Designing
Security Requirement Solutions
Using
Back Tracking Approach

A Dissertation Submitted in partial fulfillment of the requirement
For the Award of degree of

Master of Technology
In
Software Engineering

By
Krishna Chandra Soni
10\MT\SE\FT
(University Roll No. 08\SE\09)

Under Esteemed Guidance Of
Respected Shalender Kumar Verma

Department Of Computer Science And Engineering
Delhi Technological University
2009-2011

Dedicated to my parents

D E C L A R A T I O N

I hereby declare that the report of the P.G Project Work entitled " Designing Security Requirement Solutions Using Back Tracking Approach" which is being submitted to the Delhi Technological University, Delhi in partial fulfillment of the requirements for the award of the Degree of Master of Technology in , Software Engineering , in the Department of Computer Engineering, is a bonafide report of the work carried out by me. The material contained in this report has not been submitted to any University or Institution for the award of any degree.

Place: Delhi Technological University (Name & Signature of the Student)

Date: ..... Department of Computer Engineering

CERTIFICATE

Department Of Computer Science And Engineering
Delhi Technological University
Bawana Road, Delhi – 110042

==========

This is to certify that project entitled "Designing Security Requirement Solutions Using Back Tracking Approach" has been completed by Mr. Krishna Chandra Soni , University Roll No. 08/SE/09 in the partial fulfillment of the requirement for the award of degree of Master of Technology in Software Engineering. This is a bonafied work carried out by him under my supervision and support. He has completed his work with utmost sincerity and diligence.

The work embodied in this major project has not been submitted for the award of any other degree to the best of my knowledge.

==========

Date…….

Mr. Shalender Kumar Verma
Lecturer & Project Guide
Dept. Of Computer Engineering
Delhi Technological University

Acknowledgement

It is a great pleasure of mine to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor Respected Mr. Shalender Kumar Verma, Lecturer in Computer Department for his invaluable guidance, encouragement and patient reviews. His continuous inspiration only has made me complete this dissertation. Without his help and guidance, this dissertation would have been impossible. He provided the conceptions and theoretical background for this study as well as suggested us the rational approach. He remained a pillar of help throughout the project.

With his continuous inspiration only, it becomes possible to complete this dissertation. I would also like to take this opportunity to present my sincere regards to my teachers Dr. Daya Gupta, Mrs. Ruchika Malhotra ,Mrs Akshi Kumar and the other staff of computer engineering department for providing me unconditional and any time access to the resources and guidance.

I am grateful to my parents for their moral support all the time; they have been always around to cheer me up, in the odd times of this work. I am also thankful to my classmates for their unconditional support and motivation during this work. Last but not least, I special thanks to the crowd who are active in the field of Software Engineering and Ambiguity issues.

ABSTRACT

Current practices for developing secure information systems are still closer to art than to an engineering discipline. Security is still treated as an add-on and is therefore not integrated into software development practices and tools. Experienced security artisans are still the key to achieving acceptable levels of security.

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). Many methods have been proposed for framing the security requirements, but the main target is how to find the solutions fulfilling these security requirements, to produce complete secure information system.

So, we propose a Model framework for finding a complete solutions of security requirements, which are identified during the security requirement elicitation stage, using back tracking analysis. After gathering security requirements, we do back tracking analysis of the approaches used to gather security requirements, to identify the solutions necessary to fulfill the gathered security requirements. On basis of back tracking analysis, we will find conceptual solutions and security services and corresponding mechanisms which encompass the complete security of software system.

Keywords: Information System, Integrity, Availability, Confidentiality, Security Requirement Elicitation, Backtracking, Conceptual Solutions, Security Services, Security Mechanisms

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1 Introduction

What makes it so easy for attackers to target software is the virtually guaranteed presence of vulnerabilities, which can be exploited to violate one or more of the software's security properties. According to CERT, most successful attacks result from targeting and exploiting known, non-patched software vulnerabilities and insecure software configurations, many of which are introduced during design and code. The security of software is threatened at various points throughout its life cycle, both by inadvertent and intentional choices and actions taken by "insiders"—individuals closely affiliated with the organization that is producing, deploying, operating, or maintaining the software, and thus trusted by that organization—and by "outsiders" who have no affiliation with the organization.. External faults that threaten the software's dependable operation are seen as a security issue when the faults result from malicious intent or the faults, regardless of their cause, make the software vulnerable to threats to its security. According to Bruce Schneier in *Beyond Fear* [37], "Security is about preventing adverse consequences from the intentional and unwarranted actions of others."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 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. Thus security-enhanced processes and practices—and the skilled people to manage them and perform them—are required to build software that can be trusted to operate more securely than software being used today.

1.1 Why Security?

Computer security [1] 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 [2] 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 [2, 3]. 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 [1]. This results in an over constrained, inefficient and high cost system.

Many researchers [1 ,4 , 5] 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 Security Engineering

A security engineering process is a complex activity involving many special work products such as security requirement elicitation, prioritization, security design, and implementation and testing. These work products are essential in a process that aims to create trustworthy information security products [6].

Security engineering entails using practices, processes, tools, and techniques to address security issues in every phase of the software development life cycle (SDLC). Software that is developed with security in mind is typically more resistant to both intentional attack and unintentional failures. One view of secure software is software that is engineered "so that it continues to function correctly under malicious attack" and is able to recognize, resist, tolerate, and recover from events that intentionally threaten its dependability. Broader views that can overlap with software security (for example, software safety, reliability, and fault tolerance) include the notion of proper functioning in the face of unintentional failures or accidents and inadvertent misuse and abuse, as well as reducing software defects and weaknesses to the greatest extent possible regardless of their cause.

The advantage of using security engineering process is to build better and defect-free systems. Software-intensive systems that are constructed using more securely developed software are better able to do the following:

- Continue operating correctly in the presence of most attacks by either resisting the exploitation of weaknesses in the system by attackers or tolerating the failures that result from such exploits
- Limit the damage resulting from any failures caused by attack-triggered faults that the system was unable to resist or tolerate and recover as quickly as possible from those failures

The objective is to increase the security and dependability of the system, produced by these practices, both during its development and during its operation. Thus, from the above facts we can say that security engineering is becoming essential component of systems engineering.

So, to design, build, and deploy secure systems, we must integrate security into your application development life cycle and adapt your current software engineering practices and methodologies to include specific security-related activities. Security-related activities include [7] identifying security requirements, prioritizing security requirements , applying security design, Implementing security mechanisms , security testing, and conducting security deployment reviews. The different activities of Security Engineering Process (SEP) can be categorized into following phases –

- **Requirement Engineering** – Discover security requirement along with functional and non functional requirements such as Privacy, Authentication, Integrity, Non-Repudiation requirements, elicit and prioritize them.
- **Design** – With true security requirements specified most appropriate design decisions can be taken. The different activities taken in security design includes identifying cryptography services & security design attributes , structuring them with threat and asset and finally taking design decision that specifies which security protocol is best suited for the identified Security Requirement.
- **Implementation** – This includes implementing specific algorithms that are suggested in the design phase of the Security Engineering Process.
- **Testing -** It involves evaluating the system security and determining the adequacy of security mechanisms, assurances and other properties to enforce system security policies .The primary reason for testing the security of an operational system , is to uncover design , implementation and operational flaws that could allow the violation of system security And to find unidentified potential vulnerabilities and subsequently repair them before delivering the final system to the user. The identified design decisions are validated against the security

requirements and the extent to which they satisfy a particular security requirement.

1.3 Motivation

In the process of development of any computer based system (CBS) the first and the most important step is gathering requirements. Requirement engineering [8, 9] 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.

Once the security requirements are elicited and prioritized, if proper solutions fulfilling security requirements are not identified, then it can lead to an underdeveloped system with unnecessary design constraints which makes the application vulnerable & exposed to attackers during its operation. Attacks may take advantage of publicly known but un patched vulnerabilities, leading to memory corruption, execution of arbitrary exploit scripts, remote code execution, and buffer overflows. Software flaws can be exploited to install spyware, adware, and other malware on users' systems that can lie dormant until it is triggered to execute [10].

The Design Phase of Security Engineering process has not received sufficient standardization and work in the recent past. Some recent work in methodology for security policy definition using the Zachman information systems architecture [11] has been proposed .In fact, it is widely recognized that most of the threats in real-world security infrastructure stems from how we perform cryptographic operations on secret data, although only a subset of security threats relating to privacy, authentication, integrity and non-repudiation services would be

mitigated through the use of cryptography protocols. As a consequence, the design details, which normally take on a marginal role and seemingly just affect performance, are of crucial importance, as they could open door to many real-world attacks in a number of nontrivial and often unforeseen ways. Hence we must have a well defined method or technique to find the all possible relevant solutions that fulfills the entire security requirement identified during requirement gathering phase.

In the proposed framework we have used a back tracking approach which can be applied to any security requirement eliciting process. We do back track analysis from gathered requirement up to assets identified and environment constraints , and at each stage we do elimination of identified solutions, hence at last we are left with exact and accurate solutions which when implemented covers all security requirements.

In the proposed framework, we are having two types of solutions: conceptual solutions and security services solution. Conceptual solutions are like case shell over all security requirements and security services delivers security over information.

Therefore we aim to develop a well defined Framework that will have well articulated steps for Security Design Engineering. Moreover this process should be coherent with the conventional Software Engineering process so that eliciting security requirements & security design become an integral part of system engineering and security engineering.

1.4 Proposed Work

In this thesis work, we propose a Framework for finding solutions of gathered security requirements highlighting the design phase, that involves modeling of Security requirements & threats, which are identified during the security requirement elicitation stage.

Our proposed basically consist of four layers:

- **Gather Information Related to Use Case**: At this layer we will collect all the information for particular use case along with the associated assets and threats, identified during requirement elicitation phase. We may also gather information about common criteria related to use case, to refine our solution sets.
- **Identification of Environmental Constraints and System Attributes:** At this layer we will identify all the environmental constraints related to use case along with the system attributed which are required to implement the particular use case. These constraints and attributes will act as a filter in refining our solution sets and thereby more precise solution sets do we will get.
- **Refinement of Conceptual Solutions for Security Requirement Fulfillment**: These are some sets of solutions proposed by us, representing the concepts which when implemented, will satisfy the corresponding security requirement. In this, we have done the mapping of concepts to the security requirements. These concepts covered all the 12 security requirements proposed by Firesmith[1]. We have defined the limited amount of conceptual solutions, but there may exist many more concepts.
- **Deriving Security Services and Mechanism:** Based on the environmental constraints and system attributes, we will derive the most appropriate security services available and the mechanisms to implement them.

Hence at the end of all four steps we will have complete set of solutions, fulfilling the gathered security requirements. The advantage of using this approach for security engineering helps in the identification of true security requirements & design guidelines. 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

The aim of this dissertation is to provide the framework that will design solutions covering all identified security 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.

Chapter 2: gives the overview of literature survey on security requirements emphasizing on elicitation techniques, requirement engineering etc.

Chapter 3: explains the terminology conceptual solutions and security services and mechanisms

Chapter 4: explains in detail the proposed methods along with the terminologies introduced by us, thereby focusing on various layers embedded in proposed framework.

Chapter 5: explains the case study by taking a case study of ‘Online Purchasing System’.

Chapter 6: presents the conclusion and future scope.

References.

Chapter 2 Security Requirement Engineering

It comes as no surprise that requirements engineering is critical to the success of any major development project. Some studies have shown that requirements engineering defects cost 10 to 200 times as much to correct once fielded than if they were detected during requirements development [12, 13]. Other studies have shown that reworking requirements, design, and code defects on most software development projects costs 40 to 50 percent of total project effort , and the percentage of defects originating during requirements engineering is estimated at more than 50 percent. The total percentage of project budget due to requirements defects is 25 to 40 percent.

"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 kwon as shall not requirements but are not risks or threats".

Following are the points to be noted regarding security requirements:

- 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.

2.1 Assets and Threats

Assets are the reason threats exist; an adversary's goal is to gain access to an asset. The security team needs to identify which assets need to be protected from an unauthorized user. Assets can be either physical or abstract, i.e. employee safety, company's reputation etc. Assets can interact with other assets and, because of this, they can act as a pass-through point for an adversary.

2.1.1 Assets Identification and Prioritization

Assets are also identified along with their associated risks. We followed the procedure explained in [15] to identify and prioritize assets. As a first step, a brainstorming session is conducted and all the valuable assets are listed. Next step is to examine various existing documents for other important assets. Once all the assets are listed, the assets are categorized and prioritized with respect to security. To perform this, an asset is taken and viewed from different perspectives i.e. customer, administrator and attacker. From each perspective, each asset gets assigned a number indicating the importance of confidentiality, integrity or availability for this asset. All the priorities of each asset are added and the asset with lowest sum is ranked as highest priority asset.

2.1.2 Threats Identification

The second step, determining threats, is certainly the most challenging aspect of threat modeling. After the previous steps have been completed, it is time to think about the specific threats to the system. Threats may come from either inside or outside the system—from authorized users or from unauthorized users who masquerade as valid

users or find ways to bypass security mechanisms. Threats can also come from human errors. The goal of this step is to identify threats to the system using the information gathered so far. A threat is the adversary's goal, or what an adversary might try to do to a system [27]. Sometimes a threat is also described as the capability of an adversary to attack a system. The best method for threat enumeration is to step through each of the system's assets, reviewing a list of attack goals for each asset. Assets and threats are closely correlated. A threat cannot exist without a target asset. Threats are typically prevented by applying some sort of protection to assets. The process of correlating threats to an asset involves creation of adversary hypotheses.

2.1.3 Classification of all threats

The output of threat identification process is a threat profile for a system, describing all the potential attacks, each of which needs to be mitigated or accepted. In general, threats can be classified into six classes based on their effect [27]:

- Spoofing refers to usage of someone else's credentials to gain access to otherwise inaccessible assets. All the attacks in which someone uses someone else identity in the system come under this category.
- Tampering refers to concept of altering data to mount an attack. All the attacks in which someone changes some information without permission fall into this category.
- Repudiation occurs when a user denies performing an action, but the target of the action has no way to prove otherwise. All the attacks in which someone denies a transaction that was performed are mapped into this category. For example, someone denying a purchase order after receiving the merchandise and denying the payment is classified as repudiation.

- Information disclosure refers to disclosure of information to a user who does not have permission to see it. All the attacks in which someone gets to see information she has no right to access can be termed as information disclosure.
- Denial of service- Reducing the ability of valid users to access resources. All the attacks in which someone breaks the system and prevent it from working normally and supplying the service it should fall into this category. The fact that the system does not work can serve for the interest of the attacker (or the one who sent him).There a numerous ways to implement such an attack.
- Elevation of privilege occurs when an unprivileged user gains privileged status. All the attacks in which someone enhances their capabilities by raising their privileges fall into this category. Example is when the attacker manages to get administrative rights.

When identifying a threat, it is helpful to think of various attacks in terms of the above classification. On the other hand, security threats are breaches of confidentiality, integrity, or availability. Thus, threats could also be classified by these properties. This classification is useful in security requirements when deciding on a mitigation mechanism of a specific threat. For example, unauthorized modification of data en route to component B from component A poses a tampering threat which violates the integrity property. To mitigate this threat, it might make sense to apply integrity mechanism such as Secure Hashing Algorithm-1 (SHA-1) on the data being transferred.

2.2 Types of Security Requirements

Different types of security requirements as proposed by Firesmith [1] 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 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.

2.2.12 Physical protection requirements

A physical protection requirement is any security requirement that specifies the extent to which an application or center shall protect itself from physical assault. The typical objectives of physical protection requirements are to ensure that an application or center are protected

against the physical damage, destruction, theft, or replacement of hardware, software, or personnel components due to vandalism, sabotage, or terrorism. Examples

- The data center shall protect its hardware components from physical damage, destruction, theft, or surreptitious replacement.
- The data center shall protect its personnel from death, injury, and kidnapping.

2.3 Security Requirement Elicitation Methods

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 [28], abuse case [29], misuse case [30, 31], security use case [32] 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. The following list identifies several methods that could be considered for eliciting security requirements. Some have been developed specifically with security in mind (e.g., misuse cases), whereas others have been used for traditional requirements engineering and could potentially be extended to security requirements.

2.3.1 Attack Trees

Attacks trees [28] 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.

2.3.2 Abuse Cases

Abuse case [29] 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.

2.3.3 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 [30, 31] 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.

2.3.4 Security Use Cases

This approach by Firesmith [32] 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.

2.3.5 Common Criteria (CC) with use cases

This approach [33] 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.

This approach 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.

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 25. Now these threats are used to find out the security requirements.

2.3.6 Viewpoint Oriented Security Requirement Elicitation (VOSREP)

Here we would be describing the View point oriented method of eliciting security requirements given by Dr. Daya Gupta [4]. 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. In the VOSREP Process we give the techniques to elicit, analyze and manage security requirements. The process VOSREP is based on following observation:

- 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 [34].
- In this VOSREP process 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.

2.4 Security Requirement Prioritization

Once you have identified a set of security requirements, you will usually want to prioritize them. Due to time and budget constraints, it can be difficult to implement all requirements that have been elicited for a system. Also, security requirements are often implemented in stages, and prioritization can help to determine which ones should be implemented first. Many organizations pick the lowest cost requirements to implement first, without regard to

importance. Others pick the requirements that are easiest to implement, for example by purchasing a COTS solution. These ad hoc approaches are not likely to achieve the security goals of the organization or the project. A number of prioritization methods have been found to be useful in traditional requirements engineering and could potentially be used for security requirements. Few of them are discussed below:

2.4.1 Binary Search Tree (BST)

Binary Search Tree is an algorithm that is typically used in a search for information and can easily be scaled to be used in prioritizing many requirements [16]. The basic approach for requirements is as follows, quoting from [16]:

1. Put all requirements in one pile.
2. Take one requirement and put it as root node.
3. Take another requirement and compare it to the root node.
4. If the requirement is less important than the root node, compare it to the left child node. If the requirement is more important than the root node, compare it to the right child node. If the node does not have any appropriate child nodes, insert the new requirement as the new child node to the right or left, depending on whether the requirement is more or less important.
5. Repeat steps 3-4 until all requirements have been compared and inserted into the BST.
6. For presentation purposes, traverse through the entire BST in order and put the requirements in a list, with the least important requirement at the end of the list and the most important requirement at the start of the list.

2.4.2 Numeral Assignment Technique

The Numeral Assignment Technique provides a scale for each requirement. Brackett proposed dividing the requirements into three groups: mandatory, desirable, and unessential [17]. Participants assign each requirement a number on a scale of 1 to 5 to indicate its importance [18]. The numbers carry the following meaning:

1. does not matter (the customer does not need it)
2. not important (the customer would accept its absence)
3. rather important (the customer would appreciate it)
4. very important (the customer does not want to be without it)
5. mandatory (the customer cannot do without it)

The final ranking is the average of all participants' rankings for each requirement.

2.4.3 Planning Game

The planning game is a feature of extreme programming [19] and is used with customers to prioritize features based on stories. This is a variation of the Numeral Assignment Technique, where the customer distributes the requirements into three groups, “those without which the system will not function,” “those that are less essential but provide significant business value,” and “those that would be nice to have.”

2.4.4 100-Point Method

The 100-Point Method [20] is basically a voting scheme of the type that is used in brainstorming exercises. Each stakeholder is given 100 points that he or she can use for voting in favor of the most important requirements. The 100 points can be distributed in any way that the stakeholder desires. For example, if there are four requirements that the stakeholder views as equal priority, he or she can put 25 points on each. If there is one

requirement that the stakeholder views as having overarching importance, he or she can put 100 points on that requirement. However, this type of scheme only works for an initial vote. If a second vote is taken, people are likely to redistribute their votes to get their favorites moved up in the priority scheme.

