情報保護學會論文誌
第20卷 第6號, 2010. 12

소스코드의 보안성 메트릭 설계에 관한 연구*

서 동 수†
성신여자대학교

A Study on the Design of Security Metrics for Source Code

Dongsu Seo†
Sungshin Women's University

요 약

소스코드의 정적분석 기술은 소스코드 자체에 내재된 취약성을 찾는데 활용되는 중요한 기술로 인식되고 있다. 본 논문은 정보시스템의 소스코드 보안성 수준을 평가하는 방법으로 정적분석 결과인 소스코드의 취약성 정보와 프로그램이 처리하는 정보의 중요도를 활용하는 평가 메트릭의 설계 및 활용을 소개한다. 소스코드 보안성 메트릭은 소스코드의 취약점 수준을 개발 과정에서 미리 파악할 수 있도록 도움을 준다는 측면에서 평가자와 개발자 모두가 활용할 수 있다. 특히 평가자는 보안 메트릭을 통해 소스코드의 성격과 요구되는 정보의 보안수준에 따라 소스코드의 보안 수준을 점검하며, 코드 검수에 활용할 수 있다.

ABSTRACT

It has been widely addressed that static analysis techniques can play important role in identifying potential security vulnerability reside in source code. This paper proposes the design and application of security metrics that use both vulnerability information extracted from the static analysis, and significant factors of information that software handles. The security metrics are useful for both developers and evaluators in that the metrics help them identity source code vulnerability in early stage of development. By effectively utilizing the security metrics, evaluators can check the level of source code security, and confirm the final code depending on the characteristics of the source code and the security level of information required.

Keywords: security metrics, static analysis, security vulnerability, source code vulnerability,

I. 서 론

정보시스템에 대한 보안성 강화는 안정되고 신뢰할 수 있는 서비스를 제공한다는 측면에서 필수 요구사항으로 인식된다. ISMS(Information Security Management System)와 같은 제도를 통해 운영과정의 정보보호 활동을 강화시킬 수도 있지만, 개발 생명주기 상의 주요 활동들, 예를 들면, 요구, 설계, 구축 등의 과정에서 보안성을 강화시키는 것이 더욱 효과적이라는 연구결과들이 보고되고 있다[1,10].

개발 과정의 보안성을 강화시키는 수단으로 최근 활발히 논의되고 있는 정적분석(static analysis)과 시큐어 코딩(secure coding)은 코딩 단계의 점검 활동을 통해 소스코드의 보안성을 향상시키고자 하는 대표적인 시도이다[2]. 정적분석은 버퍼나 스택의 오버플로우, 메모리 누수 등 보안성과 직결된 치명적인 오류를 테스팅 이전에 탐지하며, 침투 테스팅과 같이 사용할 경우 큰 효과를 가져다주는 것으로 알려졌다[3].

접수일(2010년 9월 24일), 수정일(2010년 12월 4일), 게재확정일(2010년 12월 11일)
* 본 연구는 한국인터넷진흥원(과제명: 국내 정보시스템 안전성 보증을 위한 소프트웨어 보안성강화체계)의 지원으로 수행하였습니다.
† 주저자, dseo@sungshin.ac.kr

소스코드에 대한 보안성 향상의 핵심에는 세 가지 사항이 전제된다. 첫째, 시큐어 코딩을 통한 안전한 프로그램의 작성이다. 이를 위해 개발자는 집중적인 교육을 통해 안전한 API의 사용 방법과 취약한 함수의 문제에 대해 충분한 이해하고 있어야 한다. 둘째, 실제 코딩한 내용이 시큐어 코딩에서 제안한 기준을 준수하였는지를 정적분석 도구를 통해 점검한다. 마지막으로, 시큐어 코딩과 정적분석 결과를 코드 수정에 반영한다. 이러한 과정을 통해 관리자는 실제 목표 수준과 현재 진행되는 개발 수준을 모니터링하며, 결과를 개발 과정에 피드백 시킨다.

