The Nucleus ***54****, No. 3 (2017) 164-172*

www.thenucleuspak.org.pk

The Nucleus
ISSN 0029-5698 (Print)
ISSN 2306-6539 (Online)

Towards Capturing Security Requirements in Agile Software Development

W. Ahmed[1*], Y. Hafeez[1] and G. Chiurlea[2]

[1]*University Institute of Information Technology, Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, Pakistan*

[2]*University of Wolverhampton, Wolverhampton, WV1, U.K*

ch.waqasahmed@hotmail.com; yasir@uaar.edu.pk; gina.chiurlea@hotmail.co.uk

ARTICLE INFO

Article history :
Received : *23 January, 2017*
Accepted : *14 September, 2017*
Published : *30 September, 2017*

Keywords:
IPTFN
I-IPTFOWA operator
I-IPTFHA operator
MAGDM problem

ABSTRACT

Software use is an unavoidable reality. Increased use expands the opportunity for malicious use which threatens security and privacy. There are many factors like data loss, increase in budget cost due to security breaches, pending legislation and competitive advantage are driving software developers to integrate security into software development rather than adding security in later stages of development. The approach presented here addresses elicitation, prioritization, analysis of requirements and security requirements. This can be done by identifying candidate's security goals, their categorization and understanding with stakeholders to develop preliminary security requirements and then prioritization and at the end security requirements are output.

1. Introduction

Software security is a complex, evolving problem that has only recently begun to receive additional attention. One area that needs improvement is building security into software [1] rather than correcting security flaws after release. Integrating security requirements into the software development life cycle (SDLC) from the start can significantly improve software security and reduce rework at later stages. However, traditional SDLC processes leave non-functional requirements, such as security, as an afterthought. Usually, small software development teams have limited resources and work in shorter time frames. The need to balance resources for fast paced software development projects and to remain competitive in the market has influenced the shift from traditional to agile development processes [2]. Therefore, there is a need for a security requirements approach to aid small, agile development organizations. So the approach must be easy to implement and validate in agile software development.

2. Literature Review

Security has predominately been an afterthought to the software development process. Functional requirements are developed at the beginning of the process, but non-functional requirements such as security are often overlooked. This results in security requirements that are "bolted on" later in the development cycle or worse, after the product has been released in response to security events, market response or regulatory demands [3]. Software security vulnerability awareness increased not only for critical system software, but also for common software that impacted the general public. Highly publicized data breaches, such as the 2003 theft of over 45 million credit and debit card data [4], increased awareness among the general public. Legislation at the state and federal level has also been increasing as the need for privacy and security becomes apparent. Some legislation has been longstanding, such as the Privacy Act of 1974, but additional legislation has recently been enacted. Privacy and security rules for the Health Insurance Portability and Accountability Act (HIPAA) were enacted at the federal level in 2003. Nearly all states have enacted either security2 or data breach3 notification legislation. Vulnerability awareness also drove increased security awareness among software engineers who frequently turned to implementing security mechanisms in order to mitigate risk. However, this does not address the core problem that security requirements need to be built into software from the start, not addressed later [5]. Small organizations with fewer than twenty people on the development team are likely to operate with limited resources. For agile organizations, development will be iterative and extensive documentation will be less valuable than developing a working product. Development schedules are likely to be shorter placing increased emphasis on project cost and time constraints. Therefore, integrating security requirements into the software development process for small, agile organizations requires careful balancing of project resources and constraints. Increasing security threats, lack of software engineering security skills, consumer expectations for secure software and project constraints for small, agile organizations demonstrates the need to improve security requirements engineering [6]. The increased complexity and integration of systems increases attack surfaces and makes it difficult to understand software vulnerabilities. Software engineers traditionally do not receive adequate training or attention to address the security of software vulnerabilities. Publicity of the latest data breach or widespread virus now makes front page news. In

