Int. J. Inf. Secur.
DOI 10.1007/s10207-015-0305-x

REGULAR CONTRIBUTION

Effectiveness and performance analysis of model-oriented security requirements engineering to elicit security requirements: a systematic solution for developing secure software systems

P. Salini[1] · S. Kanmani[2]

Abstract Software systems are becoming more and more critical in every domain of human society. These systems are used not only by corporates and governments, but also by individuals and across networks of organizations. The wide use of software systems has resulted in the need to contain a large amount of critical information and processes, which certainly need to remain secure. As a consequence, it is important to ensure that the systems are secure by considering security requirements at the early phases of software development life cycle. In this paper, we propose to consider security requirements as functional requirements and apply model-oriented security requirements engineering framework as a systematic solution to elicit security requirements for e-governance software systems. As the result, high level of security can be achieved by more coverage of assets and threats, and identifying more traces of vulnerabilities in the early stages of requirements engineering. This in turn will help to elicit effective security requirements as countermeasures with business requirements.

Keywords Assets · Security requirements · Security requirements engineering · Software systems · Threats · Vulnerabilities

✉ P. Salini
salini@pec.edu

[1] Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry, India

[2] Department of Information Technology, Pondicherry Engineering College, Puducherry, India

1 Introduction

To develop secure software systems, it is essential to capture the security requirements with business requirements, but traditionally, one of the most ignored aspects of Software Development Life Cycle (SDLC) is the Security Requirements Engineering (SRE) process. Many researchers are working on SRE area; however, there is a lack in security requirements elicitation and specification process. The primary reason for this is that security is assumed to be a technical issue and therefore best handled during the later phases of SDLC.

As the idea of incorporating security into software from the very beginning of development has gained acceptance, various SRE methods were suggested. However, software is still being developed by following the conventional SDLC models or processes, and security analysis is done in the implementation phase as a reason to fit the developers.

The common problem is that when security requirements are specified they tend to be accidentally replaced with security-specific architectural constraints which may unnecessarily constrain the security team from using the most appropriate security mechanisms for meeting the true underlying security requirements. During the software development, since the analysis is on security mechanism rather than security requirements, it leads to poor security requirements specification and security flaws. Thus, these issues motivated us to propose and design a simple and usable framework Model-Oriented Security Requirements Engineering (MOSRE) to elicit and analyze the security requirements for software systems.

The main contribution of this paper is to compare the effectiveness and performance of MOSRE framework with the existing SRE methods, by identifying and specifying the security requirements of a web application case study Elec-

Published online: 21 November 2015

tronic voting (e-voting) system, which is a security critical software system. For example, assume that an electronic vote is discovered of being tampered by a attacker of a country. This fraudulent act will not only have drastic consequences for the country itself, but will also have enormous consequences for the whole world. So, the highest achievable security is never too much for an e-voting system. E-voting systems play a critical role in today's democratic societies [1,21], as they are responsible for recording and counting the votes. There are a number of reports describing the malfunctioning of these systems, suggesting that their quality is not up to the task. The e-governance initiatives take steps to ensure its elections via e-voting are secure through a cost/benefit analysis.

To build an e-voting system, tasks such as security requirements elicitation, specification, and security requirements validation are essential to assure the security of the resulting e-voting system. So the proposed MOSRE, a SRE framework, is applied to identify security requirements of an e-voting system. Security is an essential part of e-voting, so not only the technical security and the data security are important but also the security of procedures and personnel. For example, many things can go wrong if polling station officials are required to install software on a computer. In addition, threats due to viruses on computers of persons who vote from home are the challenges to be considered.

The proposal is to identify, categorize, and prioritize the list of assets [2], threats, and vulnerabilities first and then identify security requirements at the application, host, database, and network levels in the earlier phases of SDLC of software system (e-voting). This will help to design and build an e-voting system which is less prone to vulnerabilities and threats.

Security Requirements Engineering (SRE) is not only a critical process, but also counterproductive if security requirements are not analyzed and specified in the early stage of software development. Therefore, at the early stage of the SDLC, good requirements engineering is essential to elicit security requirements similar to business requirements. Thus, MOSRE will ensure the user system functionalities and security, and minimize the vulnerabilities of the software systems.

2 Related works

The Secure Software Development Life Cycle (SSDLC) process which is in practice in the industry for software systems is a multi-stage process consisting of design, coding, testing, and deployment as shown in Fig. 1.

In SSDLC, the security aspects are considered only in these stages and the developers need to solve the security issues. At times, the security analyses are not formally carried out by them which results in poor security mechanisms. Thus,

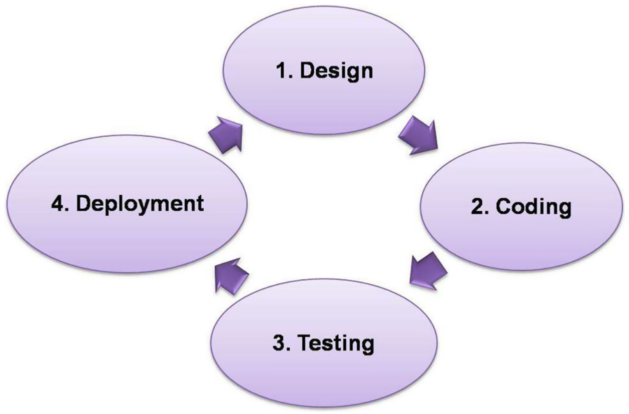

Fig. 1 Secure software development life cycle

the security of the whole system is compromised and it is also extremely hard and expensive to rectify the defects at later phases of software development. This is because of skipping security in requirements engineering phase and considering security requirements as non-functional requirements.

There is also a tremendous increase in not only the number of attacks but also vulnerabilities through which attacks can be performed on software systems. In order to protect systems from attacks exploited through vulnerabilities, attention must be given to its security requirements. So, the security should be integrated into all phases of the SDLC such as requirements, design, implementation, and testing. Hence, the SRE phase as depicted in Fig. 2 should be included in the Requirements Engineering (RE) phase of SSDLC for early identification and analysis of security requirements. Realizing security early in the RE phase is important so that security problems can be tackled before going further in the process and rework can be avoided [3].

Figure 2 shows the important activities need to be performed in each phase for developing secure software systems. In this section, we discuss the related works in security requirements engineering, and the e-voting system and e-governance in subsections.

The requirements engineers extract, analyze, specify, and manage quality requirements such as interoperability, availability, performance, reliability, portability, and usability, but many are at a loss when it comes to security requirements. Most of requirements engineer are poorly trained to extract, analyze, and specify security requirements and often confuse security requirements with the architectural security mechanisms that are traditionally used to solve security issues.

Thus, they end up specifying architecture and design constraints rather than true security requirements. The few trained requirements engineers are given only an overview of the security mechanisms such as passwords and encryption rather than actual security requirements. This motivated research in security requirements engineering, and many

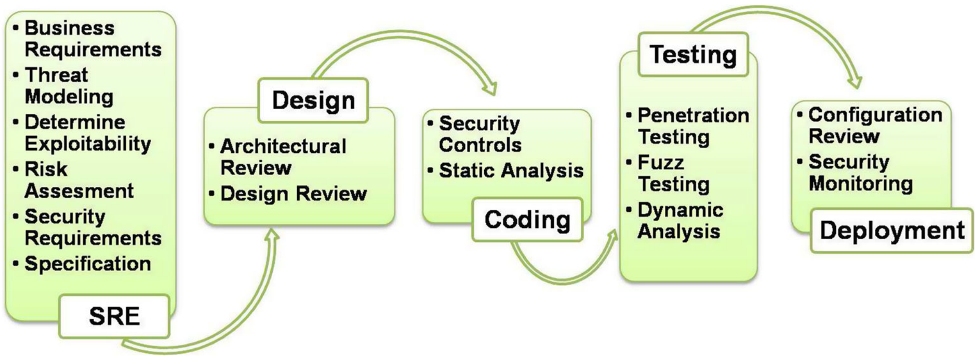

Fig. 2 Important activities in each phase of SSDLC

SRE methods, framework and process are proposed in the literature with varying characteristics.