이러한 활동이 의미있기 위해서는 소스코드 취약성에 관한 측정 메트릭을 정의하며, 활용하는 기준이 마련되어야 한다. 본 논문은 소스코드의 보안성을 향상시키기 위한 노력으로, 소스코드 보안성 메트릭을 정의하고, 이를 활용하는 절차를 제안한다. 논문의 구성은 다음과 같다. 2 장에서는 관련 연구 소개를, 3 장은 소스코드 메트릭 평가체계를, 4 장에서는 두 단계의 메트릭 개념과 평가 사례를 소개한다.

II. 관련 연구

보안 메트릭에 관한 연구는 크게 구조에 관한 메트릭과 실행에 관한 메트릭으로 구분한다. 먼저, 구조에 관한 대표적인 메트릭으로 소프트웨어 복잡도와 결합도 메트릭을 들 수 있다. 맥케이브(McCabe)의 사이클로메틱 복잡도(Cyclomatic Complexity, CC)[4]와 핸리와 카프라의 정보흐름 복잡도(Information Flow Analysis, IFC)[4] 메트릭은 각각 소스코드의 내부 논리와 외부 인터페이스의 복잡도를 나타낸다. 이들 메트릭은 모듈 내부와 외부 요소의 관계를 표현함으로써 모듈 이해와 유지보수에 관한 유용한 정보를 제공해준다. 정보 보호와 관련하여 이들 메트릭은 공통평가기준(Common Criteria, ISO/IEC 15408)의 평가보증등급 5 (Evaluation Assurance Level 5)이상의 모듈 복잡도 평가기준으로 활용이 된다[5].

실행에 관한 보안 메트릭의 대표적인 사례로는 국가 취약성 데이타베이스 (National Vulnerability Database, NVD)[6]의 공통 취약점 스코어 시스템(Common Vulnerability Score System, CVSS)[7]이 있다. CVSS는 NVD의 보고된 취약점들을 대상으로 이들의 취약성 정도를 계량화시켜 웹을 통해 제공하고 있다. CVSS는 기본, 임시, 환경 메트릭 등 세 가지 수준으로 구분하여 소프트웨어 취약성에 대한 정교한 계산을 수행한다. CVSS는 계산과정에서 사용되는 파라메터 값의 임의성, 계수 선정의 복잡성 등의 문제가 있다. 이러한 이유로 CVE(Common Vulnerability and Exposures)[8]에서는 기본 메트릭 값만 웹을 통해 공개하고 있다.

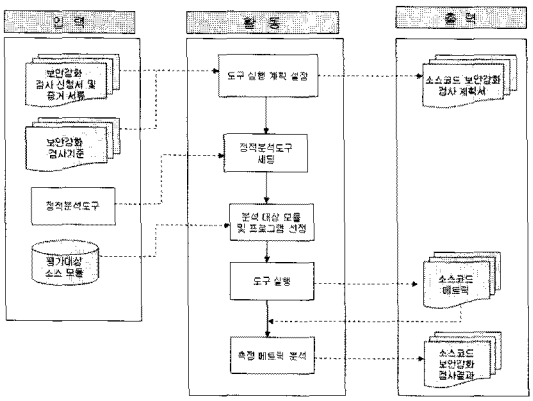
소스코드의 논리와 구조에 대해서는 CC와 IFC 메트릭을 사용할 수 있지만, 소스코드의 보안성 및 취약성에 대해서는 적절한 수단이 되지 않는다. 오히려 이들은 소스코드의 이해와 유지보수 특성과 더욱 관련이 있다고 하는 것이 타당하다. 이를 통해 볼 때 소스코드의 보안성만을 전문으로 판단하는 메트릭의 필요성은 크다고 할 수 있다.

III. 소스코드 메트릭을 활용하는 평가 체계

본 연구는 전자정부 시스템의 소스코드 안전성을 확인하기 위한 목적의 소스코드 메트릭을 개발하는 과제의 지원으로 시작하였다. 전자정부 시스템이 공공서비스를 제공하며, 개인정보에 대한 정보보호를 중요시한다는 점에서 보안에 대한 기획과 감리, 검수 및 확인은 발주자 입장에서 중요한 사안이다. 이러한 시각에서 소스코드 메트릭을 개발자와 발주자, 혹은 평가자가 어떤 절차를 거쳐 사용할 수 있는지를 설명하고자 한다.

