Abstract
Now a day’s people are more concerned about software security. Millions of dollars are spent on developing the software, but if it doesn’t have proper security then it is a big failure. For this reason security within the project has become the most important in recent days. So far there are so many techniques available for security requirement elicitation, but there is a short of completeness, i.e. each one is having their own advantage and disadvantages. In this paper, we have studied 15 security requirement elicitation methods such as Abuse cases, Misuse cases, SQUARE, OCTAVE, CLASP, etc. and each method is analyzed against different attributes such as their applicability, ease of learn and ease of use. This paper presents an overview of security requirements, different methods for identifying security requirements, how they are classified their merits and demerits.

Keywords : NFR, Security, Security requirements, Requirements elicitation

Introduction
Today, we are living in the world of computerization where security for software has become most important. Security concerns about the protection of data from unauthorized access by the attackers. Now-a-days it has been observed that some of the software systems have been failed due to lack of security constraints [1][2] during the development of software. In recent years, many people stressed about incorporation of security goals during the software engineering process. So, it became common to enumerate, analyze and model security requirements [2] in conjunction with the construction of major software systems. Generally, the requirements collection phase plays a critical role [30] in the success of any software project. So, right from the requirement analysis phase, it is better to start with specification of security goals based on the context of the system.

There were so many methods proposed for collecting security requirements so far. In this paper we have classified those methods based on the technique they have used for collecting security requirements. Some techniques can be applied in the requirements collection phase [32] and some techniques can be applied in the design phase. But most of the techniques are applicable in the requirements collection phase only.

This paper is prepared as follows: Section 2 describes the definition of security requirements and taxonomy of security requirements. Section 3 presents the classification of security requirement elicitation methods along with their merits and demerits. Section 4 illustrates comparison and analysis of SRE methods and section 5 concludes the paper.

Definition And Taxonomy Of Security Requirements
The basic purpose of eliciting security requirements is to protect the software system. A Software system [1] means along with the software it includes an operating environment, in which the software runs, the physical environment, in which the system exists, the people interacting with the system directly or indirectly, and other systems. For protecting such system security requirements need to be defined. In terms of the dictionary, software security can be defined as “The fortification of data to make sure that only approved personnel have right to use computer files”. In software development process security is defined as “constraints on the functions of a system”[1][2]. By this definition, it is clear that security is a non functional requirement (NFR). The problem with non functional requirements [31] is there are no direct methods to predict, to measure, and to estimate whatever the NFR is. Taxonomy of security requirements is important to understand clearly about the security that needs to provide for the software at different levels of its use. Security requirements are classified as follows:

a) Data access requirements: Data accessing [20] is one of the most important activities which include storing and retrieving the data. We need to provide security for that data

b) Application interaction requirements: Interaction can be both internal and external. External interaction [20] means interacting with users and/or with other systems. Internal interaction means interaction with in the components of a system, the security need to be provided for that interaction

c) Recognition requirements: The system must recognize the users and other systems [22] before communicate with them.

d) Invasion requirements: The system shall detect and record modification by the illegal parties.

e) Validation requirements: A Validation requirement is a security requirement that states the system shall confirm the identity of its users.

f) Consent requirements: A Consent requirement states the usage and access rights of valid entities.

g) Veracity requirements: Veracity requirement states that the system shall make sure that its data and interactions are not deliberately modified.

h) Resistance Requirements: Resistance requirement states that the system shall guard itself from being corrupted by illegal entities.

i) Seclusion Requirements: Seclusion requirement states that the system shall keep its sensitive data and interactions private from unauthorized entities.
j) **Survivability requirements:** Survivability requirement states that the system shall outlive the intended ruin of a component.

k) **Physical protection requirements:** Physical protection requirement states that the system shall defend itself from physical attack.

l) **Maintenance security requirements:** Maintenance security requirement states that the system shall stop unauthorized alterations from unintentionally defeating its security methods.

The Classification of Security Requirement

Elicitation Methods

After the survey, we classified security requirement elicitation methods into different categories based on the technique they have used. The techniques used in the methods are: Use case modeling, Tree modeling, Pattern technique, Agile and Risk analysis based techniques.

Use Case Modeling Techniques

**Abuse Cases:** According to John McDermott and Chris Fox an abuse case describes communication among an actor and the system that marks in damage to any actor, stakeholder or the system itself. An abuse case model [4] is one of the capable techniques for collecting and analyzing the security requirements for object oriented software projects.

Steps in building an abuse case model are:
1. Actor identification
2. Abuse case identification
3. Describe abuse cases
4. Verify granularity.
5. Verify completeness and minimality

**Merits:**
1. It is easy to learn and easy to use.
2. This technique is very efficient for capturing security requirements.