2.1 Security requirements engineering methods

Many SRE processes have been proposed, and some of the methods are McGraw's secure SDLC process [4], trustworthy computing security development life cycle or Microsoft SDLC [5], Comprehensive, Lightweight Application Security Process (CLASP) [6], Security Quality Requirements Engineering (SQUARE) [7], Security Requirements Engineering Framework (SREF) [8], Security Requirements Engineering Process (SREP) [9] and secure tropos [10]. A comparison of security requirements engineering methods is given in detail in [31].

The SRE methods can be grouped under three categories:

(i) Methods applied not only to RE phase but to the entire SDLC
The Tropos material by Giorgini et al. [11] method is a self-contained life cycle approach. If anyone was using Tropos for software development, then one would use it throughout the development cycle. CLASP is a life cycle process that suggests a number of activities across the development life cycle in order to improve security. It has the Core Artifacts approach, it is not inconsistent with SQUARE, the goals and some of the process steps are similar, but it has a different process to arrive at security requirements.

(ii) Processes aimed for SRE phase
SQUARE is aimed at only SRE phase, and it is useful to asses the quality of the identified security requirements, but it lacks in identifying assets. The SREP is a quite partially similar method to SQUARE and meant for SRE phase but incorporates consideration of the common criteria and notions of reuse. SREF is also for RE phase, but the artifacts suggested are probably too complex for regular developers [12]. SREP and SREF are two existing system SRE methods considered in this paper for comparing with MOSRE framework.

(iii) Methods that could be applied within SRE processes
Fernandez's misuse cases and attack patterns [13] are consistent with SQUARE, since they are part of the SQUARE process. However, there is less detail on how to use these specifically in the requirements area than the SQUARE process provides. Weiss's security patterns [38] are also consistent with SQUARE and can be used as part of the process. The security patterns could be used to help, to identify, and document security requirements. The security patterns developed by Rosado et al. [14] fall into the architecture domain and would be most useful once security requirements are identified.

In order to compare the effectiveness and performance of MOSRE, we have implemented the existing Haley and his colleagues framework—SREF—-to the elicit security requirements of an e-voting system. Haley et al. [8] describe an iterative process consisting of four stages that integrate RE and SRE. Iteration between requirement and design activities is an important part of the SREF. Fulfilling a security requirement might lead to new assets, thus resulting in new security requirements. The four stages are given below:

1. Identify functional requirements.
2. Identify security goals.
In this stage, three activities are done to identify the security goals which are as follows:
   (a) Identify candidate assets:
   Identify anything that has a value to the organization.
   (b) Generate threat descriptions:
   Security goals can be found by connecting the CIA concerns to the assets, which can be violated by certain actions to cause harm. Then applying prevention to the resulting threat descriptions leads to the security goals.

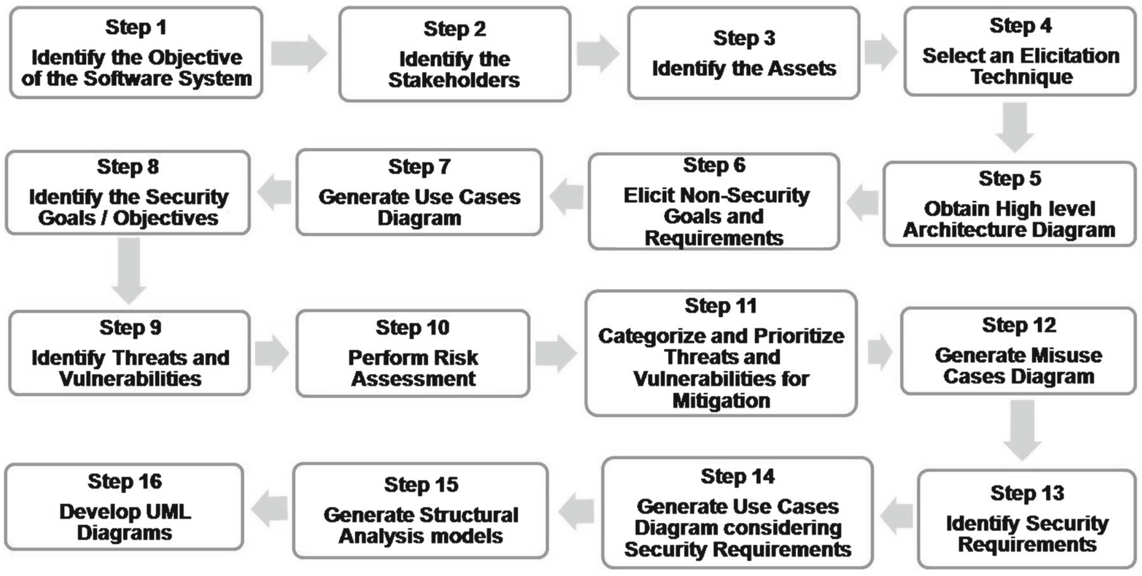
Step 1
Identify the Objective of the Software System
Step 2
Identify the Stakeholders
Step 3
Identify the Assets
Step 4
Select an Elicitation Technique
Step 5
Obtain High level Architecture Diagram
Step 6
Elicit Non-Security Goals and Requirements
Step 7
Generate Use Cases Diagram
Step 8
Identify the Security Goals / Objectives
Step 9
Identify Threats and Vulnerabilities
Step 10
Perform Risk Assessment
Step 11
Categorize and Prioritize Threats and Vulnerabilities for Mitigation
Step 12
Generate Misuse Cases Diagram
Step 13
Identify Security Requirements
Step 14
Generate Use Cases Diagram considering Security Requirements
Step 15
Generate Structural Analysis models
Step 16
Develop UML Diagrams

Fig. 3 MOSRE framework: a 16 step process

   (c) Apply management principles:
   Principles can be separation of duties, separation of function, etc.

3. Identify security requirements.
   Security requirements are constraints on functions of the system, where these constraints operationalize one or more security goals. They draw problem diagrams to demonstrate the functional requirements and to support capturing the security requirements in terms of constraints on the functions. The security requirements are denoted textually.
4. Construct satisfaction arguments.
   They verify that the system can satisfy the security requirements specified in the early stage of software development. Therefore, at the early stage of the SDLC, good requirements engineering is essential to elicit non-functional requirements and functional requirements such as business and security requirements. This will ensure the user system functionalities and security by minimizing the vulnerabilities of the software applications.

Haley et al. consider security requirements elicitation and analysis, using their security engineering framework. It covers only confidentiality, integrity, availability, and accountability goals. They evaluated the framework by applying it to a security requirements analysis within an air traffic control technology evaluation project. The framework for SRE does not consider completeness of the set of requirements, conflicting requirements, or interaction between security and other non-functional requirements.

The limitations with the existing methods such as SREP and SREF have motivated us to propose and design Model-Based Security Requirements Engineering Framework (MSREF) [32] and same was applied for online trading system [33]. Since they lack in identifying effective number of security requirements, we improved and proposed MOSRE for web applications [34,35]. Our proposed framework MOSRE for developing secure software systems has 16 steps as depicted in Fig. 3.

MOSRE was designed in order to improve the completeness, consistency, correctness, and traceability of security requirements specification. The MOSRE has six tasks of RE, namely: inception, elicitation, elaboration, negotiation, validation, and specification. MOSRE has been applied to e-voting system in [34], and this paper extends with the empirical analysis and discussion on the experimental results obtained by comparing with the existing SRE methods.

2.2 E-voting system and e-governance

E-voting is currently being employed in some countries, including the Netherlands, Belgium, and Brazil, and the trend is expected to proliferate as the part of e-governance. This e-voting method takes a form of visiting a voting place in person and voting through a computer or terminal installed at the location after which the voting result is sent to a tallying place for computerized tallying. Voters go to the polling stations and submit their ballot electronically using the voting terminal [15,16] and [17]. These systems are highly dependent on the security and correctness of the software running on the voting terminal.

Estonia became the first country that allowed online voting and plans to implement it for elections [18]. The very first Internet voting in the USA took place in real-life situation during the Democratic Presidential Primary in Arizona. The General Affairs Office of Japan announced that a proposal for revising the election law for public offices will be submitted to the Diet in order to allow electronic voting and vote counting.

