ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

소프트웨어 개발보안 활동을 위한 보안메트릭 정의☆

Definition of Security Metrics for Software Security-enhanced Development

서 동 수[1*]
Dongsu Seo

요 약

2012년 시행된 소프트웨어 개발보안 제도에 힘입어 시큐어코딩은 개발자들에게 정보시스템 구축시 보안성을 향상시킬 수 있는 기법으로 인식되고 있다. 제도의 확산에도 불구하고 지속적인 보안성 향상에 대한 관리는 개발보안 제도에서 간과된 부분이기도 하다. 본 논문은 품질관리 시각에서 보안성과 관련한 시큐어코딩의 특징을 조명한다. 또한, 보안 메트릭의 제시를 통해 구현과 유지보수 활동을 자연스럽게 연계시키는 방법과 보안 메트릭을 활용하는 방법을 제안함으로써 소스코드의 관리에 도움을 주고자 한다.

☞ 주제어 : 보안메트릭, 시큐어코딩, 코드품질, 소프트웨어 개발보안

ABSTRACT

Under the influence of software security-enhanced development guidelines announced in 2012, secure coding practices become widely applicable in developing information systems aiming to enhance security capabilities. Although continuous enhancement activities for code security is important, management issues for code security have been less addressed in the guidelines. This paper analyses limitation of secure coding practices from the viewpoint of quality management. In particular this paper suggests structures and the use of software metrics from coding to maintenance phases so that it can be of help in the future by extending the use of security metrics.

☞ keyword : Security Metric, Secure Coding, Code Quality, Security-enhanced development

1. 서 론

인터넷에 의존하는 컴퓨터 활용이 심화될수록 애플리케이션 취약점으로 인해 발생하는 보안문제 역시 급격히 증가한다. 2015년도 IBM 보안침해사고 보고서에 의하면 전체 사고의 60%만이 발견 가능하며 그 중 28.3%가 DDoS공격이며, 14%가 SQL 인젝션과 관련이 있는 것으로 나타났다 [1]. 이렇듯 인터넷과 관련한 보안사고의 비중은 높아지고 있으며, 특히 웹 애플리케이션과 관련한 문제는 점점 심각한 것으로 보고되고 있다. 보안 문제로부터 안전한 소프트웨어를 만들고자 하는 시도가 다양하게 이루어지고 있으며, 국내에 정착되고 있는 소프트웨어 개발보안 역시 그러한 시도 중 하나이다[2].

[1] School of IT, Sungshin Women's University, Seongbuk-gu, 02844, Seoul, Korea
* Corresponding author (dseo@sungshin.ac.kr)
[Received 1 July 2016, Reviewed 12 July 2016, Accepted 28 July 2016]
☆ This work was supported by the Sungshin Women's University Research Grant of 2014

소프트웨어 개발보안은 코딩단계에서 보안 문제를 점검함으로써 악의적인 공격으로부터 강인하게 작동하는 소프트웨어를 개발하는 것을 목적으로 한다. 이 활동은 웹 애플리케이션 개발시 자주 발생하는 보안오류를 코딩단계에서 제거하도록 하는 다양한 시큐어코딩 실무로 구성된다. 2012년부터 시작된 이 제도는 현재 공공분야의 시스템 개발 프로젝트 중 감리대상이 되는 모든 프로젝트에 대해 적용하도록 시행되고 있다. 소프트웨어 개발보안은 공공행정 분야를 넘어 국방, 의료, 증권, 금융, 자동차, 제조업 등 다양한 영역으로 확대되고 있다.

시큐어코딩과 관련한 소프트웨어 개발보안의 특징 중 하나는 그 범위를 구현 단계와 유지보수 단계로 한정한다는 점이다. 구체적으로 개발보안 활동은 신규개발인 경우는 소스코드 전체를, 유지보수인 경우는 변경된 부분에 국한하여 적용된다고 규정한다. 이에 반해 BSIMM[3], OWASP의 Open SAMM[4], 마이크로소프트의 MS SDL[5]과 같은 국외의 보안 프레임워크는 광의적인 보안활동을 정의한다. 이들은 시큐어코딩을 공통 활동으로 포함하지만, 더불어 교육, 보안계획, 보안설계, 평가 등 소프트웨어 개발생명주기 전반에 걸친 보안활동을 강

조하고 있다.

