that are approximations, to what degree and exactly which characteristics they are trying to estimate. This is the original framework that later became Figure 10, when the comparison between the reality and the model was removed, and the model became more detailed. Because the figure is obsolete, it is not intended to be fully comprehensible in this chapter, some terms and views have changed or evolved since it was created.

Evaluation of the Security of Components in Distributed Information Systems

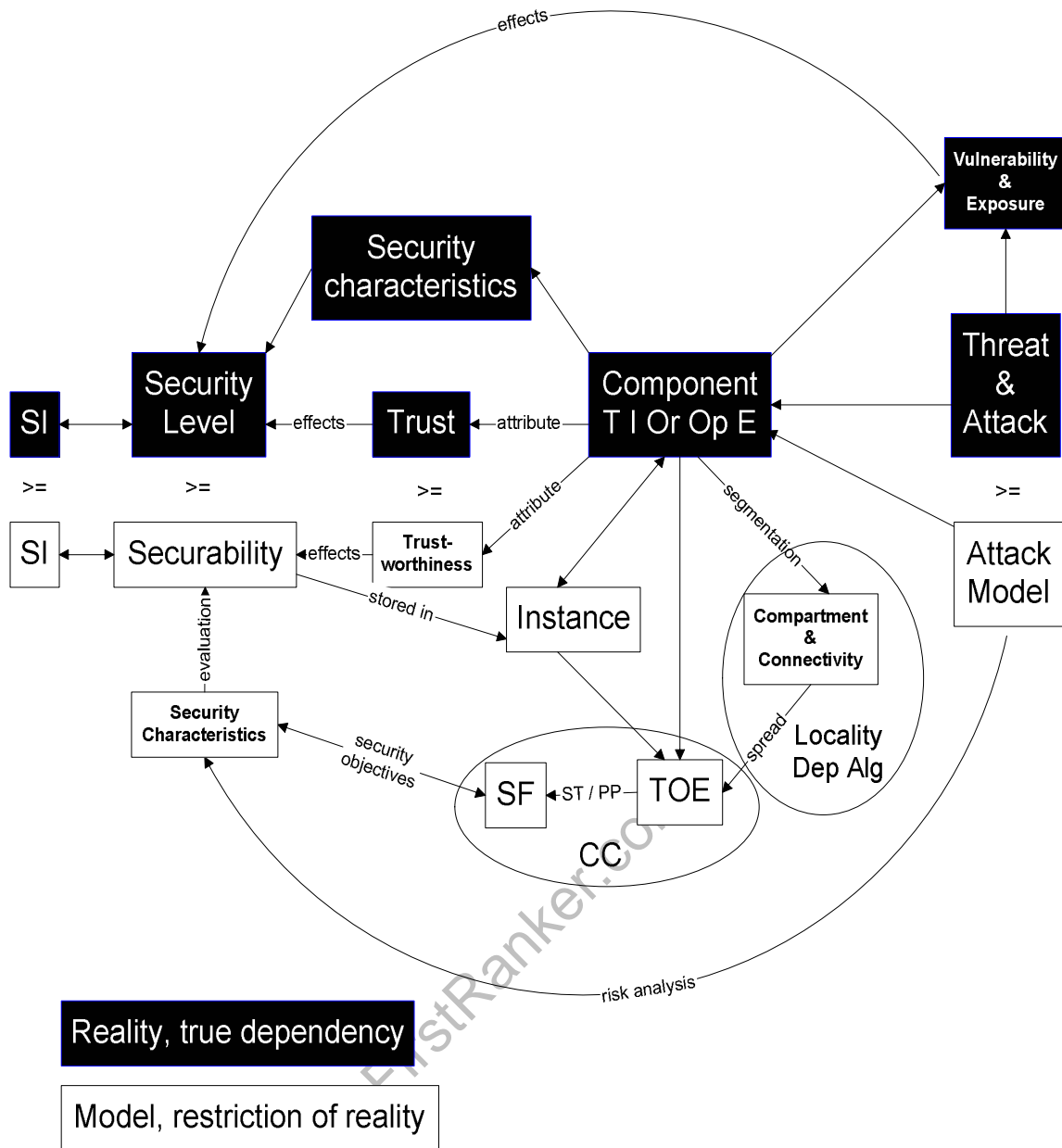


Figure 8: Initial security evaluation framework.

4. Security Evaluation Framework

The main purpose of the security evaluation framework is to generate a system modelling technique that models a system, and to generate an evaluation method that is capable of evaluating the security in the previously generated system model. This explained functionality, together with the possibility to implement a DIS given the system model, are illustrated in Figure 9.

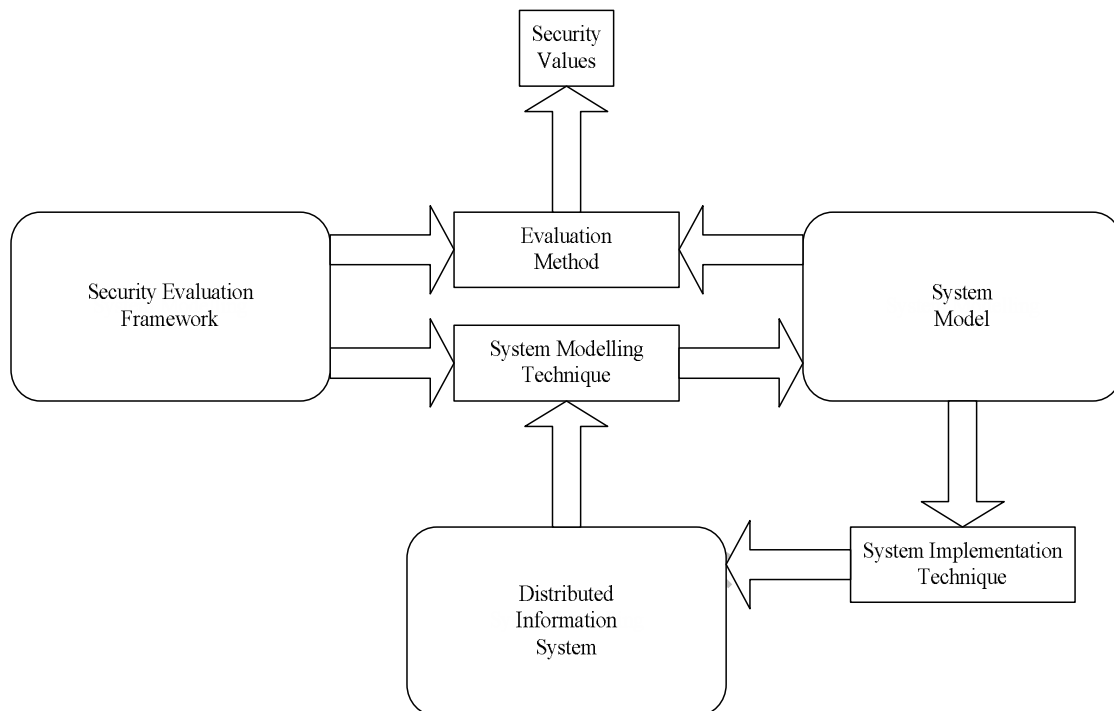


Figure 9: Overview of security evaluation framework with relations to important concepts.

To address the need to evaluate security levels in an IS a framework is introduced. The framework originates from the black component in the middle of Figure 10, which all other parts of the framework relate to in some way, as shown with the arrows. The solid arrows indicate that the concepts are inherited thus indicating a strong relationship. The dashed lines indicate a much softer relation with a meaning that will be explained later on in this chapter. Recursion is indicated with the arrows pointing in both directions (between TOE, instance and technological), showing that evaluated instances are stored in the component library to be used later on.

The component is partitioned into several concepts depending on in which partition the component belongs, represented by green blocks in the figure. The environmental component leads to a contextual concept that divides into concepts dealing with risk analysis, which are marked red in the figure. The technological component deals with the system components, their conversions into TOEs, evaluated instances of components, and systems made up of instances of components. The TOE relates to the different security values via the CC SFR. All these concepts originating from the technological component

are in the figure marked in yellow. Different security values are reached depending on which component partitions that were regarded. The different security values are coloured blue in the figure. The technological components and the security values, the concepts visualized in yellow and blue in the figure, are described in chapter 5.

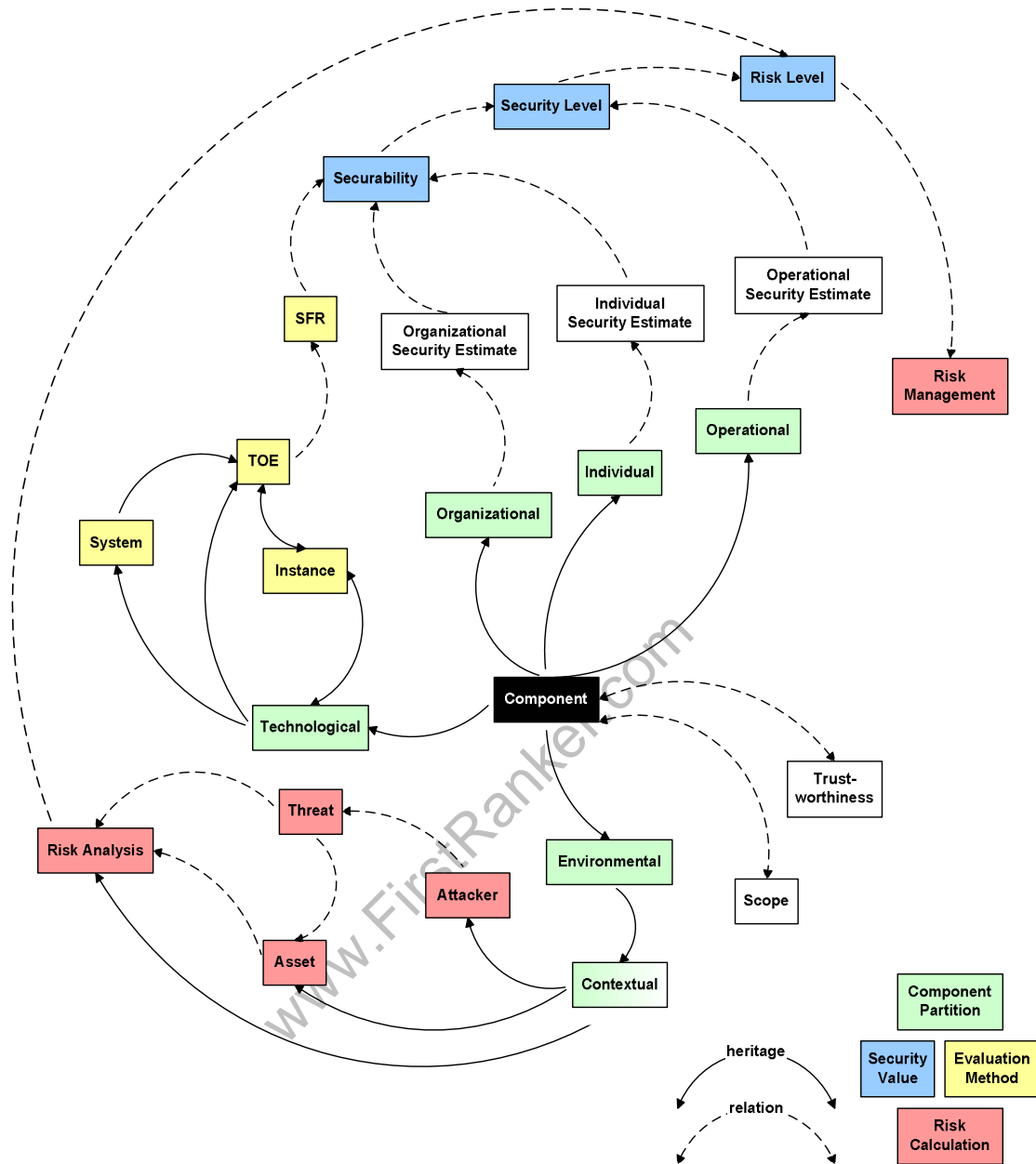


Figure 10: Security evaluation framework model.

The framework represents the entire evaluation process in which a security value is being estimated. It models the reality and those restrictions that are being made in order to get accomplishable evaluation methods. The entire evaluation process is being divided into increasingly smaller parts, where each part in the framework of evaluation can be viewed

as one independent step on the long road to the evaluation of a DIS. Every part is represented as a block in the figure and the arrows show the dependencies between the blocks and how to expand the results in order to calculate meaningful values for the securability of the TOE. Every block can also be thought of as a module, where all modules work together for the common goal of the framework. This modularity concept is explained at a greater detail later.

The framework is a highly dynamical structure, giving evaluators an opportunity to change it according to their own interests and needs. This is an excellent starting point in order to create methods for a security evaluation. There exist no apparent restrictions in the framework method as a whole; only in the modules themselves as it is here the restrictions are being made. Most restrictions are visible and marked, making the problem of accommodating for them, perhaps not always easy, but at least a specified, and hopefully solvable problem. For example, by letting the module “SFR” represent the evaluation of a TOE towards a security value, restrictions on how the security of the component is measured and evaluated are introduced. Some of these come from the fact that the set of security functions contained in SFR not cover all existing security functionality exactly, others from the mapping and evaluation of each system component to the appropriate security functions.

In the following sections, the characteristics and features of the security evaluation framework will be explained in detail. In 4.1 the modularity concept is explained together with the benefits that come with it. Depending on what the evaluation should focus on, different scopes may be useful. The scopes segments and models the DIS differently, and some examples of scopes together with clarifications can be found in 4.2. Evaluated TOEs are stored as an instance in a component library together with the results of the security evaluation as explained in 4.3. The evaluation itself is a crude estimate of the TOE that rates its security properties according to the Security Functions of the Common Criteria. The process of this evaluation is explained in 4.4. The difference between the model and the reality, together with differences depending on how detailed the model is regarding non-physical aspects are discussed in 4.5. The system with its components and instances and their relations to the SFR and the security evaluation are structured and visualized by the use of an ontology, as described in 4.6.

4.1. Modularity

An advantage of the framework is that it becomes very flexible. When creating methods for security evaluations, every restriction of the reality should be clearly visible in each module. There is not really that much of a problem restricting the evaluation method since it is easy to create a new module that is less restricted and fit it into the framework. The entire evaluation method will not have to be changed, only the module itself. Every module is a clearly stated subtask of the process of evaluating the system. Hopefully this will make the framework easy to evolve into an increasingly better structure for security evaluation. Also there will be great opportunities to creating modules in the framework suited for specific demands or tasks.

By adding more modules into the framework, the complexity of each module decreases as the problem space is divided among the existing modules. This together with the hope that the performance of the modules will improve, give reason to believe that the restrictions in the framework will decrease, making the framework more and more complete over time.

4.2. Scope

Different scopes may be chosen for the segmentation of the system, depending on focus and goals of the evaluation. Several scopes may be regarded simultaneously, to get a greater perception of the specific view of the system or model evaluation.

Examples of different scopes are the following.

