VIEW POINT APPROACH FOR ENGINEERING SECURITY REQUIREMENTS

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

Master of Engineering

In

Computer Technology and Application

By

Ashish Agarwal

College Roll No. - 01/CTA/06

University Roll No. - 10181

Under the esteemed guidance of

Dr. Daya Gupta

Department of Computer Engineering

Delhi College of Engineering

University of Delhi

2007-2008

CERTIFICATE

DELHI COLLEGE OF ENGINEERING

(Govt. of National Capital Territory of Delhi)

BAWANA ROAD, DELHI - 110042

Date:_____

This is to certify that dissertation entitled "**View Point approach for engineering Security Requirements**" has been completed by Mr. **Ashish Agarwal** in partial fulfilment of the requirement of **Master in Engineering** in **Computer Technology & Application**.

This is a record of his work carried out by him under my supervision and support during the academic session 2007 -2008. 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

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 supervisors Dr. Daya Gupta, Assistant Professor, Department of Computer Engineering, Delhi College of Engineering, for their 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.

I would also like to take this opportunity to present my sincere regards to my teachers Prof. D.Roy Choudhary, Mrs. Rajni Jindal, Dr. S. K. Saxena and Mr. Manoj Sethi for their support and encouragement.

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

(ASHISH AGARWAL)

Master of Engineering

(Computer Technology & Application)

Dept. of Computer Engineering

DELHI COLLEGE OF ENGINEERING

Table of Contents

List of Figures

List of Tables

ABSTRACT

Requirements elicitation for a new system requires extensive involvement with stakeholders who usually have varying aims, backgrounds and disciplines, and this process is complicated further if the system has critical security, or other non-functional requirements. Established approaches to the elicitation and analysis of functional and non- functional requirements are very different, the former focusing on boundary behavior and the later on threats to assets.

The development of functional requirements focuses on how a system interacts with its environment, while the development of security requirements focuses on risk, in particular the risk of particular unwanted outcomes to the business, and its assets. Functional requirements are clearly behavior-led, whereas security requirements are asset driven. As a consequence these processes are rarely fully combined; security is often dealt after the requirement engineering process. As a result of this the requirement engineer ends up in specifying architectural constraint rather than true security requirements.

Aim is to elicit and manage requirements for functionality and security as part of a single integrated process, and recognize that there will be interaction and feedback between them. We defined such process that will have well articulated steps for security engineering presenting techniques for activities like requirement discovery, analysis, prioritization, and management. With true security requirements systematically identified, architecture team can choose most appropriate mechanism to implement them.

1. INTRODUCTION

In the last decade Security has been a great concern for software engineering community in the development of system such as e-commerce, military system, online business, component engineering [5, 8, 18]etc. Insecure system is subjected to infection by virus, malicious crackers and various other threats of cyber terrorism. Besides having safety, reliability and other quality features these systems may not be acceptable as one can not depend on them.

1.1 General Concepts

Computer security [7] is defined as technological and managerial procedures applied to computer systems to ensure the availability, integrity and confidentiality of information managed by the computer system.

Security of software system means protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications).

There are following concerns related to security –

- Software security is an integral part of sound management of the organization.
- Software Security should be efficient.
- Software security requires a comprehensive and integrated approach.

Computer systems are vulnerable [13] to many threats that can inflict various types of damage resulting in significant losses. This damage can range from errors harming database integrity to fires destroying entire computer centers. Losses can range, for example, 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 many losses are never discovered, and others are "swept under the carpet" to avoid unfavorable publicity. The affects of various threats varies considerably: some affect the confidentiality or integrity of data while others affect the availability of a system.

Overlooking Software security is not an option since society relies heavily upon them. 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 for 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 can make a potential impact on the organizations as well as the persons who are using these systems.

However, software-intensive 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, 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 process of development of any computer based system (CBS) the first and the most important step is gathering requirements. Requirement engineering [21, 22] is a difficult task and any fault in this task 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 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 LibSys the librarian can issue the books, student can search the books). It is also very necessary to gather non-functional 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 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 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.

Therefore we focus in this thesis to view point oriented approach for security requirement engineering which can be well embedded in requirement engineering process. One of the well defined processes of requirement engineering is view point oriented requirements definition (VORD) [21, 22]. Analogous to this VORD process for requirement engineering we develop a process of security requirement engineering too which we name as view point oriented security requirement elicitation (VOSREP). Therefore we have chosen the topic of research as **"View Point Approach for Engineering Security Requirements.**"

Our literature study shows that there are number of proposals for eliciting, specifying security requirements. In the next section we give an overview of them.

1.3 Related Work

Computer system security attacks are one of the most urgent problems facing IT professionals today. There are various techniques for addressing security requirements during the early phases of Software Development Life Cycle (SDLC). These includes attack trees [6], abuse case [4], misuse case [1, 2], security use case [8] etc. They specify requirements using templates but these proposals of security requirements elicitation are not embedded in conventional requirements engineering process. Also they do not address security requirements managements. We here present state of art techniques for addressing security requirements that are used during the early phases.

1.3.1 ***Attack Trees***

Attacks trees [Ellison 6] are a way to represent the attacks using the most widely used data structure Trees. 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.

1.3.2 *Abuse Cases*

Abuse case [Dermott 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.

Abuse cases can be described using the same strategy as for use cases: use case diagrams and use case descriptions. We do not use any special symbols for abuse cases in diagrams, that is, an abuse case diagram is drawn with the same symbols as a use case diagram.

1.3.3 *Misuse Cases*

This approach 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 [1, 2, 19, 20] apply the concept or behavior that the system's owner does not want to occur. Use case diagrams are driven by goals of the system misuse are driven by threats to the system. Misuse cases for a system are shown on a single diagram the only difference is that they use inverted graphics to represent misuse case diagrams.

1.3.4 *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.

1.3.5 ***Common Criteria (CC) with use cases***

This approach [Eastman 24] specifies how standards such as common criteria 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.

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 name, type, location, use case association and weather or not the use case involves exchanging private and secret information. 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.

Threats once derived maps to the security objectives which again maps to security requirements. This is how true security requirements are identified. Once the true security requirements are identified the architecture and design team can choose the most appropriate technique to implement corresponding security mechanism.

