# SECURITY REQUIREMENT PRIORITIZATION

A Dissertation submitted in partial fulfillment of the requirement for the award of degree of

**Master of Engineering**

**In**

**Computer Technology and Application**

By

**Shruti Jaiswal**

**College Roll No. - 14/CTA/07**

**University Roll No. – 12211**

Under the esteemed guidance of

**Dr. Daya Gupta**

**Department of Computer Engineering**

**Delhi College of Engineering**

**University of Delhi**

**2008-2009**

## CERTIFICATE

**DELHI COLLEGE OF ENGINEERING**
(Govt. of National Capital Territory of Delhi)
BAWANA ROAD, DELHI – 110042

Date:___________

This is to certify that dissertation entitled "**Security Requirement Prioritization**" has been completed by Shruti Jaiswal in partial fulfillment of the requirement of minor project of **Master in Engineering** in **Computer Technology & Application**.

This is a record of her work carried out by her under my supervision and support during the academic session 2008 -2009. This is a beneficial work in field of Security Engineering for creating computer based systems from scratch.

**(Dr. DAYA GUPTA)**
**HOD & PROJECT GUIDE**
(Dept. of Computer Engineering)
**DELHI COLLEGE OF ENGINEERING**
BAWANA ROAD, DELHI - 110042

# ACKNOWLEDGEMENT

"At times our own light goes out and is rekindled by a spark from another person. Each of us has a cause to think with deep gratitude of those who have lighted the flame within us."

*–Albert Schweitzer*

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor Dr. Daya Gupta, HOD, Department of Computer Engineering, Delhi College of Engineering, for her invaluable guidance, encouragement and patient reviews. Her continuous inspiration only has made me complete this dissertation. She kept on boosting me time to time for putting an extra ounce of effort to realize this work. She provided the conceptions and theoretical background for this study as well as suggested the rational approach. She remained a pillar of help throughout the project. I would also like to take this opportunity to present my sincere regards to my teachers Mrs. Rajni Jindal, Dr. S. K. Saxena, Mr. Manoj Sethi and other staff members of Computer engineering Department providing me unconditional and anytime access to the resources and guidance.

Finally, I would like to thank my classmates for their unconditional support and motivation during this work.

**(SHRUTI JAISWAL)**
**Master in Engineering**
**(Computer Technology & Application)**
**Dept. of Computer Engineering**
**DELHI COLLEGE OF ENGINEERING**
**BAWANA ROAD, DELHI – 110042**

# ABSTRACT

Now a day we are having all of our work done on web so security is of great concern for software engineering community for the development of web based systems. In the last decade, security has been a great concern for software system due to the increased penetration of wireless devices. Traditional software engineering processes enforce security measures during the design phase resulting in using an inefficient measure. This calls for new software engineering process where security engineering is an integral part of software development life cycle. For integrating security requirement with SDLC, the requirement engineers must discover security requirement along with functional and non functional requirement, so security requirement should be analyzed and prioritized.

In this work we will define a process for security requirements elicitation presenting techniques for activities like requirements discovery, analysis and prioritization. And also develop a tool for it.

With true security requirements identified as early as possible and systematically identified, Architecture team can choose most appropriate mechanism to implement them.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

Chapter 1

# INTRODUCTION

Security has been a great concern for software community in the development of web based systems [5, 8, 17, 18, 24] because they involve a large amount of real time cash in between transaction. Insecure systems are subjected to get infected by intruders, virus, malicious crackers and various other threats. Besides having safety, reliability and other quality features these systems may not be acceptable to customer. To avoid failure of cost potential system software community has established that system requirements are first captured, maintained and then implemented. Same principle may be applied to derive security mechanism.

## 1.1 General Concepts

Security of software system [7] is defined as technological and managerial procedures applied to computer systems to ensure the CIA (confidentiality, integrity and availability) of information managed by the system. In other words security of software system means the protection afforded to an automated information system in order to attain the applicable objectives of preserving the CIA (Confidentiality, integrity and availability) of information system resources that includes hardware, software, firmware, information/data, and telecommunications.

Computer systems are vulnerable [13] to many threats that can inflict various types of damage resulting in significant losses. These damages can range from errors harming database integrity to fires destroying entire computer centers and many more. Losses have a wide span ranging from the actions of supposedly trusted employees defrauding a system, from outside hackers, or from careless data entry clerks. Precision in estimating computer security-related losses is not possible because either most of them are never discovered or are "swept under the carpet" to avoid unfavorable publicity. The affects of threats varies considerably depending upon the applications like some of them affect the confidentiality or integrity of data while others may affect the availability of a system.

Neglecting Software security is not an option since now a day's society relies heavily or you can say completely on computers.

As an illustration we can say that now software is found in automobiles, airplanes, chemical factories, power stations, and numerous other systems that are business and mission critical. We trust our lives, our property, and even our environment to the successful operation of these technology-based systems. With the growth of technology the use of software systems is also increasing. Now days we use software systems almost for all things like shopping, paying bills, transferring money and in various other domains of online systems which deals with financial matter which are so critical that if they get attacked by intruders, malicious crackers etc. they will make a potential impact on the organizations as well as the persons who are using these systems.

However, software-based systems are neither perfect nor invulnerable [3, 13]. They commonly fail due to software defects, hardware breakdowns, accidental misuse, and deliberate abuse. They are also the target of malicious attacks by hackers, criminals, industrial spies, terrorists, and even agents of foreign governments and their militaries. Yet, failure is becoming less and less of an option as we depend on these systems more and more.

Thus, from the above facts we can say that security engineering is becoming essential component of systems engineering.

Most of the software that is being developed today incorporates security mechanism during design or implementation [7]. This results in an over constrained, inefficient and high cost system.

Many researchers [7, 9, 10] have proposed that if security mechanisms are incorporated in requirement phase itself then it can lead to the development of cost effective, reliable and efficient systems. Therefore we need to have a well defined process for managing security requirements similar to the requirement engineering process.

## 1.2 Motivation

In the development process of any computer based system (CBS) the first and the most important step is gathering of requirements for the development process to begin. As requirement engineering [ 21, 22] is a difficult task and any fault or error in this task will lead to the development of the CBS that will either not work properly or may fail under some circumstances also the cost of adding or changing the requirement during the later stages of SDLC is very high. Thus, the process of requirement engineering should be done properly so that a good quality and reliable system can be developed.

Forgoing fact is supported by number of studies [7, 9, 10] that system failure is due to inadequate understanding of the system requirements. While requirement engineering gathers functional requirement that specifies what the system must do, these requirements depend on the type of software being developed, the users of the software etc (For ex – In a Railway Reservation Sys the railway management can issue the ticket, traveller can make inquiry about the train). It is also very necessary to gather nonfunctional requirements that do not specify the functionality of the system directly rather they are related to system emergent properties such as reliability, response time, security, availability and many more. These non-functional requirements are more critical than the functional requirements since they play a vital role in making the system acceptable. Due to high potential cost of such system failure, it is necessary that security requirements of system are captured and maintained along with other requirements [9, 10]. Maintenance of security requirements is essential as it would be easy to change them along with others, due to changes in other requirements and threats resulting from new virus cyber terrorism.

As a result of forgoing, relatively new field **security engineering** has emerged which deals with

- *Security requirements specification and management.*
- *Implementing security requirements while taking design decisions.*
- *Implementing specific algorithms and others mechanism to make system acceptable during the design phase.*

***"Security Requirements [9, 10] can be defined as the high level requirement that gives detail specification of any system that may not be acceptable if these requirements are not implemented properly such as all child application can only access data for which they are properly authorized and authenticated".***

The analysis and specification of security requirements is inherently very difficult [7, 9]. Unlike other requirements that specify a required (and desired) capability, these requirements specify what is to be prevented (e.g., accidents and attacks due to safety hazards and security threats).

These requirements deal with assets that must be protected and with the risks of harm to these assets that must be managed. It is very difficult to distinguish security requirements from architectural and behavioral constraints [7]. These requirements should be appropriate and complete there is no value in specifying a requirement that will cost far more to implement than the value of the damage to the asset. And yet, there is an inherent level of uncertainty because what these requirements seek to prevent may or may not ever happen. This situation is especially true of safety requirements because some systems (e.g., nuclear power plants, chemical factories) are so critical that even a single, rare accident may render the system a complete failure. Although other systems (e.g., e-commerce Web sites) are essentially under constant attack, harm due to security threats are less mission critical, and a successful attack will not render the system a complete failure. Another problem is that the hazards and threats associated with software-intensive systems are also constantly changing, making the risks very difficult to quantify. Estimates of risks are often actually "guesstimates," and thus the risks are typically forced to be qualitative rather than quantitative.

The problem is that requirements engineers are supposed to develop requirements that will lead to the development of safe, secure and reliable systems. Though Requirement engineer are good at eliciting functional and non functional requirements but not security requirements [7, 9, 10] so when it comes to security requirements they end up specifying design constraints which makes the system under development over constrained and inefficient.

Our study shows that security requirements are not independent of functional and non functional requirements. Hence if security requirements are first elicited, analyzed along with others functional and non-functional requirements and then design decisions are taken to implement the security requirements it will lead to more efficient and cost effective system.

Therefore we aim to develop a well defined process for Security engineering that will have well articulated steps for security requirement elicitation, security requirement analysis, security requirement specification etc, Moreover this process should be coherent with the conventional Requirement Engineering so that eliciting security requirements become an integral part of requirement engineering and security requirements can be dealt in a similar fashion as we deal with functional and non-functional requirements.

Our literature study shows that there are number of proposals for eliciting security requirements using techniques like abuse case [4], misuse case [1, 2, 18, 20], common criteria [3, 24] and attack trees [6]. They specify requirements using templates [19] but these proposals of security requirements elicitation are not integrated in conventional requirements engineering process. Recently there are proposal of modeling languages and methodologies like secure tropos extension of tropos methodology [30], intentional anti model extension of KAOS methodology with security requirement oriented construct [33]. But these proposals do not account the cost effectiveness of ensuring security measures. There is also Risk Management methods like EBIOS [31], MEHARI [32], OCTAVE [28], CORAS [26], and CRAMM [29] which enforce security levels based on risk measure. Very recently N.Mayer, etc have proposed to unify security constructs and risk management concepts in information system development method [27]. In the next section we give an overview of them.

## 1.3 Proposed Work

The process that we will describe here is an extension of well defined process of spiral model. So we are going to incorporate security engineering into conventional spiral model to address security requirements and it will resolve all the issues discussed earlier. That is it will address all the process of requirement engineering and will also address security requirements prioritization and management.

The different activities in the process are as follows –

i. **Requirement Engineering** – Discover security requirement along with functional and non functional requirements.
ii. **Design Decisions** – With true security requirements specified most appropriate design decisions can be taken.
iii. **Implementation** – The decisions should finally be implemented.

In this thesis we shall focus on the first activity that is requirement engineering and shall be presenting techniques for –

a) ***Security Requirement Elicitation*** using view point oriented requirement definition as described by Sommerville [21].

b) ***Security requirement Analysis and Prioritization*** in which we analyze security requirements as we analyze functional and non functional requirements normally then for prioritization of different security requirements we will use risk measuring techniques CRAMM [29].

c) ***Requirement Specification and Management*** a glimpse for security requirement specification and management is provided as we do for functional and non functional requirements.

Finally a tool is developed which helps to prioritize security requirements associated with the functionality required by the stakeholders.

The advantage of using this approach for engineering security requirements of software systems helps in the identification of true security requirements. With true security requirements have been identified, systematically analyzed and specified the architecture team can choose most appropriate security mechanisms to implement them and thus making the system under development to be more efficient, reliable and secure.

## 1.4 Thesis Statement and Outline

This aim of this dissertation is to provide a requirement engineering process that will elicit security requirements along with functional and non-functional requirements and prioritize them. The approach if used for development of software systems results in the systems that are less vulnerable, cost effective and secure. The rest of the thesis is organized as follows.

The requirement engineer must have a clear understanding of security requirements [7] and he is able to distinguish them from architectural and behavioral constraints to elicit true security requirements for a system hence chapter 2 addresses what is requirement engineering, what are security requirement and different types of security requirements for any computer based systems, view point oriented requirement definition, different security requirement elicitation techniques and different risk management techniques.

Chapter 3 proposed software engineering process, process of security requirement prioritization and details explaining different activities of our process.

Chapter 4 explains the approach for Security Requirements Prioritization taking the case study of “Railway Reservation System”.

Chapter 5 provides the details of implementation and what are the various tools used for developing the tool.

Chapter 6 concludes the thesis.

Chapter 7 publication during the preparation thesis.

Chapter 8 gives the list of references that I have gone through during my research.

Chapter 2

# SECURITY REQUIREMENT ENGINEERING

In this section first of all we will go through requirement engineering and various types of requirements in section 2.1, then in section 2.2 we will describe security requirement and its type given by Firesmith [7], in next section we will have a look on view point oriented requirement definition (VORD), in section 2.4 we will define different security requirement elicitation techniques and in the last we get to know about different risk management techniques. So we are going to discuss all these in following sections.

## 2.1 Requirements Engineering

It is the process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed. The requirements are nothing but are the descriptions of the system services and constraints that are generated during the requirements engineering process.

***"Requirements as defined by Davis are the high level abstraction of the service or Constraint of a system."***

The different types of requirement are as follows –

- Functional Requirements
- Non Functional Requirements
- Domain Requirements
- Safety Requirements

### 2.1.1 Functional Requirements

Functional Requirements of a system describe functionality or System services. Functional requirements may vary depending on the type of software, expected users and the type of system where the software is used.

Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.

Examples –

- The Traveller in railway reservation system can book ticket, cancel it, make enquiry. So he must be able to do his all initial set of operation.
- The system shall provide appropriate view for the traveller to read all the details of the train schedule from the database.
- Every ticket shall be allocated a unique identifier (PNR) through which the traveller shall be able to track the status of the seat confirmation.

### 2.1.2 Non Functional Requirements

Non Functional Requirements are those that define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.

*"Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless."*

Non – functional requirements are further divided in to three categories as follows –

- ***Product requirements***

  Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

- ***Organizational requirements***

  Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.

- ***External requirements***

  Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

### 2.1.3 Domain Requirements

Domain requirements are those that are derived from the application domain and describe system characteristics and features that reflect the domain. Domain requirements be new functional requirements, constraints on existing requirements or define specific computations. Again the problem with domain requirements is if they are not satisfied, the system may be unworkable.

### 2.1.4 Safety Requirements

Safety requirements are applied to the system as a whole rather than individual sub-systems, it is an emergent property in terms of system engineering. These requirements are based on the analysis of possible hazards and risks; it must be separately specified for proper analysis.

Safety is a property of a system that reflects the system's ability to operate, normally or abnormally, without danger of causing human injury or death and without damage to the system's environment. Safety requirements are exclusive requirements i.e. they exclude undesirable situations rather than specify required system services.

**Examples –**

- Instant high voltage supply will cause burn of system and other components.
- Natural disasters and attack of terrorist will cause injury to system as well as to humans.

***"From the forgoing section we conclude that Requirements set out what the system should do and define constraints on its operation and implementation. Also it is very important that the requirements document should be complete, consistent and unambiguous."***

## 2.2 Security Requirements

These requirements are very important for successful operation of the system. It is defined as the process of identification of requirements in parallel to other requirements that are functional, non-functional and domain requirements. If these requirements are not properly elicited, analyzed and managed they result in a system that can fail. So they are even more important then all the requirements described above. These requirements are generally called as security requirements since they are responsible for the security of the system. The process of eliciting, analyzing and managing these security requirements is called as **security requirements.**

***"Security Requirements is defined as a high level requirement that gives detail specification of the system behavior that is unacceptable such as all users' application can only access data for which they are properly authorized. They differ from safety requirements which are domain specific and more suitable for control systems application. They are also known as shall not requirements but are not risks or threats".***

There are some major differences between security requirements and the requirements that are described in the above section. These differences are listed below –

- Security requirement are different from functional requirements which are derived from goals of system where as security requirements are objective resulting from threats on functionality or confidential data.
- Security requirements are related to non functional requirements such as correctness, interoperability, feasibility etc. For example non functional

requirement such as correctness, if implemented covers to some extent the integrity security requirement.

