International Journal of Software Engineering and Its Applications
Vol. 9, No. 10 (2015), pp. 131-142
http://dx.doi.org/10.14257/ijseia.2015.9.10.13

A Study on the Secure Software Development Life Cycle for Common Criteria (CC) Certification

Min-gyu Lee[1], Hyo-jung Sohn[2], Baek-min Seong[3] and Jong-Bae Kim[4*]

[1,2,3,4]Graduate School of Software, Soongsil University, Seoul 156-743, Korea*
[1]marse101@naver.com, [2]hyojung.sohn@gmail.com, [3]feeling127@naver.com, [4]kjb123@ssu.ac.kr*

Abstract

Common Criteria (CC) is a global standard for the information technology security evaluation of IT products. IT products must obtain at least a certain Evaluation Assurance Level (EAL) to be supplied to governmental institutions. The general software development life cycle (SDLC) does not provide guidelines for the removal of the weaknesses in the software development process for CC certification, and software applications developed with security weaknesses that have not been removed can lead to fatal situations. At present, CC provides only security certification for the target of evaluation (TOE) and does not provide guidelines regarding the secure software development life cycle (SSDLC), which considers the weaknesses in the development process. If a TOE that has been specialized for CC is developed by SSDLC, both evaluators and developers can undertake CC certification from an objective standpoint. This paper presents an SSDLC that is appropriate for CC by discerning weaknesses, by referring to MS-SDL, McGraw's TouchPoints, and OWASP CLASP based on the weaknesses provided by CWE (Common Weakness Enumeration). The findings of this study can be used as guidelines for the development process of weakness-free software applications by developers aiming for CC certification.

Keywords: *Common Criteria, CC, SDLC, SSDLC, CWE, weakness, vulnerability, MS-SDL, CLASP, TouchPoints*

1. Introduction

Common Criteria (CC) is a globally standardized (ISO/IEC 15408) set of information technology security evaluation criteria for IT products manufactured worldwide [1]. The IT products to be introduced by governmental organizations and public institutions must acquire at least a certain Evaluation Assurance Level (EAL) of CC to be delivered. According to the materials reported by NIST, 92% of the vulnerabilities of IT products are caused by applications rather than by networks [2]. On the other hand, as the general software development life cycle (SDLC) does not suggest guidelines for eliminating the weaknesses in development process, so development outcomes, of which weakness was not eliminated, can sometimes cause a critical situation. Currently, when a number of software-related small and medium-sized companies develop software for CC certification, they usually hire CC experts when the development process has been completed and not from the software's initial development, due to cost-related issues. At the Software Engineering Institute of Carnegie Mellon University, 70% of the weaknesses occurring during a design process error were reported. When it was unable to eliminate these weaknesses during the software design phase, 30-fold higher costs occurred in the maintenance phase [3], as shown in Figure 1.

[4*] Corresponding author. Tel. : +82-10-9027-3148.
Email address: kjb123@ssu.ac.kr(Jong-Bae Kim).