[*]Corresponding author

addition, introducing project requirements strain limited project resources in terms of cost, time and personnel. Consider the analogy of bank security. A customer walking into a bank has an assumed expectation of security. They expect security via safes, locks, guards and identity verification. These basic security devices are easy to understand and can be verbalized regardless of technical expertise. There are likely to be additional security devices in place at a bank, but understanding these devices requires additional technical expertise that the general customer does not possess. While customers do explicitly request all elements of banking security, they express their requirements by choosing the bank with a combined fee and security structure that balances their needs. Consumers of software have similar security appetites. Security may again be expected, but verbalizing specific security requirements may be difficult due to the lack of understanding. It is difficult to elicit security requirements without the aid of those experienced with software security. Justifying additional costs for security, in terms of time or money, can be a difficult to sell since they are non-functional requirements.

2.1 Comparison of Security Requirement Elicitation Techniques

2.1.1 Secure Software Development Lifecycle (SSDL) touch points

SSDL Touch points consist of architectural analysis, code review and security testing practices which should be included in any software security framework. Touch points provide an overview of practices that should be followed but do not define specific tasks or processes for accomplishing these practices.

2.1.2 Open web application security project (OWASP) cheat sheets

These are used to help and aid software engineers obtain solutions to specific problems and overall guidance for application security. Cheat sheets generally focus on development specific topics rather than requirements development.

2.1.3 Agent oriented software methodology (AOSE)

The AOSE methodology extends "Formal framework for modeling and analyzing security and trust requirements [7]. AOSE takes care of computer systems as well as organizational environment in which system operates. The Drawbacks of this is that it lacks the risk information that could be used for prioritization of goals and the assumption that requirements have been discovered and identified for the purpose of modeling and analysis.

2.1.4 The software security framework

The Software Security Framework (SSF) addresses overall security, not just the development of software security requirements [7]. SSF is organized into four domains: Governance, Intelligence, SSDL Touch points, and Deployment. Each domain has three practices with individual activities (total 20 activities for all domains). Although SSF defines specific practices to address security requirements engineering, the large number of activities and abstract nature of the framework do not make SSF suitable as a requirements elicitation solution

2.1.5 Security maturity model

It provides an organization with broad security perspectives to build an initiative. Deficiencies in any practice area or domain can be prioritized to improve the security maturity level for the organization. The disadvantage is that the organization must still choose an approach to address deficiencies [8].

2.1.6 Square

To focus on methodology, elicitation and prioritization are software development phases. The steps of Square Methodology give results based on recommended input information. Each step has defined input and output. Drawback of this methodology lies in step three (Develop Artifacts). Researchers suggest that these artifacts may be related to the design phase rather than the requirements phase [7].

2.1.7 Comprehensive lightweight application security process (CLASP)

CLASP is intended to be applicable to existing software or new development projects using high-level perspectives or views. CLASP views include concepts, roles, activity assessment, activity implementation and vulnerability. The iterative nature of CLASP departs from traditional development and favors agile development. CLASP is not a one-size fits all solution for improving application security and specific tools are not defined.

2.1.8 UML sec and secure unified modeling language

Security profiles are generated consisting of a concept called stereotypes that includes tagged values and constraints. A goal of UML sec is to aid software engineers who do not have strong security backgrounds to use UML sec to model security requirements. The formal nature of UML diagramming works best in traditional development but can be a drawback for agile development teams.

2.1.9 Attack patterns and security patterns

Attack patterns describe the techniques that attackers may use to break software". Software engineers still need to have a considerable arsenal of Information available to begin constructing attack trees.

We propose a security requirements elicitation approach that is part of the requirements elicitation phase. Preliminary functional requirements artifacts are used as inputs and draft security requirements are output. Although not part of the approach, draft security requirements can be then modeled, defined and validated as part of the final software requirements specification (SRS). The security requirements elicitation approach activities are defined as follows and displayed in Fig. **1.**

Proposed Approach

- Identify candidate security goals
- Categorize security goals based on Security principles.
- Understand stakeholder goals and develop preliminary Security requirements.
- Prioritize preliminary security requirements.