- Security requirements are also different from architectural constraint because these constraints unnecessarily prevent architecture team from using efficient mechanism to satisfy needed security requirements.

Security requirements are also different from Safety requirements in following respects –

- As security requirements are having well defined security life cycle and various standards whereas a safety requirement does not have;
- Security requirements are having generic threats rather than system specific hazards;
- Mature security technology (encryption, etc.) as compared to safety requirements.

Different types of security requirements as proposed by Firesmith [7] are as follows –

### 2.2.1 Identification Requirement

Identification requirement specifies the extent to which a CBS shall identify its external environment.

**Examples –**

- The main application shall identify all its client applications, human users before allowing them to use its capabilities.
- All persons should be identified before allowing them to enter.

### 2.2.2 Authentication Requirement

It is the security requirement that specifies that CBS should verify the identity of its externals. The typical objective of this security requirement is to ensure that externals are actually who or what they claim to be.

**Examples –**

- Application shall verify the identity of all of its users before allowing them to do any interaction (message, transaction) with the system.
- Before permitting the personnel to interact with data center there identities should be verified.

### 2.2.3 Authorization Requirement

This security requirement specifies that only authenticated externals can access specific application capabilities or information only if they have been explicitly authorized to do so by the administrator of the application.

**Examples –**

- The application shall allow the customer to obtain access to his/her account information rather than of other customer.
- Application shall not allow intruders access the credit card information of customers.
- Application shall not allow users to flood the system.

### 2.2.4 Immunity Requirement

An immunity requirement is any security requirement that specifies an application shall protect itself from infection by unauthorized undesirable programs (e.g., computer viruses, worms, and Trojans).

**Examples –**

- Application shall protect itself from infection by scanning data for viruses, worms, Trojan, and other harmful programs
- Application shall delete or disinfect the file found to be infected.
- Application shall notify the user if it detects a harmful program.

### 2.2.5 Integrity Requirement

This security requirement specifies ensures that its data does not get corrupted via unauthorized creation, deletion, modification.

**Examples –**

- The application shall prevent the unauthorized corruption of emails that it sends to customers.
- The application shall prevent the unauthorized corruption of data collected from customers and other external users.
- The application shall prevent the unauthorized corruption of all communications passing through networks.

### 2.2.6 Intrusion detection Requirements

This security requirement specifies that if an application has been attacked by intruders then that can be detected and recorded so that the administrator can handle them.

**Examples –**

- The application shall detect and record all attempted accesses that fail identification, authentication, or authorization requirements.
- The application shall notify the security officer of all failed attempted accesses.

### 2.2.7 Non repudiation requirements

This security requirement specifies that a party should not deny after interacting (e.g. message, transaction) with all or part of the interaction.

**Examples –**

The application shall make and store records of the following information about each order received from a customer and each invoice sent to a customer –

- The contents of the order or invoice.
- The date and time that the order or invoice was sent.
- The date and time that the order or invoice was received.
- The identity of the customer.

### 2.2.8 Privacy Requirements

This security requirement specifies that the application should keep its data and communications private from unauthorized individuals and programs.

**Examples –**

- Anonymity Privacy – The application shall not store any personal information about the users.
- Communication Privacy – The application shall not allow unauthorized individuals or programs access to any communications.
- Data Storage Privacy – The application shall not allow unauthorized individuals or programs access to any stored data.

### 2.2.9 Security Auditing Requirements

A security auditing requirement specifies that an application shall enable security personnel to audit the status and use of its security mechanisms.

**Examples –**

The application shall collect, organize, summarize, and regularly report the status of its security mechanisms including –

- Identification, Authentication, and Authorization.
- Immunity
- Privacy
- Intrusion Detection

### 2.2.10 Survivability Requirements

The security requirement specifies that that an application should work possibly in degraded mode even if some destruction has been there in the application.

**Examples –**

- The application shall not have a single point of failure.

- The application shall continue to function (in degraded mode) even if a data center is destroyed.

### 2.2.11 System Maintenance requirements

This requirement specifies that how the modifications can be done so that security fixes that have been detected can be resolved.

**Examples –**

- The application shall not violate its security requirements as a result of the upgrading of a data, hardware, or software component.
- The application shall not violate its security requirements as a result of the replacement of a data, hardware, or software component.

## 2.3 View Point Oriented Requirement Definition (VORD)

The process of eliciting the requirements as according to view points is called as view point oriented requirements definition [22]. To understand VORD process we need to define viewpoints.

Viewpoints now fall into two classes –

**a. Direct viewpoints –** These correspond directly to clients in that they receive services from the system and send control information and data to the system. Direct viewpoints are either system operators/users or other sub-systems which are interfaced to the system being analyzed.

**b. Indirect viewpoints** – Indirect viewpoints have an ‘interest’ in some or all of the services which are delivered by the system but do not interact directly with it. Indirect viewpoints may generate requirements which constrain the services delivered to direct viewpoints.

The concept of a direct viewpoint is clear; the notion of indirect viewpoints is necessarily diffuse. Indirect viewpoints vary radically from engineering viewpoints (i.e. those concerned with the system design and implementation) through organizational viewpoints (those concerned with the systems influence on the organization) to external viewpoints (those concerned with the systems influence on the outside environment). Therefore, if we take a simple example of a bank auto-teller system, some indirect viewpoints might be –

- A **security viewpoin**t which is concerned with general issues of transaction security.
- A **systems planning viewpoint** which is concerned with future delivery of banking services.
- A **trade-union viewpoint** which is concerned with the effects of system introduction on staffing levels and bank staff duties.

Indirect viewpoints are very important as they often have significant influence within an organization. If their requirements go unrecognized, they can often decide that the system should be abandoned or significantly changed after delivery.

There are two steps in the VORD as defined by Sommerville which are as follows –

- View Point Identification.
- Documenting View Points.

### 2.3.1 View Point Identification

The method of **viewpoint identification** which is proposed by Sommerville involves a number of stages –

- Prune the viewpoint class hierarchy to eliminate viewpoint classes which are not relevant for the system under question.

- Consider the system stakeholders i.e. those people who will be affected by the introduction of the system. If these stakeholders fall into classes which are not part of the organizational class hierarchy, add these classes to it.
- Using a model of the system architecture, identify sub-system viewpoints. This model may either be derived from existing system models or may have to be developed as part of the RE process.
- Identify system operators who use the system on a regular basis, who use the system on an occasional basis and who request others to use the system for them. All of these are potential viewpoints.
- For each indirect viewpoint class which has been identified, consider the roles of the principal individual who might be associated with that class.

### 2.3.2 Documenting View Point

Viewpoints have an associated a set of requirements, sources and constraints. Viewpoint requirements are made up of a set of services (functional requirements), a set of nonfunctional requirements and control requirements. Control requirements describe the sequence of events involved in the interchange of information between a direct viewpoint and the intended system. Constraints describe how a viewpoint's requirements are affected by non-functional requirements defined by other viewpoints.

## 2.4 Different Security Requirement Elicitation Techniques

There are number of proposals for eliciting security requirements using techniques like abuse case [4], misuse case [1, 2, 18, 20], security use cases [8], common criteria [3, 24] and attack trees [6]. They specify requirements using templates [19] but these proposals of security requirements elicitation are not integrated in conventional requirements engineering process. Recently there are proposal of modeling languages and methodologies like secure tropos extension of tropos methodology [30], intentional anti model extension of KAOS methodology with security requirement oriented construct

[33]. But these proposals do not account the cost effectiveness of ensuring security measures.

**2.4.1 - Abuse Cases –** Abuse case [4] is a specification of complete interaction between a system and one or more actors, where the interaction can cause harm. A complete abuse case defines an interaction between an actor and the system that results in harm to a resource associated with one of the actors, one of the stakeholders, or the system itself. A further distinction we make is that an abuse case should describe the abuse of privilege used to complete the abuse case. Clearly, any abuse can be accomplished by gaining total control of the target machine through modification of system software or firmware. Abuse cases can be described using the same strategy as for use cases. We distinguish the two by keeping them separate and labeling the diagrams. Figure 1, 2 are showing the concept with the example Railway reservation system.

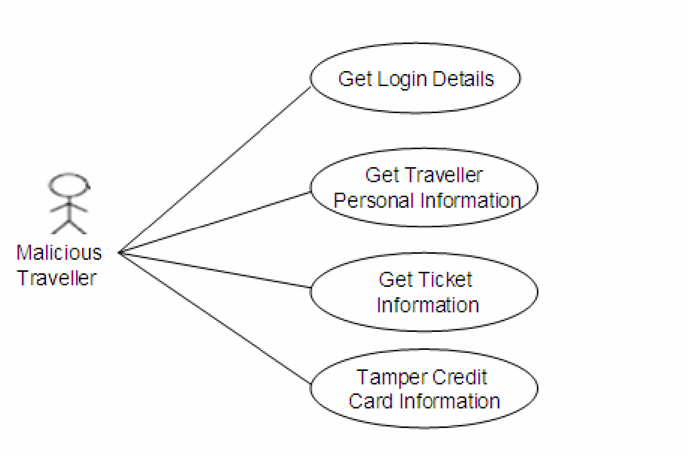

Malicious Traveller
Get Login Details
Get Traveller Personal Information
Get Ticket Information
Tamper Credit Card Information

**Figure 1: Abuse Case for Traveller for Railway Reservation System**

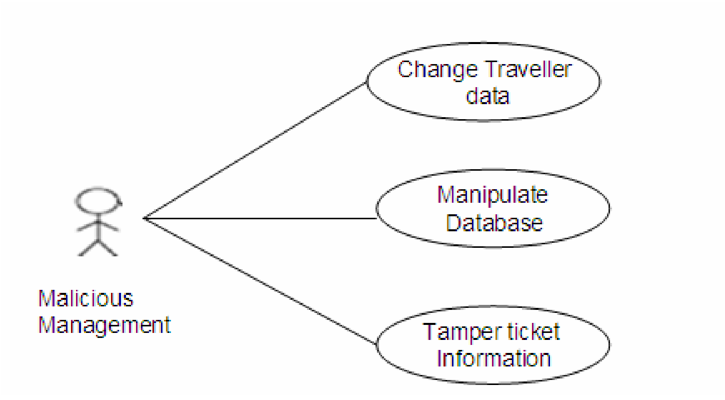


**Figure 2: Abuse Case for Railway Management in Railway Reservation System**

**2.4.2 Common Criteria (CC) with use cases –** This approach specifies how standards such as common criteria [3, 24] can be correlated with use case diagrams. The purpose of correlating use case and common criteria is to handle security in IT products during the software engineering process itself.

It has following steps:

- For the Purpose of correlating common criteria with use case diagrams the approach makes it mandatory to complete the actor profiles for each actor involved in the use case diagram.
- Actor profile has seven fields consisting of:
  - Its name
  - Functionality
  - Type of Actor that may be
    - ➢ Human
    - ➢ Corporative
    - ➢ Autonomous
  - Location
    - ➢ Local
    - ➢ Remote
  - Use Case Association
    - ➢ Read
    - ➢ Write
    - ➢ read_write
    - ➢ ask
    - ➢ answer
    - ➢ ask_answer
  - Weather or not the use case involves exchanging private information
  - Weather or not the use case involves secret information exchange.

- After the use case creator completes the actor profiles, these actor profiles are used to maps vulnerable threats to the actor from a predefined set of threat categories. As it has maintained threat repository so we can get threats by completing the threat profile as shown in Table 1. Now these threats are used to find out the security requirements.

| **Association = read** | **Association = write** |
|---|---|
| Impersonate | Change_Data |
| Repudiate_Receive | Repudiate_Receive |
| If(Private Exchange = true) | If(Private Exchange = true) |
| Privacy_Violated | Privacy_Violated |
| If(Secret Exchange = true) | If(Secret Exchange = true) |
| Data_Theft | Data_Theft |
| If(Location = remote) | If(Location = local) |
| Outsider | Insider |

**Table 1: Showing how threats are retrieved when the type of actor is human**

**2.4.3 Misuse Cases –** This approach described [1, 2, 18, 20] is an extension of use-case diagrams. A use case generally describes behavior that the system/entity owner wants the system to perform while Misuse cases apply the concept or behavior that the system's owner does not want to occur. Use case diagram are driven by goals of the system misuse are driven by threats to the system. Figure 3 is showing the Misuse Case for Railway Reservation System.

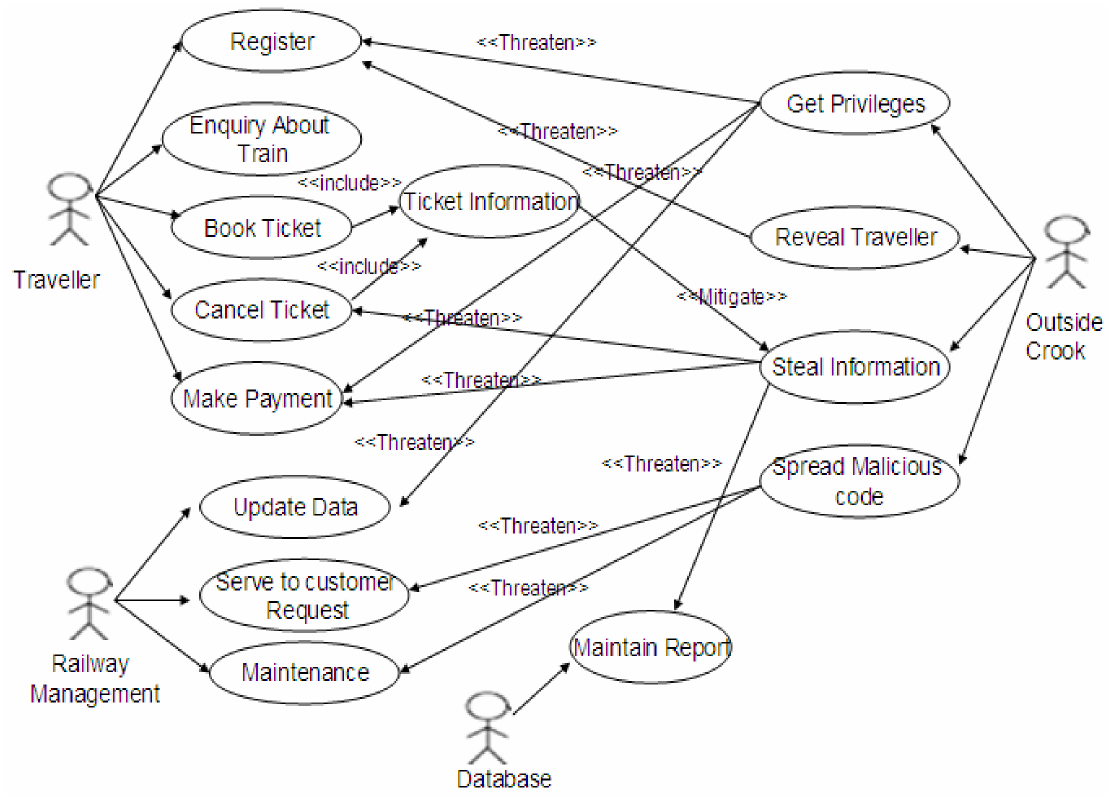


**Figure 3: Misuse Case for Railway Reservation System**

**2.4.4 Security Use Cases –** This approach by Firesmith [8] says that misuse cases are highly effective ways of analyzing security threats but are inappropriate for the analysis and specification of security requirements, Because the success criteria for a misuse case is a successful attack against an application while the security use cases specify requirements that the application shall successfully protect itself from its relevant security threats. Figure 4 showing the concept of Security Use Cases with the example of Railway Reservation System.