The first electronic voting was enforced in Okayama in order to select a mayor and a councilman of Nimi-city [19]. As a part of e-governance in India, the Government of Karnataka is considering the introduction of e-voting along with the general polling pattern in practice in the ensuing local bodies elections slated during 2013. Online voting is hoped to increase the polling percentage with more number of urban literates taking part in the electoral process [20].

The analysis of Diebold AccuVote-TS carried out by Kohno et al. [17] was one of the first paper in security analysis and mechanism for e-voting system. Diebold violates confidentiality and the integrity of the ballots and anonymity of voters. The security analysis of the Secure Electronic Registration and Voting Experiment (SERVE) given by Jefferson et al. [22] points many architectural and conceptual weaknesses. Thus they recommended shutting down the development of SERVE immediately. An independent security assessment of a complete voting system, including both the hardware and the software components, was done by Feldman et al. [23].

The risks and benefits of electronic voting with the current voting procedures and problems are given in [37].The responsibilities and privileges of the actors involved in e-voting is presented in [24]. The work focuses on Internet voting and provides a description of the role of each actor together with the clear indication of what each actor is expected to do with the system processes and defining an operational framework and to talk about secure-voting system. These related works concentrate on security analysis at either design or implementation phase. The existing systems use various forms of cryptographic and security protocols to ensure that these criteria are met in their implementation of an online election system. Most of the research works are on analysis of security mechanism of e-voting system and making the system vulnerable. Security of e-voting process and the state-of-the art technology into the election process with risks are given by Rubin [36].

MOSRE proposes a modeling in terms of use case and misuse case and provides a step to assess the threat and vulnerabilities for e-voting system. In [25,26], the authors discuss the need for procedural security in electronic elections. In [27,28], an approach to measure relative security attack is explained. However, our work is to analyze security requirements in the early stage of system development and consider security requirements as one of the functional requirements.

3 Eliciting security requirements for an e-voting system

In this section, a detailed discussion is given on each step of MOSRE framework, which is iterative and covers all phases of RE to elicit security requirements for an e-voting system extracted from [35] to analyze the effectiveness and performance of MOSRE with existing SRE methods. The functional requirements considering security requirements for an e-voting system are elicited and specified in the SRS as the work product of SRE phase. The client with respect to e-voting system is the election department. With the details given by the client, the requirements team sketched the workflow for the e-voting process in three parts with one or more processes as given below:

Part 1: Pre-voting

– Voters registration
– Candidates registration
– Processing candidates for eligibility

Part 2: Voting

– E-voting

Part 3: Post-voting

– Approval of votes and ballots
– Registration of paper votes
– Counting electronic and paper votes
– Declaration of results

The first part of e-voting process is pre-voting, where voters are registered and the candidates register with their details and are processed for valid candidatures. The second part is electronic voting which happens on a particular day, and the voters cast their votes. The last part is post-voting, where the votes are approved, the electronic and paper votes are counted, and the results are declared. Since the analysis of the e-voting system is only used for validating the MOSRE framework, the security requirements of first and second part of e-voting system from [35] are only considered in this paper. The work product obtained for each step of MOSRE for an e-voting system is discussed below.

3.1 Step 1: Identify the objective of the e-voting system

The main objective of the software system is to establish a secure electronic ballot vote resolution for political elections in India. The e-voting shall simplify voting and give higher accessibility than current paper-based ballot. It should guarantee economical resource usage and facilitate the put into effect of direct democracy. It should also uphold the present high level of trait at holding elections, supported on the principle of secret ballots.

An online voting application has to be developed with a high level of security. The detailed objectives of e-voting system are:

(i) to build an online system which would enable voters to cast their votes on chosen candidate,
(ii) to check and validate users logging into the voting system,
(iii) to create a database to store votes and user information on the system,
(iv) to enable the system to tally the votes casted according to candidate,
(v) to create admin to manage the election system effectively and
(vi) to display voting results, for the administrator to analyze and declare the results.

3.2 Step 2: Identify the stakeholders

Stakeholders for the e-voting system are people from the voters community, candidates, political reference group, security experts, election officers, government representatives, developers, and RE team and others who are interested in giving different views and ideas, and read the software system requirements specification.

3.3 Step 3: Identify the assets

With the objective of the e-voting system, the business assets identified are authentication data, list of candidates, voters register, registration process, registration period, vote, the right to vote, ballot, voting process, casting of a vote counting process, counting result, election report, voter's secret, the number of votes cast for each candidate, and reporting process, and the system assets are application software, database, network, server, and voter's system.

3.4 Step 4: Select an elicitation technique

There are many elicitation technique exists such as brainstorming, interview with the customers, surveys, or questionnaires, observation and focus groups. In this analysis, brainstorming was one of the techniques used for elicitation of requirements such as business, security, and non-functional requirements. The members of the brainstorming session are the stakeholders. The requirements team arranges for meeting, and all stakeholders attend the meeting at a common place. They give their ideas, suggestions, resolve conflicts, and help in categorizing and prioritizing the requirements.

3.5 Step 5: Obtain high level of architecture diagram of e-voting system

With the help of the objective of the software system, the number of tiers in the application can be identified. A rough architecture diagram with a high level of abstraction is drawn. This diagram helps to analyze the data flow and the entry points in the system. The high-level network architecture for e-voting system is depicted in Fig. 4. The systems use a three-layer architecture, where front-end servers are separated from the application and database servers.

The front-end web server receives requests from voters, admin, and reverse-proxies to the application server, which hosts election software and stores both blank and completed ballots. A database server stores voter credentials and tracks voted ballots. Multiple firewall reduces the attack and complicate attacks by disallowing outbound connections.

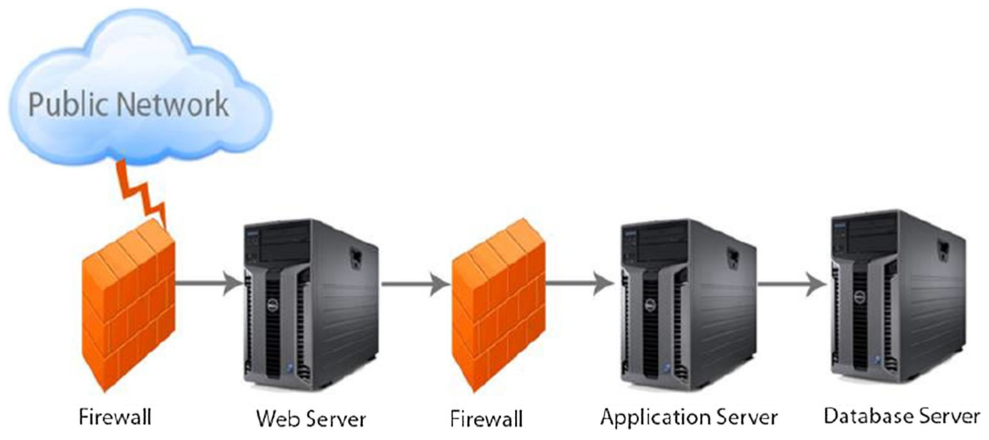

Fig. 4 High-level network architecture for e-voting system

3.6 Step 6: Elicit non-security goals and requirements

The next step is to elicit non-security goals and requirements: in each part of the e-voting system, the business goals (BG) and high-level requirements (HLR) are identified. The following are some of the sample non-security goals and requirements for the e-voting system [35].
Part 1: Pre-voting

- BG 1.1 Voters registration
  - HLR 1.1.1 Approve or reject the application
  - HLR 1.1.2 Update the electoral roll
- BG 1.2 Candidates registration
  - HLR 1.2.1 Submission of candidate detail
  - HLR 1.2.2 Verification of candidate against the electoral roll
  - HLR 1.2.3 Duplication check for candidate
  - HLR 1.2.5 Approval of candidate by representatives from party
- BG 1.3 Processing candidates for Eligibility
  - HLR 1.3.1 View candidates list
  - HLR 1.3.2 Approve/reject candidate
  - HLR 1.3.3 Edit candidates details
  - HLR 1.3.4 Publish approved party's candidates
  - HLR 1.3.5 Publish candidates list to the general public
  - HLR 1.3.6 Notification of candidature to the candidates