- **Component structure**
How the components, both hardware and software, fit together is one possible aspect to consider. This is the most intuitive scope and is the one which is considered in the report if nothing else is mentioned.
- **Information flow**
How the information flows in the system could be important for certain security tasks. If the security evaluation was to focus on the importance of the assets, and which subjects that were to be granted access to these assets, this scope of interest would probably be adequate.
- **Services provided**
It is possible to view the system as a set of services. This scope is similar to information flow, but on a higher level of abstraction since it is the services and applications that are focused on and not the information. Services could for example be to supply internet-pages or download files through ftp-servers.
- **Physical structure**
This model is a physical segmentation of the network, regarding compartments as nodes, and having connectivity links connecting the nodes. Connectivity-links are used for doors and entrances to connect the compartments, thus making it possible to model threats regarding physical intrusion. This scope is described as the locality model and dependency algorithm in 3.2.
- **Attack model**
An attack model shows how the system is supposed to handle intrusion-attempts. This scope is derived from the attack-modelling described in 2.3.7, where the system states is represented by nodes, and links represent the events that lead to a change of state. The other scopes are likely to benefit from collaboration with this scope, as it is useful in highlighting weak spots in the security modelled by other scopes.

- **Security prioritized components**

One way to divide the system could be to focus on the most relevant parts. What is considered as the non-important parts of the system are simply ignored in order to concentrate on the more important parts, i.e. prioritizing the decisive components. Another aspect dealing with effectiveness is that instead of worrying about what might get wrong; it is better to study what is likely to fail [28].

4.3. Component/System Structure

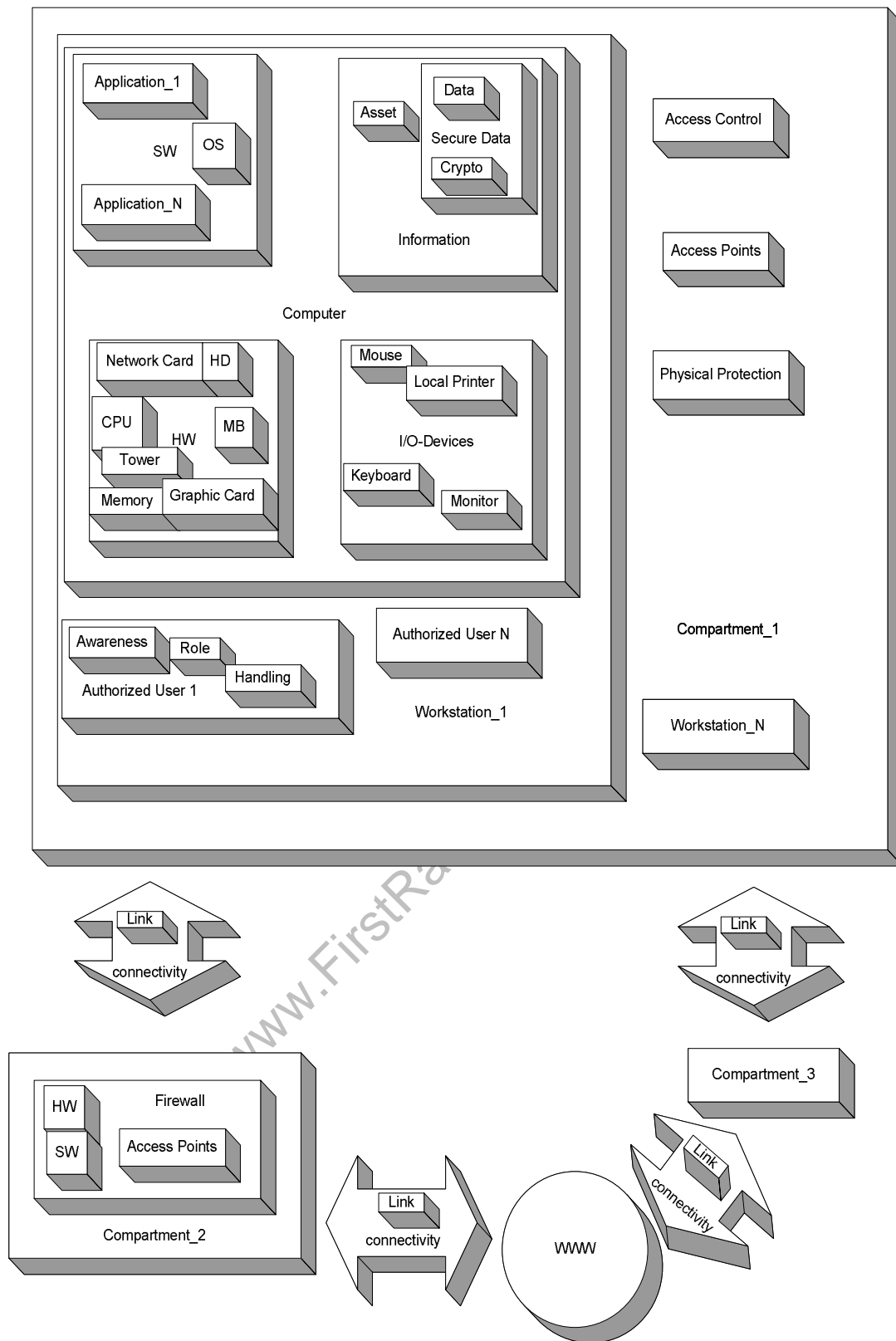
This section is divided into three subsections. The first explains the idea of a component library. The second reasons about the dependability in the system and in the third subsection some thoughts about how the human user fit into the model are presented.

4.3.1. Component Library

The idea is to build up a component library by storing instances in a component knowledge base. The typical instance will be a component of a specified brand and model or a general standard component. The instances are already evaluated TOEs with an estimated security value.

It is up to the evaluator to choose how to model the system to be evaluated, but a standardized component library, with already evaluated default components, is useful since they are reusable and can be put straight into other system models. The evaluations will become less demanding, since already evaluated components will not have to be analyzed and evaluated further.

An example of a component structure can be seen in Figure 11. Every block is a TOE. In order to evaluate it, all blocks that are contained within have to be previously evaluated and put into the component library. The system described in the example contains several workstations, each containing a computer accessible by several different users. These workstations are supposed to be situated in the same room area (denoted compartment). The compartment also contains physical protection (i.e. lock, bio-scanner or anything else that aims at keeping unauthorized people out of the compartment). Different compartments are then hooked up to each other and to the Internet by connectivity links. This segmentation with compartments connected with connectivity links derives from the early “*locality model*” (explained in 3.2), and is now only regarded in the physical structure scope (see 4.2). This particular scope is chosen for this example in order to better illustrate the features of the component library, as this scope better divides the information system into clear blocks.

Evaluation of the Security of Components in Distributed Information Systems**Figure 11: Example of a component structure for a distributed information system.**

4.3.2. System Dependability

A comparison between building houses and making IT systems more secure, stresses the importance of dependability [3];

“Building secure systems is like building a house. We liken correct low-level coding to using refractory bricks instead of bricks made of sawdust. The kinds of bricks we use are important to the house’s integrity. But even more important (if the goal is to keep bad things out) is having four walls and a roof in the design. The same thing goes for software: which system calls are used and how they are used is important, but overall design properties often count for more. In general, software security research has paid much more attention to bricks than to walls.”

This metaphor is similar to the notion of “single points of failure”, where collapse of critical points in a system leads to major or catastrophic system failures [17]. It is an important task for the evaluation to recognize these points, so that the security values truly will depend on them.

The security values are functions of identified variables that will change depending on the specific system that the instance is in. Relations and details regarding change of the security in components, when combining them with other components into subsystems or when the scope of the system changes in some way, are decided for each specific component. All these different characteristics have to be decided when evaluating the TOE, which will require large amounts of research and extensive practical testing. A specific method for the combination of components will probably not be possible to develop since most components are unique in their behaviour and with characteristics that differ enormously among each other.

4.3.3. Users in the model

Exactly like in CC, the evaluation is being made within a TOE. A major difference is that the framework makes it possible to include the end-user in the TOE, while in CC the end-user is outside the scope and of the entire evaluation whatsoever. The only users that to some degree are considered in CC are those with predefined roles that set up tasks needed for the SFs to function properly (e.g. administrators).

CC is a strictly technological evaluation, meaning that only the specifications of requirements of human users can be assured, not the user itself. However, since all aspects have to be regarded when evaluating a system, the user itself can not be ignored in order to receive a meaningful evaluation. As observed in [28], component-oriented security standards often ignore one of the most important factors, the human element. They fail to ensure that the skills and performance required of various kinds of staff are included.

Observations are being made that the most serious threats arise from within organizations [29]. Several examples of the security failures introduced by human users are being enumerated. Note that these examples deal with the administrative aspects of failure. Some of the most common ones are:

Evaluation of the Security of Components in Distributed Information Systems

- misconfigured firewalls
- servers with no specific protection
- anti-virus software with old virus-definitions
- poorly managed operating system and application patching
- shaky account administration
- dead accounts
- logs for the sake of logs. Or as described in the following quote: *“Advanced monitoring techniques are rendered useless by the last two feet between the telephone-book-sized weekly report and the eyes of the officer”* [18].

Also the end-users may introduce severe security problems into the system, for example the following ones:

- sloppy password-handling
- accidentally installing viruses or worms into the system
- counteract or disable security functions in the system
- sending important information over the Internet without encryption.

The insecurity problem can be blamed mainly on the interactions between humans and computers, which is captured in the following phrase [18]:

“We’re trying to secure a system that embodies human processes and includes human users, but we restrict our analysis and designs to the computers themselves.”

Part of the problem is that there exists a gap between what the system does and what the user actually believes it does [18].

With the ability to also take the user into account, the possibilities of the model increases, since attributes and relations concerning the user can be modelled. Of course, if the scope of the evaluation is to focus on the technological view, it is easy to make restrictions to more or less ignore the influences of the actual user, making the evaluation more independent of the user, just like it is in CC.

4.4. Method of evaluation

What to measure is a very important aspect of the model. This thesis has chosen to concentrate on the evaluation of technological components, i.e. the estimation of securability according to 5.1. For this purpose, the method of evaluation uses Common

Criteria (CC) as a foundation, or more specifically; its set of Security Functional Requirements (SFR). This is because CC targets the security in technological components. In order to evaluate other components than technological, other means than using the SFR must be used. For example, when evaluating organizational components, the certification method ISO 17799 may be used (see 2.3.3).

The component whose security is being measured is called the Target of Evaluation (TOE). An ST or a PP may be obtained and analyzed to clearly map the needed characteristics of the system component to the components of the Security Functions of CC.

The system component gives specific values to the CC components, depending on how effective the component is at enforcing the specific functionality.

These values can be mapped to CIA and/or PDR if a more meaningful, or distinct security value is of interest. One securability-evaluated TOE is then stored together with its securability characteristics in the component library as an instance (see 4.3.1), so it can be used at a later stage as a building block for a subsystem.

This method of evaluation is explained at a much greater detail in the next chapter. Examples were created to show what an evaluation can look like, and even though the model still lacks some vital parts (e.g. thoroughly testing of system components so that fair security values for the security functions may be estimated), the method of evaluation is easy to follow.

4.5. Modelling and Implementation

One important function of the security evaluation framework is the ability to implement a system model, or model an implemented system, as shown in Figure 12. This could be done in the IS for different aspects, which may be restricted or expanded at any point in both the modelled and the real system. The different aspects are to view the system as only a system structure, a system in use or a contextual system in use. The aspects of the system represent the outer boxes in the figure; while the specific values of the system are embodied in the inner boxes (see 5.1 for an explanation of the different values).

It should be observed that the diverse aspects of the figure do not signify the level of abstraction which remains the same in the entire figure. The only things that differ are if the system is in operation, and whether it involves a risk analysis or not.

The figure shows that it is possible to model the system and implement the model back and forth, as well as append or remove different aspects to both the model and the real system. This gives a high flexibility to the framework since although different aspects are ignored at the beginning of the system modelling, they could be appended to the model at a later stage, if that is desirable.

The above discussion could be illustrated by the following example. The common way to do a typical threat modelling would, adapting to the figure, be to simply model the

contextual real system in use. However, suppose the areas from the risk analysis were to be disregarded and the scope was to be further restricted into the system structure only, the modelling would be simpler and more precise. In order to get to the contextual modelled system in use, the modelled system is first merged with an operational model, and then expanded into the desired system with the use of a risk analysis. By modelling the different parts of the system like this, the modelling becomes more structured and precise, since it is easier to model the system in several steps than to model the whole system

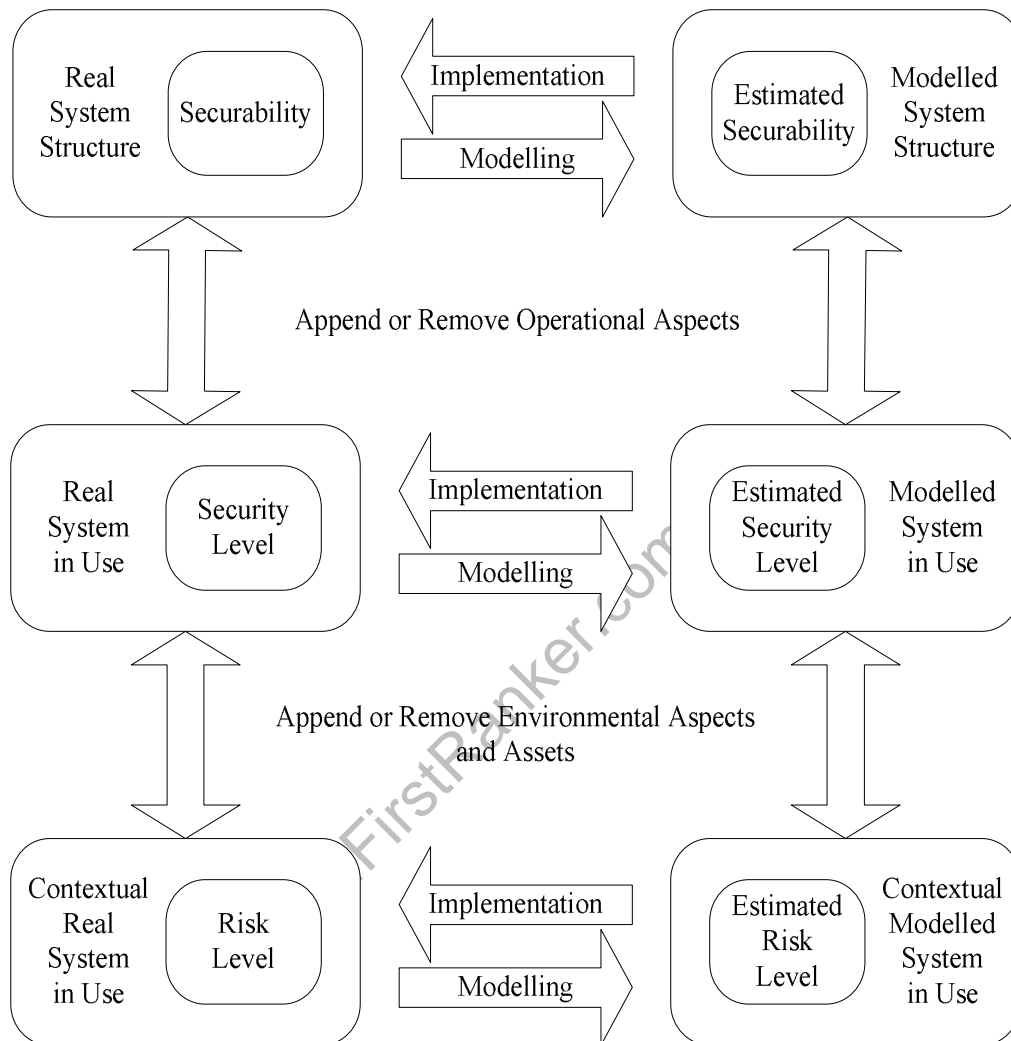


Figure 12: Modelling and implementation. The inner boxes represent the aspect of security value, and the outer ones the system scope.