2.4.5 Theory-W

Theory-W was initially developed at the University of Southern California in 1989 [21]. It is also known as "Win-Win." An important point is that it supports negotiation to solve disagreements about requirements, so that each stakeholder has a "win." It has two principles:

1. Plan the flight and fly the plan.
2. Identify and manage your risks.

The first principle seeks to build well-structured plans that meet predefined standards for easy development, classification, and query. “Fly the plan” ensures that the progress follows the original plan. The second principle, “Identify and manage your risks,” involves risk assessment and risk handling. It is used to guard the stakeholders’ “win-win” conditions from infringement. In win-win negotiations, each user should rank the requirements privately before negotiations start. In the individual ranking process, the user considers whether there are requirements that he or she is willing to give up on, so that individual winning and losing conditions are fully understood. Theory-W has four steps:

1. Separate the people from the problem.
2. Focus on interests, not positions.
3. Invest options for mutual gain.
4. Insist on using objective criteria.

2.4.6 Requirements Triage

Requirements Triage [22] is a multistep process that includes establishing relative priorities for requirements, estimating resources necessary to satisfy each requirement, and selecting a subset of requirements to optimize probability of the product's success in the intended market. This is clearly aimed at developers of software products in the commercial marketplace. Davis's more recent book [23] expands on the synergy between software development and marketing; we recommend that you read it if you are considering this approach. It is a unique approach that is worth reviewing, although it clearly goes beyond traditional requirements prioritization, considering business factors as well.

2.4.7 Wiegers' Method

This method relates directly to the value of each requirement to a customer [24]. The priority is calculated by dividing the value of a requirement by the sum of the costs and technical risks associated with its implementation [24]. The value of a requirement is viewed as depending on both the value provided by the client to the customer and the penalty that occurs if the requirement is missing. This means that developers should evaluate the cost of the requirement and its implementation risks, as well as the penalty incurred if the requirement is missing. Attributes are evaluated on a scale of 1 to 9.

2.4.8 Requirements Prioritization Framework

The requirements prioritization framework and its associated tool [25, 26] includes both elicitation and prioritization activities. This framework is intended to address the following:

- elicitation of stakeholders' business goals for the project
- rating the stakeholders using stakeholder profile models

- allowing the stakeholders to rate the importance of the requirements and the business goals using a fuzzy graphic rating scale
- rating the requirements based on objective measure
- finding the dependencies between the requirements and clustering requirements so as to prioritize them more effectively
- using risk analysis techniques to detect cliques among the stakeholders, deviations among the stakeholders for the subjective ratings, and the association between the stakeholders' inputs and the final ratings.

2.5 Conclusion

In this section, we have discussed the various techniques regarding assets identification, threats identification and prioritization, then we discussed security requirement elicitation and prioritization techniques. These all forms the foundation of our framework, and provides lot of knowledge to understand the security importance and issues related to development of information system.

Chapter 3 Conceptual Solutions & Security Services

Systems are often developed without security in mind. Often we ignore security because either security policies are not available or it seems easier to postpone the security issues. Ignoring the security issues is dangerous because it can be difficult to retrofit security in an application. While an application's design could initially be more complicated by incorporating security from the start, the design will be cleaner than the result of integrating security late in the development cycle. This omission of security concerns is primarily because the application programmer is focusing more on trying to learn the domain rather than worrying about how to protect the system. The developer is building prototypes and learning what is needed to satisfy the needs of the users. In these cases, security is usually the last thing he or she needs or wants to worry about. When the time arrives to deploy these systems, it quickly becomes apparent that adding security is much harder than just adding a password protected login screen.

Firesmith [1] stated twelve different kinds of security requirements, which when implemented in correct manner, provide a complete secure system. It is generally observed that we generally focuses on available security services like integrity, confidentiality, availability, authenticity, non repudiation. But if we analyze deeply these security services then we will observe that these services are not sufficient to fulfill all the security requirements stated by Firesmith [1].

Hence we introduce the new terminology called Conceptual Solutions, which indicate the concepts, which when applied to the information system, will fulfill all the security requirements stated by Firesmith [1].Thus along with the security services , if we apply conceptual solutions in the system , we can guarantee the complete security.

3.1 Conceptual Solutions

'Conceptual Solutions' is the term introduced by us, which refers to set of concepts which when implemented, ensures the security of the information system. The idea to introduce this term came from Joseph [35] work. Initially we have found some collection , mentioned below, which covers all the 12 requirements stated by Firesmith [1] which we have mapped to security requirements stated by Firesmith [1] thus covering security in all perspectives . These are:

- Access Controls
- Access Points
- Behavioral Report
- Checkpoints
- Identifiers
- Log reports
- Privileges
- Roles
- Restore Points
- Third Party Support
- Trapdoors
- Views

Access Control :

Access control is a concept which enables an authority to control access to areas and resources in a given physical facility or computer-based information system. Access control is, in reality, an everyday phenomenon. A lock on a car door is essentially a form of access control. A PIN on an ATM system at a bank is another means of access control. The

possession of access control is of prime importance when persons seek to secure important, confidential, or sensitive information and equipment.

Access Points:

This concept provides a security module and a way to log into the system. With access point concepts , control flow is simpler since everything must go through a access points of responsibility in order for access to be allowed. The typical solution is to create a login screen for collecting basic information about the user, such as username, password, and possibly some configuration settings.

Behavioral Reports:

This concept identifies common communication pattern between objects or user with system and realize these patterns. Also we can keep track of behavior of user with system so that we may keep auditing user interaction with system and can verify authenticity of user. By doing so, these concepts increase flexibility in carrying out communication.

Check Points:

These are the set of concepts that encapsulates set of rules, policies etc. which verifies the criteria's to prove authenticity of user or system. These sets of policies or rules may also be used to verify the criteria's to prove identity of the system.

Identifiers:

This concept refers to the set of assets which is hold by user or interacting objects that acts as identity of that object or user.

Log Reports :

This concept refers to maintaining a file that lists actions that have occurred. For example, Web servers maintain log files listing every request made to the server. With log file analysis tools, it's possible to get a good idea of where visitors are coming from, how often they return, and how they navigate through a site.

Privileges:

This concept refers to set of advantage, immunity, or right held as a prerogative of status or rank, and exercised to the exclusion or detriment of others. Thus it ensures authorizations.

Roles:

This concept refers to the actions and activities assigned to or required or expected of a person or group. Thus on basis of roles we may assign privileges to different users.

Restore Points:

A restore point is a saved "snapshot" of a computer's data at a specific time. By creating a restore point, you can save the state of the operating system and your own data so that if future changes cause a problem, you can restore the system and your data to the way it was before the changes were made. When a restore point is established, your computer creates a backup copy of all data at that particular time.

Third Party Support:

This concept refers to **Third party** is often used to refer to a person or entity who is not one of two involved in some relationship, but may provide some sort of functionality or support indirectly to perform an activity between directly communicating parties.

Trapdoors:

This concept refers to an entrance or exit point in an information processing system which circumvents the normal security measures. It is generally a hidden program or an electronic component which makes the protection system ineffective if certain not documented orders are placed to him. Moreover, the trap door is often activated by an event or a normal action.

A trap door can also be a hole of security in a system which was deliberately set up by the creators or the people in charge of maintenance. The principal interest of these trap doors is not always harmful: certain operating systems, for example, have accounts users with high privileges intended to facilitate the work of the maintenance men. But in this case, they must be documented.

Views:

This concept refers to the permission that lets a user see the metadata of the securable on which the permission is granted. With the help of this concept , we can manage the information and data privacy, thus delivering security.

3.2 Security Services and Mechanisms

Information security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. The rapid growth and widespread use of electronic data processing and electronic business conducted through the Internet, along with numerous occurrences of international terrorism, fueled the need for better methods of protecting the computers and the information they store, process and transmit.

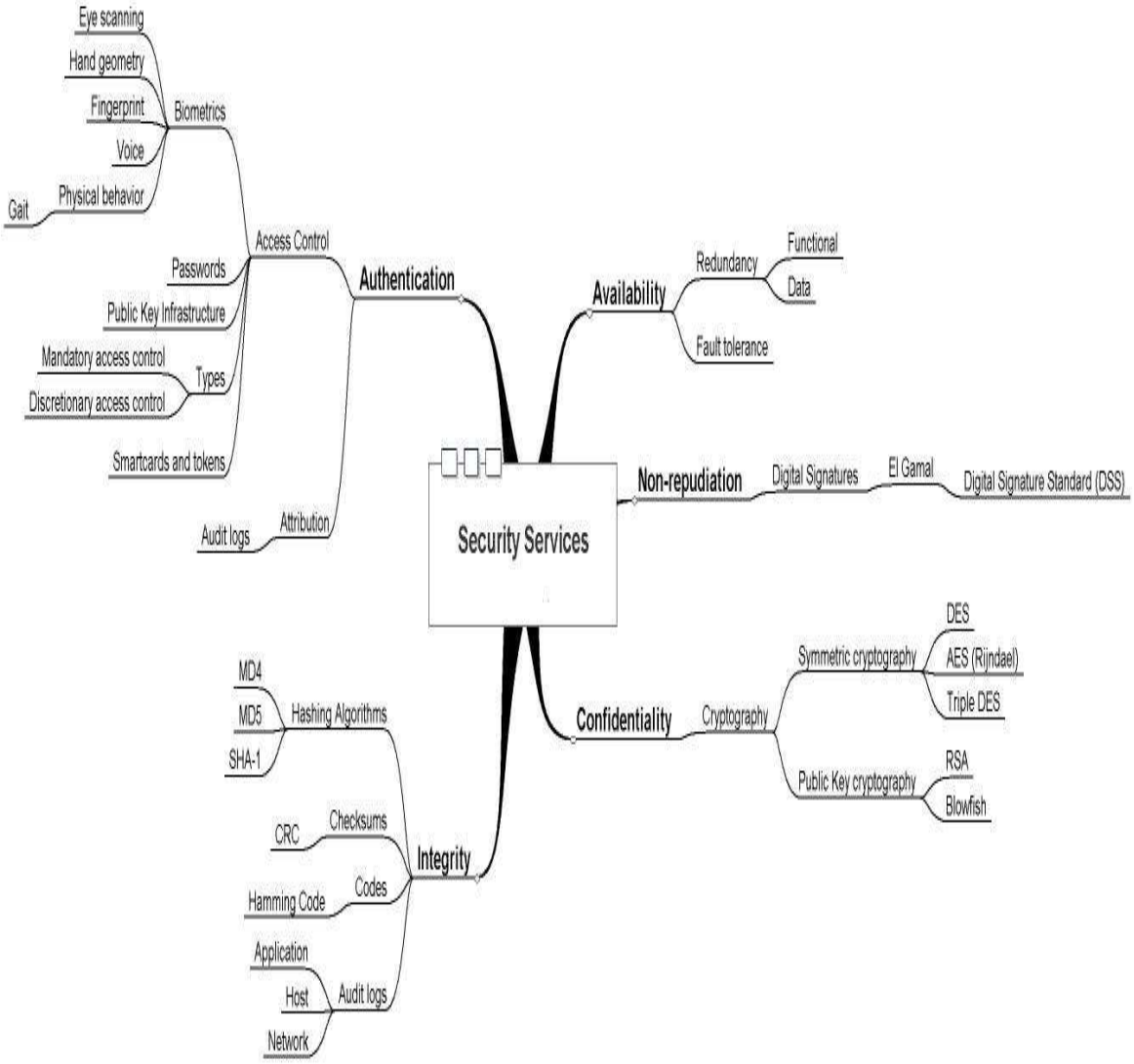

Figure 1: Security Services and Security Mechanisms, Usher[36]

Security is an attribute of system that prevents the system from revealing, changing and denying of resource services and system information in an illegal way. Generally three aspects of security are: confidentiality, integrity and availability of service of resources and information. To achieve these aspects and develop a secure system, security services and mechanisms should be considered. Below we have mentioned some security services that ensure security to information.

- **Confidentiality**

  Confidentiality is the term used to prevent the disclosure of information to unauthorized individuals or systems. For example, a credit card transaction on the Internet requires the credit card number to be transmitted from the buyer to the merchant and from the merchant to a transaction processing network. The system attempts to enforce confidentiality by encrypting the card number during transmission, by limiting the places where it might appear (in databases, log files, backups, printed receipts, and so on), and by restricting access to the places where it is stored. If an unauthorized party obtains the card number in any way, a breach of confidentiality has occurred. Confidentiality is necessary (but not sufficient) for maintaining the privacy of the people whose personal information a system holds.

- **Integrity**

  In information security, integrity means that data cannot be modified undetectably. Integrity is violated when a message is actively modified in transit. Information security systems typically provide message integrity in addition to data confidentiality.

- **Availability**

  For any information system to serve its purpose, the information must be available when it is needed. This means that the computing systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it must be functioning correctly. High availability systems aim to remain available at all times, preventing service disruptions due to power

outages, hardware failures, and system upgrades. Ensuring availability also involves preventing denial-of-service attacks.

- **Authenticity**

  In computing, e-Business and information security it is necessary to ensure that the data, transactions, communications or documents (electronic or physical) are genuine. It is also important for authenticity to validate that both parties involved are who they claim they are.

- **Non-repudiation**

  In law, non-repudiation implies one's intention to fulfill their obligations to a contract. It also implies that one party of a transaction cannot deny having received a transaction nor can the other party deny having sent a transaction. Electronic uses technology such as digital signatures and encryption to establish authenticity and non-repudiation.

3.3 Conclusion

In this chapter we describe various terminologies that acts as foundation for understanding the proposed framework. We also introduces the security services which are related to information security , which are not sufficient to fulfill all security requirements stated by Firesmith [1].Hence we introduced the term Conceptual Solutions , covering various security concepts.

Chapter 4 Back Tracking Framework

In the previous chapters, we have analyzed the approaches used to elicit security requirements along with the terms and concepts. Lots of methods are available to elicit security requirements, but less focus has been paid in designing solutions fulfilling these security requirements. The Design Phase of the SDLC represents a critical time for identifying and preventing security flaws before they become part of the software. . During this phase in the software development effort, architects, designers, and security analysts have an opportunity to ensure that requirements are interpreted appropriately through a security lens and that appropriate security knowledge is leveraged to give the software structure and form in a way that minimizes security risk.

4.1 Problem Statement

Firesmith [1], stated 12 different kinds of security requirements, which when implemented with proper solutions ensures completely secure software. Lots of effort has been done to ensure security in system, but these efforts mainly contain information security only, which covers some requirements stated by Firesmith. Requirements like Survivability requirement, Physical Protection requirement, Security Auditing requirements, Intruder Detection requirements, System Maintenance requirement, Immunity Requirement are often not considered as we consider solutions keeping existing security services in mind. Hence these requirements are postponed for later stage of development and after designing the complete information system , we find that it is hard or difficult to embed the solutions of these left out requirements and may cross budget and timeline if implemented.

4.2 Back Tracking Framework

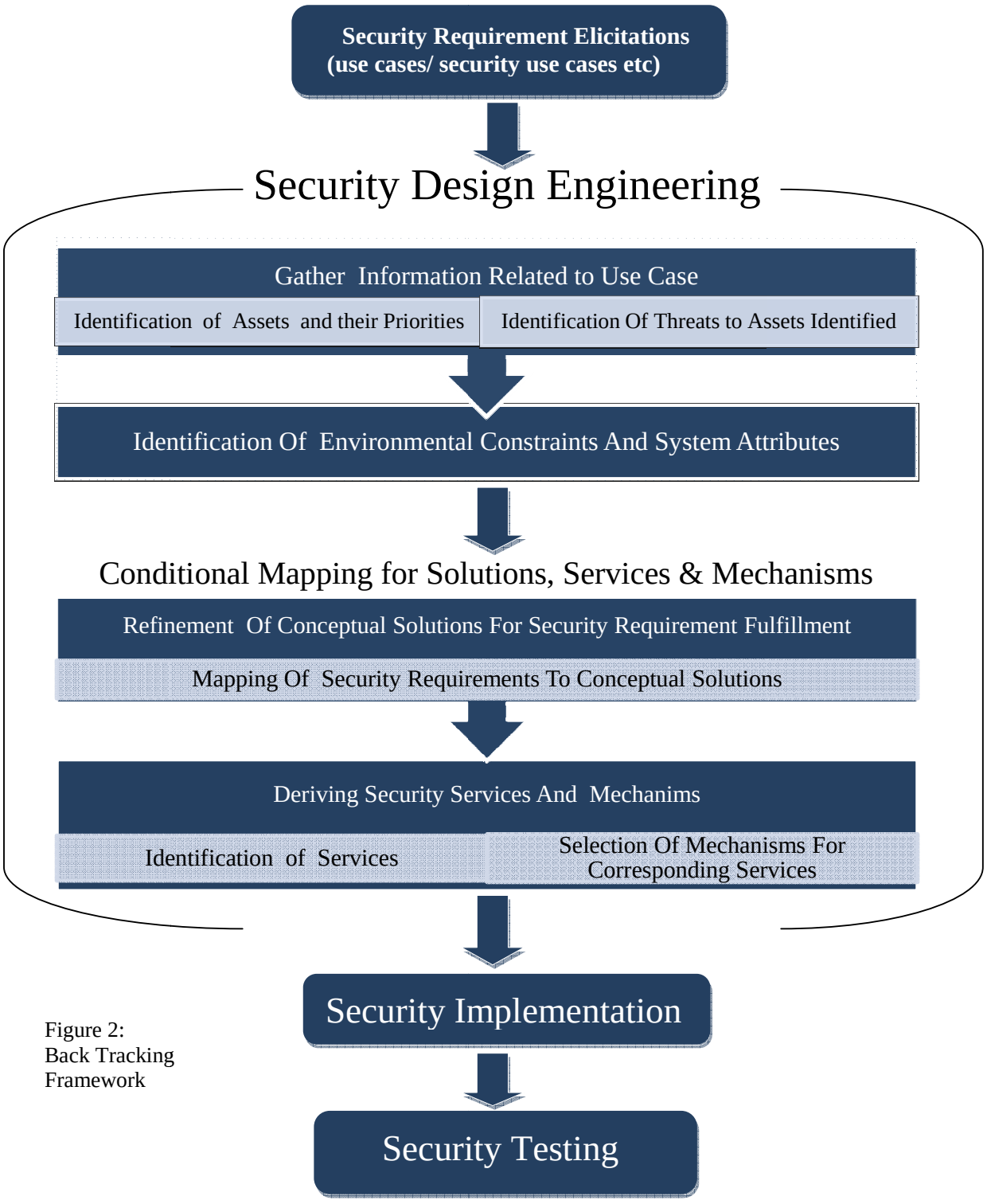

Figure 2:
Back Tracking
Framework

The framework consists of four different layers, and at each layer we are performing activities as follows. We will display the activities happening at each layer with the help of tool designed for this purpose.

1. Gather Information Related to Use Case:

At this layer, we are collecting different information gathered during requirement elicitation phase. This layer basically consists of two steps:

- Identification of assets and their priorities.
- Identification of threats related to assets.

