MASTER'S THESIS

Software Security Design and Testing

LAPPEENRANNAN TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä:	Sami Masalin
Työn nimi:	Ohjelmistojen tietoturvan suunnittelu ja testaus
Päivämäärä:	1.10.2000 Sivumäärä: 83
Osasto:	Tuotantotalous
Professuuri:	Teollisuustalous / Tietoliikennetekniikka
Työn tarkastajat:	Professorit Markku Tuominen, Esa Kerttula
Työn Ohjaaja:	Section Manager Anna Pietilä, Nokia Networks Oyj.

Elektroninen kaupankäynti ja pankkipalvelut ovat herättäneet toiminnan jatkuvuuden kannalta erittäin kriittisen kysymyksen siitä, kuinka näitä palveluja pystytään suojaamaan järjestäytynyttä rikollisuutta ja erilaisia hyväksikäyttöjä vastaan.

Tietojärjestelmien suunnitteluprojekteissa ei ohjelmistojen tietoturvaa ole yleensä huomioitu laisinkaan tai vain hyvin vähäisessä määrin. Tästä johtuen monet järjestelmät ovat erittäin kehittymättömiä tietoturvan suhteen ja ovat riippuvaisia myöhemmin tehtäville korjauksille. Tämä asettaa rajoituksia elektroniselle kaupankäynnille ja pankkipalvelujen yleistymiselle ja yleiselle hyväksymiselle. Se myös antaa tietoturvaan erikoistuneille yrityksille erikoisaseman myydä tietoturvaratkaisuja hyvillä katteilla järjestelmien valmistajille ja palvelujen toimittajille.

Tietojärjestelmät voidaan kuitenkin suunnitella turvallisiksi heti alusta pitäen järjestelmällisen systeemisuunnittelun avulla, missä kaikki relevantit turvallisuusaspektit on huomioitu. Näin menettelemällä vältytään tuoteriskiltä ja samalla pystytään itsenäisiin turvallisuusratkaisuihin. Tämä tutkimustyö käsittelee niitä vaiheita, jotka tulisi sisällyttää systeemisuunnitteluun, jotta tietojärjestelmistä tulisi turvallisia ja tietoturvaan liittyvää tuoteriskiä pystyttäisiin tätä kautta välttämään.

Avainsanat: Tietoturva, Systeemisuunnittelu, Ohjelmistotekniikka

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY **ABSTRACT OF THE MASTER'S THESIS**

Author:	Sami Masalin
Name of the thesis:	Software Security Design and Testing
Date:	1.10.2000 Number of pages: 83
Department:	Industrial Engineering and Management
Professorship:	Industrial and Technology Management / Telecommunications
Supervisors:	Professors Markku Tuominen, Esa Kerttula
Instructor:	Section Manager Anna Pietilä, Nokia Networks Ltd.

E-commerce, M-commerce and electronic banking has introduced a growing concern about information system security. The main questions from the point of information system developer and application provider is how these critical information systems can be secured against all types of compromises in the future, where electronic transactions are playing the leading role.

There has not been much research in the field of Software Security Engineering. System developers have used to pay a little or no concern for system security and this has made the information systems vulnerable for all types of attacks. The insecurity of information systems prevents the acceptance of information systems in critical applications and delays the pace that new information systems for security critical applications can be introduced to the global markets. This insecurity of information systems has also allowed a new growing business that is selling security related applications for system developers and application providers.

Information systems can be developed to be secure, if all product development phases are committed in a systematic way, same time considering security aspects and implications of the information system. This research work studies all relevant phases for System Security Engineering and gives a basis for secure application development.

Keywords: Information Security, System Design and Testing, Software Security Engineering

TABLE OF CONTENTS:

1. INTRODUCTION

System engineering has evolved to a stage where quality attributes of software are playing increasingly important role. Introduction of E-commerce, M-commerce and banking related applications have resulted to a whole new set of requirements for information systems.

In addition of reliability, performance and other quality attributes, the level of system security is starting to play a major role when customers are making buying decisions. There are also some clear signs that product liability laws may start taking security aspects in to consideration. For these reasons there is an obvious need for a special consideration of system security when designing software products for high security applications. This thesis summarises the results of personal research in the field of System Security Engineering, done in Nokia Networks from the start of the year 1999 and provides a theoretical basis for product process development concerning product security.

The thesis describes all relevant steps when building secured information systems and provides required information for formalising product process model for improved security engineering. Information system security standards have been used to compose this document. To help the reading of this document, the document structure and the theory provided in each chapter are described in the Table 1.

Table 1. The Structure of the Master Thesis "Software Security Design and Testing"

Input information	Chapter	Output information
The basics of System Design and Testing, Information Security and Technology Management are required for understanding the theory presented in this chapter	**1. Introduction**	General information of the thesis and its limitations
The information provided in previous chapter	**2. Introduction to System Security Engineering**	Basics of System Security Engineering, laws and regulations related to product security and the objectives of System Security Engineering
The reader of this chapter should know the basics of tools used in Customer Requirement Analysis	**3. Customer Requirements Analysis and System Security Engineering**	This chapter explains how a Quality Function Deployment (QFD) method can be used to decide the system security features for System Security Engineering and System Engineering purposes.
The information provided in previous chapter	**4. Software Design and Development**	This chapter provides information how security features are analysed, designed and implemented to the system and how the process phases are formalized using the information of how critical the application will be.
The information provided in previous chapter	**5. Security Testing**	This chapter provides information of the steps related to the System Security Testing.
The information of code review processes is required to understand the methods for assurance that the source code of the software does not include any malicious code.	**6. Code review**	Code review process related to Security Engineering is explained.
The information provided in all previous chapters is required to understand the basics of system security engineering, when security quality standards are used and the certification related to this process.	**7. Certification process**	This chapter provides information about Common Criteria quality certification process.
It is required that the reader understands the basic steps related to System Security Engineering.	**8. Conclusions and Recommendations from the pilot projects.**	Introduced System Security Engineering Product process model is evaluated using the results from the pilot projects.
The basic knowledge about Information Security is required to understand this chapter	**9. Security in support functions**	System Security Engineering related support functions are reviewed in a general level.
	10. Summary	Conclusions from the research are presented.
	APPENDIXES A-H	Practical examples to aid in the understanding of the theory presented in the thesis

The process model introduced in this paper has evolved to the current form during one and a half year study of best practices in system security engineering. There were several information system projects that played as a pilot projects varying from the basic telecommunication server systems where security implications were quite insignificant to the highly secure systems as banking and E-commerce applications.

It was possible to test the efficiency of the process model introduced in this paper with a real information system development projects and evaluate the steps and results using these projects as a reference. The whole product process model introduced here is a result of the research done in a real environment with a real projects in Nokia Networks Ltd. with the help of the leading national experts in the field of System Engineering and Information Security.

1.1 Master thesis description

The purpose of this thesis is to provide a theoretical background for software's security development and testing process in the detail that is required for a full-scale implementation. The main emphasis is on the use of security standards and risk analysis methods in the system engineering. Other methods are also studied but not in the same detail. This study is a result of research carried out in computer security field, and it follows mainly ISO's standardised system security design process called Common Criteria (CC) and System Security Engineering Capability Maturity Model (SSE-CMM) approaches.

ISO is an International Organisation for Standardisation (ISO). ISO's quality standard "Common Criteria for Information Technology Security Evaluation" is the latest quality standard in information security field. It is probably also the best global level security standard to be used to formalise security design and security requirements for software products, and it is used as a baseline standard for this document. The basic structure of ISO's security standards is shown in Figure 1.

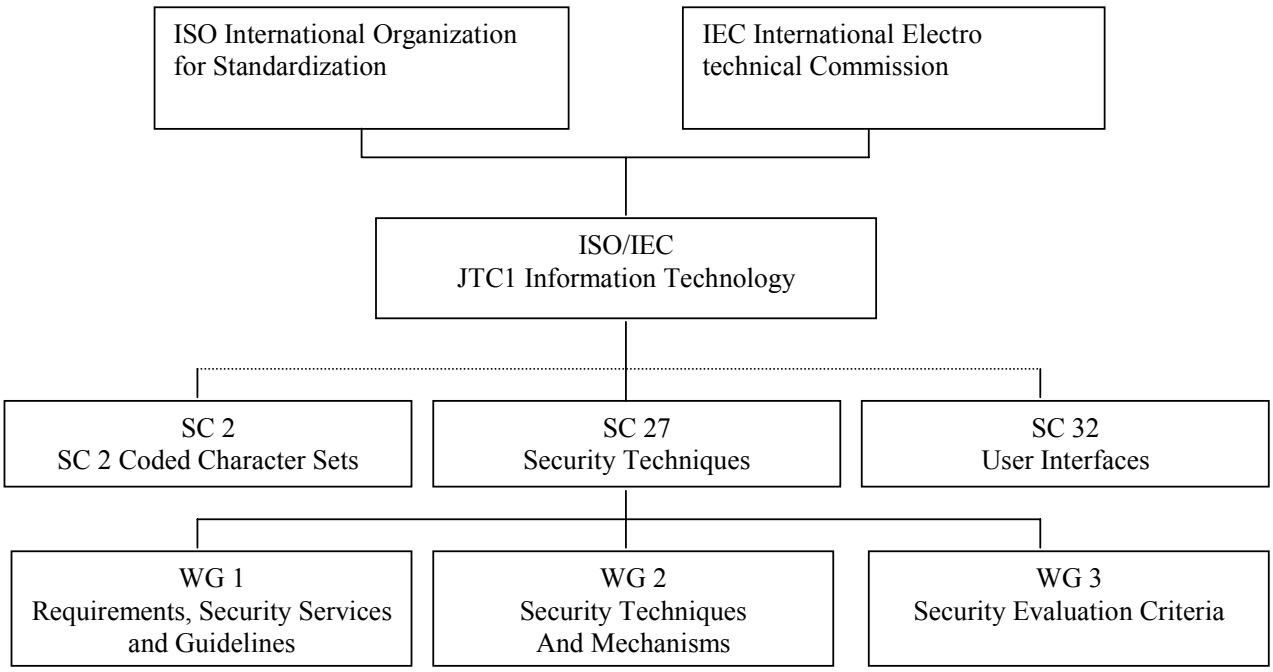


Figure 1. ISO standards related to System Security (Van Essen, 1999).

Evaluation criteria for IT Security is composed from three parts where the first part includes Introduction and General model; the second part includes Security Functional requirements; and the last part includes Security assurance model (ISO/IEC 15408, 1999). This standard is commonly known also as Common Criteria for Information Technology Security Evaluation (CC) and the latest version in the beginning of 2000 was version 2.1.

The software security development process with security standards presented in this paper is structured using worldwide known CC standard and SSE-CMM, because in the end it makes the certification process easier. If the management team decides to certify the security level of the product using CC, it is a good to have the process already appropriate for the certification. This is because in addition to the product evaluation, the product process is evaluated with the evaluation criteria. Phases of security engineering presented in this document are also recognized steps in the SSE-CMM and are because of that a basis for maturity level improvements and benchmarking (Van Essen, 1999).

Common Criteria is a product security standard that defines security assurance level for the software. Security assurance requirements for different security levels are defined in CC Part 3, where the assurance model is characterized using following general level security requirements for the software:

- EAL0 Unassured
- EAL1 Functionally tested
- EAL2 Structurally tested
- EAL3 Methodically tested & checked
- EAL4 Methodically designed, tested & reviewed
- EAL5 Semi formally designed & tested
- EAL6 Semiformal verified design & tested
- EAL7 Formally verified design & tested

(ITSEC, 15.5.1999)

EAL 0 is the lowest level and usually a starting point for software security development awareness. The level EAL 7 is the highest level, and in the practice almost impossible to reach due to the strict process requirements. The levels presented in CC are an easy way to rank the system security.

We use security certification standards as a basis for product process steps and that is why this document is structured in the following way. At first we look aspects that have to be considered when designing information system with and without standards. These include the definition of security level and security engineering process phases for the product; the system security design with and without standards; how to get assurance that the system functions are as specified; how to test the system security; and how to get assurance that the product is secure enough to be released in to the markets. Finally, the basics of a certifying process are presented. This is a trusted third party assurance for customers that the product process meets the required security level. In the end of the thesis conclusions of system security engineering process model, and a summary is presented. The reader should be able to implement a more secure approach for system engineering using methods presented in this thesis.

1.2 Objectives

The main goal of the system security design and testing process description presented in this thesis is to summarise the main aspects when developing a system for security critical applications. The process introduced here can be applied to all process models with slight adjustments.

The readers of this document are assumed to know the basic software development models (SDM) that are:

- The linear sequential model
- The prototyping model
- The Rapid Application Development (RAD) model
- Evolutionary software process models:
  - The incremental model
  - The spiral model
  - The component assembly model
  - The concurrent development model
- The formal methods model
- Fourth generation models (Pressman, 1996)

Also related E-milestones are required to be known to effectively implement phases introduced in this study. Refer to book "Software Engineering: A practitioner's approach" for more information about SDM alternatives and the implementation of those (Pressman, 1996). The generic process structure for software development is presented in Figure 2; steps introduced in the following chapters are referred to this structure.

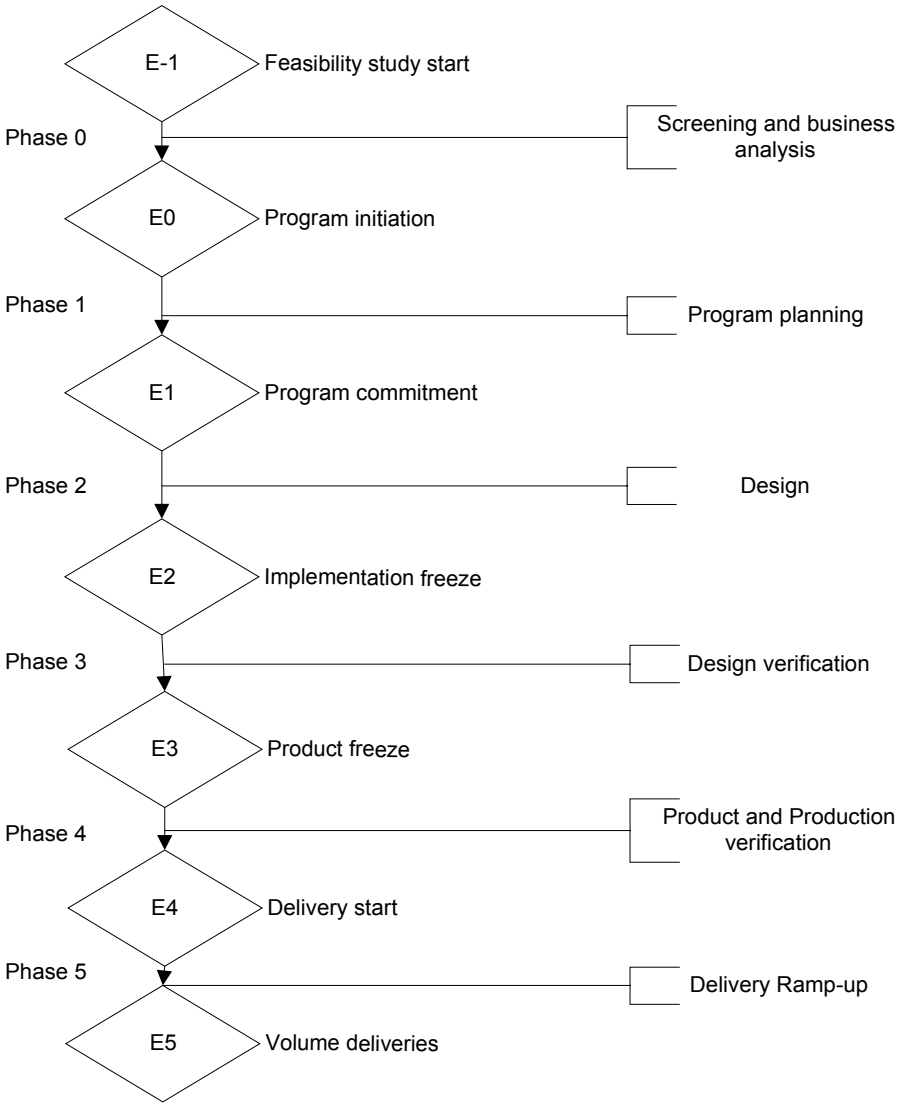


Figure 2. Product process and the related milestones.

1.3 The main concepts

Table 2 presents the main terms, acronyms and abbreviations of the software security design and testing process.

Table 2. The main terms, acronyms and abbreviations of the software security design and testing process.