소프트웨어 개발보안이 악의적인 공격에 대한 선제적 대응을 하는 좋은 코딩실무를 제시하고 있지만 그럼에도 불구하고 BSIMM이나 Open SAMM과 같은 보안프레임워크와 비교할 때 프로세스 관점에서 코딩과 유지보수 단계의 연계를 체계적으로 설명하지 못하고 있는 점은 아쉬움으로 남는다. 본 논문은 소프트웨어 개발보안이 갖는 한계를 개발자, 유지보수자 관점에서 고찰하며, 유지보수 활동과 연계될 경우 소스코드의 품질관리 활동을 개선하는 방법을 모색한다.

논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 소개하며, 3 장에서는 현 개발보완의 문제점을 프로세스 관점에서 살펴본다. 4 장에서는 본 논문의 핵심이 보안메트릭을 설명하며, 5 장에서는 메트릭의 활용을 논한다.

2. 관련연구

악의적인 공격으로부터 안전한 소프트웨어를 개발하고자 하는 기존의 노력은 다음 두 가지 관점에서 살펴볼 수 있다. 먼저, 거시적 관점에서 제안된 보안구축(build security-in) 시도는 소프트웨어 개발자, 설계자, 보안담당자가 개발단계에서 보안기능을 구현하는 데 사용할 수 있는 방법, 도구, 지침, 규정 및 제반 자원을 포함한다. BSIMM, Open SAMM, MS SDL[5] 등은 이러한 시도의 대표적인 예이다. BSIMM의 경우 회사별로 존재하는 다양한 보안 활동을 객관적으로 평가하기 위해 제안된 공통 프레임워크의 성격을 갖는다. BSIMM은 2008년도에 최초버전이 등장한 이래 2015년 BSIMM 6로 발전하였으며, 현재 78개의 기업을 대상으로 202개의 개별 활동을 추출한 것으로 보고되었다.

반면, 미시적 접근의 경우 구현 단계에 초점을 맞추어 소스코드의 정적분석, 리뷰 등을 통해 보안 결함을 발견하는 활동에 집중한다. 대표적인 예로 미국 JPL 코딩 표준, CERT C 코딩 표준[6], MISRA-C 표준[7], 행정자치부의 공개 SW를 활용한 소프트웨어 개발보안점검가이드[8] 등을 들 수 있다. 이들 표준 혹은 가이드 기반의 접근의 특징은 준거성을 중요하게 여기며, 이에 관한 도구의 자동화를 제공한다는 점이다. 준거성이란 표준에서 요구하는 특성을 만족함으로 보여주어야 하는 요구사항으로 보안 결함 목록의 제거 여부를 도구 혹은 문서를 통해 입증해야 하는 규정이다. 예로서 MISRA-C의 준거성은 자동차 기능안전성 국제표준인 ISO/IEC 26262에 바탕하며, 행정자치부의 소프트웨어 개발보안 가이드 역시 행정기관 및 공공기관 정보시스템 구축·운영 지침[8]에 근거하고 있다.

3. 시큐어코딩 프로세스

2012년 6월에 발표된 행정자치부의 소프트웨어 개발보안가이드는 공공 시스템 개발자로 하여금 소스코드에 존재하는 보안약점(weakness)을 제거하는 것을 의무화하고 있다. 보안약점이란 운영환경에서 발견되는 취약점(vulnerability)의 원인을 말하며 주로 소스코드에 존재하는 보안오류이다. 2013년 개정된 소프트웨어 개발보안 가이드에서는 보안약점 진단목록을 8개 유형으로 구분하여 총 47개의 항목을 정의한다 (표 1).

(표 1) 개발보안의 보안약점 유형
(Table 1) Weakness category in Security-enhanced Development

유형(47개)	세부내용
입력데이터 검증 및 표현 (15개)	SQL 삽입, 경로조작 및 자우너삽입, 크로스사이트 스크립트, 운영체제 명령어 삽입, 위험한 형식의 파일 업로드, XQuery 삽입, XPath삽입, LADP삽입, 크로스사이트 스크립트 요청위조, 정수형 오버플로우 등
보안기능 (16개)	부적절한 인가, 중요정보 평문저장, 하드코드된 패스워드 등
시간 및 상태 (2개)	경쟁조건(TOCTOU), 제어문을 사용하지 않는 재귀함수 등
에러처리 (3개)	오류상황 대응 부재, 오류메시지를 통한 정보노출 등
코드오류 (4개)	NULL 포인터 역참조, 부적절한 자원 해제 해제된 자원사용 등
캡슐화 (5개)	제거되지 않고 남은 디버그 코드, 시스템 데이터 정보노출 등
API 오용 (2개)	DNS lookup에 의존한 보안결정, 취약한 API사용

소프트웨어 개발보안가이드는 시큐어코딩에 관한 국내 최초의 제도화된 접근이며 시큐어코딩의 중요성을 확산에 주된 역할을 했다. 그럼에도 불구하고 개발보안가이드는 다음의 시각에서 보완이 필요하다.