4.6. Ontology

A quotation enlightens why ontologies are regarded to be necessary for the future of computer security [30];

“What’s missing [in computer security] is a broader context that we can use to organize our thinking and discussion. What the field need is an ontology – a set of descriptions of the most important concepts and the relationships among them. [...] A great ontology will help us report incidents more effectively, share data and information across organizations, and discuss issues among ourselves.”

Objects and concepts in the security evaluation framework are easily described using ontologies. This is a highly versatile way to describe a specific conceptualization of the world. An ontology is very dynamic in its structure, using heritage-relations to specialize specific concepts. All concepts may contain slots of attributes and relations to other concepts. Instances of concepts may be created to turn the ontology into a knowledge base.

The main reason for choosing this way of representation is that it makes the framework model and it’s modularity-thinking, described above, easier to represent. Every module in the framework may be described with its own ontology and then the ontologies may be put together with relations over the ontology-boundaries. If a specific module is replaced by a, at least in some specific aspect, better module, then only a small ontology has to be thrown away, not the whole one. The meaning of the relations between the discarded ontology and the others may easily be put into the new ontology instead.

The ontology created for this work tries to cover two diverse aspects. The first is to have a more dynamical tool to express the security objectives, functions, system components, their properties and the different relations that connect them all, which could not be expressed in a strict and comprehensive manner in the previous work [8] (see 3.1). The second reason for using an ontology is the one advocated in the introduction of this chapter; that we need a common defined terminology in computer security to be able to cooperate and be able to express ourselves understandable. This second reason is more introduced for the purpose that it will become necessary, or at least desirable, at a later stage. As this method evolves, and as the entire area of IT security evolves, the need for a well-defined and specific ontology will increase accordingly.

In Figure 13, the security evaluation framework of Figure 10 has been described using an ontology. The root node, *thing*, is the concept from which all other concepts inherit. From here, the concepts component, TOE, security functional requirements, security value, instance and risk handling originates. The component concept is divided into five partitions depending on focus. One of the partitions, the technical component, has been specified further to exemplify the structure. An extensive specification of this component would be far too huge to be shown here. Another partition, the environmental component is a generalization of the contextual concepts of attacker and asset. These few concepts that are mentioned under the technological and environmental components are only a small part of the whole component library idea presented in 4.3.1. The security functional requirements concept is a generalization of the eleven SFR-classes. They can all be further specified into families, CC components and elements, even though these concepts are not shown in the figure.

Evaluation of the Security of Components in Distributed Information Systems

The concepts of securability, security level and risk level all inherit from security value, and risk management, risk analysis and threat inherits from risk handling. These concepts are connected by named relations. For example, the technical component is translated into a TOE, which after it has been evaluated will be stored as an instance together with its security values. The individual, organizational and operational components are needed to be able to estimate different security values. There are also some relations between the contextual environmental component and the concepts under risk handling.

www.FirstRanker.com

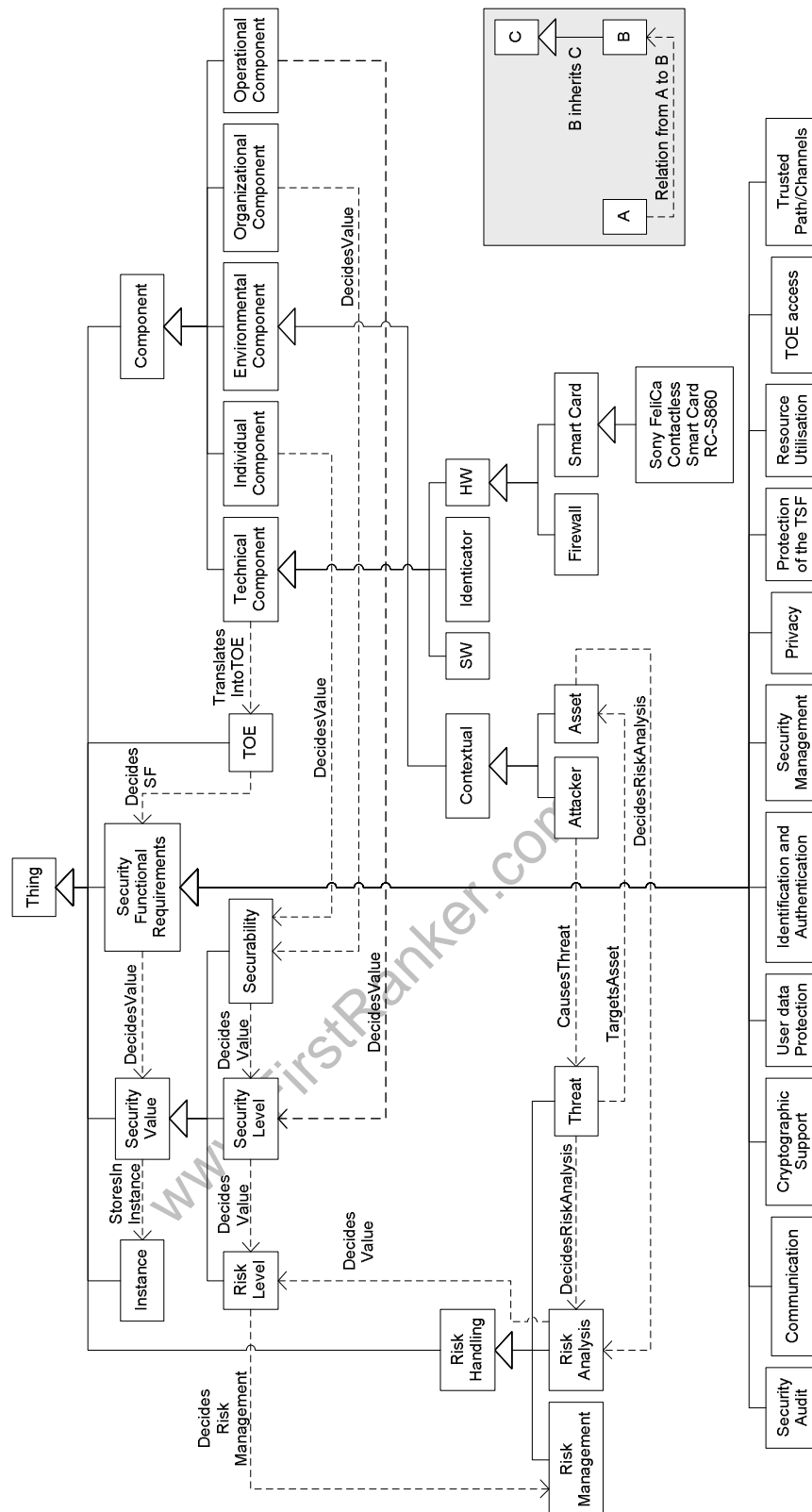


Figure 13: Ontology of Security Evaluation.

The main advantage with the ontology compared to the framework model in Figure 10, or even the characteristics tree in Figure 1, is that the objects in the model and their relations become more structured. It becomes easier to state relations between objects, and the hierarchical inheritance introduces taxonomy-like partitions wherever it is convenient in the model.

Some more general reasons for developing and using an ontology, apart from the ones mentioned above, are the following [31].

- Ontologies will make it possible to share common understanding of the structure of information among people or software agents.
- Creating ontologies will enable reuse of domain knowledge by integrating several existing ontologies that describe portions of a large domain or reuse a general ontology and extend it.
- The use of ontologies makes domain assumptions explicit, making it possible to change these assumptions easily.
- Ontologies separate domain knowledge from the operational knowledge. It will be possible to describe a task of configuring a product from its components according to a required specification and implement a program that does this configuration independent of the products and components themselves.
- Ontologies are helpful in analyzing domain knowledge. A formal analysis of terms is valuable when attempting to reuse existing ontologies and extending them. Developing an ontology is akin to defining a set of data and their structure for other programs to use.

5. Evaluation method

This chapter explains the five steps of the evaluation method focusing on technological components and is, accordingly, divided into five sections.

The first section explains the need for a metric and security values in security evaluation. It also covers some of the problems that occur when trying to reach these values. An idea of using probability estimates as security values is proposed.

The second section explains the Security Functional Requirements (SFR) of CC, the meaning of the TOE, and the changes made in the SFR in order to suit the purpose of this thesis.

The third section explains how to interpret evaluated SFs and also explains and exemplifies the steps to follow in order to reach meaningful security values.

The fourth section explains how to map the characteristics of a system component to the SFR of CC. It also explains and exemplifies the steps to follow in order to perform this mapping.

The fifth and last section explains the combination of several evaluated components into one evaluated subsystem. It introduces graphs where the system components are represented by the nodes. It also gives some examples on mathematical functions to use for the combination of the estimated components.

5.1. Security values and metric

Some significant, quantifiable measure is needed in order to reach a meaningful evaluation of a distributed information system. As quoted from [3];

“Only by quantifying our approach and its impact can we say if we are better off now than before”.

Quantifiable security properties will be needed for efficient decision support, since it is easier to assess a security property if it has been assigned a security value, preferably relating to an intuitive metric. It will also be easier to report an IS's status or posture if such a security value do exist. Exactly what it should cover is not an easily formulated problem, since most evaluators tend to demand and focus on completely different aspects of security. For example, government and commercial sectors have different agendas; the former is policy driven and the latter is profit driven. IT security is a vague and imprecise word, and therefore no single security value will be able to successfully represent the security value of a system. Additionally when ISs evolve and extend, the problem of making statements of their properties increases drastically.

A universal security measure will not solve these quantifying-problems, but it might be a step in the right direction; to best approximate the security of a system [22].

This security measurement can not be made directly by simply measuring the strength of the components. The dependency and structures of the ISs are simply too complex, but instead an estimation of these properties must be made.

Since security professionals put different meaning in the word security, it will not suffice to have only one security measure. Different kinds of values will therefore be necessary depending on the focus of the evaluation. The security needs are different depending on the situation. For example, a system that has quite mediocre defences, but contains no relevant assets whatsoever, may be considered secure enough. It can not fail in protecting any vital information. On the other hand, a system that is very secure in its physical structure may be rendered insecure due to improper handling by the staff. The three following values all originate from the same basis, but differ since they are put into different contexts. Anything similar to this segmentation was not found during the literature study, so these categories had to be introduced in this thesis. The values and their mutual relations are illustrated in the inner boxes in Figure 12 (see 4.5). The term “*security value*” could be used for all of them, or for any single one of them, if the specification is unknown or unnecessary.

- **Securability**

The securability in each component is evaluated. This is a good basis for establishing goals in information systems and measure how well these goals are fulfilled. It also enables comparison among different products or brands. However, the targeted system is not in operation, but estimates of how the organization, as well as individuals within, handles security is also regarded.

- **Security level**

If the operational components are applied to the securability, the evaluation will instead be a measure of the security of the component in operation. This measure is called the security level of the component, and can be viewed as how secure the system is once it has been “*switched on*”.

- **Risk level**

Finally, the component can be put into its contextual environment where threats and assets are regarded, which results in the risk level. This final security value enables the possibility for risk management.

The proposed security value should range from 0 to 1. This value will be somewhat of a probability estimate, i.e. probability may be regarded as the metric for the security value. It could for example give an indication of the probability that a random attack or vulnerability leads to a security failure. A zero (0) for a specific SF, will indicate the significant possibility of a vulnerability. A one (1) will imply that the system is as secure as it possibly can be regarding the specific security functionality. There is no possibility whatsoever for the particular function to fail. If an SF is not valid for a specific TOE, the NULL-value is used.

Of course, there is no possible way to give a correct and justified exact evaluation of such a number, instead the value is intended to be used in situations like if you have two

different systems with evaluated security values, you should be able to predict that the one with the larger value is more secure than the other. Exactly what a specific value will mean is not decided at this stage, i.e. no metric is yet defined. Later, if more systems become evaluated, it will be easier and more meaningful to rank systems according to their security values.

The introduction of a security value gives advantages to the evaluation compared to the ways of certifications like CC, since systems can be rated and compared to each other. The only thing in CC that is similar to a security value is the “*Strength of Function*” (SOF), which is estimated for a few CC components (e.g. cryptographic operation) during a CC assessment. The SOF is a discrete estimation, that is either SOF-basic, SOF-medium or SOF-high.

Although the security value has a theoretical meaning, even if it is ever so vague, the metric on which the value is decided is not clearly specified. Only the extreme end-points of the metric are indicated, but the rest of the metric will evolve together with the evaluation model.

5.2. CC Security Functional Requirements

This section explains the meaning of the CC Security Functional Requirements (SFR), and also the purpose it has for the evaluation as a whole. Some basic information about CC can be found in 2.3.4. For the purpose of an information system model, it is only the SFR of CC that is interesting since there is no place in the evaluation process for the assurance as it is regarded in CC. This is because of the fact that the classes defined in CC Standard Assessment Requirements (SAR) deals with the development process of a system component; it has nothing to do with the security requirements covered in SFR.

This way of using SFR for the evaluation process, but ignoring SAR, is utterly due to the differences between CC and this thesis regarding the meaning of the evaluation and the end result they are trying to accomplish. CC aims at establishing trust in existing IT products by estimating their level of assurance, while this thesis uses the SFR from CC as a tool to estimate and assign security values to the different security functions in ISs.

CC deals with a TOE, a concept which this thesis also has adopted. This means that no matter how simple or complex the component to be evaluated actually is, it is still in many ways looked upon as shown in the simple structure in Figure 14 below. Users interact with subjects within the TOE, and the TSF, containing most of the security functions, decides which subjects are permitted access to which objects.

Evaluation of the Security of Components in Distributed Information Systems

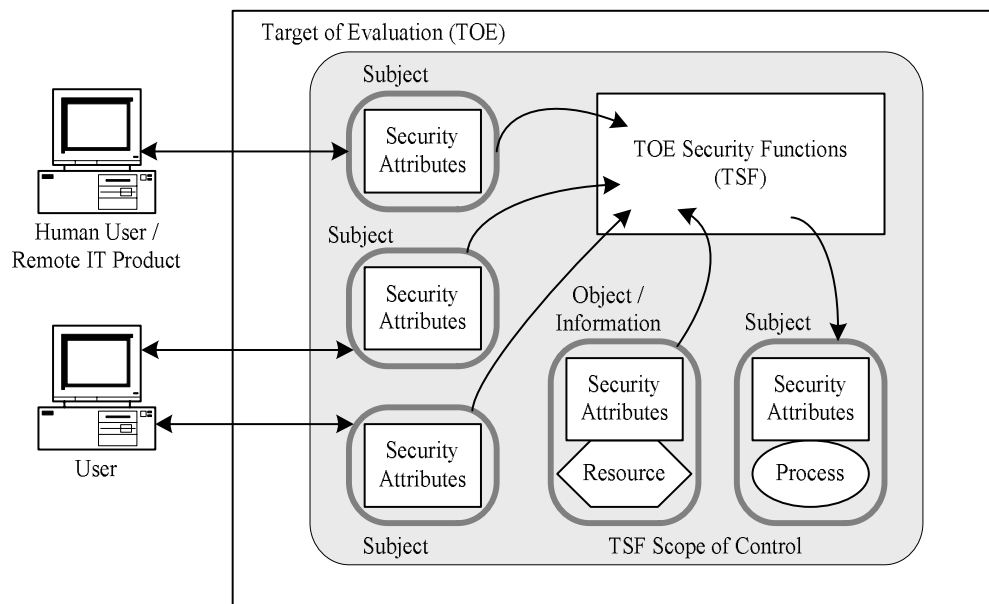


Figure 14: Security functions in a TOE of a system according to CC [13].