Term, Concept, Acronym or Abbreviation	**Explanation**
CC	Common Criteria for Information Security Evaluation, the latest version 2.1, identical with ISO 15408 (see to http://csrc.ncsl.nist.gov/cc/ccv20/ccv2list.htm for more information of CC/IS 15408) (NIST, 2000)
Certification	In the context of this study the certification and certificates are an instrument to achieve an international assurance for the product quality.
Developer's Pedigree	Prior success rate in developing reliable products and systems
DMZ	De-militarised-zone, an outer area on the network where exists web servers and applications that are better to leave out from the main network and firewall.
GR 815	General requirements for security of Network Systems
ISO-9000	A quality assurance standard where, ISO 9001 encompasses product development process quality and ISO 9000-3 describes how to implement this standard to a software production (BSI, 2000).
IDS	Intrusion Detection Systems. The application that detect unauthorised alterations and access to the firewalls and hosts
IT	Information Technology
Phreaking	Phone line hacking

Security Safeguards	Security features that are implemented to the software to prevent information security violations
SETE	Security Testing
SSE-CMM	System Security Engineering Capability Maturity Model. A model that is used to benchmark general ability of an organisation to produce secured systems.
Supplier's declaration	Formalised developer's commitment to his products
SUPO	National Security Agency in Finland, "Suojelupoliisi".
SYTE	System Testing
TOE	Target of Evaluation, evaluated software or hardware
TCMM	Trusted Capability Maturity Model. Developmental security software assurance standard
VPN	Virtual Private Network, a virtual local area network that is achieved using encryption in the links
Warranty Assurance	Formalised developer commitment to stand behind their products and systems

2. INTRODUCTION TO SYSTEM SECURITY ENGINEERING

System Security Engineering is a term that is used to describe a process of making secure information systems. This term is usually miss-understood. Even some of very famous security experts still think that this means adding security components for the target systems.

Making information systems by just adding security components or security safeguards makes them vulnerable for attacks, because system security is not considered to be a quality factor of the whole system. In this approach the security of the system is as weak as the weakest link in the system and may be significantly weaker than the security level that security safeguards are meant to provide.

According to Bruce Schneier, one of the top cryptographers, "Security is not a product - it's a process.". This study aims to provide a systematic process for assessing the system security requirements and provides a method to design a balanced security portfolio for the product and in the same time ways to avoid security holes that incomplete system security engineering introduces. This paper also identifies the main aspects that are influencing the system security decisions (Schneier, 1999 B).

2.1 Laws and Regulations concerning Information System Security

Information security in the software products is not only a quality attribute that will influence customer buying decisions, it is also a matter which is regulated by laws. There are several laws that should be identified to understand the need for system security from the perspective of application developer, and from the perspective of the customers.

The laws and regulations concerning system security are currently quite limited but they are evolving rapidly. Already it can be seen that Product Liability Law and Electronic Communications Privacy Act are most important laws when considering minimum Protection Profile (PP) for the system in the perspective of application developer. There are also some Criminal Laws that will introduce requirements for information systems from the perspective of customers. For example, information system must have means for identifying and logging attackers to make the prosecution possible. For system security purposes it is not enough to delay and deter the improper access, it must be also detected. When aspects like this are understood during product development, it provides a strong basis for system security engineering.

Some of the computer related crimes that are addressed by Criminal Laws are:

- Unauthorised access
- Exceed authorised access
- Intellectual property theft or misuse of information
- Child pornography
- Theft of services
- Forgery
- Property theft (i.e., computer hardware, chips, etc.)
- Invasion of privacy
- Denial of service
- Computer fraud
- Viruses
- Sabotage (data alteration or malicious destruction)

- Extortion
- Embezzlement
- Espionage
- Terrorism

(Tipton & Krause, 2000)

Customer must consider all these laws when making SW system buying decisions. This is one of the reasons why product security certification is going to play a major role in security critical application markets. Information systems are starting to be so complex that it is difficult for customers to evaluate the system security without the help of quality standards and third party assessments.

There are also laws that are sometimes introducing upper limits for Information System Security (ISS). Examples of this are Export Control and Lawful Interception laws and regulations that are mandated by governments.

2.2 Information Technology Contracts

As in all manufacturing industry the application developer aims to develop the software using minimal resources and shortest time to markets. This always leads to compromises in the system security. This is also the reason, why application developer should make Information Technology (IT) contracts in a way that will encompass the responsibility issues, when the target system is compromised. The losses for customers in these cases may be very high. Therefore, it is wisest for an application developer point of view not to agree any responsibility of compensations. This is the way how the IT contracts are currently issuing other types of faults in the software. IT contracts for security critical applications must include the security issues to prevent time and resources consuming legal

cases. It is also advised that there will be a predefined procedure to handle security incidents in the terms of manufacturer actions (patch fixes, etc.).

2.3 The formulation of the System Security Engineering model for the target organization

There are several issues to consider, when choosing the best system security engineering methods for the target organization. The first issue to consider is all the things that have impact to the acceptance of the model between System Engineers, Project Managers, and Product Managers. The second issue is the security engineering efficiency of the model. Following points must be understood before the successful implementation of a new product process:

1. The basic requirements for System Security Engineering from the system engineering point of view are:

- Product process model must be easy to implement in various System development projects.
- It must be as easy to use in object-oriented models than in conventional models.
- It must not require Security expert to commit (limited resources).
- It must support the existing process model in the terms of system security engineering.
- It must not make existing process models harder to commit.
- The benefits must be visible (faults that are found must be traced and fixed).
- All tasks done must add value to the product (refer to chapter 3.1.)

2. The basic requirements for the System Security Engineering point of view are:

- The model must be able to identify all major security weaknesses in the system.
- The model must allow the fixing of problems in a cost effective manner.
- The model must include steps to fix the problems, and test the results of system security engineering.

- The model must recognise the most relevant phases that are seen in the current system security engineering trends around the world (that is why the standards are described in greater detail).
- The model must be adaptable from basic security level systems to the Top Secret systems.
- The model must be simple to allow easy approval.

The model described in the following chapters is formulated using these issues as a pre-requirement for process model.

3. CUSTOMER REQUIREMENTS ANALYSIS AND SYSTEM SECURITY ENGINEERING

The customer requirement analysis is the first step that is committed when system engineering process starts. There are several methods to do Customer Requirements analysis. The best way to find customer requirements for the security features of the system is to use Quality Function Deployment (QFD) method to analyse how important attribute the system security is for customers. With the help of QFD it is possible to allocate enough resources for security engineering, decide the required security level for the product, find if security standards should be used and find out most important security features for the target system. The importance of QFD is obvious for cost management purposes of system security engineering. It is not wise to develop properties to the product that are not appreciated by customers.

3.1. Quality Function Deployment (QFD)

Quality Function Deployment (QFD) is a relatively new method that is used to analyse which product features are important for customers and how good are the competitors' products compared to the target organisation products. The rule of thumb is that the most

important product features should be better than in competitor products and less important features should be near the same level with competitors. With QFD it is possible to find out which features are appreciated by the customers and from this it is easy to find out the most important features for marketing and allocate resources to improve those features in product design.

QFD takes account of how good are the competitor product features and which features customers are requiring and appreciating; by this it is easy to achieve the right combination of features without wasting resources and overdeveloping some features. QFD analysis should be done for product line, before implementing a full-scale system security engineering process model, to find out the most profitable balance between system security and resource usage. QFD analysis should be redefined for each product project to find out the right security level, applicable security standards and features for the product.

How secure the system must be, what approach to use in system security engineering and which security features to have in the target system is decided on the basis of QFD analysis. Using QFD approach it is possible to prove the feasibility of intended system security engineering level for the target software and find out at the same time how much customers and competitors know about information security. This is a critical factor in system security engineering decisions. The target of QFD is to find such security attributes for the software that will result better product than competitors have from the perspective of customers. This will make the market position of the information system better and in the end increases profits.

The minimum level for product security is not always decided using QFD because there are some threats that inproper security engineering introduces to the company image and customer relationships. An example of this is a possibility of negative press releases because of security compromises during information system's life cycle. That is why minimum level for product security can be higher than QFD has demonstrated to be feasible. The highest level for the product security is best to be decided using QFD method to avoid over development.

There are also other methods than QFD to define customer requirements and preferences. An example of these is a Conjoint-analysis, which uses relative preferences to find out the most important features and feature combinations for the product. QFD is the best method currently available for deciding quality attributes such as security requirements for System Security Engineering and that is why it is analysed in detail. (LUT, 1999)

3.1.1 How to use QFD to define product features

The QFD analysis starts by defining customer requirements for information system (A). After that the customer requirement correlation's for competitor products are defined and competitor analysis is performed for the target market segment (B). Then the company defines how these customer requirements can be met, implementing functions and features to the product (C). If needed, correlations of the defined product features with each others are also analysed. With this it is possible to detect if some customer requirements can be met with more than one product feature (D).

The most important phase in QFD is to define the correlations between product features and with customer requirements, which is performed using information from previous phases (E). After that all product features are prioritised using customer requirements, product feature and correlation weight information (F). Usually 3-5 product features are selected for improvement and preliminary product specifications are made on the basis of that (G). The values of matrix are entered to the QFD matrix as seen in Figure 3. (LUT, 1999)

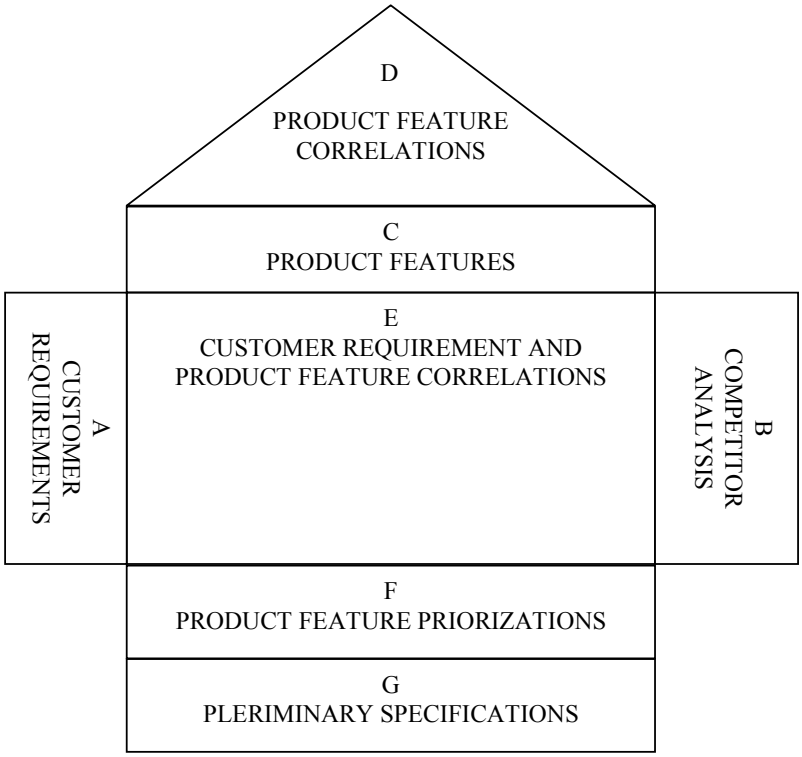


Figure 3. Quality Function Deployment matrix (LUT, 1999)

Parts A and B of QFD matrix are called "Customer table". The information for this part is collected from outside the company. Other parts (C, D, E, F and G) are called "Technical table" and the information for these parts is collected during QFD process. The customer table part provides information from the market segment of the product and the technical table provides information of how the company can differentiate it's products for the intended market segment.

3.1.2 Use of QFD when identifying security features for the product.

The QFD study provides product requirement information for system security engineering. From the QFD we can see if customers prefer evaluated and certified products, how good are competitors products compared to our products and which security features to use and develop further to differentiate the product in market segment. In Appendix D* is

demonstrated a very basic QFD analysis matrix for security critical application. From there we can see that following aspects must be included in to the QFD analysis for system security engineering purposes:

1. Intended applications and how secure system must be for those application
2. How secure and good products competitors will have and how good competitors are in system security engineering
3. Security standards that should be used and the target security levels
4. Preliminary Security level (as a reference) for the product if the product is not evaluated and certified
5. Security Features that customers or/and laws are requiring for the product (must be features)
6. Preliminary security features and functions that the intended security level requires

This information is then used for making prioritisation for development in system security engineering and marketing. This matrix can also be used to demonstrate the feasibility of system security engineering resource allocations for product line management.

If security attributes are not found to be important using QFD analysis then the product is manufactured using conventional methods. The matrix is usually done with the co-operation of system security engineers, marketing and product/project management during product process phases E–1 - E1. When the QFD analysis is performed the other phases in the System Security Engineering process follows the prioritisation analysed and decided in the QFD study.

** The example in Appendix D is not a complete QFD analysis and is used only for illustration purposes.*

3.1.3 Additional tools for QFD analysis

For system security engineering purposes, additional tools can be used with the QFD. Plain QFD does not provide any information about following aspects: (1.) Previous product launch security related reclamations and security holes found by the customers from the previous product versions, (2.) if the system security is one of the main marketing arguments (a so-called strong marketing argument), we must define in QFD how the product will be marketed and (3.) previously mentioned requirements introduced by laws and regulations (see chapter 2.1).

To include these additions to the QFD analysis, we have to add numbered fields as seen in figure 4 to the matrix.

1. The first and the most important addition to the matrix is the definition of the product feature fields where has been problems in the earlier product releases. If customers have been complaining in with regard of some security feature, the information must be used for the QFD analysis to fix the problem before it has negative effect to customer relationships.

2. The second improvement for QFD analysis is to define the strong marketing arguments from the implemented security features. In each product there should be only two to four strong marketing arguments that will be used to differentiate the product and that is why these arguments must be carefully chosen.

3. The third aspect to include in QFD is requirements introduced by the laws.

Refer to Appendix E for more detailed example of QFD's additional tools.

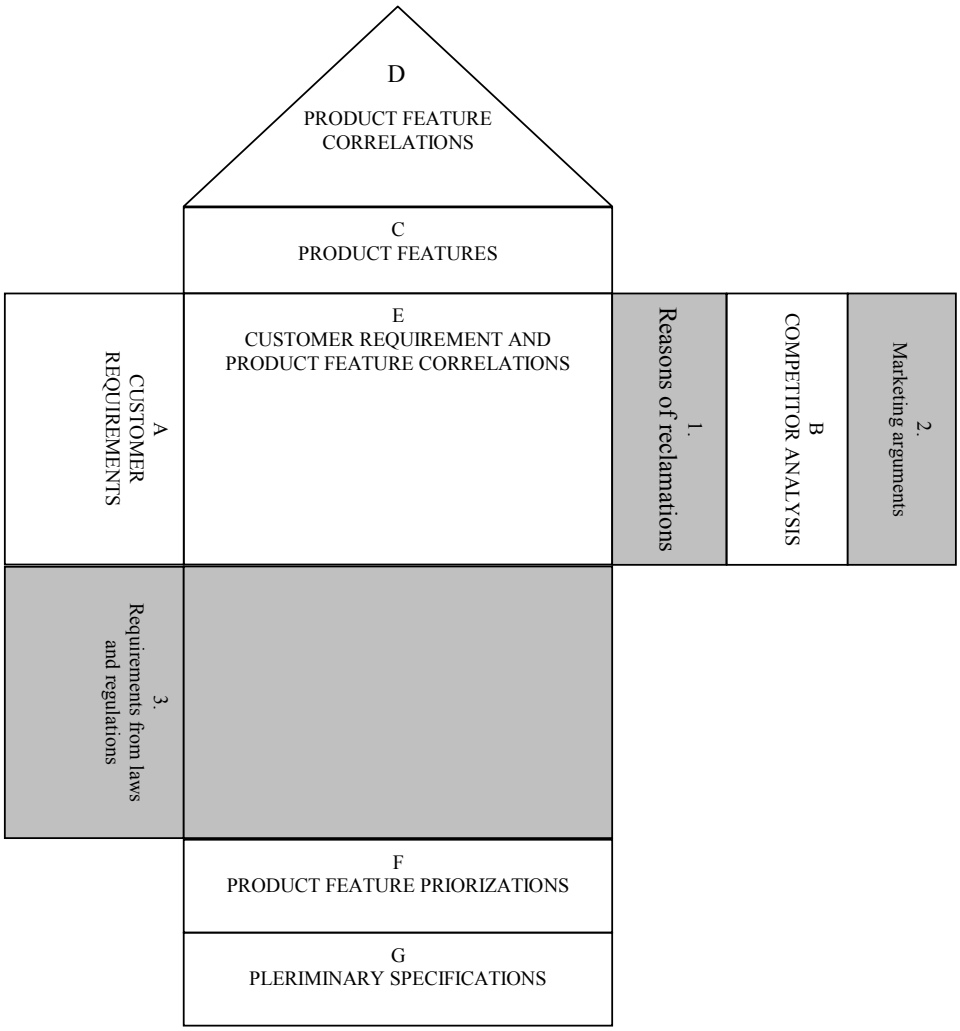


Figure 4. Quality Function Deployment matrix with additional tools (ANASTA, 2000)

Previously mentioned additions are important to the QFD analysis from the System Security Engineering point of view. There are in addition of these tools several other tools to be used with QFD. Although it is not recommended to use too many tools, because it makes the QFD analysis complex and time consuming for the implementation. At least the first QFDs committed for the target product should be as simple as possible. (ANASTA, 2000)

3.2 System Security Engineering

As identified earlier one of the most important issues is system security, when developing software that is going to be used in applications critical in the security matters/issues, e.g. when developing information system for the banking area. The only way to develop a secure product is to systematically assess and improve security functions and features during the software development process using the help of the previously explained QFD method to choose the right level for the system security and the appropriate security features. This means that the software product process must be looked at from different angle. Appropriate design for security is analysed during product process. This kind of approach has been a common method in manufacturing industry where the product is specially designed using a pre-defined quality attribute, for example reliability. This approach is commonly called Design For X (DFX). In this study we concentrate to the Design For Security (DFS).

