International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8, Issue-2S11, September 2019

SecureMEReq: A Tool Support to Check for Completeness of Security Requirements

Nuridawati Mustafa, Massila Kamalrudin, Safiah Sidek

Abstract Quality security requirements help secure software development to succeed. While considerable research can be discovered in the field of demands elicitation, less attention has been paid to the writing of full security specifications. The demands engineers (REs) are still challenged and tedious in implementing and reporting full safety needs derived from Natural language. This is due to their tendency to misunderstand the real needs and the security terms used by inexperienced REs leading to incomplete security requirements. Motivated from these problems, we have developed a prototype tool, called SecureMEReq to improve the writing of complete security requirements. This tool provides four important key-features, which are (1) extraction of template-based components from client-stakeholders; (2) analysis of template-based density from SRCLib; (3) analysis of requirements syntax density from SecLib; and (4) analysis of completeness prioritization. To do this, we used our pattern libraries: SecLib and SRCLib to support the automation process of elicitation, especially in writing the security requirements. Our evaluation results show that our prototype tool is capable to facilitate the writing of complete security requirements and useful in assisting the REs to elicit the security requirements.

KEYWORDS: Tool security requirements, template-based approach, security requirements completeness, template-based density, syntax density.

I. INTRODUCTION

The most important aspect of requirements quality is requirements completeness. Evidences show that one of the root causes of safety incidents is due to incompleteness of safety requirements . Incomplete requirements will interrupt the reliability and accuracy of the prediction system. The lack of requirement completeness causes uncertainty of the project foundations . Research by shows that as many as 50% of all accidents are due to requirements problems and many of these accidents are caused by missing or incomplete requirements.

Over the time, more and more software-intensive systems have been given safety-related responsibilities. Considering software safety is directly influenced by requirements completeness, it is essential to have a complete requirements so that the stakeholder’s needs are readily found and understood, and mistakes and misunderstandings are avoided.

The incomplete requirements generate poor requirements and lack of clarity. For that reason, it will contribute to eliciting incomplete security requirements. Low clarity and incomplete security requirements therefore lead to the inability to create secure software. In addition, the issues with the method of obtaining and compiling security specifications are complex and tedious. In order for the stakeholders to have consistent security demands, the requirments engineer (RE) needs security expertise. Furthermore, the specifications requirements are acquired from natural language. This causes issues for REs to generate stakeholder security demands because the actual needs and the security conditions used are inappropriate to deal with. The study in[5] shows that most clients refuse to understand the security that their systems require. In addition, the captured security requirements do not fulfilled the standards requirements such as NIST, ISO and Common Criteria. All these problems lead to the production of incomplete security requirements.

This paper is organized as follows. After the introduction, we discuss the related works in background and motivation in Section 2. This is followed by Section 3, which we discuss the overview of our proposed approach for security requirements elicitation. Then, we discuss the tool usage examples. Next, we present the results and discussion to evaluate the effectiveness of our approach in Section 5. Finally, we end this paper with conclusion and future works.

II. BACKGROUND AND MOTIVATIONS

Based on our investigation in Table 1, there are thirteen current works done in writing security requirements with nine different techniques contributions. Despite these efforts, only four researchers developed the tools, but none were found to provide with completeness validation. There requirements engineers are still facing with incomplete security requirements and none were found in security requirements contexts, specifically on the density and syntax level.

Revised Version Manuscript Received on September 16, 2019.

Nuridawati Mustafa, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal, Melaka, Malaysia.
(email: nuridawati@utem.edu.my)
Massila Kamalrudin, Universiti Teknikal Malaysia Melaka, Hang Tuah **Jaya, Durian Tunggal**, Melaka, Malaysia. Institute of Technology Management and Entrepreneurship, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal, Melaka, Malaysia.
(email: massila@utem.edu.my)
Safiah Sidek, Institute of Technology Management and Entrepreneurship, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal, Melaka, Malaysia.
(email: safiahsidek@utem.edu.my)