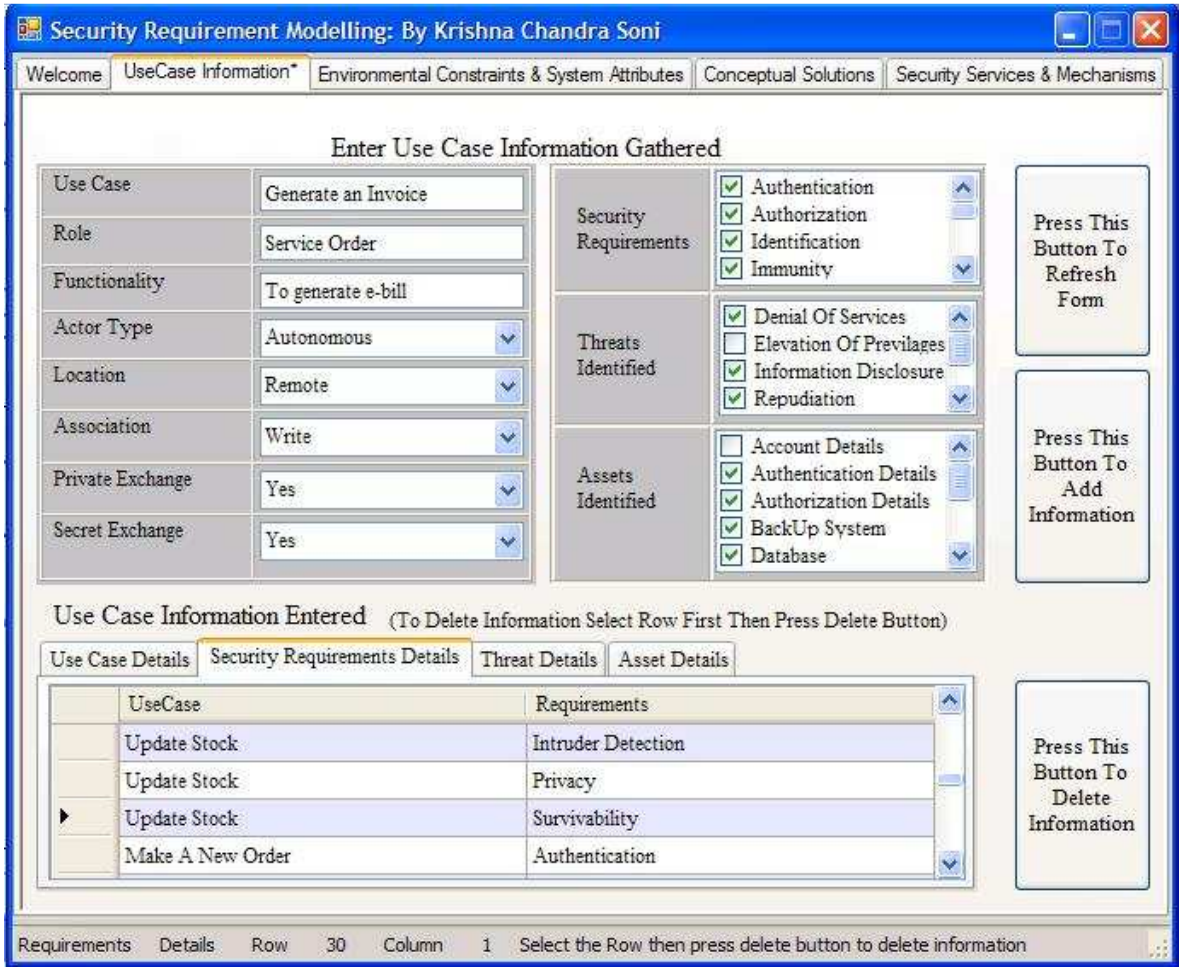

Figure 3: Gather Information Related to Use Case

At this stage , our aim is to collect all requirements and retails related to particular use case. In the tool designed, we are collect following requirements:

Common Criteria's:

- Actor Name: E.g. Specialist Doctor , Paramedics
- Use Case : E.g. Add Patients, Access Patient Reports
- Type: 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.

Apart from this we are collecting the security requirements identified during requirement gathering phase, which includes 12 different kinds of requirements:

- Authentication Requirement
- Authorization Requirement
- Identification Requirement
- Immunity Requirement
- Integrity Requirement
- Intruder Detection Requirement
- Non Repudiation Requirement
- Physical Protection Requirement
- Privacy Requirement
- Security Auditing Requirement
- Survivability Requirement
- System Maintenance Requirement

After requirements, we have gathered threats. Rather than gathering threats in detail we have just gathered the classes to which they belong [27]. In general, threats can be classified into six classes based on their effect [27]:

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

Then we will identify the assets involved in that use case like login information, web server etc.We can also do prioritization and can gather further details for more refinement. Larger the details better will be the refinement of solutions and hence more compact and effective solutions we will have at last, fulfilling all security requirements gathered.

2. Identification Of Environmental Constraints And System Attributes:

For a particular use case , before implementation , it is required that we should keep in mind all the environmental constraints where the use case will be implemented and the system attributes over which we will implement use case. Similarly , the security of the use case also depends on the environmental constraints and system attributed. Hence while designing security, one should keep in mind all the gathered constraints and attributes.

Environmental constraints may include attributes like:

- Coverage Area, whether LAN, WAN or Internet.
- Communication Channel like public channel or private channel
- Interface type via which user will interact with use case like hardware or software.
- Information state like storing, transmission or processing.
- User Type whether human or autonomous.

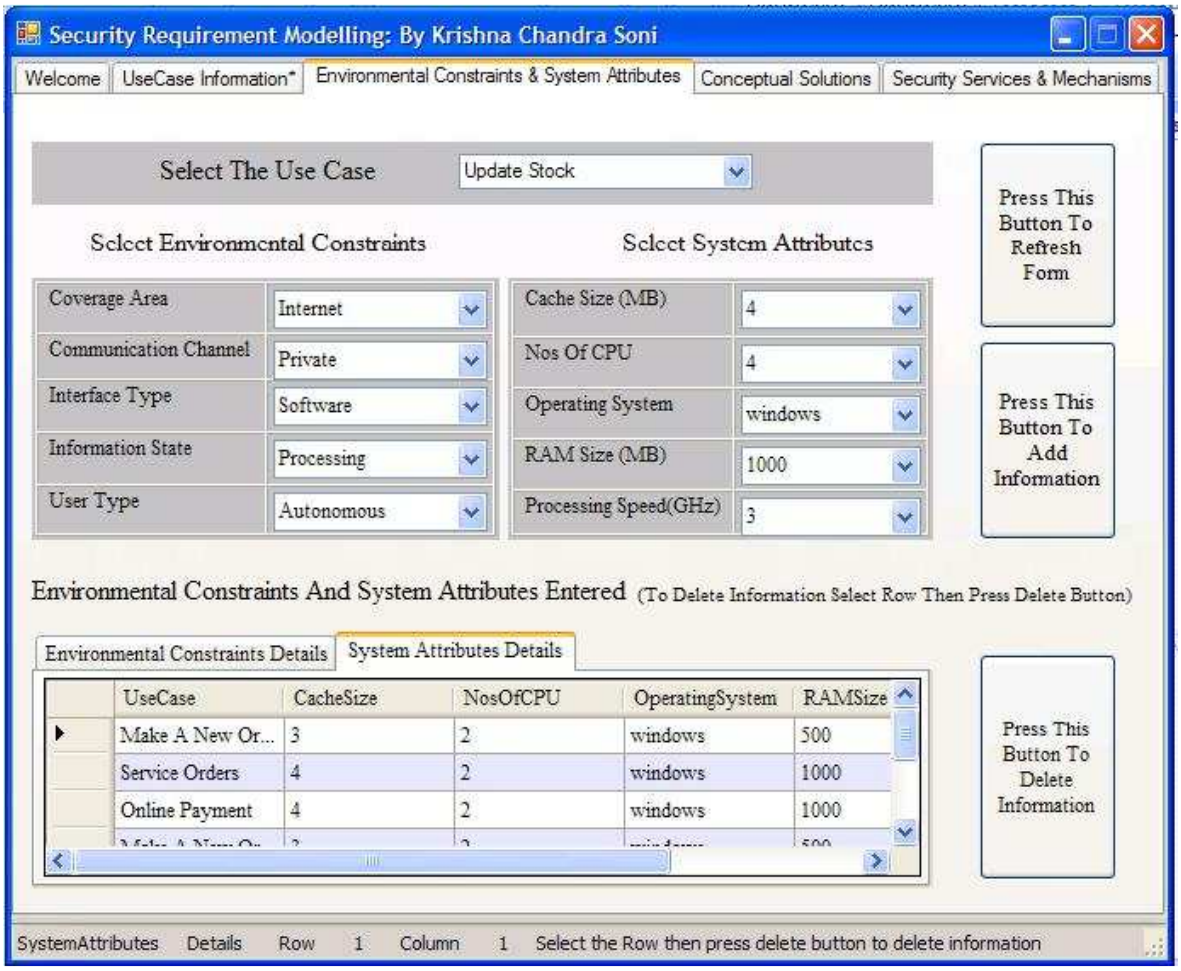

Figure 4: Identification of Environmental Constraints and System Attributes

System attributes may contain the properties of system over which use case will be implemented like RAM, cache memory, number of cpu's , processing speed etc.

3. Refinement of Conceptual Solutions for Security Requirements Fulfillment

While proposing the framework, we have mapped the different conceptual solutions , mentioned in chapter 3 on the basis of literature and internet survey's, with the security requirements, stated by Firesmiths [1].

Security Requirements	Conceptual Solutions
Authentication Requirement	• Behavioral Report

	• Identifiers • Third Party Support • Check Points
Authorization Requirement	o Access Points o Privileges o Roles o Views
Identification Requirement	• Access Points • Identifiers • Third Party Support
Immunity Requirement	o Access Controls o Behavioral Report o Identifiers o Log Reports o Privileges o Restore Points
Integrity Requirement	• Access Controls • Restore Points
Intruder Detection Requirement	o Behavior Report o Identifiers o Privileges
Non Repudiation Requirement	• Log Reports • Sessionization • Third Party Support
Physical Protection Requirement	o Access Controls

	o Trap Doors
Privacy Requirement	• Access Controls • Roles • Sessionization • Views
Security Auditing Requirement	o Log Reports o Privileges o Roles
Survivability Requirement	• Behavioral Report • Restore Points • Trap Doors
System Maintenance Requirement	o Privileges o Roles

Table 1: Mapping of Security Requirements with Conceptual Solutions

The mapping done above is high level analysis of relationship between security requirements and conceptual solutions. This can be further refined by considering the other details related to use case. For example, we have found that a particular use case like 'login account' requires identification requirement, for which we have conceptual solutions that include access controls, identifiers, and third party. But if we look at details then we will find that assets associated with it is user login information, hence we can refine our conceptual solutions to identifiers and access controls. So as much we will go in detail of any use case, we will get more refined conceptual solutions fulfilling all associated security requirements.

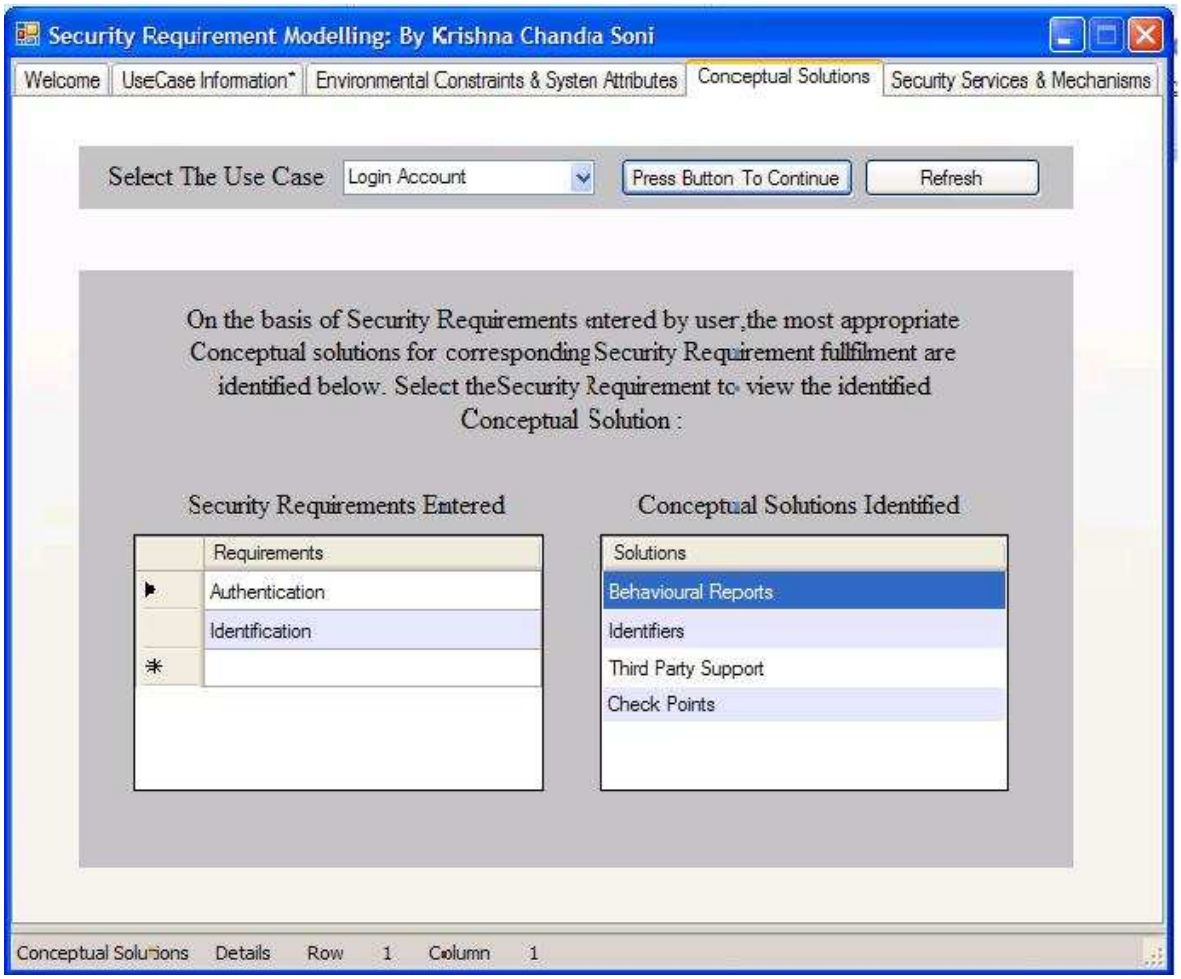

Figure 5: Conceptual solutions identified corresponding to security requirements

4. Deriving Security Services and Mechanisms:

On the basis of the threats identified during the security requirement gathering phase, we have mapped the threats to information with relevant security services. In this layer, we are performing two different tasks:

- Identification of relevant security services corresponding to threats identified.
- Then we are finding the appropriate mechanism to implement that service, on the basis of environmental constraints and system attributes. The more detail will provide us with the most suitable security mechanism.

We have mapped the threats with the relevant security services.

Threats	Security Services
Denial Of Services	Availability Security Services
Elevation Of Privileges	Authorization Security Services
Information Disclosure	Confidentiality Security Services
Repudiation	Non Repudiation Security Services
Spoofing	Authentication Security Services
Tempering	Integrity Security Services

Table 2: Mapping of Threats with Security Services

Threats considered are grouped into classes based on their effect [27]:

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

After identifying the relevant security services, now its time to map the security mechanisms that well implements the identified security services. For this we have to consider all the environmental constraints and system attributes and based on that we have framed rules which will provide us with more precise security mechanisms.. In the tool designed by us, we have framed 576 different kinds of rules by considering the threats, coverage area, user type and interface type as a criteria of selection.

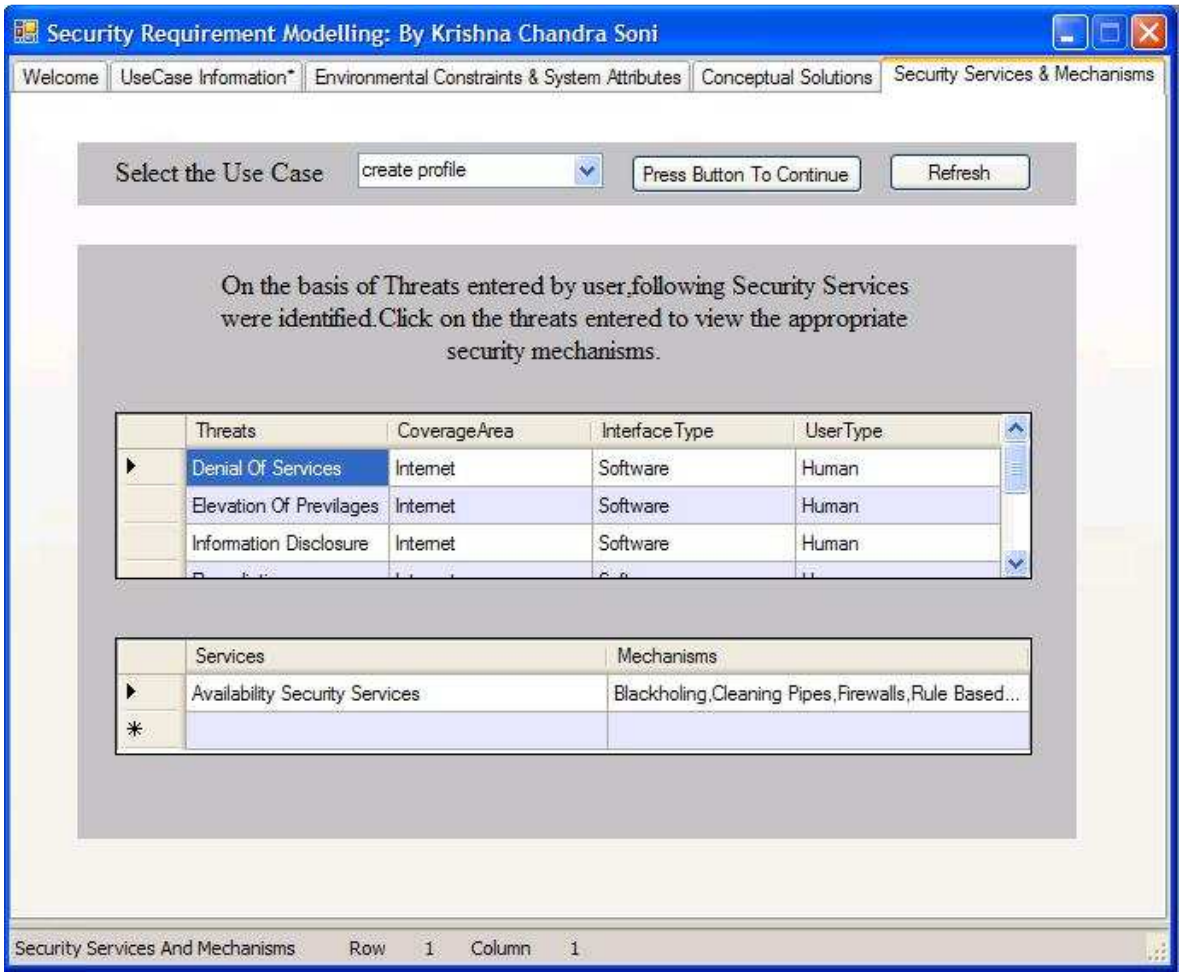

Figure 6: Identification of security services and corresponding mechanisms based on rules

Following are security mechanisms for corresponding security services:

Security Services	Security Mechanisms
Authentication Security Services	• Password mechanism • Biometric Devices (e.g.: fingerprint reader etc) • Paraphrases mechanism • Smartcards • Tokens • Symmetric key infrastructure • Public Key Infrastructures
Authorization Security Services	o Reviews Of Logs o Update Patches

	o Use Paraphrases (e.g.: Strong Passwords) o Policy based Routers & Firewalls
Availability Security Services	• Data Redundancy • Fault Tolerant Mechanisms • Firewalls • Intruder Prevention System • Cleaning Pipes • Firewalls, • Rule Based Router & Switches • Black holing • Sinking
Confidentiality Security Services	o DES o AES o Triple DES o RSA o IDEA
Integrity Security Services	• Hashing • Data Redundancy • Hamming Codes • Checksums • Hashing Algorithms(MD5, SHA1) • CRC Checksums
Non Repudiation Security Services	o Transaction Logs o Digital Signatures(Elgamal) o Digital Signatures Standards o Trusted Third Parties

Table 3: Security mechanisms corresponding to security services

We can further refine the identified security mechanisms based on environmental constraints and system attributes. E.g. for authentication security services we may have following rule:

IF

{

(USERTYPE = “HUMAN”) AND (INTERFACE = “HARDWARE”) AND (COMMUNICATION CHANNEL TYPE = “PUBLIC”)

}

THEN

{

SECURITY MECHANISMS = {“BIOMETRIC DEVICES” OR “SMART CARDS” OR “TOKENS”

}

ELSE

IF

{

(USERTYPE = “AUTONOMOUS”) AND (INTERFACE = “SOFTWARE”) AND (COMMUNICATION CHANNEL TYPE = “PUBLIC”)

}

THEN

{

SECURITY MECHANISMS = {“PUBLIC KEY INFRASTRUCTURE” OR “PASSWORDS”

}

Thus the more details of environment constraints and system attributes do we will add, the more refined will be our solutions. Hence at the end of this layer we will get all security mechanisms covering all set of threats. And in the end of whole procedure we will get sets of conceptual solutions and security mechanisms covering all sets of security requirements.