All these proposals for security requirements engineering do not have

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

1.4 Proposed Work

The process that we will describe here is an extension of a well defined process of requirement engineering called VORD (view point oriented requirement definition) [21] to address security requirements and it will resolve all the issues discussed earlier. That is it will have well articulated steps for security engineering, will be embedded in Requirement Engineering process, will also address security requirements management. The process we

will describe will be called as VOSREP (view point oriented security requirement elicitation) since we have integrated it with the view point process of requirement engineering.

The different activities in VOSREP [9,10] 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 [26, 27, 28, 29] such as CRAMM to quantify the risk of threats on the assets .

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 elicit security requirements associated with the functionality required by the stakeholders. This tool is an extension of VIOLET [24] (Open Source) software that is used for making UML Diagrams.

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.

1.5 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. 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 clears 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 in chapter 2 addresses we describe different types of security requirements for any computer based systems and what are mechanism to implement them.

Chapter 3 our process i.e. View Point Oriented Security Requirement Elicitation Process (VOSREP) in detail explaining different activity of the process.

Chapter 4 explains the VOSREP approach for engineering security requirements taking the case study of “Online Book Store”.

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

Chapter 6 concludes the thesis.

Chapter 7 lists the papers that have been published during the preparation of the thesis.

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

2. REQUIREMENTS ENGINEERING AND SECURITY REQUIREMENTS ENGINEERING

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.

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 user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.

- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to track the order of the items purchased by them.

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.

Example –

The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.

Organisational requirements

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

Example –

The format for representing the account in a BANKSYSTEM should be consistent to the standard defined by the bank.

External requirements

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

Example –

The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

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.

Examples –

The domain requirement for a LIBSYS can be that there shall be a standard user interface to all databases which shall be based on the ISO standard

"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 Engineering

It is the process of identification of requirements other than functional, non-functional and domain requirements. These requirements are very important since they are very critical for the successful operation of the system. 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 engineering.**

"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 [9]. They differ from safety

requirements which are domain specific and more suitable for control systems application. They are also kwon 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.

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.

Architecture Mechanism –

- Digital possessions such as a digital certificate or token.
- Physical possessions such as an employee ID card, a hardware key, or a smart card enabled with a public key infrastructure (PKI).

- Physiological traits (e.g., finger print, hand print, face recognition, iris recognition, and retina scan).
- Behavioral characteristics (e.g., voice pattern, signature style, and
- keystroke dynamics)

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.

Architecture Mechanism –

- Digital possessions such as a digital certificate or token.
- Physical possessions such as an employee ID card, a hardware key, or a smart card enabled with a public key infrastructure (PKI).
- Physiological traits (e.g., finger print, hand print, face recognition, iris recognition, and retina scan).
- Behavioral characteristics (e.g., voice pattern, signature style, and
- keystroke dynamics)

Note that some of the above authentication security architecture mechanisms can be used to simultaneously implement both identification and authentication requirements

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.

Architecture Mechanism –

- Authorization lists or databases.
- Person vs. role-based vs. group-based authorization.
- Commercial intrusion prevention systems.
- Hardware electronic keys.
- Physical access controls (e.g., locks, security guards).

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.

Architecture Mechanism –

- Antivirus programs.
- Firewalls.
- Programming standards (e.g., for ensuring type safety and array bounds checking).

- Prohibition of type-unsafe languages (e.g., C) that allow buffer overflows.

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.

Architecture Mechanism –

- Cryptographic Techniques.
- Hash Functions.

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.

Architecture Mechanism –

- Alarms.
- Event Reporting.
- Use of a specific commercial-off-the-shelf (COTS):

- Intrusion Detection System (IDS).
- Intrusion Prevention System (IPS).

2.2.7 Nonrepudiation 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.

Architecture Mechanism –

- Authenticated identity of all parties involved in the transaction.
- Date and time that the interaction was sent, received, and acknowledged (if relevant).
- Significant information that is passed during the interaction.

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.

Architecture Mechanism –

- Public or private key encryption and decryption.
- Commercial-off-the-shelf cryptography packages.

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

Architecture Mechanism –

- Audit Trails.
- Event Logs.

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.

Architecture Mechanisms -

- Hardware redundancy.
- Data center redundancy.
- Failover software.

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.

Architecture Mechanism –

- Maintenance and enhancement procedures.
- Associated training.
- Security regression testing

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 organisation) 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 organisation. 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:

1. Prune the viewpoint class hierarchy to eliminate viewpoint classes which are not relevant for the system under question.
2. 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.
3. 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.
4. 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.
5. 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 non-functional 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 Conclusion

In the above section we have explained requirement engineering, Security Engineering, and different types of security requirements. We also explained the VORD process for requirement engineering as given by Sommerville. This section provides background knowledge to understand the later sections.

3. VIEW POINT ORIENTED SECURITY REQUIREMENT ELICITATION PROCESS(VOSREP)

After establishing the foundation stone of security engineering, various types of security requirements and viewpoint oriented requirements definition (VORD) for requirement engineering, we now present our process for security engineering with view points. We call the process as VOSREP i.e. "***VIEW POINT ORIENTED SECURITY REQUIREMENT ELICITATION PROCESS".*** The VOSREP process defined is well embedded in VORD process making security engineering a unified approach with requirement engineering. Hence we can deal with security requirements as we deal with other functional and non –functional requirements.

Our Security requirements elicitation [9, 10] process VOSREP is based on following observation from forgoing section:

- Implementation of Security mechanisms effectively mitigate threats which can be considered as special kind of risk. Hence they can be assessed and analyzed using techniques from Risk assessment and risk analysis [12].

- Security requirements are driven from functionalities and data which are accessed by user of the system which may be internal or external to the system.

- Non functional requirements to some extent avoid security threats or cover security requirements.

- Security requirements are related to each other. For ex. - authorization requirements require existence of both identification and authentication requirements.

IN the VOSREP process, we give the techniques to elicit, analyze, prioritize and manage security requirements.