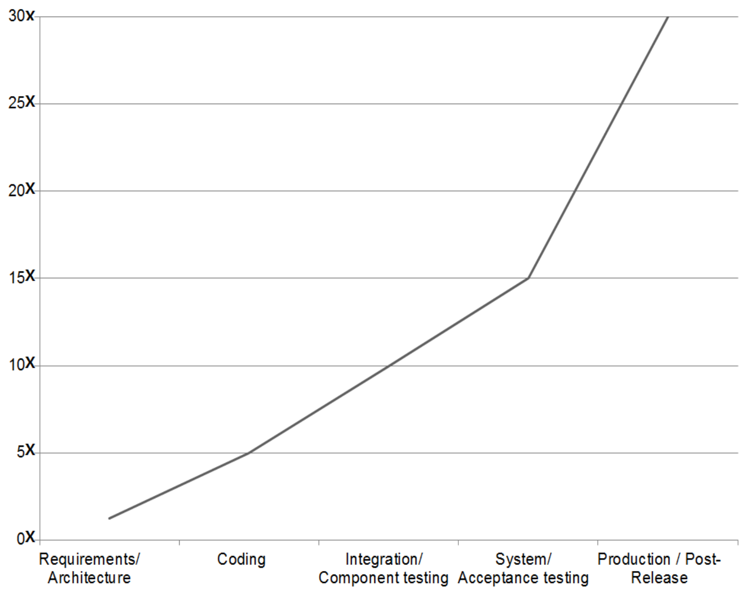


Figure 1. Relative Cost of Fixing based on Time of Detection

In addition, CC currently performs security certification only for target of evaluation (TOE), and does not suggest a guideline related to the secure software development life cycle (SSDLC), which can analyze the weaknesses during the software development process. This paper suggests that SSDLC be used to develop a TOE suitable for CC based on the weakness provided by Common Weakness Enumeration (CWE), by determining the vulnerabilities and weaknesses in reference to MS-SDL, OWASP Comprehensive, Lightweight Application Security Process (CLASP), and TouchPoints. The results of this study can be utilized as guidelines for software developers aiming for CC certification, for the process of developing software with eliminated weaknesses.

2. Related Works

2.1. Analysis of MS-SDL, CLASP, and TouchPoints

Up front, it has been seen that if there are vulnerabilities that have not been eliminated in the SDLC design, implementation, and testing phases, the costs will increase exponentially with time. One weakness is accompanied by multiple vulnerabilities. Therefore, eliminating the weaknesses should be prioritized. As such, in this paper, the removable items from the CWE Weakness List (Ver. 2.8) are mapped by item in the design, implementation, and testing phases, using the study [4] data obtained by Bart, who conducted a comparative study of MS-SDL, CLASP, and TouchPoints.

The comparison data presented by Bart were based on the detailed items in MS-SDL, CLASP, and TouchPoints. The total number of items in the design, implementation, and testing phases is 81.

2.2. CWE List

For the CWE Weakness List [5], the Weaknesses Introduced during Design (CWE-701) of the Development Concept (CWE-699) will be used as a weakness list. The weaknesses presented are shown in Table 1.

Table 1. Weaknesses Introduced during Design (CWE-701)

CWEs in this view	Total CWEs		
Total	383	out of	1,003
Views	0	out of	32
Categories	3	out of	244
Weaknesses	377	out of	719
Compound Elements	3	out of	8

The view in Table 1 is a list of the weaknesses classified based on a specific point of view, such as that of the users or of the developers and the categories are CWE entries containing a set of other entries that share a common characteristic. The weaknesses are classified into base, class, and variant. The base is a weakness that is not dependent on a specific technology, language, *etc.*; the class is a weakness that is described in a very abstract fashion, typically independent of any specific language or technology; and the variant is a weakness described with a very low level of details, typically limited to a specific language or technology. The compound element is a weakness that is not a single weakness but is caused by multiple weaknesses [6]. Therefore, the weaknesses that can occur in the design phase total 380, except for the view and categories.

For the weakness list in the implementation phase, the Weaknesses Introduced during Implementation (CWE-702) of Development Concept (CWE-699) will be used.

Table 2. Weaknesses Introduced during Implementation (CWE-702)

CWEs in this view	Total CWEs		
Total	604	out of	1,003
Views	0	out of	32
Categories	4	out of	244
Weaknesses	596	out of	719
Compound_Elements	4	out of	8

Similarly, the weaknesses that can occur in the implementation phase total 600, except for the view and categories.

Table 3. Weaknesses Introduced during the Test

CWEs in this view	Total CWEs		
Total	697	out of	1,003
Views	0	out of	32
Categories	6	out of	244
Weaknesses	683	out of	719
Compound Elements	4	out of	8

Although there are some parts that overlap in the weakness lists in the design and implementation phases, assuming that the weaknesses will be removed in the implementation phase if a mistake occurs in the removal of weaknesses in the design phase, the duplicates were not removed in the sense of complement. In the testing phase, the duplicates were removed from the two CWE lists in the design and implementation phases mentioned above. As all the existing vulnerabilities must be removed, the analysis was conducted based on the 687 total weaknesses, as shown in Table 3.