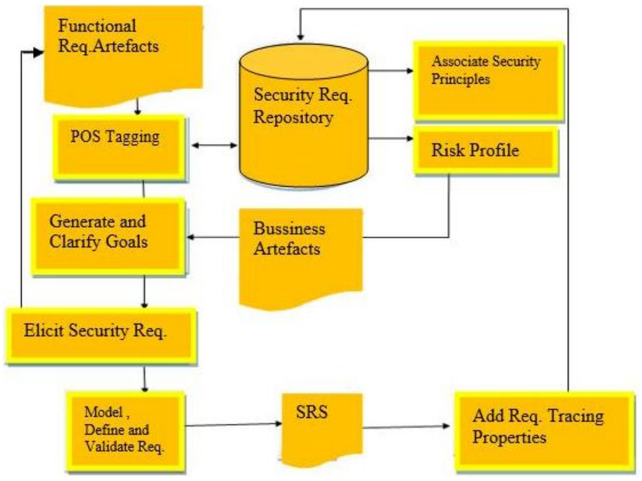
Functional Req.Artefacts
POS Tagging
Security Req. Repository
Associate Security Principles
Risk Profile
Generate and Clarify Goals
Bussiness Artefacts
Elicit Security Req.
Model , Define and Validate Req.
SRS
Add Req. Tracing Properties

Fig. 1: Proposed approach for capturing security requirements

The proposed requirements elicitation approach will be iterative. Part of Speech (POS) tagging activities and the implementation of a security requirements repository are also innovative in that they are not currently implemented by any other approach.

3.1 *Security Requirements Repository Design*

Activities in the security requirements elicitation approach depends on the development of a security requirements repository and below is detail of entities and attributes for the repository. (Primary keys are denoted as PK.)

3.1.1 *Security terminology entity*

Attributes for the security terminology entity are TerminologyID (PK), Security Term, and Security Term Description. Security Term attributes is single term that will be used during POS tagging. The repository will be populated with terms identified by the requirements engineer based on experience or using a dictionary of security terms. Each security term has additional details, such as definitions or phrases, which enhance the understanding of each security term (Table 1).

3.1.2 *Security principles entity*

Attributes for the security principles entity are PrincipleID (PK), Principle, and Description. Security principles are confidentiality, integrity, and availability (CIA), but additional security principles can be defined as well. Description attributes are definitions or details to provide a common basis of understanding among stakeholders (Table 2).

Table 1: Security terms

Security Terms		
Access	Certificates	Malicious
Audit	Deny	Password
Authenticate	Encrypt	Permission
Authentication	Encryption	Privileges
Authorize	https	Risk
Authorized certificate	Logon	Security

Table 2: Security principles and description

Principle	Description
Confidentiality	Unauthorized disclosure of information
Integrity	Unauthorized modification of information
Availability	Disruption of access to an information system.

3.1.3 *Terminology and principles entity*

Attributes for the terminology and principles entity are TermPrincipleID (PK) and secondary keys, TerminologyID and PrincipleID.

3.1.4 *Requirements artifacts entity*

Attributes for the requirements artifacts entity are ArtifactID (PK), Artifact Name, Artifact Description and Artifact Type.

3.1.5 *Security requirements entity*

Attributes for the security requirements entity are SecReqID (PK), Term PrincipleID, SecReq Description, SecReq Comments and ArtifactID.

3.1.6 *Software requirements entity*

Attributes for the software requirements entity are SoftwareReqID (PK) and secondary key, SecReqID.

3.2 *Process of Proposed Technique*

The activities in the security requirements elicitation approach are:

- Identify candidate security goals
- Categorize security goals based on security principles
- Understand stakeholder goals and develop preliminary security requirements
- Prioritize preliminary security requirements.

Each activity defines inputs, roles, techniques and output. Inputs are requirements related artifacts. Roles are the development team and business stakeholders responsible for the activity.

3.2.1 Identify candidate security goals

Identifying security requirements can be difficult if stakeholders have difficulty in expressing security related needs. The result may be functional requirements written with security terminology that implies security requirements but that are not explicitly defined. If security terminology can be discovered, candidate security goals can be identified that with further analysis could be used to develop security requirements.