Table 1: Security Requirements Elicitation Techniques and Tool

Year	Initiative	Contributions					Validation		
		Tool	Technique	Security Requirements Template					
				M	SA	FA	C_R	C_{SC}	C_M
2017	[6]		Misuse Case Oriented Quality Requirements (MCOQR) metrics				√		
2016	[7][8][9] [10][11]	√	Discovering Goals for Security (DIGS) Framework		√				
2016	[12]	√	Essential Use Cases (EUCs) and Essential User Interface (EUI) models				√	√	
2015	[13]		Risk Management and PBSE Approach						
2014	[14]		Problem Frames						√
2014	[15]	√	Analysis Patterns				√		
2014	[16]		Essential Use Cases (EUCs) model.				√	√	
2012	[17]		Spiral Process Model						√
2010	[18]	√	CC, HeRA, and UMLsec.						
Total	13	3	9		1		4	2	2

III. OUR APPROACH

Considering from the gaps found in the previous section, our research aims to propose a security requirements template-based approach to improve the density of requirements that can lead in writing complete security requirements. We proposed an overall automated approach, as illustrated in Figure 1. This approach composes of six (6) main steps, which are shown in Table 2.

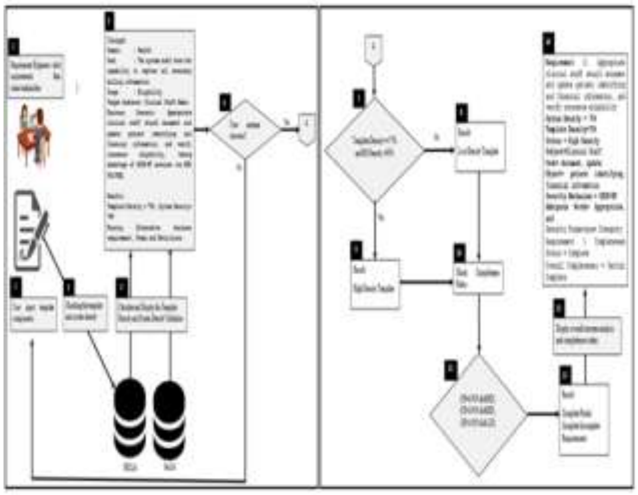


Figure 1: An Overview of SecureMEReq Approach

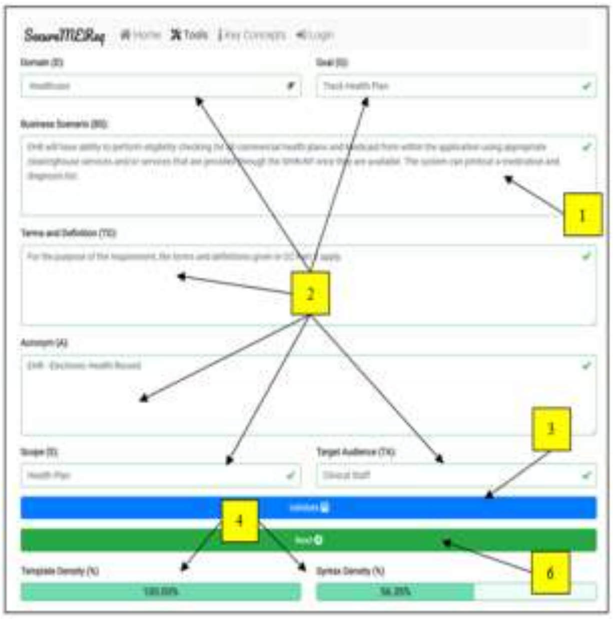
Table 2: The process of SecureMEReq

Step	Process	Descriptions
1	1	RE requires customers to meet the demands during the gathering of requirements. The security specifications gathered will take the form of textual specifications for natural languages.
2	2-3	In the tool editor, RE will then enter the template-based parts. Two elements are used to analyze the textual security demands. First, the SRCLib template-based elements are analyzed in order to identify elements which are to be included in the framework. Second, the company scenario is analyzed on the basis of syntax analysis which identifies elements that need to be included in the security requirements classification framework. The search of the security keywords and mechanism through SecLib is performed with a keyword matching system.
3	4	Next, the tool calculates the density for template-based components and business scenario based on our security requirements pattern library SRCLib and SecLib.
4	5	For further checking, RE can view the analysis for each security requirement sentence structure.
5	6-13	The tool displays the status of the security requirements density whether it is high or low density. The subject, verb, object, security mechanism, ambiguity words and security properties will be displayed. Besides, the completeness status for each requirement is displayed. Additionally, the missing components will also be highlighted. RE will have the ability to edit/update the input and choose the option whether to edit/update the original input.
6	14	Finally, the tool displays the overall result of the security requirements completeness level either Complete/Partial Complete/Incomplete. The results will help RE to give early status on the level of completeness for their written security requirements.