4.3 Why named it “Back tracking”?

We described the whole framework, but what is the significance of word “Back Tracking” associated with proposed framework? If we analyze the proposed framework in depth we will find that we are doing the whole analysis and finding the solutions for identified security requirements in the direction which is reverse of the any methods used to elicit security requirements. The refinements of solutions are done at each step that is required to elicit security requirements. Let’s look in depth.

To elicit security requirements, use analyze use case first, then we identify assets, then we identify threats then finally we identify security requirements. It’s a general strategy that is used in security requirement elicitation method. Now analyze our proposed framework. What we are doing is that we are moving from large set of mapped solutions to corresponding security requirements and threats towards refined set of solutions. We are ruling out or eliminating our solutions based on the criteria’s like asset identified, threat identified, environmental constraints. And if we notice then we will observe that these things (environmental constraints, system attributes, criteria, assets, threats) are identified in sequence (in general) before identifying any security requirements. Now what we did is that , we first map all possible solutions to corresponding security requirements, then we refined our solutions by mapping threats to information via security services and finding security mechanisms, then we are further

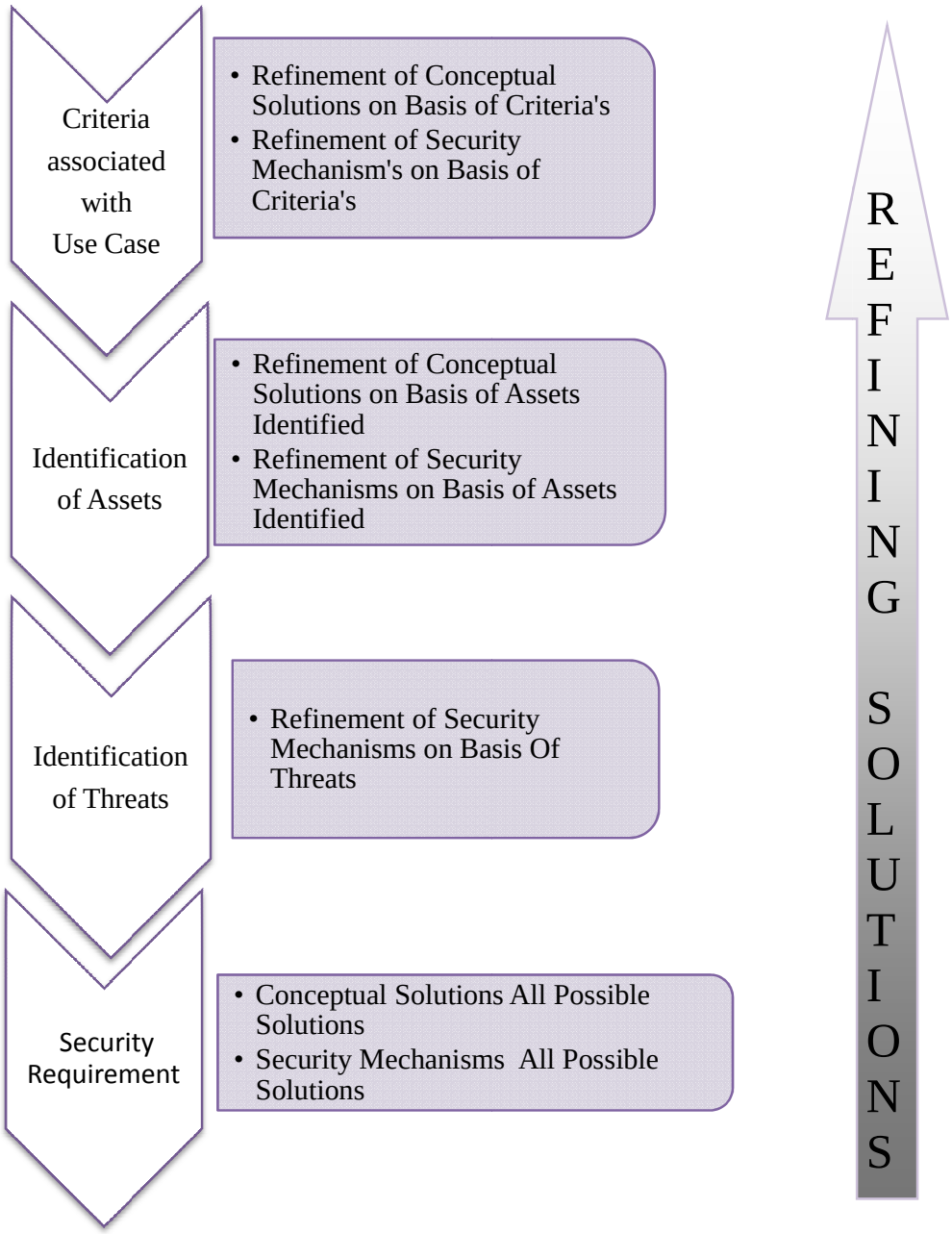

Figure 7. Back tracking of General Security Requirement Gathering Approach for Refining Solution's

refining our solutions on basis of criteria, then on basis of assets , then on basis of environmental constraints and system attributes. Thus we can see that we are traversing in backward direction to refine our solution sets covering all security services in all perspectives. Hence we named it as Backtracking approach.

4.4 Conclusion

The proposed framework covers all the 12 different kinds of security requirements as proposed by Firesmith [1]. Also provides more precise and minimum sets of solutions covering each aspect, based on details provided, thereby ensuring the complete secure software.

Chapter 5 Case Study: Online Purchasing System

In this chapter we will do a case study of “Online Purchasing System”, in which an order is made on behalf of customer. An order consists of number of items in the stock. The system should keep track of stock level of each item. The order is either pending or serviced. An invoice is made at the time of servicing the order after online payment. The sale clerk is one authorized to view details, to maintain inventory and to serve orders.

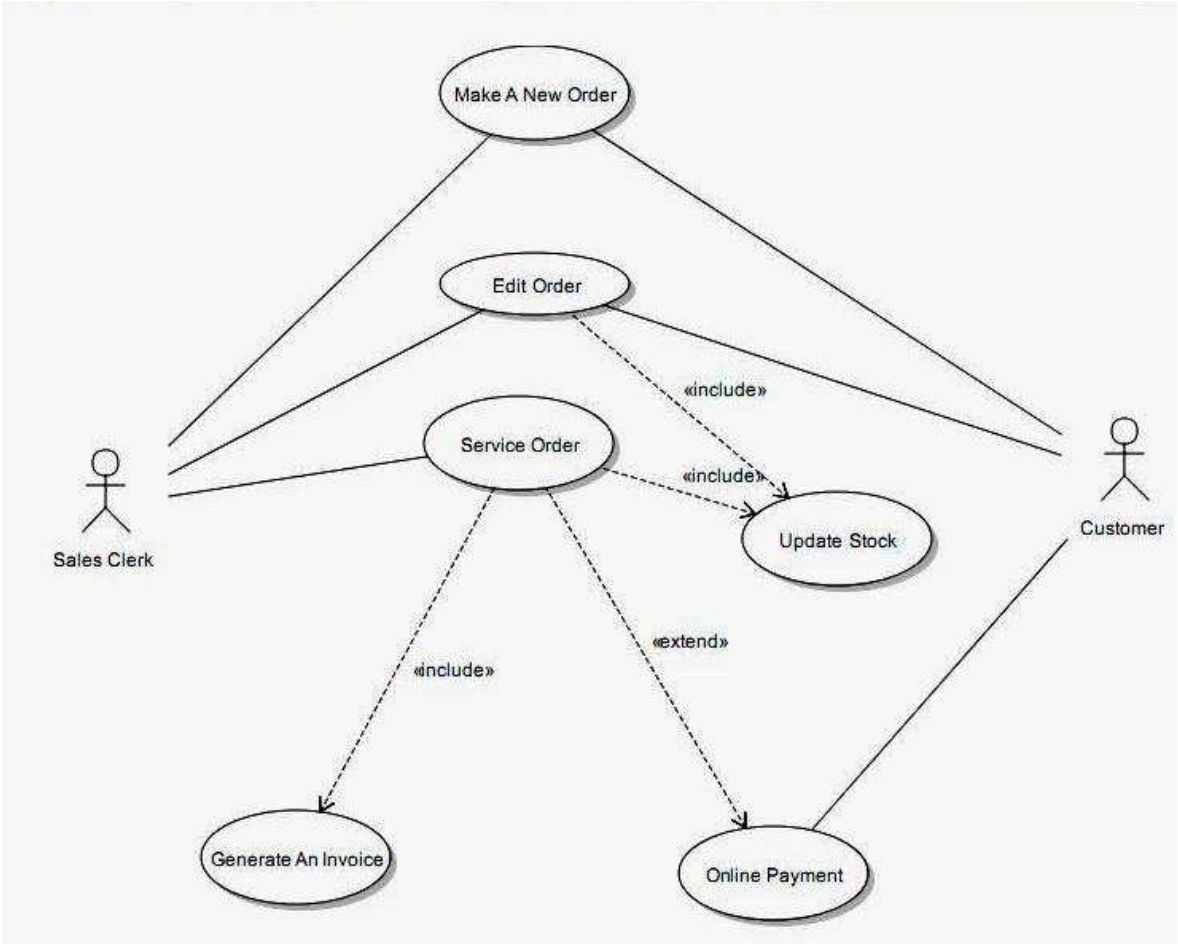

Figure 8: Use Case diagram for online purchasing system

1. Use Case: Make A New Order and Edit Orders

Step 1: Gathering Use Case Information

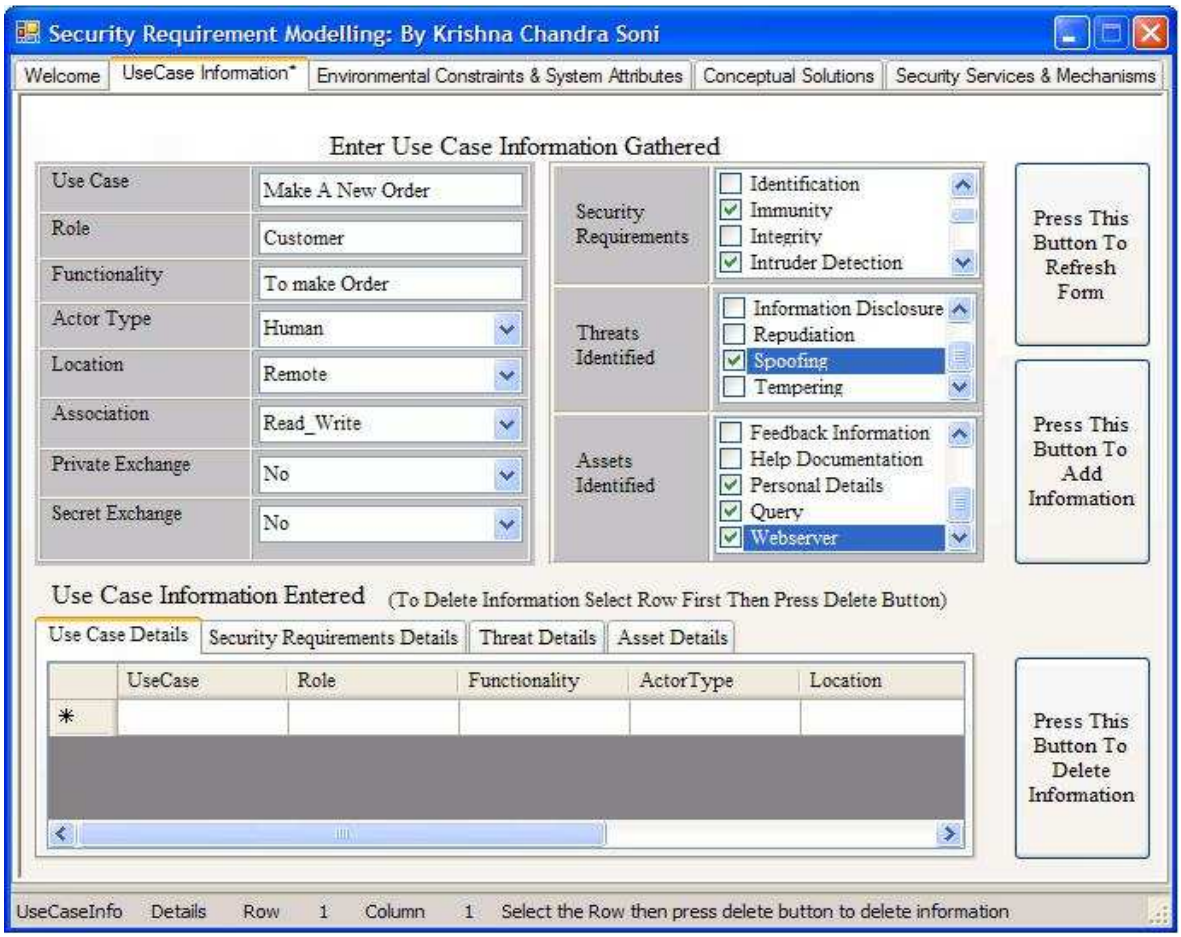

Figure 9: Gathering Use Case Information along with Common Criteria for make a new order

Assets	Threats	Security Requirements
Database	Elevation Of Privileges	Authentication
Query	Spoofing	Identification
Web server		Immunity
Personal Details		Intruder Detection
Help Documentation		Survivability

Table 4: Assets, Threats, security Requirement Details for Male a New Order

Step 2: Gathering Of Environmental Constraints and System Attributes:

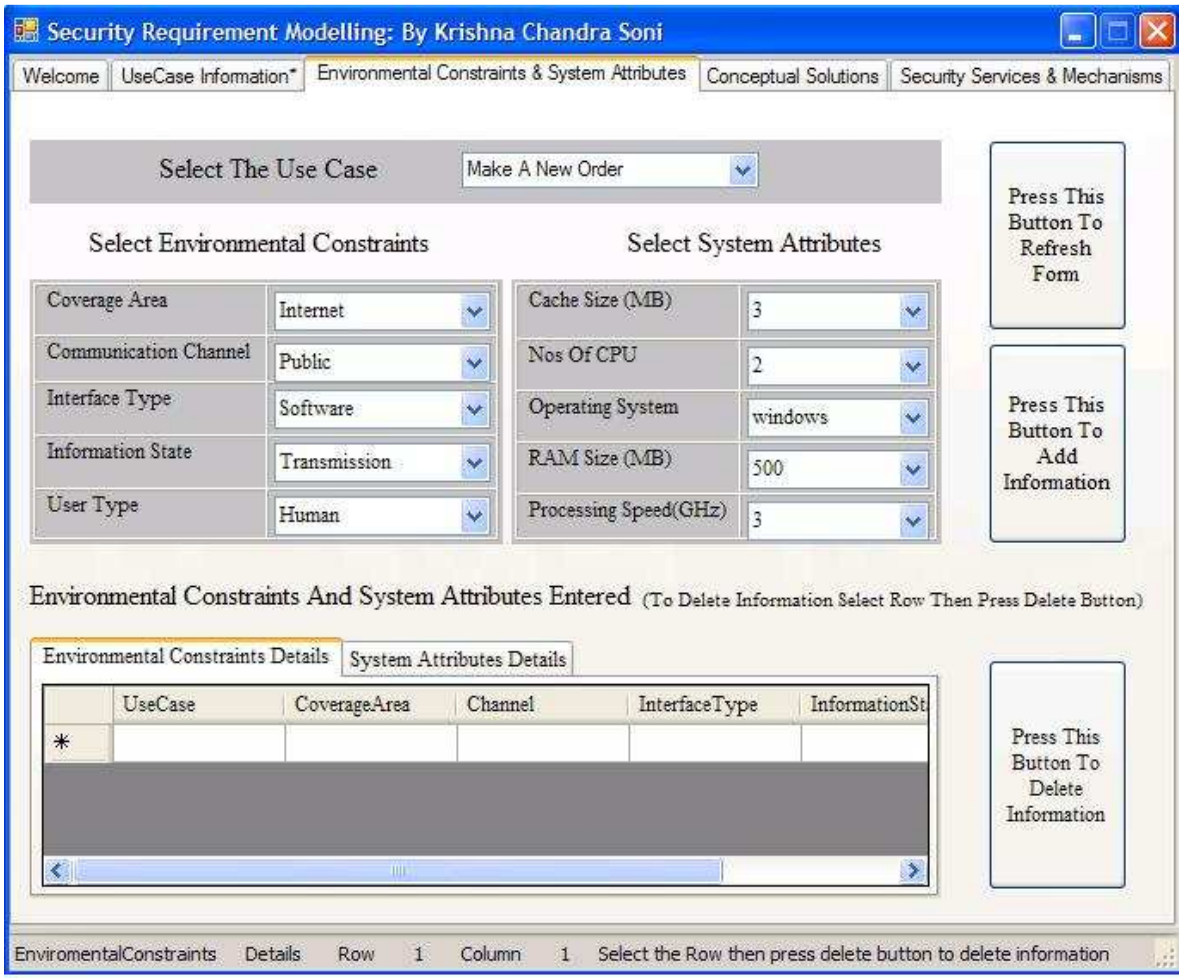

Figure 10: Environmental Constraints and System Attributes for Make a New Order

Environmental Constraints		System Attributes	
Coverage Area	Internet	Cache (MB)	3 MB
Channel	Public	Nos. Of CPU	2
Interface	Software	Operating System	Windows
Info. State	Transmission	RAM Size	500 MB
User Type	Human	Processing Speed	3 GHz