In Figure 15 below, a distributed system is the TOE, which leads to a different view. The primary difference is in the number of connections and the main problem when evaluating the TOE lies in the combination of the SFs.

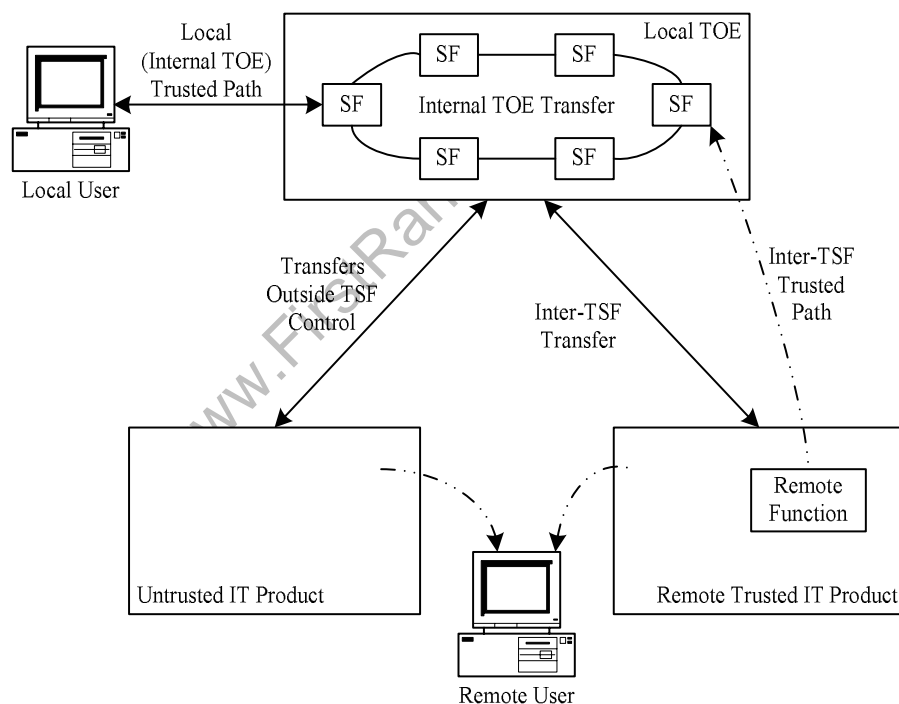


Figure 15: Security functions in a TOE consisting of a distributed system according to CC [13].

The eleven classes of SFs are, as mentioned earlier (in 2.3.4), divided into several families that exist of one or many components. The components consist of elements, i.e. specifically stated security tasks. The classes together with a description of them and an explanation of how they are to be interpreted in this thesis are described below.

The most important alteration of the CC SFR is how to regard the hierarchical ordering of the CC components. This ordering is, as mentioned earlier, based on the fact that the previous components are complete subsets of the following ones. This means that all security functions in the components exist in the hierarchically following ones, and new functionality is added as well. In CC, the foremost meaning of the components is to provide a vocabulary for the description of the security functions in a computer system. However, since this thesis has changed the purpose of the CC components from description to evaluation, these hierarchically ordered components are hardly justified anymore. A better way would be to change the component structure in the following way.

- If the intersection of the security functions of two hierarchically ordered components is of the same type as the conjunction of the components, but with a higher degree of security, both components should be combined into a single one. When evaluating these components in a system using an ST, the only difference is that the component being a subset of the other should be given a lower security value than the other. Components that are combined in this way are marked with an asterisk (*) under the ID-column in Table 4.

For example, in FCO_NRO (non-repudiation of origin), the first component (selective proof of origin) is very similar to the second one (enforced proof of origin). The main difference is that the first component specifies exactly which sort of transmissions that are guaranteed proofs, while that second component guarantees proofs for all transmissions that are being made. Therefore they should be combined into one component, and if the second component was to be found in an ST, the system component would be assigned a higher security value than if only the first component was found.

- If the intersection of the two hierarchically ordered components is of a different type than the complement, the intersection itself should, if possible, form a new component, while the complement forms another. Components that are created in this way have their component number put inside a parenthesis under the ID-column in Table 4.

For example, in FDP_SDI (stored data integrity), the first component (stored data integrity monitoring) is very similar to the second one (stored data integrity monitoring and action). The difference is that the second component will perform an action to accommodate for the loss of integrity. Since this action is of a completely different type than the monitoring, it should be put into a new class. The two original classes would instead become: stored data integrity monitoring, and action due to loss of stored data integrity.

- For families that consist of only a single component, the purpose of the component will be only to introduce symmetry in the component structure, since the meaning and functionality of the family and the single component will be

equivalent. Since the number of families containing only a single component will increase when combining components as mentioned above, it will be easier to manage the SFR structure if these single components were removed. The loss of symmetry in the structure has no negative influence. If a symmetric structuring for some reasons is needed, the family that is without components could be considered as a single component as well. These families are marked with an asterisk (*) under the ID-column in Table 4.

In the following subsections, the classes and their families are explained in detail [12, 14]. It should be noted that even though the elements are not thoroughly discussed in this chapter, they play a great part in the evaluation process, since they are the smallest part of the security functions that the CC component can be broken up into, and it is ultimately in the element that the evaluation of the SFs are being made.

A complete list of all the classes, families and components, after the changes discussed above have been made, can be found in Table 4.

5.2.1. FAU - Security Audit

The 6 families in this class address:

- recognition and responding to security-relevant events and activities (FAU_ARP)
- recording security-relevant events and activities (FAU_GEN, FAU_SEL)
- storing and protecting security-relevant events and activities (FAU_STG)
- review and analysis of security-relevant events and activities (FAU_SAA, FAU_SAR).

This class is used for detecting security events, with help from FDP and FPT.

The Security Audit class is special, since it affects almost every other component in CC. The minimal requirements for audit that every component is supposed to fulfil is specified in CC. Based on these statements a classification of the dependability of each component of this class is specified, depending on how much it is affected by it. For example, if the effectiveness for a certain component to succeed is heavily dependent on that the security audit log-files are controlled at regular intervals, then the dependability will be very high. But if the impact of not monitoring the log-files will not result in severe security failures, then the dependability values will be low.

The audit values will be statically set for each role in a subsystem, i.e. those responsible for controlling and acting on the log files will be evaluated according to their skill. All audit security functions that are controlled by those persons will yield the same effect on the dependent components, but how much that effect will be is decided for each SF.

Components that are dependant on any kind of audit management are marked with an "A" in the last column of Table 4 in the Appendix.

5.2.2. FCO – Communication

The 2 families in this class address:

- proof of origin of transmitted information (FCO_NRO)
- proof of receipt of transmitted information (FCO_NRR).

This class is used for enforcing proof of transmission in communications.

5.2.3. FCS - Cryptographic Support

The 2 families in this class address:

- generation, distribution, access and destruction of cryptographic keys (FCS_CKM)
- operational use of cryptographic keys (FCS_COP).

This class is used for cryptographic protection, with help from FDP and FPT.

Characteristics for deciding the security value are crypto algorithm, key length and key distribution method. Evaluations concerning the practical strength of cryptographic keys are hard to accomplish since there are, for apparent reasons, hardly ever any feedback on when or how cryptographic systems fail [28].

5.2.4. FDP - User Data Protection

The 13 families in this class address:

- security function policies for protection of user data (FDP_ACC, FDP_IFC)
- access control and information flow control functions for protection of user data (FDP_ACF, FDP_IFF)
- authenticity and integrity for protection of user data (FDP_DAU, FDP_ITT, FDP_SDI)
- reuse and rollback for protection of user data (FDP_RIP, FDP_ROL)
- protection of import and export of user data (FDP_ETC, FDP_ITC)
- protection of user data for communications between the TOE and SFs of other trusted IT products (FDP_UCT, FDP_UIT).

This class is used to protect user data when controlling access to information (together with FMT and FPT), monitoring loss of user data integrity when detecting security events (together with FAU and FPT), and protecting transmitted user data in cryptographic protection (together with FCS and FPT).

Access control is the control of subjects, objects and the operations among them; while information flow control rules the subjects, information and operations which causes information to flow to or from subjects. The latter control should be measurable and possible to evaluate on each of the 7 OSI layers (Open Systems Internetwork).

5.2.5. FIA - Identification and Authentication

The 6 families in this class address:

- establishing claimed user identity (FIA_ATD, FIA_SOS, FIA_USB)
- verifying claimed user identity (FIA_UAU, FIA_UID)
- failures when authenticating claimed user identity (FIA_AFL).

This class is used for controlling access to systems, with help from FTA and FTP.

A major characteristic for deciding correctness of validation in FIA_UAU is the False Acceptance Rate (FAR) that states the possibility of a person being falsely granted access when the person should rightfully have been denied access. This rate is sometimes calculated for specific products like bio-scanners and card-readers. The False Rejection Rate (FRR) is a minor characteristic, only affecting the availability. It states the chance of being denied access although access should have been granted.

5.2.6. FMT - Security Management

The 7 families in this class address:

- specifying the management functions of the TOE (FMT_SMF)
- management of TSF data (FMT_MTD)
- management of security attributes of the TOE (FMT_MSA, FMT_REV, FMT_SAE)
- management of the security functions of the TOE (FMT_MOF)
- security roles of the TOE (FMT_SMR).

This class is used for managing TSF data to control access to information, with help from FDP and FPT.

Much like the Security Audit class, the Security Management class is also special and affects many other CC components. For example, if the effectiveness of a certain CC component to succeed depends heavily on that an administrator has updated and set the correct security attributes, then the dependability will be very high. But if the impact on incorrectly set attributes by the administrator will not result in severe security failures, then the dependability values will be low.

The management values will be statically set for each role in a subsystem, i.e. administrators are evaluated according to their skill. All functions that are administrated by those persons will yield the same effect on the dependent components.

Components that are dependant on any kind of security management are marked with an “M” in the last column of Table 4 in the Appendix.

5.2.7. FPR – Privacy

The 4 families in this class address:

- discovery of an individual’s identity by others (FPR_ANO, FPR_PSE)
- association with actions of an individual’s identity by others (FPR_UNL, FPR_UNO).

This class is used for anonymity and identity protection.

This class might be of no interest if the users only are regarded as part of an organization, and therefore do not need their privacy to be protected regarding association with performed actions.

5.2.8. FPT - Protection of the TSF

The 16 families in this class address:

- testing of the TSF mechanisms and data (FPT_AMT, FPT_TST)
- physical and anti-tamper protection of the TSF mechanisms and data (FPT_PHP)
- secure TSF data transfer of the TSF mechanisms and data (FPT_ITA, FPT_ITC, FPT_ITI, FPT_ITT, FPT_RPL, FPT_TDC, FPT_TRC)
- failure and recovery of the TSF mechanisms and data (FPT_RCV, FPT_FLS)
- state and timing of the TSF mechanisms and data (FPT_SSP, FPT_STM)
- reference mediation and domain separation of the TSF mechanisms and data (FPT_RVM, FPT_SEP).

This class is used to protect TSF data when controlling access to information (together with FMT and FDP), monitoring loss of TSF data integrity when detecting security events (together with FAU and FDP), and protecting transmitted TSF data in cryptographic protection (together with FCS and FPT).

5.2.9. FRU - Resource Utilisation

The 3 families in this class address:

- availability of resources (FRU_FLT)
- prioritizing of resources (FRU_PRS)
- allocation of resources (FRU_RSA).

This class is used for controlling use of resources, thus enforcing availability in the system.

5.2.10. FTA - TOE Access

The 6 families in this class address:

- attributes of a user session (FTA_LSA, FTA_TAB, FTA_TAH)
- establishment of a user session (FTA_MCS, FTA_SSL, FTA_TSE).

This class is used for controlling access to systems, with help from FIA and FTP.

5.2.11. FTP - Trusted Path / Channels

The 2 families in this class address:

- trusted communication paths between users and the TSF (FTP_TRP)
- trusted communication channels between the TSF and other trusted IT products (FTP_ITC).

This class is used for controlling access to systems, with help from FIA and FTA.

5.3. Security Evaluation of CC SFs

This step explains and argues on how to interpret the evaluated Security Functions.

A security function of CC often affects other security functions in some way. Some may be specified in CC as being dependent on other functions, and thus their security values will depend on these functions. However, there can also exist a more subtle dependency, not specifically stated in the CC. This might depend much on the given situation, e.g. circumstances regarding the components or system, that decides whether functions are dependable or not.

5.3.1. Interpretation of security values

Once the components have been evaluated according to the CC Security Functions, the question arises about what the values for the SFs really mean in terms of IT security.

- One possible solution is to present these values as the final result. Although they perhaps are hard to interpret – especially for one who is a layman to CC and it's

Security Functions, they may to some people be more precise and useful than any other values.

- Another solution is to traverse the values for the SFs upwards, yielding results for the more general security functions, and finally at the top level there will be measurable security values for the 11 classes of CC SFR.
- A third solution is to translate the values into a more, in the world of IT security, accepted and recognizable terminology. The two most used terminologies are CIA and PDR. They cover entirely different aspects of security, the former is on how information assets may be compromised and the latter covers abilities required to maintain the system security.

CIA – Confidentiality, Integrity and Availability

Which CC components and families that are judged to belong to which of the CIA-categories, are marked in the third column of Table 4. How the SFR map to CIA is also visualized in Figure 18. The table and the figure are found in the Appendix.

PDR – Prevent, Detect and React

Which CC components and families that are judged to belong to which of the PDR-categories, are marked in the fourth column of Table 4. How the SFR map to PDR is also visualized in Figure 19. The table and the figure are found in the Appendix. Prevent is further divided into storage (S), transportation (T) and/or execution (E) to specify the area of prevention. If applicable, this is marked in the fifth column of Table 4.

The different SFR families and classes have been thoroughly analyzed and categorized both according to the CIA and the PDR terminologies. Some families belong to more than one category. This categorization has also been done on the component level and for those families where the category of one or more of the components differ from that of the family, the category of the family is either a combination of the categories, or the one that best fits to the family. The purpose of the categories is to reflect what characteristics that are applicable or valid for each family, and perhaps even component.

There is really no restriction in how to represent the security values. One of the strategies discussed above may be used, or any combination of them if that is found useful.