POS tagging method is used to extract opinions from reviews [9] and is commonly applied to identify features as noun phrases and opinions as close proximity adjectives. Parsing tools, such as the Stanford Parser, are also available to automate POS tagging and determine word frequency. Proximity of terms may reveal relevant information within a software requirements document. For example, if the terms "security" and "encryption" are located within close proximity of each other, then the terms may be associated with each other and could reveal an underlying security requirement. Security terms should therefore be tagged and follow-up analysis performed to determine if security requirements can be captured. This is the proposed method in which POS tagging will be implemented to discover security requirements.

The requirements engineer take as input preliminary requirements documents. These documents can be draft of software requirements specifications (SRS), requests for proposals (RFP's) and |other documents i.e. regression test documents, design documents that will be used to generate the final software requirements specification. Artifacts are scanned for commonly used security terminology. Generating commonly used security terms can be left up to the knowledge of the requirements engineer or a dictionary of security terminology can be used if available. Discovered security terminology and the location within the requirements artifacts are tagged for additional review. After all artifacts have been tagged, the requirements engineer reviews the requirements artifacts and identifies candidate security goals CSG). CSG are general requirements written with implied security needs that may be developed into security requirements. For example, a requirement artifact was scanned and tagged for the word malicious. The following functional requirements (FR) were found:

FR–1: "Malicious applications are detected and stopped"

FR–2: "Malicious applications are handled appropriately"

The requirements engineer would tag the location(s) where the term malicious was found and generate a CSG such as:

CSG-1: The system will recognize, catch and calculate appropriate actions to malicious requests.

Further analysis of the requirement documents also opens requests related to access policies. CSG can be refined to include this information:

CSG-1: The system shall recognize, catch and take appropriate action to malicious requests using security policies.

After all artifacts have been scanned, tagged and reviewed, a candidate security goals artifact will be created as output to identify activity. This artifact will be used as input to categorize security goals activity.

3.2.2 Categorize security goals based on security principle

Candidate security goals identified from previous activity are used as input for categorizing activity. The requirements engineer and business stakeholders work together to review all requirements artifacts that have tagged candidate security goals. Interactive meetings (face-to-face, web facilitated, teleconference) will likely be the most efficient, but virtual document review can also take place. Prior to meetings, the requirements engineer can assess the goals for quick categorization to facilitate efficient communications. Each candidate security goal should be categorized with at least one security principle.

Referring to the earlier example of CSG, from the identify activity; the following security principles can be associated with CSG :

SP-1: Confidentiality: Save from unauthorized disclosure of information.

SP-2: Integrity: Save from unauthorized changing or destruction of information.

The requirements engineer and business stakeholders will agree upon the general security principles. If a candidate security goal cannot be categorized, additional elicitation and analysis can be iteratively undertaken with the stakeholders. If CSG's still cannot be categorized after additional iterations, it will fail the activity andCSG will be discarded.

3.2.3 Understand stakeholder goals and develop preliminary security requirements

Using the refined security goals from the categorize activity, the requirements engineer and business stakeholders seek to further understand the implications of the security goals. Additional artifacts such as policies and regulations are also used as input to this activity. The requirements engineer chooses techniques and tools to further elicit information from business stakeholders. Face-to-face or virtual meetings are a good choice of techniques for generating discussion. The choice of tools is likely to be influenced by the requirements engineer but could include generating misuse or abuse cases, attack trees, or other security related modeling [10, 11]. The output from this activity is a set of preliminary security requirements based on the CSG's. Continuing with the previous example, the preliminary security requirement (PSR) generated from CSG-1 could be:

PSR-1: The system should save the confidentiality and integrity of data by identifying, detecting and ignoring malicious applications using security policies.