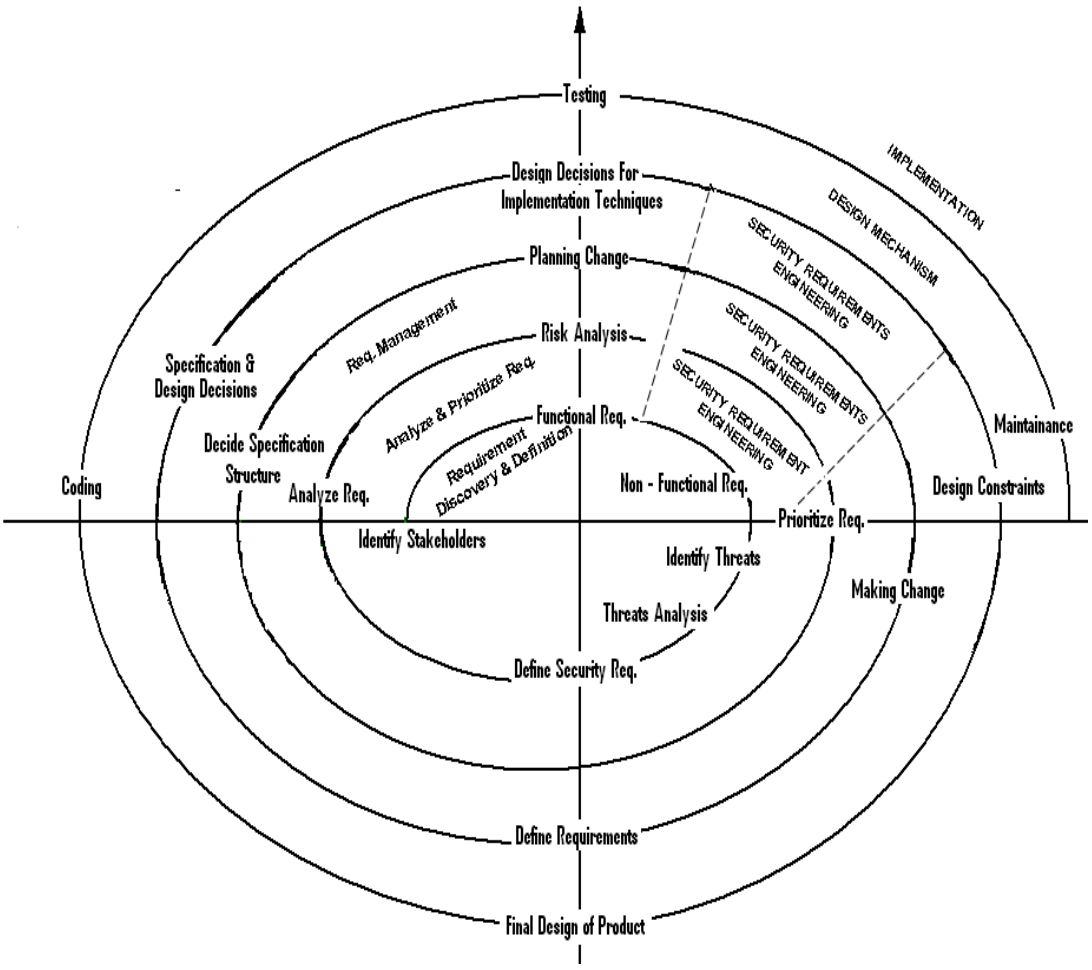

Figure 1 - Different Tasks in Security Engineering

3.1VOSREP

The different activities in the VOSREP as shown in Figure 1 are as follows: -

Security Requirements Discovery and Definition

It is the first activity of the VOSREP process. In this step the security requirements along with functional and non functional requirements are discovered and defined for the system to be developed. In VOSREP we extend the conventional VORD process for requirement engineering so that we can elicit the corresponding security requirements. We use the concept of stakeholders and the threat that different stakeholders can cause to the system. From these threats we can determine the security requirements.

In section 3.2 we discuss this activity in detail.

Analysis and Prioritization of Security Requirement

In this activity we analyze the various security requirements discovered in step 3.1.1 above for their completeness, Consistency, Unambiguousness, Feasibility etc as we analyze other requirements. Once the security requirements are analyzed the corresponding security requirements are prioritized based on the measure of risk of threat on an asset. For measuring risk there are various techniques such as OCTAVE [27], CORAS [26], CRAMM [28, 29] etc.

In particular we will use CRAMM [28, 29] method of measuring risks to prioritize our security requirements. The main aim of prioritization of security requirements is that only those security requirements should be considered which are under the budget of the project. In other way we can say that we can make the system under development to be cost effective.

In section 3.3 we discuss this activity in detail explaining how we can apply CRAMM to prioritize security requirements.

Management of Security Requirements

After the security requirements are discovered, analyzed, prioritized we have to mange them. This is very important activity of the security engineering. As we know that functional and non – functional requirements tend to change during the course of the project same is the case with security requirements too. They also change and grow up during the entire project.

Hence it is very essential that the corresponding security requirements should be managed properly so that in further stages they don't grow up making the system under development a failure.

Section 3.4 will discuss this activity in detail.

All the activities defined in the forgoing section are spiral in nature (Figure 1). 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 figure1 show the three activities of the VOSREP process described above as shown along the radial dimension of second quadrant. After our VOSREP 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.

3.2 Security Requirement Discovery and Definition

This is the first activity of the VOSREP process and is the first spiral of the figure1shown above. In this section we explain this activity in detail.

The different steps in this activity are:

i. Identify various stakeholders (actors) of the system using view-point analysis (Figure 2). We have identified the various abstract classes of actors as direct and indirect actors. 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. Our interest is in direct actor. For detail classification of actors refer section 2.3 above.

For ex- for a LIBSYS the direct actors can be Students, Library Staff etc. while the indirect actors can be the booksellers from where books are purchased.

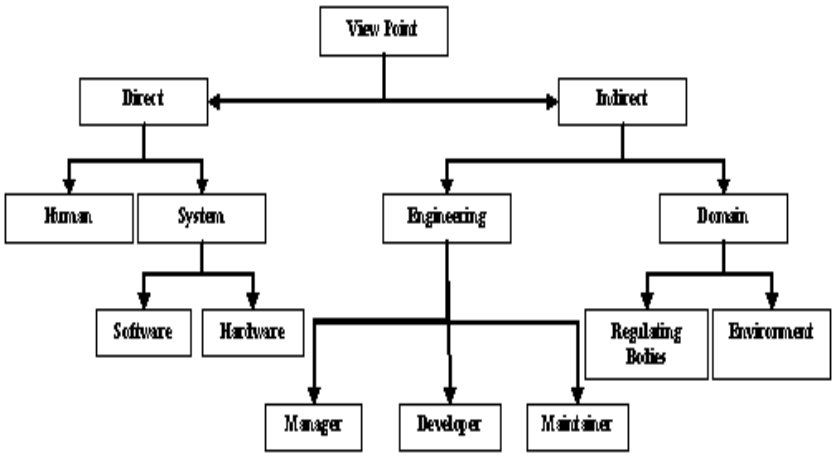

Figure 2- Different types of stake holders as according to view point

ii. Identify the functionalities of each actor conceptualized in step i. Also determine associated non - functional requirements.