Sometimes several different kinds of security values are needed for the same evaluation in order to cover different aspects of IT security.

There should be a possibility to multiply the values with a weighting-matrix in order to specify properties in the system that are more important (or less important) for the specific evaluation. This matrix has values ranging from zero (>0) to one (1) or a NULL-value. A one means that the property is fully regarded, while lower values means the property is proportionally less regarded. A NULL-value means that the selected security function should be entirely neglected, (e.g. a matrix filled with only ones (1) does not affect the resulting security values whatsoever).

5.3.2. Calculations of security values

The following steps explain how to reach meaningful security values once the evaluation of a system component according to CC SFs has been made.

1. Choose which of the above explained way(s) to represent the security values.
2. Calculate mean values of the above chosen type(s) for every family.
3. If there are CC components that should be prioritized before others, their security values should be multiplied with a weighting matrix to reflect this prioritization as explained above.
4. NULL-values have no effect whatsoever during the calculations and should simply be ignored.
5. Calculate mean values for every class, or corresponding concept depending on the chosen representation.

Example 1

To exemplify the steps above, security values for one of the 11 CC classes are calculated. The system component for this example is the Sony FeliCa Contactless Smart Card [32], and how the security values are estimated for this component is explained in Example 2 (see 5.4). For those families which have more than one component (according to Table 4), the value of the family is calculated as the mean value of the components.

The resulting security value for the class FPT (Protection of the TOE Security Functions) is then calculated as the mean value of the 16 components (grey rows in Table 1). The security value for the specific class would be:

$$\frac{0.95 + 0.92 + 0.85 + 0.80 + 0 + 0 + 0 + 0 + 0.91 + 0 + 0 + 0}{11} \approx 0.403.$$

If the resulting security value is better represented according to CIA, first check the fourth column of Table 1 to see what category each component and family belong to. Note that one family that represent more than one category is divided equally among the categories. The three security values for the FPT-class of the smart card would according to Table 1 be the following:

$$C: \frac{0.85 + \frac{1}{2} \cdot 0 + \frac{1}{3} \cdot (0 + 0 + 0)}{1 + \frac{1}{2} + \frac{3}{3}} = 0.34$$

$$I: \frac{0.80 + 0.91 + 0 + \frac{1}{2} \cdot (0.95 + 0.92 + 0 + 0) + \frac{1}{3} \cdot (0 + 0 + 0)}{3 + \frac{4}{2} + \frac{3}{3}} \approx 0.441$$

$$A: \frac{\frac{1}{2} \cdot (0.95 + 0.92 + 0) + \frac{1}{3} \cdot (0 + 0 + 0)}{0 + \frac{3}{2} + \frac{3}{3}} = 0.374$$

Note that compared to the general security value above, the confidentiality (C) and the availability (A) security values are worse. However, the integrity (I) value is better than the general security value.

Component	Component description	Value	CIA
FPT_AMT	Underlying abstract machine test	0.95	IA
FPT_FLS	Fail secure	0.92	IA
FPT_ITA	Availability of exported TSF data	NULL	A
FPT_ITC	Confidentiality of exported TSF data	0.85	C
FPT_ITI	Integrity of exported TSF data	0.80	I
FPT_ITI.1	Inter-TSF detection of modification	0.80	I
FPT_ITI(2)	Correction of modified Inter-TSF data	NULL	I
FPT_ITT	Internal TOE TSF data transfer	0	CI
FPT_ITT.1	Basic internal TSF data transfer protection	0	CI
FPT_ITT(2)	Transfer separation of TSF data	NULL	CI
FPT_ITT.3	TSF data integrity monitoring	0	I
FPT_PHP	TSF physical protection	0	CIA
FPT_PHP.1	Passive detection of physical attack	NULL	CIA
FPT_PHP(2)	Notification when detection of physical attack	NULL	CIA
FPT_PHP.3	Resistance to physical attack	0	CIA
FPT_RCV	Trusted recovery	0	IA
FPT_RCV.1*	Recovery	0	IA
FPT_RCV(3)	No undue loss after recovery	NULL	IA
FPT_RCV.4	Function recovery	NULL/0.9	IA
FPT_RPL	Replay detection	0.91	I
FPT_RVM	Reference mediation	0	CIA
FPT_SEP	Domain separation	0	CIA
FPT_SEP.1	TSF domain separation	0	CIA
FPT_SEP.2	SFP domain separation	NULL	CIA
FPT_SEP.3	Complete reference monitor	0	CIA
FPT_SSP*	State Synchrony Protocol	NULL	IA
FPT_STM	Time stamps	NULL	CIA

Component	Component description	Value	CIA
FPT_TDC	Inter-TSF TSF data consistency	NULL	I
FPT_TRC	Internal TOE TSF data replication consistency	NULL	I
FPT_TST	TSF self test	0	I

Table 1: Security values for components in grey columns, and families in white columns, of the FTP-class (Protection of the TOE Security Functions).

5.4. Map CC Security Functions to evaluated component characteristics

In this section, the process of evaluating the system components according to their security properties and how this evaluation relates to the security functions of the Common Criteria, is explained.

The appropriate security functions that are desired for a specific component could be found reading a PP or ST. However, these documents do not give any information about the security values of the SFs for that component. The reality is that most of these, for IT security so essential security characteristics, are hard to evaluate. It is very challenging to make an appropriate and objective judgment of the security of the component. Even basic security methods that are commonly used today, and regarded as having a high security assurance, may in fact not be reliable at all. Too many aspects of deployed measures are dreamed up in the absence of any empirical evidence [18]. For example, one commonly used security method is to freeze the accounts of the users when an incorrect password has been entered three times in a row. Where is the analysis that guarantees the soundness of the exact value three (3) for this parameter? Could it not instead be so that the number ten (10) is almost as secure, but gives the administrator more time for relevant security work, resulting in a better overall security for the system environment?

A way to decide the correct security values for components could be to practically test them (e.g. penetration tests for firewalls, see 2.3.8). However, testing could be difficult to perform objectively and independently. Another way could be to fill in surveys with multiple-choice questions regarding the system. The accuracy of the evaluation will of course suffer, but the query forms will be easy to create and the results easy to fill in and evaluate. The ideal way is of course if all security-relevant characteristics could be found and measured for all of the components, which will result in an objective and just security value for each and every evaluated component.

There are four different smaller steps that together form this mapping of SFs to a component.

1. Select those SFs that are relevant for the component. Here a PP may be of good use.
2. Add those SFs that the SFs selected in step 1 are depending on. Those SFs that are not regarded at all are given the NULL-value, as they are not considered important or valid for the given component.

3. Determine which SFs from step 2 that actually exist and are covered in the specific component. Here an ST may be of good use, if one exists for the component. Those SFs from step 2 that are not covered in step 3 are assigned the zero value to signify that here resides a security void.
4. Evaluate all existing SFs from step 3 and assign values, in the range 0 to 1, representing the strength of the implementation of the SF.

It is important for the security of a DIS that components with one or more functions containing a zero are combined with components that cover these gaps, resulting in better values for those security functions. If a system with zero-valued functions is put into use, there will be great security vulnerabilities in that system.

Observe that a component may also affect the information system negatively. For example, if some really time-consuming methods are used to verify the identification of a user, the availability of the entire IS will slightly worsen. Protections may cause other functionality to fail, or at least to make the actual work harder and more time-consuming to perform. Another aspect of this problem that is focusing on availability was observed by John Ousterhout, who claimed that security was “*anti-CS*” (computer science) because it was getting in the way of people getting things done [18].

Example 2

As an example on how to map SFs to a component, look at the FPT class (Protection of the TSF) for the same Sony FeliCa Contactless Smart Card as in Example 1 [32], following the four steps mentioned above.

According to a PP [33], and some additional reasoning about smart cards [2], the relevant SFs were chosen to be the ones marked in the first two columns of Table 2.

After checking for dependencies in [12], we add TST, since RCV.1 is dependant on that component. Also, TST depends on AMT, and ITT.3 on ITT.1, but these two components already exist in the set. When summarizing, the relevant SFs for a smart card turned out to be the following: AMT, FLS, ITC, ITI.1, ITT.1, ITT.3, PHP.3, RCV.1, RPL, RVM, SEP.1, SEP.3 and TST.

When the Smart Card is evaluated according to these requirements, the ST indicates the SFs of the product, as shown in column three in Table 2.

Null- and zero-values are now easily determined when comparing what SFs the smart card is supposed to contain to what SFs it actually do contain. The other SF's security values, ranging from above zero to one, are harder to estimate. For the sake of this example, these values were, without any further proofs or motivations, found to be the ones shown in the last column of Table 2.

PP	Gollman	ST	Component	Component description	Value
X	X	X	FPT_AMT	Underlying abstract machine test	0.95
	X	X	FPT_FLS	Fail secure	0.92
			FPT_ITA	Availability of exported TSF data	NULL
	X	X	FPT_ITC	Confidentiality of exported TSF data	0.85
X		X	FPT_ITI.1	Inter-TSF detection of modification	0.80
			FPT_ITI(2)	Correction of modified Inter-TSF data	NULL
X	X		FPT_ITT.1	Basic internal TSF data transfer protection	0
			FPT_ITT(2)	Transfer separation of TSF data	NULL
X			FPT_ITT.3	TSF data integrity monitoring	0
			FPT_PHP.1	Passive detection of physical attack	NULL
			FPT_PHP(2)	Notification when detection of physical attack	NULL
X	X		FPT_PHP.3	Resistance to physical attack	0
X			FPT_RCV.1*	Recovery	0
			FPT_RCV(3)	No undue loss after recovery	NULL
		X	FPT_RCV.4	Function recovery	NULL/0.9
X		X	FPT_RPL	Replay detection	0.91
X			FPT_RVM	Reference mediation	0
X			FPT_SEP.1	TSF domain separation	0
			FPT_SEP.2	SFP domain separation	NULL
X			FPT_SEP.3	Complete reference monitor	0
			FPT_SSP*	State Synchrony Protocol	NULL
			FPT_STM	Time stamps	NULL
			FPT_TDC	Inter-TSF TSF data consistency	NULL
			FPT_TRC	Internal TOE TSF data replication consistency	NULL
(X)			FPT_TST	TSF self test	0

Table 2: Lists all components of the FPT-class, in which papers they were found to be relevant for the smartcard-component and their estimated security values. Note that some components are merged according to 5.2.

5.5. Evaluation of Systems made up of Components

This section explains how several instances of evaluated components are put together into an evaluation of a subsystem, which afterwards becomes a new component itself. It also describes how to calculate and estimate new security values for the subsystem by combining the security values of the individual components.

A great problem exists in the fact that the CC functions are assembled by combinations of system components and it is a difficult task to single out exactly what functions a specific component affect. There is also a problem in deciding how one component affects the others, since they could be dependent on each other.

The larger the system to be evaluated, the more abstract will the evaluation become, since the security evaluation of the system is a combination of the evaluations of all of the components. It will be possible to “zoom in” on a specific part of the system to get a more accurate view of this part and its security values. It will also be possible to “zoom out” from a system focus in order to get a better overview of the entire system. This method of zooming causes a kind of prioritization between overview and accuracy in the model.

5.5.1. Components represented by graphs

The subsystems and how they work together in achieving their security goals could be modelled using a graph, with nodes representing the different components with a given value for the security in that component and links representing the probability of that the specific link is being followed. Different graphs could be used for different scopes depending on the needs of the evaluator (see 4.2). A simple example of this, for three computers that are connected to the Internet through a firewall via a hub, is shown in Figure 16. Security indicators (SI) are stated for each node, as an estimation of the security value.

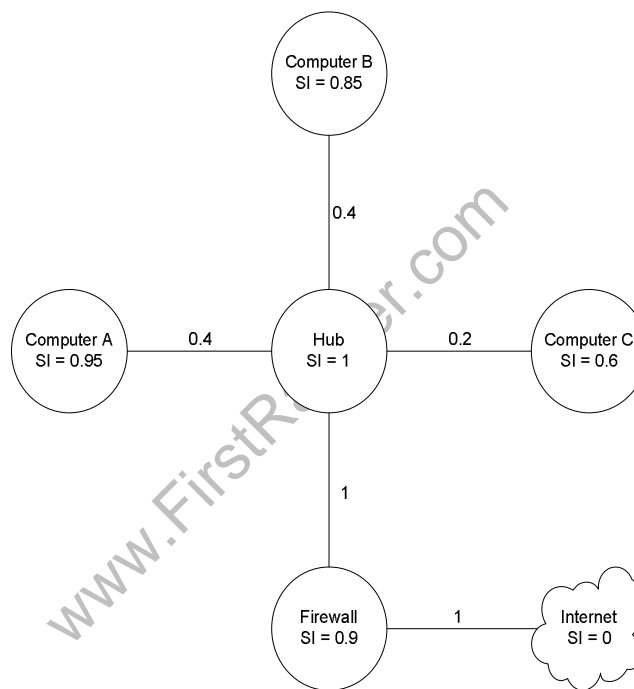


Figure 16: Simple node structure.

5.5.2. Merging of the components

To get a more simplified view of the system, more abstract system views can be created by combining nodes and recalculating the security values for the nodes. The idea for this method is also described in [22]. Different calculation methods are being used depending on the specific situation.

However, as the nodes are being combined, the specification of the model becomes blurred since the evaluations regard several combined components instead of only one. Therefore it should be possible to undo the merging and return to the original representation of the model. This could for example be done to check what component in a system that is the most efficient to replace with a more secure one in order to achieve a better overall security rating.

5.5.3. Mathematical functions

Since the proposed security values are of a probabilistic nature, and ranges from 0 to 1 (see 5.1), certain types of mathematical functions are needed when combining the different components. This is needed in order to achieve meaningful values (i.e. values between 0 and 1) after the combination as well as prior to it.

For example, a maximum-function will be appropriate, but a simple addition will not be suitable since the result may be higher than one (1). It is an intricate challenge to research these mathematical functions if they are to capture the actual security relations between components.

Here follows some examples of functions that possibly could be used for the integration of components into systems. They were developed during the thesis project and similar functions could also be found in another paper, but unfortunately, the mathematical soundness, correctness and reliability of those functions were not established in that paper [22].

In the following text, the term Security Indicators (SI) is used to denote the security values. It is regarded to be a vector, so a maximum function between two different SI would have to compare several scalar values, not only a single one. Furthermore, only binary combinations have been covered in the examples below, but it should of course be possible to combine more than two components as well. Caution should be taken when the components to be combined come from different categories, e.g. when combining two “cooperative” components with one “coexisting”. Then it is also important to decide the order in which to perform the combination, since the result will differ depending on this.

Cooperative

If the security functions in the components are working together, it should result in a positive effect on the security values in the combined component. How much this increase should be is primarily based on how independent these two components are on each other.

For example, if a software firewall (SI_1) was to be combined with a virus protection application (SI_2), the resulting SI could be calculated in the following way:

$SI = \max(SI_1, SI_2)$. Note that this function works for components with very overlapping SFs.

If the components are completely independent, the mathematical function for a union can be used. $SI = SI_1 + (1 - SI_1) \cdot SI_2 = SI_1 + SI_2 - SI_1 \cdot SI_2$.