먼저, 소프트웨어 개발활동과 관련한 시큐어코딩 범위에 관한 문제이다. 시큐어코딩의 효과가 지속되기 위

해서는 개발 뿐 아니라 유지보수 과정에서도 지속적으로 적용되어야 한다. 그러나 많은 경우 발주자나 개발자는 기능보강 과정에서 생산된 소스코드에 대한 고려가 충분치 않은 것으로 나타났다. 예로서, 2014년도에 공공기관에서 발주된 19개의 제안요청서를 분석한 결과, 보안취약점 점검 요구를 하거나 개발보안제도의 명시적인 준수를 요구한 경우가 70% 수준이었지만 이중 추가적으로 유지보수 과정에서까지 개발보안을 요구한 경우는 18%에 지나지 않았다[10]. 이의 원인으로 비용 문제가 제기되기도 했지만 근본적인 문제는 소프트웨어 개발보안은 구현에 국한된 활동이라는 인식 때문이기도 하다.

두 번째, 프로세스 관점에서의 품질평가에 관한 문제이다. 공공기관의 정보시스템 구축에 적용되는 개발프로세스는 정보시스템 구축·운영지침[8]에 근거하여 정의된다. 이 지침은 정보시스템을 개발할 때 적용되는 주요 활동들, 예를 들면 기획·구축·운영·유지보수 활동을 보안활동 발주자 시각에서 정의한다. 소프트웨어 개발보안 가이드 역시 정보시스템 구축·운영지침에 근거한 아래와 같은 절차를 제시한다. (그림1)

(그림 1) 개발보안 단계별 활동
(Figure 1) Activities for security-enhanced development phases

계획 단계에서 평가자는 소스코드 안전성 평가를 수행하기 위한 계획과 절차를 수립을, 개발자는 소스코드 검사 신청을 의뢰하며, 사업 담당자는 코드 검사를 수행하기 위한 계획을 설정한다. 개발/구축 단계에서 개발자는 소스코드 위험성이 적절한 수준에서 계획되고 관리되었다는 증거를 제공한다. 평가 단계에서 평가자는 선정된 코드를 대상으로 정적분석 도구를 실행한다. 만일 인증 기준을 만족시키지 못한 코드에 대해서는 탈락 근거와 함께 보안강화 검사 결과를 신청자에게 통보하여 개선을 요구한다.

이 과정에서 중요한 점은 소스코드의 위험성이 적절한 수준에서 관리되고 있다는 증거를 제공하는 프로세스가 정의되는가 하는 것이다. 원칙적으로 이 작업은 정적분석을 통해 나타난 오류 리포트를 바탕으로 품질요소의 수집과 품질메트릭 분석이 선행되어야 하며, 실제 오류의 제거를 통해 메트릭 요소가 어떤 방식으로 개선되었는의 내용이 증거로 제공되어야 한다. 그럼에도 불구하고 현실적으로 이 과정은 체크리스트에 의해 오류가 수정되었음을 보여주는 수준에서 수행되고 있다.

4. 소스코드 보안메트릭

4.1 보안메트릭의 필요성

전통적으로 소스코드와 관련된 메트릭으로 많이 인용되는 것은 사이클로메틱 복잡도(Cyclomatic Complexity, CC)와 정보흐름 복잡도(Information Flow Analysis, IFC)이다[11]. CC와 IFC 는 각각 소스코드의 내부 논리 복잡도와 외부 인터페이스 복잡도를 표현한다. 특히 CC의 경우 모듈의 논리 복잡도와 에러발생 가능성 사이의 인과관계를 설명해주는 메트릭으로 알려져 있으며, 이런 이유로 국제공통평가기준[12]과 MISRA 코딩 표준에서는 단위함수의 CC 복접도가 10을 넘지 말 것을 권고한다.