For ex- The functionality of student can be issue books, return books etc, the functionality of librarian is to check inventory, collect fine, while the booksellers have to supply the orders on time.

iii. Identify the threats associated with each of the functional requirements or data which is used by this functionality. As in common criteria based approach we shall be using predefined repository of the threats. Actor Profiles will be maintained as in common criteria based approach used for eliciting security requirements. Our actors will be classified as mentioned in view point analysis. Also threats will be classified, based on functionality and the actor kind predefined threats can be retrieved from the repository according to the profile of the actor. Define Security requirements which will mitigate these threats.

To identify true security requirements first of all we will identify threats. To identify threats we will make a repository of threats. For each stakeholder we will make a stakeholders profile [24] that helps in the automatic generation of threats from the repository made by us. The profile of the stakeholder will be based on seven fields.

- Actor Name – Ex- Student, Customer
- Use Case – Ex- Check Results. Display Books
- Type – Ex – Direct, Indirect etc.
- Location – Local Or Remote
- Private Exchange – Yes or No
- Secret Exchange – Yes or No
- Association – read, write, ask, answer, retrieve, store, send, display, update etc.

In general we can say hat to identify threats we have correlated the Common Criteria (CC) with use case diagrams for the generation of threats based on stakeholders profile described above.

The repository that we have defined will be limited to the following category of threats [5, 24] shown in table 1.

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

Table 1– 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 (Table 2) [3] which are mapped to security requirements (Table 3) [3] and this is how true security requirements are identified.

	Threat Name	**Security Objective**
1.	T.Change_Data	***O.Access_Control*** ***O.Authen*** ***O.Integrity*** ***O.Recover*** ***O.Sequence*** ***O.Status*** ***O.System_Integrity***
2.	T.Data_Theft	***O.Access_Control*** ***O.Authen***
3.	T.Deny_Service	***O.Authen*** ***O.Access_Control*** ***O.Resource***
4.	T.Disclose_Data	***O.Access_Control*** ***O.Authen*** ***O.Flow_Control*** ***O.Resid_Prot*** ***O.Authen_Protect***

5.	T.Impersonate	***O.Assoc_User_Action*** ***O.Authen*** ***O.Authen_Address*** ***O.Integrity*** ***O.Replay*** ***O.Sequence***
6.	T.Insider	***O.Access_Control*** ***O.Assoc_User_Action*** ***O.Audit*** ***O.Authen*** ***O.Dynamic***
7.	T.Outsider	***O.Access_Control*** ***O.Authen*** ***O.Dynamic*** ***O.Flow_Control***
8.	T.Privacy_Violated	***O.Access_Control*** ***O.Anon*** ***O.Authen***
9.	T.Repudiate_Receive	***O.Access_Control*** ***O.Integrity*** ***O.Revoke_Cert***
10.	T.Repudiate_Send	***O.Access_Control*** ***O.Integrity***
11.	T.Spoofing	***O.Anon*** ***O.Access_Control***

12.	T.Social_Engineer	***O.Anon*** ***O.Integrity***

Table 2– Mapping Threats to Security Objectives

Mapping Objectives to CC Security Requirements

This section will identify CC [3] security functional requirements. The SFR is divided in to 11 classes which are as follows:

- Security audit (FAU)
- Communication (FCO)
- Cryptographic support (FCS)
- User data protection (FDP)
- Identification and authentication (FIA)
- Security management (FMT)
- Privacy (FPR)
- Protection of the TSF (FPT)
- Resource utilization (FRU)
- TOE access (FTA)
- Trusted path/channels (FTP)

Security Threat	**CC Functional Components**
O.Access_Control	FDP_ACC.1 FDP_ACF.1 FPT_SEP.1 FMT_SMR.2 FAU_SAR.2 FMT_MSA.3 FPT_RVM.1 FPR_UNO.1
O.Authen	FIA_UID.1 FIA_UAU.1 FAU_GEN.1 FIA_UAU.3 FIA_ATD.1 FMT_MSA.1 FTA_TAB.1 FMT_MTD.1 FTA_TAH.1 FTA_SSL.1 FTA_SSL.3 FIA_UAU.6 FTA_TSE.1 FIA_UAU.5 FCS_CKM.2

O.Authen_Protect	FPT_SEP.1 FCS_CKM.3 FIA_SOS.1
O.Assoc_User_Action	FAU_GEN.2 FIA_USB.1
O.Authen_Address	FCO_NRO.2 FCO_NRR.2 FDP_ACF.1 FCS_CKM.2
O.Audit	FAU_GEN.1 FAU_SEL.1 FMT_MOF.1 FAU_STG.2 FMT_MTD.1 FAU_STG.3 FAU_SAR.1 FAU_SAR.3
O.Anon	FPR_ANO.1 FPR_PSE.1
O.Dynamic	PBC_DYN.1
O.Flow_Control	FDP_IFC.1 FDP_IFF.1

O.Integrity	FDP_SDI.1 FDP_UIT.1 FDP_ITT.1 FPT_ITI.1 FPT_ITT.1
O.Recover	FPT_RCV.1
O.Resource	FRU_RSA.1
O.Resid_Prot	FDP_RIP.2
O.Replay	FIA_UAU.3 FPT_RPL.1 FCO_NRO.2 FCO_NRR.2
O.Revoke_Cert	FCS_CKM.4
O.Sequence	PBC_SYN
O.Status	FMT_MTD.1

Table 3 – Mapping Objectives Security Functional Components

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 process of risk analysis to prioritize security requirements. The steps for prioritizing security requirements are as follows:-

i. Prioritize each threat to the asset using any the risk measuring techniques such as CRAMM [28, 29], OCTAVE [26], or CORAS [27].

ii. Once the threats have been prioritized we have to back track threats to security requirements so that we can prioritize security requirements. The security requirement corresponding to highest priority threat will be the highest priority security requirement.

A brief explanation of the CRAMM process is given below.

CRAMM

CRAMM [28, 29] (CCTA Risk Analysis and Management Methodology) . The method for risk analysis used by CRAMM and most other methodologies consists of evaluating the following three factors:

a) The threats that can affect that asset,
b) The vulnerabilities that can be exploited by a threat,
c) The cost in case of impact on an asset.