The security is not a quality attribute, which can be archived in the software by using simple security function and feature add-ons such as encrypted communications between objects of the system and ISO 9001 quality approach. Even though if customers are not able to explicitly define security attributes for the system, certain security engineering steps must be committed. Software development model must be studied systematically to prevent hacking, internal misuse etc. during the product lifecycle (Australian Defence Force Academy, 1999).

The rule of thumb in information system development is that costs of fixing faults are rising during the product process. Pressman (1997) has demonstrated that costs are exponentially rising during product life cycle using the cost factors as seen in Figure 5.

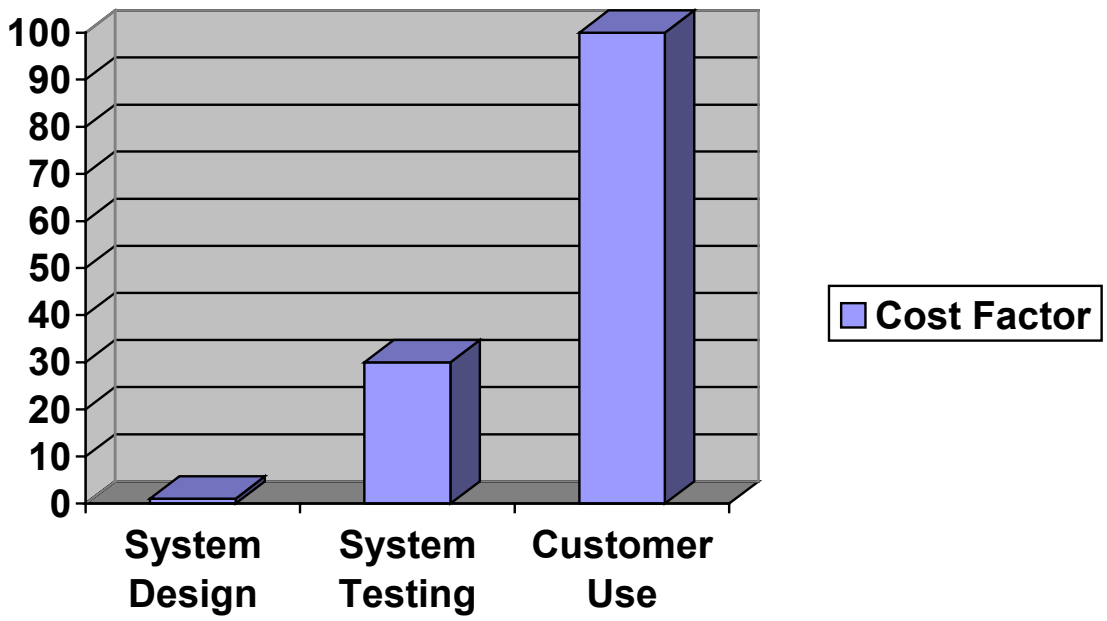


Figure 5. The cost of fixing defects, and faults in information systems (Pressman, 1997).

As we see from the figure 5 more emphasis should be given on system designing phases to avoid faults to be found in the later phases of the software lifecycle. Currently the practice is the opposite: software developer introduces a product to the markets, and then security holes are fixed in the form of patch fixes. Some software firms are doing some kind of security testing to find weaknesses but this is not the proper way either.

Security faults can be avoided using proper system engineering practices during product process. Information systems can be engineered as safe as is needed. However, it is usually decided that the system is engineered to the level that is required for intended applications. It is clear that proper system security engineering can save money from developers and customers.

4. SOFTWARE DESIGN AND DEVELOPMENT

The system design (SD) has to be studied systematically to achieve the reasonable and required security level for the software. The following V-model presented in Figure 6 can be applied for security evaluation during System Engineering. This means that security evaluation of target product should be addressed in each software development phase.

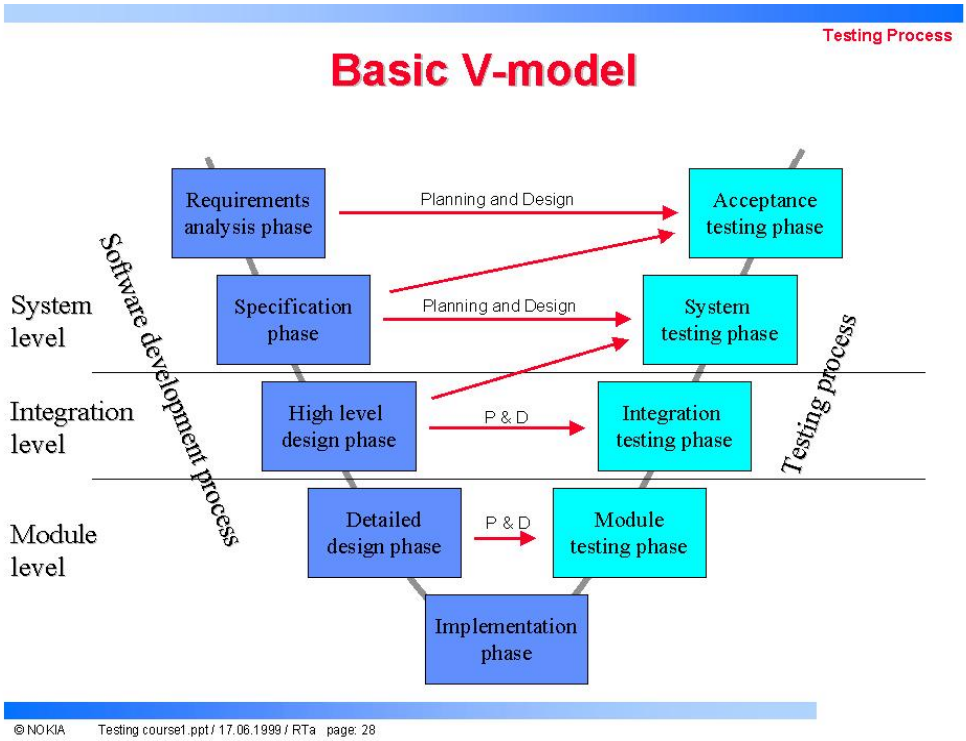


Figure 6. The software evaluation and testing V-model

All these phases presented in figure 6 are connected to the evaluation of system security. This means that the system security must be analysed in all these phases to provide required assurance for system security. The security development process must be divided into the following stages, because of this requirement:

- **Security Requirements Analysis (Phase 1)**

-Customer requirements, QFD

-System specification

-Standards/criteria's to be used (Van Essen, 1999)

-Experience

- **Threat Analysis (Phase 1-3)**

-Standards, or attack tree and check list methods

-Experience

- **Theoretical Evaluation (Phase 2)**

-Functionality

-Assurance for the functionality of the features

- **Implementation (Phase 3)**
- **System Testing / Security Testing (Phase 3-4)**
- **Examination of the Source Code (Phase 3 & 4)**
- **Delivery (Phase 4-5)**
- **Learning from the mistakes (Phase 5)**

-Actual penetrations during product operational phase (defects found and used by criminals)

-System penetration tests that are carried out in customer sites during α-testing

-Evaluation by trusted third party (ITSEC, 1999)

-Defects found by customers during normal operation (Huhantti, 1998)

These stages are presented in the following chapters. Defining security requirements, threat analysis, theoretical evaluation, and examination of the source code are under the responsibility of the development/design group. Testing belongs to the responsibility of testing group. After the delivery of the product when the product is in its normal operation (during phase 5) there may be cases when the system is compromised, and the security safeguards are penetrated. It is customer responsibility to report these cases as fast as possible to the developer and developers' responsibility to fix the problems in a reasonable time limit (Warranty assurance). This leads to positive iteration for security safeguards when piloting and versioning development models are used in system design. *(Refer to Appendix B for detailed description of the System Security Engineering Model.)*

4.1 Security Requirements

The first step to start with the product process and the system security engineering is to define security requirements for the product. During this phase it is decided using QFD, if the product process follows some predefined security quality standards as Common Criteria (Chapters 4.1.1 and 4.2.1), or is it better to start the process without standards (Chapters 4.1.2 and 4.2.2), and use attack tree and check list based approaches (Structured Analysis). Main security features (security safeguards) are defined in a general level using QFD information.

When defining security requirements and features for the product the most valuable information comes from customer requirements and QFD studies. It is a relatively easy to see from customer requirements how secure the product must be (security level) even if customers cannot explicitly define a security level for the product. If customers are able to define a security level for the product, it is usually best and easiest approach to use security standards. Otherwise, the decision is made between standards and Structured Analysis (SA) on the basis of customer security requirements, product differentiation strategies, time schedule, and available resources for security engineering.

The system security engineering process is a quite similar when using the attack tree and checklist method (Structured Analysis) comparing to the security standards. The major difference is that security analysis with security standards is performed more systematically, and the possibility of serious security holes left in the product is somewhat smaller. The drawback of the approach with standards is that it takes more time and work, and because of that it is usually unfeasible approach for majority of SW projects. The better method with tight deadlines is to use a variety of security risk management tools and experience to analyse the system and it's main vulnerabilities.

4.1.1 Classification of Requirements for product process if standards are used

The simplest approach to start using security standards in engineering is to define a security risk level for the computer system in a general level before E1-milestone. This is implemented by using the risk rating of the most sensitive information that is handled by the computer system and the minimum user clearance of logical and physical access to the system. Risk index is simply calculated using equation of *Risk Index = Max Info Sensitivity – Min User Clearance*. (See Figure 7)

Rating Level Table:

Rating	Info Sensitivity	User Clearance
0	Unclassified	Uncleared
1	Restricted	Restricted
2	Restricted (categories) Confidential	Confidential
3	Confidential (categories) Secret	Secret
4	Secret (1+ categories)	Top Secret
5	Secret (2+ categories) Top Secret	Top Secret
6	Top Secret (1+ categories)	Top Secret - 1 category
7	Top Secret (2+ categories)	Top Secret - many categories

Risk Index = Max Info Sensitivity - Min User Clearance

Figure 7. Risk Index Rating (Australian Defence Force Academy, 1999).

After deciding the Risk Index for the product, it is allocated to the appropriate security level. The mapping method is following presented in Figure 8. CC is used as a reference.

Risk Index	Security Mode	Min Class Open Env	Min Class Closed Env
0	dedicated	EAL 1	EAL 0
0	system high	EAL 2	EAL 2
1	limited access, controlled, compartment, multi-level system	EAL 3	EAL 3
2	limited access, controlled, compartment, multi-level	EAL 4	EAL 4
3	controlled, multi-level	EAL 5	EAL 5
4	multi-level	EAL 6	EAL 5
4	multi-level	EAL 7	EAL 6
>=6	multi-level	EAL 7	EAL 7

Figure 8. Mapping Security Requirements to the appropriate security class.

The security mode of the system is one of the following, according Australian Defence Force Academy (1999):

- **Dedicated (Single-Level) Systems**

Handles subjects and objects with same classification.

Relies on solely on other security procedures (e.g. physical).

- **System-High**

Only provides need-to-know protection between users.

Entire system operates at highest classification level.

All users must be cleared (Authenticated and Authorized) for that level of information.

- **Compartment**

Variation of System-High that can process two or more types of compartment information.

Not all users are cleared for all compartments, but all must be cleared to the highest level of information processed.

- **Multi-Level System**

Is validated for handling subjects and objects with different rights and levels of security simultaneously.

Major features of such systems include:

- User identification and authentication

- Resource access control, and object labeling
- Audit trails of all security relevant events
- External validation of the systems security

When the appropriate security class, called Security Target (ST) for the product, is chosen using the help of Protection Profile (PP) of similar products or creating a new one, then the next step is to study the product structure and system engineering process phases that the required security level demands. Required system engineering phases for chosen security level are listed in Common Criteria standard, but the security features for system must be mapped using the information provided by PP.* The main task is to define which of those security requirements the product already fulfills, and which improvements must be implemented using security functions to achieve the goals stated in PP. ** This work is called a pre-evaluation phase. After this starts the security evaluation phase, where security functions are defined and designed in the detail (see chapter 4.2.1.)

** The procedure of mapping security features from threats is presented in CC and is out of scope of this thesis.*

** This is the defining order if we already have existing product specification and requirements. Usually this is the case because security is not the main functionality of the product. Therefore we are seeking basically security improvements to the existing product architecture or more secure alternatives for the architecture. For the new products the security features are chosen after the main features using the standards if applied.*

4.1.2 Process when standards are not used (Structured Analysis)

For systems that do not need a waterproof security and the project time limits are tight, the best approach is to follow the process steps introduced earlier in the paper and use experience, attack trees, and various check lists to reveal the biggest faults in the design and review these during security testing (More information of these tools and methods in chapter 4.2.2.).

System security engineering without standards is a process that has to be planned and customised separately for each project. During security requirement definition step the methods and tools that are used to analyse the system during threat analysis are selected, the responsible persons nominated and the time limits defined.

In this paper the study is concentrated mainly to the attack tree method introduced by Bruce Schneier (1999A) and to the security checklists. There are other more formal methods that can be used in designing and evaluation of information system security but are usually not required if the system is not intended for critical applications. These methods are called Security Models (SM) and they are tools to be used with the attack tree method to study and analyse information flow security for example an information flow inside the information system. These tools are also used in some security standards to evaluate and design security constrains for system processes. In this paper these tools are listed but to gain a deeper understanding about formal methods it is advised to study these methods further.

4.2 Threat Analysis

Threat Analysis (TA) is a systematic security analysis of information system where all relevant security weaknesses are identified, assessed and, possible countermeasures evaluated. The traditional way for software engineering is to design the user interface and other security related features and in the same time consider security implications. If the specifications are not studied, reviewed and analyzed after the design, there will be security holes left in the product design, because security considerations are not systematically analyzed. Threat Analysis is intended to prevent these security weaknesses before the system is build.

There are two different methods to perform a threat analysis and security feature design. Systematic and time consuming analysis starts with using security quality standards such as

Common Criteria to evaluate system's general security requirements, and to identify the countermeasures for the threats identified (security functions/features). The other method uses attack trees and systematic reviews to identify the main vulnerabilities for the system, and evaluates/designs the supporting security functions from the basis of that.

When standards are used the threat analysis is only performed if the security objectives are (or have been) identified without using a systematic threat identification mechanism. Threat analysis in this case is performed after the specification of security requirements and security functions (PP & ST). The better approach is to use attack trees before specifying security objectives for the system. However as identified in this paper the work can be done also after the specification of security features.

Threat Analysis is conducted before E2 milestone, at the same time as system and functional specifications. The first draft of threat analysis is made and reviewed for the E2 milestone and updated during the design phase until the final version is reviewed for the E3 milestone. (See Appendix B for additional information about the milestones and related design steps.) If the system is scanned using vulnerability scanning tools during testing, the results of the scanning can be added at later phases to the threat analysis.

4.2.1 Threat analysis when security standards are used

The most systematic method for security design is to use security standards as a requirement reference for the system design. Threat Analysis (TA) with security standards is a semi-formal method to produce secure systems. TA can be done before or after the design as identified earlier. If the TA is done after the security design it uses system specifications and security objectives as stated in security standards to review that the implementation meets the required security target and to derive new security objectives. In this case the TA is a process to verify that specified security features are enough to provide

the required protection for the product. Formal security models are also used in the higher security classes to analyze security.

TA can be performed also before the PP specification and before the product design to define the security objectives/requirements for the system. The combination of these security objectives/requirements (that can be derived from the results of TA) and applicable security features presented for the product is called a Security Target (ST) in CC.

For example CC's Functional class, called Privacy (FPR) stated in a CC part 2, defines following requirements for the privacy of systems user data:

- FPR_ANO: Anonymity
- FPR_PSE: Pseudonymity
- FPR_UNL: Unlinkability
- FPR_UNO: Unobservability (ISO/IEC 15408-3, 1999)

Each of them has different sub-requirements for fulfilling security objectives. If the system does not meet all requirements for security (as a result of TA done after the definition of functional requirements), the implementation has to be changed, and new system specifications have to be made. Refer to Appendix G for an example of functional requirement preparation.