CC나 IFC가 유용한 메트릭임에도 불구하고 보안 문제에 관해서는 제한적으로 사용된다. 그 이유 중 하나로 보안문제는 내부논리와 모듈 인터페이스의 복잡성 때문에 직접적으로 발생하는 것은 아니라는 점을 들 수 있다. 유용한 보안 메트릭이 되기 위해서는 CC나 IFC와 같은 구조 메트릭보다는 코드의 보안 특성을 반영하는 메트릭이 사용되어야 한다고 본다. 본 연구에서는 다음 특성을 고려하여 메트릭을 정의한다.

1) 보안약점은 발생원인과 영향 범위에 따라 그 영향이 틀리다.
2) 보안약점은 코드의 내부 구조와 별도로 다루는 정보에 따라 영향을 받을 수 있다.
3) 정적분석에 의해 보고된 보안약점은 반복적인 프로그래밍 과정을 거쳐 제거된다.

먼저, 기준 1)의 경우 보안문제의 발생 원인을 반영할 필요가 있음을 말한다. 이와 관련해서 CVSS라 불리는 NVD(National Vulnerability Database)[13]의 공통 취약점 스코어 시스템(Common Vulnerability Scoring System)[14]

이 취약성 정도를 계량화시켜 제공하고 있다. 본 논문의 보안약점을 반영하는 메트릭은 CVSS와 OWASP의 Top10 [15] 취약점 목록을 반영하여 정의된다.

기준 2)의 경우 기존의 CC, IFC가 간과했던 요소로 자료의 보안성이 취약점에 영향을 주는 경우이다. 예를 들어 안전하지 못한 API로 인해 발생하는 취약점과 더불어 오염된 데이터에 의한 취약점을 구분할 필요가 있다. 이를 위해 코드취약성 메트릭과 더불어 정보메트릭을 별도로 정의하여 활용한다.

코드취약성 메트릭은 소스코드에 대한 취약성을 정보를 담고 있다. 대상으로는 버퍼 오버플로우, 스택 및 힙 오버플로우, 포멧 스트링 취약성 등 CWE와 소프트웨어 개발보안가이드와 CWE에 근간한 소스코드 보안약점을 포함한다. 정보메트릭은 서비스 기능별로 제공하는 서비스를 구분하여 각 서비스에 대한 업무 중요도와 해당 모듈에서 처리하는 정보의 중요도를 종합적으로 평가한다. 정보메트릭과 코드취약성 메트릭의 결과는 최종적인 보안성 메트릭 계산에 이용된다.

기준 3)은 최초 점검과 이행점검 사이에 나타난 코드의 변화 정보를 표현 표현한다. 시큐어코딩 활동은 일반적으로 최소 두 번의 점검 활동을 필요로 한다. 먼저 최초 작성된 코드에 대한 정적분석을 수행하고, 수행결과에 따른 오류를 수정한다. 이행점검이라 불리는 두 번째 점검에서는 실제 오류들이 올바르게 제거되었는지 확인하는 목적으로 정적분석을 재수행한다. 이 과정에서 코드 변경 정보는 프로그래머와 유지보수 담당자 사이에 공유되어야 하며, 코드 메트릭은 이 과정에서 유용한 정보를 제공하는 매개체로 활용될 수 있다.

4.2 코드취약성 메트릭과 정보메트릭의 구성

코드취약성 메트릭은 환경과 시간에 독립적인 소스코드의 취약성 요소를 표현한다. 이 메트릭은 응용 프로그램에 독립적이며, 동시에 언어 종속적인 명령어의 취약성을 표현한다. 메트릭의 측정 대상은 정적분석을 통해 확인할 수 있는 보안점검 대상항목의 근간은 CWE 보안약점[14]과 소프트웨어 개발보안 점검 항목들에 대한 발생 빈도와 위험구분이다. 발생빈도는 응용 프로그램에 따라 차이를 보이며, 자주 발생하는 항목 일수록 집중 관리가 필요하게 된다. 위험에 관한 항목은 보안약점이 잠재적으로 유발할 수 있는 유험 수준을 구분한 것으로 고위험군의 약점을 우선적으로 대체해야 함을 알 수 있게 한다.