Part 2: Voting
- BG 2.1 E-voting
  - HLR 2.1.1 Voter's authentication
  - HLR 2.1.2 Check voter against the electoral roll
  - HLR 2.1.3 The system presents valid elections based on voter rights
  - HLR 2.1.4 Voter selects election and cast vote
  - HLR 2.1.5 System makes a mark off against the voter in the electoral roll and flags whether the vote has been cast in a controlled or uncontrolled environment
  - HLR 2.1.6 Vote is stored securely

Part 3: Post-voting
- BG 3.1 Approval of votes and ballots
  - HLR 3.1.1 Approval of votes and ballots received in a cover envelope
- BG 3.2 Registration of paper votes
  - HLR 3.2.1 Entry of paper vote by scanning of ballots
- BG 3.3 Counting electronic and paper votes
  - HLR 3.3.1 Unofficial vote tally occurs once the voting has been officially closed
  - HLR 3.3.2 Check and count total votes for each candidate
- BG 3.4 Declaration of results
  - HLR 3.4.1 Announce election results
  - HLR 3.4.2 Generate election reports

With the HLR, the low-level non-security requirements are defined. Some of the important low-level functional requirements (LLFR) and low-level non-functional requirements (LLNFR) for Parts 1 and 2 of the e-voting process are listed below:

- LLFR 1.1.1 Enter voter's details
- LLFR 1.1.2 Proof of voter identity
- LLFR 1.1.3 Submit application
- LLFR 1.1.4 Check for voter citizenship and age
- LLFR 1.1.5 Check for voter duplication
- LLFR 1.1.6 Approve the application of voter registration
- LLFR 1.1.7 Update the electoral roll of new voter
- LLFR 1.1.8 Notify voter and provide smart card
- LLFR 2.1.1 Voter listed in electoral roll
- LLFR 2.1.2 Select election
- LLFR 2.1.3 Select voting options
- LLFR 2.1.4 Cast vote
- LLFR 2.1.5 Store vote and make a mark off electoral roll
- LLFR 2.1.6 Notify voter
- LLFR 2.1.7 Store exception
- LLFR 2.1.8 Notify voter
- LLNFR 1.1.1 The system should be available for voters' registration
- LLNFR 2.1.1 Scalable for performance of e-voting in peak hours
- LLNFR 2.1.2 A voter should not receive messages about previous votes

3.7 Step 7: Generate use cases diagram

The next step is to generate use cases diagram. A sample use case diagram generated is given for Part 1 of the e-voting process in Fig. 5. It depicts the use case diagram with the functional requirements and actors.

Use case name—voters registration
Purpose—to register eligible and valid voter
Actors—voters and election system
Preconditions—the voter must be connected to the Internet via a browser, and the electoral roll should be open for updates.
Post-conditions—the voter details and identity are stored in the system.

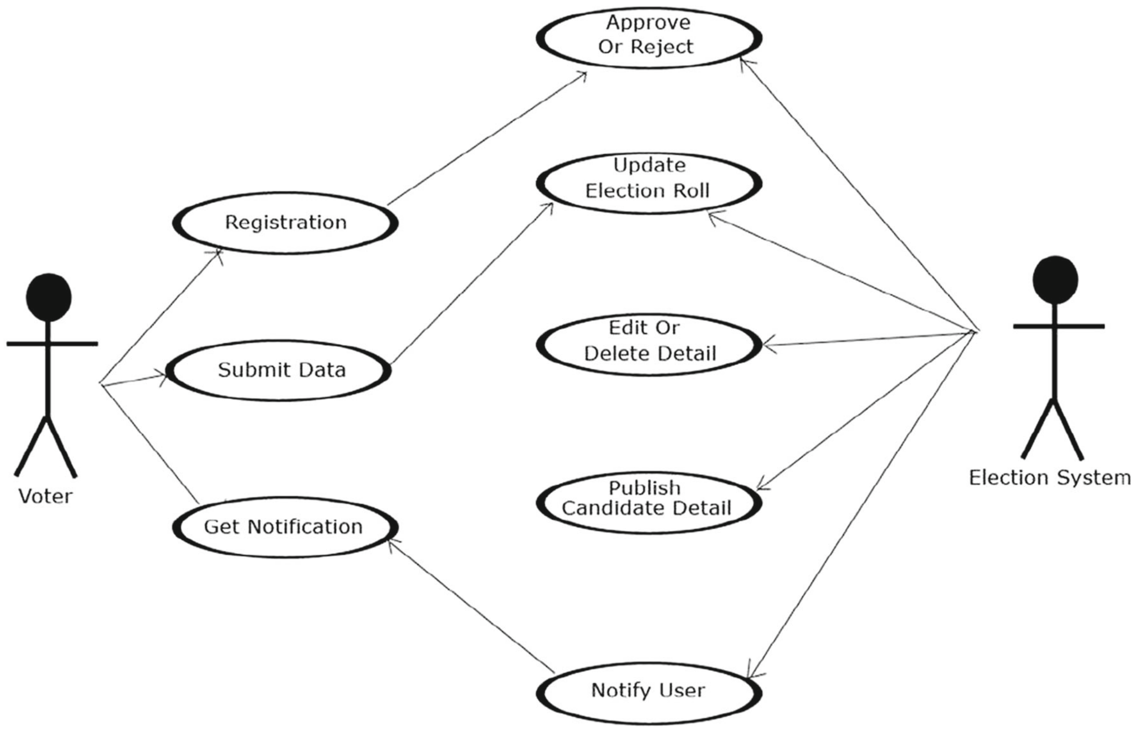

Fig. 5 Use case diagram for Part 1 of e-voting system

3.8 Step 8: Identify the security goals/security objectives

The security goals/objectives can be identified with respect to assets, business goals, and organizational principles. The following are the important and sample security objectives [35] for e-voting system to protect the security of data, system, and person who are involved in the election.

- Authenticity—ensuring that the voters identity while registering and administrator identity to accept/decline the voter.
- Confidentiality—ensuring that the voter's credentials and vote are secret.
- Anonymity—ensuring that votes must not be associated with voter identity.
- Integrity—ensuring that each voter's details and his vote are recorded as intended.
- Auditability—ensuring that the election records are reliable and demonstrably authentic.
- Disclosability—ensuring that the system and the process to be open for external inspection and auditing.
- Availability—ensuring that the system protects against accidental and malicious denial-of-service attacks.
- Uniqueness—ensuring the voter's unique identity and no voter able to vote more than once.
- Non-coercibility—ensuring that voters are not able to prove how they voted.
- Accountability—ensuring that system operations are logged and audited.
- Reliability—ensuring that the system minimizes accidental bugs and omits malicious code.
- Accuracy—ensuring that voting systems record voter's details and the votes correctly.
- Secrecy/privacy—ensuring that the identity of the candidate to whom the voter has voted.

3.9 Step 9: Identify threats and vulnerabilities

The threats to the system will be from internal and the external users such as legitimate users, system developers, system operators, hostile individuals, criminal organizations, protest groups, foreign intelligence services, terrorist organizations, and other internal users. The possible methods of electronic threats to the system are hacking, malicious software, denial-of-service, domain name service attacks, vote buying/selling and coercion, theft or forgery of election details, deliberate repudiation of transactions, and accidental damage by users, operators, equipment, and natural disasters. Some of the sample threats and vulnerabilities [35] for Parts 1 and 2 of the e-voting process are as follows:
Threats

- The voter details get corrupted by buffer overflow.
- Code injection to change the behavior of the election system at the time of registration.