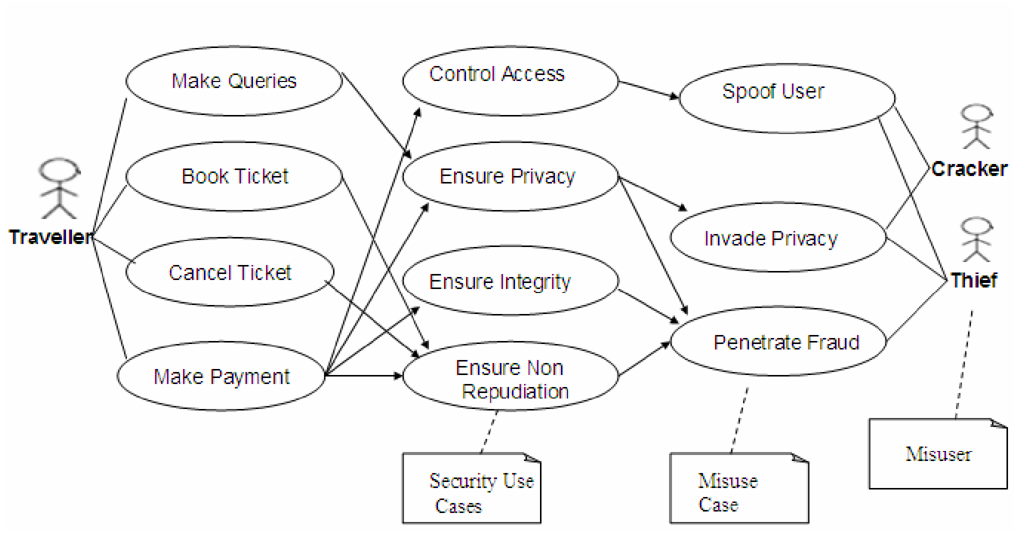


**Figure 4: Security Use Case for Railway reservation System**

**2.3.5 - Attack Trees** - Attacks trees [6] are a way to represent the attacks using the most widely used data structure Trees. Attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks. Basically, you represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes. In this method the attack is represented with the attacker goal as the root node and the different ways of achieving that goal as leaf nodes. Satisfying a tree node represents either satisfying all leaves (AND) or satisfying a single leaf (OR). The value of attack tree analysis is derived from the attributes associated with each of the nodes. Figure 5; illustrate the simple attack tree for Railway reservation System.

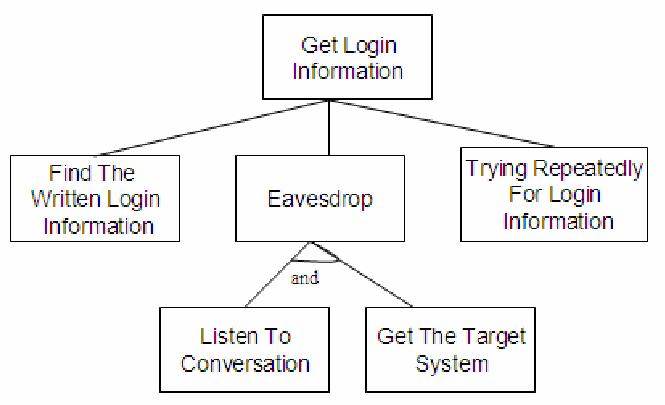


**Figure 5: Attack Tree for Railway Reservation System**

In any real attack tree, nodes will have many different values corresponding to many different variables, both Boolean and continuous. Different node values can be combined to learn even more about a system's vulnerabilities. For instance, determines the cheapest attack requiring no special equipment. You can also find the cheapest low-risk attack, most likely nonintrusive attack, best low-skill attack, and cheapest attack with the highest probability of success, most likely legal attack, and so on. Every time you query the attack tree about a certain characteristic of attack, you learn more about the system's security. So simple attack trees are extended to incorporate various other constraint for analysis.

To make this work, you must marry attack trees with knowledge about attackers. Different attackers have different levels of skill, access, risk aversion, money, and so on. If you're worried about organized crime, you have to worry about expensive attacks and attackers who are willing to go to jail. If you are worried about terrorists, you also have to worry about attackers who are willing to die to achieve their goal. If you're worried about bored graduate students studying the security of your system, you usually don't have to worry about illegal attacks such as bribery and blackmail. The characteristics of your attacker determine which parts of the attack tree you have to worry about.

**2.4.6 Extension of tropos Methodology –** Tropos [30] is a methodology, for building agent oriented software system. It uses the concept of actors, its role, goal and dependencies. It provides constraint on the system so that the objective can be achieved only by the desired personals. It is being extended to incorporate security feature by introduction of security diagram. With the introduction of security diagram one can easily get to know about what are the security features, protection objectives and security mechanism. Security diagram basically captures the possible security mechanism that contributes positively or negatively to protection objective. We can add security constraint to security diagram. If it imposed by stakeholder it is during the early requirement stage that may be positive or negative and if it by security diagram it is during the late requirement stage and is only positive. We have secure entities introduced by actor for achievement of security constraint. And secure dependencies for successful satisfaction of actor dependencies. As tropos covers the four phases of software development that are early requirement, late requirement, architectural design, detailed design. But here the focus is only given to early and late requirement phases only.

**2.4.7 Intentional Anti-models extension of KAOS –** This approach extends the KAOS framework for goal oriented requirements engineering. As in any project we have various layers like at the bottom we have crypto layer, security protocol layer, system layer, application layer. This model deals with security engineering at the application layer. They presented a requirement engineering method for elaborating security requirements based on the incremental building and specification of two concurrent models: an intentional model of the system-to-be and an intentional anti-model yielding vulnerabilities and capabilities required for achieving the anti-goals of threatening security goals from the original model. The method allows threat trees to be derived systematically through anti-goal refinement until leaf nodes are reached that are either attackee vulnerabilities observable by the attacker or anti-requirements implementable by this attacker. The original model is then enriched with new security requirements derived as countermeasures to the anti-model. Figure 6 showing the Intentional Anti Model for Railway Reservation System.

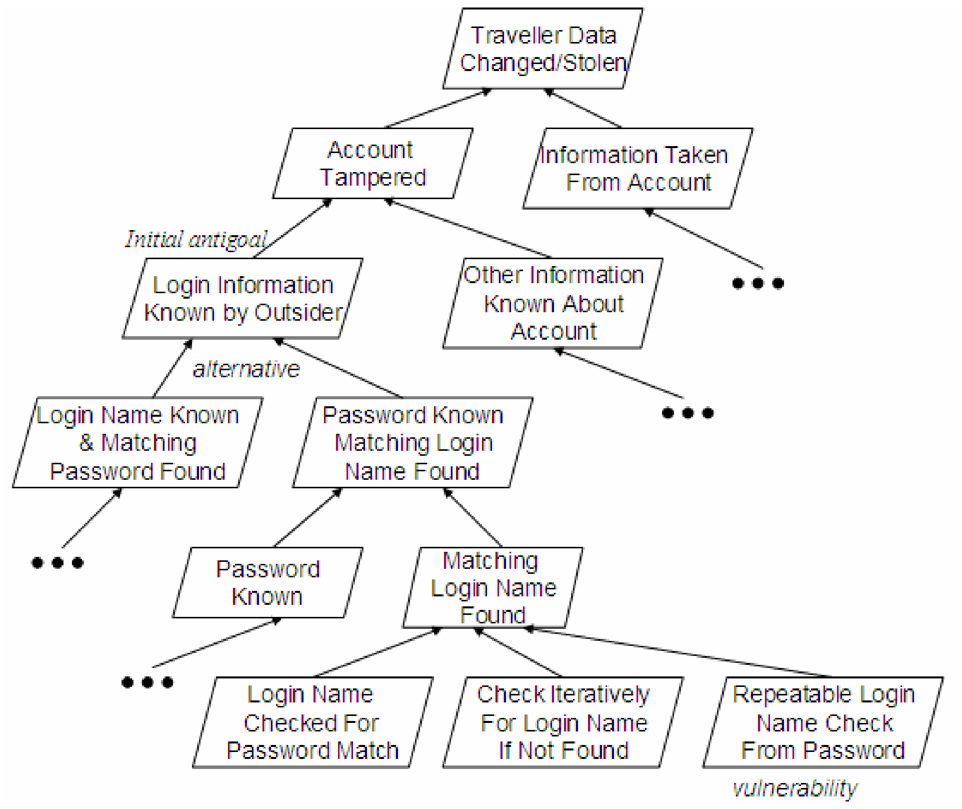

Traveller Data Changed/Stolen
Account Tampered
Information Taken From Account
Initial antigoal
Login Information Known by Outsider
Other Information Known About Account
alternative
Login Name Known & Matching Password Found
Password Known Matching Login Name Found
Password Known
Matching Login Name Found
Login Name Checked For Password Match
Check Iteratively For Login Name If Not Found
Repeatable Login Name Check From Password
vulnerability

**Figure 6: Intentional Anti Model for Railway Reservation System**

## 2.5 Different Risk Management Techniques

There are various Risk Management methods like OCTAVE [28], CORAS [26], and CRAMM [29], which enforce security levels based on risk measure. Very recently N.Mayer, etc have proposed to unify security constructs and risk management concepts in information system development method [27].

**2.5.1 OCTAVE –** OCTAVE [28] stands for Operationally Critical Threat, Asset and Vulnerability Evaluation, was developed by the Carnegie Mellon Software Engineering Institute (SEI). The core elements of OCTAVE are criteria which organizations can use to develop their own methodology. The basic building block of OCTAVE is structured interviews at various levels within the organization to identify critical assets and then determine the risks to those assets.

- **Overview of OCTAVE method –** The OCTAVE Method uses a three-phase approach (see Figure 7) to examine organizational and technology issues, assembling a comprehensive picture of the organization's information security needs. Each phase consists of several processes. These phases and their processes are described below –

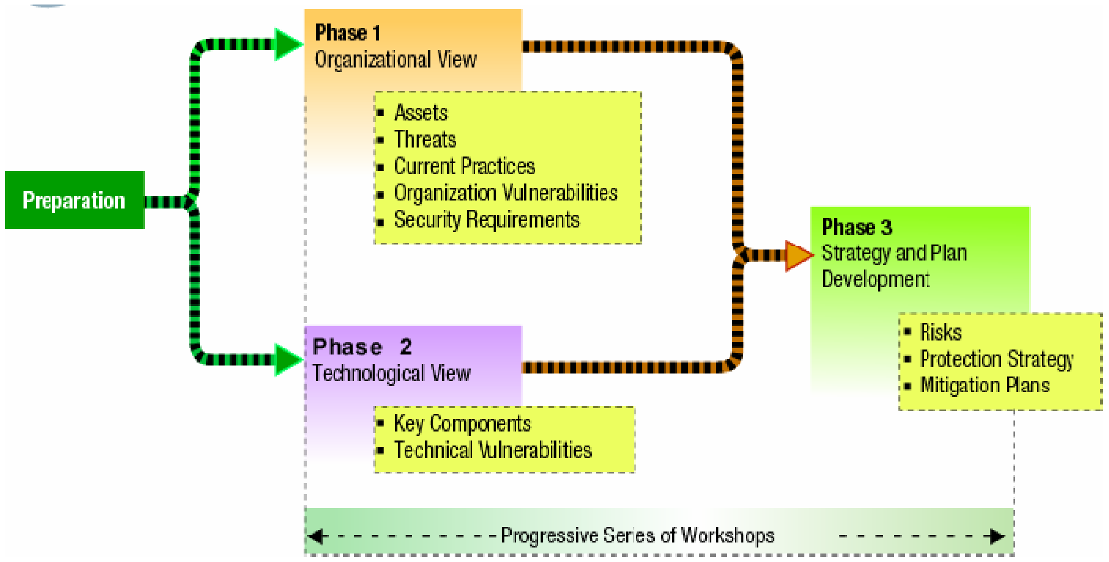


**Figure 7: The OCTAVE method**

## Phase 1 – Build Asset-Based Threat Profiles

This phase is an organizational evaluation. The analysis team determines which assets are most important to the organization (critical assets) and identifies what is currently being done to protect those assets. Surveys based on the catalog of practices are used to elicit the information from the organization's personnel about what is being done well with respect to security practices. The processes of Phase 1 are –

- **Process 1 –** Identify Senior Management Knowledge – Selected senior managers identify important assets, perceived threats, security requirements, current security practices, and organizational vulnerabilities.

- **Process 2 –** Identify Operational Area Management Knowledge – Selected operational area managers identify important assets, perceived threats, security requirements, current security practices, and organizational vulnerabilities.

- **Process 3 –** Identify Staff Knowledge – Selected general and IT staff members identify important assets, perceived threats, security requirements, current security practices, and organizational vulnerabilities.

- **Process 4 –** Create Threat Profiles – The analysis team analyzes the information from Processes 1 through 3, selects critical assets, refines the security requirements associated with those assets, and identifies threats to the critical assets, creating threat profiles.

**Phase 2 – Identify Infrastructure Vulnerabilities**

This phase is an evaluation of the information infrastructure. The analysis team examines key operational components for weaknesses (technology vulnerabilities) that can lead to unauthorized action against critical assets. The processes of Phase 2 are –

- **Process 5 –** Identify Key Components – The analysis team identifies key information technology systems and components for each critical asset. Specific instances are then selected for evaluation.

- **Process 6 –** Evaluate Selected Components – The analysis team examines the key systems and components for technology weaknesses. Vulnerability tools (software, checklists, and scripts) are used. The results are examined and summarized, looking for the relevance to the critical assets and their threat profiles.

**Phase 3 – Develop Security Strategy and Plans**

During this part of the evaluation, the analysis team identifies risks to the organization’s critical assets and decides whether and how to address those risks. The processes of Phase 3 are –

- **Process 7 –** Conduct Risk Analysis – The analysis team identifies the impact of threats to critical assets to define risks, develops criteria to evaluate those risks, and evaluates the risk impacts based on those criteria. This produces a risk profile for each critical asset.

- **Process 8 –** Develop Protection Strategy – The analysis team creates a protection strategy for the organization and mitigation plans for the critical assets, based upon an analysis of the information gathered. Senior managers then review, refine, and approve the strategy and plans.

- **OCTAVE Is Part of a Continuum**

OCTAVE creates an organization-wide view of the current information security risks, providing a snapshot in time, or a baseline, that can be used to focus mitigation and improvement activities. During OCTAVE, an analysis team performs activities to –

- *identify* the organization’s information security risks
- *analyze* the risks to determine priorities
- *plan* for improvement by developing a protection strategy for organizational improvement and risk mitigation plans to reduce the risk to the organization’s critical assets.

An organization will not improve unless it implements its plans. The following improvement activities are performed after OCTAVE has been completed. After OCTAVE, the analysis team, or other designated personnel –

- *plan* how to implement the protection strategy and risk mitigation plans by developing detailed action plans (This activity can include a detailed cost-benefit analysis among strategies and actions, and it results in detailed implementation plans.)
- *implement* the detailed action plans.
- *monitor* the action plans for schedule and for effectiveness (This activity includes monitoring risks for any changes.)
- *control* variations in plan execution by taking appropriate corrective actions.

An information security risk evaluation is part of an organization's activities for managing information security risks. OCTAVE is an evaluation activity, not a continuous process. Thus, it has a defined beginning and end. Figure 8 shows the relationship among these activities and where OCTAVE fits in. Note that risk management activities define a *plan-do-check-act* cycle.