**Demerits:**
1. It can be applicable to only object oriented software projects.
2. Not a replacement for security models involved with mathematics.
3. This model does not consider relations between use and abuse

**Security Use Cases:** Since the last decade, use cases have developed to be one of the promising techniques for modeling functional requirements. Donald G. Firesmith describes Security Use Cases [7] are derived like normal use cases, and they are used to organize, analyze, and specify security requirements. Security Use Cases need a number of distracted security facilities with security constraints. When security constraints are fulfilled, they will broaden the desired functions of a system.

**Merits:**
1. Security use cases are more suitable for large systems.
2. For defining conflicts Security use cases are relying on scientific and accurate ways.

**Demerits:**
1. Not much suitable for small systems.
2. Must be used with at most care by avoiding unnecessary architectural and design constraints.

**Misuse Cases:** Guttorm Sindre Andrees and L. Opdahls Misuse Cases are an expansion of Use Cases. Misuse Cases [10][21] were based on principles of Use cases. Use Cases were mainly used for examining of functional requirements. Use cases were unsuccessful in eliciting nonfunctional requirements whereas Misuse Cases are successful in exploring non functional requirements because they are well suitable in inspecting deviations and hazards.

Steps in creating Misuse Cases:
1. Defining actors and use cases in a usual way.
2. Identification of misuse cases along with mis-actors.
3. Identification of “includes” relations between Use cases and Misuse Cases.
4. For detecting or preventing Misuse Cases create new use cases
5. Use detailed requirements document for continuing the process.

**Merits:**
1. Misuse Cases are useful in identifying exceptions, generating test cases
2. Useful in outline and organize the requirement specification
3. Misuse cases are reusable

**Demerits:**
1. A Use case/Misuse case is informal so no clear semantics and no formal analysis.
2. No knowledge on how to write good quality misuse cases.
3. This method is not suitable for all kinds of threats.

Tree Modeling Techniques

**Fault Trees:** Phillip J.Brooke and Richard F Paige’s Fault tree [6] is an efficient method of analyzing system breakdown, from that analysis one can elicit security requirements. It examines the entire system. For ease of understanding it provides graphical symbols and it focuses on critical areas with mathematical tools. It uses Boolean logic to combine a series of lower-level events.

Steps in building Fault tree are:
1. An unwanted event is defined
2. The event is analyzed to find out immediate causes
3. Based on analysis identify the basic causes
4. A logical diagram needs to be constructed for representing relationships among the logical events

**Merits:**
1. Investigate potential faults, its modes and causes.
2. Logical fault paths can be identified from a specific effect, to prime causes.
Demerits:
1. In some cases they lead to very large trees.
2. Fault trees are developed based on skill of the analyst.
3. Fault Trees require more time and effort.

**Attack Trees:** Andrew P. Moore Attack trees describe the ways in which a system can be attacked. Attack trees [18] are conceptual diagrams in which nodes represent attacks and they represent the way an asset might be attacked. Attack trees describe security of a system systematically based on different attacks. They present a method to sense the security, to detain and reclaim knowledge about security, and to retort to modifications in security.

Steps in building Attack Tree:
1. Discover possible attack goals and build a tree for each goal.
2. With respect to every goal identify all attacks.
3. Add attacks to the tree and repeat this along the branches and nodes

Merits:
1. Attack trees follow a systematic procedure for analyzing security.
2. Attack trees capture knowledge in a reusable form.

Demerits:
1. Attack trees mainly rely on knowledge of the attacker.
2. Attack trees are difficult for applying large scale projects.

**Anti-models:** Axel van Lamsweerde Anti-model [28] is a methodology for elaborating security requirements. Anti model builds two different models jointly, in which the first one describes software, its environment and everything related to it. The second one is extracted from the first model which is the real Anti-model. It shows how the stipulation of model elements could be spitefully endangered, why and by whom.

Steps in building Anti Models:
1. Find out anti-goals by negating appropriate secrecy, seclusion, veracity and accessibility.
2. For every anti-goal, identify the potential attacker.
3. Identify attacker class for an every anti-goal.
4. Elaborate an anti-goal AND/OR graph.
5. Using anti-goal specification, derive the agent anti-models and objects.
6. Identify anti-requirements (i.e. Capabilities of attackers) from an AND/OR graph.

Merits:
1. Applicable to both traditional and modern approaches.

Pattern Based Techniques: 

**Security Patterns:** Nobukazu Yoshioka’s security patterns [3] are like design patterns. They are used to depict the security problems of software in different contexts and provide constructive predefined solution. The security pattern consists of identifying business resources and factors with the security influence, the relation between resources and business factors and the security nature. The template of security pattern includes different fields like application, behavior, conditions, consequences, supporting principles and other related security patterns.

Merits:
1. Novices can perform as security experts.
2. Problems can be identified and solved efficiently.
3. Different security patterns can be created for different phases of SDLC.