2.3. CC Analysis

2.3.1. Weakness Analysis for the Security Functional Requirements: In CC Part 2, Secure Functional Requirements (SFR) provides a total of 11 security functional classes.

This is a set of functional components that can be used as a standard to define the security features and security mechanisms of TOE [7]. For the weaknesses from the security functional requirements that can be removed in the design and implementation phases, the methodology [8] using the security features (CWE-254) was utilized. The weaknesses presented are shown in Table 4. The listed weaknesses total 110 in the design phase and 75 in the implementation phase. The security features refer to the functions that can be tracked in the security functional requirements defined in CC Part 2. Therefore, this study did not consider the non-security features but considered only the weaknesses that can be tracked using SFR.

Table 4. Security Features (CWE-254)

CWEs in this view	Total CWEs		
Total	186	out of	1,003
Views	0	out of	32
Categories	1	out of	244
Weaknesses	185	out of	719
Compound Elements	0	out of	8

2.3.2 Weakness Analysis for the Security Assurance Requirements: Security Assurance Requirements (SAR) is a set of assurance components that can be used as a standard to define the assurance level of TOE. The ultimate goal of CC certification is to acquire an Evaluation Assurance Level (EAL). Therefore, the software developers shall develop software with the goal of obtaining an EAL. EAL is formed from step 1 in step 7. EAL1 is a functional test while EAL2 is a structural test. EAL3, a systematic test and inspection, evaluates TOE, and EAL4 evaluates the systematic design, test, and review. EAL5 evaluates the semi-standardized design and test, and EAL6 evaluates the semi-standardized design verification and test. Finally, EAL7 evaluates TOE in the standardized design verification and test [9]. In CC, when the software developer tries to acquire a TOE with a level higher than EAL4, he must submit the source code to the rating institution. If one gets a certification with a TOE below EAL3, only the test (ATE) and penetration test (ADV_VAN) are done. Source code analysis, the most important analysis method for discovering weaknesses, is conducted from an evaluation rating of above EAL4, but it is explicitly stated in CC Standard that even source code analysis is not required to be conducted by evaluators. Similarly, this vulnerability analysis is also not a requirement on the part of evaluators. Thus, the items [10] that are running can be utilized to remove the weaknesses in CC.

3. SSDLC Research Method for Specialized CC

3.1. Research Process

This paper constructs maps to find the weaknesses to be removed by comparing the comparison items of SSDLCs that can remove the vulnerabilities described above with the predefined CWE list. It then constructs maps to find the weaknesses that can be removed based on the CWE list, with CC security functional requirements (SFR) and security assurance requirements (SAR). Then it proposes an SSDLC by determining the correlation of the weaknesses that were found in the two previous phases [11].