Note that there are also other security standards than CC that can be used to define functional requirements e.g. GR-815 for Network element/Network security. CC is used here to demonstrate the designing process when standards are used, (See Appendix C for more information about the alternative standards.).

The studies of operation environment architecture, and system specifications are the required inputs for preparation of threat analysis as seen in Figure 9. The environment study (Network configuration) for threat analysis includes how the Firewalls, VPNs, and other security tools are used to secure the system. It is a mandatory element for understanding the system security requirements. When the system architecture and expected operational environment are defined, the standard, attack trees and formal

methods (Security Models) are used to analyze the security weaknesses and countermeasures.

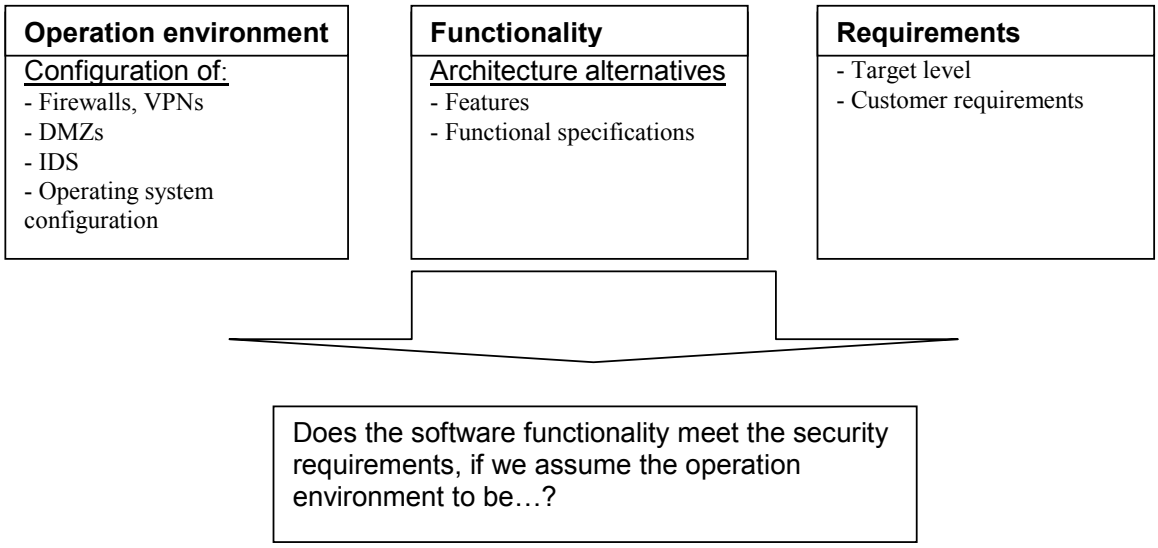


Figure 9. Threat analysis.

It should be noted that security standards do not have all required information for security design and that is why system design with standards must sometimes be combined with independent Structured Analysis to analyze all relevant information for system's security. This is for the reason that standards have some deficiencies related to the security requirements and features because a standard is basically already old when it is publicized. A good approach is to use attack trees to describe the security aspects that are not included in the used security standard (refer to the chapter 4.2.2). An example of this is seen in Appendix H.

When the threat analysis is prepared we have all relevant information of how the system is designed to prevent the compromises. Threat analysis provides a balanced picture of the system security for the design. It also proves that all security features are as good as required for the intended usage of the system. Threat analysis is also an entry document for thorough security testing and that is why it is the most important activity in system security engineering.

4.2.2 Threat analysis with the help of Structured Analysis (SA)

Usually information system projects have very tight schedules, and limited resources. When this is the case, and the product is still intended for security critical applications, then the proper approach is to structurally evaluate the biggest security threats and countermeasures for those with the help of structured analysis. System vulnerabilities may be found in both security and non-security related parts of the system. In many cases, non-security mechanisms that support security functions, or work with security mechanisms, are found to have exploitable vulnerabilities. The methodology of attack scenarios should be followed to the extent that vulnerabilities are validated. All system vulnerabilities should be recorded in to the threat analysis. The most recognized methods for threat analysis with the help of SA of information system are varying security checklists and 'attack tree' method introduced by Bruce Schneier (1999).

The attack tree method seems to be easiest and the most systematic way to perform threat analysis for the system if security standards are not used. It is also a valuable tool to be used with security standards.

Attack tree represents the attacks and countermeasures as a tree structure similarly as 'fault trees' introduced for fault-tolerant information systems (Pressman, 1997). With the help of attack tree it is possible to select the methods and techniques by which system vulnerabilities in a defined environment are prevented. Structured Analysis starts by identifying the general system architecture, and possible attack scenarios in a general level e.g. denial of service attack. It grows branches: how those attacks are performed, how obvious they are and what is the possible impact (Figure 10. and Appendix F). With the help of this it is quite easy to choose the proper countermeasures for the system vulnerabilities, and combined with security checklists it gives nearly as good results as standards. For more details about TA refer to Canadian Communications Security Establishment handbook MG-3 for a good description of Threat/Risk Assessment of information systems (CSE, 1999).

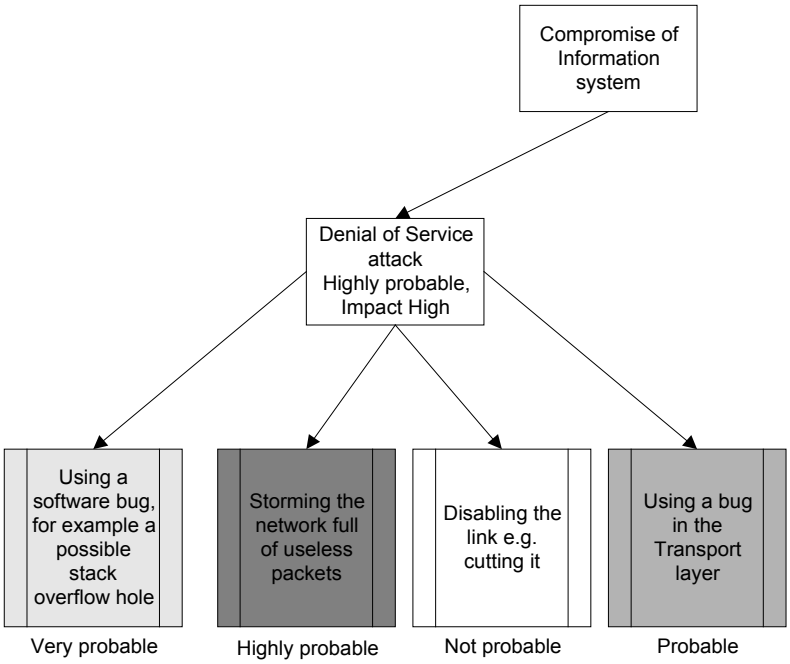


Figure 10. Attack tree with probability and impact estimations (Schneier, 1999 A).

The branches of 'Attack tree' are further analyzed if needed in the process and system call level using Security Model requirements. Improvements to the design are made in accordance of the results.

Most common Security Models are:

- Bell-LaPadula
- Biba
- Clark & Wilson
- Non-interference
- State machine
- Access matrix
- Information flow and information flow matrixes (CISSP, 2000)

Where Bell-LaPadula is an abstract formal method that defines the process Confidentiality. It defines a concept of secure state and fundamental modes of access. It also gives rules for

giving subjects access to objects. Where secure state means, that only permitted access modes, subjects to objects, are in accordance with specific security policy. There are three modes of access: read only, read & write, write only.

Bell-LaPadula access rights are illustrated in Figure 11 as a process between security layers, where subject can not read object of higher sensitivity and where a star property means that subjects can not write to object of lower sensitivity. The Strong star property enforces that objects can not read or write to any other security layer.

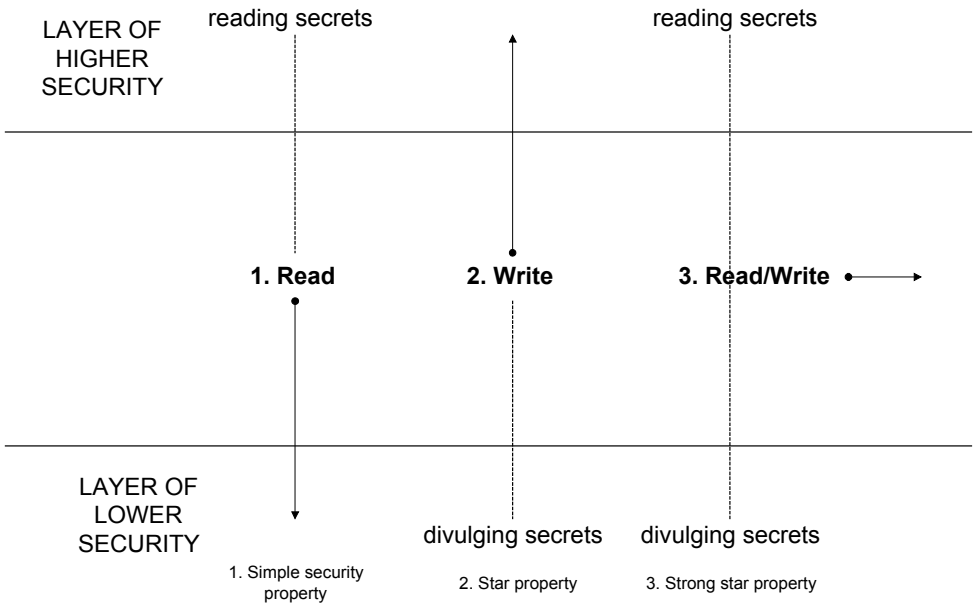


Figure 11. Bell-LaPadula (CISSP, 2000)

Biba and Carl & Wilson are intended to solve the problems with data integrity. Biba was first security model to address integrity in computer systems. Biba is based on hierarchical lattice of integrity levels and addresses first goal of integrity, prevent unauthorised users from making modifications.

Biba defines two type of elements: 1) Set of subjects that are active, performing information processing 2) Set of objects that are passive, being information repository and

integrity roles for those. Biba integrity properties are illustrated in Figure 12 as a process between security layers, where subjects can not observe (read) objects of lesser integrity and where a star property means that subjects can not write to object of higher integrity.

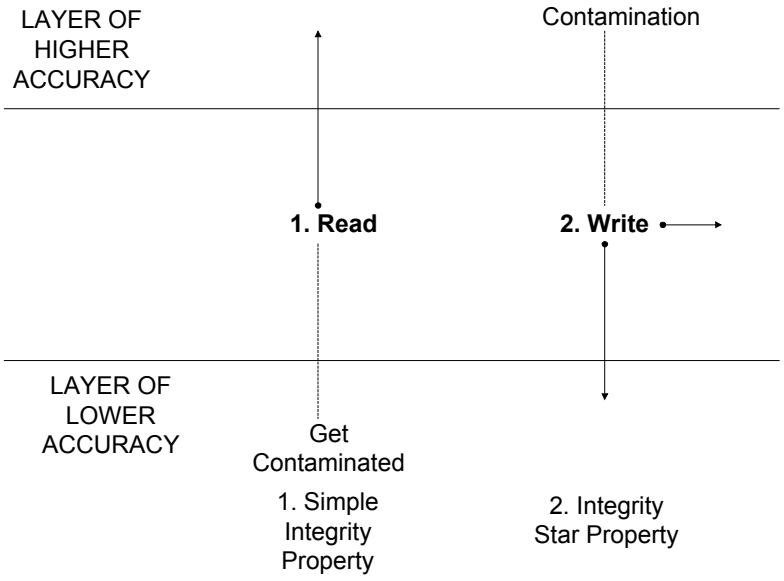


Figure 12. Biba model (CISSP, 2000)

Carl & Wilson addresses all three integrity goals (Preventing unauthorised users of making modifications, Preventing authorised users from making improper modifications and Maintaining internal & external consistency). All transactions are well formed, where users can manipulate data only in ways that ensure internal consistency (CISSP, 2000).

There are following implementations of integrity models that should be known but are out of scope of this study:

- Biba (Hierarchical Lattice)
- Goguen & Meseguer (Capability System & Domain Separation)
- Clark-Wilson (Well formed Transactions & Separation of Duty)

- Brewer & Nash (Mathematical Theory)
- Lipner (Bell-LaPadula Lattice & Biba)
- Boebert & Kain (Goguen & Meseguer – Lock)
- Lee & Shockley (Clarck-Wilson & Biba with Sensitivity Labels)
- Karger (Clark-Wilson & Lattice)
- Jueneman (Integrity Labels & Detection Oriented)
- Gong (Access Control Lists & Capability Based) (CISSP, 2000)

In the Lattice Based Access Control is defined partially ordered set of classes, where every pair of elements has a greatest lower bound and a least upper bound. Every resource and user is associated with the class.

From the first list Non-interference model compasses ways to prevent subjects operating in one domain from affecting each other in violation of security policy. State machine model is an abstract mathematical model that represents the state transitions of a system. Access matrix is a state machine model that specifies modes of access (subject-subject or subject-object). The state transitions are collected to the matrix that has user axis and resource axis where one row is for subject and one column for object.

Information flow model is also representing access modes where object-object information flow is constrained in accordance with objects security attributes. Covert channel analysis is simplified with using information flow matrix. The matrix is presented using object axis-object axis (CISSP, 2000).

In addition of these security models there are also other mathematical and formal models that can be used to analyse and design information system security deeper in the process and system call level. Formal models are usually intended to study certain security functions, for example analyse encrypted communications where trust relationships are analysed using formal methods. In addition of formal models there are various mathematical theories to prove and analyse security functions that are using encryption, using methods of Cryptography and Cryptanalysis (Schneier, 1996 C).

Security of the target system is guaranteed if formal and mathematical methods are used, but it makes the system engineering so complex process that the formal methods are seldom used to improve system security. Formal Methods should be recognised during system security engineering to provide tools for deeper security analysis if required.

4.3 Theoretical Evaluation

The last step before the actual coding work (implementation) starts, is a theoretical evaluation. This is a more thorough security assessment than the threat analysis, and it is targeted for every security function separately or in some cases for most vulnerable security functions. This step and all the following steps are similar whether the standards or structured analysis are used for performing the threat analysis.

The security features (functions) technical specifications are reviewed in the theoretical evaluation to analyze the strength of the functionality in those features. The specifications are analyzed at the same time studying how the attacker could trespass software security features. If important holes are found in the functionality then the specification must be changed to block those holes. All possible ways to compromise the system sub-functions must be evaluated. The evaluation should be done using the attacker resources, time, and dedication as a reference model. Formal models can be used during theoretical evaluation phase to provide ways to analyze security functions in the required level.

Software can never be totally secure so the evaluator has to calculate what is the probability of the penetration to the system and weight the risks according to the probability (Schneier, 1999 A). The main objective is that the security weaknesses are known, and risks related to them are not too high to compromise the system security level. Theoretical evaluation should be included as a part of the technical specifications for sub-functions. General conclusions of the results found should be added to the Threat Analysis.

4.4 Implementation

After the software and all it's sub-functions are specified, starts the implementation phase. Coding is a pretty straightforward work when the specifications and designs are well made. Coding work usually includes some code reviews, mainly intended for analyzing that the code is a well commented and follows the good programming practices. For these reviews it is practical to implement also some security checks.

Most of the good programming practice rules apply also to system security improvements. An example of this is the modularity (encapsulation) of developed program. Security aspects of code should be systematically reviewed during the code reviews for example buffer overflows can be detected by disassembling programs and looking at their operations. From the recent discussions in the security related programming in UNIX we can make some suggestions and analysis for the programming principles that can help to assure that the program is as fault free as possible:

1. **Least privilege**. Program and use the minimum sufficient privilege to accomplish the task. Ask, "What privileges does the software need?" not, "What privileges does the software want?"
2. **Economy of mechanism**. Short, simple code will have fewer bugs than long, complex code. Determine the minimum necessary to do the job.
3. **Complete mediation**. Check every access to an object, every return code from every call, and every variable value at a decision point.
4. **Open design**. Do not depend on security through obscurity.
5. **Separation of privilege**. Keep privileges necessary at different times in different routines or programs.
6. **Least common mechanism**. Users should share resources as little as possible; minimise shared resources.
7. **Psychological acceptability**. Security controls must be easy to use or users will bypass them.
8. **Fail-safe defaults**. Deny by default, and fail "closed" (without granting the request).

9. **Code reuse**. Reuse previously tested code when possible.
10. **Distrust the unknown**. Anything provided by users or from outside of the program is suspect.
11. **Anticipate problems before they arise**. Determine what security problems may arise from the functionality of your program and design to minimise these problems before you start writing the program.

(Galvin, 2000)

A good and a bad programming practices was presented first time by Simson Garfinkle and Gene Spafford in Chapter 23 of *Practical Unix and Internet Security (1996)*. The book is a valuable reference for people that are doing security related programming in Unix world. The tips presented in the book are a solid basis for programming policy to be implemented in security related software projects (Garfinkle and Spafford, 1996).