3.2.4 Prioritize preliminary security requirements

Preliminary security requirements need to be prioritized to generate the final security requirements. During this activity, the requirements engineer continues to work with business stakeholders to analyze the input preliminary security requirements. Recommended analysis techniques are Failure Modes and Effects Analysis (FMEA) [12]. This approach is very useful to communicate and clarify the impact of technical materials in an easy to understand format. Analysis requires creating severity, occurrence and detection rankings in order to determine a risk priority number (RPN). The RPN is calculated as the product of the risk rank. FMEA standard scale, Occurrence scale, Detection scale and FMEA analysis of security requirements are shown in Tables 3, 4, 5 and 6, respectively.

Table 3: FMEA standard scale

Impact Rating	Rating	Criteria: A Failure Could…
Very high	9-10	Virtually inevitable
high	8-7	Failure likely, many known cases
moderate	4-6	Somewhat likely, some known cases
low	3-2	Few known cases
unlikely	1	no known cases

Severity scale = Likely impact of failure

Table 4: FMEA occurrence scale

Impact	Rating	Criteria: A failure could
Bad	10	Injure a customer
-	9	Be illegal
-	8	Render the software unfit
-	7	Extreme customer dissatisfaction
-	6	Result in partial malfunction
-	5	Cause loss of performance
-	4	Cause minor performance loss
-	3	Cause a minor nuisance
-	2	Be unnoticed
Good	1	Be Unnoticeable and will not effect performance.

Occurrence Scale = Frequency of failure

Table 5: FMEA detection scale

Impact	Rating	Criteria: A failure could
Bad	10	>30%
-	9	<=3%
-	8	<=5%
-	7	<=1%
-	6	<=0.3 per 1000
-	5	<=1 per 10,000
-	4	<=6 per 100,000
-	3	<=6 per million
-	2	<=3 per billion
Good	1	<=2 per billion

Detection scale = Ability to detect failure

Table 6: FMEA analysis of security requirements

Failure	Effect	Severity	Occur	Detection	RPN
malicious request	Viewed	3	7	9	189
malicious request	Stolen	9	4	9	324
malicious request	Corrupted	5	4	4	82

RPN= (severity ranking) (occurrence ranking) (detection ranking).

The resulting RPN generates a prioritized list of potential security requirements.

4. Implementation of Proposed Work

The security requirements elicitation approach will be evaluated empirically by analyzing publically available software requirements specifications (SRS). An internet search of PDF and Word documents was conducted using the search term "software requirements specification". A base set of 46 SRS documents were downloaded of which three contained sections specifically for security requirements. The remaining 43 SRS documents were used and analyzed using POS tagging. After tagging analysis, a smaller subset of the tagged documents was selected and analyzed using the security requirements elicitation steps. We present POS tagging, security requirements elicitation and results in the next section.

4.1 POS Tagging

We developed a POS scanner to scan and tag the set of SRS documents. Small organizations are likely to generate SRS documents using word processing software rather than sophisticated software development management software. All PDF documents were converted to Word 2010 format (doc) in preparation for scanning. The scanning software was written in C# which integrates with Microsoft Word and can easily facilitate the scanning process. The basic steps in the scanning process are:

1. Open the document
2. Clear all bookmarks
3. Scan for, count and bookmark the location of each security term
4. Write the document name, security term and frequency to a text file
5. Save and close the document

Multiple files can be automatically scanned sequentially. The entire scanning and tagging process is automated and processing time was approximately 1.5 minutes per document. Table 7 shows the security terminologies with frequency and rank.

Table 7: Security terminology frequency and rank

Security terminology		
Security Term	Frequency	Rank
Access	416	2
Audit	28	10
Authenticate	5	17
Authentication	30	8
Authorize	0	19
Authorized	146	5
Certificate	205	4
Certificates	85	7
Deny	3	18
Encrypt	12	14
Encryption	20	12
https	14	13
Logon	8	15
Malicious	8	15
Password	237	3
Permission	86	6
Privileges	24	11
Risk	30	8
Security	551	1
No of scanned documents = 43		