IV. TOOL USAGE EXAMPLE

We have developed a prototype tool, called SecureMEReq. The SecureMEReq was developed using PHP programming language and adopts Model-View-Controller (MVC) design pattern and three-tier architecture. Our tool provides the (1) extraction of template-based components from client-stakeholders, (2) analysis of template-based density from SRCLib (3) analysis of requirements syntax from SecLib and (4) completeness prioritization. We demonstrated the features of our tool using the user persona as per described below:

Hardy, a requirements engineer would like to validate the requirements provided by the client-stakeholder using SecureMEReq. He sits with Lew, who is the project manager to validate the requirements, which he captured earlier.



Domain (D): Healthcare

Goal (G): Track Health Plan

Business Scenario (BS):

Terms and Definition (TD): For the purpose of this requirement, the terms and definitions given in DC apply.

Acronym (A): EHR - Electronic Health Record

Scope (S): Health Plan

Target Audience (TA): Clinical Staff

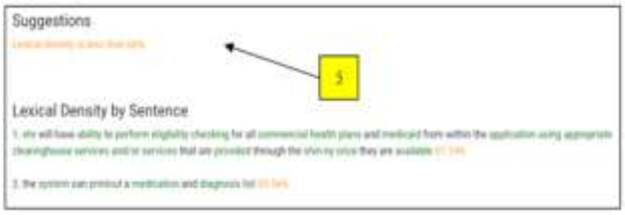
Template Density (%): 100.00%

Syntax Density (%): 56.25%

Figure 2: User Interface of SecureMEReq in used

As shown in Figure 2, he inserts the requirements in the form of business scenario in the text editor (1). Besides, he also needs to insert several template-based components, which are the domain, goal, terms and definitions, acronym, scope and target audience as in (2). From there, he clicks the "Calculate" button to generate the density for template-based components and syntax density (3). Then, Hardy can view the template-based density and syntax density results (4). If Hardy is unhappy with the result, he can edit/update the inputs and recalculate, if needed. Besides, Hardy and Lew can review the "Suggestion" and "Lexical Density by Sentence" as in Figure 3 (5). In order to allow Lew to get better understanding of the requirements structure, he then clicks the "Next" button to review the analysis of security requirements (6).

In Figure 4 , Hardy and Lew can review the analysis for each requirement completeness. Here, Hardy and Lew can validate each requirement density status and structure, such as the Subject, Object, Verb, Security Mechanism, Ambiguous Words used, Security Properties and the completeness status for each requirement (7)(8). They can also view the examples of each component if needed (11). Finally, he can view the overall completeness for all requirements (9) and Lew can decide whether to proceed with the requirements or amend it (10).


Suggestions

[illegible]

Lexical Density by Sentence

1. [illegible]

2. [illegible]