Demerits:
1. The common formats of patterns are not fixed enough to detain security requirements.
2. In the profitable environment it is difficult to structure the data in to hierarchical sensitive levels.
3. The selection of a specific pattern format for a specific target security requirement is difficult.

**Problem Frames:** Denis Hatebur’s Problem frames [8] are used for describing software development problems. It is one type of pattern. Frame diagrams [6] are used for describing problem frames. The problem frames wrap software development problems which are pretty general in nature and they are more useful for developers in identifying and analyzing the security problems.

Merits:
1. A problem frame is coupled with other methods for identifying the problem in full detail and develops a solution.
2. Applicable to both traditional and modern approaches.

Demerits:
1. Problem Frames are difficult to learn and use.
2. Problem frames require some mathematical background to understand.

**Risk Analysis Based Techniques**

**Security Design Analysis (SeDAn):** Howard Robert Chivers Security Design Analysis is intended for analyzing the risk and designing security for large distributed systems. The inspiration behind SeDAn [29] is to know the logical relation between system-level risks, and security requirements at the component level. This is because for conveying information to the security design team and development team with a risk-context for security mechanisms.

The main components of the framework are:
2. Probable attackers and their goals.
3. The description of the system.
4. Asset Concerns.
5. The Protection Strategy, and

**Merits:**
1. It measures the amount of risk for familiar security problems.
2. Service constraints can be specified easily.

**Demerits:**
1. Data produced by SeDAn is not much compatible with UML representation.
2. Dynamic risks can be difficult to handle.

**Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE):** Richard A, et.al OCTAVE is a framework [17] that shall be useful for the organizations in understanding, assessing and addressing the security risks. Initially, it was urbanized by the Software Engineering Institute for the US Department of Defense. It is a well-organized methodology for identifying, prioritizing and managing information security risks.

OCTAVE defines three phases:
1. Building Threat Profiles based on assets.
2. Identifying Infrastructure Vulnerabilities.

**Merits:**
1. It can be applicable for both small and large organizations.

**Demerits:**
2. It is very complex to learn and use.
3. It doesn’t produce in depth quantitative analysis of security exposure.

**Security Quality Requirements Engineering (SQUARE):** Nancy Meads SQUARE process [25] helps in eliciting, categorizing, and prioritizing security requirements for information technology systems and applications. This model is useful in incorporating security concepts into the early phases of software development. This method performs all the activities of SREP i.e. elicitation, analysis, prioritization and documentation of security requirements.

SQUARE is a nine-step process, they are:
1. Be in agreement with the definitions.
2. The security goal identification.
3. Artifact development for supporting security requirement definition.
4. Assessing the risks.
5. Elicitation technique(s) selection.
7. Requirements categorization.
8. Requirements prioritization.
9. Requirements inspection.

**Merits:**
1. The SQUARE is easy to learn and use.
2. More expected schedules and, and low cost.

**Demerits:**
1. This method is more suitable to large scale projects only.

**Threat Modeling for Security:** Suvda Myagmar Adam J. Lee William Yurcik Threat modelling [27] is a method of conniving a security specification and then finally testing that specification. It is more suitable for conducting threat modelling at the application design time. It is formal technique and it helps in identifying and rating threats systematically. Threat modeling follows a controlled approach for addressing the top threats that affects the application or system first.

Steps in threat modelling are:
1. Assets identification.
2. Application decomposition.
3. Threats identification.
4. Documenting the threats, and
5. Rating the threats.

**Merits:**
1. It is a well-organized approach that justifies security efforts.
2. Threat modeling supports all the activities of security requirements engineering process

**Demerits:**
1. All threats cannot be identified except the code is easy.

**Agile-Based techniques**

**Abuser Stories:** Johan Peeters Agile processes [13] have been considered inappropriate for security sensitive software development due to the trivial and informal nature of agile processes. Abuser stories are used for capturing vulnerabilities in software systems. Abuser stories are written like User stories, but User stories are written based on users thought process whereas abuser stories are written based on the attackers thought process. Abuser stories describe the attacker’s intention and impetus. The activities described by Abuser stories must be barren or mitigated by the application.

**Merits:**
1. This technique helps us in understanding how the system behaves under attack or failure conditions.
2. Weaknesses or gaps that are needed to be addressed can be documented.

**Demerits:**
1. This technique is applicable for the agile process only.
2. This simple technique isn’t enough to make a system secure.

**Comprehensive Lightweight Application Security Process (CLASP):** John Viega’s CLASP is an ordered and planned methodology for placing security concerns into the early
phases of software development. CLASP provides the finest practices for security processes for an active application or for a new application. The CLASP [20] is a collection of different activities that can be included in any software development process.

Steps in CLASP framework are:
1. Identification of resources and system roles.
2. Categorization of resources into abstractions.
3. Identifying the interactions of resource, and
4. Specify techniques to address core security services for every category.