Secure programming methods:

- Check all command-line arguments.
- Check all system call parameters and system call return codes.
- Check arguments passed in environment parameters and don't depend on Unix environment variables.
- Be sure all buffers are bounded.
- Do bounds checking on every variable before the contents are copied to a local buffer.
- If creating a new file, use O_EXCL and O_CREAT flags to assure that the file does not already exist.
- Use lstat() to make sure a file is not a link, if appropriate.
- Use the following library calls instead of their alternatives: fgets(), strncpy(), strncat(), snprintf(). Generally speaking, use functions that check lengths (termination character check isn't enough).
- Likewise, use execve(), carefully, if you must spawn a process.
- Explicitly change directories (chdir()) to an appropriate directory at program start.

- Set limit values to disable creation of a core file if the program fails: a core file could hold passwords or state information that were in memory.
- If using temporary files, consider using tmpfile() or mktemp() system calls to create them (although most mktemp() library calls have problematic race conditions).
- Have internal consistency-checking code.
- Include lots of logging, including date, time, uid and effective uid, gid and effective gid, terminal information, pid, command-line arguments, errors, and originating host.
- Make the program's critical portion as short and simple as possible.
- Always use full pathnames for any file arguments.
- Check user input to be sure it contains only "good" characters.
- Make good use of tools such as lint.
- Be aware of race conditions, including deadlock conditions and sequencing conditions.
- Place timeouts and load-level limits on incoming network-oriented read requests.
- Place timeouts on outgoing network-oriented write requests.
- Use session encryption to avoid session hijacking and hide authentication information.
- Use chroot() to set program context to a subset of the system whenever possible.
- If possible, statically link secure programs.
- Do reverse DNS lookups on a connection when you need a hostname.
- Shed or limit excessive loads in network daemons.
- Put reasonable timeout limits on network reads and writes.
- Prevent more than one copy of a daemon from running, if appropriate.

Insecure programming methods to note are:

- Avoid routines that fail to check buffer boundaries when manipulating strings, particularly gets(), strcpy(), strcat(),sprintf(), fscanf(), scanf(), vsprintf(), realpath(), getopt(), getpass(), streadd(), strecpy(), and strtrns().
- Likewise, avoid execlp() and execvp().

- Never use system() and popen() system calls .
- Do not create files in world-writable directories.
- Generally, don't create setuid or setgid shell scripts.
- Don't make assumptions about port numbers, instead, use getservbyname().
- Don't assume connections from low-numbered ports are legitimate or trustworthy.
- Don't trust any IP address; if you want authentication, use cryptography. (Reverse DNS lookup provides a minimal level of assurance.)
- Don't require clear-text authentication information.
- Avoid any guessable or re-playable seed to random number generators.
- Don't try to recover from a serious error; output details and terminate.
- Bracket sections of code that require higher privilege with setuid() and setgid() functions.
- Consider using perl -T or taintperl for writing setuid programs.

(Galvin, 2000)*

** Refer also to LeBlanc's article for information about Windows based secure programming practices (LeBlanc, 2000).*

It is a worthwhile effort to compose a separate secure programming policy for programmers, to be used during implementation phase to avoid security holes in the program.

During code security review it is possible to use Code Security Analysis tools to guarantee the security level of the software and prevent practices, for example bad use of memory allocations and function calls that compromise the software security. This can be done also during unit testing, if the code reviews are not a common practice in the target product process. These tools should be used if the programmer is not a security expert. Note that code security review tools are pretty new in the markets and they do not detect all faults yet. Positive is that they are found already for many well-known programming languages. An example of these tools is a tool called “IST4: A Static Vulnerability Scanning tool for C and C++ code” (Reliable Software Technologies, 2000).

During the implementation it should be also decided if the code is treated as an Open Software Foundation (OSF). OSF is the current commercial trend to prove that the code is secure revealing it to the customers after the market release. Sometimes this is a good way to provide security assurance to customers even though they would not know anything about system security engineering.

4.5 Results from the design and implementation phases

The results of the theoretical evaluation and all previous phases are composed in to the Threat Analysis. Threat Analysis is an entry document for product security testing concerning the weaknesses already known to exist. With the threat analysis security testing knows all the aspects that have been analyzed during design and can make the decisions about new weaknesses on the basis of that. Threat analysis is also a document that can be used to demonstrate to the customers, that the system is secure enough for intended applications. Threat analysis is one of the main security documents to present for evaluators if the product is certified.

The final deadline for the TA is the E3 –milestone, but first version is composed for E1, because fixing the found vulnerabilities in the later phases of the product process would be increasingly expensive and as mentioned earlier the document is needed for security testing planning.

5. SECURITY TESTING

Security testing is a phase during system testing where the Target Of Evaluation (TOE) is tested and demonstrated to be secure. Testing can find several unknown weaknesses in the design and faults in the functionality. Therefore it is an important step toward secure system.

Security testing is composed by using a Security Testing and Evaluation Plan that will describe when and how the Penetration testing, Functionality testing, Vulnerability testing, and Quality Assurance reviews are done (see Figure 13). It includes the development groups' statement for Security Target (ST) and Protection Profile (PP) to guide the testing effort if security standards are used. The final results of these activities are included to the Security Testing Report, which is incorporated to the Certification Report, if the product is going to be certified and evaluated by Trusted Third Party (TTP) (Van Essen, 1999). The summary of security testing results is included to a System Testing Report, being a high level report of testing results for the customers and product management (CSE, 1999).

The Security Testing and Evaluation Plan describes the philosophy for testing and evaluation of the system security. It defines the strategy, which will be used to demonstrate that the security components with non-trusted components and applications do not introduce new system vulnerabilities.

The most important documents for Security Testing Plan are the threat/risk analysis of TOE and the study of expected network architecture, which is usually a part of the threat analysis. These documents are entry documents for testing phase and must be available for E1 -milestone. The system testing report is finished for E4 -milestone and certification process is started after that if applied.

Testing includes as many iterations as is necessary to reach the final exit criteria. All new iterated builds are tested until there is a "bug free" version. All tests have relative importance level that is described using numbers in the block diagrams (see Figure 13), because usually testing schedule is a very tight. It is wise to divide tests to different classes using relative importance between them to choose the tests that are performed during each

round. The time limits being too tight for all tests, it is possible to drop the least important tests to stay in the testing schedule and in the intended market window for the product release. System security testing group decides, with system security engineers, which tests are performed to evaluate the security of the TOE, if the test time limits are too tight for all tests to be completed.

System security engineers and product managers eventually decide about the certification and when the system is ready for the market release. Security testing of the product can continue after the market release in the form of Change Deliveries (CD). CDs are recommended approach to improve the system security for certification purposes because testing can continue much longer time and the new faults can be fixed in the CDs. This also shows to the customers that manufacturer is providing a good quality and improving the systems security to the best possible level.

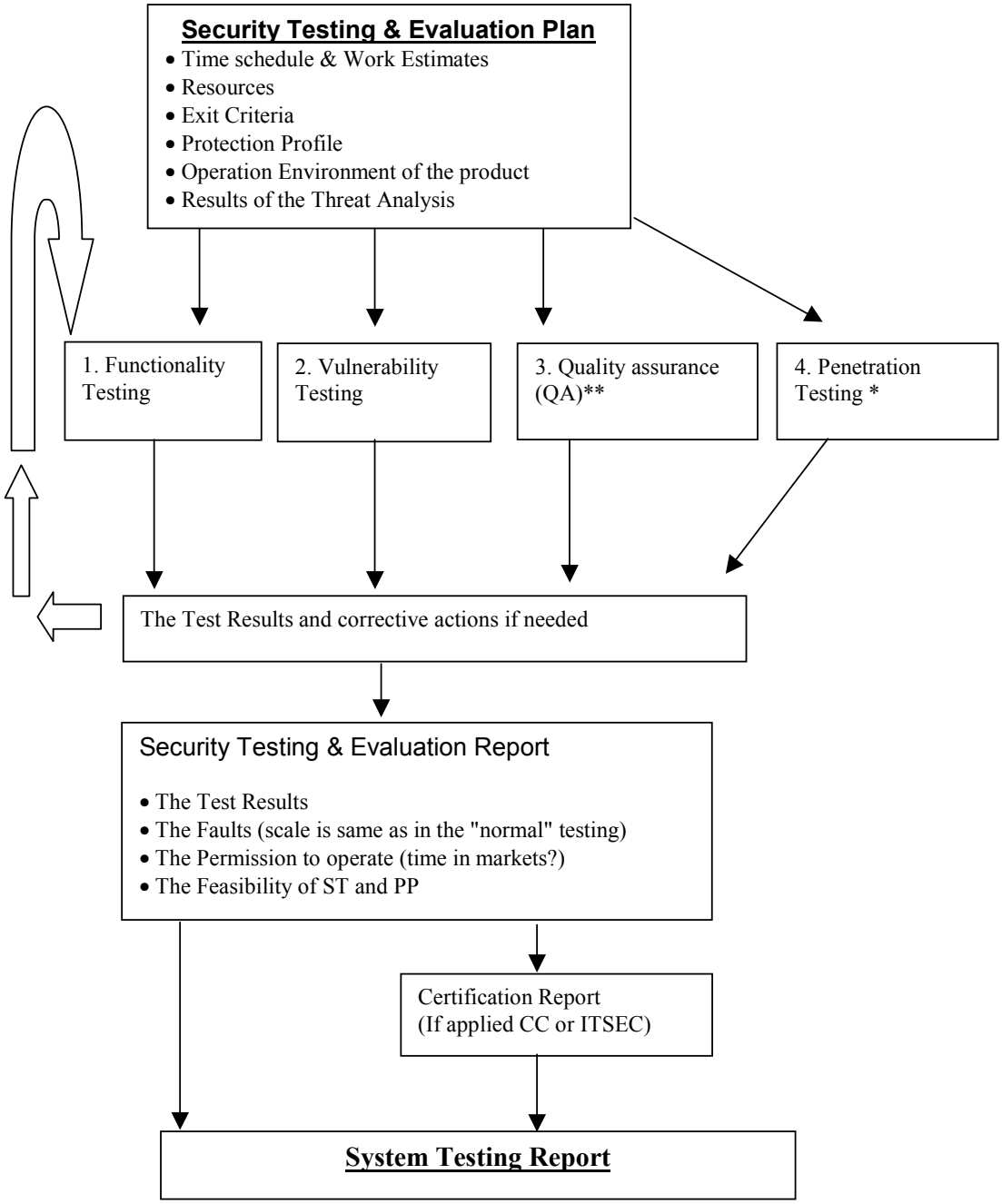


Figure 13. High-level model of security testing and evaluation (CSE, 1999.)

* Penetration tests are not required for all systems.

** The development environment and the system under development are sufficiently well controlled. Review of the security documents, development environment and process, including milestone requirements, standards, specifications and support tools is required. The evaluators/system security engineers should

follow software changes to evaluate the security implications. As security safeguards often use system resources, evaluators must be vigilant to prevent design measures that seek to improve performance by violating system security policy. Configuration management ensures that changes do not adversely affect the security of the system.

5.1 Functionality testing

The main objective of the functionality testing is to demonstrate that all the functions planned to protect the system from malicious behavior are working as specified. The functional tests will consist of at least testing of passwords, wrong use of the software, levels of access to the information, authorization and covert channels ("hidden programmer short-cuts"). The testing structure of the functional testing and the evaluation of the security functional elements of the software are presented in Figure 14.

Thorough security functionality testing includes the following functional components and classes: Security audit (Security related events monitoring), Communication (Transport of information), Cryptographic support (Cryptographic functionality), User data protection, Identification and authentication of users/entities, Security management of functions and attributes, Privacy of the users, Protection of the security functions, Resource utilization, Access and Trusted paths/channels. Usually the time limits and resources allow only tests that are presented in Figure 14.

Testing of all these components are performed if possible. Note that some parts of these components belong to unit and integration testing phases e.g. Identification of entities. All security functional tests are performed by security experts along with other functionality tests done for the system.

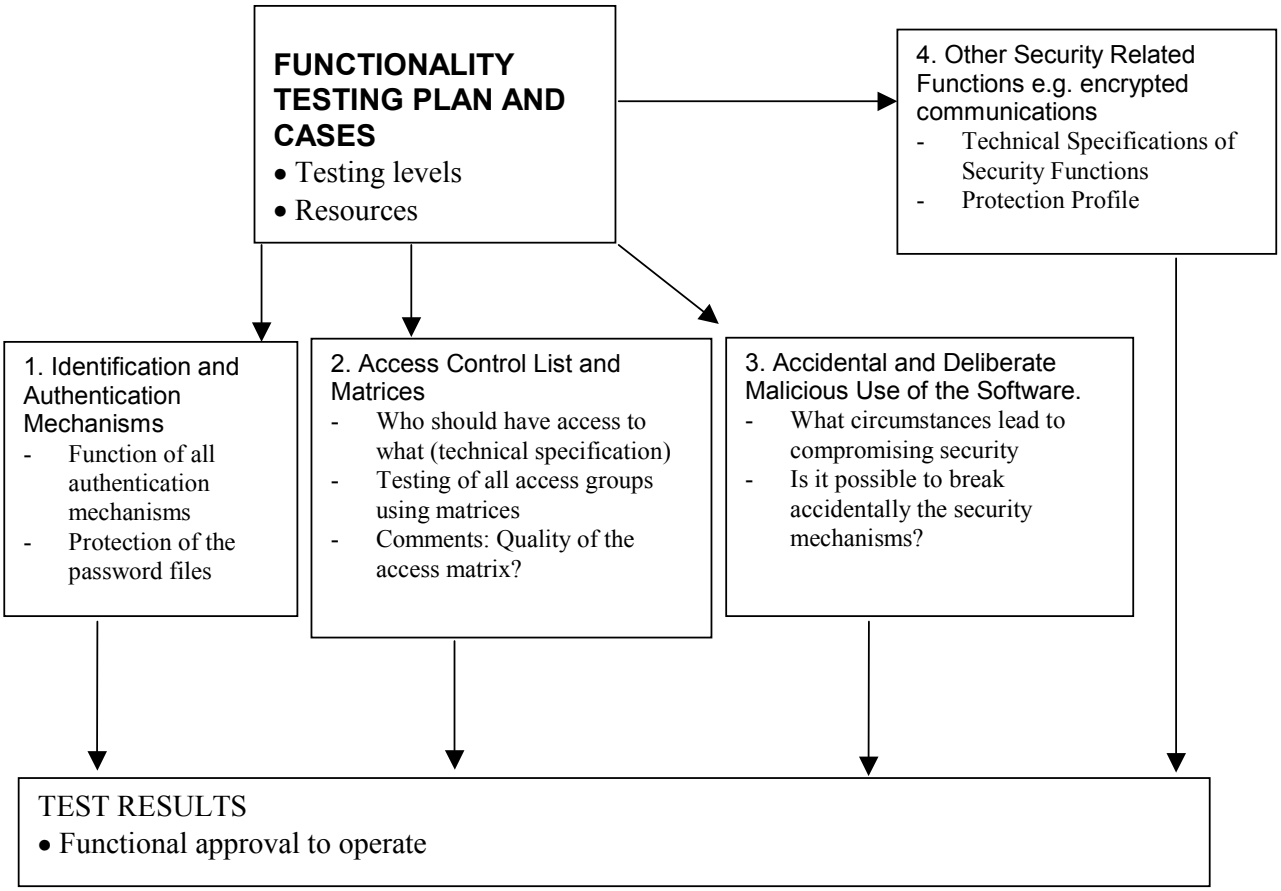


Figure 14. Testing structure of Functional Testing.

5.2 Vulnerability testing

Vulnerability tests are designed to test the security functions and system configuration a little bit deeper than functional tests and are committed during system validation or system testing phases. The goal is to demonstrate that security functions of system are strong enough. Vulnerability testing includes the set-up configuration, encryption, physical security and hardware permission devices (see Figure 15).