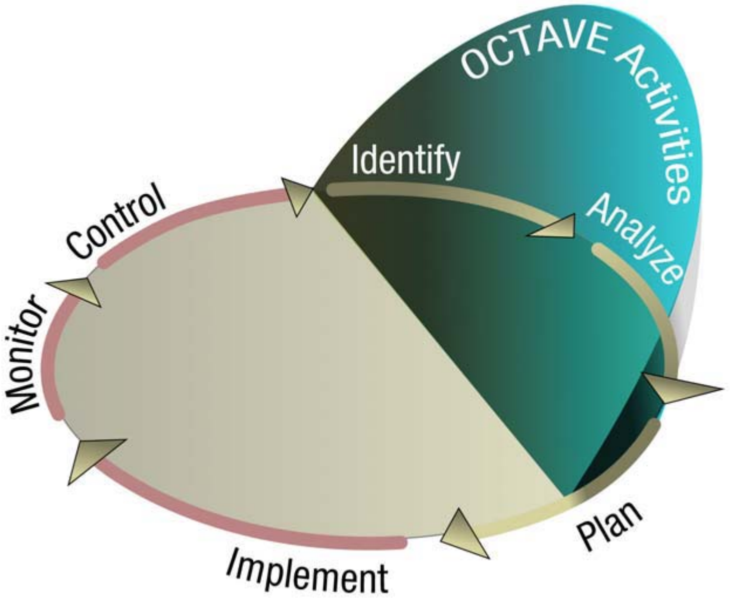


**Figure 8: OCTAVE and Risk Management Activities**

Periodically, an organization will need to "reset" its baseline by conducting another OCTAVE. The time between evaluations can be predetermined (e.g., yearly) or triggered by major events (e.g., corporate reorganization or redesign of an organization's computing infrastructure). Between evaluations, an organization can periodically identify new risks, analyze these risks in relation to existing risks, and develop mitigation plans for them.

**2.5.2 – CRAMM –** CRAMM (CCTA Risk Analysis and Management Method) [29] was developed by the CCTA (Central Computer and Telecommunication Agency) in 1985. The CCTA was tasked by the UK Government Cabinet's Office to investigate the risk analysis and management methods within the central government for IT. CRAMM provides steps to determine the likelihood and the impact of a threat on an asset. Then these determined values are used to calculate the risk value for each threat to all the assets. The identification stage of the processes is a very strong area of CRAMM. It takes into consideration all the different elements of risk, the tangible and intangible assets, threats, asset values, safeguards, consequences and the likelihood of threats. CRAMM has a cause and effect relationship through a threat-asset based relationship.

The security of information systems and networks has been of major concern for many years. The rapid expansion in the use of information technology, and a growing awareness of the associated security risks, has highlighted the need to ensure that all risks are identified, assessed and managed.

This section provides an overview of risk analysis and management. It also describes how the CRAMM method can be used to identify, analyze and manage the risks associated with an information system.

- **Risk analysis**

Risk is normally defined as the chance or likelihood of damage or loss. In CRAMM this definition is extended to include the impact of damage or loss. That is, it is a function of two separate components, the **likelihood** that an unwanted incident will occur and the **impact** that could result from the incident.

Risk Analysis involves identifying and assessing risks to data and the information system and network which support it. Typical risks include –

- data being lost, destroyed or wiped
- data being corrupted
- data being disclosed without authority.

The processes involved in risk analysis are identifying assets, asset values, threats and vulnerabilities, and then calculating the risk. These are detailed as follows –

**i. Identification of Assets**

Assets within an information system or network can be considered under three categories–

- information or data assets
- software assets
- physical assets, such as file servers, workstations, bridges, routers.

Key assets need to be identified.

**ii. Valuation of Assets**

All assets have a value to the organization and this can be measured in terms of the impact that could result if the confidentiality, integrity or availability of the assets were compromised. The asset valuation process measures the impacts that could result if:

- data assets were disclosed, modified, destroyed or made unavailable in an unauthorized or unexpected manner
- physical assets were damaged or destroyed
- software assets were damaged, destroyed, corrupted or, in the case of sensitive software, disclosed in an unauthorized manner.

Valuation of assets provides the **impact** component of the risk assessment.

**iii. Threat Assessment**

A Threat Assessment involves identifying and assessing the level of threat to the assets of a system. Typical threats include:

- deliberate attacks such as hacking, spoofing, insertion of false messages, introduction of damaging or disruptive software, theft, wilful damage
- disasters such as fire, flood, lightning strike
- errors by individuals
- technical failures.

The 'level of threat' is a measure of the likelihood of an attack or incident actually occurring.

### iv. Vulnerability Assessment

A Vulnerability Assessment involves identifying and assessing the extent to which the assets are vulnerable to the identified threat. Vulnerability is a measure of inherent weakness within the system or network. The threat assessment and vulnerability assessment together provide the **likelihood** component of the risk assessment.

### v. Risk Assessment

A Risk Assessment involves measuring the level of risk to the system or network. The level of risk is identified from the value of the assets, the level of threat and the extent of the vulnerability. If a system contains highly valuable assets, the level of threat is high, and significant vulnerabilities exist, then the security risk to the business is considered to be high. Measures of risk translate directly into measures of security requirement, so that if there is a high risk there is a high requirement for security.

- **Risk management**

Risk Management involves identifying, selecting and adopting justified security and contingency 'countermeasures' to reduce risks to an acceptable level.

Countermeasures may act in different ways such as:

- reducing the likelihood of attacks or incidents occurring
- reducing the system's vulnerability
- reducing the impact of an attack or incident should it occur

- detecting the occurrence of attacks or incidents
- facilitating recovery from an attack or incident.

Figure 9 summarizes the IT risk analysis and management process.

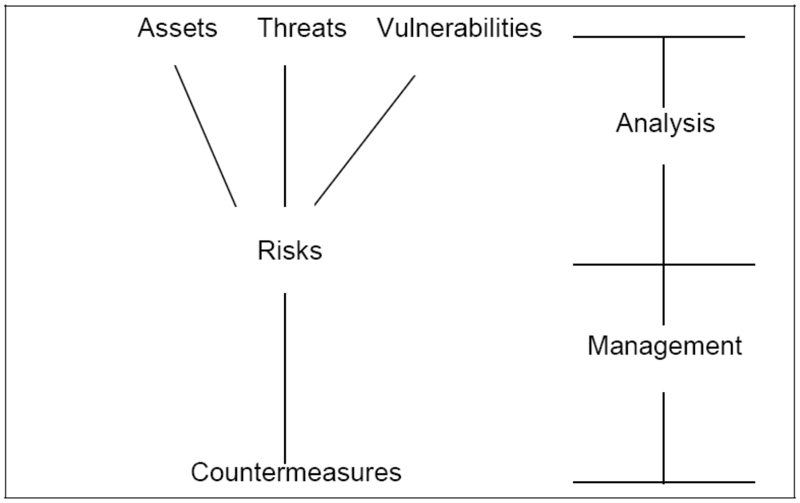


**Figure 9: The IT Risk Analysis and Management Process**

**2.5.2 – CORAS -** CORAS [26] is a model-based risk assessment method and is a research and technological development project under the Information Society Technologies (IST) programme. CORAS's goal is to adapt, refine, extend and combine methods for risk analysis. Some of the methods employed include Fault Tree Analysis (FTA), Markov Analysis, HazOp and Failure Mode and Effect Criticality Analysis (FMECA). The CORAS methodology is a risk management process based on the standardized modeling technique UML. The information used for the evaluation process is part of the CORAS Risk Assessment Platform which utilizes XML (Extensible Mark-up Language) and is a Java application server based platform. The platform includes all the documentation required to install and operate the software, as well as detailed documentation regarding the CORAS methodology.

In the CORAS method a security risk analysis is conducted in seven steps shown in Figure 10 –

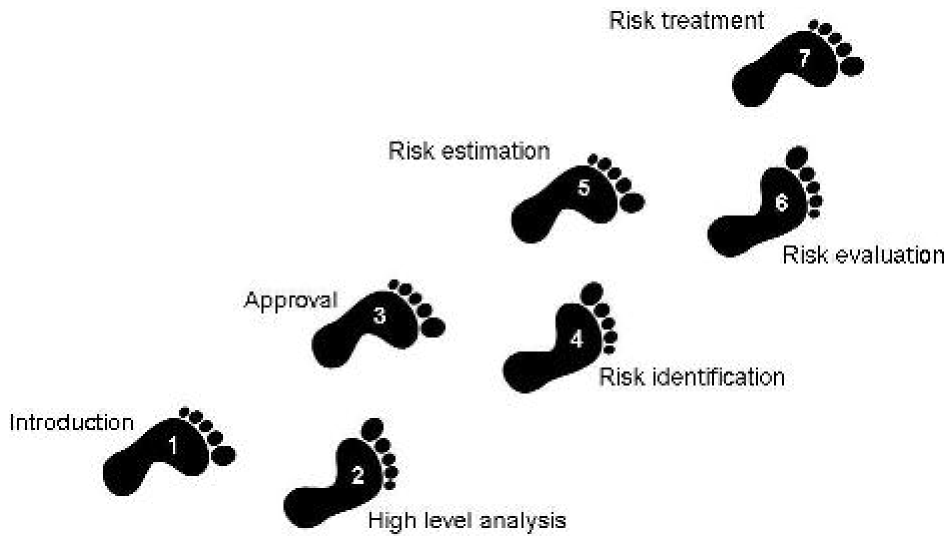


**Figure 10: Showing the steps of CORAS method**

The seven steps of the CORAS method are summarized as follows.

- **Step 1**: The first step involves an introductory meeting. The main item on the agenda for this meeting is to get the representatives of the client to present their overall goals of the analysis and the target they wish to have analyzed. Hence, during the initial step the analysts will gather information based on the client's presentations and discussions.
- **Step 2**: The second step also involves a separate meeting with representatives of the client. However, this time the analysts will present their understanding of what they learned at the first meeting and from studying documentation that has been made available to them by the client. The second step also involves a rough, high-level security analysis. During this analysis the first threats, vulnerabilities, threat scenarios and unwanted incidents are identified. They will be used to help with directing and scoping the more detailed analysis still to come.
- **Step 3**: The third step involves a more refined description of the target to be analyzed, and also all assumptions and other preconditions being made. Step three is terminated once all this documentation has been approved by the client.
- **Step 4**: This step is organized as a workshop, drawn from people with expertise on the target of the analysis. The goal is to identify as many potential unwanted incidents as possible, as well as threats, vulnerabilities and threat scenarios.

- **Step 5**: The fifth step is also organized as a workshop. This time with the focus on estimating consequences and likelihood values for each of the identified unwanted incidents.
- **Step 6**: This step involves giving the client the first overall risk picture. This will typically trigger some adjustments and corrections.
- **Step 7**: The last step is devoted to treatment identification, as well as addressing cost/benefit issues of the treatments. This step is best organized as a workshop.

**Chapter 3**

# PROPOSED SOFTWARE ENGINEERING PROCESS

From the forgoing section we come across various proposals for security requirements engineering but all of them are having following disadvantages –

- Well articulated steps for security engineering.
- Not embedded in conventional requirements engineering process.
- They do not have techniques for security requirements management.

To overcome these problems we have proposed our software engineering process. That will be discussed in following section.

## 3.1 Proposed Software Engineering Process –

Proposed Software engineering process that is shown in Figure 11 is modified version of conventional spiral model.

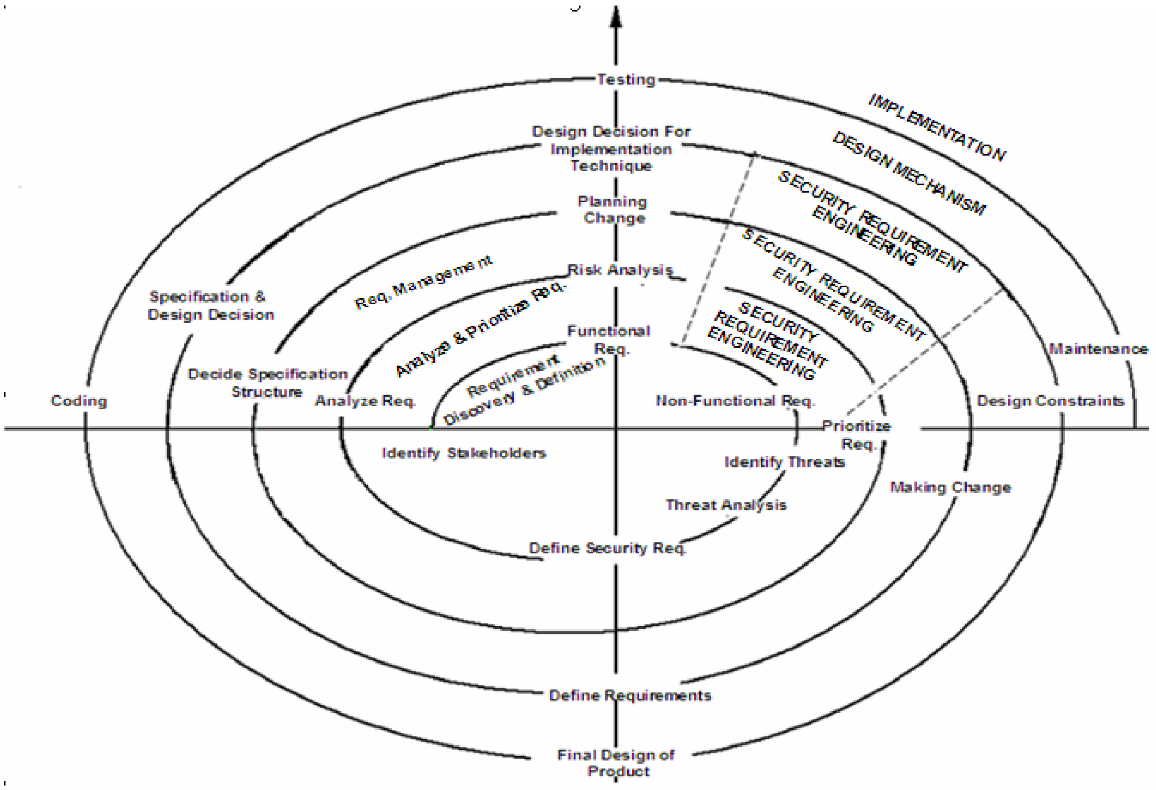


**Figure 11: Different Tasks in Software Engineering Process**

As in the conventional spiral model the process starts from the requirement discovery & definition where the developer will meet the client and collect all the information related to project development then analyze them and check feasibility of the project and if it is feasible, then he will proceed and plan the further phases of the development, and follow all the successive steps accordingly. *But in this process there is no consideration of security requirement in the early phase that is the requirement engineering phase which is the first phase.* It will consider all the constraint related to security at the end of development process. So handling of these at later phase will cost more and will sometime lead to over budgeting or failure of project.

So it will be better to incorporate security requirement in the beginning of the project. So we have modified the conventional spiral model to incorporate security requirement in it.

Different phases of proposed software engineering process are as follows –

**a) Requirements Engineering** – This process is based on well known process of View Point Oriented Requirement Definition [21, 22]. Here security requirements along with functional and non functional requirements are discovered and defined for the system to be developed, then analysis, prioritization is done, finally they are defined and managed for future use.

**b) Design mechanism –** Once true security requirements are specified, appropriate design mechanism/constraint such as use of biometrics like finger print recognition, face recognition for authentication, cryptography for Privacy are decided according to the budget of the system to be developed.

c) **Implementation** – All functionalities are implemented incorporating design decision in the implementation phase of the system.

All the activities defined in the forgoing section are spiral in nature (Figure 11). The merit of doing these activities in spiral is that one has not to wait for the first activity to finish rather he can start the other activities in the process this helps in the development of the product from scratch or it can also be used for the enhancement of the existing systems.

The first three spirals that are shown in figure11 shows the three activities of the process described above as shown along the radial dimension of second quadrant. After our process is completed, means that we have elicited, analyzed, prioritized and managed all the security requirements of the system under question. Once this process is done the design and architecture team can take the most appropriate design decisions for mitigating the threats (security requirements). This is shown along the fourth spiral of the figure shown above. Once the design decisions are taken for different security requirements the implementation team can write the appropriate code so that the system under development is completed fully as shown in last or fifth spiral of the figure 11.