소스코드 메트릭은 코딩 단계에만 적용되는 보안 메트릭이라 생각하기 쉽다. 그러나 소스코드 메트릭의 종류와 수준을 정의하는 것은 프로젝트 기획에서부터 검수에 이르는 전체 과정에 영향을 미친다. 계획 단계에서 평가자는 개발 생명주기 내의 소스코드 안전성 평가를 수행하기 위한 계획과 절차를 수립하고, 독립적인 평가를 준비한다. 이 단계에서 개발자는 소스코드 검사 신청을 의뢰하며, 사업 담당자는 코드 검사를 수행하기 위한 계획을 수립하며, 코드 검사 활동의 대상 코드를 선정하고 검사 기준의 항목을 선정한다.

개발 단계에서 개발자는 개발 과정에서 소스코드 위험성이 적절한 수준에서 계획되고 관리 되었다는 증거를 제공한다. 이 때 개발자가 수행하는 코드 검사의 목표는 사업수행계획서 상에 명시된 소스코드 보안성 강화기준의 목표와도 일치해야 한다. 마지막으로 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. [그림 1]은 평가의 시작으로부터 종료에 이르는 활동 과정을



〔그림 1〕 소스코드 메트릭과 평가 체계 관계

역할별로 구분한 것이다.

IV. 소스코드 보안성 메트릭의 구성

소스코드 보안성 메트릭이란 소스코드 내에 잔존하는 보안 약점과 소스코드가 다루는 정보의 중요도를 종합적으로 반영한 평가 메트릭이다. 이 메트릭은 언어취약성 메트릭과 정보중요도 메트릭으로 구성된다. 먼저, 언어 취약성 메트릭은 소스코드에 대한 취약성을 측정하는 것으로 취약점의 종류에는 버퍼 오버플로우, 스택 및 힙 오버플로우, 포맷 스트링 취약성 등 CWE(Common Weakness Enumeration)와 CERT C에 열거된 소스코드 관련 취약성을 사용한다. 정보 중요도 메트릭은 서비스 기능별로 제공하는 서비스를 구분하여 각 서비스에 대한 업무 중요도와 해당 모듈에서 처리하는 정보의 중요도를 종합적으로 평가한다. 정보 중요도 메트릭과 언어 취약성 메트릭의 결과는 최종적인 소스코드 보안성 메트릭 계산에 이용된다.

4.1 언어 취약성 메트릭

언어 취약성 메트릭은 환경과 시간에 독립적인 소스코드의 취약성 요소를 표현한다. 이 메트릭은 응용프로그램에 독립적이며, 동시에 프로그래밍 언어 종속적인 명령어의 취약성을 표현한다. 메트릭의 측정 대상은 정적분석을 통해 확인할 수 있는 취약성 항목들로서 CVE 취약성[8]과 CERT C 취약성[9] 표준 목록에서 열거된 취약한 명령어들이다. 언어 취약성 메트릭은 모듈단위, 패키지 단위 혹은 필요에 따라 함수 단위로 계산될 수 있다. 대표적인 소스코드 취약성에 대한 위험성과 침투성 수준은 〔표 1〕에 정의된다.

척도 별 배점은 Very High는 5점, High는 4점, Medium은 3점, Low는 2점, Very Low는 1점을 할당한다.

〔표 1〕 소스코드 취약성 목록

번호	소스코드 취약성	위험성	침투성
1	버퍼 오버플로우	Very High	Very High
2	임의값 할당 (Write-what-where)	Very High	High
3	스택 오버플로우	Very High	Very High
4	힙 오버플로우	Very High	Very High
5	버퍼 언더런	High	Medium
6	Wrap-around error	High	Medium
7	정수 오버플로우	High	Medium
8	정수 형변환	High	Medium
9	절삭 에러	Low	Low
10	Sign 확장에러	High	High
11	Signed로부터 unsigned 형변환 에러	High	Medium
12	Unsigned로부터signed 형변환 에러	High	Medium
13	미확인 배열 인덱싱	Medium	Medium
14	잘못된 null 종료	High	Medium
15	부적절한 문자 길이 확인	High	High
16	Covert storage channel	Medium	High
17	잘못된 switch 문의 디폴트 사용	Medium	Medium
18	Null 포인터 역참조	Medium	Medium
19	free된 메모리의 사용	Very High	High
20	메모리의 이중 free	High	Medium
21	신뢰할 수 없는 모바일 코드 이용	Medium	Medium
22	크로스 사이트 스크립팅	Medium	Medium
23	포멧 스트링 에러	High	Very High
24	인젝션 문제	High	Very High