From this determine a risk level or establish some measure of risk. This is conceptually illustrated in Figure 3 shown below.

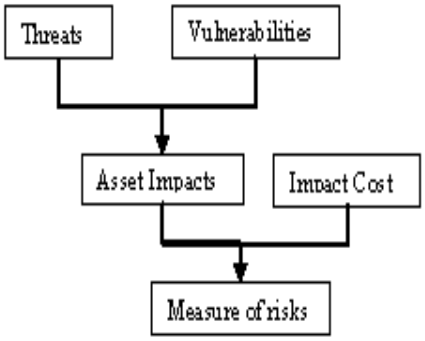

Figure 3- Risk analysis using CRAMM

The risk analysis itself consists of four activities:

(i) **Identify assets, threats** - Each potential asset that can have some impact on it must be identified. All possible threats, of all relevant vulnerabilities, and of all potentially affected assets are established.

(ii) **Identify potential asset impacts -** A list of all combinations of threat and vulnerabilities which potentially can cause an impact on an asset are identified.

(iii) **Value assets and measure threats and vulnerabilities -** Each potential asset that can be affected must be valued according to the cost of loss or damage of the asset into a scale from 1 to 10. The strength of the threats and the level of the vulnerabilities must be quantified. Possible values for threat are very low (.1), low (.34), medium (1), high (3.33) and very high (10). The levels of vulnerability are equated as low (.1), medium (.5) and high (1).

(iv) **Calculate the risk -** A fixed 3 dimensional lookup table (Table 4) shown below where the strength of the threat, the level of the vulnerability and the value of the asset are input parameters, gives the final security requirement (= 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 4– Three – Dimension lookup table to measure the level of risk

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 figure4.

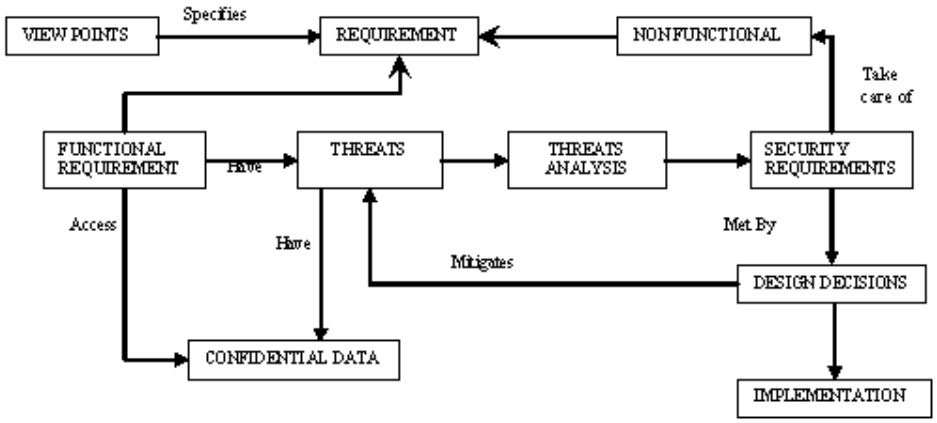

Figure 4– Model for Managing Security Requirements

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.

4. CASE STUDY : Online Book Store

In this section we go through the development stages of the VOSREP [9] process defined above using a case study of a CBS system "online book store" which is an online application for selling the books online to explain the VOSREP process. In this application the publisher can sell his books electronically while the customer can get the books on their seat. The customer can do the transactions online through electronic payment gateways.

4.1 Requirement Discovery and Definition

Step 1 – *Identify various stakeholders (actors)* - The direct stakeholders of the system will be **Employees of the publication house** and **Customer** who want to purchase the books.

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 Employees

- List book information (e.g., title, author, price) and quantity-in-stock of books.
- List information about those orders assigned to him/her.
- Update order status.
- Insert new books.

Functionalities of Customer

- New user account registration.
- User login.
- User can update his address, password etc.
- Book search (by author name, title, category, year or combinations).
- Place the order.
- Order trace.

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 Customer profile is as follows –

Stakeholder	Customer
Functionality	Account Registration
Type	Direct
Location	Remote
Private Exchange	True
Secret Exchange	False
Association	Write

Table 5 - Customer Profile for Account Registration

Now the threat to the stakeholders is evaluated based on their profile as follows

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

Table 6 - 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 for the system to mitigate those threats.

The detailed list of threats and security requirements after performing this step3 and step 4 together is shown below in table.

Viewpoints	Services	Non-functional Requirements	Threats	Security Requirements
Customer	1. New user account registration. 2. Book search. 3. Place the order.	1. Reliability. 2. Response time is not more than 30 sec. 3. Execution of the order is correct.	**T.Spoofing.** **T.Flooding.** **T.Disclose_Data** **T.Privacy_Violated** **T.Change_Data** **T.Repudiate_Receive**	1. Authorization Requirement.. 2. Privacy Requirements. 3.. Nonrepudiation Requirements
Publisher	1. Update data of the books. 2. Update Order Status. 3. List book information	1. Correctness. 2. Minimize response time. 3. Robustness. 4. Scalable.	**T.Change_Data** **T.Privacy_Violated** **T.Social_Engineer** **T.Outsider**	1. Integrity Requirements. 2. Authentication Requirements 3. Identification Requirements

Table 7– Example "Online Book Store" explaining VOSREP

4.2 Analysis and Prioritization of the requirements

We have limited our scope to twelve categories of threats as shown in table1 we analyze them for their completeness, consistency and group similar requirements. As shown (in table 8) below we have considered all the threats that different stakeholders can cause to the system (Completeness). Also all the threats that have been considered be implemented in some way. For ex – the threat T.Change_Data can be implemented by using any of the cryptographic technique or by using Hash Functions.

Now we will apply the risk analysis process CRAMM [28, 29] for prioritizing security requirements. First of all we will evaluate the risks of the various threats on the assets through CRAMM then based on the measure of risk we will backtrack and prioritize security

requirements. The CRAMM process of risk analysis is applied and the output of each phase is shown.

i. ***Identify assets and threats of the system in question***

Assets

- Customer Information.
- Book Information.
- Order Information.
- User Login Information.
- Credit Card Information.
- Communication Channels.