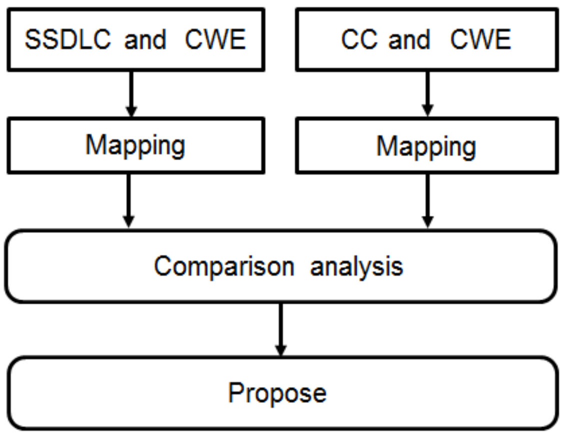


Figure 2. Research Process

3.2. SSDLC Comparison Items and CWE List Mapping

Table 5. CWE/SANS Top 25 and SSDLC Comparison Table Mapping

CWE-ID	NAME	No. of Vulnerabilities	Weakness Detection Method: Design Phase	Weakness Detection Method: Implementation Phase	Weakness Detection Method: Test Phase
CWE-89	SQL Injection	7		• Security analysis tools for the source management process • Automated source-level security review • Manual source code inspection	• Fuzz testing • Unit testing • Penetration testing
CWE-120	Classic Buffer Overflow	5		• Security analysis tools for the source management process • Automated source-level security review • Manual source code inspection	•Unit testing •Fuzz testing •Penetration testing
CWE-306	Missing Authentication	3	• Threat modeling	• Security analysis tools for the source management process • Automated source-level security review • Manual source code inspection	•Unit testing •Fuzz testing •Penetration testing

As the mapping data for all the weaknesses could not be added, an example of the Top 25 Most Dangerous Software Errors [12] that CWE/SANS announced was given. The above reference shows the top 25 kinds of the most dangerous weaknesses. The results mapping the weaknesses in the above material to the SSDLC comparison table are shown in Table 5. First, SQL injection shows that there are seven kinds of vulnerabilities to be derived, and that there is no way of finding the weaknesses at the design phase. On the other hand, for the methods that can find the weaknesses in the implementation phase, there are the security analysis tools for the source management process, automated source-level security review, and manual code inspection. As for the method of finding the weaknesses for SQL injection in the testing phase, there are the fuzz and unit testing methods.

The above table confirms all the weaknesses that can be removed from SSDLC, which could serve as a cornerstone for the weaknesses that could be found in CC in the future, as well as a comparative material.

3.3. Common Criteria and CWE List Mapping

As shown in Section 3.2, CC and CWE list mapping was done using the 1st, 3rd, and 5th ranks of CWE/SANS Top 25. For CC comparison, the SFR that can become a standard for defining the TOE's security functions and the security mechanisms described in Section 2.3, and the SAR that can become a standard for defining the TOE assurance level, were utilized. On the other hand, source code analysis and vulnerability analysis are not mandatory for the assurance requirements, but are used when required by an evaluator.

Table 6. CWE/SANS Top 25 and CC Comparison Table Mapping

CWE-ID	NAME	No. of Vulnerabilities	Weakness Detection Method
CWE-89	SQL Injection	7	• SAR
CWE-120	Classic Buffer Overflow	5	• SAR
CWE-306	Missing Authentication	3	• SFR • SAR

3.4. Comparison Analysis

In this section, the weaknesses that CC and SSDLC can remove were compared using the data mapped in Sections 3.2 and 3.3. The organized results are shown in Table 7. Such table shows the detailed items of the weaknesses that were found using SAR and SFR that can be removed in the design, implementation, and testing phases of SSDLC.

The weaknesses that were found using SAR were CWE-89, CWE-120, and CWE-306, and the items whose applicable weaknesses can be removed from SSDLC are presented. For the items whose security vulnerabilities can be removed from SSDLC to satisfy SAR, except the duplicates, there are one method in the design phase, four methods in the implementation phase, and three methods in the testing phase. Similarly, to meet SFR, there are one, three, and three methods in the design, implementation, and testing phases, respectively.