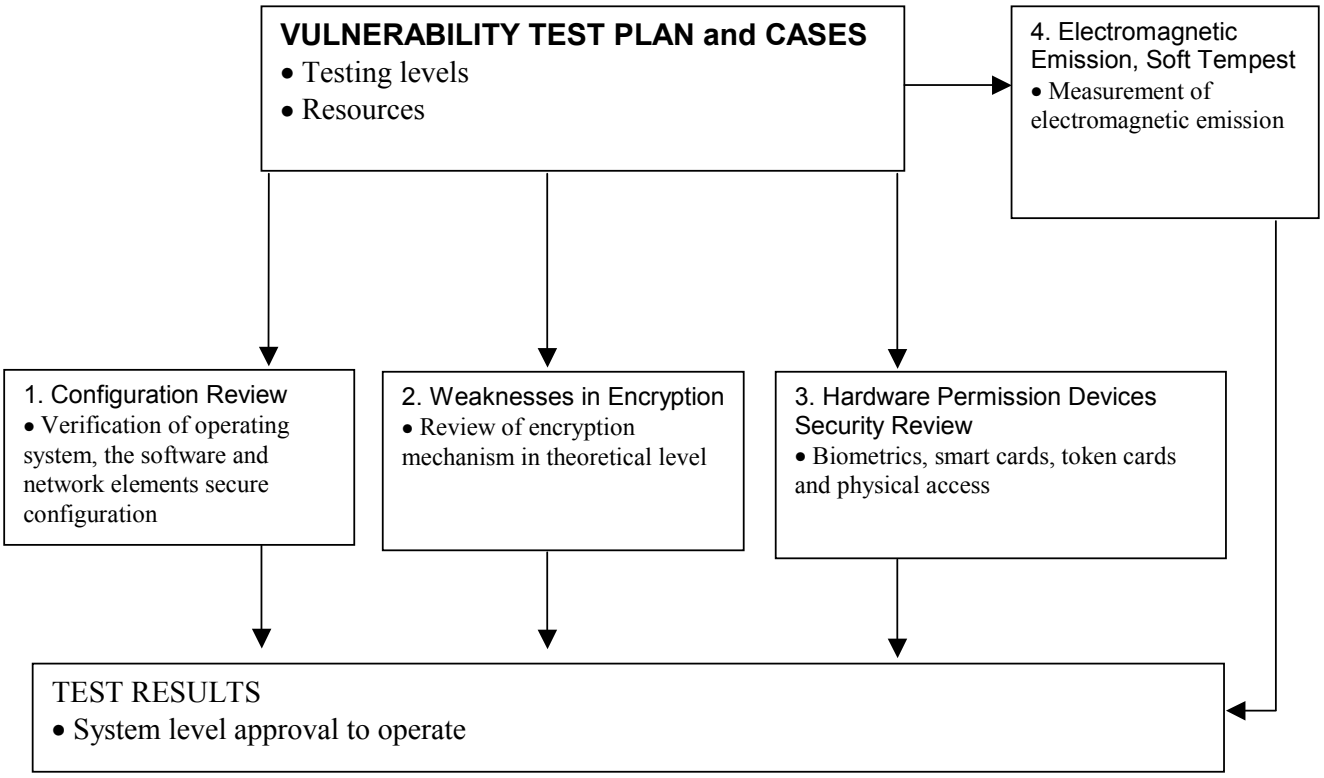


Figure 15. The testing structure of the Vulnerability Testing.

5.3 Quality Assurance

Quality Assurance (QA) will include: factors that are affecting security in the development environment, the review of technical and customer documents, and the code review conclusions from the development. The objective is to summarize internal security controls, and review that the security of the product is not compromised during development and distribution process, and it is reasonable easy to operate the software in a secured manner. The main reason for this phase is to collect material for certification process, improve overall quality and improve security procedures in the product process.

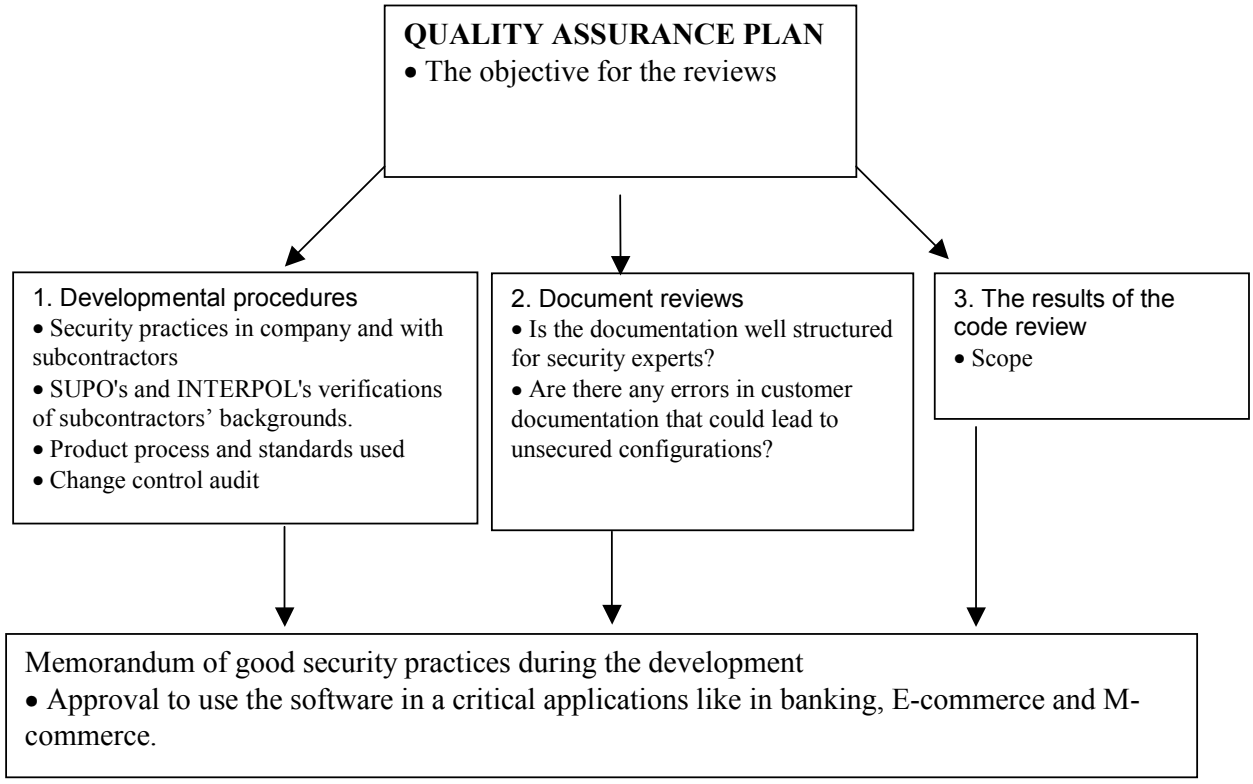


Figure 16. The structure of the operational controls evaluation.

5.4 Penetration Testing

Penetration tests should include some of the most common hacker techniques, hacker tools, and other methods that can be used to compromise the system. For this reason it is important to follow phreaking (phone line hacking) and hacking trends around the world. Testing is composed of similar test cases than in other security testing phases. The objective is to break in to the system and record the steps to the Penetration Study (PS) document.

A good approach for penetration testing is to test the software using the same methods as crackers use:

- Try to overflow every buffer in the package

- Try to abuse command-line options
- Try to create every race condition conceivable
- Have someone besides the designer and implementer review and test the code
- Read through the code, thinking like a cracker, looking for vulnerabilities

(Galvin, 2000)

The results of the threat analysis, theoretical evaluation and code reviews can also be used when planning test cases. The test cases are quite similar to the test cases in the functional security testing. For example testing of access mechanism using brute force penetration testing is a quite similar to functional testing of access mechanisms. The main difference is that the penetration tests are performed and targeted for certain pre-defined system functions. The objective is to break in to the system using realistic environment with all documents and technical aids possible. Some of the penetration tests might be performed in the customer premises during α-testing to gain as realistic environment as possible.

When the system is in the real use the security faults found are called "Penetrations" and steps recorded to PS similarly as in penetration testing to provide enough information for development to fix the problems.

5.5 Security Testing tools

Security Testing is a time consuming and difficult effort and that is why security testing needs responsible persons for developing new tools, handling purchasing procedures and managing the version control of the applied security testing software tools. The version control is especially important, because some times a former hackers are used for security testing purposes and for the operational security point of view, the usage of security critical applications must be well controlled. Usually software manufacturers who have a high

capability maturity model level (CMM) have a separate testing tools support group. The security testing tools support should be included to this working group to provide help with testing tool issues and control the tools used. These tools can be also used during Threat Analysis to help the system security engineers to design the system security functions. Because of this it is advised that, the expertise in security testing tools support will be used by both the development and the testing group.

There is currently a large amount of different tools that can be used for security testing purposes. These tools can be divided in to a system administration and hacker tools. System administration tools include system scanners, port scanners (Internet scanners) and system vulnerability assessment tools. These tools are good for testing system security in the operation system platform level. Hacker tools group includes various password cracking, buffer overflow, denial of service and vulnerability scanning software. In addition of previously mentioned tools, own special purpose build tools can be developed for Penetration Testing purposes.

For penetration testing purposes there are several different hacker tools that can be used to demonstrate the insecurity of the target system. The most famous penetration testing related attacks are network probing attacks, vulnerability scanners, buffer overflow attacks large packet ping attacks (ping of death), Denial of Service attacks (DoS), SYN attacks, teardrop attacks, Smurf attacks, IP spoofing attacks, TCP sequence number attacks and IP fragmentation attacks. Some of security functional testing and regression testing can be also automated by using special purpose build software that tests predefined data flows in the functional level.

5.6 Security Testing and Evaluation Results

Results of all security tests are composed in to Security Testing Report. It should be noted that the report is classified as a Secret and should be given to system engineers using a need to know basis. This is because company's own employees and subcontractors will

introduce the biggest threat to the finished product. To guarantee that results do not leak to malicious outsiders, utmost care should be practised with the vulnerabilities found from the system. The classification of errors/defects follows the classification presented in Table 3.

Table 3. The classification of the faults found in the software (Tiwana, 1999; Tipton, Krause, 2000).

Fault class	Impact
A	**Immediate Repair / Critically Heavy Impact** Preventing essential operation of the network repeatedly or continuously resulting critical danger in maintaining the network in operational conditioning. No approval to operate. • Constant danger of malicious use • Complete accessibility from a remote host • No trust that the data could not be modified, destructed, fabricated, disclosured, interrupted or stolen during processes • Large number of subscriber complaints • Large number of losses in business image, charging, keeping system operational etc. • Design errors
B	**Critical Fault / Heavy Impact** Prevents or severely hinders the operation of an essential functioning of Subsystem or a system level feature. No approval to operate. • Increase in privileges and access • Incorrect functionality of feature that is commonly used by customer • Impossible to track attackers, some monetary losses and image losses • Traffic logging is lost

C	**Important Fault / Moderate Impact** Affecting but not preventing normal operation of the network. Performance or operability of the system is affected, or a malfunction in an operational feature. Approval to operate in the markets certain predetermined time if requested. • Denial of service • Service degradation • Incorrect functionality of feature having minor impacts • Incorrect function of feature not used by customer yet • Incorrect functionality of feature not detected by customer or end user • Errors in logging and alarming of malicious use
D	**Minor Fault / Minor Impact** Minor faults, some effect on operation or service quality. Some effect on functionality or service. Operational approval. • Documentation errors
E	**Not Important Fault / No Impact** Minor fault, not affecting operation or service quality. No effect on functionality or service. Operational approval.
F	**Enhancements** • Enhancements to an existing feature, user interface improvements • Proposal for a new feature

When classifying faults special consideration has to be practiced. All faults found should be classified using possible real life implications to the applications that the system is intended for. The software must be left maximum of one C -level fault and all the others must be D or below. A good approach is to define a certain limits also for lover level faults. Before the defined exit criteria is reached, software does not get permission to a market release and new testing rounds are required.

The summary of all faults found is presented finally in the System Testing Report. The faults are presented in the detail that is required to present results, and same time to prevent

outsiders to understand the possible ways to break in to the system. Error (a bug found by testers) and defect (a bug found by customers or third party) ratio should be counted to define the quality of test cases and the efficiency of security testing.

6. CODE REVIEW

After final run of all tests, when the product is frozen (E3-E4), is time for an operational code review. The Code Review is performed after the product is frozen. This is because it is the only way to prevent developers and subcontractors freezing a wrong version for the reviews and inserting malicious code e.g. back doors and logic bombs to the software after the reviews. The code review is usually performed just for security subsystems, because the software is already scanned in implementation phases using vulnerability scanning tools. It would be impractical to review the whole system including several million lines of code. This phase is usually responsibility of the development security experts or the trusted third party.

The operational security code review is required only for very critical applications for example the launch system of a nuclear weapon. The results of the code review are included in to the quality assurance part of security testing report.

7. CERTIFICATION PROCESS

When the product reaches E4 milestone, it is the time to start the certifying process, if the product is planned to be certified. Certification is usually a costly and a time-consuming process, and it is only performed, if it increases the value of the product, if the customers require it, or QFD shows that manufacturer can get marketing benefits from the

certification. There are several certification schemes to follow, but we mainly concentrate to the Common Criteria in this document because it is currently the leading certification scheme.

The CC certification provides a third party assessment for the level of confidence for customers. In the other words how much trust customers can have for the security functions of the IT-product or system. If the product and process fulfils the requirements presented in the CC, a certifying body grants a quality certificate for the product. After that the product is listed in the certified products list that is administered by ISO and certification bodies. Parties belonging to the evaluation process include:

- DEVELOPER/CUSTOMER: sponsor of system (TOE) being developed
- DEVELOPER: developer of system (may be sponsor)
- EVALUATOR: evaluation facility which performs evaluation, in Europe following commercial evaluators (1.11.1999):
  - Admiral Management Services Ltd.
  - EDS Ltd.
  - IBM Global Services
  - Logica UK Ltd.
  - Syntegra
- CONTROLLER: certification body (in Europe - ITSEC) (ITSEC, 1999)

The CC provides a common set of requirements for the security functions of IT products and systems (functionality). The level of confidence in the security functions is assessed by evaluators (assurance). Evaluation consists of following parts:

- Functionality works as specified
- Specification of layers are correctly instantiated
- Effectiveness / Penetration testing
- Secure development

The scope of CC evaluation is 1) All IT products and systems with IT security functionality, 2) Hardware and software where the focus is protection of information from unauthorised disclosure, 3) Modification, and 4) loss of use. CC is not applicable for evaluation of:

- Compromising electromagnetic emanations
- Non-IT measures, e.g. procedural regulations
- Legal and administrative framework
- Strength of cryptographic functions

That is why the attack trees and QFD are required for the best quality.

The evaluation process may be carried out in parallel with development process, or it may follow the development process, as described in Figure 17. As we can see from the figure the usage of CC starts by defining Protection Profile (PP) for the product. Protection Profile is a non-classified document that can be used for defining security properties for similar products. If there is not any similar product PPs, one must be prepared.

PP will include the general description of the product and the security requirements for the product (Functional and Assurance requirements from CC) and the intended operational network architecture. PP is prepared during E-1 – E1. After the preparation of PP the Security Target (ST) for product must be defined. This is done as presented in CC standard. ST is quite similar than PP but it is done in a much deeper level than PP and is intended for just one TOE (the manufactured and evaluated product), not several as PP. ST is classified as a Secret.

The evaluation of TOE means that the TOE is evaluated using ST's requirements. Every part of the evaluation (PP, ST and TOE) is verified by Trusted Third Party (TTP) to get assurance that the objectives are met (Veteläinen, 2000).

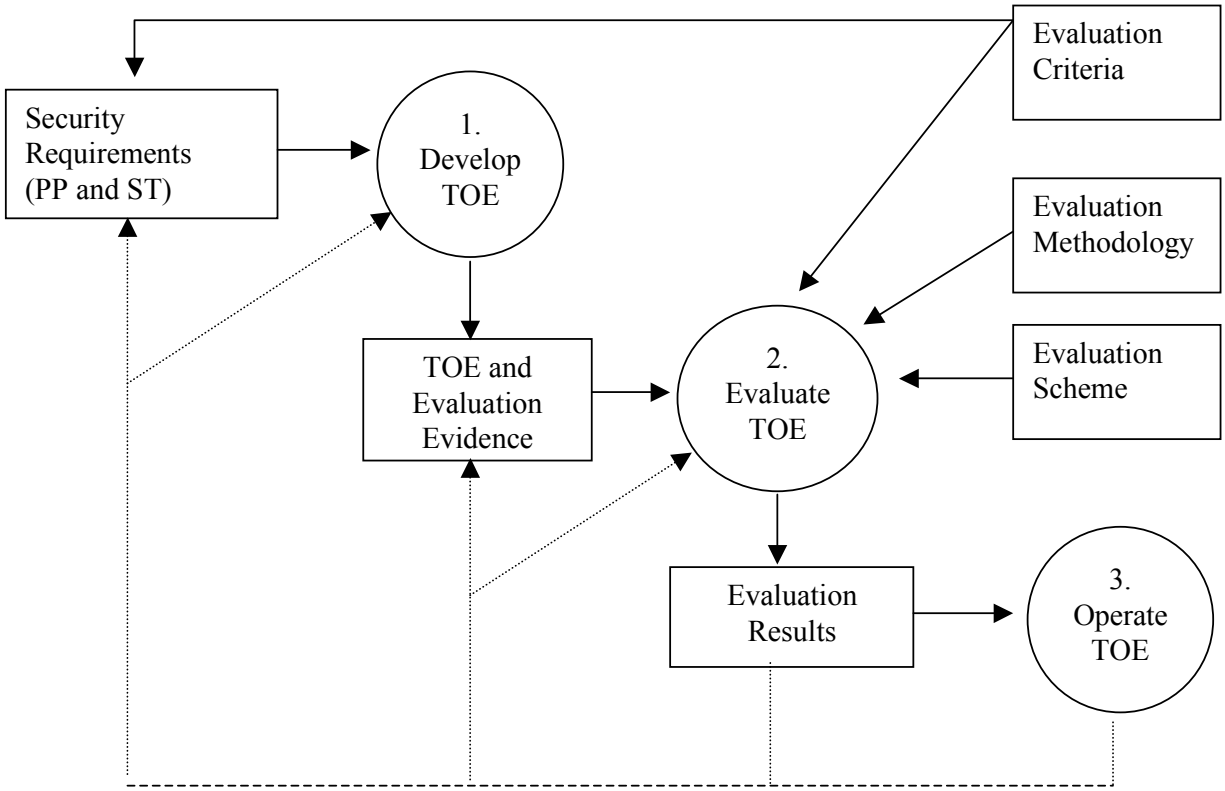


Figure 17. Certification process (Van Essen, 1999).