(표 2) 보안약점 목록
(Table 2) Weakness list

	보안약점 명	빈도	위험
1	크로스사이트 스크립트	VH	VH
2	시스템 정보노출	H	H
3	오류메시지를 통한 정보노출	H	M
4	오류사항 대응부재	VH	H
5	SQL 삽입	H	VH
6	HTTP 응답분할	H	H
7	디렉토리 경로조작	H	H
8	잘못된 세션에의한 데이터 정보 노출	H	H
9	부적절한 자원 해제	M	M
10	널(Null)포인터역참조	M	H
11	신뢰되지 않은URL 주소 자동접속연결	M	H
12	위험한 형식 파일 업로드	M	M
13	제거되지 않고남은 코드	M	M
14	사용자 주요정보 평문저장 또는전송	M	H
15	취약한 암호화 알고리즘	M	H
16	사용자 하드저장쿠키를 통한 노출	M	H
17	취약한 패스워드 허용	M	H
18	적절하지 않은 난수 값 사용	M	M
19	패스워드 평문저장	L	H
20	자원삽입	L	L
21	하드코드된 패스워드	L	H
22	주석문 안에 포함된 시스템 주요정보	L	M
23	불충분한 세션관리	L	H
24	정수 오버플로우	L	H
25	운영체제 명령어 삽입	L	VH

실제 메트릭을 근간하여 점수 계산을 할 경우 척도별 배점은 Very High는 5점, High는 4점, Medium은 3점, Low는 2점, Very Low는 1점을 할당한다.

개별 취약성 V_i 는 표 2에 나타난 개별약점에 대한 취약성을 상대적으로 계산한 값으로 다음 조건으로 계산된다.

- 발생 빈도(Frequncy, *Fr*)와 위험성(Serverity, *Sv*) 척도: 1~5사이 값
- 빈도 가중치(*Wf*)와 위험성 가중치(*Ws*)와 :0~2 사이의 실수값

$$V_i = Sv(i)\times Ws + Fr(i)\times Wf$$

발생빈도와 위험성이 반영된 개별 취약성 값 V_i에 근거하여 두 항목을 반영한 취약성 구분은 표 3과 같이 구성된다.

(표 3) 개별 취약성 수준
(Table 3) Individual Vulnerability level

항목	구간 값				
Sv(i)	5	4	3	2	1
Fr(i)	5	4	3	2	1
Vi 값	9~10	8~7	6~5	4~3	2
구간	VH	H	M	L	VL

예를 들어, 위험성과 침투성이 모두 Very High인 경우 V_i 값은 10의 값을 가지게 된다. 만일, 이들 두 요소가 VeryHigh와 High로 구성되어 9의 값을 갖더라도 높은 등급의 보안 속성을 따라가는 보안권한 상승 원리로 이들의 V_i 값은 VeryHigh 구간으로 분류되도록 설정하였다.

이를 바탕으로 각 5개의 등급별 코드취약성 판정등급 VL (Vulnerability Level)은 다음과 같이 정의된다.

VL(level) =

$$\frac{\sum((Sv(level)\times W_s + Fr(level)\times W_f)\times TEC)}{\sum((Sv(All)\times W_s + Fr(All)\times W_f)\times TEC)}$$

단, Sv(All)과 Fr(All)은 모든 구간의 Sv와 Fr 값을 의미하며, TEC(Total Error Count)는 동일 취약점의 반복 횟수 측정치다. 만일 수준이 VeryHigh로 정의될 경우의 판정식은 다음과 같다.

VL(VeryHigh) =

$$\frac{\sum((Sv(VeryHigh)\times W_s + Fr(VeryHigh)\times W_f)\times TEC)}{\sum((Sv(All)\times W_s + Fr(All)\times W_f)\times TEC)}$$

Sv(VeryHigh)와 Fr(VeryHigh)는 표 2의 보안약점 목록에 근거한 Very High 수준을 갖는 위험성과 침투성요소를 의미한다.

정보 중요도란 취약 명령어로 인해 공격자 침투가 발생할 경우 영향을 받는 정보의 기밀성, 무결성, 가용성의 영향 정도를 나타내는 지표이다. 자료 중요도는 다음의 3가지 기준으로 평가된다.

(표 4) 정보 중요도 목록
(Table 4) list for information importance

항목	설명	척도	가중치
기밀성(C)	허가되지 않은 정보의 누출을 방지하는 수준	1~5	Wc=1
무결성(I)	허가되지 않은 정보의 수정을 방지하는 수준	1~5	Wi=1
가용성(A)	공격에 대해 시스템 운영이 지속되는 수준	1~5	WA=1

정보중요도(CIA)는 다음과 같이 평균값으로 계산된다.

$$CIA = Ave((C\times W_c)+(I\times W_i)+(A\times W_A))$$

이상에서 얻어진 2 가지의 메트릭인 코드취약성 메트릭(V), 정보 중요도 값(CIA)과 보정 계수 Adj를 사용하여 최종적인 소스코드 보안값(Source Security Value, SSV)을 계산한다. 단, 보정계수 값 Adj는 기관별 혹은 대상 시스템별 편차를 보정해주기 위하여 0.5~1까지의 범위에서 기관 특성에 따라 할당할 수 있다.