Table 7. Comparison of the Mapped Data

		CWE	SSDLC		
		CWE ID	Design	Implementation	Test
CC	SAR	CWE-89: SQL Injection		• Security analysis tools for the source management process • Automated source-level security review • Manual source code inspection	• Fuzz testing • Unit testing • Penetration testing

		CWE-120: Classic Buffer Overflow		• Security analysis tools for the source management process • Automated source-level security review • Manual source code inspection	• Fuzz testing • Unit testing • Penetration testing
		CWE-306: Missing Authentication	• Threat modeling	• Security analysis tools for the source management process • Automated source-level security review • Manual source code inspection	• Fuzz testing • Unit testing • Penetration testing
	SFR	CWE-306: Missing Authentication	• Threat modeling	• Security analysis tools for the source management process • Automated source-level security review • Manual source code inspection	• Fuzz testing • Unit testing • Penetration testing

4. Comparison Results

4.1. Comparison Results in the Design Phase

The total number of weaknesses that can be discovered in the design phase is 60. In Figure 3, the items marked <SSDLCs> indicate the results of mapping the SSDLC items and CWE, and the items marked <SFR> are the results of mapping SFR and CWE. Threat modeling <SSDLCs> can remove 20% of the weaknesses, and threat modeling <SFR>, 15%. Attack modeling <SSDLCs> can remove 30% of the weaknesses, and attack modeling <SFR>, 13%. Furthermore, database security configuration <SSDLCs> can remove 6% of the weaknesses, and database security configuration <SFR>, 8%.

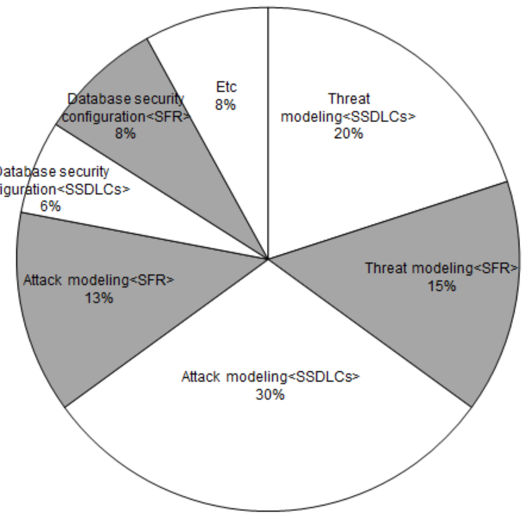


Figure 3. Analysis Results in the Design Phase

4.2. Comparison Results in the Implementation Phase

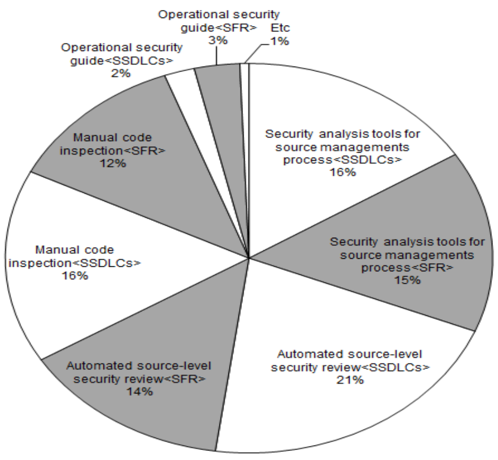


Figure 4. Analysis Results in the Implementation Phase

In the implementation phase, a total of 330 weaknesses were discovered. As shown in Figure 4, the security analysis tools for the source management process <SSDLCs> can remove 16% of the 330 weaknesses discovered in the implementation phase, and those for the source management process <SFR>, 15%. Automated source-level security review <SSDLCs> can remove 21% of the weaknesses, and automated source-level security review <SFR>, 14%. Manual code inspection <SSDLCs> can remove 16% of the weaknesses, and manual code inspection <SFR>, 12%. Lastly, operational security guide <SSDLCs> and <SFR> can remove 2 and 3% of the weaknesses, respectively.

4.3. Comparison Results in the Test Phase

As mentioned above, there are a total of 226 weaknesses that can be discovered in the testing phase. Black box testing <SSDLCs> can remove 10% of the weaknesses, and black box testing <SFR>, 6%. White box testing <SSDLCs> can remove 7% of the weaknesses, and white box testing <SFR>, 4%. Fuzz testing <SSDLCs> can remove 16% of the weaknesses, and fuzz testing <SFR>, 9%. Penetration testing <SSDLC> can remove 26% of the weaknesses, and penetration testing <SFR>, 13%. Lastly, security testing <SSDLCs> and <SFR> can remove 5 and 4% of the weaknesses, respectively.