These both functions could be combined into a single one, if the level of dependability could be measured or estimated. A dependability-parameter, x , may be used to specify the rate of to what degree the both components cover the exact same security functions.

$$SI = x \cdot \max(SI_1, SI_2) + (1 - x) \cdot (SI_1 + (1 - SI_1) \cdot SI_2).$$

Coexisting

When combining components that have a negative effect on the security values, other functions have to be used. For overlapping SFs, the lowest value for the SF is chosen. Perhaps a small adjustment of the security value is in place to account for number of SFs being overlapped.

For example, when combining multiple users (SI_i) of a single computer, the combination could be: $SI = \min(SI_1, \dots, SI_N)$. Note that this is for components with completely overlapping security functions. For combinations of independent components, the same reasoning as in the “cooperative” relation could be used to deduct similar functions.

The same mathematical expression as the one above is referred to as “Weakest link” in [22] and is described as singling out the component with the least security. The relationship exists between sibling components that are all vital to the success of their common security task. There is also a possibility to weight the components to specify the level of impact each component has for the common task.

Counter effective

This relation covers issues where the SFs in one component somewhat negates the SFs in another component.

For example, if a firewall (SI_1) was to filter packets that had its content protected with cryptography (SI_2), the functionality of the firewall would suffer since it no longer could search the packets for illicit data. The security values for those affected SFs could be calculated in the following way: $SI = SI_1 \cdot SI_2$.

Perplexing

The SI of a component will become more and more inaccurate as it is combined with components with less trustworthiness. This is due to the fact that components that are not trusted will inflict untrustworthiness to trusted components as well.

For example, if a workstation of local user (SI_1) was connected to a workstation of an unknown user (SI_2) with a trustworthiness-rating (TW), the resulting SI could be:

$$SI = \min(SI_1, SI_2 \cdot TW).$$

Example 3

An example of this step is the combination of the smart card of previous examples, with a card reader. The security value for the smart card (from Example 1) is in the third column of Table 3 while the estimated security values of the card reader are in the fourth column. If the functions were regarded as “cooperative” from the above method, with maximum

Evaluation of the Security of Components in Distributed Information Systems

dependability (for mathematical convenience), then the security values of these two components combined are calculated using the max-function on the evaluated families of the two components. The security value for the new component is calculated in the following way:

$$\frac{0.95 + 0.92 + 0.40 + 0.85 + 0.80 + 0.46 + 0.70 + 0 + 0.91 + 0 + 0 + 0 + 0}{13} \approx 0.461.$$

This result is slightly better than the one received in Example 1. The increase is mostly due to the fact that some security holes are covered in the card reader even though new holes are introduced.

Component	Component description	Smart	Card	Security
FPT_AMT	Underlying abstract machine test	0.95	0.90	0.95
FPT_FLS	Fail secure	0.92	0	0.92
FPT_ITA	Availability of exported TSF data	NULL	0.40	0.40
FPT_ITC	Confidentiality of exported TSF data	0.85	0.85	0.85
FPT_ITI	Integrity of exported TSF data	0.80	0.72	0.80
FPT_ITI.1	Inter-TSF detection of modification	0.80	0.72	
FPT_ITI(2)	Correction of modified Inter-TSF data	NULL	NULL	
FPT_ITT	Internal TOE TSF data transfer	0	0.46	0.46
FPT_ITT.1	Basic internal TSF data transfer protection	0	0.92	
FPT_ITT(2)	Transfer separation of TSF data	NULL	NULL	
FPT_ITT.3	TSF data integrity monitoring	0	0	
FPT_PHP	TSF physical protection	0	0.70	0.70
FPT_PHP.1	Passive detection of physical attack	NULL	NULL	
FPT_PHP(2)	Notification when detection of physical attack	NULL	NULL	
FPT_PHP.3	Resistance to physical attack	0	0.70	
FPT_RCV	Trusted recovery	0	NULL	0
FPT_RCV.1*	Recovery	0	NULL	
FPT_RCV(3)	No undue loss after recovery	NULL	NULL	
FPT_RCV.4	Function recovery	NULL/0.	NULL	
FPT_RPL	Replay detection	0.91	NULL	0.91
FPT_RVM	Reference mediation	0	0	0
FPT_SEP	Domain separation	0	0	0
FPT_SEP.1	TSF domain separation	0	0	
FPT_SEP.2	SFP domain separation	NULL	NULL	
FPT_SEP.3	Complete reference monitor	0	0	
FPT_SSP*	State Synchrony Protocol	NULL	NULL	NULL
FPT_STM	Time stamps	NULL	NULL	NULL
FPT_TDC	Inter-TSF TSF data consistency	NULL	0	0

Component	Component description	Smart	Card	Security
FPT_TRC	Internal TOE TSF data replication consistency	NULL	NULL	NULL
FPT_TST	TSF self test	0	0	0

Table 3: Security values for components (white columns) and families (grey columns) of the FTP-class (Protection of the TOE Security Functions). The values are the security estimates from the smart card, the card reader and the resulting values from the combination of the two.

www.FirstRanker.com

6. Discussion

This chapter sums up and considers the work and results presented in the two previous chapters.

6.1. Security Evaluation Framework

The central part of chapter 4, and of the whole thesis, is the security evaluation framework. It aspires to be able to handle all possible aspects that may affect the security of a DIS, as well as divide the evaluation process into different parts, making it less complex. The framework generates a modelling technique used to model an information system, and an evaluation method to evaluate the model. However, it is possible that the framework makes the problem space too wide, and that a more narrow and restricted problem space will yield better results.

As the framework divides the evaluation into smaller blocks, the evaluation process becomes more flexible and manageable. The parts of the evaluation process are solved using modules, created and configured for this purpose. By combining these modules, it becomes easier to adapt the framework to fit specific aspects and needs of the evaluator. Arguments may arise whether the different problem aspects easily can be divided into smaller problems that are handled by the modules. There is nothing yet that contradicts this conclusion, but because of the abstract state of the thesis at this point, no proofs of the opposite may be given. Also, this module-thinking in the framework may perhaps turn out to be more restricted and limited than what was hoped for in order for the evaluation to reach meaningful results.

The concept of scope gives the evaluator means to focus the evaluation to specific demands and needs. This has its obvious reasons, as the desired evaluation differ depending on the situation. Unfortunately, it might not always be an easy task to facilitate the partition of the distributed information systems into scopes.

The component structure defines the way that a system can be divided into its components, and components can be merged into a system. When a component can not be divided any further, measurable attributes should be extinguishable and yield security values to the system component. There may be problems in how to partition a component into new components as the boundaries not always are apparent.

The component library provides means to store instances, keeping standard components and reuse already evaluated components. This could save a lot of work and time when evaluating large systems. This assumes that evaluated components not are too dependable on the system they are presently in, so that the evaluation may be valid for other systems as well. This assumption might very well be incorrect. There are also no proposed general ways to combine components, making the process of combining components a demanding and time-consuming task.

The user is part of the evaluation, which is necessary because it is often the human threats that are the most serious ones. However, how to measure the user and to fit its result into the whole evaluation is not an easy problem to solve.

The ontology structure is in many ways better than the original tree-structure, as it, when using another categorization than CIA, solves all the problems with the tree-structure from Figure 1, that were mentioned earlier (3.1). The ontology structure gives a dynamical and versatile way to structure the attributes, components, security functions and the security values together with their different relations. The ontology divides the objects into taxonomy-like partitions, and also helps in defining a common terminology.

This structure is in many ways better than the original tree-structure, which however may for some be regarded as more clear since it is simpler and the structure of ontologies is perhaps unknown.

6.2. Evaluation method

The main result of this thesis, the core of the security evaluation framework, is the evaluation method. It focuses on technological components, as this is the fundamental step of the evaluation, according to the security evaluation framework. Other aspects than technological are explained more generally, but evaluations on components of these non-technological (e.g. organizational) components should be simple to develop using similar techniques. Five different steps were developed to cover each different step of the method.

The first step is the security values and the metric that are needed for the purpose of the evaluation and to the ability to compare different systems. Different security values were proposed; securability, security level and risk level, all covering different aspects and specifications of security. Different values make the evaluation more versatile as several values may be chosen depending on the situation. These values were given a probabilistic meaning, but the exact meaning of this probability value, what exactly it shows and how it is helpful, are not given at this point. Also that only the endpoints of the metric has an associated meaning might be troublesome.

As a second step, the Security Functional Requirements of the Common Criteria were adopted as a base for the evaluation. Also the aspects of a TOE and some other terminology were obtained from CC. The families and components of the eleven classes of the SFR were changed in order to better fit with the thesis. However, there may exist a better basis for the evaluation than the SFR, which was not discovered in the thesis, or the development of a better foundation, that is specified with this evaluation in mind, could perhaps have been made. Still, since CC is so widely used and renowned in the world of computer security, it should be good enough for this purpose. Also objections may arise for the fact that large and complex distributed information systems are described with only the simple terms and modelling of a TOE.

In the third step explanations are given on how to interpret the values that have been given to the evaluated security functions. One idea is to calculate the values for the

extensive eleven SFR-classes. Alternatively, a thorough mapping has been made that relates the SFR of both the component, family and class level to CIA and PDR respectively. A weighting matrix is proposed, making it possible to prioritize some aspects of the security to accommodate for certain needs and situations. An algorithm for reaching the final security values is exemplified for a Smart Card to show how this step of the evaluation can look like. There might of course be other security values or ways to illustrate security that are better than these suggested ones. Also, there might arise objections toward the mapping of the SFR to CIA and PDR, since parts of the mapping are not very intuitive and clear as it is the interpretation of the formulations of the security functions that decides the mapping.

How component characteristics should be reflected to the Security Functional Requirements of CC is explained in the fourth step. Sometimes PPs and STs can be used to extract which functions that are deemed relevant for certain products. An algorithm of the process is defined and exemplified, using a Smart Card, to make this step easier to understand. One problem is that even if a PP or an ST might state what functions that are needed for a product, they do not rate those functions. It is this security rating that is the biggest problem of this step, the fair and just evaluation of the attributes of the components.

The fifth and last step of the process states how the evaluated components can be used to get a meaningful evaluation of the combined components. This thesis suggests that a graph can be introduced, where the evaluated components are represented as nodes, and the links are assigned values of probability. Then by merging nodes, yielding security values to the new nodes, an evaluation of the combined components has been made. Since the evaluation of a single component gets less specific in an evaluation of the component combined with other components, it should be possible to undo the evaluation of combined components and receive the original evaluation. Also, some mathematical functions are proposed to show some initial attempts on how to combine the security values of the components. An example of how to combine a Smart Card to a card reader is given to clarify this step. One major problem is that it is very hard to know how the security functions of different components depend on each other, and therefore also affect each other. Also, complaints towards the soundness and validity of the mathematical functions used to combine the components may be raised.

7. Conclusions

This chapter summarizes the entire thesis and suggests some possible ways to improve it.

7.1. Summary

The computerized world of today puts an increasingly high demand on the development of IT security products. To be able to control the security in these IT security products, as well as in large distributed information systems, a sort of estimation or evaluation has to take place. That is the only way to produce a measure, and measures are needed to rate security products as well as information systems, and compare them to similar products or systems. This thesis aims at finding a method to evaluate distributed information systems.

The security evaluation framework is the central part of the thesis as it establishes a wide, unspecific space where all the different aspects that affect the security in distributed information systems could be modelled. The problem space is divided into smaller problems, decreasing the complexity. The sub-problems are then solved in the different modules, the blocks that together build up the framework. The modules introduce flexibility and means to customize the evaluation framework to fit specific needs. This customization is also enhanced by the possibility for the evaluator to choose the scope of the evaluation and thereby deciding which aspects the evaluation should focus on. After the modules have been configured to suit the evaluator, the framework generates a modelling technique used to model an information system, and an evaluation method to evaluate the model.

The distributed information systems are modelled using a component structure that makes it possible to divide the system into smaller and smaller components. Also the opposite is of course possible, combining components into a system. When components have been evaluated, they are to be stored in the component library, where also evaluated standard components are put, in order to save time and work in later evaluations.

The ontology structure gives a dynamical and effective way to structure all the objects in the security evaluation framework and their jointed relations. It also divides and partitions the objects of the framework more strictly than, for example a tree-structure does, which gives a more precise and detailed structure. One additional advantage of the ontology is that it helps in defining a common terminology, which might be helpful in later collaborations in the areas of computers and security.

The main part of the thesis is the evaluation method. It focuses on technological components, and uses the Security Functional Requirements (SFR) of the Common Criteria as a basis. Different security values were proposed, having a probabilistic meaning but different aspects. The security characteristics of the system components are evaluated by mapping security values to the SFR representing the specific components. An evaluation of a system is performed by combining evaluations of the components that contributes to the system. A greater meaning of the security values could also be obtained

due to the extensive mappings from SFR to more meaningful security categorizations like CIA and PDR.

7.2.Future Work

Here follows some ideas to further improve the security level evaluation method presented in this thesis.

7.2.1. Improved security evaluation

Develop the evaluation, fine-tune it further and create more specific instances for the knowledge-base.

A trustworthiness value might be assigned to the component, simulating the level of trust that resides in the component. Also some kind of Critical Time value could be introduced and used to approximate the time from an attack to a reaction, via detection. This value would estimate how much damage an attack may cause before it is attended to. There may even be a negative effect of trusted components, because they tend to foster complacency, making a user less willing and prepared to deal with security problems [28].

A sensitivity analysis could be introduced to assess the impact of variations to the individual components [22]. Thus, a sensitivity analysis should be able to show which component to improve for the biggest pay-off in increasing the overall security. It should also function as a sanity check for the modelling effort in order to verify the correctness of the evaluation.

7.2.2. Attack model

Map an attack model to point out and highlight vulnerabilities in the IS, or directly in the security functions. This can be regarded as one of the scopes from 4.2.

This could be done in parallel with the framework method. The obvious benefit from this model would be that flaws and vulnerabilities in a system are easier to detect when attacks targeting certain points or components in a system are regarded.

State diagrams, perhaps influenced from automata-theory may be used to simulate the different states in a system that leads to vulnerabilities, and what is needed in order to jump to another state.

7.2.3. Distribution for the attacks

By introducing a number distribution for the attacks, a better understanding and meaning of the model could be achieved.

Poisson random distribution may be used as a method to show for example an approximation of how many attacks that occur in a certain time interval. However, nothing says that attacks are random over time, or independent among each other, which

are two requirements that must be met in order to use the poisson distribution. Ignoring this fact, letting the poisson distribution model attacks to the system, a security value could be an estimate of the probability for each attack to be unsuccessful, either by countermeasures or protection. A parameter in the poisson distribution decides the frequency. An example of what the distribution may look like is given below in Figure 17, with the parameter λ set to 5. The mathematical expression of a poisson distribution is also found below.

The main reason for the need to simulate the attacks is that since security values have been assigned a probabilistic meaning, simulated attacks would give a greater understanding to this meaning. The security metric and values could become more precise at actual validation, and testing of their statistical correctness could be performed using this attack distribution.