$$SSV = (CIA \times V) \times Adj$$

SSV값은 전체 시스템의 보안 속성을 표현하는 값으로 기관별로 별도의 평가 범위를 두어 활용할 수 있다. 예를 들면, 표 5는 정보중요도가 VH, H 수준을 넘어서지 못하도록 규정한 분류표이다. 이 경우 SSV의 경계 값은 $MAX_{VH}(CIA)+MAX_{VH}(V)$ ~ $MAX_H(CIA)+MAX_H(V)$범위이므로 25~16으로 그 이상을 위험 영역으로 제시한다 (검은 영역 참조).

(표 5) 허용가능한 소스코드 보안값
(Table 5)Acceptable Source code security values

		소스코드 위험수준				
		VH(5)	H(4)	M(3)	L(2)	VL(1)
정보 중요도	VH(5)	25	20	15	10	5
	H(4)	20	16	12	8	4
	M(3)	15	12	9	6	3
	L(2)	10	8	6	4	2
	VL(1)	5	4	3	2	1

5. 소스코드 보안메트릭의 적용

5.1 케이스 스터디

이 장은 앞서 설명한 정보메트릭과 코드취약성 메트릭을 활용하는 예로서 본 연구에 사용한 자료는 2014년 시큐어코딩 진단을 통해 얻어진 A사의 자료를 토대로 한다. 대상 프로젝트는 Java로 구현된 웹 정보시스템으로서 최초 진단은 정적분석과 동적분석을 통합 수행하여 총 4,073개의 보안약점을 발견하였다.

보안약점 제거 작업을 통해 얻어진 수정 코드에 대해 2차 평가를 하여 2,761개의 보안 약점을 발견한 경우이다. 점검활동을 통해 찾아낸 보안에러의 종류와 TEC 측정치, 그리고 각 취약성에 대한 발생빈도와 위험성은 표

6과 같다. 각 위험 수준별 취약 등급을 판정하기 위해서는 표 6에 나타난 위험성과 침투성을 근거로 코드취약성 값 V를 산출한다.

이들은 표 2의 취약성 수준 평가 기준을 참고로 Very High에서 Very Low의 수준별로 분류하였다[표 8]. 예를 들면, 1차 점검 결과에 대한 VH 수준의 고위험도 V 값을 구하기 위해서는 Fr과 Sv값 하나라도 VH를 포함하고 있는 보안약점 항복들이 대상이다. 따라서 이 경우는 다음 항목이 대상이다.

$TEC_{(크로스사이트\ 스크립트)}+TEC_{(SQL삽입)}+TEC_{(정수오버플로우)}$ =
2364+456+2 = 2821

(표 6) 보안약점 발생빈도
(Table 6) Occurrence frequency for weakness

진단 결과	Fr	Sv	1차	2차
크로스사이트 스크립트	5	5	2364	1705
오류메시지를 통한 정보노출	4	4	1183	545
SQL 삽입	4	5	456	167
HTTP 응답분할	4	4	36	10
디렉토리 경로조작	4	4	10	10
사용자 하드디스크 저장쿠키를 통한 노출	3	4	6	0
오류상황 대응부재	5	4	0	34
취약한 패스워드 허용	3	4	5	0
취약한 암호화 알고리즘	3	4	3	0
위험한 형식 파일 업로드	3	3	2	0
신뢰되지 않은URL 주소로 자동접속연결	3	4	2	0
자원삽입	2	2	2	0
불충분한 세션관리	2	3	2	0
운영체제 명령어 삽입	2	4	1	0
정수 오버플로우	2	5	1	0
널포인터 역참조	3	4	0	16
하드코드된 패스워드	2	4	0	9
패스워드 평문저장	2	4	0	1

이들에 대한 구간값 VL(VeryHigh)은 다음과 같이 계산된다.

$$\frac{\sum((Sv(VH)\times 1+Fr(VH)\times 0.8)\times TEC(VH)}{\sum((Sv(All)\times 1+Fr(All)\times 0.8)\times TEC)}=\frac{25021}{34011}$$

VL(VeryHigh) = 73.6% 구간에서 Very High의 위험성이 존재함을 알려준다.