Threats

- T.Change_Data
- T.Repudiate_Receive
- T.Spoofing
- T.Flooding
- T.Disclose_Data
- T.Privacy_Violated
- T.Outsider
- T.Integrity
- T.Physical
-

ii. ***Identify Potential Asset Impact***

THREAT	ASSETS THAT CAN BE AFFECTED
T.Change_Data	Customer Information, Book Information, Order Information
T.Repudiate_Receive	Order Information
T.Spoofing	User Login Information

T.Flooding	Communication Channels
T.Disclose_Data	Customer Information, Book Information, Credit Card Information
T.Privacy_Violated	Book Information, Order Information
T.Outsider	Credit Card Information
T.Physical	Customer Information, Book Information, Order Information, Communication Channels

Table 8– Possible Vulnerable Assets

iii. ***Value Assets and Threats***

Asset	**Value(1 to 10)**
Customer Information.	**7**
Book Information.	**5**
Order Information.	**5**
User Login Information.	**4**
Credit Card Information.	**9**
Communication Channels	**6**

Table 9– Measure of Assets

Threat	Level Of Threat	Value(.1, .34, 1, 3.33, 10)
T.Change_Data	Medium	1
T.Repudiate_Receive	High	3.33
T.Spoofing	Medium	1
T.Flooding	Very Low	.1
T.Disclose_Data	Medium	1
T.Privacy_Violated	Medium	1
T.Outsider	High	3.33
T.Integrity	Very High	10
T.Physical	Very Low	.1

Table 10– Measure of various threats

iv. **Calculate the Risk –** Once we have calculated the value of threats and asset**s** we will use Table 4 to measure the value of risk. For Ex – Suppose asset is Customer Information (5) the threat is T.Change_Data (1) and Vulnerability being medium (.5) the measure of risk will be 4. Similarly consider the asset credit card information (9) the threat to this is T.Disclose_Data (1) 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 then we can prioritize the threats based on their measure of risk.

v. **Backtrack to Prioritize security Requirements –** Although the first four activities in 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 that is calculated. The more the value higher is the priority.

We have considered only two threats T.Change_Data and T.Disclose_Data with T.Disclose_Data having higher priority then T.Change_Data. The security requirement corresponding to T.Change_Data is Integrity requirements and with T.Disclose_Data is privacy requirements. The value of these security requirements will be based on the corresponding measure of risks to the threat like for Integrity requirements we will have a value of 4 while for privacy requirements it is 6.

Since one threats will have different measure of risk based on the assets so the corresponding security requirements will have many values we will calculate the average of all the values of the security requirements and then prioritize them according to their estimated values. Higher the value higher is the priority.

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 section 3.3 (step iv) w have considered 11 classes of security functional requirements according to CC. These 11 classes have been made keeping the dependency in mind for each class.

- 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.

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.

NetBeans IDE 5.5.1: **-** The standard distribution of NetBeans IDE(Integrated Development Environment) gives everything to develop Java SE applications, web applications, and Java EE enterprise applications. The main theme of NetBeans IDE 5.5 is support for the Java, which makes programming enterprise applications and web services much simpler. This can be downloaded from NetBeans site. It is open source software.

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.

Violet (Very Intuitive Object Layout Editing tool): **-** This is an open source program in java for making use case diagrams. violet is chosen because it is an open source and hence the code is available we can concentrate on implementing threats generation process in violet rather than starting making the use case diagram from scratch.

5.2 Files and Relations Used

The code consists of the following java files in addition to the two packages that are provided by 'Violet':

- UMLEditor.java
- GraphPanel.java
- actorprofile.java
- decidethreat.java
- pictablename.java
- writetofile.java
- thttoobj.java

UMLEditor.java - This is the main class that checks that weather the Java Runtime Environment (JRE) is installed in the system or not. If the JRE is there than it displays a interface to select and make Use Case Diagram

GraphPanel.java - This file displays the user interface for making use case diagram further more if the node to be clicked is the 'actor' one then it displays a window to complete the actor profile.

actorprofile.java – This file and helps the use case creator to complete the actor profile based on seven fields described earlier though a simple user interface so that the use case creator can complete the actor profile easily. After the actor profile is completed this file also saves the actor profile to the permanent storage for future references.

decidethreat.java – This file contains a function decidethreatcategory that decides list of threats associated with the actors according to the actor profile saved earlier in the database from a predefined list of 12 categories of threats.

pictablename.java – This file displays a dialog box that asks for a file and table name associated with a given project. The filename is needed because we have to store the actor name, threats associated and security objectives to the corresponding file.

writetofile.java – This file is responsible for documentation of actor names, threats associated and security objectives for each actor associated in a particular IT product under development.

thttoobj.java – This file has a function that maps each security threat to the corresponding security objectives. This file also has a function that maps the security objectives to the security functional requirements.

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

Actor – used to maintain the profiles of actors associated in the use case diagram.

Threatcategory – used to maintain the threats associated with the actors based on there profile.

Securityobjective – relation maps the threats to the corresponding security objective.

Functioalreq – This relation maps the security objective to corresponding functional requirements that fall under 11 classes.

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 violetdb. 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 *.java**.
3. And, to run the code, type **java UMLEditor**

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.

Or

We may run the code using NetBeans Software. If NetBeans IDE is used there is no need to set any of the environment variables as it automatically set all the environment variables. In both the cases, the same window will open containing a standard menu will be displayed. Choose the corresponding options from the menu and get done with your use case diagrams.

5.4 Functional Decomposition Diagram

The Figure shown below shows the functional decomposition diagram of the tool implemented.