Demerits:
1. It is not suitable for traditional software development.

Usage-centric Security Requirements Engineering (USeR):
Niklas Hallberg describes USeR [26] is a methodology for elicitation, analysis and prioritization of Security Requirements. It is more suitable for small organizations. This method is relying on part of speech (POS) tagging [19] for identifying security goals. This method integrates quality tools into requirements engineering to extract security requirements from software requirements.

Steps in USeR approach are:
1. Identification of Security associated statements.
2. Determination of Security needs.
3. Determination of Security requirements.
5. Exploring design implications.

Merits:
1. This method can be simply included into existing development processes.

Demerits:
1. This method is more suitable for small organizations only.
2. Misuse Cases are needed for visualizing requirements.

Comparison and Analysis
In comparison among the methods, we have concentrated on the techniques they have used for security requirement elicitation and those methods are analyzed against different attributes like applicability to traditional and modern approaches, suitability to small scale and large scale projects and ease of learn and use. Different methods are relying on different techniques like Use case modeling, Tree modeling, Pattern matching, Risk analysis and Agile-based techniques. After survey and analysis our observation is different methods are suffering with different problems and no method is showing complete satisfaction in all aspects.

John McDermott and Chris Fox describe abuse cases [4] are easy to learn and use but applicable to object oriented projects only. Donald G. Firesmith security use cases [7] are not suitable to small scale projects. Guttom Sindre. Andreas and L.Opdahls Misuse cases [10] are suffering with clear semantics due to that formal analysis is not possible. Fault trees [6] and Attack Trees [18] are useful in systematic analysis of system failure, but they mainly dependent on skill of the analyst. Axel van Lamsweerde’s Anti Model [28] also uses Tree modeling technique, but it is applicable at the application layer only. The following Table 1 shows how the different security requirement elicitation methods affect security requirements engineering process.

Table 1: Different SRE methods and their affect on SREP.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Abuse Cases</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Misuse Cases</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Security Use Cases</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Fault Trees</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Attack Trees</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Anti-models</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Problem Frames</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Security Patterns</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>9</td>
<td>Threat Modeling</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>10</td>
<td>SQUARE</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SeDAn</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>OCTAVE</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>USeR</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CLASP</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Abuser Stories</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

1. : Security requirement elicitation
2. : Security requirement analysis
3. : Security requirement prioritization
4. : Security requirement documentation

The following table shows the different security requirement elicitation methods and their applicability.

Table-2: Different SRE methods and their applicability

<table>
<thead>
<tr>
<th>S.No</th>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Abuse Cases</td>
<td></td>
<td>×</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Misuse Cases</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Security Use Cases</td>
<td></td>
<td>×</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Fault Trees</td>
<td></td>
<td>×</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Attack Trees</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Anti-models</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Problem Frames</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Security Patterns</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Threat Modeling</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SQUARE</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SeDAn</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>OCTAVE</td>
<td></td>
<td>√</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>USeR</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CLASP</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Abuser Stories</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

68
1. Suitable to Large Scale Projects
2. Suitable to Small Scale Projects
3. Suitable to Traditional approach
4. Suitable to Modern approach
5. Easy to Learn
6. Easy to Use

Nobukazu Yoshioka, et al. Security patterns [3] provide constructive predefined solutions for security problems but difficult for selecting the specific format pattern for specific security requirement. Denis Hatebur, et al problem frames [8] are useful in SRE but difficult to apply as they require mathematical knowledge. Howard Robert Chivers SeDan [29] is difficult in handling dynamic risks. Richard A, et al OCTAVE [17] is also a risk based technique; it can be applicable to small and large scale projects, but difficult to learn and use. Nancy Meads SQUARE [25] is observed as the best technique, but it is more suitable to large scale projects only. Suvda Myagmar Threat modeling [27] is only applicable at the design phase of SDLC. Abuser stories [13], CLASP [20] and UseR [26] are agile-based techniques they cannot be applicable to traditional software development.

Conclusion
This paper presents the description and taxonomy of security requirements. We have surveyed 15 methods for security requirement elicitation; based on that understandability we classified the techniques in to different categories along with their merits and demerits. These 15 methods are compared and analyzed for their applicability and those results are presented. Even though, the number of methods available for security requirement elicitation, no method is perfect in all aspects for the application of software development. So, a method is required, that is easy to learn, easy to use, applicable to small and large scale projects, applicable to projects developed by both traditional and modern approaches.

References
[5] Paco hope, gary mcgraw digtal annie i.Anto’n, Misuse and abuse cases: Getting past the positive, the ieee computer society, 540-7993/04© 2004 IEEE.


[24] Howard Chivers, Information Modeling for Automated Risk Analysis, Department of Information Systems, Cranfield University, Defence Academy of the United Kingdom, Shrivenham, Swindon, SN6 8LA, UK.