As shown in the Figure 11 first of all, different stakeholders must be identified as shown at the start of spiral then we will define functional and non- functional requirements. As we have functionalities we can identify various possible threats to the functionalities, these threats help in eliciting security requirements. Finally these security requirements are prioritized based on risk measure in requirement engineering phase.

It is first phase which is of interest to us.

All the activities are described below in detail.

## 3.2 Requirements discovery and definition –

Different steps in this activity are –

i. Identify various stakeholders of the system using view-point analysis [21, 22]. We have identified the various abstract classes of actors as direct and indirect actors as shown in Figure 12. Direct actors are those who directly interact with the system such as human, software system and hardware devices. Indirect actors refer to Engineering personals who develop software and people who regulate application domain. For our process our interest is in direct actor.

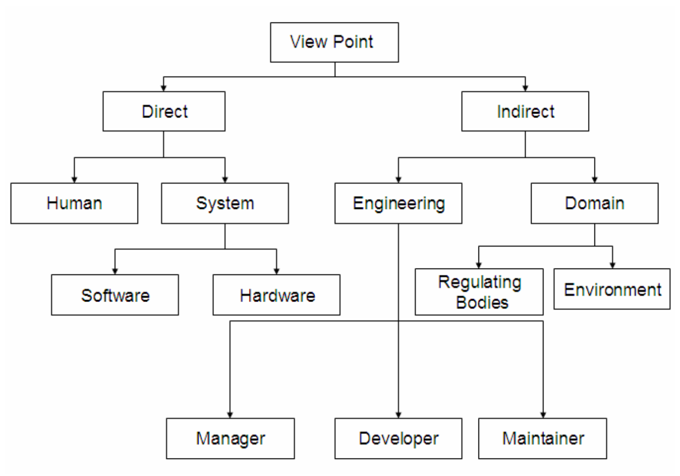


**Figure 12: Different types of stake holders as according to view point**

ii. Identify the functionalities of each actor conceptualized in step (i) and also determine associated non - functional requirements.

iii. Identify the threats associated with each of the functional requirements or data which is used by the functionality. Threats are identified corresponding to functionality based on common criteria approach [3, 24] by filling the stakeholders profile which contains seven fields which define the whole function and actor relationship.

- Actor profile which contains seven fields is as follows –

  - Actor name (Defines the name of Actor),
  - Actor functionality (what actor wants to do),
  - Type (Weather it is human, cooperative or autonomous depending upon its functionality),
  - Location (Weather the access location is local or remote),
  - Private exchange (Weather the interaction of Actor with other actor or during a particular function there is any exchange of private data),

- Secret exchange (Weather the interaction of Actor with other actor or during a particular function there is any exchange of secret data) and
- Association (Weather the data to be accessed is read, write, read_write, ask, answer, ask_answer)

As in common criteria based approach [3, 24] we shall be developing the repository of the threats. The repository that we have defined will be limited to the following category of threats [3, 24] shown in table 2.

| | **Threat Name** | **Description** |
|---|---|---|
| 1. | Change_Data | Information may be changed (insertions, replacements, modifications, deletions) while it being stored or processed. |
| 2. | Data_Theft | Business process data may be stolen. |
| 3. | Deny_Service | Application and network services may not be available for use. |
| 4. | Disclose_Data | Information may be disclosed in to unauthorized users while being stored or processed. |
| 5. | Impersonate | Someone may obtain unauthorized access by impersonating an authorized user. |
| 6. | Insider | An authorized user may gain unauthorized access. |
| 7. | Outsider | An individual who is not an authorized user of the system may gain access to the TOE. |
| 8. | Privacy_Violated | Unauthorized access to privacy data of system users may occur without detection. |
| 9. | Repudiate_Receive | An entity may deny that it has received business or commitment data. |
| 10. | Repudiate_Send | An entity may deny that it has send business or commitment data. |
| 11. | Spoofing | An entity may cheat for money. |
| 12. | Social_Engineer | Tricking someone into giving you his or her password for a system than to spend the effort to hack in. |

**Table 2: Threats Category and description**

**iv.** Once the threats have been identified we can define security requirements to mitigate these threats. The threat that have been identified in step (iii) above maps to security objectives [3, 25] which are mapped to security requirements [3] and this is how true security requirements are identified. It is shown in Table 3.

| Viewpoints | Services | Non-functional Requirements | Threats | Security Requirement |
|---|---|---|---|---|
| Traveller | 1.Registration of Customer<br>2.Enquiry of Train<br>3.Booking of Ticket<br>4.Cancellation of Ticket<br>5.Make Payment | 1.Reliability<br>2.Response Time<br>3.Execution Time | 1.Flooding<br>2.Privacy_Violated<br>3.Change_Data<br>4.Repudiate_Receive | 1.Authorization<br>2.Privacy<br>3.Integrity |
| Railway Management | 1.Update Data<br>2.Serve for Customer Request<br>3.Maintenance | 1.Correctness<br>2.Response Time<br>3.Robustness<br>4.Scalable | 1.Flooding<br>2.Privacy_Violated<br>3.Spoofing<br>4.Outsider | 1.Authentication<br>2.Identification<br>3.Integrity |
| Database | 1.Maintain the reports of the transaction | 1.Reliability<br>2.Response Time<br>3.Integrity | 1.Outsider<br>2.Flooding<br>3.Privacy_Violated<br>4.Spoofing | 1.Integrity<br>2.Authentication |

**Table 3: An example "Railway Reservation System" explaining our process**

## 3.3 Analysis and Prioritization of the requirements

In this process the security requirements that we have discovered in the previous step are unstructured. In this step we collect these unstructured security requirements, groups related security requirements and organize them into coherent clusters.

### 3.3.1 Analyze Requirements

The various steps in this analyzing the security requirements are as follows –

**i. Checking for completeness**

In this step we will make a check list to check that the security requirements that have been elicited have mitigated all the threats to the functionality of the system.

It means that we just check that all the threat have been taken in to consideration or not, because if any of the threat has been left over it can cause the system to fail or can be attacked in later stages. We simply create a table containing a list of threats to the stakeholders and put yes/no against each threat if the threat has been checked or not respectively.

**ii. Checking for Consistency**

In this step we resolve the contradictions that may exist in the security requirements elicited from different view points. This is very essential to ensure that the final specifications of security requirements are consistent.

In this step we also check that the security requirement for realism i.e. security requirements can be implemented in some or other way in the budget of the project.

**iii. Group Related Requirements**

This step consists of identifying the security requirements that can be grouped together. We group them according to if one of the security requirements are implemented the others in the group are implemented automatically.

### 3.3.2 Prioritize Requirements

After the security requirements have been analyzed for completeness, consistency and are grouped together then we prioritize them by following simple steps. We will use the CRAMM [29] process of risk analysis to prioritize security requirements. We have chosen CRAMM [29] method for identification of risk as it offers structured and fast approach to risk analysis over other methods [35].

As security requirement are to mitigate threats and avoid vulnerability and risk they will be prioritized on the measure of threat, vulnerability and risk. So Prioritization is done following two steps:

1. Evaluation of threats
2. Prioritization of security requirement

### 3.3.2.1 Evaluation of Threats –

We will evaluate threats based on the calculated value of risk. In this task we have to follow the following steps that are as follows and as shown in Figure 13.

i) Threat Assembling
ii) Threat Rating
iii) Assigning value to corresponding vulnerability
iv) Identify the concerned affected Asset and give them a value.
v) Estimate the value of Risk

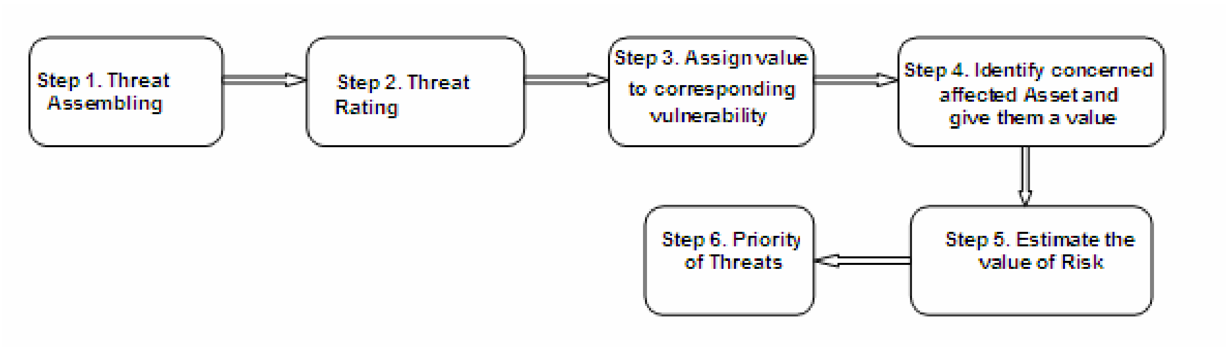

Step 1. Threat Assembling
Step 2. Threat Rating
Step 3. Assign value to corresponding vulnerability
Step 4. Identify concerned affected Asset and give them a value
Step 5. Estimate the value of Risk
Step 6. Priority of Threats

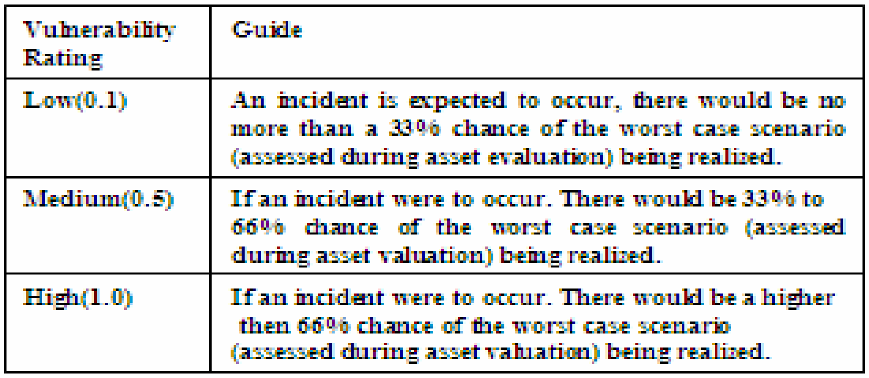
**Figure 13: Process for Threat Priority**

Now all the above steps are defined below and are explained with an example of Railway Reservation system. We have chosen CRAMM method for identification of risk as it offers structured and fast approach to risk analysis over other methods [35].

**Step1. Threat Assembling –** Assemble threats which are a source of each security requirements. As in common criteria based approach [3, 24] we shall be developing the repository of the threats. Actor Profiles will be maintained. Predefined threats can be retrieved from the repository according to the profile of the actor that is defined in section 3.2.

**Step2. Threat Rating –** It is the value which defines the probability of occurrence of particular threat in a project. We have already defined the process of threat assembling, so whatever threats have been identified we have to assign them a value according to CRAMM [29]. These values can be very low (.1), low (.34), medium (1), high (3.33) and very high (10) depending on their occurrence.

**Step3. Assigning value to corresponding vulnerability –** Vulnerability is defined as the weakness in the system that makes an attack more likely to succeed. Value of vulnerability defined by CRAMM [29] will be taken that are low (.1), medium (.5) and high (1). The value can be assigned according to following Table 4.



| Vulnerability Rating | Guide |
|---|---|
| Low(0.1) | An incident is expected to occur, there would be no more than a 33% chance of the worst case scenario (assessed during asset evaluation) being realized. |
| Medium(0.5) | If an incident were to occur. There would be 33% to 66% chance of the worst case scenario (assessed during asset valuation) being realized. |
| High(1.0) | If an incident were to occur. There would be a higher then 66% chance of the worst case scenario (assessed during asset valuation) being realized. |

**Table 4: Measure of Vulnerability**

**Step4. Identify the concerned affected Asset and give them a value –** An asset can be anything that has a value to an organization (e.g. IT systems, information, staff, reputation, goodwill). Different asset in the project is identified and their value is measured by weighing the impact of it when threat will occur.

**Step5. Estimate the risk level –** Risk is defined as the probability that a threat agent (cause) will exploit system vulnerability (weakness) and thereby create an effect detrimental to the system, or in short risk is an unwanted event that has negative consequences on the system. We can say that Risk is defined in one line as (1) –

**Risk= value based on measure of (Threat, Vulnerability, Asset) (1)**

After we have rated the threats, assigned vulnerability value and asset value we will use the 3 dimensional lookup table given by the CRAMM [29] shown in Table 5, where the strength of the threat, the level of the vulnerability and the value of the asset are input parameters, gives the final value of risk in the range 1 through 7.

| Threat Rating | .1 | .1 | .1 | .34 | .34 | .34 | 1 | 1 | 1 | 3.33 | 3.33 | 3.33 | 10 | 10 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Vulnerability/ Asset Value** | .1 | .5 | 1 | .1 | .5 | 1 | .1 | .5 | 1 | .1 | .5 | 1 | .1 | .5 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 3 |
| 2 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 4 |
| 3 | 1 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 4 |
| 4 | 2 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 5 |
| 5 | 2 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 5 | 4 | 5 | 5 |
| 6 | 3 | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 6 |
| 7 | 3 | 4 | 4 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 6 | 5 | 6 | 6 |
| 8 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 6 | 5 | 6 | 6 | 6 | 6 | 7 |
| 9 | 4 | 5 | 5 | 5 | 5 | 6 | 5 | 6 | 6 | 6 | 6 | 7 | 7 | 7 | 7 |
| 10 | 5 | 5 | 6 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 7 | 7 | 7 | 7 | 7 |

**Table 5– Three – Dimension lookup table to measure the level of risk**

### 3.3.2.2 Prioritization of security requirement –

Prioritization of security requirement incurs the following step shown in Figure 14.

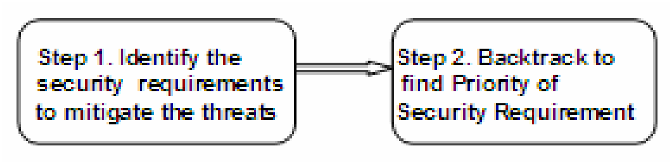

Step 1. Identify the security requirements to mitigate the threats
Step 2. Backtrack to find Priority of Security Requirement

**Figure 14: Security Requirement Prioritization**

**Step1. Identify The Security Requirements –** Identify the various security requirements corresponding to threats [3, 25] so that we can give them priority to mitigate the threats.

**Step2. Backtrack to find out priority of security requirement –** Although till now all the process of analysis and prioritization of requirements are the steps of CRAMM, this activity is the real part of our process.

When we have identified the measures of risk to all the threats and prioritize them based on value of risk. The more the value higher is the priority. We have to consider various cases while prioritization of security threat.

**Case1: Simple Priority** – We have only one asset corresponding to a threat then its risk measure will be calculated based on equation (1). Whatever value comes of risk measure is the priority for that threat.

**Case2: Complex Priority -** We have two assets corresponding to a threat. Then we have two values for this threat corresponding to two assets whose risk values are also comes to be two. So priority for this threat will come by adding the two values.

Finally we have arrived to our real task that is the calculation of priority of Security Requirement. We will calculate the priority of Security Requirement just from the value of threat priority. The following case may arise –

**Case1 –** It is the sum of the value of threat priority, if corresponding to one security requirement there are more than one threat.

**Case2 –** If there is only one threat corresponding to security requirement then what so ever is the value of threat priority that will be assigned to the security requirement.

As we have the priorities of Security Requirement now the developer will check what are the Security Requirement corresponding to the various actors that are listed in Table 3 and correspondingly deal with them.

## 3.4 Management of the requirements

As we manage functional, non- functional and other requirements we have to mange security requirements too. If we do not manage the security requirements with other activities of the system under development we led to a system that will not be efficient and cost effective.

To manage security requirements we have to keep trace of each security requirements and its associated attributes such as requirement identity, view point identity, functional requirement, nonfunctional requirements, threats, design constraint, other security requirement, design constraints etc. This information is modeled in figure 15.