Figure 3: Template-Based and Syntax Density Embedded in SecureMEReq

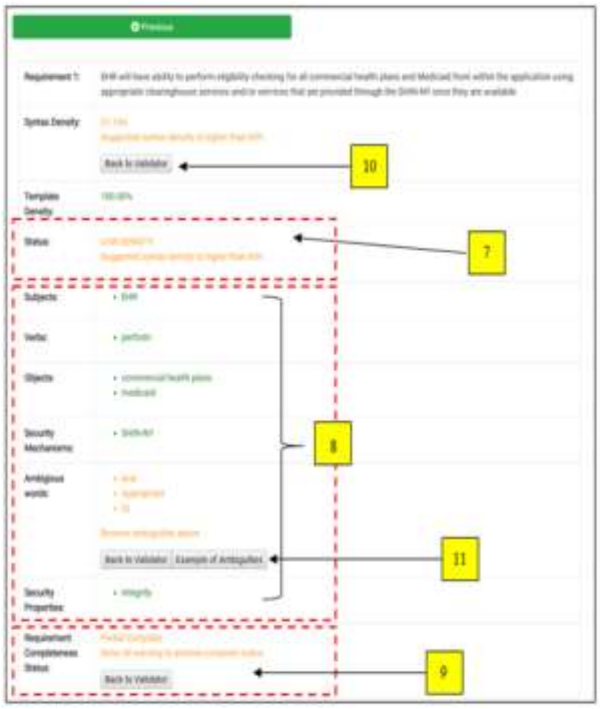
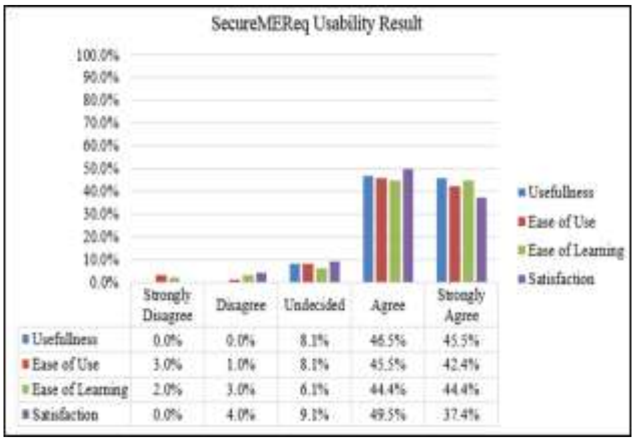


Figure 4: Security Requirement Completeness Prioritization in SecureMEReq

V. RESULT AND DISCUSSIONS

To evaluate our approach and tool, we conducted usability tests. The purpose of the usability tests was to evaluate the usefulness of our tool's features for extraction of template-based components from client-stakeholders, syntax checking template from pattern library and completeness analysis. This usability test was conducted with novice participants (novice RE) represented by 33 undergraduate students from the course of Software Validation and Verification. These students were majoring in Software Engineering in Universiti Teknikal Malaysia Melaka. Basically, they have sufficient background and knowledge to understand about software requirements. The purpose of selecting novice participants is to investigate how they explore and use the tool.

Figure 5 shows the outcomes for the questionnaire-based usability criteria. Figure 5 shows the outcomes of the usability research for our tool's template-based strategy. The findings are very positive, with the participants ' strong agreement on the usefulness of the instrument (92% highly agree to or agree on its effectiveness), its ease of use (more than 87%), its easy learning (more than 88%) and its satisfaction (86.9%). Only a small amount (less than 10%) of the participants had not made a decision or disagreed with the utility of the tool. In general, the results of usability show that our prototype tool is helpful, simple to use and simple to learn. When using the tool, users have also shown their increased amount of satisfaction.



	Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree
Usefullness	0.0%	0.0%	8.1%	46.5%	45.5%
Ease of Use	3.0%	1.0%	8.1%	45.5%	42.4%
Ease of Learning	2.0%	3.0%	6.1%	44.4%	44.4%
Satisfaction	0.0%	4.0%	9.1%	49.5%	37.4%

Figure 5: SecureMEReq Usability Study Result

VI. CONCLUSION AND FUTURE WORKS

In summary, we have presented our prototype tool, called SecureMEReq that provides the (1) extraction of template-based components from client-stakeholders; (2) analysis of template-based density from SRCLib; (3) analysis of requirements syntax from SecLib; and (4) analysis of completeness prioritization. The results of our evaluation indicate that our prototype tool can make it easier to compile full security specifications and help requirement engineers to elicit safety demands.For future research, we will extend the evaluation of our tool by evaluating the efficacy of our approach in terms of completeness. We will conduct completeness testing to evaluate the completeness of eliciting security requirements by comparing manual elicitation with our prototype tool. This is to determine the ability of our SecureMEReq tool to produce complete security requirements. We firmly think that our template-based strategy can help to increase clarity about the requirements which will enable secure software development to be complete and successful.

VII. ACKNOWLEDGEMENTS

We would like to express our gratitude to the university, UTeM and MoHE for the research funding: FRGS/1/2015/ICT01/FTMK/02/ F00325.