〔표 2〕 취약성 수준표 (단, W_s,W_e 값이 1인 경우)

항목	구간 값				
Sv(i)	5	4	3	2	1
Exp(i)	5	4	3	2	1
V_i 값	9~10	8~7	6~5	4~3	2
구간	Very High	High	Medium	Low	Very Low

개별 소스코드 취약성 V_i 값은 〔표 1〕의 소스코드 취약성 분류에 근거하여 다음 조건에 의해 계산된다.

- 위험성(Sv)과 침투성(Exp) 척도는 1~5의 값 사이에서 결정된다.
- 위험성 가중치(W_s)와 침투성 가중치(W_e)는 필요에 따라 1~2 사이의 값이 부여된다. 별도의 요구가 없으면 이 값은 1로 할당된다.
- V_i는 다음 식에 의하여 구해진다.

$V_i = Sv(i) \times W_s + Exp(i) \times W_e$

개별 소스코드 취약성 값 V_i에 근거한 취약성 구분은 〔표 2〕와 같다.

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

이를 바탕으로 각 수준별 언어취약성 판정등급 VL (Vulnerability Level)은 다음과 같이 정의된다.

$$VL(level) = \frac{\sum((Sv(level) \times Ws + Exp(level) \times We) \times TEC)}{\sum((Sv(All) \times Ws + Exp(All) \times We) \times TEC)}$$

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

$$VL(VeryHigh) = \frac{\sum((Sv(VeryHigh) \times Ws + Exp(VeryHigh) \times We) \times TEC)}{\sum((Sv(All) \times Ws + Exp(All) \times We) \times TEC)}$$

단, Sv(VeryHigh)와 Exp(VeryHigh)는 〔표 1〕의 취약성 목록에 근거한 Very High 수준을 갖는 위험성과 침투성요소를 의미한다. 〔표 3〕은 VL은 수준에 따라 갖는 판정 구간의 범위를 나타낸다.

〔표 3〕 소스코드 위험성 판정 등급

판정수준 (VL)	구간 값
Very High(5)	5% ≤ VL(VeryHigh)
High(4)	1% ≤ VL(VeryHigh) 〈 5% 혹은 5% ≤ VL(High) 〈 15%
Medium(3)	VL(VeryHigh) = 0% 인 조건에서 1% ≤ VL(High) 〈 5% 혹은 10% ≤ VL(Medium) 〈 50%
Low(2)	VL(High) = 0% 인 조건에서 5% ≤ VL(Medium) 〈 10%
Very Low(1)	VL(High) = 0% 인 조건에서 VL(Medium) 〈 5%

4.2 정보 중요도 메트릭

자산으로서의 정보는 그 성격에 따라 보호하는 수준을 분리하여 관리하는 것이 중요하다. 보안 시각에서 정보란 소스코드와 더불어 보호되어야 할 대상이다. 본 논문에서 정보 중요도 메트릭은 행정안전부의 통합인증프레임워크 가이드[11]의 내용을 반영하여 설계하였다.

통합인증프레임워크 가이드는 정보에 대한 중요도를 서비스 중요도와 자료 중요도로 구분하여 설명한다. 서비스 중요도란 취약한 명령어로 인해 공격자 침투가 발생할 경우 영향을 받는 서비스의 영향 정도를 나타낸다. 서비스 중요도는 다음의 3가지 기준으로 평가된다.