Table 5: Environmental Constraints and System Attributes Details for Male a New Order

Step 3: Conceptual Solutions

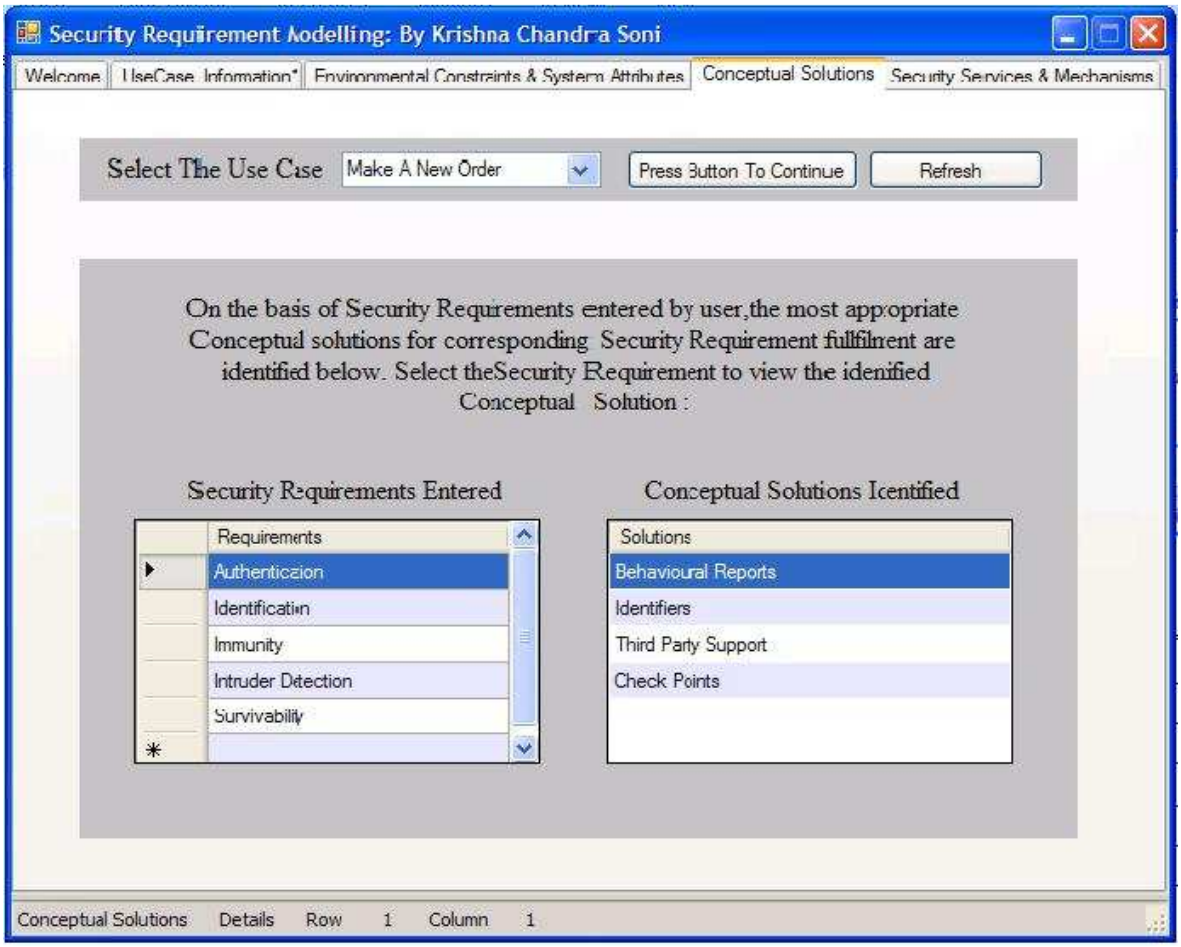

Figure 11: Conceptual Solutions for Corresponding Security Requirements of Make a New Order

Security Requirements	**Conceptual Solutions**
Authentication	• Behavioral Reports • Identifiers • Third party Support • Check Points
Identification	o Access Points o Identifiers o Third Party Support
Immunity	• Access Controls • Behavioral Reports • Restore Points

	• Identifiers • Log Reports • Privileges
Intruder Detection	o Behavioral Reports o Identifiers o Privileges
Survivability	• Behavioral Reports • Restore Points • Trap Doors

Table 6: Conceptual Solutions for Corresponding Security Requirements Details For Male New Order

Step 4: Deriving Security Services and Mechanisms:

Figure 12: Deriving Security Requirements and Relevant Mechanisms for Make a New Order

Threats	Security Services	Security Mechanisms
Elevation Of Privileges	Authorization	• Policy based Routers • Policy Based Firewalls • Server Security Software's • Update Patches • Use Paraphrases (e.g.: Strong Passwords)
Spoofing	Authentication	o Password Mechanisms o Paraphrase Mechanism o Public Key Infrastructures o Symmetric key infrastructure

Table 7: Deriving Security Requirements and Relevant Mechanisms Details for Make a New Order

2. Use Case: Service Orders

Step 1: Gathering Use Case Information

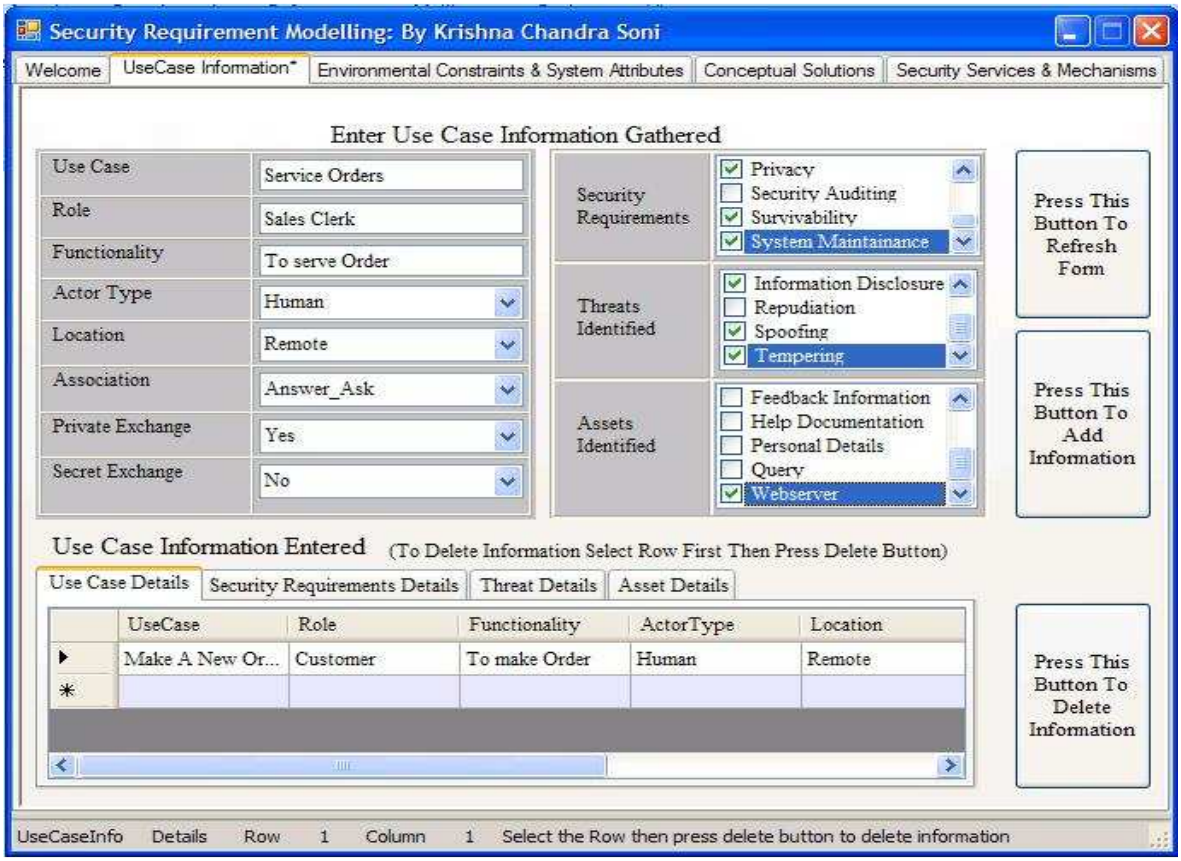

Figure 13: Gathering Use Case Information along with Common Criteria for Service Order

Assets	Threats	Security Requirements
Authentication Details	Elevation Of Privileges	Authentication
Authorization Details	Information disclosure	Authorization
Database	Spoofing	Identification
Query	Tempering	Immunity
Web server		Intruder Detection
		Privacy
		Survivability
		System Maintenance

Table 8: Gathering Assets, Threats, Requirements details for Service Order

Step 2: Gathering Of Environmental Constraints and System Attributes:

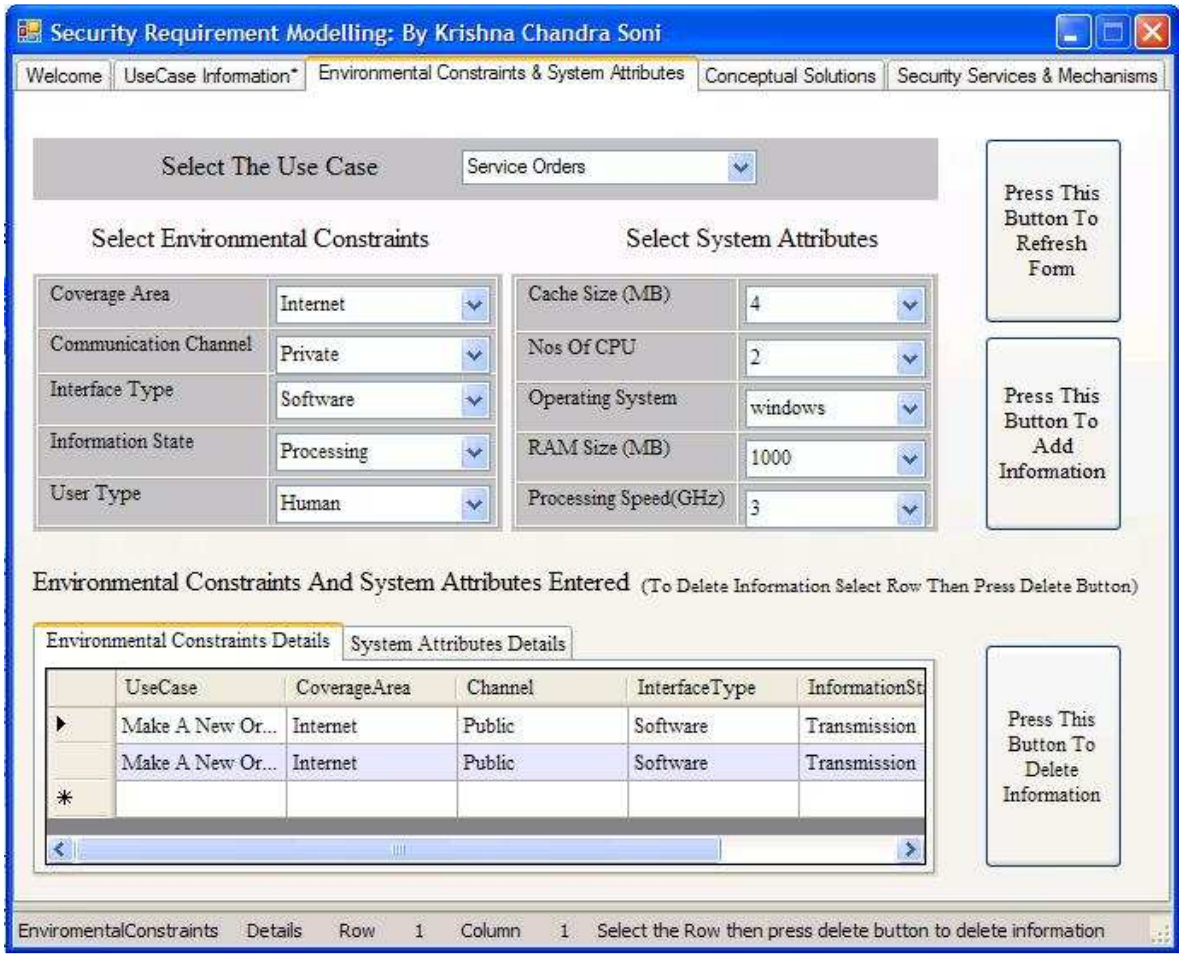

Figure 14: Environmental Constraints and System Attributes for Service Order

Coverage Area	Internet
Channel	Private
Interface	Software
Info. State	Processing
User Type	Human

Cache (MB)	4 MB
Nos. Of CPU	2
Operating System	Windows
RAM Size	1000 MB
Processing Speed	3 GHz

Table 9: Environmental Constraints and System Attributes details for Service Order

Step 3: Conceptual Solutions

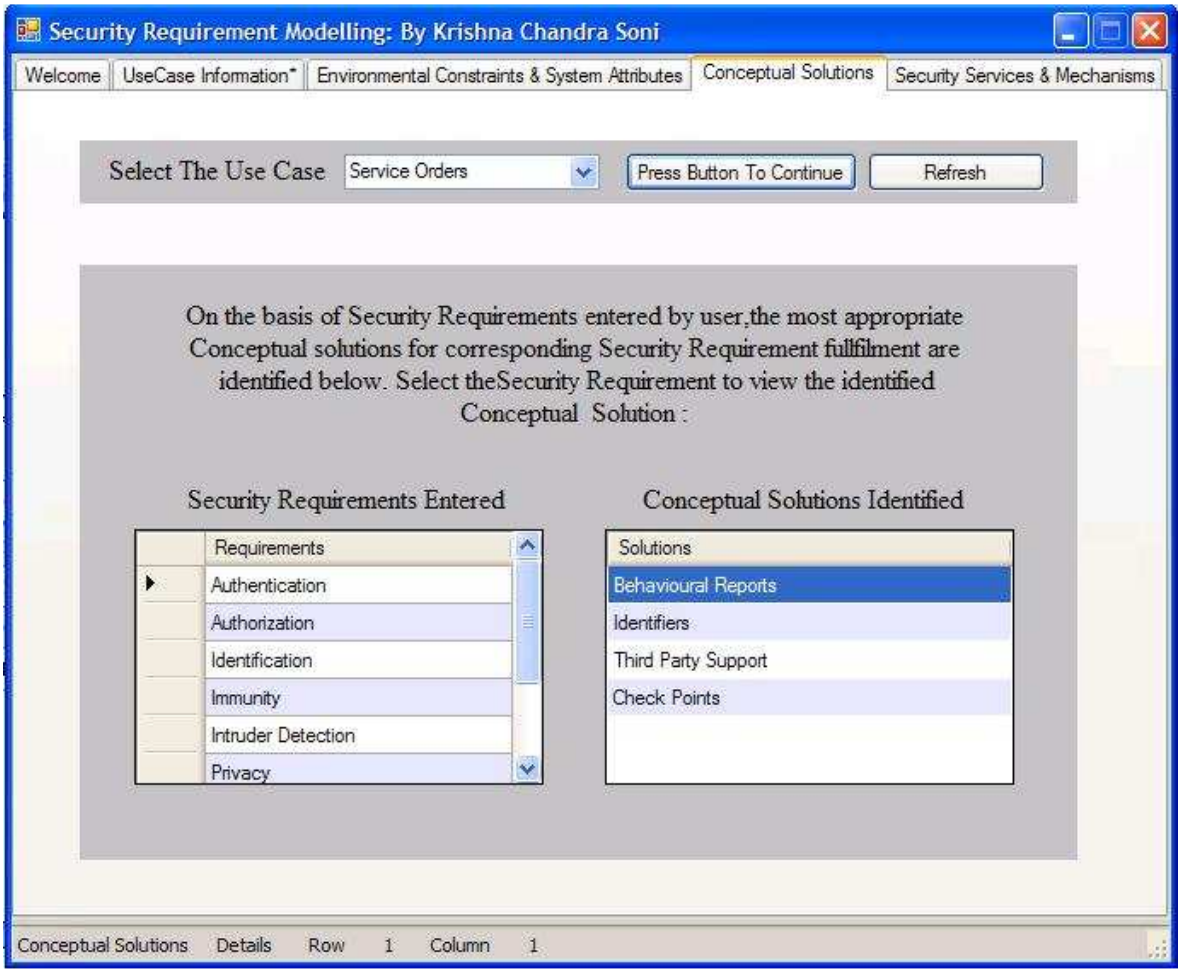

Figure 15: Conceptual Solutions for Corresponding Security Requirements for Service Order

Security Requirements	Conceptual Solutions
Authentication	• Behavioral Reports • Identifiers • Third party Support • Check Points
Authorization	o Access Points o Privileges o Roles o Views
Identification	• Access Points • Identifiers • Third Party Support
Immunity	o Access Controls o Behavioral Reports o Restore Points o Identifiers o Log Reports o Privileges
Intruder Detection	• Behavioral Reports • Identifiers • Privileges
Privacy	o Access Controls o Roles o Sessionization o Views
Survivability	• Behavioral Reports • Restore Points • Trap Doors

System Maintenance	o Privileges o Roles

Table 10: Conceptual Solutions for Corresponding Security Requirements for Service Order

Step 4: Deriving Security Services and Mechanisms:

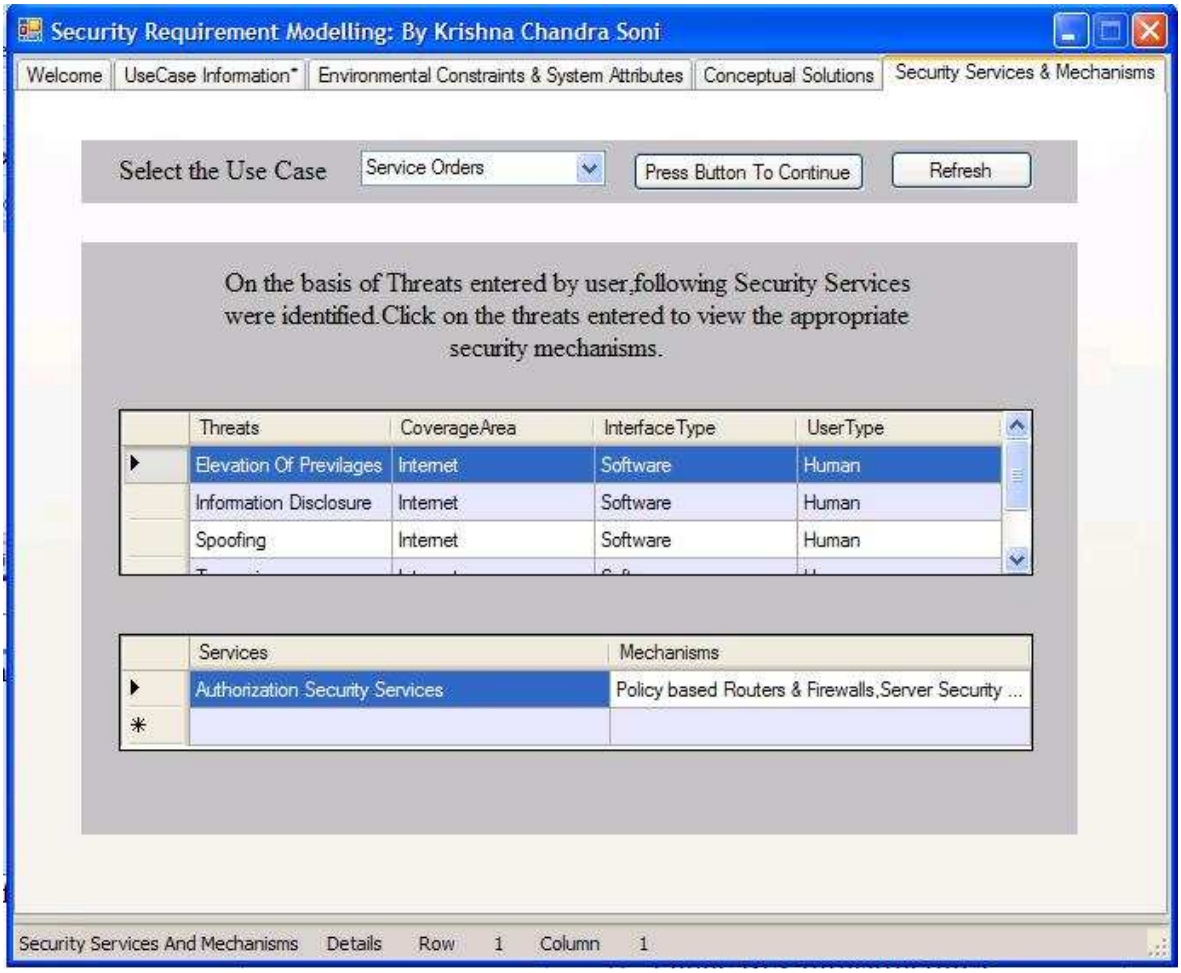

Figure 16: Deriving Security Requirements and Relevant Mechanisms for Service Order