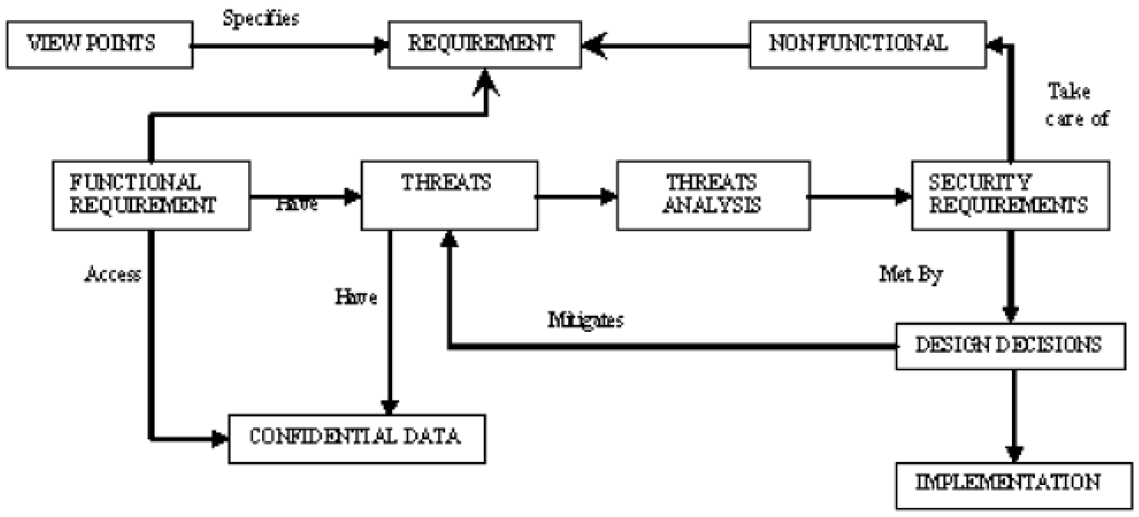


**Figure 15 – Model for Managing Security Requirements**

There are three types of traceability information that must be maintained for the management of security requirements.

- Source Traceability.
- Security Requirement Traceability.
- Design Traceability.

### 3.4.1 Source Traceability

This information for traceability refers to the information of threats on the functionality of the system that misuser can create for the system from where we have derived our security requirements as modeled in figure above the cause of security requirements is the threats to the functionality of the system.

### 3.4.2 Security Requirement Traceability

This traceability information refers to information of

- Functional requirements that are the root cause of the security requirements.
- Interdependent Functional Requirements.
- Interdependent security requirements.

If the main security requirement changes the dependent security requirements also tends to change.

### 3.4.3 Design Traceability

This information about traceability links to the design modules where they are going to be implemented. The information is kept so that the corresponding design decisions that have been taken are actually be implemented.

Traceability information for security requirements can be managed using traceability metrics which relate security requirements to stakeholders (in our case as according to view point), each other and design modules.

# Chapter 4

# CASE STUDY – Railway Reservation System

In this section we go through the development stages of the process defined above using a case study of a CBS system "railway reservation system" which is an online application of normal railway reservation system to explain our process of security requirement prioritization. In this application the customer can book his ticket electronically through internet and can get the ticket on their seat. The customer can do the transactions online through electronic payment gateways. Customer can also inquire about train, cancel reservation, etc.

## 4.1 Requirement Discovery and Definition

Step 1 – *Identify various stakeholders (actors)* - The direct stakeholders of the system will be **Traveller** who want to book the ticket, cancel ticket, make enquiries about train schedule, seat availability, etc., **Railway Management** who will serve to traveller queries and do other operations and the **Database** where the entire operation log will be maintained.

The indirect stakeholders of the system may be the maintenance manger, operations manager etc.

Our interest is in direct actors only.

Step 2 – *Identify Various Functionalities* – The functionalities that are required by different stakeholders are as follows.

**Functionalities of Traveller**

1. Registration of Customer
2. Enquiry of Train
3. Booking of Ticket

4. Cancellation of Ticket
5. Make Payment

**Functionalities of Railway Management**

1. Update Data
2. Serve for Customer Request
3. Maintenance

**Functionalities of Database**

1. Maintain the reports of the transaction

Step 3- *Identify the threats* associated with each of the functional requirements based on the stakeholders profiles. Stakeholders profile has seven fields consisting of name, functionality, type (as according to view point), Physical location (local or Remote), use case association (read, write, store, update etc.) and weather or not the use case involves exchanging private and secret information. For ex – the Traveller profile when the functionality is account registration is shown in Table 6.

| | |
|---|---|
| ***Stakeholder*** | Traveller |
| ***Functionality*** | Account Registration |
| ***Type*** | Direct |
| ***Location*** | Remote |
| ***Private Exchange*** | True |
| ***Secret Exchange*** | False |
| ***Association*** | Write |

**Table 6: Traveller Profile for Account Registration**

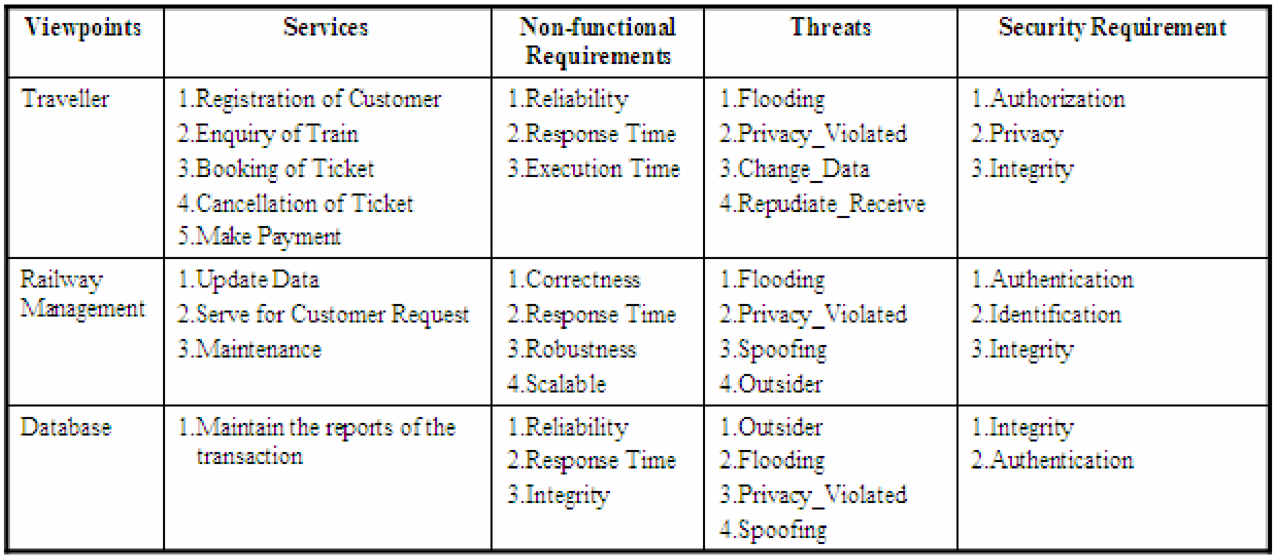
Now the threat to the stakeholders is evaluated based on their profile as shown in Table 7 given below.

| **Association = read** | **Association = write** |
|---|---|
| Impersonate | Change_Data |
| Repudiate_Receive | Repudiate_Receive |
| If(Private Exchange = true) | If(Private Exchange = true) |
| Privacy_Violated | Privacy_Violated |
| If(Secret Exchange = true) | If(Secret Exchange = true) |
| Data_Theft | Data_Theft |
| If(Location = remote) | If(Location = local) |
| Outsider | Insider |

**Table 7: Example showing the evaluation of threats based on Stakeholder Profiles**

Step 4- Once the threats have been identified to the system in question we can define the true security requirements [25] for the system to mitigate those threats.

The detailed list of threats and security requirements after performing this step3 and step4 together is shown below in Table 8.



| Viewpoints | Services | Non-functional Requirements | Threats | Security Requirement |
|---|---|---|---|---|
| Traveller | 1.Registration of Customer<br>2.Enquiry of Train<br>3.Booking of Ticket<br>4.Cancellation of Ticket<br>5.Make Payment | 1.Reliability<br>2.Response Time<br>3.Execution Time | 1.Flooding<br>2.Privacy_Violated<br>3.Change_Data<br>4.Repudiate_Receive | 1.Authorization<br>2.Privacy<br>3.Integrity |
| Railway Management | 1.Update Data<br>2.Serve for Customer Request<br>3.Maintenance | 1.Correctness<br>2.Response Time<br>3.Robustness<br>4.Scalable | 1.Flooding<br>2.Privacy_Violated<br>3.Spoofing<br>4.Outsider | 1.Authentication<br>2.Identification<br>3.Integrity |
| Database | 1.Maintain the reports of the transaction | 1.Reliability<br>2.Response Time<br>3.Integrity | 1.Outsider<br>2.Flooding<br>3.Privacy_Violated<br>4.Spoofing | 1.Integrity<br>2.Authentication |

**Table 8: Example "Railway Reservation System"**

## 4.2 Analysis and Prioritization of the Security Requirements

We have checked for completeness, consistency and group the threats so that it will not cause any error in the system. Then we will start the process of prioritization.

Prioritization will be done in two steps:

- Evaluation of threats
- Prioritization of security requirement.

### 4.2.1 Evaluation of threats –

We will evaluate threats based on the calculated value of risk. In this task we have to follow the following steps:

i) Threat Assembling

ii) Threat Rating

iii) Assigning value to corresponding vulnerability

iv) Identify the concerned affected Asset and give them a value.

v) Estimate the value of Risk

**Step1. Threat Assembling –** Predefined threats can be retrieved from the repository according to the profile of the actor.

Threats identified for our example are –

a. Change_Data

b. Repudiate_Receive

c. Spoofing

d. Flooding

e. Privacy_Violated

f. Outsider

g. Physical

**Step2. Threat Rating –** We have already defined the process of threat assembling, so whatever threats have been identified we have to assign them a value according to CRAMM [29].

| **Threat** | **Level of Threat** | **Value(.1,.34,1,3.33,10)** |
|---|---|---|
| Change_Data | high | 3.33 |
| Repudiate_Receive | medium | 1 |
| Spoofing | high | 3.33 |
| Flooding | medium | 1 |
| Privacy_Violated | low | .34 |
| Outsider | high | 3.33 |
| Physical | very low | .1 |

**Table 9: Measure of various threats**

**Step3. Assigning value to corresponding vulnerability –** All values are project specific and are taken by observation. Value of vulnerability defined by CRAMM [29] will be taken that are low (.1), medium (.5) and high (1).

**Step4. Identify the concerned affected Asset and give them a value –** Different asset in the project is identified and their value is measured by weighing the impact of it when threat will occur. Various assets in our project are:

a) Traveler Information

b) User Login Information

c) Credit Card Information

d) Communication Channels

e) Ticket Information

Various vulnerable assets that will be affected corresponding to particular threat are as in Table 10.

| THREAT | ASSET THAT CAN BE AFFECTED |
|---|---|
| Change_Data | Traveler information, Ticket Information |
| Repudiate_Receive | User Login Information, Credit Card Information, Communication Channel |
| Spoofing | Credit Card Information, Communication Channel |
| Flooding | Credit Card Information, Traveler Information |
| Privacy_Violated | Ticket Information |
| Outsider | Communication Channel, User Login Information |
| Physical | User Login Information |

**Table 10: Possible Vulnerable Assets**

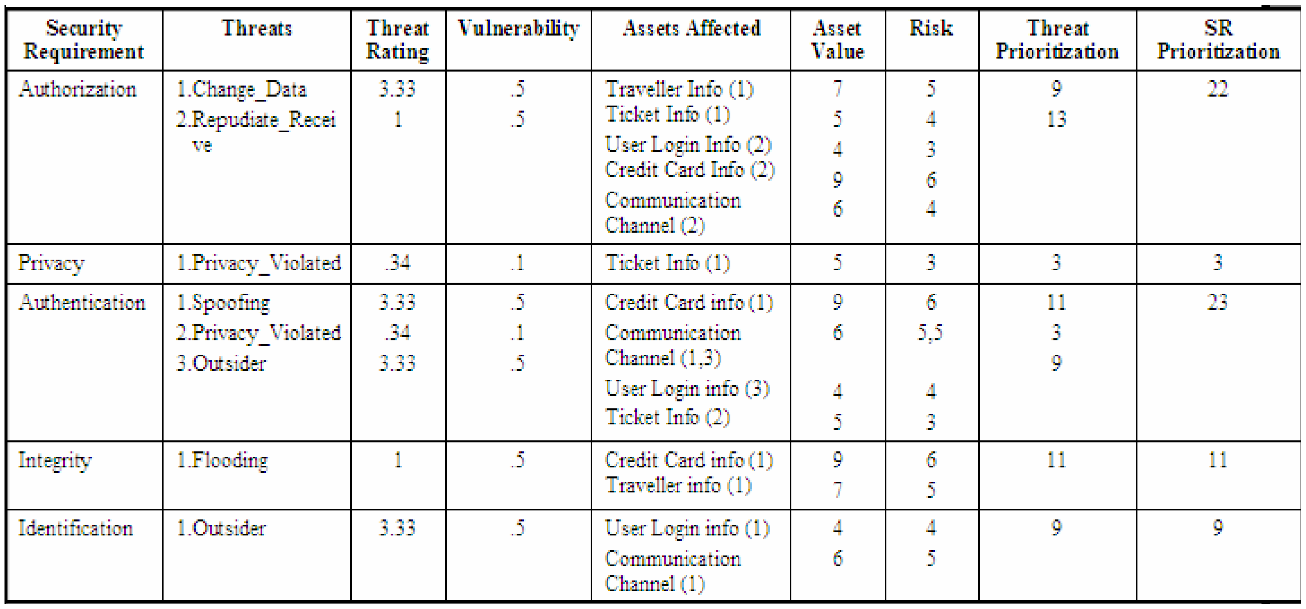
Values of various assets are depicted in Table 11.

| Asset | Value (1 to 10) |
|---|---|
| Traveler information | 7 |
| Ticket Information | 5 |
| Credit Card Information | 9 |
| User Login Information | 4 |
| Communication Channel | 6 |

**Table 11: Measure of Assets**

**Step5. Estimate the risk level –** After we have rated the threats, assigned vulnerability value and asset value we will use the 3 dimensional lookup table given by the CRAMM [29] where the strength of the threat, the level of the vulnerability and the value of the asset are input parameters, gives the final value of risk in the range 1 through 7.

For example – Suppose asset is Ticket Information (5) the threat is Privacy_Violated (.34) and Vulnerability being the low (.1) the measure of risk will be 3 as shown in Table 12. Similarly consider the asset Credit Card Information (9) the threat to this is Spoofing (3.33) and Vulnerability being medium (.5) the measure of risk will be 6. In the similar fashion we can calculate the measure of risk for each threat to an asset and vulnerability and then we can evaluate the threats based on their measure of risk.



| Security Requirement | Threats | Threat Rating | Vulnerability | Assets Affected | Asset Value | Risk | Threat Prioritization | SR Prioritization |
|---|---|---|---|---|---|---|---|---|
| Authorization | 1.Change_Data<br>2.Repudiate_Receive | 3.33<br>1 | .5<br>.5 | Traveller Info (1)<br>Ticket Info (1)<br>User Login Info (2)<br>Credit Card Info (2)<br>Communication Channel (2) | 7<br>5<br>4<br>9<br>6 | 5<br>4<br>3<br>6<br>4 | 9<br>13 | 22 |
| Privacy | 1.Privacy_Violated | .34 | .1 | Ticket Info (1) | 5 | 3 | 3 | 3 |
| Authentication | 1.Spoofing<br>2.Privacy_Violated<br>3.Outsider | 3.33<br>.34<br>3.33 | .5<br>.1<br>.5 | Credit Card info (1)<br>Communication Channel (1,3)<br>User Login info (3)<br>Ticket Info (2) | 9<br>6<br>4<br>5 | 6<br>5,5<br>4<br>3 | 11<br>3<br>9 | 23 |
| Integrity | 1.Flooding | 1 | .5 | Credit Card info (1)<br>Traveller info (1) | 9<br>7 | 6<br>5 | 11 | 11 |
| Identification | 1.Outsider | 3.33 | .5 | User Login info (1)<br>Communication Channel (1) | 4<br>6 | 4<br>5 | 9 | 9 |

**Table 12: Measure of Risk Level and SR Prioritization**

## 4.2.2 Prioritization of security requirement –

Prioritization of security requirement incurs the following step –

- Identify the Security Requirements
- Backtrack to find out priority of security requirement

**Step1. Identify The Security Requirements –** Identify the various security requirements corresponding to threats shown in Table 12, so that we can give them priority to mitigate the threats.