표 7은 이와 같은 방식으로 1차와 2차 점검 기간에 나타난 모든 Very Hogh구간에서 Low구간까지의 4개 영역에 대한 모든 V값의 결과를 보여준다.

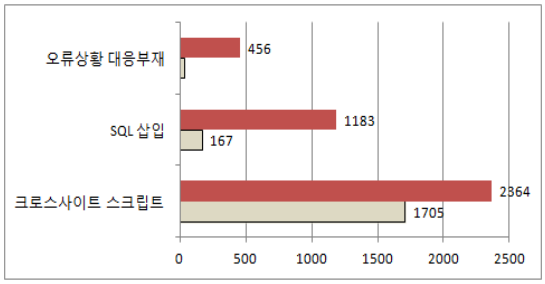
(표 7) 위험등급 판정
(Table 7) Decision for risk level

	위험수준	구간별 TEC합	구간별 V합	V비율	판정
1차	VH	2821	25021	73.6%	VH
	H	1245	8949	26.3%	
	M	4	22	0.1%	
	L	2	2	0%	
2차	VH	1906	16986	49.9%	H
	H	166	5398	15.8%	
	M	144	277	0.8%	
	L	0	0	0	

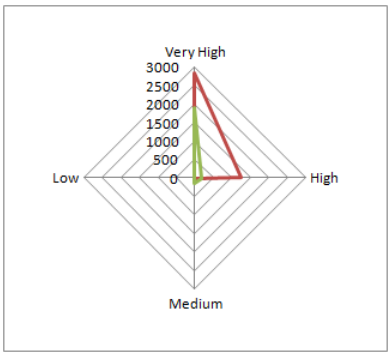
정보메트릭이 중요한 경우는 시스템에서 의도하지 않은 형태로 정보가 노출되거나 변조되는 경우를 말한다. 이에 해당되는 보안약점명과 위험수준은 다음과 같다.

- 오류메시지에 의한 정보노출 (H)
- 사용자 하드디스크 저장쿠키를 통한 노출 (H)
- 취약한 패스워드 허용 (H)

본 논문에서 다루는 웹정보시스템의 경우 오류메시지를 통한 정보노출이 1,183건으로 보고되었다. 이 경우 자료의 중요도(H)와 코드취약성 중요도(H)로 볼때 [표6]에서는 High수준의 관리 대상으로 분류함을 알 수 있다. 따라서 이들은 SQL삽입, 크로스사이트 스크립트와 더불어 중요하게 관리되어야하는 보안약점으로 분류하며, 이 경우 2차 점검을 통해 그 추이를 관찰할 필요가 있다. 실제 수정작업을 통해 개선된 이들 취약점의 추이는 아래의 그림 2와 그림 3을 통해서도 확인할 수 있다. (음영막대는 1차 점검, 밝은 막대는 2차 점검 결과임)



(그림 2) 보안약점 개선
(Figure 2) Improvement for weakness



(그림 3) 보안약점 개선 분포
(Figure 3) Improvement distribution for weakness

5.2 소스코드 보안메트릭 평가

본 절에서는 소스코드 보안메트릭의 특징을 살펴보기 위해 4절에서 언급되었던 공통취약성 스코어링시스템(CVSS), 사이클로메틱 복잡도(CC)와 정보흐름 메트릭(IFC)등과 비교한다. 비교에 앞서 메트릭의 특성을 비교하는 과정에서 적용된 기준을 소개하면 다음과 같다.

- 목표: 메트릭이 평가하고자 하는 대상 및 목표
- 보안 지향성: 메트릭의 보안성 관련 정도
- 이해성: 평가자 측면의 메트릭 적용 및 이해용이성
- 확장성: 새로운 요소를 추가할 경우 확장 용이성
- 등급: 등급의 구체성
- 표준화: 일반적으로 통용되거나 표준으로 채택됨
- 선택성: 정보 시스템의 성격에 따른 요소별 선택 적용 정도

표 8은 이들 기준을 바탕으로 본 논문의 접근방법의 특성을 비교한 것이다.