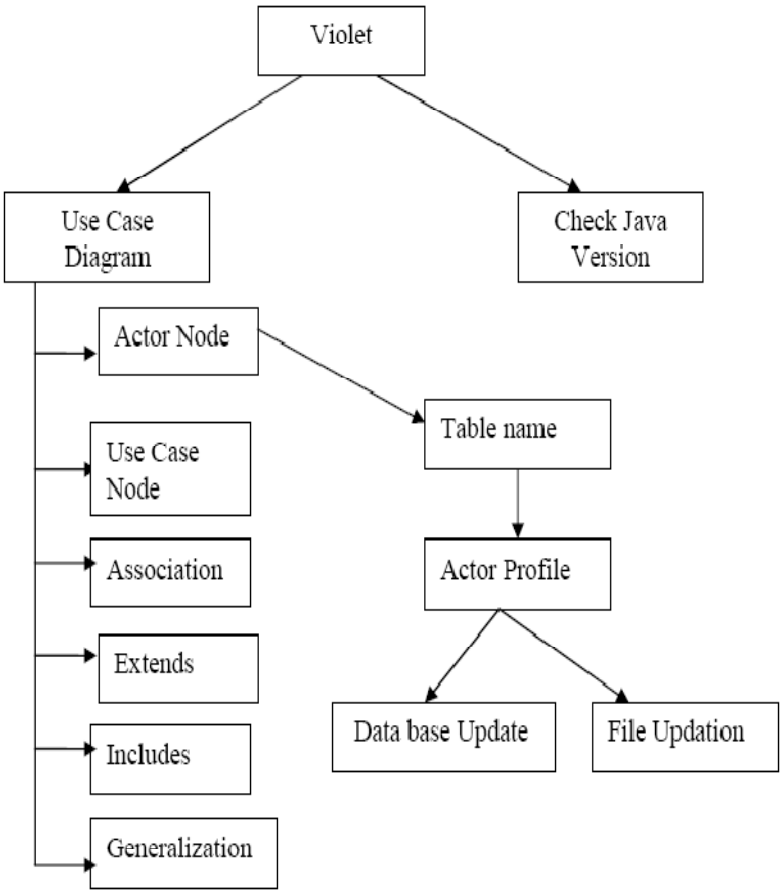

Figure 5 - Functional Decomposition Diagram

5.5 Snapshots

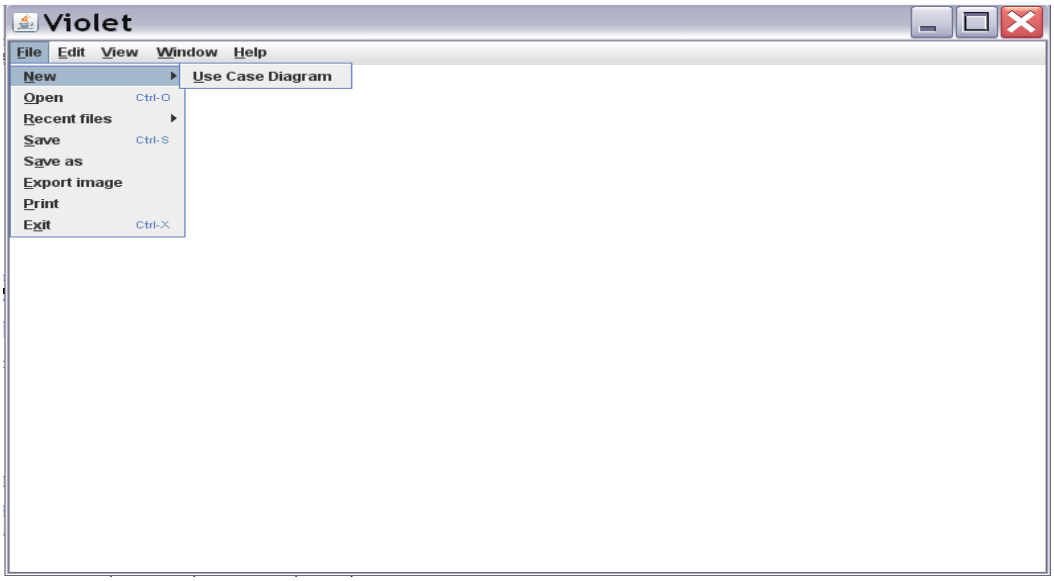

Figure 6- Main Window of Violet

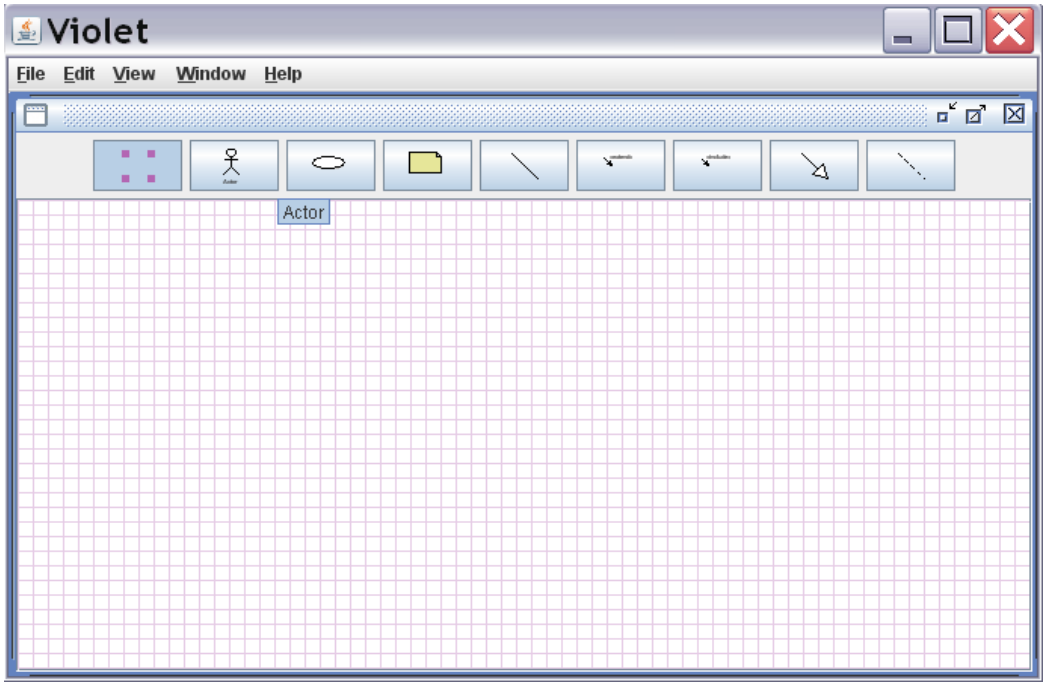

Figure 7- Main Window with Use Case Diagram Editing Frame

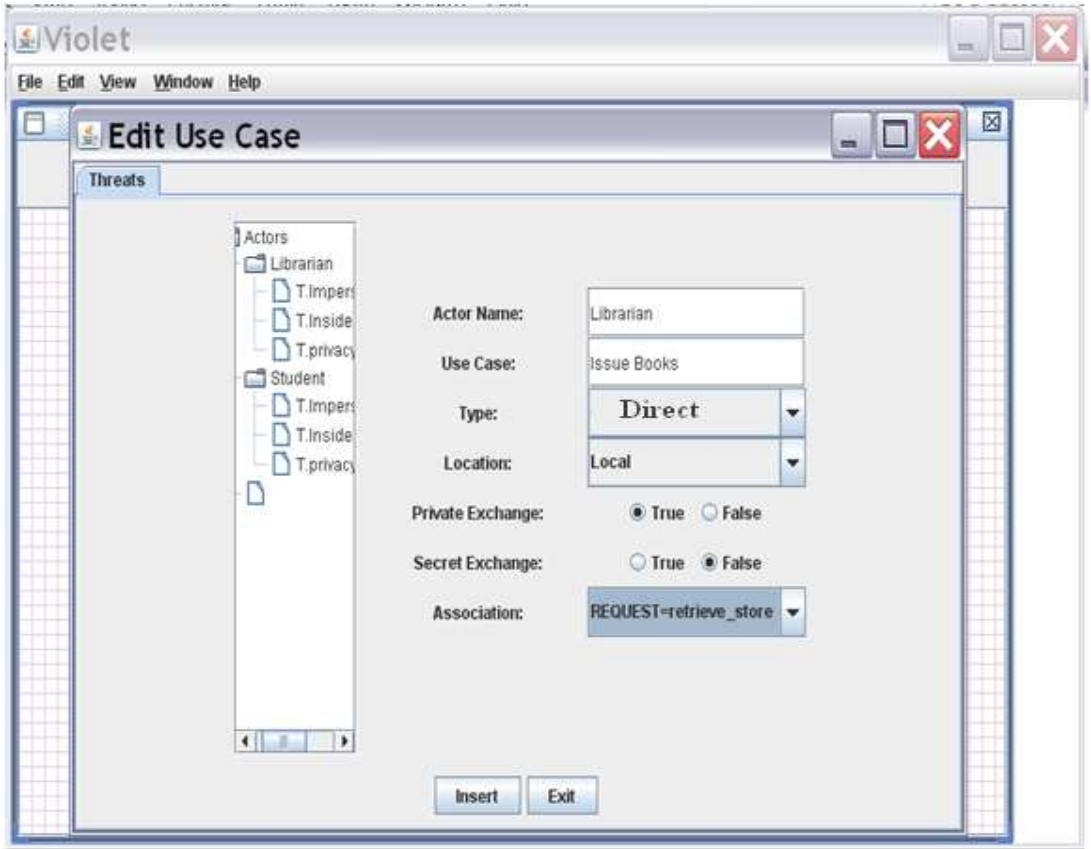

Figure 8 - Use Case Diagram Editing Frame Actor Profile Window

6. CONCLUSION

The work described in thesis provides you with a solid foundation for engineering security requirements. We have a developed a process that consist of well articulated steps for security requirement engineering. The focus of this work is not in the design and implementation phase rather the main aim of the work is to discover 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.

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 process that we have defined for the elicitation of security requirements which we called as VOSREP 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, prioritize and manage security requirements.

In the VOSREP process we also developed techniques to analyze, prioritize and mange various security requirements that we have elicited based on the threats to the functionality of the system under development. Prioritization of security requirements is very essential since this helps in the development of software systems that are less vulnerable and within budget. We have also devised a method to prioritize security requirements based on the measure of risk of the threat to the asset. To measure risk we have used an existing CRAMM method of risk analysis and with slight extensions to this process we have used it to prioritize security requirements.

The process that we have devised has well articulated steps for security requirement engineering. It also explains how the existing risk analysis techniques can be used to analyze various security requirements base on the measure of risk.

The tool that has been developed helps in the automatic generation of threats as well as security requirements of the system under development based on the predefined repository of the threats, security objectives and security requirements. We will generate actor's profiles that will automate the process. We made it mandatory to complete the actor profile for each stakeholder of the system.

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.

7. PUBLICATIONS

7.1 Accepted Papers

1. *Gupta D., Agarwal A., "Security Requirement Elicitation using view points for online system", International Conference on Emerging Trends in Engineering and Technology, Nagpur.*