자료 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〕 서비스 중요도(S) 구분

항목	설명	척도	가중치
서비스의 사회적, 기관 내부적 업무 중요도	국가 안전보장, 사회질서 유지, 국가 경제이익 보호, 국민 생명보호 등	5점 (E1)	2(W1)
서비스 대상 이용자 규모	서비스 이용자의 규모를 분류	5점 (E2)	1(W2)
대외 업무 연계정도	대상서비스와 연계된 서비스의 규모	5점 (E3)	2(W3)

〔표 5〕 자료 중요도(CIA) 구분

항목	설명	척도	가중치
기밀성	허가되지 않은 정보의 누출을 방지하는 수준	5점 (E4)	1(W4)
무결성	허가되지 않은 정보의 수정을 방지하는 수준	5점 (E5)	1(W5)
가용성	공격에 대해 시스템 운영이 지속되는 수준	5점 (E6)	1(W6)

이들 서비스 중요도(S)와 자료중요도(CIA)는 다음과 같이 계산된다.

$$S = (E1\times W1)+(E2\times W2)+(E3\times W3)$$
$$CIA = (E4\times W4)+(E5\times W5)+(E6\times W6)$$

이상에서 얻어진 3 가지의 메트릭인 언어 취약성 메트릭(V), 서비스 중요도 값(S)과 보정 계수 C1, 정보 중요도 값(CIA)과 보정 계수 C2를 사용하여 최종적인 소스코드 보안값(Source Security Value)을 계산한다. 단, 보정계수 값 C1, C2는 기관별 혹은 대상 시스템별 편차를 보정해주기 위하여 0.5~1까지의 범위에서 기관 특성에 따라 할당할 수 있다.

$$SSV = (CIA\times C1+S\times C2)\times VL$$

SSV값은 전체 시스템의 보안 속성을 표현하는 값으로 기관별로 별도의 평가 범위를 두어 활용할 수 있다. 예를 들면, 〔표 6〕은 자료 중요도와 서비스 중요도가 각각 VH, H 수준을 넘어서지 못하도록 규정한 분류표이다. 이 경우 SSV의 경계 값은 16+16=32로 그 이상을 위험 영역으로 제시한다(검은 영역 참조). 따라서 이 경우 허용 가능한 최대 소스코드 위험 수준은 Medium 이하임을 알 수 있다.

〔표 6〕 허용가능한 소스코드 보안값의 구분 예

		서비스별 위험수준				
		VH(5)	H(4)	M(3)	L(2)	VL(1)
자료 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
서비스 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(2)	5	4	3	2	1

4.3 소스코드 보안성 메트릭의 적용

소스코드 취약성 메트릭의 적용 과정을 살펴보기 위해 C언어로 구현된 P2P 파일 다운로드/업로드 프로그램을 대상으로 보안수준을 분석한다. 이 프로그램의 총 라인 수는 198,182며, 정적분석기를 통해 발견된 취약점은 총 18종으로 700라인의 코드가 이들 취약성에 관련이 있는 것으로 보고되었다. 이들 에러의 종류와 TEC 측정치, 그리고 각 취약성에 대한 위험성과 침투성 수준은 〔표 7〕과 같다.

〔표 7〕 P2P프로그램 정적분석결과

취약성 이름	위험성 (Sv)	침투성 (Exp)	측정치 (TEC)
버퍼 오버런	5	5	14
버퍼 언더런	4	3	4
위험한 함수 캐스팅	5	4	40
0으로 나누기	4	4	4
Null 포인터의 free	5	5	5
반환값의 무시	3	3	5
누수(leak)	3	2	31
음수 파일 설명자	4	4	27
Null 포인터 역참조	4	4	1
역참조 후의 Null 포인터 테스트	4	4	9
중복 컨디션	3	3	109
초기화 안 된 변수	3	3	109
도달 불가능한 코드	3	3	85
미 사용 값	4	4	114
Free 후 사용	2	2	69
불필요한 할당문	2	2	29
new[]로 생성된 객체의 제거	5	4	2
잘못된 malloc 버퍼길이	4	4	43
소 계			700

각 위험 수준별 취약 등급을 판정하기 위해서는 〔표 7〕에 나타난 위험성과 침투성을 근거로 취약성값 V를 산출한다. 이들은 〔표 2〕의 취약성 수준 평가 기준을 참고로 Very High에서 Very Low의 수준별로 분류하였다〔표 8〕.