**Step2. Backtrack to find out priority of security requirement –** When we have identified the measures of risk to all the threats and prioritize them based on value of risk. The more the value higher is the priority. We have to consider various cases while prioritization of security threat with reference to Table 12.

**Case1: Simple Priority** – We have only one asset i.e. Ticket Information corresponding to threat i.e. Privacy and its risk measure will be calculated based on equation (1) which is 3, so its priority will also be 3.

**Case2: Complex Priority -** We have two assets i.e. Credit Card Information and Traveller Information corresponding to threat i.e. Flooding, we have two values for this threat corresponding to two assets whose risk values are 6 and 5 as shown in Table 12, so its priority will come by adding two value (6 and 5) i.e. 11.

Finally we will calculate the priority of security requirement just from the value of threat priority. Basically it is the sum of the value of threat priority, if corresponding to one security requirement there are more than one threat. If there is only one threat corresponding to security requirement then what so ever is the value of threat priority that will be assigned to the security requirement. For example, Privacy security requirement has priority 3 as corresponding to this SR we have only one threat that is Privacy_Violated. Now in case of Authorization SR it priority will be the sum of (9 and 13) as corresponding to it there are two threats that are Change_Data and Repudiate_Receive whose values are 9 and 13 respectively.

## 4.3 Management of the Security requirements

To manage the various security requirements we must have to make some more information so that we can trace all the security requirements.

The information should be organized in a proper manner so that for the traceability defined above they can actually be traced.

The traceability information must be maintained in such a way so that we can trace for the following trace abilities.

- **Source Traceability -** We have maintained a database when deriving threats based on the stakeholders profiles as explained in 3.2 (step iii) of the requirement discovery and definition process. From this information stored we can trace each and every security requirement source. This information is stored automatically hence no need to keep ay extra information for source traceability.

- **Security Requirement Traceability –** As in this section we will consider security functional requirements according to CC.

- **Design Traceability –** While taking the design decisions about the implementation of security requirements that mitigate threats we can trace that weather the security requirement is implemented properly or not. This information also helps in checking that each requirement has been properly addressed.

# Chapter 5

# IMPLEMENTATION

## 5.1 Tools Used

***Java Platform Standard Edition 6 Development Kit (JDK 6):* -** JDK 6 provides tools and other utilities that help to develop, execute, debug, and document programs written in the Java programming language. It can be downloaded from Sun Microsystems website.

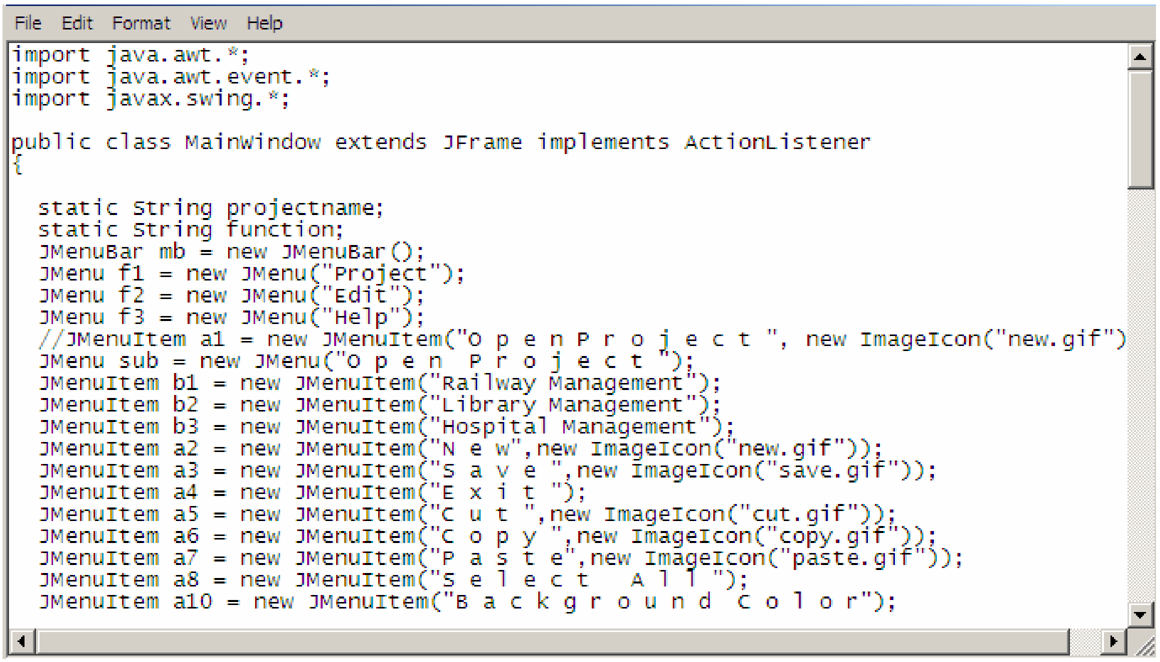
***Microsoft Access:* -** Microsoft access has been used to make relations for our development tool. The main theme to use access is it is a very light weight database and provides all the basic database utilities that we need in our project. We do not want any security feature to the database hence we have used this database.

## 5.2 Files and Relations Used

The code consists of the following java files:

- MainWindow.java
- ActorWindow.java
- createtable.java

**MainWindow.java –** It is the class that provides the main window from where the user can choose to open the desired project from the list of project given. When the user will select project of his choice then the ActorWindow.java would be called.


File Edit Format View Help

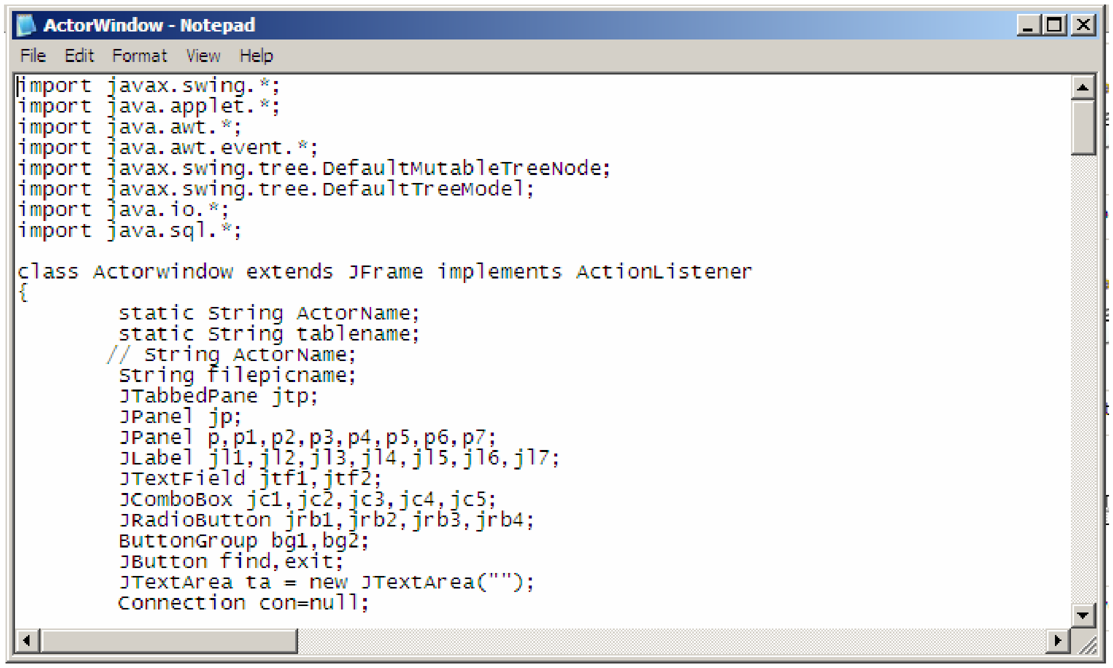
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MainWindow extends JFrame implements ActionListener
{

  static String projectname;
  static String function;
  JMenuBar mb = new JMenuBar();
  JMenu f1 = new JMenu("Project");
  JMenu f2 = new JMenu("Edit");
  JMenu f3 = new JMenu("Help");
  //JMenuItem a1 = new JMenuItem("O p e n P r o j e c t ", new ImageIcon("new.gif")
  JMenu sub = new JMenu("O p e n  P r o j e c t ");
  JMenuItem b1 = new JMenuItem("Railway Management");
  JMenuItem b2 = new JMenuItem("Library Management");
  JMenuItem b3 = new JMenuItem("Hospital Management");
  JMenuItem a2 = new JMenuItem("N e w",new ImageIcon("new.gif"));
  JMenuItem a3 = new JMenuItem("S a v e ",new ImageIcon("save.gif"));
  JMenuItem a4 = new JMenuItem("E x i t ");
  JMenuItem a5 = new JMenuItem("C u t ",new ImageIcon("cut.gif"));
  JMenuItem a6 = new JMenuItem("C o p y ",new ImageIcon("copy.gif"));
  JMenuItem a7 = new JMenuItem("P a s t e",new ImageIcon("paste.gif"));
  JMenuItem a8 = new JMenuItem("S e l e c t   A l l ");
  JMenuItem a10 = new JMenuItem("B a c k g r o u n d  C o l o r");
```

**Figure 16: Showing the MainWindow code**

**ActorWindow.java -** This file and helps the user to complete the actor profile based on seven fields described earlier though a simple user interface so that the user can complete the actor profile easily. After the actor profile is completed this file calls createtable.java.

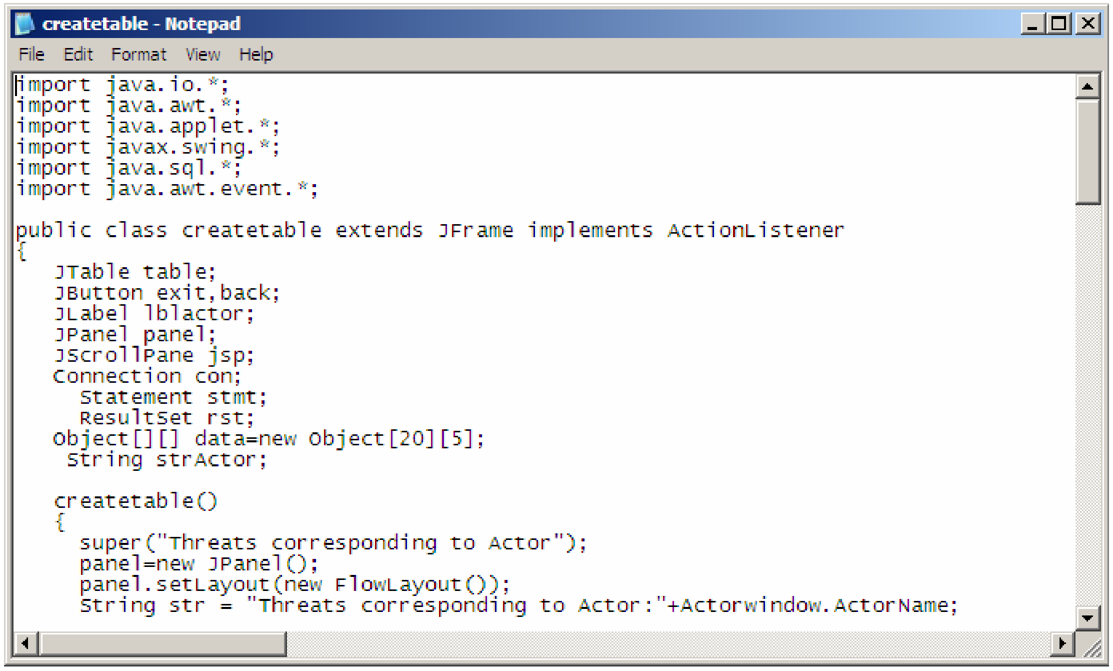


```
import javax.swing.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;
import java.io.*;
import java.sql.*;

class Actorwindow extends JFrame implements ActionListener
{
        static String ActorName;
        static String tablename;
       // String ActorName;
        String filepicname;
        JTabbedPane jtp;
        JPanel jp;
        JPanel p,p1,p2,p3,p4,p5,p6,p7;
        JLabel jl1,jl2,jl3,jl4,jl5,jl6,jl7;
        JTextField jtf1,jtf2;
        JComboBox jc1,jc2,jc3,jc4,jc5;
        JRadioButton jrb1,jrb2,jrb3,jrb4;
        ButtonGroup bg1,bg2;
        JButton find,exit;
        JTextArea ta = new JTextArea("");
        Connection con=null;
```

**Figure 17: Showing the code of ActorWindow code**

**createtable.java –** It is the file which will display the output what are the various threats, their threat rating, vulnerability, asset value, etc in the form of table.



```
import java.io.*;
import java.awt.*;
import java.applet.*;
import javax.swing.*;
import java.sql.*;
import java.awt.event.*;

public class createtable extends JFrame implements ActionListener
{
   JTable table;
   JButton exit,back;
   JLabel lblactor;
   JPanel panel;
   JScrollPane jsp;
   Connection con;
     Statement stmt;
     ResultSet rst;
   Object[][] data=new Object[20][5];
    String strActor;