As we have earlier identified, the customers are starting to demand certified products for critical applications. That is the main reason why the Product Process model described in this document is composed using the same phases and steps as are required for the process in the TTP evaluation and certification.

7.1 Certification as a value adding factor in the buying decisions

Recently there has been a lot of debate about the certification as a value-adding factor when marketing products. The picture that commercial third party evaluators give from the certification is a little bit too optimistic, because certification means revenues for them. Our experience is that certification is required and appreciated more often when the customers are government agencies, banks or similar parties. Other customers usually

appreciate the certification as a proof of secure product, but are not very often ready to pay the added costs and delays that the certification causes.

Certification process is a time consuming and relatively costly process and that is why it is advised, that products to be certified only if the marketing benefits are outweighing the added costs. The quality of product does not substantially rise and benefit from the certification or at least it should not. Certification is only a proof for customers that the process meets certain security requirements. For customer there are following benefits to buy certified products:

- Assurance level of component is given
- Components with similar assurance level can be chosen to co-operate
- Traceability tests gives less risk for malfunction
- Security functionality is carefully documented

As mentioned earlier it depends greatly of market segment whether to gain benefits from the TTP certification. That is why certification decisions should be made case-by-case basis.

7.2 Conclusion from the usage of Common Criteria

The usage of security standard Common Criteria seems to be a good way to improve the product line quality but it is a relatively time consuming method and greatly depends upon the expertise of the people that are following the certification scheme. It is also a relatively hard standard to follow because of a poor usage of terms and abbreviations. This might be due of the commercial application of the certification. Though CC gives a good information of what security functions to implement to the product and what methods to use to get assurance that the TOE functions as specified and the product process is well formed for system security engineering purposes (Veteläinen, 2000).

CC is not a perfect method for system security engineering and that is why it requires some additional security methods to improve the product and process quality. An example of these additional methods is the product process that we have introduced here to include Threat Analysis and systematic Security Testing for the TOE when certification standard CC is used. SSE-CMM provides a good guideline for system security engineering to be used with CC. Even though CC defines some good practices to be used during system security engineering. For example, in the Assurance Requirements part of the standard, CC defines for example the delivery procedures for security or "secured" product. This is not mentioned in SSE-CMM. That is why the best results are achieved by using both of these methods.

The concept of security level is also a relatively good way to categorise system security design procedures for information systems. Because of these good sides of the CC, it is a worthwhile standard for some applications, but maybe not the best one for new products that do not have existing PPs, because of the PP formulation effort. Though in some cases when the system is intended for very critical applications it might be useful for a market leader to define the PP and certify the system and with this approach do the life of the smaller competitors harder, because of the difficulty of meeting the certification requirements.

8. CONCLUSIONS AND RECOMMENDATIONS FROM THE PILOT PROJECTS

As described in the previous chapters there are two different methods to commit design of information system security. During the research phase for this document we had two different pilot projects in Nokia Networks Ltd./Messaging and Service Platforms for System Security Engineering. Both of these projects were quite similar system engineering projects for information systems intended for telecommunication systems. One was a security critical application intended for wireless banking and other was a wireless multimedia application where security concerns were much lighter than in the first one.

Common Criteria and design without standards were combined in the first project. The application of Common Criteria for this system was studied in the form of graduate study of engineering prepared by Altti Veteläinen. Structured Analysis (SA) using checklists was done in addition of this. The results of the project demonstrated that using security standards during designing is a very time consuming related for security analysis with attack tree and other methods. It is also quite hard to show any real benefits from security standard approach related to the SA.

Security Testing in this project was performed by testing the Security Modules separately to proof the correctness of security features. Study of the Security Modules demonstrated that system was secured in the functional level. All faults found were fixed during testing. Some vulnerability tests were also performed. This revealed the biggest security holes in the form of system configuration. The threat analysis was considering this aspect, but because of limited resources some actions were not performed and it resulted to a less secure configuration. Common Criteria did not reveal any configuration-related weaknesses. Quality assurance was committed in the form of specification and document reviews. This was not done well enough, and it resulted to previously explained configuration problems. No penetration tests were performed because of the limited resources. To conclude Security Testing revealed additional security holes in the target system and that is why it was beneficial for the project.

Security considerations were also important factors when wireless multimedia system was developed. Main threats there were Denial of Service (DoS) attacks. Threats were identified using Threat analysis done by Structured Analysis (SA) approach. Attack tree and check lists were combined to get the best results. Threat analysis revealed the main threats for system and identified places where special consideration must be practised. Threats were further analysed using risk analysis method during module specification phases. This resulted to so secure product that it was really hard for security testing to find any serious weaknesses. Security testing was performed only in functional level and more weight was put on Quality Assurance (QA) steps. This approach demonstrated the importance of specification reviews and systematic design for system security.

To conclude the results found in the both pilot projects it was clearly shown that if we put more emphasis on the design of secure system the results is always better than to put the same amount of effort in to the security testing. It was also found that good approach is to have a designer and a tester to be responsible of security quality issues in the system development. If the responsibilities are not well defined, System Security Engineering does not provide acceptable results, because of the commitment and responsibility issues.

9. SECURITY IN SUPPORT FUNCTIONS

Special consideration must be practised always when developing and marketing secured products. The production environment of the software must be secured if the product is intended for critical applications like banking and e-commerce. Also the marketing arguments of the information system must be well planned to support the system security engineering effort.

9.1 Operational security controls

Operational security controls must be well planned or there will be left a possibility that some insider, for example a sub-contractor or disgruntled personnel, will introduce new threats to the product. It is easy to understand that trusted systems must be developed by trusted individuals and without this trust the software can not be totally reliable.

British Standard 7799 (BS 7799) is intended for improving and evaluation security level of the development environment. Some customers might ask conformance statements to the operational standards like BS 7799 if the product is intended for TOP SECRET or very critical applications (See Appendix C for more information).

According of static's made about security breaches in company, company's own workers and subcontractors are introducing the biggest threat for the operation, because of this

reason, the operational security standards are an internal part of the system security engineering for critical applications.

Change control procedures are also an important part of operational controls and operational security controls. Change Controls covers system reliability aspects of information security and defines, that control mechanisms must be in place to control all changes to the production programs and to produced information system. Changes must be authorised, tested, and recorded. Changed systems must be certified and accredited, programmer changes must be made through controlled libraries, policies must be in place for change control and even vendor-supplied changes must be reviewed.

Program change controls are:

- Applications are centrally developed
- Change control procedures are reviewed during system audit
- Procedures to change production systems (aspects that must be controllable):
  - Programmer changes source code on test version
  - Program is tested with test data
  - Program is certified and accredited
  - Code is updated by system personnel
  - Right versions are freezed for reviews
- Programmers should never had access to the production data

Change control process includes following steps:

1. Change Request
2. Change Request Analysis
   - Implementation strategy
   - Cost of implementation
   - Security implications
3. Change Request Recording
4. Submit Request for Approval
5. Develop Change

- Make software changes
- Record changes and link to the request
- Submit software for quality approval
- Repeat until quality is adequate

6. Make version change

The change control system and policy must be in place to provide required reliability for software development process (CISSP, 2000).

In some projects it might be a good approach to have a separate plan for internal controls, the plan should state all relevant aspects of internal security controls and give guidelines for control procedures during product development such as subcontracting procedures and contingency planning.

9.2 Marketing of secured information systems

Marketing is also one of the aspects that must be carefully planned if manufacturer is selling information systems for very critical applications. Words like Secure and Tamper free etc. should not be used if the security level of the product is lover than EAL 6 (CC), because these words can be used by unhappy customers and competitors in a possible compromise situation. Over-promising may lead to the troubles. Although the marketing must be able to make market differentiation using security features for the marketed software product with the help of QFD results (strong marketing arguments).

The documentation is also a special field of marketing where to pay attention because from the documentation customers will get the picture of how secure the system is and what has been done to provide the secure system. Documentation must give assurance for customers that the system functions are as specified, otherwise the whole system security engineering has been a waste of time. One good way to give this assurance for customers is to provide Trusted Computing Base (TCB) documentation where all security related aspects of the

information system are explained. This document could be provided for marketing purposes even if security standards would not be used. System security engineers should take part of reviewing this document to prevent risky errors.

10. SUMMARY

On this document was presented the whole production process of developing secure systems with security standards and without standards. The development approach used with standards was to follow Common Criteria (CC) for IT Security Evaluation standard. CC is the main standard to be used in the future to certify computer systems, because the recognition for international standard in ISO and that is why it was used as a baseline for security standard portions of the document.

The markets are demanding more secure systems and same time introducing a critical question how to assure customers that the developed products are secure. To answer this requirement there are several methods and standards as seen in APPENDIX C. The main question is, which standards to choose, how to effectively implement them and when standards should be used. This work did identify all relevant phases to develop secure information systems and did provide enough information for people that are implementing system security engineering product processes.

The security standard portfolio should be build using recent security standards, for example by using combination of Common Criteria for product design, SSE-CMM to build the right development process structure, and BS 7799 to certify the operational procedure controls. The level of system security engineering should be always decided using business information gained from analysis such as Quality Function Deployment (QFD). Further study is recommended of the methods and standards currently available using Appendix A. links and APPENDIX C information as a reference material for choosing the combination that fits best for improving the product security engineering process model in the target product line and organisation.

References

ANASTA project, "Asiakastarpeista menestystuotteiksi ja –palveluiksi", http://anasta.verkonmerkki.fi, Finland, 20.4.2000

Australian Defence Force Academy, Information Security Course: Cryptography and Computer Security 3, http://www.cs.adfa.edu.au/teaching/studinfo/csc/, University College UNSW, School of Computer Science, Australia, 1.6.1999

BSI, ISO TC/176/SC2 Home Page, http://www.bsi.org.uk/iso-tc176-sc2/, 23.2.2000

Certification course for Information System Security Professionals (CISSP), Security Architecture and Models module, Security Models, ISC2, Finland, 30.3.2000

Communications Security Establishment, Canadian Trusted Computer Platform Evaluation Criteria (CTCPEC) MG-4, http://www.cse.dnd.ca/, Canada, 1.6.1999

Galvin Peter, The Unix Secure Programming FAQ, http://www.sunworld.com/sunworldonline/swol-08-1998/swol-08-security.html, USA, 29.5.2000

Garfinkel Simon & Spafford Gene, Practical Unix and Internet Security, 2nd edition, O'Reilly & Associates, USA, 1996

Huhantti Hellevi, Finnish System Development newsletter 'SYTYKE', article 'Tietoturvallisuus järjestelmäkehityksessä', Finland, 1998

ISO/IEC 15408-1,2,3, Common Criteria for information security evaluation (CC), Information technology - Security techniques -- Evaluation criteria for IT security, assurance requirements part 1,2 & 3: 1999 (E), ISO/IEC JTC 1/SC27 N 2163, 18.12.1998

ITSEC certifying body, Information Technology Security Evaluation Criteria, www.itsec.gov.uk, 15.5.1999

Lappeenranta University of Technology (LUT), CD-ROM "Asiakastarpeista menestystuotteiksi ja –palveluiksi" (From the customer requirements to the leading products and services), Finland, 1999

LeBlanc David, Writing Secure Code, http://www.ntsecurity.net/go/loader.asp?id=/columns/seccode/default.asp, USA, 28.5.2000

National Institute of Standards and Technology (NIST), Computer Security Resource Clearinghouse, http://csrc.ncsl.nist.gov/, 23.2.2000

Reliable Software Techniques, ITS4: A Static Vulnerability Scanning Tool for C and C++ code, http://www.rstcorp.com, United States of America, 20.3.2000

Pressman Roger S., Software Engineering: Practitioner's Approach, McGraw-Hill, United States of America, 1997.

Schneier Bruce, Attack Trees, http://www.counterpane.com/, 1.11.1999 (A).

Schneier Bruce, Secure System Engineering Methodology, http://www.counterpane.com/, 1.6.1999 (B).

Schneier Bruce, Applied Cryptography, John Wiley and Sons, United States of America, 1996.

Tipton Harold F. and Krause Mickey, Information Security Management 4th edition, Auerbach, United States of America, 2000

Tiwana Amrit, Web Security, Digital Press, United States of America 1999

U.S. Department of Defence, Trusted Computer System Evaluation Criteria ("Orange book"), http://www.multics.demon.co.uk/orange/summary.html, United States of America, 1.6.1999

Van Essen Ulrich, Data Security '99 International conference, IT Security Evaluation and Standards, German Information Security Agency BSI., Finland, 14.10.1999

Veteläinen Altti, Graduate study in engineering, Security of a WAP Gateway (NAMP2.1 WAP) and Applicability of ISO15408, Finland, 2000

APPENDIX A. Related websites

Common Criteria for IT security evaluation (1.12.1999):

http://www.cse-cst.gc.ca/cse/english/cc.html

http://www.scssi.gouv.fr

http://www.bsi.bund.de/cc

http://www.tno.nl/instit/fel/refs/cc.html

http://www.radium.ncsc.mil/tpep/

http://www.iso.ch

National Security Agency (Rainbow series, 1.12.1999):

http://www.fas.org/irp/nsa/rainbow.htm

SFS ry (1.12.1999):

http://www.sfs.fi/

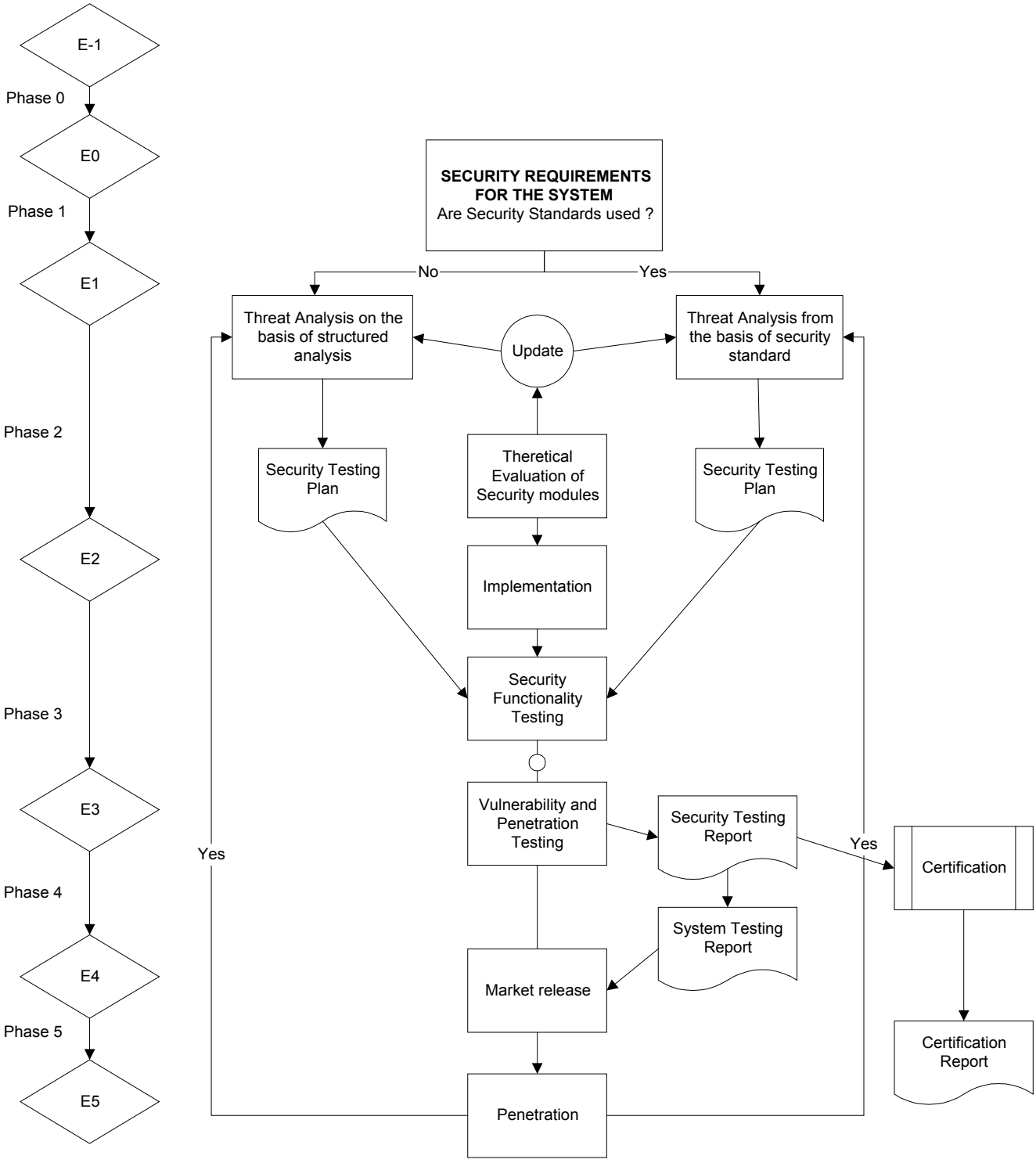
Information Technology Security Evaluation Criteria (ITSEC, 1.12.1999):