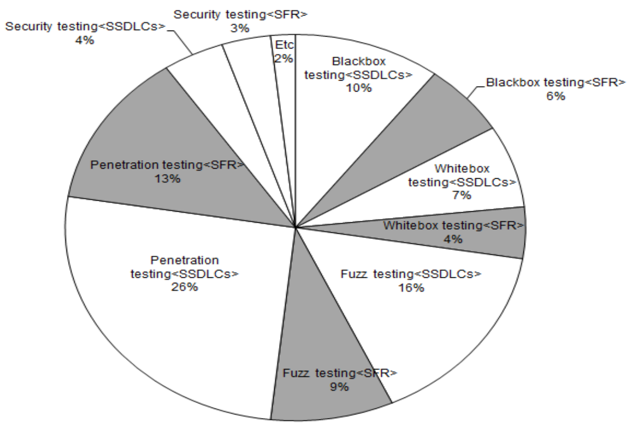


Figure 5. Analysis Results in the Test Phase

5. Proposed SSDLC

In this section, the most efficient SSDLC for CC certification is proposed using the analysis results obtained and presented in section 4. The education, project inception, requirement analysis, and release steps remove as much weaknesses as possible in the design and implementation phases. Therefore, as mentioned above, the most cost-effective policy, CLASP, is followed, considering small software companies, etc. As shown in Table 8, in the education step, the software developers are educated regarding the domain and security features of TOE.

In the project inception step, the leadership team is built or the security manager is appointed, security strategies are established, and accountability for the security issues is instituted. In the design stage, using the analysis results presented in section 4, threat modeling and attack modeling are performed for the modeling of software threats. In addition, database security configuration is performed to identify and verify the candidate configurations. Furthermore, software attack surface reduction is performed to remove the weaknesses that can occur in the design phase. In the implementation phase, the evaluator does not analyze the source code if the TOE is certified for EAL3 or a lower level.

Thus, the SSDLC items in the implementation phase can vary depending on the EAL to be certified and the situation of the developer. The items in the implementation phase include security analysis tools for the source management process, which involves the selection of tools and techniques and the determination of the analysis integration points. The implementation phase also includes automated source-level security review and manual code inspection. The operational security guides include a pre-installed configuration requirement document, a security architecture document, a security configuration mechanism document, and a significant risk and compensating control document.

In the testing phase, penetration testing can remove the largest number of weaknesses, followed by fuzz testing, black box testing, white box testing, and security testing.

CC certification performs penetration testing by default, and the analysis results can remove 26% of the weaknesses that can be discovered in the testing phase. As not every test item can be applied, however, penetration testing, fuzz testing, and black box testing can be applied in such order, according to the software developer's circumstances.

Finally, in the release phase, code signing credentials are obtained, signing targets are identified, and code signing for finding vulnerabilities from the previous studies is performed.

Table 8. SSDLC Appropriate for CC Certification

SSDLC Process	SSDLC Item
Education	• Education about software development security
Project inception	• Organization of security team • Security metrics
Requirement analysis	• User roles and resource capabilities • Resources and trust boundaries • Security-relevant requirements • Requirements for the operational environment
Design	• Threat modeling • Attack modeling • Database security configuration • Software attack surface reduction
Implementation	• Security analysis tools for the source management process • Automated source-level security review

	• Manual code inspection • Operational security guide
Test	• Fuzz testing • Penetration testing • Blackbox testing • Whitebox testing • Security testing
Release	• Code signing

6. Conclusions

In this paper, the weaknesses that can be found in the secure software development life cycle (SSDLC) based on the Common Weakness Enumeration (CWE) list, and the weaknesses that can be found in Common Criteria (CC), were mapped, and the correlations between the weaknesses found in SSDLC and those found in CC were compared and analyzed.