There might also be other distributions or functions that can simulate attacks better than the poisson distribution.

$$p(X = x) = \frac{\lambda^x}{x!} e^{-\lambda} \text{ where } \begin{cases} x = 0, 1, 2, \dots, n \\ \lambda > 0 \end{cases}$$

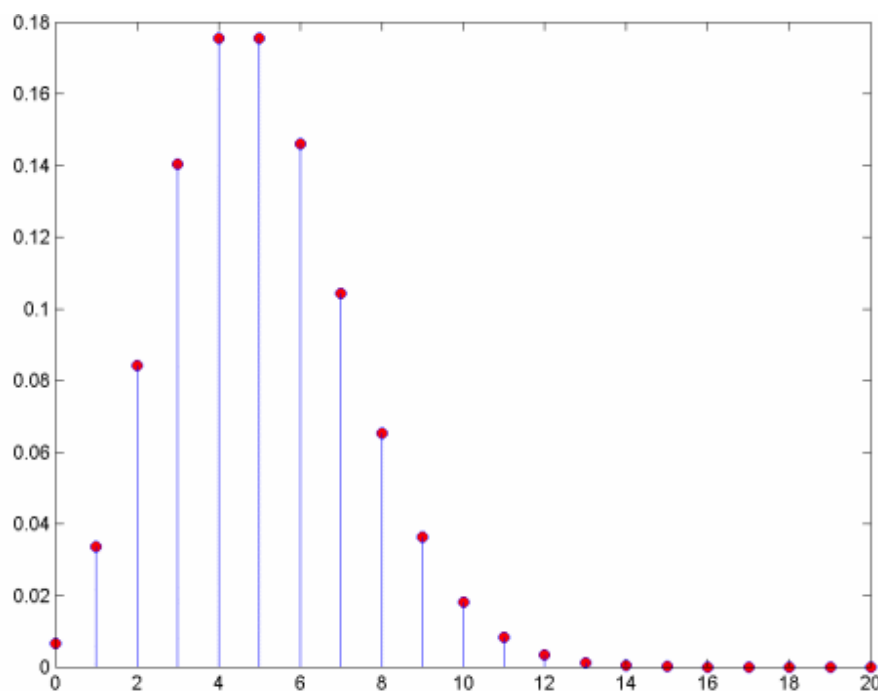


Figure 17: Poisson distribution with parameter $\lambda = 5$.

7.2.4. Practical system testing

Perform tests on real components and test the evaluation process and see how it works out in practise. Then use the data received in the tests to develop the evaluation process further and in more detail. With a combination of the top-down approach used in this

thesis, and a bottom-up approach received from practical testing, the model and evaluation process have great opportunities to evolve further.

7.2.5. Ontology development

Develop the ontology further. A thorough analysis of the current computer security, leading to a well-defined standard computer security ontology, is needed to make it possible for the scientists from all the different specializations of the computer security fields to cooperate and work together towards a common goal.

7.2.6. Automated security calculations

Develop a program which uses the knowledge-base of the ontology to automatically calculate the security values, using specified relations and mathematical functions. Components would be created as instances and as the system components are specified, yielding in instances in the knowledge base, the program should be able to calculate an estimate of the security level for the given system.

7.3. Acknowledgements

I would like to thank Jonas Hallberg and Amund Hunstad for helping and supporting me during the production of this thesis, and Tobias Horney for some valuable help regarding ontologies. I would also like to thank Elinor Bolyos for the support and everlasting confidence she always have in me.

References

- [1] Lord Thomson, W. (also known as Lord Kelvin), *Popular Lectures and Addresses (1891-1894)*.
- [2] Gollman, D. (1999), *Computer Security*, John Wiley & Sons.
- [3] Mead, N. R., McGraw, G. (2003), *From the Ground Up: The DIMACS Software Security Workshop*, IEEE Security & Privacy, March/April 2003, pp 59-66.
- [4] Hallberg, J., Hunstad, A. (2001), *Towards quantifying computer security: System structure and system security models*, position paper for Workshop on Information Security System Scoring and Ranking.
- [5] ACSA (2002), *Proceedings: Workshop on Information Security System Scoring and Ranking*, Applied Computer Security Associates.
- [6] Hunstad, A., Hallberg, J. (2002), *Design for securability – Applying engineering principles to the design of security architectures*, ACSA Workshop on the application of engineering principles to system security design, Boston, November 6-8 2002, Linköping FOI 2002, FOI-S--0721--SE.
- [7] Hallberg, J., Hunstad, A. (2002), *Modeling of Distributed Systems Focusing on IT Security Aspects*, FOI-R—0712-SE, FOI, Linköping, Sweden.
- [8] Stjerneby, A. (2002), *Identification of Security Relevant Characteristics in Distributed Information Systems*, Master's Thesis, LiTH-ISY-EX-3278-2002, University of Linköping.
- [9] Eloff, von Solms (2000), *Information Security Management: An Approach to Combine Process Certification And Product Evaluation*, Computers & Security, Volume 19, Issue 8, 2000, pp 698-709.]
- [10] ITAA, *Information Technology Association of America*, <http://www.ita.org> (acquired 22 October 2003).
- [11] Abrams, M., Joyce, M. (1995), *Trusted System Concepts*, Computer & Security, Issue 14 1995, pp 45-56.
- [12] CC (1999), *Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements*, version 2.1, August 1999, CCIMB-99-032.

- [13] CC (1999), *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model*, version 2.1, August 1999, CCIMB-99-031.
- [14] Bottomly, R. (2002), *CC Part 2: Security Functional Requirements*.
- [15] IACS, *Information Assurance and Certification Service*, <http://www.cesg.gov.uk/site/iacs/> (acquired 22 October 2003).
- [16] Olthoff, K. (2000), *Thoughts and Questions on Common Criteria Evaluations*, 23rd National Information Systems Security Conference, 2000.
- [17] Ware, W. H. (1997), *New vistas on info-system security*, Chapman & Hall.
- [18] Smith, S.W. (2003), *Humans in the Loop: Human-Computer Interaction and Security*, IEEE Security & Privacy, May/June 2003, pp 75-79.
- [19] Shapiro, J. S. (2003), *Understanding the Windows EAL4 Evaluation*, IEEE Security & Privacy, February 2003, pp 103-105.
- [20] Whitmore J. J. (2001), *A method for designing secure solutions*, IBM Systems Journal, Armonk 2001, Volume 40, Issue 3, pp 747-768.
- [21] Kruger, R., Eloff, J. (1997), *A Common Criteria framework for the evaluation of Information Technology system security*. IFIP TC11 12, international conference on Information Security (SEC '97). Copenhagen, Denmark, Chapman & Hall.
- [22] Wang, C., Wulf, W. (1997), *A framework for security measurement*, NISSC97.
- [23] Jonsson E., Strömberg L., Lindskog, S. (1999), *On the Functional Relation Between Security and Dependability Impairments*, Proceedings of the New Security Paradigms Workshop, pp 104-111, Ontario, Canada, September 22-24, 1999.
- [24] Phillips C., Painton Swiler (1998), L., *A Graph-based system for network-vulnerability analysis*, Workshop on New Security Paradigms 1998, pp 71-79.
- [25] CVE, *Common Vulnerabilities and Exposures*, <http://cve.mitre.org> (acquired 22 October 2003).
- [26] *Network Vulnerability Analysis and Network Testing*, <http://www.blackmagic.com/assessment.html> (acquired 22 October 2003).

References

- [27] Wood, B. J., Bouchard, J. F. (2001), *Red Team Work Factor as a Security Measurement*, First Workshop on Information-Security Rating and Ranking.
- [28] Anderson R. (1993), *Why Cryptosystems fail*, 1st Conference – Computer & Comm. Security '93.
- [29] Hinde, S. (2003). *The law, cybercrime, risk assessment and cyber protection*, Computers & Security, Volume 22, Issue 2, 2003, pp 90-95.
- [30] Donner, M. (2003), *Toward a Security Ontology*, IEE Security & Privacy, May/June 2003, pp 6-7.
- [31] Noy, N. F., McGuinness, D. L. (2000), *Ontology Development 101: A Guide to Creating Your First Ontology*, KSL 01-05, Stanford University, 2001.
- [32] Sony Corporation (2002), *FeliCa Contactless Smart Card RC-S860*, EAL4, March 2002.
- [33] NIAP, National Information Assurance Partnership (2001), *Validation Report Smart Card Protection Profile (SCPP)*, Version 3.0, CCEVS-VR-01-0007, October 2001.

Glossary

CIA

The abbreviation is used to represent the three concepts of confidentiality, integrity and availability.

Class

This is the uppermost structure for the CC SFR, consisting of eleven different classes. A class could be considered a high-level security goal where the members (called families) all share a common focus.

Common Criteria (CC)

The Common Criteria for information technology security evaluation is a standardized method for evaluating the assurance of the correct implementation of the specific security design in products.

Component (in CC)

This is a structure for the CC SFR. Components belong to a certain family.

Component (physical system)

A system component is a physical or logical part of a DIS.

Distributed Information System (DIS)

In this thesis, the term distributed information system is used to emphasize the distribution of information in the system and the fact that users and organizations are considered to be part of the system.

Element

This is the lowest-level structure for the CC SFR. It contains a stated requirement of a single specific security task. One or more elements form a CC component.

Evaluation Assurance Level (EAL)

A package consisting of assurance components that represent a point on the CC predefined assurance scale. It ranges from 0 to 7.

Family

This is a structure for the CC SFR. Families belong to a specific class, and contain components. All family members share the same security goal, but they may differ in emphasis.

Framework

A framework sets the rules and structures that the model inside the framework has to follow.

Information System (IS)

This is an abbreviation for Information System, a system dealing with information, e.g. a computer.

Instance

An instance is an ontology concept that represents an object in the knowledge base. An instance is a concept with given attributes and thus has become objectified.

Ontology

A way to structure formal language, based on concepts, attributes and relations.

Open Systems Interconnection (OSI)

OSI is a reference model dealing with connecting open systems, which are systems that are open for communication with other systems. The different seven layers are: application, presentation, session, transport, network, data link and physical. Common security services performed in these layers are authentication, access control, data confidentiality, data integrity and non-repudiation.

PDR

The abbreviation is used to represent the three concepts of prevention, detection and reaction.

Protection Profile (PP)

A Protection Profile the implementation-independent requirements for a class of products or systems that meet specific customer needs.

Risk Level

The security value for a system in use that has been put into its contextual environment, i.e. threats and assets are included in the model. After reaching this value, risk management could be performed on the model.

Securability

The goal of designing for securability is that systems can be secured to an aspired level during operation. This is the security value for a physical system that is not in use, but has its organizational and individual aspects included in the model.

This term is in the thesis used to describe the security value for a physical system that is not in use but has its organizational and individual aspects included in the model.

Security Function (SF)

Security functions are part of the SFR from CC and represent those functions in a TOE that are considered to be relevant.

Security Functional Requirements (SFR)

SFR is the second part of the Common Criteria. It is a listing of a set of all the available security functions and a structured categorization of them into classes, families, components and elements.

Security Indicators (SI)

A term used in text to exemplify unspecific security values that somehow estimates the security present in a system.

Security Level

Security level is used to denote the security value of a system in use. Thus, its operational aspects are included in the model.

Security Metric

Security metrics are the methods measuring security and specifying, if not the meaning of the security value, at least the different values that can be assigned.

Security Target (ST)

A Security Target specifies the implementation-dependent "*as-to-be-built*" or "*as-built*" requirements that are to be used as a basis for a particular product or system.

Security Value

Security value denotes an unspecified value that gives some kind of estimation of the security of a system component. It could be specified further into securability, security level or risk level.

Standard Assurance Requirements (SAR)

SAR is the third part of the Common Criteria. It is a listing of all available assurance requirements and a structured categorization of them into classes, families, components and elements. It also defines the different EALs and explains the evaluation process of PPs and STs.

System

The system concept in this thesis is closely related to the component, as several components combines into a system.

Target of Evaluation (TOE)

An IT product or system that is the subject of an evaluation is in CC called the Target of Evaluation.

TOE Security Functions (TSF)

TOE Security Functions. Enforces the security provided in the TOE. It could be regarded as the TCB for a TOE.

Trusted Computing Base (TCB)

A TCB consists of a Reference Monitor, which validates all references made by programs in execution, together with all other functionality that affects the correct operation. Must be tamperproof, must always be invoked and must be small enough to be subjected to analysis and tests to ensure that it is correct. Consequently, it does not exist for general computers.

www.FirstRanker.com

Appendix

ID	Descriptive Name	CIA	PDR	ETS	MA
FAU	Security audit				
FAU_ARP	Security audit automatic response	CIA	R		MA
FAU_GEN	Security audit data generation	CIA	D		
FAU_GEN.1	Audit data generation	CIA	D		
FAU_GEN.2	User identity association	CIA	D		
FAU_SAA	Security audit analysis	CIA	D		MA
FAU_SAA.1	Potential violation analysis	CIA	D		
FAU_SAA.2	Profile based anomaly detection	CIA	D		
FAU_SAA.3*	Attack heuristics	CIA	D		
FAU_SAR	Security audit review	CIA	D		
FAU_SAR.1	Audit review	CIA	D		MA
FAU_SAR.2	Restricted audit review	C	P	E	A
FAU_SAR.3	Selectable audit review	CIA	D		A
FAU_SEL	Security audit event selection	CIA	D		MA
FAU_STG	Security audit event storage	IA	PDR	S	
FAU_STG.1	Protected audit trail storage	IA	PDR	S	
FAU_STG(2)	Guarantees of audit trail storage	A	P	S	M
FAU_STG.3*	Prevention of audit data loss	A	PDR	S	MA
FCO	Communication				
FCO_NRO*	Non-repudiation of origin	I	P	E	MA
FCO_NRR*	Non-repudiation of receipt	I	P	E	MA
FCS	Cryptographic Support				
FCS_CKM	Cryptographic key management	CI	P	T	MA
FCS_CKM.1	Cryptographic key generation	CIA	P	T	MA
FCS_CKM.2	Cryptographic key distribution	CIA	P	T	MA
FCS_CKM.3	Cryptographic key access	CIA	P	T	MA
FCS_CKM.4	Cryptographic key destruction	CIA	P	T	MA
FCS_COP	Cryptographic operation	CI	P	T	A
FDP	User data protection				
FDP_ACC*	Access control policy	CIA	P	E	
FDP_ACF	Access control functions	CIA	P	E	MA
FDP_DAU	Data authentication	I	P	E	MA
FDP_DAU.1	Basic data authentication	I	P	E	MA
FDP_DAU(2)	Identity of guarantor of data	I	P	E	MA
FDP_ETC	Export to outside TSF control	CIA	P	T	
FDP_ETC.1	Export of user data without security attributes	CIA	P	T	A
FDP_ETC.2	Export of user data with security attributes	CIA	P	T	MA
FDP_IFC*	Information flow control policy	CIA	P	E	
FDP_IFF	Information flow control functions	CIA	P	E	
FDP_IFF.1*	Security attributes	CIA	P	R	MA