Table 1 Threats and the rate of risk for e-voting system

Threats	*D*	*R*	*E*	*A*	*D*	Total
The voter detail gets corrupted by buffer overflow	3	3	2	3	3	14
Code injection to change the behavior of the election system at the time of registration	3	2	2	2	2	11
SQL injection leads theft of voters or candidates details of election system	3	3	2	3	2	13
A malware accesses to the selected voting options at the voter's PC	3	3	2	2	2	12
Authentication token from the smart card can be forged	3	3	2	3	2	13
Brute force attack against the credentials of the voter and election officer	3	3	2	3	2	13
Man-in-the-middle, voter contest modification between the electoral roll service and the authentication service	3	3	2	3	3	14

– SQL injection leads theft of voters or candidates details of election system.
– Brute force attack against the credentials of the voter and election officer.
– Session hijack is easy on unprotected sessions.
– Man-in-the-middle in selection of eligible candidates from the electoral roll after the candidates registration process.
– A malware access to the selected voting options at the voter's PC.
– Authentication token from the smart card can be forged.
– Man-in-the-middle, voter contest modification between the electoral roll service and the authentication service.
– Authentication token replay attack.
– Voter impersonation and vote casting.
– A malware modifies the client application at the voter's PC.
– Denial-of-service attack over the voting platform.
– Man-in-the-middle, ballot template modification between voter and voting servers.
– A malware modifies the voting options at the voter's PC.
– Change of vote while storing.
– Decrypted storage of votes—alteration at counting.

Vulnerabilities

– Weak or blank passwords, passwords that contain everyday words.
– Missing or weak validation at the server of the election system.
– Lack of password complexity enforcement and without proper encryption.
– Data transfer as plain-text without encryption.
– Failure to check for malicious code and validate cookie input.
– Failure to encode output leading to potential cross-site scripting issues.
– Failure to check for SQL entities of the voter's and candidate's details at the time of registration.
– Exposing an administration function through the customer-facing web application.

The list of threats and vulnerabilities can be gathered and identified for e-voting system from the standards such as Online Web Application Security Project (OWASP) [39] , National Vulnerability Database (NVD) [40], and Web Application Security Consortium (WASC) [41].

3.10 Step 10: Perform risk assessment

The next step is to assess and determine the risk when the threats and vulnerabilities occur. The impact of threats and vulnerabilities is analyzed, and risk determination process is carried out. Risk assessment is done by using the Microsoft method [29] of risk analysis for an e-voting system.

In the Table 1, threats and the level of risk by the threat to e-voting system are presented. In DREAD method [(Damage Potential + Reproducibility + Exploitability + Affected Users + Discoverability)/5] in Table 1, count the values 1–3 for a given threat, i.e., 1 for low, 2 for medium, and 3 for high. The result can fall in the range of 5–15. The threats with overall ratings of 12–15 are considered as high risk, 8–11 as medium risk, and 5–7 as low risk. This method is followed by Microsoft for threat analysis of the software developed in the design phase. Common Vulnerability Scoring System (CVSS) [42] can also be used to rate the risk by the vulnerability, because CVSS provides a universal open and standardized method for rating IT vulnerabilities.

3.11 Step 11: Categorize and prioritize the threats and vulnerabilities for mitigation

From Table 1, the threats are categorized under the security goals it affects and based on the level of risk they are prioritized. If a threat is rated high, it causes a significant damage

Table 2 Categorization and prioritization of threats and vulnerabilities for e-voting system

Threats	Category	Priority
The voter detail gets corrupted by buffer overflow	Integrity	High
Code injection to change the behavior of the election system at the time of registration	Reliability	Medium
SQL injection leads theft of voters or candidates details of election system	Integrity, availability	High
A malware accesses to the selected voting options at the voters PC	Confidentiality	High
Authentication token from the smart card can be forged	Confidentiality	High
Brute force attack against the credentials of the voter and election officer	Confidentiality	High
Man-in-the-middle, voter contest modification between the electoral roll service and the authentication service	Integrity, Authentication	High

to the security of e-voting system; consequently high priority is given to this threat in Table 2. The high risk need to be addressed as soon as possible and medium threats need to be addressed, but with less urgency. In the next step, with the identified threats it will be able to identify misuses of the system and generate misuse cases diagram for an e-voting system.

3.12 Step 12: Generate misuse case diagram for the e-voting system

The next step is to generate misuse cases diagram and to demonstrate a sample misuse case diagram for Part 1 of the e-voting system. Fig. 6 depicts the misuse case diagram with misuses, functional requirements, and actors.

Misuse case name—misuse of e-voting system
Purpose—for analyzing how e-voting system is misused by the hacker
Actors—voters and hackers
Preconditions—the voter and hackers must be connected to the Internet via a browser, the voter is authenticated, and the registration process is started
Post-conditions—the voter details are stored in the system

3.13 Step 13: Identify security requirements

The sample security requirements [35] for Part 1 and Part 2 of e-voting system with respect to BG 1.1 and BG 2.1 are listed below:

SR 1.1.1 It should not be possible to insert, delete, or modify any voter detail without authorization in the election system.

SR 1.1.2 It should be ensured that the election system presents an authentic registration to the voter.

SR 1.1.3 The solution for voter registration in an uncontrolled environment should issue a message to inform the voter whether the vote has been successfully registered.

SR 1.1.4 The election system should provide the e-voter with 'end-to-end' proof that the voter's application is received and recorded.

SR 1.1.5 The election system should ensure that the voter's choice is accurately represented in the vote and that the sealed vote is successfully stored.

SR 2.1.1 To allow for a delay in messages when passing over the election channel, the acceptance of electronic votes into the election system should remain open for a configurable period of time after the end of the polling phase.

SR 2.1.2 The voter can, at any time up to the point of vote casting, abort his polling process without losing his right to vote due to timeout or errors during communication.

SR 2.1.3 A voter should only be able to vote in contests that he/she is entitled to vote in.

SR 2.1.4 The e-voting components of the election system should be configurable to require authentication for every contest, every vote or every session.

SR 2.1.5 The voter authentication should expire after an idle period. The length of the idle time-out period should be configurable.

3.14 Step 14: Generate use cases diagram considering security requirements

The security requirements are gathered; for better understanding, the use case diagram of the applications should be developed that encompasses the security requirements of

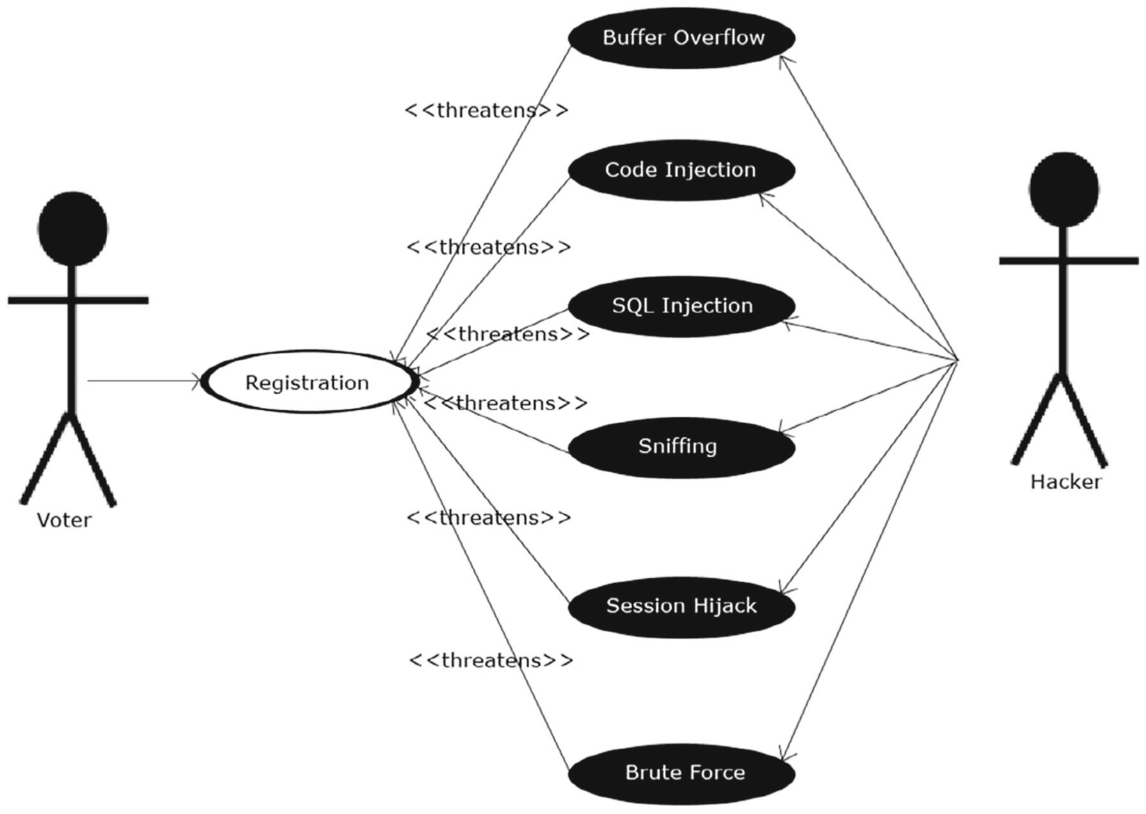

Fig. 6 Misuse case diagram for Part 1 of e-voting system

the system. The sample use case diagram generated for the e-voting system considering security requirements is shown in the Fig. 7.