4.1.1 *Analysis of tagged security terms*

Five security terms with the highest frequency are security, access, password, certificate, and authorized. Security terms with the lowest frequency are authorize, deny, authenticate, logon and malicious. Figs. 2 and 3 graphically display the security term frequency and average frequency for each of the selected security terms. The security term frequency per document revealed a total of 2,854 terms tagged with an average per document frequency of 66.4. Tagged term frequency ranged from a low of 14 to a high of 701. The average term frequency may be skewed by one document that has a very high term frequency. Without this document the average is closer to 51 but even at 66.4, it is low enough that manual review by a requirements engineer would not be cumbersome. We will analyze the SRS documents to determine if the size of the security term dataset impacts the viability of discovering candidate security goals. Results from the elicitation activities were analyzed to determine if the set of security terms can be pruned to a smaller set or if additional security terms are needed to generate security requirements. Table 8 shows the SRS document security term frequency.

Table 8: SRS document security term frequency

Security Term Frequency per SRS Document					
Doc No.	Frequency	Doc No.	Frequency	Doc No.	Frequency
1	119	16	20	31	54
2	24	17	20	32	52
3	20	18	41	33	63
4	14	19	113	34	56
5	35	20	27	35	36
6	22	21	83	36	52
7	701	22	141	37	50
8	17	23	90	38	64
9	44	24	31	39	84
10	29	25	183	40	73
11	36	26	26	41	43
12	21	27	35	42	47
13	18	28	87	43	49
14	14	29	44		
15	27	30	49		
Average frequency of security terms per document :66.4					
Total Security Term Frequency : 2854					
Total SRS Documents Scanned: 43					

4.2 *Security Requirements Elicitation*

Eight tagged documents with the highest frequency were chosen for further analysis using the security requirements elicitation activities. One of the documents was eliminated due to formatting issues. The document with the highest security term frequency was a highly complex document that specified multiple sub-systems and contained hundreds of functional requirements. The complexity of the SRS was not representative of the type of product that would be developed by a small organization and was also eliminated.

4.2.1 *Identify candidate security goals*

Each document was manually reviewed to determine if the tagged security terms were relevant to identifying candidate security goals. Custom code facilitated the process by selecting each tagged term allowing the reviewer to accept or reject each term based on the context of the language surrounding each term. Terms could be rejected (false positives) for a variety of reasons. Acronym lists, glossaries and references to other documents were common reasons for rejecting or “un-tagging” a term. Figs. 4 and 5 show the results of security term frequency before and after false positives are removed.

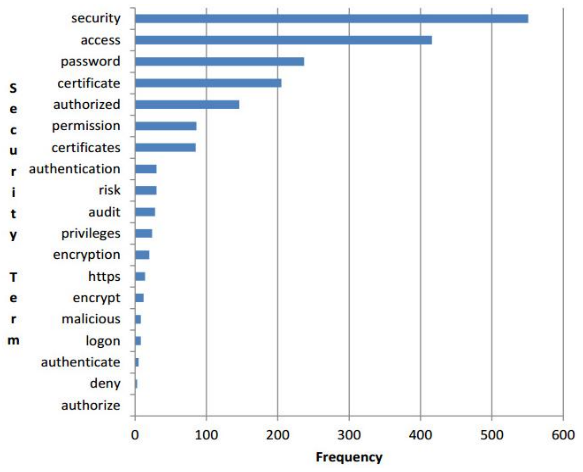

Fig. 2: Security term frequency from POS tagging

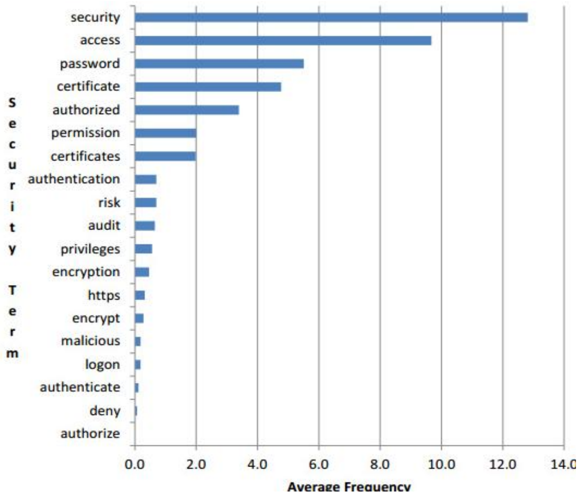

Fig. 3: Security term average frequency from POS tagging

Carrying out the identification activity requires that the remaining security terms are analyzed to identify candidate security goals (CSG). Analysis from one of the SRS documents reveals the following CSG's:

CSG-1: The application will also allow for remote access through a firewall via outside telecommunications networks by legal users.