(표 8) 메트릭의 특성비교
(Table 8) Comparisond for metrics

메트릭	목표	보안지향	이해도	확장성	등급	표준화	선택성
공통취약성 스코어시스템	실행모듈 취약성	●	○	⊙	○	●	●
사이클로메틱 복잡도/ 정보흐름복잡도	소스코드 복잡도	⊙	●	○	○	×	○
제안 메트릭	소스코드 취약성	●	⊙	●	●	×	●

●: 강함, ⊙: 중간, ○: 약함, ×: 없음

5. 결 론

안전한 소프트웨어를 구축하는 방법으로서의 시큐어코딩은 소프트웨어 개발보안 제도의 확산에 힘입어 공공기관의 시스템 구축시 중요 요구사항으로 정착되고 있다. 시큐어코딩은 도구와 기법 측면에서 많은 발전을 이루고 있지만 관리 측면에서는 활발히 연구가 되지 못한 것도 사실이다.

본 연구를 통해 제안된 소스코드 보안메트릭은 두 부분으로 구성된다. 첫 째, 정적분석을 통해 획득한 보안약점에 관한 평가용도의 메트릭이다. 코드취약점 메트릭과 정보메트릭을 통해 코드 자체에만 집중될 수 있는 관심을 정보와 결합하여 평가할 수 있도록 제안하였다. 두 번째는 메트릭을 통한 유지보수 활동과의 결합이다. 기존의 소프트웨어 개발보안이 코딩 단계에 집중되었다면 소프트웨어의 유지보수에서 발생하는 코드 변경에 관해서도 접근이 필요하다. 본 논문은 시큐어코딩 활동이 반복되어 수행된다는 점에 착안하여 1차 점검 결과와 이행점검 사이의 코드 개선을 구분하여 새로운 버전에 대해 품질 관점에서 보안성이 개선되는 과정을 위험등급 판정과 보안약점 개선추이 그래프를 통해 측정, 관리할 수 있도록 고려하였다. 이를 통해 유지보수 담당자는 측정된 보안메트릭에 근거하여 보안활동 계획을 수립할 경우 자원과 일정배정에 활용할 수 있다.

향후 본 논문을 통해 제안된 보안메트릭이 전자정부 정보보호관리체계(G-ISMS) 혹은 민간 정보보호관리체계(K-ISMS)와 같은 다양한 정보보호관리체계와 연동하여 개발 프로세스에 적용할 수 있는 방법으로 확장된다면 다양한 시너지 효과를 기대할 수 있을 것으로 판단한다.

참 고 문 헌 (Reference)

[1] IBM X-Force Threat Intelligence Quaterly, 1Q 2015
[2] Software security-enhanced development guides version 2, Ministry of the Interior 2013.11, http://www.moi.go.kr/frt/bbs/type001/commonSelectBoardArticle.do?bbsId=BBSMSTR_000000000012&nttId=42149
[3] BSIMM 6, http://www.BSIMM.com, 2015
[4] Open SAMM, http://www.opensamm.org
[5] Microsoft SDL, https://www.microsoft.com/en-us/sdl/
[6] R. Seacord, The CERT C Secure Coding Standard, Addison Wesley, 2008.

[7] MISRA-C:2012 http://www.misra.org.uk
[8] Software security-enhanced development guides using open source software, Ministry of the Interior, 2016. 2 http://www.moi.go.kr/frt/bbs/type001/commonSelectBoardArticle.do?bbsId=BBSMSTR_000000000012&nttId=48386
[9] Development and operation guide for public and administrative information systems, Ministry of the Interior 2013.11 2013. 8
[10] D. Seo, S, Kim et al, Study on application of Software security-enhanced development, KISA, 2014
[11] L. Laird, M Brennan, Software Measurement and Estimation: A Practical Approach, Wiley Inter-Science, 2006
[12] Common Criteria, http://commoncriteriaportal.org, 2006.
[13] National Vulnerability Database Version 2.2, http://nvd.nist.gov/
[14] Common Vulnerability Scoring System version 3, http://nvd.nist.gov/cvss.cfm, 2016
[15] OWASP Top 10 https://www.owasp.org/index.php
[16] Common Weakness Enumeration https://cwe.mitre.org/
[17] Guidelines for certification of G-ISMS, Ministry of the Interior, 2010. 6. http://www.law.go.kr/LSW/admRulInfoP.do?admRulSeq=2000000014026

◐ 저 자 소 개 ◑

서 동 수 (Dongsu Seo)
1986년 중앙대학교 컴퓨터공학과 졸업(학사)
1990년 영국 맨체스터대학교 대학원 Computation학과 졸업 (석사)
1994년 영국 맨체스터대학교 대학원 Computation학과 졸업 (박사)
1998～현재 성신여자대학교 IT학부 교수
관심분야 : 정보보호, 소프트웨어공학, 정형기법
E-mail : dseo@sungshin.ac.kr