3.15 Step 15: Generate structural analysis models

The structural view for Part 1 of e-voting system can show the flow of each process while a voter starts his voting process. The processes are explained below.

Secure voter client In the voter computer, the voter registration on the client side should be encrypted, to guarantee voter's details integrity, even from insider attacks.

Voter registration Accessible web application handles all the interaction with the voter. It presents the user interface, downloads the secure voter client, and acts as a proxy to the electoral database.

Voter registration server In this server, each voter details are collected and stored securely.

Pre-processing server The server pre-processes the encrypted voter details before being sent to the storage, by eliminating duplicates and verifying the voter against the electoral roll. In the next step, the specification can be added with the detailed UML diagrams which are improved in iterations.

3.16 Step 16: Develop UML diagrams

With the use case diagrams developed for the e-voting system considering security requirements, the structural analysis models such as data flow diagram, overall structural diagram, and UML diagrams such as class, activity, and sequence diagrams for e-voting system can be developed. By iterations, the level of abstraction of the diagrams can be decreased.

Along with these steps, validation and specification of security requirements can be done in parallel. These steps help to elicit and analyze the security requirements in RE phase, thereby reducing the burden of the developers to solve security issues. The elicited security requirements and generated UML diagrams can be used for the later phases such as design, implementation, and testing, to develop a vulnerability-free system. These requirements can also be used to generate test cases and perform penetration testing.

4 Evaluation and discussion

In this section, we discuss about the experimental setup and results obtained by evaluating the effectiveness and perfor-

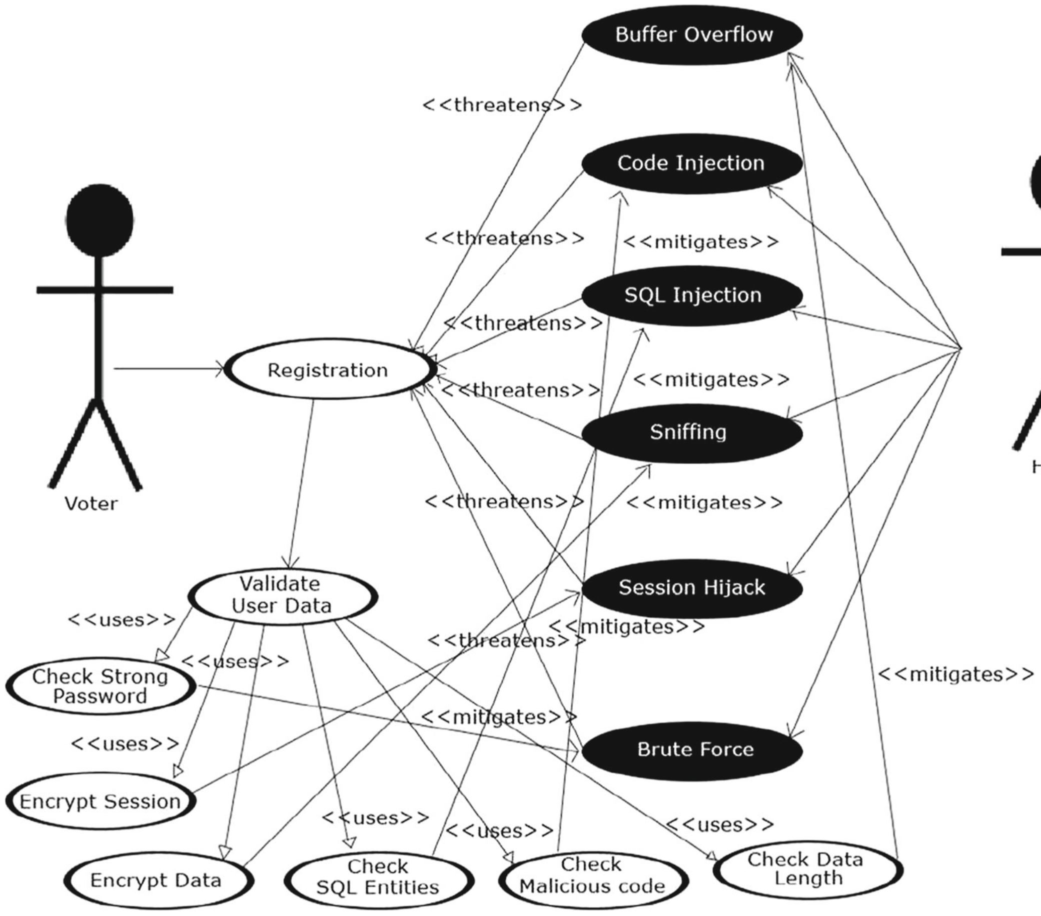

Fig. 7 Use case diagram for e-voting system considering security requirements

mance of MOSRE framework to elicit and specify security requirements. Further, the results obtained with the proposed SRE method (MOSRE) are compared with other existing methods such as (i) SREF, (ii) SREP (iii) MSREF, and (iv) Without using SRE methods.

4.1 Experimental setup

The evaluation was conducted by 30 people: academic users such as 6 professors, 6 research scholars, 10 postgraduate students, and 8 industry professionals. They acted as one of the stakeholders such as voter, candidate, election officer, requirements analyst, developer, security expert, tester, designer to participate, provide their views and idea, and to identify security requirements. They were divided into five groups and performed security requirements analysis for five different e-voting software systems using SREF, SREP, MSREF, MOSRE framework and without using any SRE methods respectively.

Four groups applied different SRE methods and identified the security requirements and business requirements which are considered as functional requirements in the MOSRE framework. The last group identified functional and non-functional requirements for e-voting system without using any SRE methods. Each group prepared Software Requirements Specifications (SRS) for e-voting systems as the product of RE phase. These 30 participants are considered to be valid participants, because they have developed requirements specifications in the past in both academic and industry settings, but none has worked with security in the context to requirements analysis and specifications.

Two evaluation studies were conducted on the results obtained by applying MOSRE and other SRE methods. The SRS of five e-voting systems developed by the different groups with the respective SRE methods have been examined. The effectiveness and performance of MOSRE were evaluated and compared with SRE methods such as SREF, SREP, and MSREF.