CSG-2: The logon screen shall request user name and corresponding password.

CSG-3: For system login purposes, the hash function shall also be used to encrypt user passwords.

4.2.2. Categorize security goals based on security principle

Each of the CSG's is categorized based on security principle. Security principles (SP) are commonly known as the CIA triad which stands for confidentiality, integrity and availability. Common definitions for the security principles are:

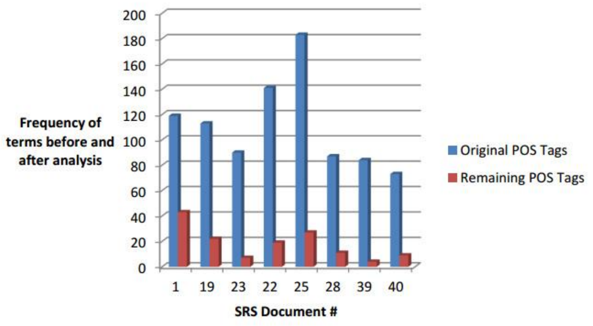

Fig. 4: Comparison of original and remaining term frequency

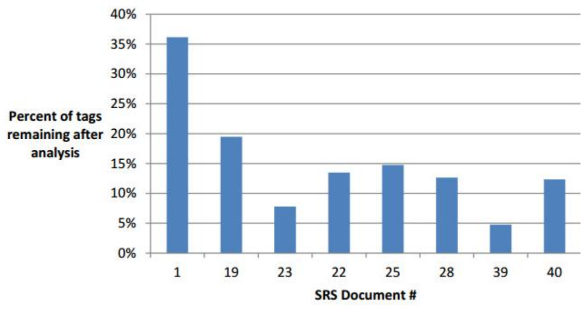

Fig. 5: Average security term frequency after reduction

SP-1: Confidentiality: protect against unauthorized disclosure of information.

SP-2: Integrity: protect against unauthorized modification or destruction of information.

SP-3: Availability: protect against disruption of access to or use of information of an information system.

CSG's can be categorized with multiple security principles. If no security principles can be applied, CSG would be rejected.

CSG-1: SP-1, SP-2

CSG-2: SP-2

CSG-3: SP-2

4.2.3 Understand stakeholder goals and develop preliminary security requirements

Stakeholder goals are elicited for each of the categorized CSG's and preliminary security requirements (PRS) are developed.

PSR-1: The system shall protect confidentiality and integrity of data by allowing remote access through a firewall only to authorized users.

PSR-2: The system shall protect integrity of data by requesting a user name and password prior to access.

PSR-3: The system shall protect confidentiality of user passwords by encrypting passwords.

Table 9: Security requirements elicitation template

Security Requirements Elicitation				
Document Name:				
	Document ID :	19	Original tag count	113
	Project ID :		Final tag count	43
1. Identify candidate security goals				
Candidate Security Goals (CSG)				
	CSG-1	The application will also allow for remote access through a firewall via outside Telecommunication networks by authorized users		
	CSG-2	The logon screen shall request user name and corresponding password		
	CSG-3	For system login purposes, the hash function will also be used to encrypt user password		
2. Categorize security goals based on security principle				
	Apply security principle(s) to CSG			
	CSG-1	SP-1, SP-2		
	CSG-2	SP-2		
	CSG-3	SP-2		
3. Understand stakeholder goals and develop preliminary security requirements				
	Preliminary Security Requirement (PSR)			
	PRS-1	The system shall protect confidentiality and integrity of data by allowing remote access Through a firewall only to authorized users		
	PRS-2	The system shall protect integrity of data by requesting a user name and password prior To access		
	PRS-3	The system shall protect confidentiality of user passwords by encrypting passwords		
4. Prioritize preliminary security requirements				
	PSR	Effects	FMEA RPN	Accept/Reject
	PRS-1	Data Stolen	189	Accept
	PRS-2	Data viewer	84	Accept
	PRS-3	Password compromised	162	Accept
Prioritized Security Requirements (SR)				
	SR-1	The system shall protect confidentially and integrity of data by allowing remote access Through firewall… only to authorized users		
	SR-2	The system shall protect integrity of data by requesting a user name and password prior To access.		
	SR-3	The system shall protect confidentiality of user passwords by encrypting passwords.		
Notes		All of the identified requirements should be reclassified as security requirements		