Threats	Security Services	Security Mechanisms
Elevation Of Privileges	Authorization	• Policy based Routers • Policy Based Firewalls • Server Security Software's • Update Patches • Use Paraphrases (e.g.: Strong Passwords)

Information Disclosure	Confidentiality	o Triple DES o RSA o IDEA
Spoofing	Authentication	• Password Mechanisms • Paraphrase Mechanism • Public Key Infrastructures • Symmetric key infrastructure
Tempering	Integrity	o Hashing Algorithms(MD5, SHA1) o Hamming Codes o CRC Checksums

Table11: Deriving Security Requirements and Relevant Mechanisms Details for Service Order

3. Use Case: Online Payment

Step 1: Gathering Use Case Information

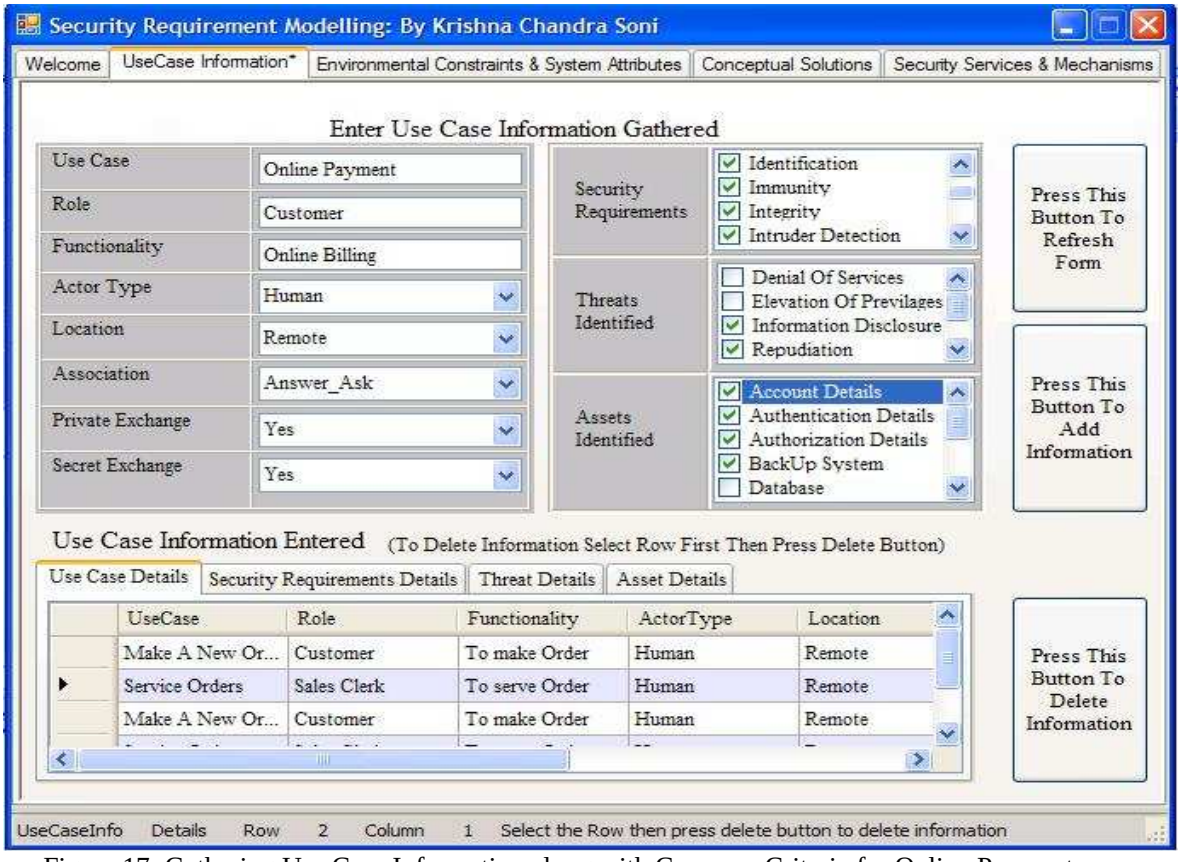

Figure 17: Gathering Use Case Information along with Common Criteria for Online Payment

Assets	Threats	Security Requirements
Account Details	Elevation Of Privileges	Authentication
Authentication Details	Information Disclosure	Authorization
Authorization Details	Repudiation	Identification
Backup system	Spoofing	Immunity
Database	Tempering	Integrity
Query		Intruder Detection
Web server		Privacy
Personal Details		Survivability
Help Documentation		

Table 12: Use Case Information details for Online Payment

Step 2: Gathering Of Environmental Constraints and System Attributes:

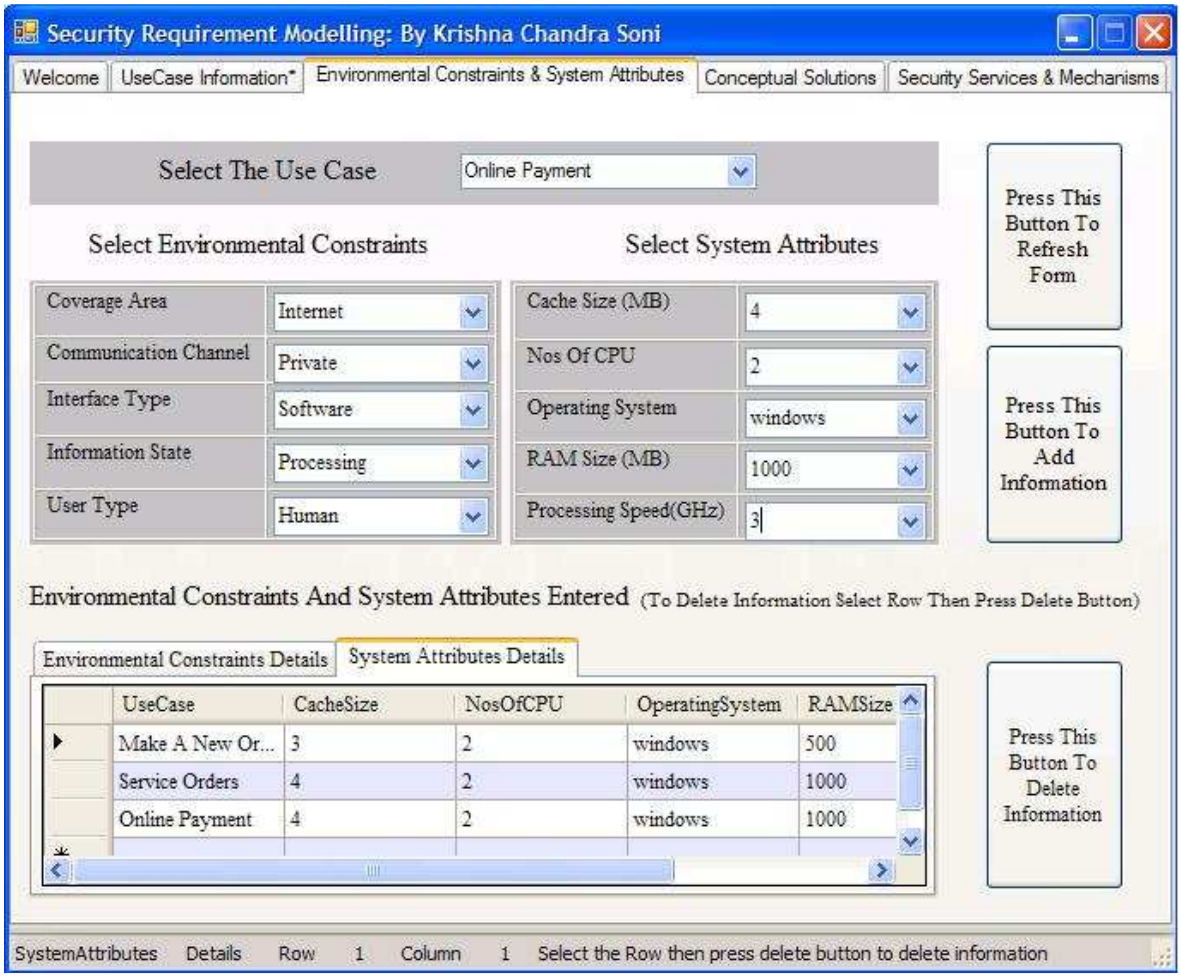

Figure 18: Environmental Constraints and System Attributes for Online Payment

Coverage Area	Internet
Channel	Private
Interface	Software
Info. State	Processing
User Type	Human

Cache (MB)	4 MB
Nos. Of CPU	2
Operating System	Windows
RAM Size	1000 MB
Processing Speed	3 GHz

Table 13: Environmental Constraints and System Attributes details for online payment

Step 3: Conceptual Solutions

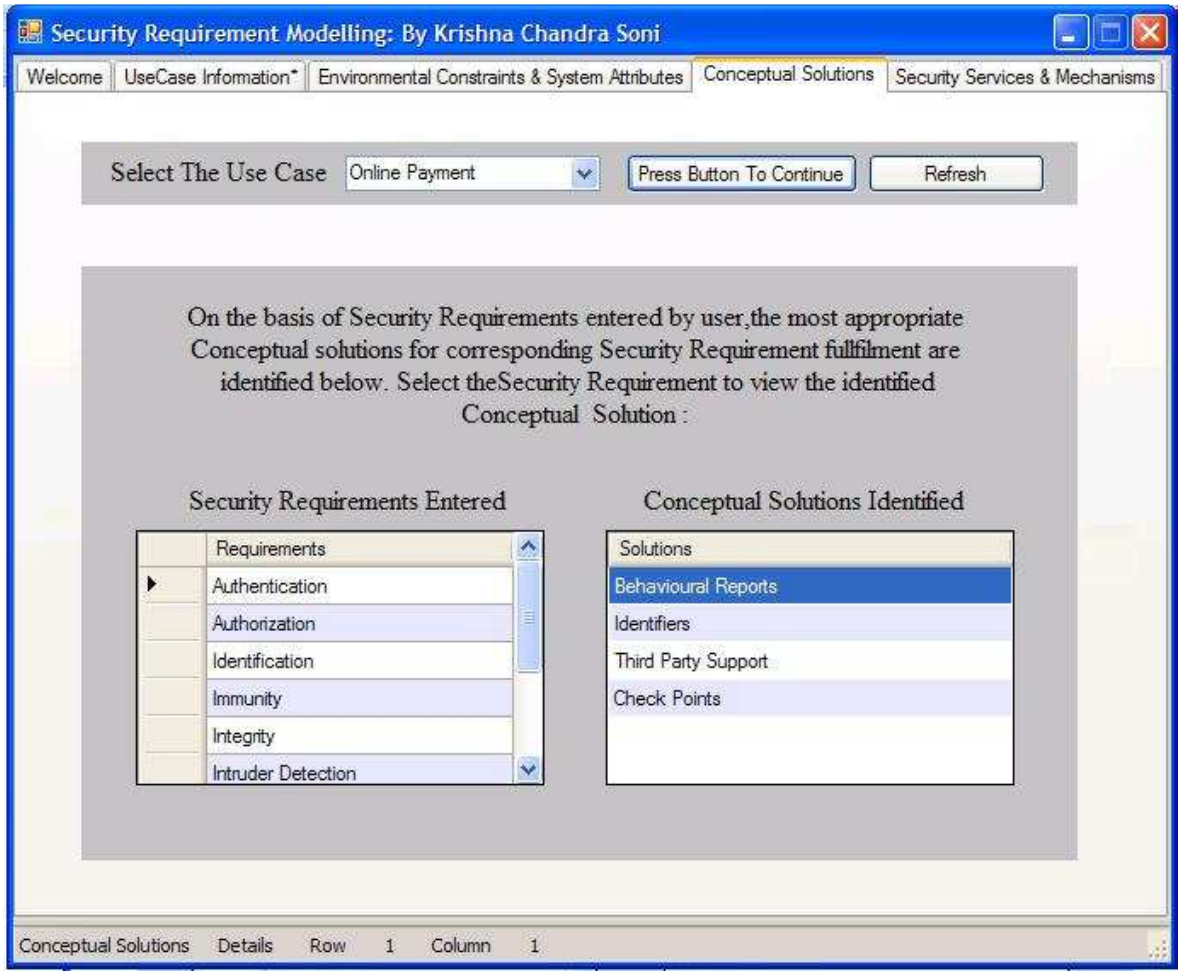

Figure 19: Conceptual Solutions for Corresponding Security Requirements for Online Payment

Security Requirements	Conceptual Solutions
Authentication	• Behavioral Reports • Identifiers • Third party Support • Check Points
Authorization	o Access Points o Privileges o Roles o Views
Identification	• Access Points • Identifiers • Third Party Support
Immunity	o Access Controls o Behavioral Reports o Restore Points o Identifiers o Log Reports o Privileges
Integrity	• Access Controls • Restore Points
Intruder Detection	o Behavioral Reports o Identifiers o Privileges
Non Repudiation	• Log Reports • Sessionization • Third Party Support
Privacy	o Access Controls o Roles o Sessionization o Views
Survivability	• Behavioral Reports • Restore Points • Trap Doors

Table 14: Conceptual Solutions for Corresponding Security Requirements for Online Payment

Step 4: Deriving Security Services and Mechanisms:

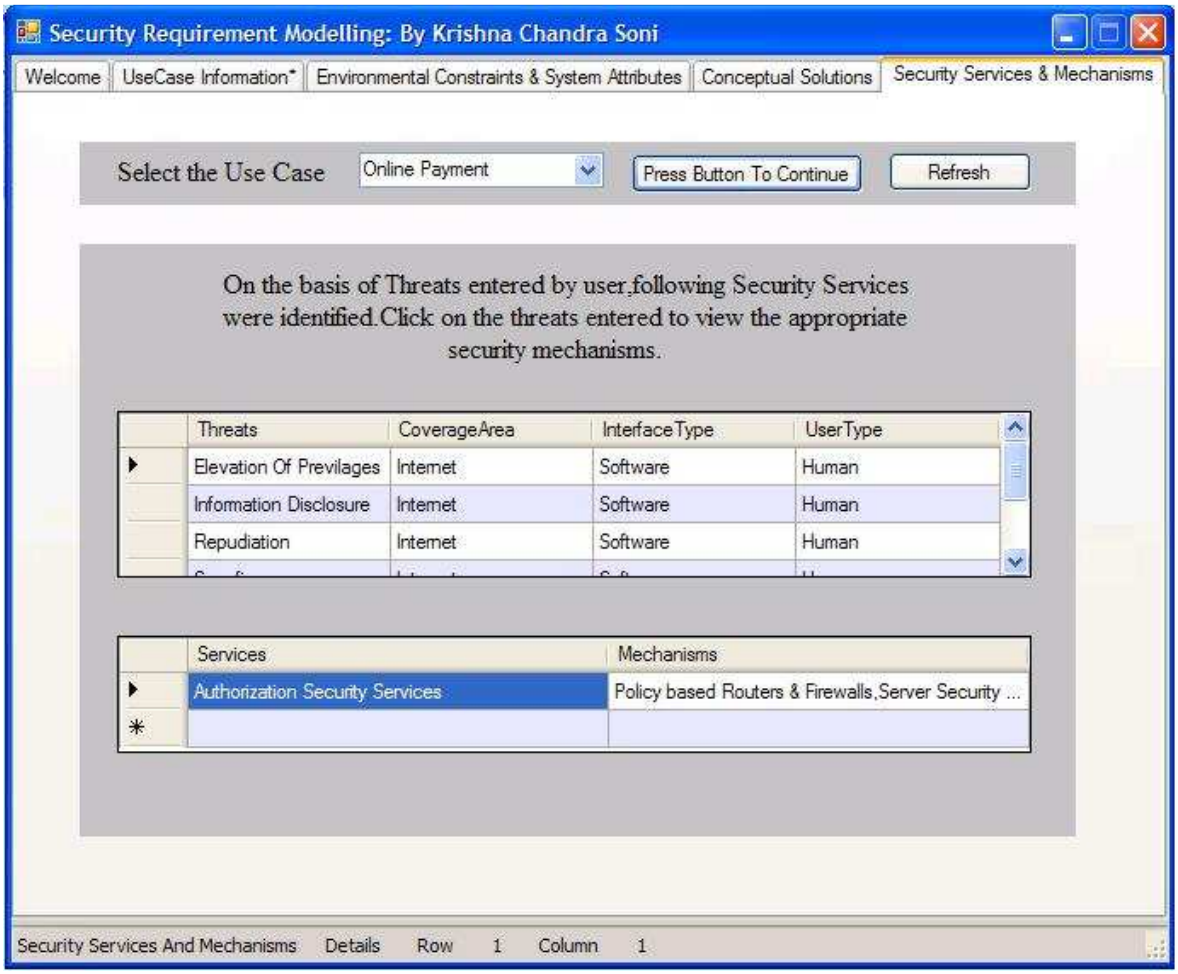

Figure 20: Deriving Security Requirements and Relevant Mechanisms for Online Payment

Threats	Security Services	Security Mechanisms
Elevation Of Privileges	Authorization	• Policy based Routers • Policy Based Firewalls • Server Security Software's • Update Patches • Use Paraphrases (e.g.: Strong Passwords)
Information Disclosure	Confidentiality	• Triple DES • RSA • IDEA

Repudiation	Non Repudiation	• Trusted Third Parties • Transaction Logs • Digital Signatures Standards
Spoofing	Authentication	o Password Mechanisms o Paraphrase Mechanism o Public Key Infrastructures o Symmetric key infrastructure
Tempering	Integrity	• Hashing Algorithms(MD5, SHA1) • Hamming Codes • CRC Checksums

Table 15: Deriving Security Requirements and Relevant Mechanisms details for Online Payment