Table 3 Parameters to analyze the effectiveness of MOSRE

Parameters	Qualitative analysis values	Measured in counts
Assets	High, medium, low	Number of assets
Threats	Very high, high, medium, low, very low	Number of threats
Vulnerabilities	Very high, high, medium, low, very low	Average number of vulnerabilities

Table 4 SR types to categorize identified security requirements

SR types
Identification
Authentication
Authorization
Integrity
Intrusion detection
Confidentiality
Availability

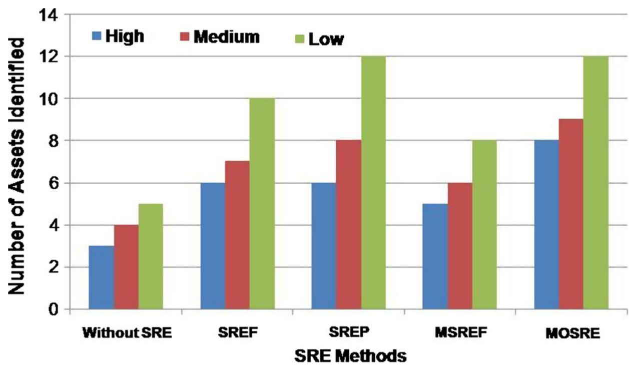

Fig. 8 Effectiveness of MOSRE in identifying assets

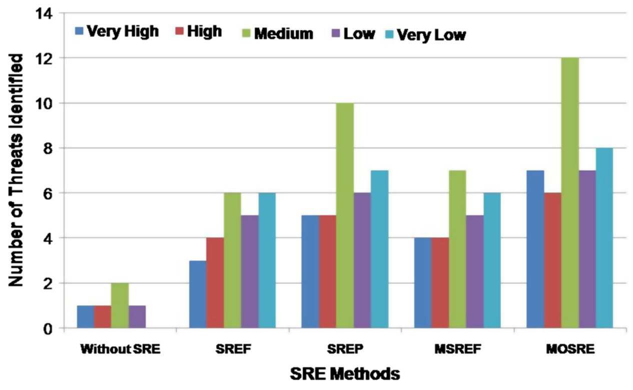

Fig. 9 Effectiveness of MOSRE in identifying threats

4.1.1 *Comparative analysis of effectiveness with existing SRE methods*

MOSRE framework was evaluated by analyzing its effectiveness in identifying assets, threats, and vulnerabilities and compared with existing SRE methods. The parameters considered to analyze the effectiveness of MOSRE are shown in Table 3. They were assigned qualitative values as given by the NIST for risk assessment and measured by count values.

The evaluation methodology used was to examine the coverage of assets, threats, and vulnerabilities at each level of qualitative values in the SRS of respective e-voting system listed by adopting the existing and proposed SRE methods and without using SRE methods respectively.

4.1.2 *Comparative analysis of performance of MOSRE with existing SRE Methods*

The number of security requirements identified and the time taken by each SRE method to elicit business and security requirements for the e-voting system are analyzed, and thereby the performance of MOSRE was compared with other SRE methods.

The collected SRS were examined for the number of security requirements identified under each category of security requirement types [30] given in Table 4. The performance is high if more number of security requirements are identified for the software system being developed. The total time needed in person-days for each SRE method to elicit security requirements was obtained.

4.2 Result analysis

The results obtained were analyzed to understand the effectiveness and the performance of MOSRE with the existing SRE methods.

4.2.1 *Effectiveness comparison of MOSRE with other SRE methods*

The effectiveness of MOSRE was computed based on the number of assets, threats, and vulnerabilities identified in each e-voting system.

From the chart shown in Figs. 8, 9, and 10, it is clear that more number of assets, threats, and vulnerabilities are identified by using MOSRE framework than using existing SRE methods.

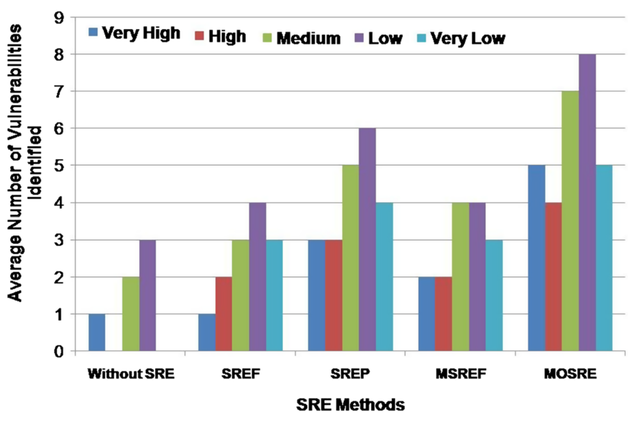

Fig. 10 Effectiveness of MOSRE in identifying vulnerabilities

It is found that higher coverage of assets, threats, and vulnerabilities is identified by using MOSRE framework than existing SRE methods at each level, thereby increasing the effectiveness of MOSRE in identifying the security requirements for developing secure e-voting system. SREP method lacks in framing objective of the software system and to identify the stakeholders, thus less number of important assets are identified and secured. They also fail to elicit business requirements, reducing the identification of threats and vulnerabilities. It is also found that less number of vulnerabilities are covered by using SREF, since they do not analyze and identify vulnerabilities of software systems. MSREF identified less threats and vulnerabilities, due to lack of an activity to obtain the artifacts of the software system. From the results of examining the SRS of the group those who adopted "without SRE method," it is clear that the SRE activity need to be performed systematically in iterative fashion at the early stages of SDLC.

4.2.2 *Performance comparison of MOSRE with other SRE methods*

The number of security requirements identified by using each SRE method by different group of participants were gathered from the SRS and categorized under security requirement types as tabulated in Table 4.

Table 5 shows that more number of security requirements were identified in each category of SR types by using MOSRE. The performance was computed based upon the total number of identified security requirements to the time taken to complete the requirements analysis and specification which was normalized for person-days with respective SRE methods.

In Fig. 11, though the time to implement SREF, SREP and without using SRE method is less than MOSRE, they fail to identify all security requirements of the system. MOSRE helped to elicit more security requirements for e-voting system which in turn will be cost effective since quality such as security requirements plays a major role for an effective secure software system and it decides the security level of the software system. Thus, the performance of MOSRE is high when compared with the methods to elicit security requirements.

From the results given in Figs. 8, 9, 10, 11 and Table 5, it can be inferred that:

(i) MOSRE has improved the identification of assets, threats, and vulnerabilities, thereby eliciting security requirements;
(ii) MOSRE has identified more number of assets, threats, and vulnerabilities when compared to existing SRE methods;

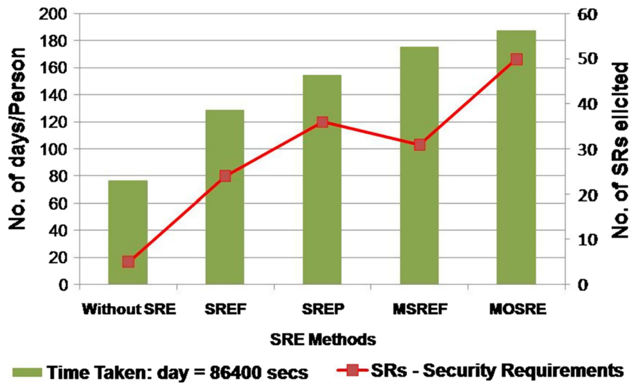

Fig. 11 Performance of MOSRE to elicit security requirements

Table 5 Number of security requirements identified by each SRE method

SR types	Without SRE	SREF	SREP	MSREF	MOSRE
Identification	1	2	5	3	6
Authentication	0	5	7	6	8
Authorization	2	6	8	7	9
Integrity	0	4	6	4	11
Intrusion detection	0	3	3	3	4
Confidentiality	2	3	6	7	11
Availability	0	1	1	1	1

(iii) In assets identification, MOSRE has given 10 % effectiveness when compared to SREP and 19 % than SREF;
(iv) In case of threats, MOSRE is 18 % more effective than SREP and 40 % than SREF;
(v) For vulnerabilities identification, MOSRE has resulted in 28 % improvement than SREP and 55 % than SREF;
(vi) The performance of MOSRE is comparatively higher than the existing SRE methods in identifying more number of security requirements.
(vii) Though the time to implement MOSRE is high when compared to other SRE methods, MOSRE helps the developers to elicit and specify quality security requirements rather to develop vulnerability or risky software system;
(viii) The overhead of developers is also reduced since security requirements are identified at the early stage of software development.

With the MOSRE framework, the level of security is improved since the threats and vulnerabilities are covered at network, application, and database levels in the early stages of software system development. Thus, the resultant security requirements obtained with MOSRE framework were very promising to attain threat- and vulnerability-free software systems. However, it is not able to reach 100 % results. It is due to the fact that the assessments are carried out at very early phase (RE phase) of SDLC. It is also due to the variations realized in design and implementation; it may be possible to cover all threats and vulnerabilities by testing phase.