4.2.4 Prioritize preliminary security requirements

FMEA analysis is performed on for each PSR. Potential failure modes and effects are identified. The failure modes and effects are written in general terms for ease of understanding and quick analysis. Security Requirement template is elaborated in Table 9.

5. Conclusion

Resulting security requirements are integrated into SRS documents and security requirements repository enables rapid reuse of developed requirements. Key elements of the elicitation solution are (1) identifying security goals, (2) categorizing goals by security principle, (3) understanding stakeholder goals to develop preliminary requirements and (4) prioritizing security requirements for inclusion into the SRS document. Stakeholder roles, input artifacts, techniques and output artifacts are defined for each phase of the solution. The solution is flexible in order to accommodate the needs of small, agile software development organizations but outlines a basic structure that can be easily implemented. The solution takes place at the earliest phase of the software development process during requirements elicitation in order to reduce cost and rework at later stages of development. The main purpose of this research is to integrate POS tagging for enhancing the security requirements elicitation approach. During evaluation of the solution, we observed that additional work in POS tagging is needed. An automated tool for this approach is needed without the involvement of security expert.

References

[1] M. Beznosov and L. Kruchten, “Get ready for agile methods with care” Computer, vol. 1, pp. 64-69. 2006.

[2] D. Firesmith, ”Developing secure software and systems”, IEC Network Security: Technology Advances, Strategies, and Change Drivers, Chicago, International Engineering Consortium, vol. 1, pp. 10, 2003.

[3] I. Alexander, “Misuse cases help to elicit non-functional requirements”, Computing & Control Engineering Journal, vol. 14, pp. 40-45, 2003.

[4] B. Barnum and S. Sethi, “A software security engineering: A guide for project managers”, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, vol. 1, pp. 1, 2013.

[5] J. Moffett and B. Nuseibeh, “Security requirements engineering: A framework for representation and analysis, IEEE Transactions on Software Engineering”, vol. 1, pp. 133-153, 2008.

[6] D. Dave and R. Lawrence, “SQUARETool”, http://www.cert.org/sse/square/square-tool.html. 2003.

[7] D. Firesmith, "Developing Secure Software and Systems”, IEC Network Security: Technology Advances, Strategies, and Change Drivers, Chicago, International Engineering Consortium, vol. 1, pp. 1, 2003.

[8] K. Limerick, F. Ireland and M. Morisio, “Software engineering for security: a roadmap”, Paper presented at the Proc. of the Conference on The Future of Software Engineering, vol. 1, 2004.

[9] J. Mylopoulos and N. Zannone, “Requirements engineering meets trust management: Model, methodology and reasoning”, Deptt. of Information and Communication Technology, University of Torento, Canada, August 29 – September 2, 2005.

[10] J. Hafterson, ”Security Requirements Engineering: A Framework for Representation and Analysis” , vol.1, pp.18, 2008.

[11] C.G. Harris, “The usage-centric security requirements engineering (USeR) method”, Information Assurance Workshop, vol. 1, pp. 2, July, 2012

[12] M. Hu and B. Liu, “Detecting deceptive opinion spam using human computation”, Paper presented at the Proc. of the 4th Human Computation Workshop (HCOMP'12), vol. 1, pp. 2, 2004.