4. Use Case: Update Stock

Step 1: Gathering Use Case Information

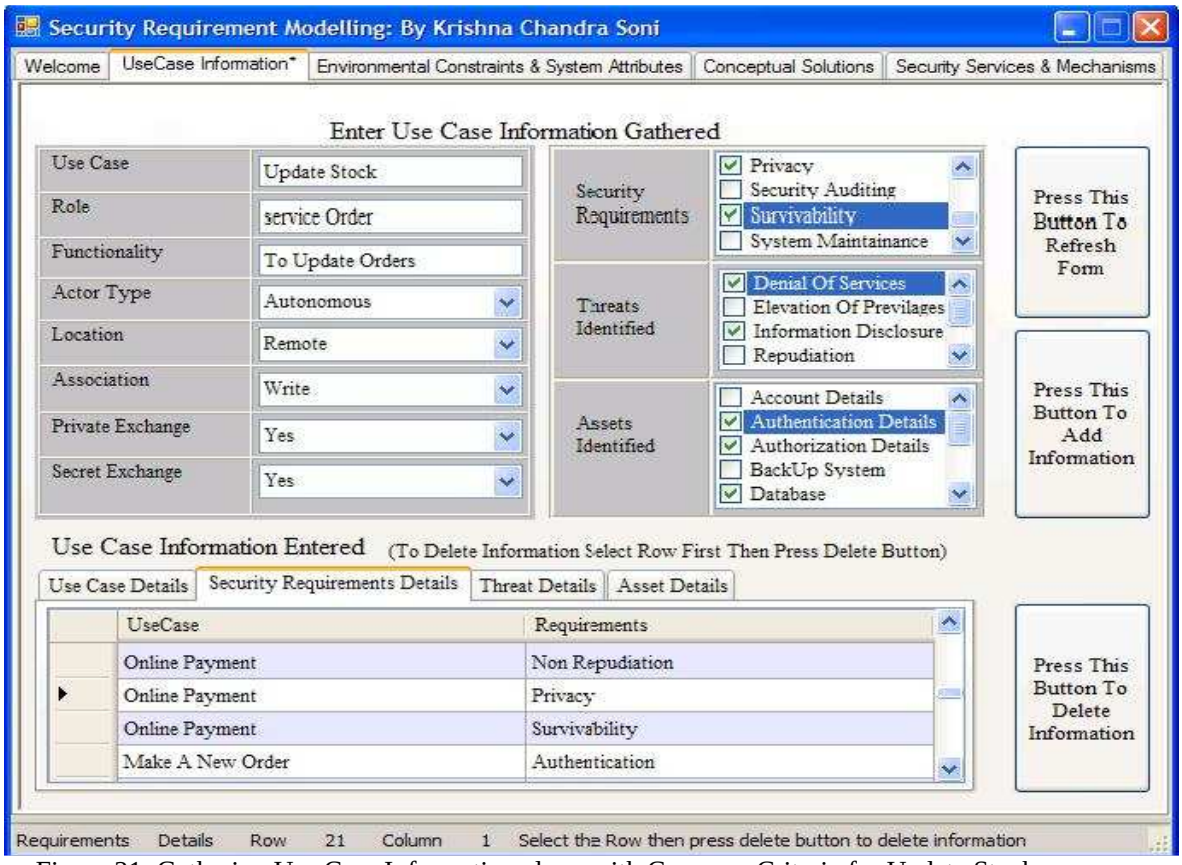

Figure 21: Gathering Use Case Information along with Common Criteria for Update Stock

Assets	Threats	Security Requirements
Account Details	Denial Of Services	Authentication
Authentication Details	Information Disclosure	Authorization
Authorization Details	Spoofing	Identification
Backup system	Tempering	Immunity
Database		Integrity
Web server		Intruder Detection
		Privacy
		Survivability

Table 16: Gathering Use Case Information for Update Stock

Step 2: Gathering Of Environmental Constraints and System Attributes:

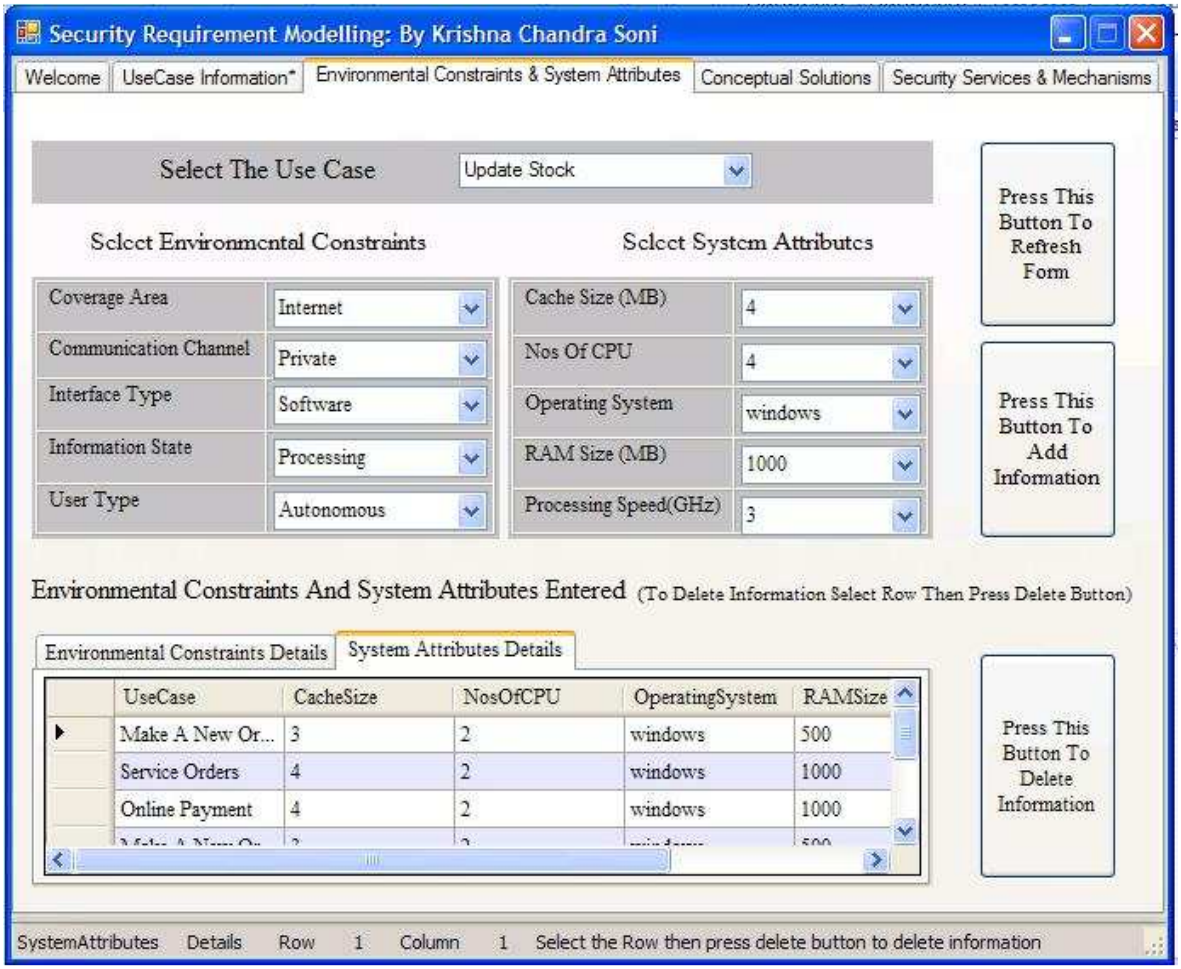

Figure 22: Environmental Constraints and System Attributes for Update Stock

Environmental Constraints System Attributes

Coverage Area	Internet
Channel	Private
Interface	Software
Info. State	Processing
User Type	Autonomous

Cache (MB)	4 MB
Nos. Of CPU	4
Operating System	Windows
RAM Size	1000 MB
Processing Speed	3 GHz

Table 17: Environmental Constraints and System Attributes details for Update Stock

Step 3: Conceptual Solutions

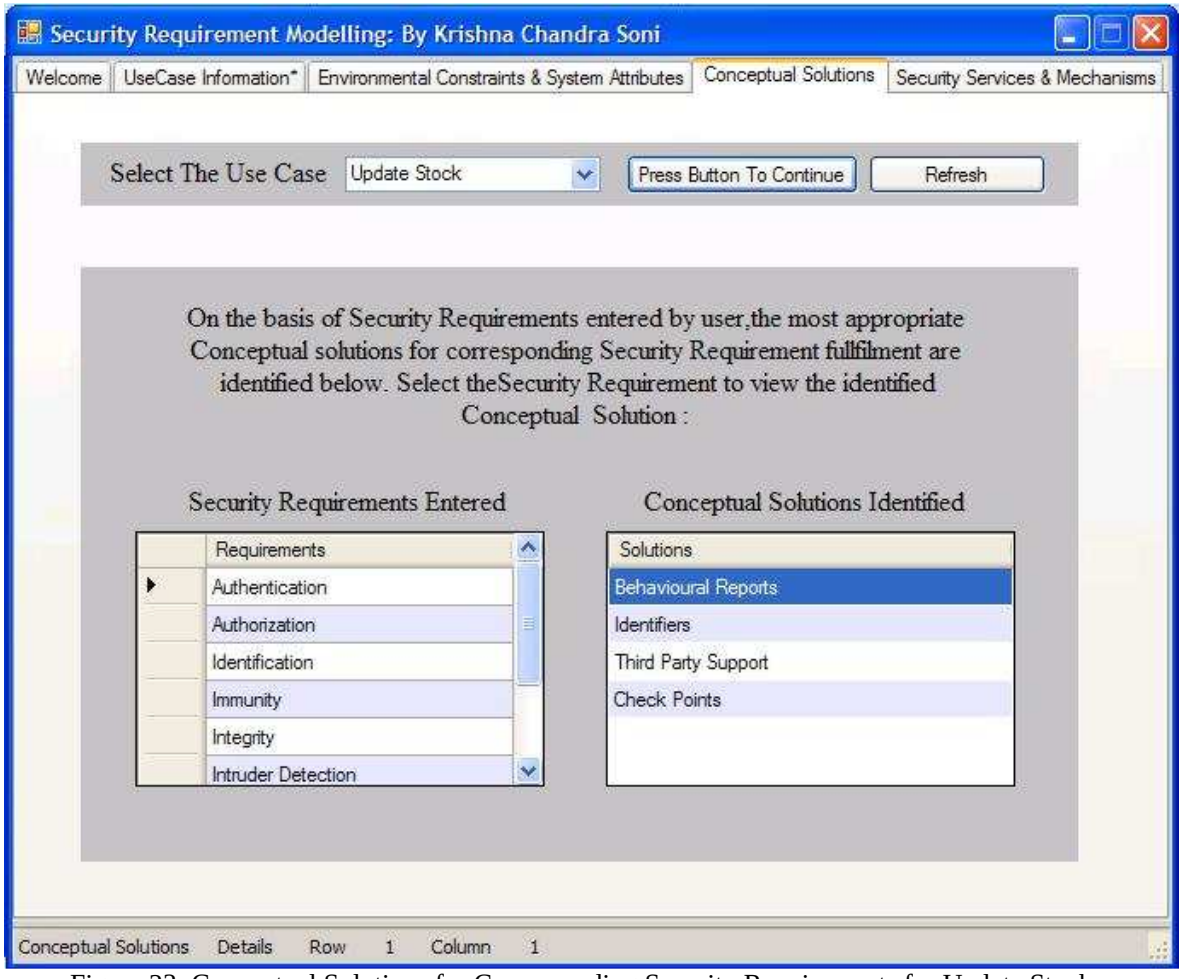

Figure 23: Conceptual Solutions for Corresponding Security Requirements for Update Stock

Security Requirements	Conceptual Solutions
Authentication	• Behavioral Reports • Identifiers • Third party Support • Check Points
Authorization	o Access Points o Privileges o Roles o Views
Identification	• Access Points • Identifiers • Third Party Support
Immunity	o Access Controls o Behavioral Reports o Restore Points o Identifiers o Log Reports o Privileges
Integrity	• Access Controls • Restore Points
Intruder Detection	o Behavioral Reports o Identifiers o Privileges
Privacy	o Access Controls o Roles o Sessionization o Views
Survivability	• Behavioral Reports • Restore Points • Trap Doors

Table 18: Conceptual Solutions for Corresponding Security Requirements details for Update Stock

Step 4: Deriving Security Services And Mechanisms:

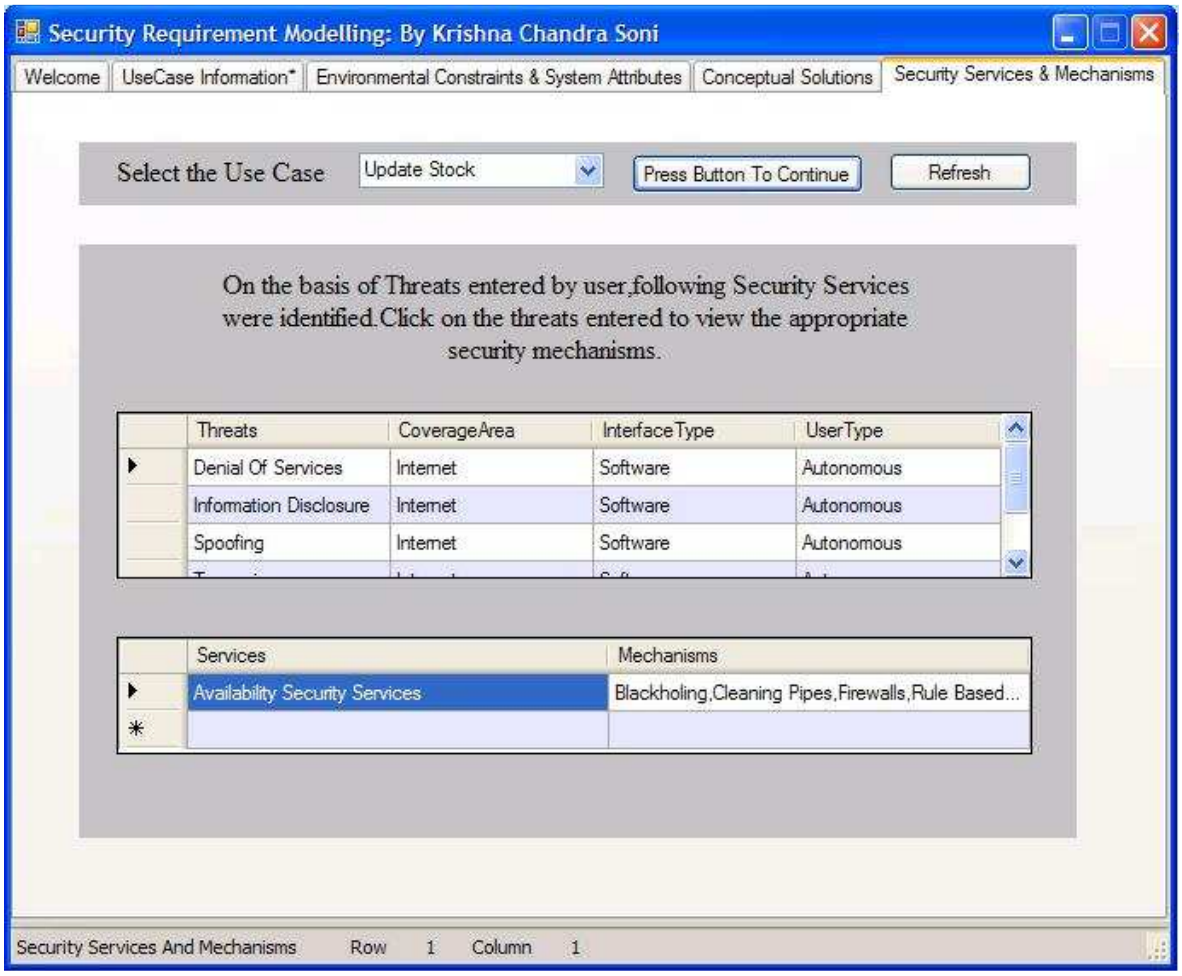

Figure 24: Deriving Security Requirements and Relevant Mechanisms for Update Stock

Threats	Security Services	Security Mechanisms
Denial Of Services	Availability	• Black holing • Cleaning Pipes • Firewalls • Rule Based Router & Switches • Sinking
Information Disclosure	Confidentiality	o Triple DES o RSA o IDEA
Spoofing	Authentication	• Password Mechanisms • Paraphrase Mechanism • Public Key Infrastructures

		• Symmetric key infrastructure
Tempering	Integrity	o Hashing Algorithms(MD5, SHA1) o Hamming Codes o CRC Checksums

Table 19: Deriving Security Requirements and Relevant Mechanisms Details for Update Stock

5. Use Case: Generate an Invoice

Step 1: Gathering Use Case Information

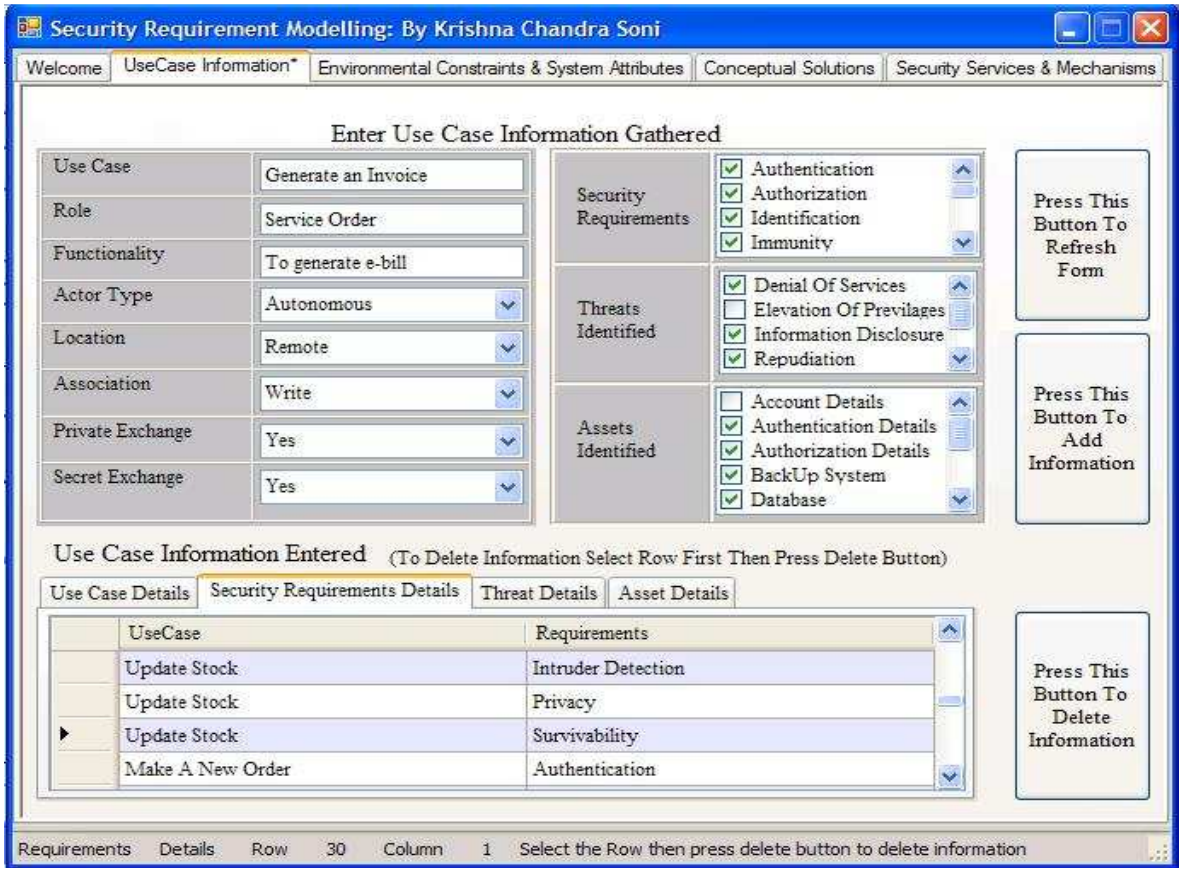

Figure 25: Gathering Use Case Information along with Common Criteria for Generate an Invoice