2. *Gupta D., Agarwal A., "Guidelines and case study for eliciting Security Requirements", In the Proceedings of the 2nd National Conference on Computing for Nation Development,* Delhi , pages – 445 - 448 .

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: E-COFC 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, Vol2, 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., Agarwal A., "Security Requirement Elicitation using view points for online system", International Conference on Emerging Trends in Engineering and Technology, Nagpur, July 2008.
10. Gupta D., Agarwal A., "Guidelines and case study for eliciting Security Requirements", Proceedings of the 2nd National Conference on Computing for Nation Development, Delhi , pages - 445 – 448.
11. Gupta D. and Prakash N., “Engineering Methods from their Requirements Specification”, Requirements Engineering Journal 2006, 3, pages 133 – 160.
12. Ian Hawkins, “Risk Analysis Techniques”, 1998 Available at http://www.euclidresearch.com/current.html

13. J. Johnson, Chaos: "The Dollar Drain of IT project Failures". Application Development, pp.41-47, January 1995.
14. 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
15. Nancy R. Mead, "Requirement Elicitation Introduction", Software Engineering Institute, Carnegie Mellon University, 2006.
16. Prakash N., "On generic Method Models", Requirements Engineering Journal 2006, pages- 221 – 237.
17. Computer Emergency Response Team (CERT) Coordination Center, CERT/CC Statistics 1988-2005, Vulnerabilities report, from http://www.cert.org/stats/cert_stats.html.
18. Sindre G, Opdahl AL, "Eliciting security requirements with misuse cases". Requirements Eng. Journal, 2005, pages.34-44
19. Kotonya G., Sommerville I., "Requirement Engineering with view points", 1995.
20. Sommerville I., "Software Engineering". Seventh edition 2003. ISBN - 8129708671. Pearson Education.
21. The Standish group, Chaos. Standish Group Internal Report, 1995 http://www.standishgroup.com/chaos.html
22. M. Ware, J. Bowles, C. Eastman, "Using the common criteria to Elicit security Requirements with use cases", 2006, IEEE.
23. Mead, N. R., Hough, E., Stehney T., Security Quality Requirements Engineering (SQUARE) Methodology, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.
24. CORAS website at, http://www2.nr.no/coras/whatiscoras.html
25. 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
26. The Logic behind CRAMM's Assessment of Measures of Risk and Determination of Appropriate Countermeasures available at www.cramm.com
27. Insight Consulting- CRAMM Measures of Risk, Determination of Appropriate Countermeasures Document at www.insight.co.uk
28. Violet: Very Intuitive Object Layout Editing Tool, Retrieved from http://www.horstmann.com/violet