5 Conclusion

The evaluation of MOSRE on web application was carried out by a group of participants to elicit and specify security requirements. The SRS gathered from the participants were examined for the lists of assets, threats, and vulnerabilities and compared with the SRS of existing SRE methods. On investigating the SRS of MOSRE, it is found that the effectiveness and performance are comparatively better than the existing methods.

It is also inferred from the evaluation that MOSRE is simple and understandable for the requirements engineers to elicit security requirements, which guarantee the desired level of protection to the software systems.

The identified security requirements can be reused with the activities of any SRE methods to elicit effective security requirements. We intend to extend further our work for security requirements reusability, in order to reduce security knowledge and dependency on security experts. It will help to save time/cost and make better choices in applying security requirements, since the framework allows requirements engineers to exploit the accumulated knowledge. Thus, very high level of security can be achieved for the software systems.

Acknowledgments We would like to acknowledge and thank the reviewers and the editors for their valuable comments and suggestions to improve our paper.

References

1. Balzarotti, D., Banks, G., Cova, M., Felmetsger, V., Kemmerer, R., Robertson, W., Valeur, F., Vigna, G.: Are your votes really counted? Testing the security of real-world electronic voting systems. In: ACM Proceedings of the international symposium on Software testing and analysis, pp. 237–248 (2008)
2. Prosser, A., Kofler, R., Krimmer, R., Unger, M.K.: Security assets in E-voting. In: Proceedings of the 1st international workshop on electronic voting, pp. 171–180 (2004)
3. Sindre, G., Firesmith, D.G., Opdahl, A.L.: A reuse-based approach to determining security requirements. In: Proceedings of 9th international workshop on requirements engineering: foundation for software quality, pp. 16–17 (2003)
4. Viega, J., McGraw, G.: Building secure software. Addison-Wesley, Boston (2001)
5. Lipner, S., Howard, M.: The trustworthy computing security development life cycle. Microsoft Corporation. http://msdn.microsoft.com/en-us/library/ms995349.aspx (2005)
6. Graham, D.: Introduction to the CLASP process. Build security. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/requirements/548.html (2006)
7. Mead, N.R., Houg, E.D., Stehney, T.R.: Security quality requirements engineering (SQUARE) methodology. Technical Report CMU/SEI-2005-TR-009, Software Engineering Institute, Carnegie Mellon University (2005)
8. Haley, C.B., Laney, R., Moffett, J.D., Nuseibeh, B.: Security requirements engineering: a framework for representation and analysis. IEEE Trans. Softw. Eng. **34**(1), 133–152 (2008)
9. Mellado, D., Fernndez-Medina, E., Piattini, M.: A common criteria based security requirements engineering process for the development of secure information systems. Comput. Stand. Interfaces **29**(2), 244–253 (2007)
10. Mouratidis, H., Giorgini, P., Manson, G.: When security meets software engineering: a case of modeling secure information systems. J. Inf. Syst. **30**(8), 609–629 (2005)
11. Giorgini, P., Mouratidis, H., Zannone, N.: Modeling security and trust with secure tropos. Integrating security and software engineering: advances and future visions. IGI Global, Pennsylvania (2007)
12. Tndel, I.A., Jaatun, M.G., Meland, P.H.: Security requirements for the rest of US: a survey. IEEE Softw. **25**(1), 20–27 (2008)
13. Fernandez, E.B.: A methodology for secure software design. In: Proceedings of the international symposium, web services and applications. www.cse.fau.edu/~ed/EFLVSecSysDes1 (2004)
14. Rosado, D.G., Gutirrez, C., Fernndez-Medina, E., Piattini, M.: Security patterns and requirements for internet-based applications. Internet Res. **16**(5), 519–536 (2006)
15. Appel, A.W., Ginsburg, M., Hursti, H., Kernighan, B.W., Richards, C.D., Tan, G., Venetis, P.: The New Jersey voting-machine law suit and the AVC advantage DRE voting machine. EVT/WOTE09, Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (2009)
16. Hursti, H.: Diebold TSx evaluation: critical security issues with diebold TSx. Black Box Voting. http://www.blackboxvoting.org/BBVtsxstudy.pdf (2006)

17. Kohno, T., Stubbleeld, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: IEEE symposium on security and privacy. IEEE Computer Society, pp. 27–48 (2004)
18. http://www.emarketer.com
19. Mainichi newspaper. http://www.mainichi.co.jp (Japanese), June 24 (2002)
20. A NASSCOM eGovernance study on issues, challenges and recommendations. www.egovreach.in/social/content/karnanataka-proposing-e-vote
21. Caarls, S.: E-voting handbook: key steps in the implementation of E-enabled elections. Council of Europe, Strasbourg (2010)
22. Jefferson, D., Rubin, A.D., Simons, B., Wagner, D.: A security analysis of the secure electronic registration and voting experiment (SERVE). http://www.servesecurityreport.org/paper.pdf (2004)
23. Feldman, A., Halderman, J., Felten, E.: Security analysis of the diebold AccuVote-TS voting machine. In: Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop (2007)
24. Lambrinoudakis, C., Kokolakis, S., Karyda, M., Tsoumas, V., Gritzalis, D., Katsikas, S.: Electronic voting systems: security implications of the administrative workow. In: Mark, V., Stepankova, O., Retschitzegger, W. (eds.) DEXA2003. LNCS, vol. 2736, p. 467. Springer, Heidelberg (2003)
25. Xenakis, A., Macintosh, A.: Procedural security analysis of electronic voting. In: Rauterberg, M. (ed.) ICEC2004. LNCS. Springer, Heidelberg (2004)
26. Xenakis, A., Macintosh, A.: Procedural security and social acceptance in E-voting. In: HICSS2005: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS2005)-Track5, p. 118.1. IEEE Computer Society, Washington, DC, USA (2005)
27. Manadhata, P., Wing, J., Flynn, M., McQueen, M.: Measuring the attack surfaces of two FTP daemons. In: QoP2006: Proceedings of the 2nd ACM workshop on Quality of protection, pp. 3–10. ACM Press, New York (2006)
28. Howard, M., Pincus, J., Wing, J.: Measuring relative attack surfaces. Computer Security in the 21st Century, pp. 109–137, Springer, US (2005)
29. Swiderski, F., Snyder, W.: Threat modeling. Microsoft Press, US (2004)
30. Suleiman, H., Svetinovic, D.: Evaluating the effectiveness of the security quality requirements engineering (SQUARE) method: a case study using smart grid advanced metering infrastructure. Requirements engineering. Springer, Berlin (2012)
31. Salini, P., Kanmani, S.: Survey and analysis on security requirements engineering. Int. J. Comput. Electr. Eng. **38**(3), 1785–1797 (2012)
32. Salini, P., Kanmani, S.: A model based security requirements engineering framework. Int. J. Comput. Eng. Technol. **1**(1), 180–195 (2010)
33. Salini, P., Kanmani, S.: A model based security requirements engineering framework applied for online trading system. In: Proceedings of IEEE international conference on recent trends in information technology, pp. 1195–1202 (2011)
34. Salini, P., Kanmani, S.: Application of model oriented security requirements engineering framework for secure E-voting. In: Proceedings of CSI 6th international conference on software engineering, IEEE, pp. 1–6 (2012)
35. Salini, P., Kanmani, S.: Security requirements engineering for specifying security requirements of an e-voting system as a legitimate solution to e-governance. Int. J. Wirel. Mobile Comput. **7**(4), 400–413 (2014)
36. Rubin, A.D.: Security considerations for remote electronic voting. Commun. ACM **45**, 39–44 (2002)
37. Smith, R.G.: The risks and benefits of electronic voting. In: 15th Australian Forum—Melbourne (2001)
38. Weiss, M.: Modeling security patterns using NFR analysis. Information security and ethics: concepts, methodologies, tools, and applications. Idea Group Publishing, Pennsylvania (2008)
39. OWASP. https://www.owasp.org
40. NVD. http://nvd.nist.gov/scap.cfm
41. WASC. http://www.webappsec.org/
42. CVSS. http://www.rst.org/cvss