Assets	Threats	Security Requirements
Authentication Details	Denial Of Services	Authentication
Authorization Details	Information Disclosure	Authorization
Backup system	Spoofing	Identification
Database	Tempering	Immunity
Web server		Integrity
		Intruder Detection
		Privacy
		Survivability

Table 20: Gathering Use Case Information for Generate an Invoice

Step 2: Gathering Of Environmental Constraints and System Attributes:

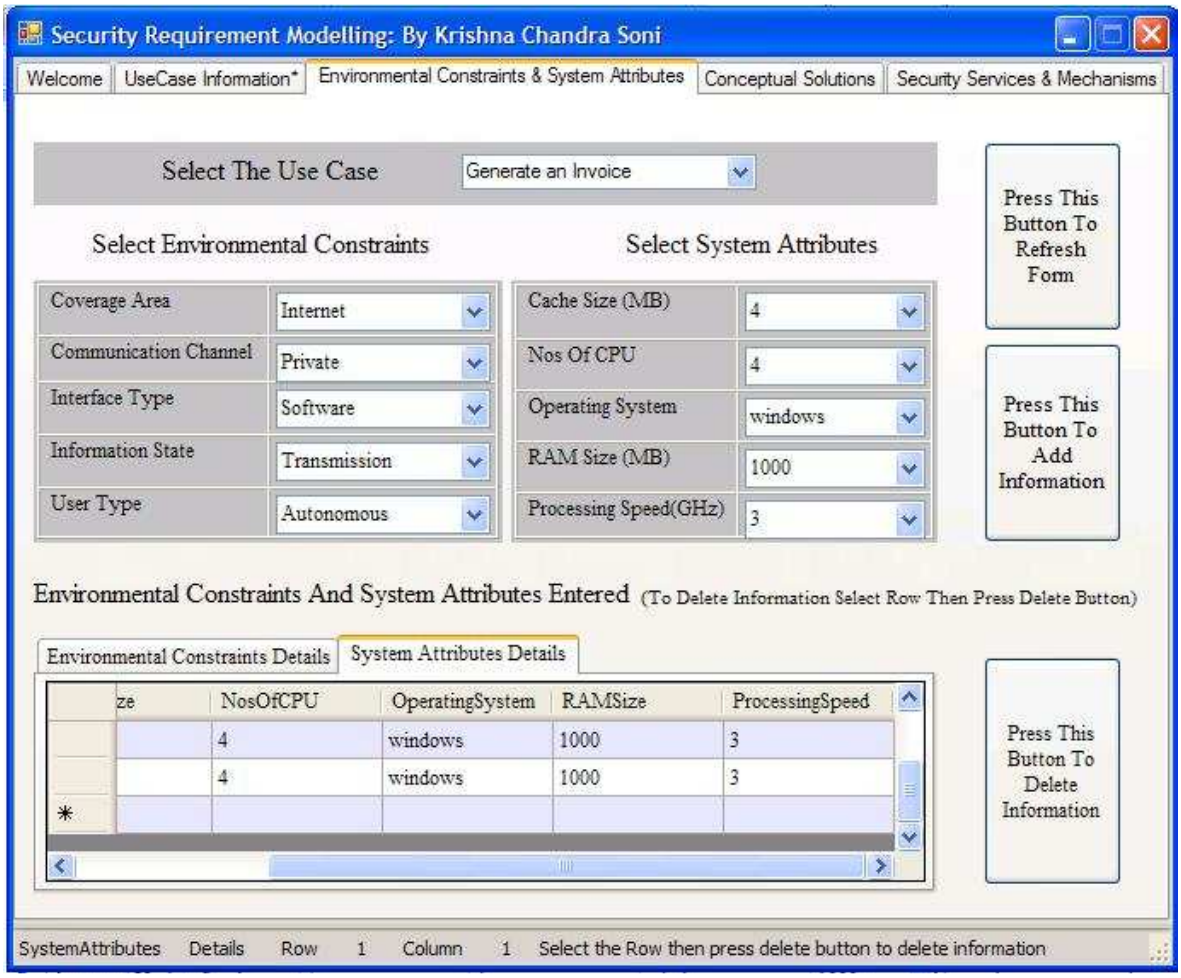

Figure 26: Environmental Constraints and System Attributes for Generate an Invoice

Environmental Constraints		System Attributes	
Coverage Area	Internet	Cache (MB)	4 MB
Channel	Private	Nos. Of CPU	4
Interface	Software	Operating System	Windows
Info. State	Transmission	RAM Size	1000 MB
User Type	Autonomous	Processing Speed	3 GHz

Table 21: Environmental Constraints and System Attributes Details for Generate an Invoice

Step 3: Conceptual Solutions

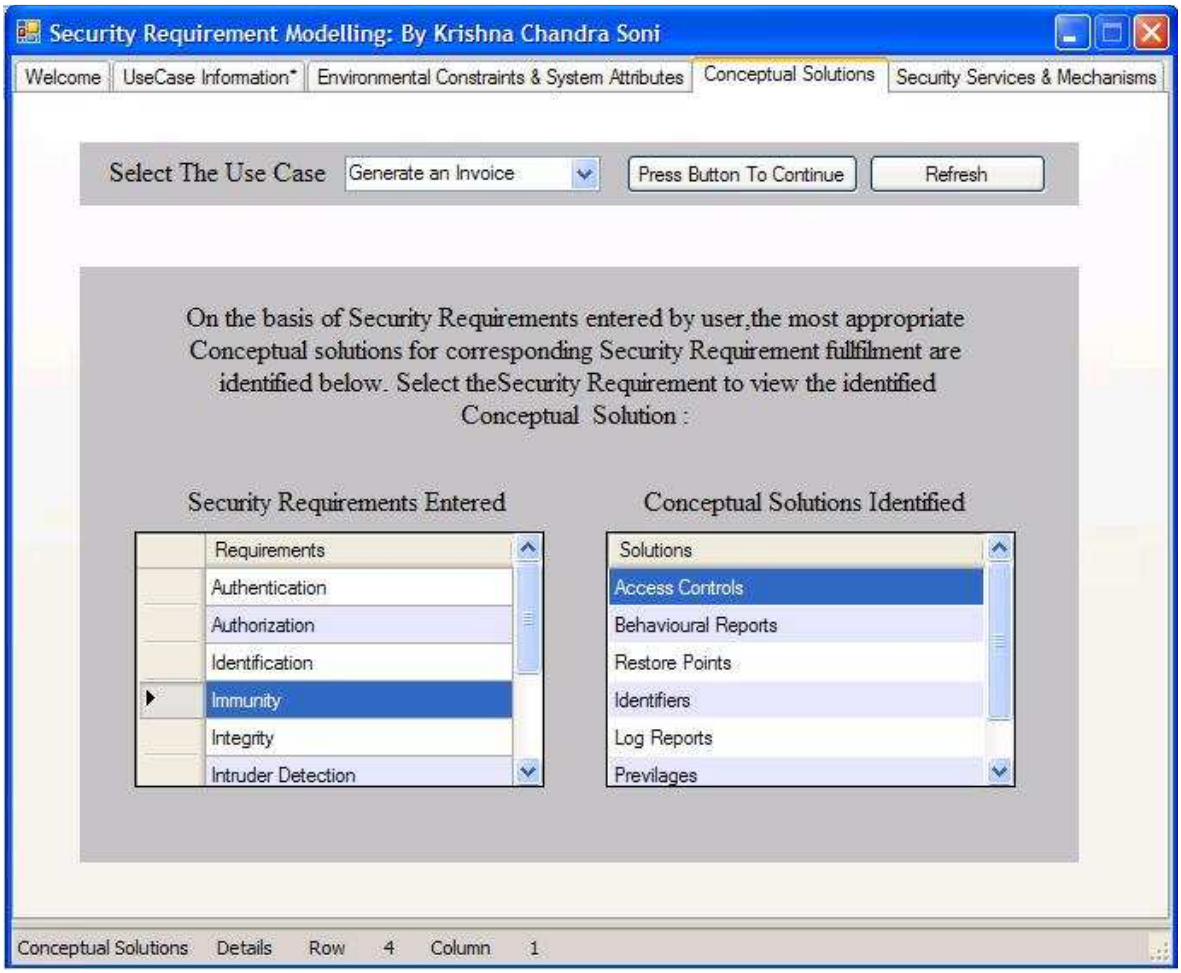

Figure 27: Conceptual Solutions for Corresponding Security Requirements for Generate an Invoice

Security Requirements	Conceptual Solutions
Authentication	• Behavioral Reports • Identifiers • Third party Support • Check Points
Authorization	o Access Points o Privileges o Roles o Views
Identification	• Access Points • Identifiers • Third Party Support
Immunity	o Access Controls o Behavioral Reports o Restore Points o Identifiers o Log Reports o Privileges
Integrity	• Access Controls • Restore Points
Intruder Detection	o Behavioral Reports o Identifiers o Privileges
Non Repudiation	• Log Reports • Sessionization • Third Party Support
Privacy	o Access Controls o Roles o Sessionization o Views

Survivability	• Behavioral Reports • Restore Points • Trap Doors	

Table 22: Conceptual Solutions for Corresponding Security Requirements details for Generate an Invoice

Step 4: Deriving Security Services And Mechanisms:

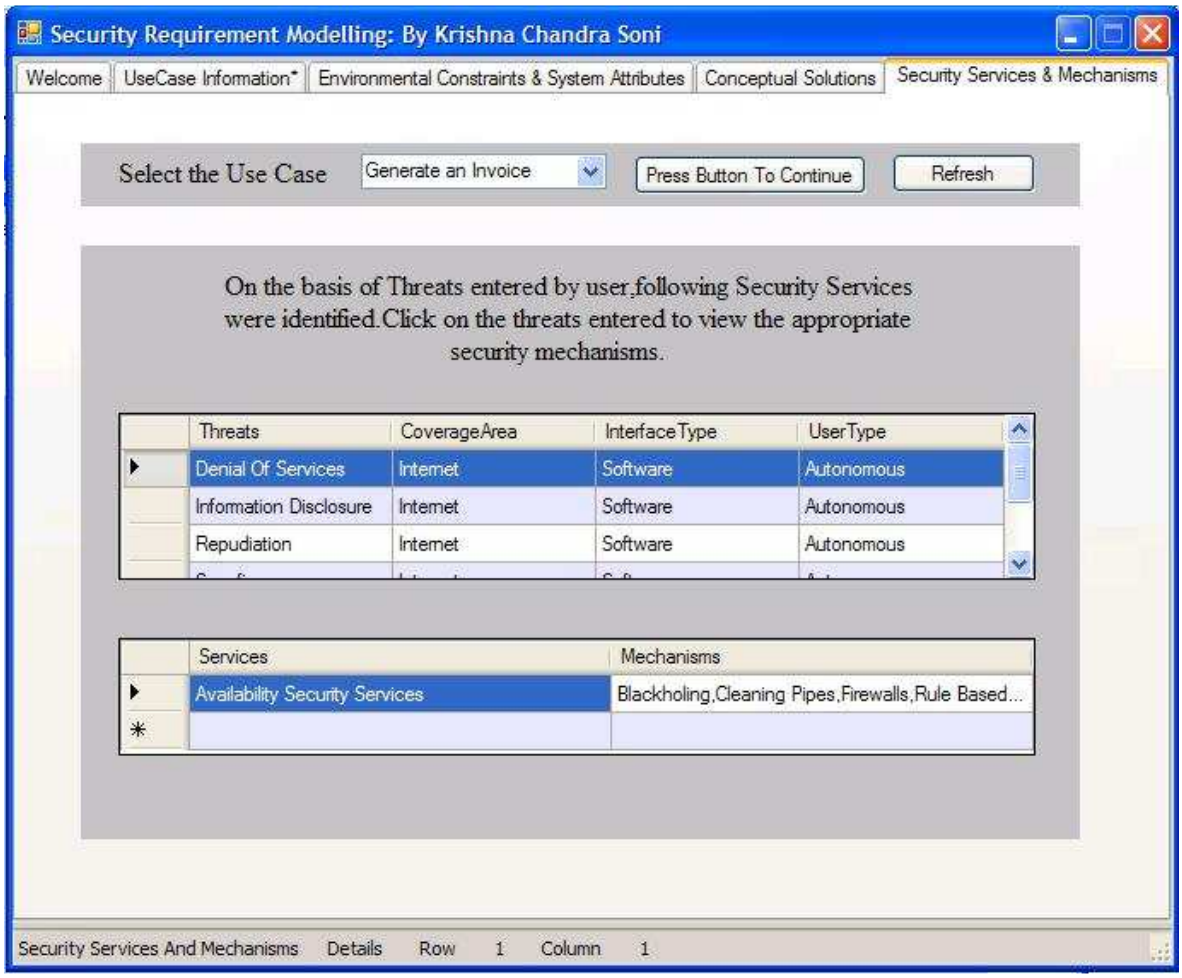

Figure 28: Deriving Security Requirements and Relevant Mechanisms for Generate an Invoice

Threats	Security Services	Security Mechanisms
Denial Of Services	Availability	• Black holing • Cleaning Pipes

		• Firewalls • Rule Based Router & Switches • Sinking
Information Disclosure	Confidentiality	o Triple DES o RSA o IDEA
Repudiation	Non Repudiation	• Trusted Third Parties • Transaction Logs • Digital Signatures Standards
Spoofing	Authentication	o Password Mechanisms o Paraphrase Mechanism o Public Key Infrastructures o Symmetric key infrastructure
Tempering	Integrity	• Hashing Algorithms(MD5, SHA1) • Hamming Codes • CRC Checksums

Table 23: Deriving Security Requirements and Relevant Mechanisms details for Generate an Invoice

Chapter 6 Conclusion and Future Scope

After analyzing the need of security in information system , we have proposed a new framework which not only covers the 12 requirements stated by Firesmith[1] , but also provides the most compact solution set on basis of the rules which are used to filter out the solution sets.

Apart from the , the solution set generated by the framework targets the particular requirement on basis of all satisfied conditions , hence if we embed these solutions in software we are sure it will work efficiently as we have also encountered environmental constraints and system attributes.

For the future work, we can make this framework stronger by defining strong sets of rules on basis of analysis and surveys. Hence providing the strong backbone to this framework by enhancing the rule database of this approach.

References

1. Firesmith ,D. G., “Engineering Security Requirements”, Journal of object technology, 2003, Vol2, no.1, pages 53-68.
2. Johnson ,J., Chaos: “The Dollar Drain of IT project Failures”. Application Development,

   , January 1995, pp.41-47
3. Common criteria for information technology security evaluation. Technical report,

   CCIMB-99–031, Common Criteria Implementation Board, 1999.
4. 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.
5. Gupta D., Agarwal A., "Guidelines and case study for eliciting Security Requirements",

   Proceedings of the 2nd National Conference on Computing for Nation Development Emerging Trends in Engineering and Technology, Nagpur, pg – 445 – 448, Feb 2008 .
6. Dermott, J.,M., Fox,C. , “Using abuse case models for security requirements analysis.”Department of Computer Science, James Madison University, 1999.
7. Meier, J.,D., Mackman, A. , Wastell, B., Bansode, P. , Gopalan, K. , “Patterns & Practices Security Engineering Index”, Microsoft Corporation http://msdn.microsoft.com/ -en-us/library/ff648032.aspx, 2005
8. Sommerville , I. , “Software Engineering”. . ISBN - 8129708671. Pearson Education. Seventh edition 2003

9. Kotonya G., Sommerville I., “Requirement Engineering with view points”, 1995.
10. MITRE Corporation “ Common Weakness Enumeration “ , http://cwe.mitre.org (2007)
11. Ronda R. Henning ,”Use of the zachman architecture for security engineering” ,Harris Corporation , Information Systems Division .
12. Boehm, B., W. , Papaccio, P., N. “Understanding and Controlling Software Costs. IEEE Transactions on Software Engineering 14, 10 (October 1988): 1462-1477.
13. McConnell, S., “From the Editor - An Ounce of Prevention.” IEEE Software 18, 3 (May 2001): 5-7.
14. Jones, C., . Tutorial: Programming Productivity: Issues for the Eighties, 2nd Ed. Los Angeles: IEEE Computer Society Press, 1986.
15. Martin ,G. J. , Tøndel, I. A., “Covering Your Assets in Software Engineering”, IEEE, 2008
16. Ahl, V. "An Experimental Comparison of Five Prioritization Methods." Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden, 2005.
17. Brackett, J. W. Software Requirements (SEI-CM-19-1.2, ADA235642). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.
18. Karlsson, J. "Towards a Strategy for Software Requirements Selection. Licentiate." Thesis 513, Linköping University, October 1995.
19. Beck, K. ,Andres, C. Extreme Programming Explained: Embrace Change, 2nd ed. Boston, MA: Addison-Wesley, 2004.
20. Leffingwell, D. & Widrig, D., Managing Software Requirements: A Use Case Approach, 2nd ed. Boston, MA: Addison-Wesley, 2003.

21. Boehm, B. ,Ross, R. "Theory-W Software Project Management: Principles and Examples." IEEE Transactions on Software Engineering 15, 4 (July 1989): 902-916.
22. Davis, A. "The Art of Requirements Triage." IEEE Computer 36, 3 (March 2003): 42-49.
23. Davis, A. Just Enough Requirements Management: Where Software Development Meets Marketing. New York: Dorset House, 2005 (ISBN 0-932633-64-1).
24. Wiegers, K. E. Software Requirements, 2nd ed. Redmond, WA: Microsoft Press, 2003.
25. Moisiadis, F. "Prioritising Scenario Evolution." International Conference on Requirements Engineering (ICRE 2000). June 2000.
26. Moisiadis, F. "A Requirements Prioritisation Tool." 6th Australian Workshop on Requirements Engineering (AWRE 2001). Sydney, Australia, November 2001.
27. Swiderski ,F., Snyder, W., "Threat Modeling." Microsoft Press., 2004
28. Robert J. Ellison, "Attack Trees" Software Engineering Institute, Carnegie MellonUniversity, 2005.
29. Dermott, J.,M. , Fox, C., "Using abuse case models for security requirements analysis." Department of Computer Science, James Madison University, 1999.
30. Alexander ,F. , "Modeling 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.
31. Alexander ,F. , "Misuse cases, use cases with hostile intent". IEEE Software, 2003
32. Firesmith, D., G., "Security Use cases", Journal of object technology, 2003, vol 2, no.3, pages 53-64.
33. Ware, M., Bowles, J., Eastman, C., "Using the common criteria to Elicit security Requirements with use cases", 2006 IEEE.

34. Hawkins,I.,“Risk Analysis Techniques”, http://www.euclidresearch.com/current.html, 1998
35. Yorder, J., Bercalow, “Architectural Patterns for Enabling Application Security”, PLoP ‘97
36. Usher, A. , “Towards a Taxonomy of Information Assurance”.
37. Schneier, B.” Beyond Fear” Heidelberg, Germany: Springer-Verlag, 2006.