Appendix

ID	Descriptive Name	CIA	PDR	ETS	MA
FDP_IFF.5*	No illicit information flows	CA	DR		A
FDP_IFF.6	Illicit information flow monitoring	CA	D		MA
FDP_ITC	Import from outside TSF control	CIA	P	T	MA
FDP_ITC.1	Import of user data without security attributes	CIA	P	T	MA
FDP_ITC.2	Import of user data with security attributes	CIA	P	T	MA
FDP_ITT	Internal TOE transfer	CI	P	T	MA
FDP_ITT.1	Basic internal transfer protection	CI	P	T	MA
FDP_ITT(2)	Internal transfer separated by attribute	CI	P	T	MA
FDP_ITT.3*	Integrity monitoring	I	D		
FDP_RIP*	Residual information protection	C	P	S	M
FDP_ROL*	Rollback	I	P	E	MA
FDP_SDI	Stored data integrity	I	D		
FDP_SDI.1	Stored data integrity monitoring	I	D		A
FDP_SDI(2)	Action due to loss of stored data integrity	I	DR		MA
FDP_UCT	Inter-TSF user data confidentiality transfer	C	P	T	A
FDP_UIT	Inter-TSF user data integrity transfer	I	D		A
FDP_UIT.1	Data exchange integrity	I	D		A
FDP_UIT.2*	Data exchange recovery	I	R		A
FIA	Identification and authentication				
FIA_AFL	Authentication failures	CIA	R		MA
FIA_ATD	User attribute definition	CIA	P	E	M
FIA_SOS	Specification of secrets	CIA	P	E	MA
FIA_SOS.1	Verification of secrets	CIA	P	E	MA
FIA_SOS.2	TSF Generation of secrets	CIA	P	E	MA
FIA_UAU	User authentication	CIA	P	E	
FIA_UAU.1*	Timing of authentication	CIA	P	E	MA
FIA_UAU.3	Unforgeable authentication	C	DR		A
FIA_UAU.4	Single-use authentication mechanisms	CIA	P	E	A
FIA_UAU.5	Multiple authentication mechanisms	CIA	P	E	MA
FIA_UAU.6	Re-authenticating	CIA	P	E	MA
FIA_UAU.7	Protected authentication feedback	C	P		
FIA_UID*	User identification	CIA	P	E	MA
FIA_USB	User-subject binding	CIA	P	E	MA
FMT	Security management				
FMT_MOF	Management of functions in TSF	CIA	P	E	MA
FMT_MSA	Management of security attributes	CIA	P	E	
FMT_MSA.1	Management of security attributes	CIA	P	E	MA
FMT_MSA.2	Secure security attributes	CIA	P	E	A
FMT_MSA.3	Static attribute initialisation	CIA	P	E	MA
FMT_MTD	Management of TSF data	CIA	P	E	
FMT_MTD.1	Management of TSF data	CIA	P	E	MA
FMT_MTD.2	Management of limits on TSF data	CIA	P	E	MA
FMT_MTD.3	Secure TSF data	CIA	P	E	A
FMT_REV	Revocation	CIA	P	E	MA

Evaluation of the Security of Components in Distributed Information Systems

ID	Descriptive Name	CIA	PDR	ETS	MA
FMT_SAE	Security attribute expiration	CIA	P	E	MA
FMT_SMR	Security management roles	CIA	P	E	
FMT_SMR.1*	Security roles	CIA	P	E	MA
FMT_SMR.3	Assuming roles	CIA	P	E	A
FPR	Privacy				
FPR_ANO	Anonymity	C	P	E	A
FPR_ANO.1	Anonymity	C	P	E	
FPR_ANO(2)	No solicit information while having anonymity	C	P	E	
FPR_PSE	Pseudonymity	C	PD	E	A
FPR_PSE.1	Pseudonymity	C	P	E	
FPR_PSE(2)	Reversibility in pseudonymity	C	P	E	
FPR_PSE(3)	Alias used in pseudonymity	C	P	E	
FPR_UNL	Unlinkability	C	P	E	MA
FPR_UNO	Unobservability	C	P	E	
FPR_UNO.1*	Unobservability	C	P	E	MA
FPR_UNO.3	Unobservability without soliciting information	C	P	E	
FPR_UNO.4	Authorised user observability	C	P	E	MA
FPT	Protection of the TOE Security Functions				
FPT_AMT	Underlying abstract machine test	IA	D		MA
FPT_FLS	Fail secure	IA	P	E	A
FPT_ITA	Availability of exported TSF data	A	P	E	MA
FPT_ITC	Confidentiality of exported TSF data	C	P	T	
FPT_ITI	Integrity of exported TSF data	I	D		
FPT_ITI.1	Inter-TSF detection of modification	I	D		A
FPT_ITI(2)	Correction of modified Inter-TSF data	I	R		MA
FPT_ITT	Internal TOE TSF data transfer	CI	P	T	
FPT_ITT.1	Basic internal TSF data transfer protection	CI	P	T	M
FPT_ITT(2)	Transfer separation of TSF data	CI	P	T	M
FPT_ITT.3	TSF data integrity monitoring	I	D		MA
FPT_PHP	TSF physical protection	CIA	P	S	
FPT_PHP.1	Passive detection of physical attack	CIA	D		A
FPT_PHP(2)	Notification when detection of physical attack	CIA	R		MA
FPT_PHP.3	Resistance to physical attack	CIA	P	S	M
FPT_RCV	Trusted recovery	IA	R		
FPT_RCV.1*	Recovery	IA	R		MA
FPT_RCV(3)	No undue loss after recovery	IA	R		MA
FPT_RCV.4	Function recovery	IA	R		A
FPT_RPL	Replay detection	I	D		MA
FPT_RVM	Reference mediation	CIA	P	E	
FPT_SEP	Domain separation	CIA	P	E	
FPT_SEP.1	TSF domain separation	CIA	P	E	
FPT_SEP.2	SFP domain separation	CIA	P	E	
FPT_SEP.3	Complete reference monitor	CIA	P	E	
FPT_SSP*	State synchrony protocol	IA	P	E	A

ID	Descriptive Name	CIA	PDR	ETS	MA
FPT_STM	Time stamps	CIA	P	E	MA
FPT_TDC	Inter-TSF TSF data consistency	I	P	T	A
FPT_TRC	Internal TOE TSF data replication consistency	I	P	T	A
FPT_TST	TSF self test	I	D		MA
FRU	Resource utilisation				
FRU_FLT*	Fault tolerance	IA	P	E	A
FRU_PRS*	Priority of service	A	P	E	MA
FRU_RSA*	Resource allocation	A	P	E	MA
FTA	TOE access				
FTA_LSA	Limitation on scope of selectable attributes	CIA	P	E	MA
FTA_MCS*	Limitation on multiple concurrent sessions	A	P	E	MA
FTA_SSL	Session locking	CIA	P	E	MA
FTA_SSL.1	TSF-initiated session locking	CIA	PR	E	
FTA_SSL.2	User-initiated locking	CIA	P	E	
FTA_SSL.3	TSF-initiated termination	CIA	PR	E	
FTA_TAB	TOE access banners	I	P	E	M
FTA_TAH	TOE access history	CIA	D		
FTA_TSE	TOE session establishment	CIA	P	E	MA
FTP	Trusted path/channels				
FTP_ITC	Inter-TSF trusted channel	CIA	P	T	MA
FTP_TRP	Trusted path	CIA	P	T	MA

Table 4: Revised CC structure (according to 5.2) including categorization of class, family or component. The classes are coloured dark grey, the families are light grey and the components are white. The meanings of the id and the last four columns are explained in the text.

Evaluation of the Security of Components in Distributed Information Systems

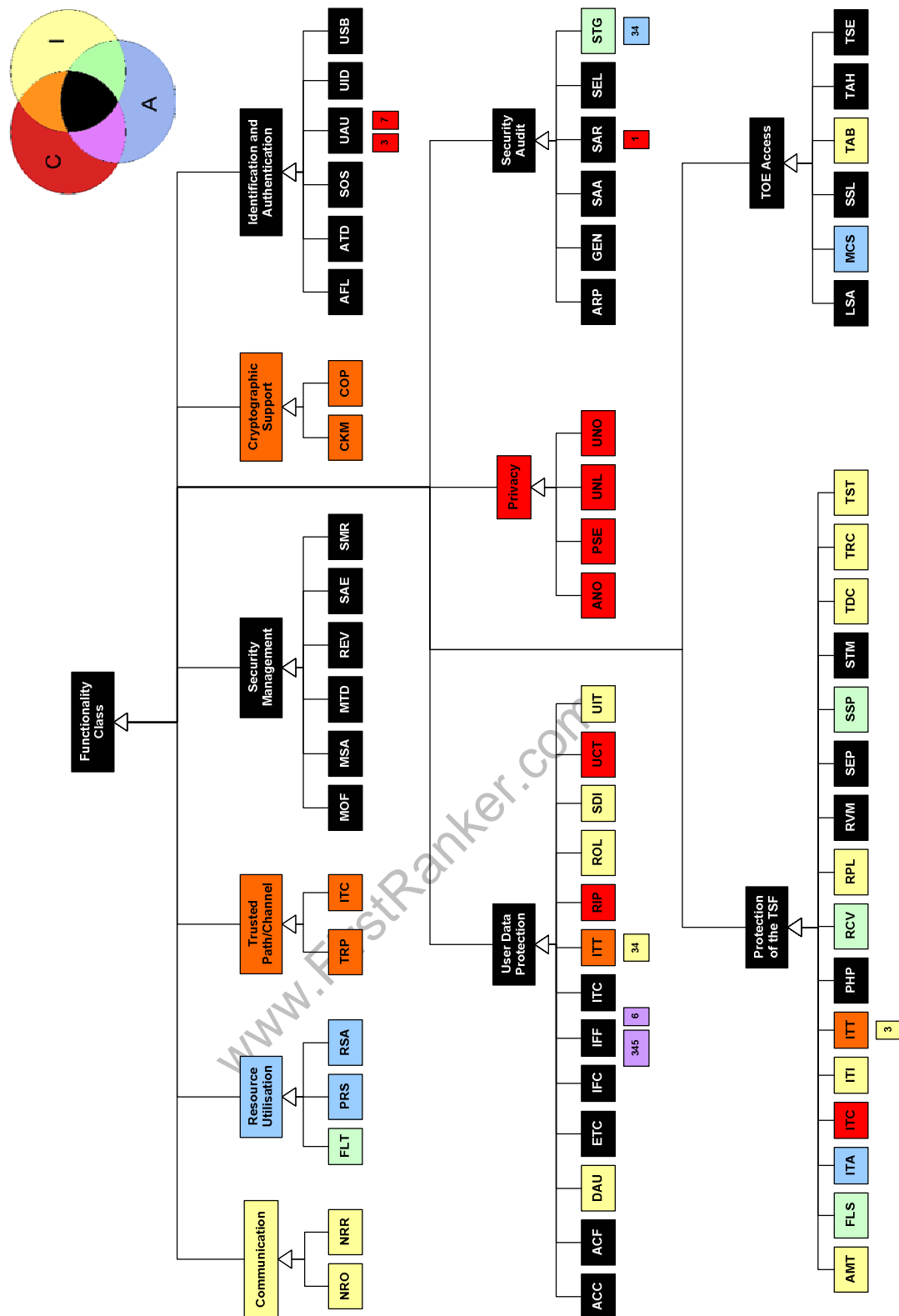


Figure 18: SFR coloured according to CIA. Those components that do not entirely share the view of their family are marked with a different colour than that of their family.

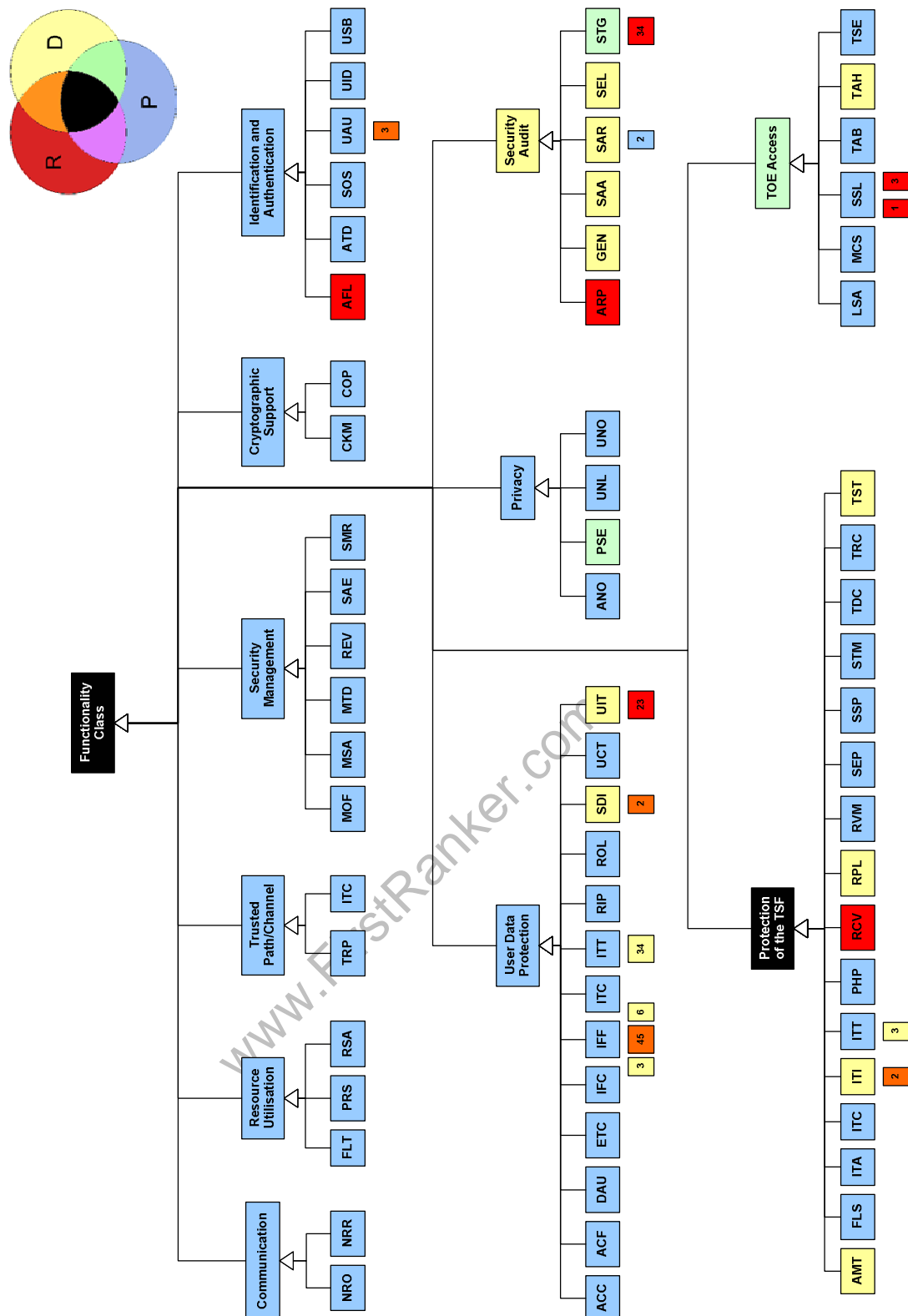


Figure 19: SFR coloured according to PDR. Those components that do not entirely share the view of their family are marked with a different colour than that of their family.

På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

In English

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

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

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

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

© Richard Andersson