   createtable()
   {
     super("Threats corresponding to Actor");
     panel=new JPanel();
     panel.setLayout(new FlowLayout());
     String str = "Threats corresponding to Actor:"+Actorwindow.ActorName;
```

**Figure 18: Showing the code of createtable code**

The **database** used for the projects consists of the following relations:

**Threat –** Used to maintain the threats associated with the actors based on there profile.

**Function –** Used to maintain the functionality of various actors corresponding to various projects.

**Risk –** Used to implement the 3-D table of CRAMM for the purpose of risk value identification corresponding to various threats.

**Security -** Used to maintain all the possible security requirements.

## 5.3 Running the Code

First of all you need to create a DSN connection so that the database can be accessed through JDBC. Give the name of DSN threats. To Create the DSN do the following steps:-

- Control Panel.
- Administrative tools.
- ODBC.
- Click System DSN tab.
- Click on the ADD Button select from a list of available drivers. In our case it is
- Microsoft Access Driver (*.mdb).
- Give the appropriate name to the DSN and select the database where relations are stored.
- Click the FINISH Button.

The DSN will be created with the name given and now the JDBC can be used for executing the queries.
To run the code you need to have JDK 6 installed on the system on which you want to run the code. To run the code under windows environment open command prompt window and do the following steps

1. Go to the directory where all files are present
2. Then, compile the files, type **javac <filename>.java**.
3. And, to run the code, type **java Front**

The path and class path environment variables must be set to the bin directory where **javac** and **java** are present so that javac and java can be executed from any directory.

## 5.4 Snapshots

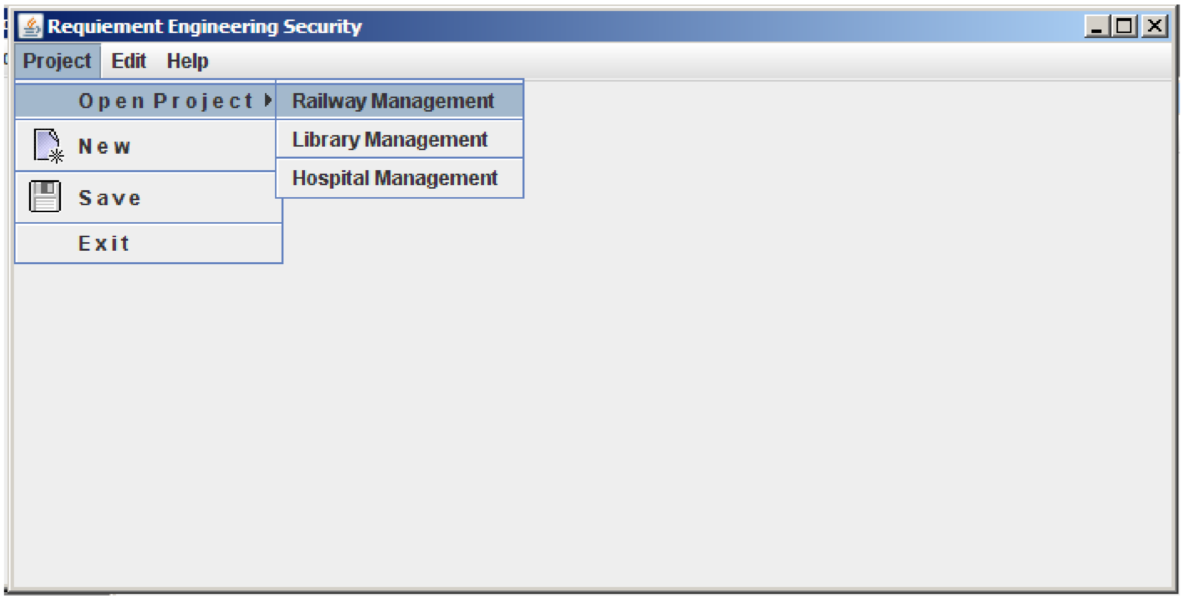
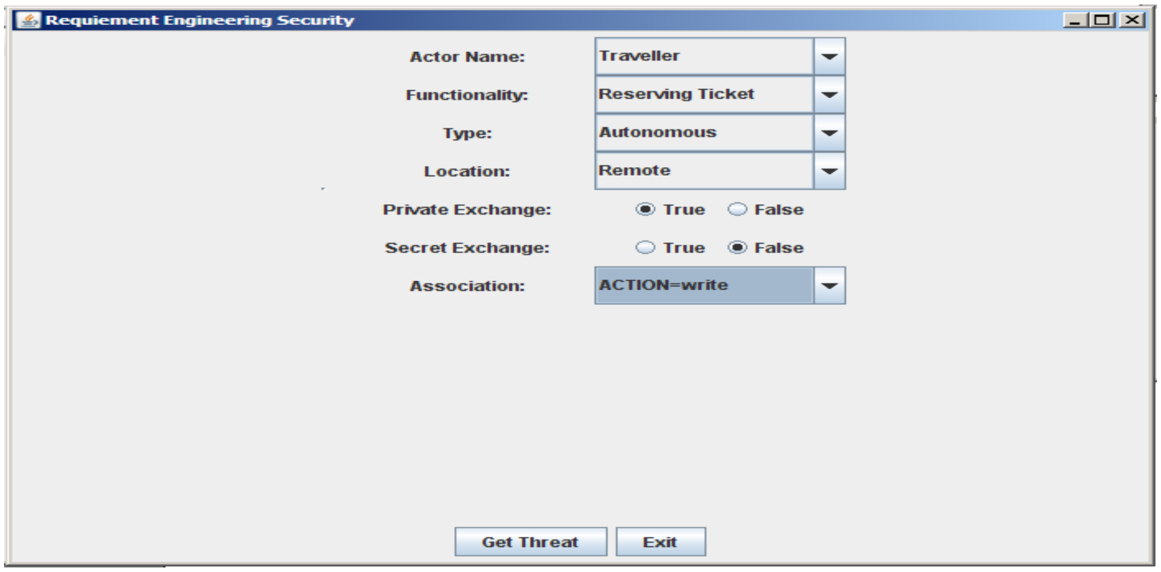
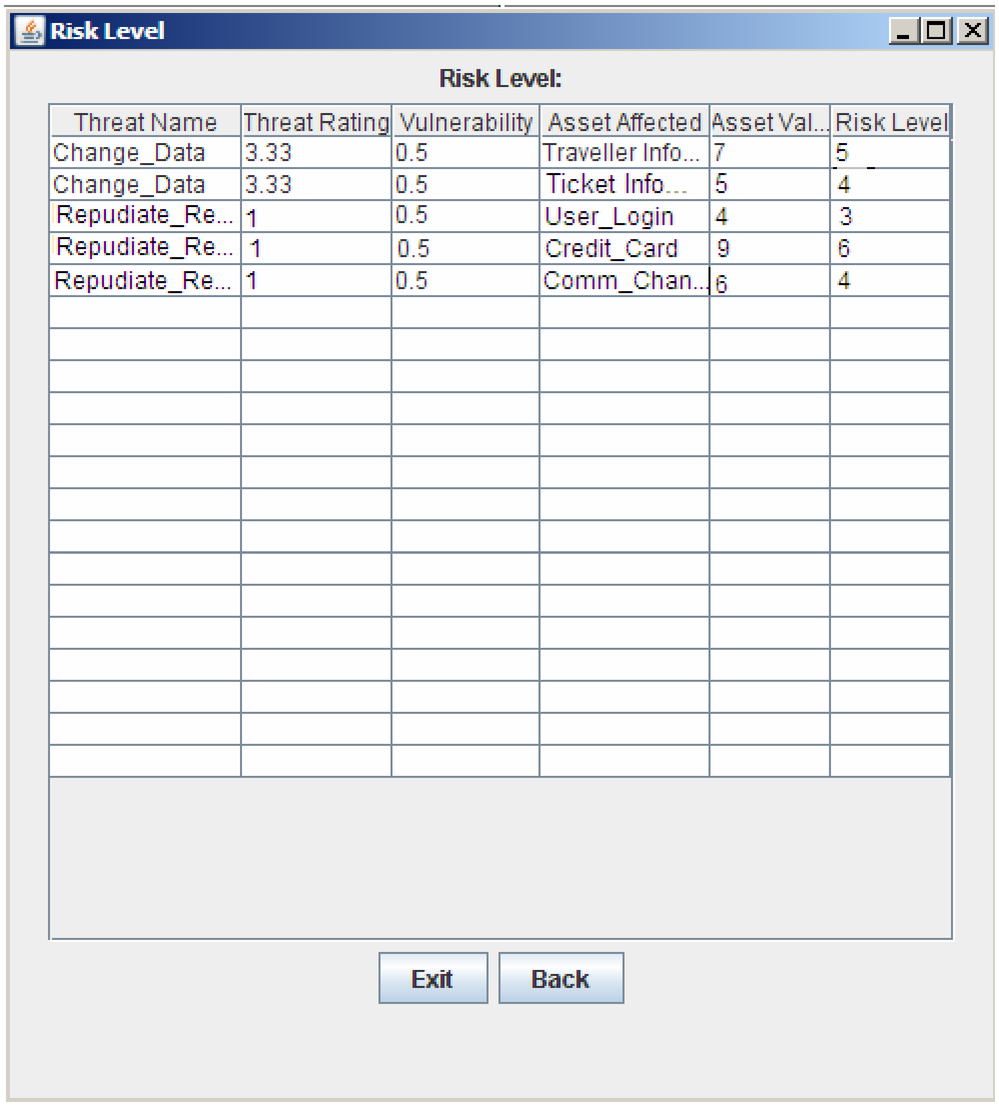


**Figure 19: Main Window**



| | |
|---|---|
| Actor Name: | Traveller |
| Functionality: | Reserving Ticket |
| Type: | Autonomous |
| Location: | Remote |
| Private Exchange: | [✓ True] False |
| Secret Exchange: | True [✓ False] |
| Association: | ACTION=write |

**Figure 20: Actor with seven fields as specified in common criteria**



| Threat Name | Threat Rating | Vulnerability | Asset Affected | Asset Val... | Risk Level |
|---|---|---|---|---|---|
| Change_Data | 3.33 | 0.5 | Traveller Info... | 7 | 5 |
| Change_Data | 3.33 | 0.5 | Ticket Info... | 5 | 4 |
| Repudiate_Re... | 1 | 0.5 | User_Login | 4 | 3 |
| Repudiate_Re... | 1 | 0.5 | Credit_Card | 9 | 6 |
| Repudiate_Re... | 1 | 0.5 | Comm_Chan... | 6 | 4 |

**Figure 21: Showing the Level of Risk Corresponding to Threats**

Chapter 6

# CONCLUSION

The work carried out in this report provides with strong base in security requirement engineering. Here we have presented the techniques for discovering security requirement along with functional and non-functional requirement in additional we have shown a novel approach to prioritize these security requirement based on threat analysis. Our main emphasis is on to discovering the security requirement as early as possible so that the system under development is efficient and less vulnerable which is the need of the day in current scenario since the system in today's world are the target of hackers, malicious crackers which is not an option since the society relies heavily on them not on the design and implementation phase.

The process that we have defined for the elicitation of security requirements is seamlessly integrated with the conventional process of requirement engineering with viewpoints as specified by Sommerville. The novel approach defined in this work is a betterment of the existing approaches that are normally used for the purpose of elicitation of security requirements. Also there is no other method to analyze and prioritize.

Our approach is different from Misuse Case approach and common criteria. As we have considered both functional and non functional requirements to derive security requirements.

The tool that has been developed helps in the automatic generation of threats as well as security requirements and their prioritization of the system under development. The automatic generation of threats and security requirements has an advantage as considered to manual process because the requirement engineer when doing the process manually may not consider some of threats that are very critical for the system.

# Chapter 7

## PUBLICATIONS

During the period of working over this project we interacted with International community working on requirement engineering. We discussed our approach for prioritization with them and collected the reviews and worked over the suggestion send to us. One Research papers have been accepted in International conference for presentation and will be published in their proceedings.

This paper presents the concept of security requirement prioritization.

### 7.1 The details of Conference publications:

**Conference Name:** *International Conference on Software Engineering and Research Practices (SERP-09), Las Vegas, USA.*

**URL:** **http://www.world-academy-of-science.org/**

**Paper Title:** *"Security Requirement Prioritization"*

**Authors:** Dr. Daya Gupta, Shruti Jaiswal

**Location:** Monte Carlo Resort, Las Vegas, Nevada, USA

**Publishers/ proceedings:** The accepted papers will be included in the conference proceedings, which has an ISBN number. The proceedings will be made available during the conference. The proceedings will also be submitted for several database indexes. The previous conferences are submitted for several reputed database indexes. Paper will be indexed at **DBLP** Bibliography Server.

# Chapter 8

## REFERENCES

[1] Alexander IF, “Modelling the interplay of conflicting goals with use and misuse cases”. In: Proceedings of the 8th international workshop on requirements engineering: foundation for software quality (REFSQ’02), Essen, Germany, 2002.

[2] Alexander IF, “Misuse cases, use cases with hostile intent”. IEEE Software, 2003, pages 58– 66

[3] Common criteria for information technology security evaluation. Technical report CCIMB 99–031, Common Criteria Implementation Board, 1999.

[4] John Mc Dermott, Chris Fox, “Using abuse case models for security requirements analysis.” Department of Computer Science, James Madison University, 1999.

[5] European Computer Manufacturers Association International, ECMA protection profile: ECOFC public business class. Technical report, TR/78, Geneva, Switzerland, 1999.

[6] Robert J. Ellison, "Attack Trees” Software Engineering Institute, Carnegie Mellon University, 2005.

[7] Donald G. Firesmith, “Engineering Security Requirements”, Journal of object technology, 2003, vol 2, no.1, pages 53-68.

[8] Donald G. Firesmith, “Security Use cases”, Journal of object technology, 2003, vol 2, no.3, pages 53-64.

[9] Gupta D. and Prakash N., “Engineering Methods from their Requirements Specification”, Requirements Engineering Journal 2006, 3, pages 133 – 160.

[10] Ian Hawkins, “Risk Analysis Techniques”, 1998 Available at http://www.euclidresearch.com /current.htm.

[11] Johnson, J. "Chaos: The Dollar Drain of IT Project Failures," Application Development Trends, January 1995, pp. 41-47.

[12] Lubars M., Potts C., Richer C., “A review of the state of the practice in requirements modeling”, Proc. IEEE Symp. Requirements Engineering, San diego 1993

[13] Nancy R. Mead, “Requirement Elicitation Introduction”, Software Engineering Institute, Carnegie Mellon University, 2006.

[14] Karen Mc Graw, Karan Harbison, “User Centered Requirements, The scenario based” , 1997

[15] Prakash N, Sibal R, “Modelling method heuristics for better quality products, advanced information systems engineering”. Springer, Berlin Heidelberg New York, 1999, pages 429– 433.

[16] Prakash N. , “On generic Method Models ”, Requirements Engineering Journal 2006, pages- 221 – 237.

[17] Potts C Scenario noir (panel statement, p2). In: Proceedings of the symposium on requirements engineering for information security (SREIS’01), Indianapolis Engineering process. Lawrence Erlbaum Associates Publishers, 1997.

[18] Sindre G, Opdahl AL, “Eliciting security requirements by misuse cases”. In proceeding 37th Conference Techniques of Object-Oriented Languages and Systems, TOOLS Pacific 2000, pp 120-131.

[19] Sindre G, Opdahl AL, “Templates for Misuse Description”. Proceeding 7th Int’l Workshop Requirements Eng.: Foundation for Software Quality (REFSQ 2001), 2001.

[20] Sindre G, Opdahl AL, “Eliciting security requirements with misuse cases”. Requirements Engineering 10, Springer-Verlag London Ltd, January 2005, pp. 34-44.

[21] Kotonya G., Sommerville I., “Requirement Engineering with view points”, 1995.

[22] Sommerville, I., “Software Engineering”. Seventh edition 2003. ISBN - 8129708671. Pearson Education.

[23] The Standish group, Chaos. Standish Group Internal Report, 1995 http://www/standishgroup.com/chaos.html.

[24] M. Ware, J. Bowles, C. Eastman, “Using the common criteria to Elicit security Requirements with use cases”, 2006 IEEE Computer Society.

[25] Agarwal A, Gupta D, “Security Requirement Elicitation Using View Points for online System”. 2008 IEEE Computer Society.

[26] CORAS - http://www2.nr.no/coras.

[27] N. Mayer, P. Heymans, R. Matulevičius “Design of a Modelling Language for Information System Security Risk Management”, In Proceedings of the First International Conference RCIS – 2007.

[28] Alberts, Christopher and Dorofee, Audrey. *OCTAVE* Method Implementation Guide v2.0. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. http://www.cert.org/octave

[29] The Logic behind CRAMM's Assessment of Measures of Risk and Determination of Appropriate Countermeasures available at www.cramm.com

[30] a) Paolo Giorgini, G.Manson, Haralambos Mouratidis. I.Philip, "A Natural Extension of Tropos Methodology for Modelling Security". In the workshop on Agent-oriented methodologies, at OOPSLA 2002.

b) Paolo Giorgini, G.Manson, Haralambos Mouratidis. I.Philip, "Modelling Secure Multi agent System". AAMAS - 2003.

[31] EBIOS- Expression of need and identification of security objectives , DCSSI, France, February,2004.

[32] MEHARI- (Information Risk Analysis and management Methodology), V-3, Coceps and Mechanism, CLUSIF, October, 2004.

[33] A. van Lamsweerde, Elaborating security requirements by construction of intentional anti-models, in: Proceedings of the 26th International Conference on software engineering (ICSE'04), IEEE Computer Society, Washington, DC USA,2004, pp. 148-157.

[34] CERT, http://www.cert.org

[35] W.G. Bornman, L. Labuschagne, "A Comparative Framework for Evaluating Information Security Risk Management Methods", (*icsa.cs.up.ac.za/issa/2004/ Proceedings/Full/015.pdf*)