As a result, the numbers of weaknesses that can be removed in the design, implementation, and testing phases were expressed as percentages. Then, based on the comparison data, the tasks that must be initiated in SSDLC were selected, and an SSDLC specialized for CC was proposed. The proposed SSDLC presented items that must be performed in the education, project inception, requirement analysis, design, implementation, testing, and release phases.

The findings of this study can provide guidelines for the SSDLC in line with the target Evaluation Assurance Level (EAL) when software developers start software development to obtain CC certification.

In future studies, a better SSDLC specialized for CC certification will be presented, which can detect and remove the weaknesses outlined in this study by mapping the improvement methods for the weaknesses provided by CWE.

References

[1] D. Pierre, S. Damien and T.-H. Kim, "Some limits of Common Criteria certification", International Journal of Security and Its Applications, vol. 2, no. 4, **(2008)** October.

[2] M. M. Morana, "Building Security Into The Software Life Cycle", **(2012)**.

[3] G. Tassey, Ph.D, "The Economic Impacts of Inadequate Infrastructure for Software Testing", National Institute of Standards and Technology, **(2002)**.

[4] B. De Win, R. Scandariato, K. Buyens, J. Gre´goire and W. Joosen, "On thesecure software development process CLASP, SDL and Touchpoints compared", Information and Software Technology, vol. 51, **(2009)**, pp. 1152-1171

[5] Common Weakness Enumeration, http://cwe.mitre.org.

[6] Dr. R. A. Gandhi, Dr. H. Siy and Y. Wu, "Studying Software Vulnerabilities", The Journal of Defense Software Engineering, **(2010)**.

[7] M. Kara, "Review on Common Criteria as a Secure Software Development Model", International Journal of Computer Science & Information Technology (IJCSIT), vol. 4, no. 2, **(2012)** April.

[8] Common Criteria v3.1, Part 2:Security functional components, **(2012)**.

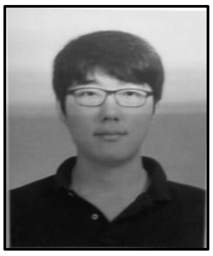
[9] J. Park and S. Kim, "How the CC Harmonizes with Secure Software Development Lifecycle", Journal of The Korea Institute of Information Security & Cryptology, vol. 24, no. 1, **(2014)** February.

[10] B. für Sicherheit in der Informationstechnik, "Guidelines for Developer Documentation according to Common Criteria Version 3.1", **(2007)**.

[11] M. G. Lee, H. J. Sohn, B. M. Seong and J. B. Kim, "A Study on Secure SDLC Specialized in Common Criteria", Proceedings of Advanced Science and Technology Letters, vol. 93, **(2015)**, pp. 19-23.

[12] MITRE, CWE/SANS Top 25 Most Dangerous Software Errors, **(2011)**.

Authors



Min-Gyu Lee, he received her bachelor's degree of Information and Telecommunication in Dongguk University (2014). And he is studying his master's degree of software engineering in the Graduate School of Software, Soongsil University, Seoul. His current research interests include Open source software and Security.

Hyo-Jung Sohn, she received her bachelor's degree of Business Administration in Soongsil University, Seoul(2006). And she is studying her master's degree of software engineering in the Graduate School of Software, Soongsil University, Seoul. Her current research interests include pen source development and management information system.

Baek-Min Seong, he received his bachelor's degree of Business Administration in Soongsil University, Seoul(2014). And he is studying her master's degree of software engineering in the Graduate School of Software, Soongsil University, Seoul. His current research interests include database.

Jong-Bae Kim, he received his bachelor's degree of Business Administration in University of Seoul, Seoul(1995) and master's degree(2002), doctor’s degree of Computer Science in Soongsil University, Seoul(2006). Now he is a professor in the Graduate School of Software, Soongsil University, Seoul, Korea. His research interests focus on Software Engineering, and Open Source Software.