http://www.itsec.gov.uk/

SSE-CMM (1.12.1999):

http://www.sse-cmm.org

APPENDIX B. System Security Engineering process model

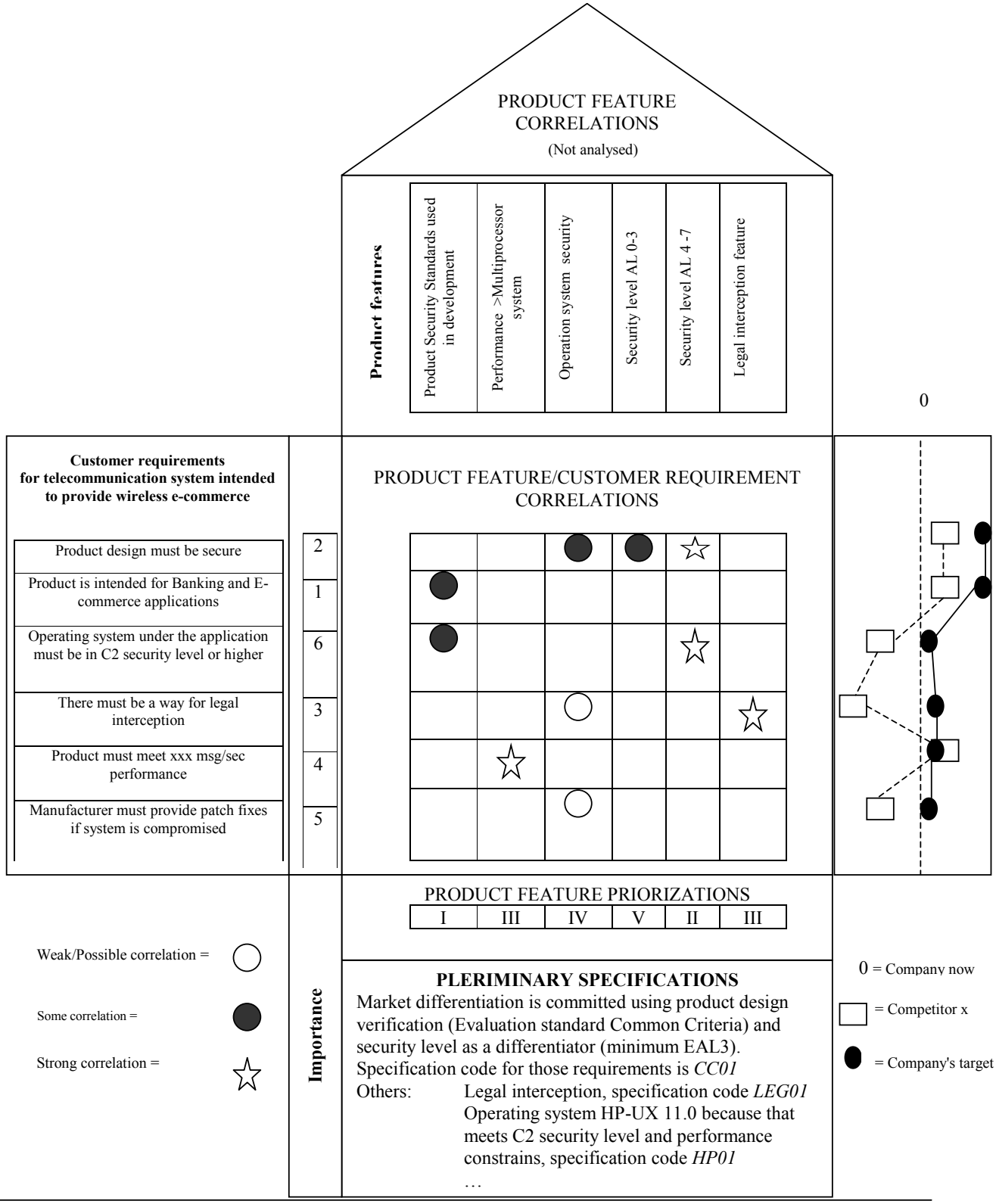


APPENDIX C. Security Assurance Methods and Standards.

Assurance Approach	*Assurance phase* **DESIGN/DEVELOPMENT ORIENTED ASSURANCE**	*Assurance phase* **OPERATIONAL ASSURANCE**
PROCESS APPROACH TO ASSURANCE	• SSE-CMM (System Security Engineering – Capability Maturity Model) • TCMM • Developer's Pedigree • Warranty Assurance • Supplier's declaration • Evaluation Rating Maintenance • ISO 9000	• SSE-CMM • Personal Assurance • GMITS • BSI Code of Practice • BS 7799
PRODUCT/SYSTEM APPROACH TO ASSURANCE	• CC/CEM • ITSEC/ITSEM • TCSEC • CTCPEC • Conformance Testing • Penetration Testing • X/Open branding • Personal Assurance • GR 815 • SECMAN Australian Defence Force	• Personal Assurance

(Van Essen, 1999)

APPENDIX D. An example of QFD analysis for system security engineering.



APPENDIX E. QUALITY FUNCTION DEPLOYMENT AND ADDITIONAL TOOLS

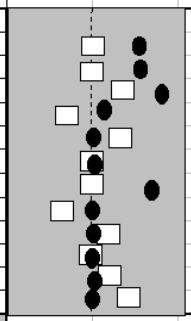
X = strong correlation
O = correlation
o = weak/possible correlation

Customer requirements	**Product features**	Electronic Code Book encryption	Cipher Block Chaining encryption	HP-UX 11.0 in trusted mode	HP-UX 11.0	Sun	SSLv3 connection feature	Multiprocessing	Lawful interception feature	SSE-CMM process model	Common Criteria standard	ITSEC product standard	ES 7799	Unassured	Functionally tested	Structurally tested	Semiformally designed and tested	Methodically tested & checked	Methodically designed, tested & reviewed	Semiformally designed and tested	Semiformally verified design & tested	Formally verified design & tested
Product intended for safety critical applications	1									X	X	o			o	o	O	O	O	X	X	X
Product intended for e-commerce	1									X	X	X			o	o	O	O	O	X	X	X
Operating system security C2 or higher	3			X																		
Possibility for lawful interception	2								X													
Encrypted communications	4						X															
Evaluated and certified product	5										X	X										
System Security assurance	2									X	X	O										
Operational security assurance	8									o			X									
Strong resistance to outsider attacks	7		O	X			O								o	o	O	O	O	X	X	X
Minimum performace x msg/sec	4	X	o					X														
Priorization before legal requirements		**V**		**IV**			**V**		**III**	**I**	**I**									**II**		
Electronic espionage act			o				X									o	O	O	X	X	X	X
Electronic privacy act				X			X									o	O	O	X	X	X	X
Product liability law																						X
Piriorization		**V**		**III**			**IV**		**III**	**I**	**I**									**I**		
SSE-CMM used in design										x												
Common Criteria used in design (target EAL 5)											x									x		
Lawful interception feature									x													
HP-UX 11.0 operating system in trusted mode				x																		
Secured web connection with SSLv3							x															
(ECB algorithms only)		x																				

Formally verified design & tested

0 level

Marketing arguments



x01	Operating system is not secure enough for intended application	1	**C2 operating system**
		3	**TTP evaluated system design**
x02	There has been penetrations during product life cycle	2	**Tested security**

0 = Company now

= Competitor

= Company's target

PLANS AND SPECIFICATIONS

SSE-CMM

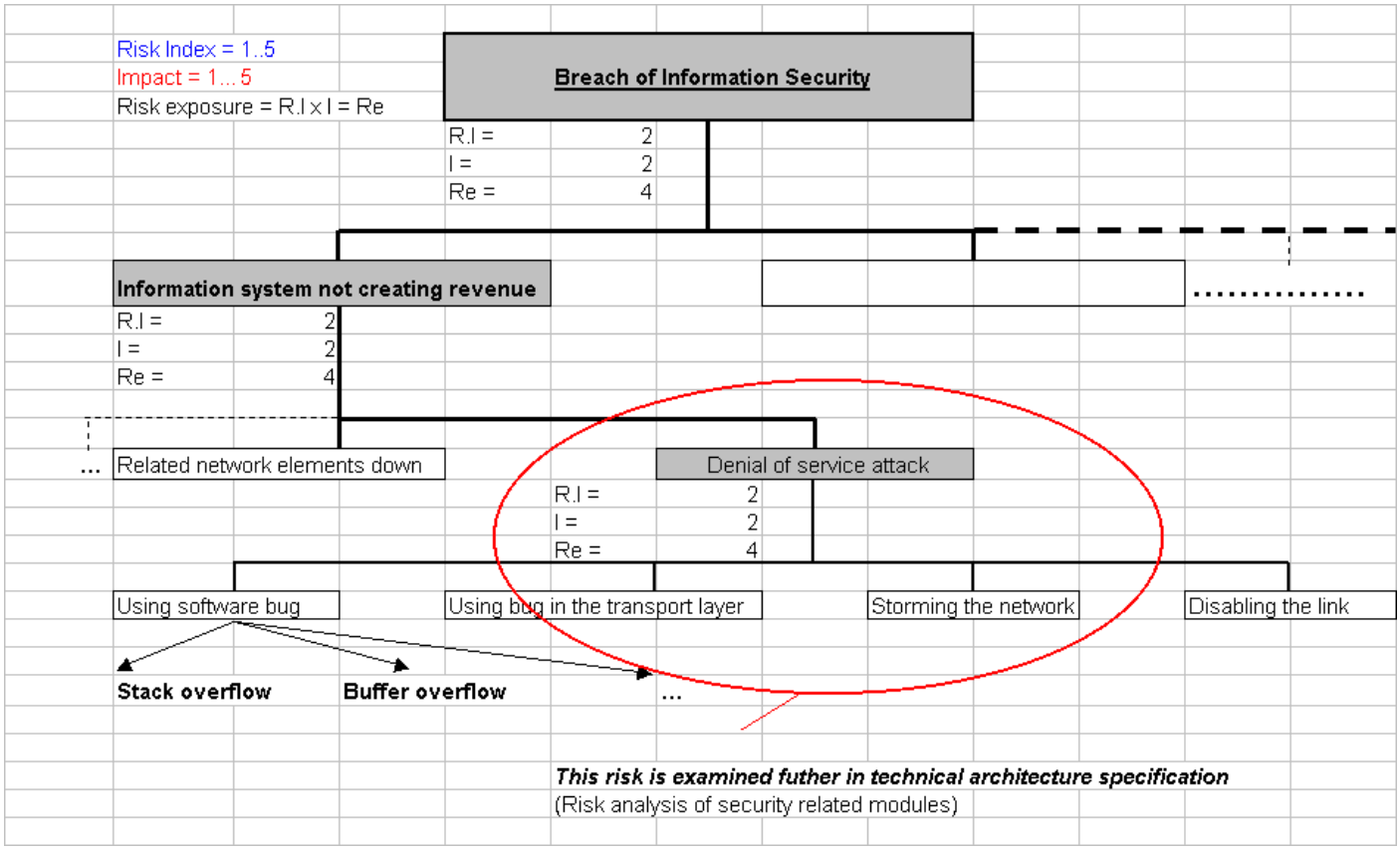
CC

law

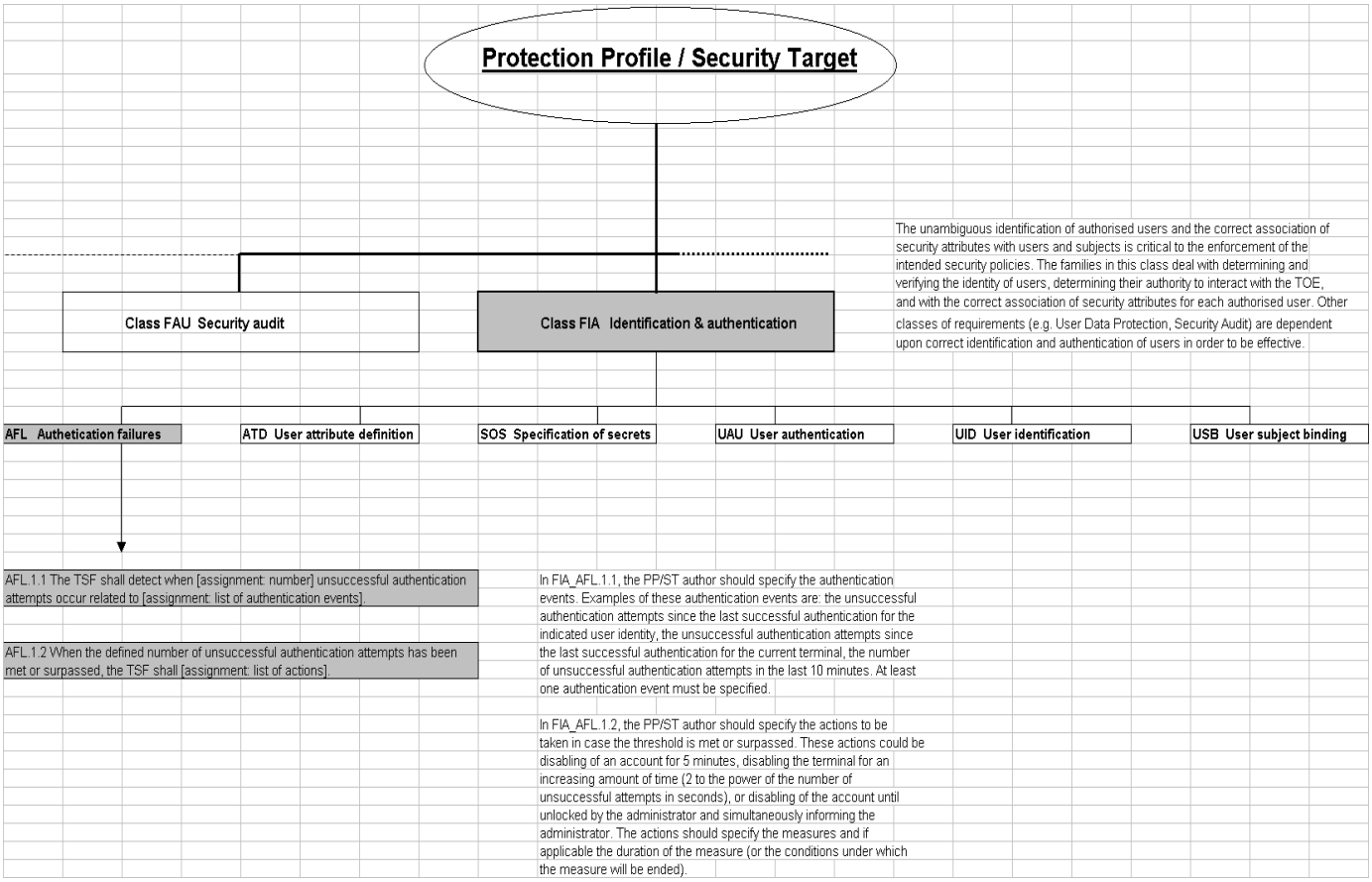
HPUXsec

SWEB

APPENDIX F. THREAT ANALYSIS WITHOUT STANDARDS



APPENDIX G. THE PREPARATION OF FUNCTIONAL SECURITY REQUIREMENTS FOR THE PRODUCT USING REQUIREMENTS INTRODUCED IN COMMON CRITERIA



APPENDIX H. THREAT ANALYSIS WITH COMMON CRITERIA STANDARD