REFERENCES

1. P. O. Antonino, M. Trapp, and A. Venugopal, "Automatic Detection of Incomplete and Inconsistent Safety Requirements," Apr. 2015.
2. A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity As A Resource To Disclose Tacit Knowledge," in IEEE 23rd International Requirements Engineering Conference (RE 2015), 2015, pp. 26–35.
3. D. Firesmith, "Are Your Requirements Complete?," J. Object Technol., vol. 4, no. 1, pp. 27–43, 2005.
4. A. Banerjee, M. Sharma, C. Banerjee, and S. K. Pandey, "Research On Security Requirements Engineering: Problems And Prospects," MATRIX Acad. Int. Online J. Eng. Technol., vol. III, no. 1, pp. 32–35, 2015.
5. M. Kamalrudin, N. Mustafa, and S. Sidek, "A Preliminary Study: Challenges In Capturing Security Requirements And Consistency Checking By Requirement Engineers," J. Telecommun. Electron. Comput. Eng., vol. 10, no. 1–7, pp. 5–9, 2017.
6. C. Banerjee, A. Banerjee, and S. . Sharma, "Use Case And Misuse Case In Eliciting Security Requirements : MCOQR Metrics Framework Perspective," Int. J. Mod. Electron. Commun. Eng., vol. 5, no. 3, pp. 35–39, 2017.
7. M. Riaz, J. Stallings, M. P. Singh, J. Slankas, and L. Williams, "DIGS – A Framework for Discovering Goals for Security Requirements Engineering," in ACM International Symposium on Empirical Software Engineering and Measurement (ESEM 2016), 2016, p. 35.
8. M. Riaz, J. King, J. Slankas, L. Williams, F. Massacci, C. Quesada-lópez, and M. Jenkins, "Identifying the Implied: Findings from Three Differentiated Replications On The Use Of Security Requirements Templates," Empir. Softw. Eng., vol. 22, no. 4, pp. 2127–2178, 2016.
9. M. Riaz, S. Elder, and L. Williams, "Systematically Developing Prevention, Detection, and Response Patterns for Security Requirements," in Requirements Engineering Conference Workshops (REW), 2016, pp. 62–67.
10. M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden In Plain Sight: Automatically Identifying Security Requirements From Natural Language Artifacts," in IEEE 22nd International Requirements Engineering Conference, RE 2014, 2014, pp. 183–192.
11. M. Riaz, J. Slankas, J. King, and L. Williams, "Using Templates To Elicit Implied Security Requirements From Functional Requirements - A Controlled Experiment," in ACM The 8th International Symposium on Empirical Software Engineering and Measurement, ESEM 2014, 2014, p. 22.
12. N. Yusop, M. Kamalrudin, S. Sidek, and J. Grundy, "Automated Support to Capture and Validate Security Requirements for Mobile Apps," in Communications in Computer and Information Science, vol. 671, no. November, 2016, pp. 97–112.
13. A. Motil, B. Hamid, A. Lanusse, J.-M. Bruel, A. Motii, B. Hamid, A. Lanusse, and B. Jean-Michel, "Guiding The Selection Of Security Patterns Based On Security Requirements And Pattern Classification," in ACM The 20th European Conference on Pattern Languages of Programs, EuroPLoP 2015, 2015, p. 10.
14. H. El-Hadary and S. El-Kassas, "Capturing Security Requirements For Software Systems," J. Adv. Res., vol. 5, no. 4, pp. 463–472, Jul. 2014.
15. K. Beckers, I. Côté, and L. Goeke, "A Catalog of Security Requirements Patterns For The Domain of Cloud Computing Systems," in ACM The 29th Symposium On Applied Computing, 2014, pp. 337–342.
16. S. Yahya, M. Kamalrudin, S. Sidek, and J. Grundy, "Capturing Security Requirements Using Essential Use Cases (EUCs)," in The First Asia Pacific Requirements Engineering Symposium, APRES 2014, 2014, vol. 432 CCIS, pp. 16–30.
17. P. Salini and S. Kanmani, "Elicitation of Security Requirements for E-Health System by Applying Model Oriented Security Requirements Engineering (MOSRE) Framework," in ACM The Second International Conference on Computational Science, Engineering and Information Technology, CCSEIT 2012, 2012, pp. 126–131.
18. S. H. Houmb, S. Islam, E. Knauss, J. Jürjens, and K. Schneider, "Eliciting Security Requirements And Tracing Them To Design: An Integration Of Common Criteria, Heuristics, and UMLsec," Springer Requir. Eng., vol. 15, no. 1, pp. 63–93, Mar. 2010.