〔표 8〕을 통해 평가자는 Very High의 취약성을 갖는 함수들이 어떤 것들이고, 이들의 빈도수는 어떠한지를 알 수 있으며, 취약성 수준별 분포 관계 역시 유도할 수 있다. 예를 들면, L5 수준의 고위험도 V값의 총 합은 140+50+360+18 = 568이며, 구간값 VL(VeryHigh) =568/4575 = 12.42% 구간에

〔표 8〕 취약성 수준별로 재배치된 P2P 프로그램 취약성값

취약성 수준	취약성 이름	위험성 (Sv)	침투성 (Exp)	측정치 (TEC)	취약성 값 (V)
Very High	버퍼 오버런	5	5	14	140
	Null 포인터의 free	5	5	5	50
	위험한 함수 캐스팅	5	4	40	360
	new〔〕로 생성된 객체의 제거	5	4	2	18
High	0으로 나누기	4	4	4	32
	음수 파일 설명자	4	4	27	216
	Null 포인터 역참조	4	4	1	8
	역참조 후의 Null 포인터	4	4	9	72
	미 사용 값	4	4	114	912
	잘못된 malloc 버퍼길이	4	4	43	344
	버퍼 언더런	4	3	4	28
Medium	반환값의 무시	3	3	5	30
	중복 컨디션	3	3	109	654
	초기화 안 된 변수	3	3	109	654
	도달 불가능한 코드	3	3	85	510
	누수(leak)	3	2	31	155
Low	Free 후 사용	2	2	69	276
	불필요한 할당문	2	2	29	116
	소 계			700	4575

〔표 9〕 P2P 프로그램에 대한 위험등급 판정

위험수준	구간별 TEC합	구간별 V합	구간값 (VL)	판정
L5	61	568	12.42%	Very High
L4	1612	1612	35.23%	
L3	2003	2003	43.78%	
L2	392	392	8.57%	

서 Very High의 위험성이 존재함을 알려준다. 따라서 이 경우 P2P 프로그램은 〔표 3〕을 기준으로 판단할 경우 VL(VeryHigh)값이 5% 이상이므로 이 프로그램의 위험수준은 Very High 등급에 해당한다.

다음으로 정보 중요도 요소가 반영된 메트릭을 산출하는 과정을 살펴본다. 먼저, P2P 파일 다운로드/업로드 프로그램에 의해 공유되는 자산의 가치를 평가하는 일이 선행되어야, 이는 통상적으로 기관의 정보화 담당자에 의해 이루어진다. 본 논문에서 P2P 프로그램의 자료 중요도는 High 수준이며 서비스 중요도는 Medium 수준이라 가정한다. 자료 중요도의 보정계수와 서비스 중요도 보정계수는 1이라 가정한다. 앞서 계산된 VL 값과 정보 중요도, 서비스 중요도에 근거하여 최종 소스코드 보안 값 SSV를 계산하면 다음과 같다.

$$SSV = (CIA \times C1 + S \times C2) \times VL = (4 \times 1 + 3 \times 1) \times 5 = 35$$

이 경우 〔표 6〕에 근거한 위험 구간의 소스코드 보안값의 범위는 50~32 사이이며 따라서 평가 대상 P2P

〔표 10〕 P2P 프로그램에 대한 소스코드 보안값 메트릭

		서비스별 위험수준				
		VH(5)	H(4)	M(3)	L(2)	VL(1)
자료 중요도	VH(5)					
	H(4)	○				
	M(3)					
	L(2)					
	VL(1)					
서비스 중요도	VH(5)					
	H(4)					
	M(3)	○				
	L(2)					
	VL(2)					

프로그램의 소스코드 보안수준은 위험구간 내에 있다고 판정된다.

4.4 소스코드 보안성 메트릭의 평가

소스코드 보안성 메트릭의 특징을 살펴보기 위해 본 논문은 공통 취약성 스코어 시스템(CVSS), 사이클로메틱 복잡도(CC)와 정보흐름 메트릭(IFC), 행정안전부 통합인증 프레임워크, 국가정보원의 보안관리수준 평가 메트릭 등과의 비교 결과를 제시한다. 메트릭의 특성을 비교하는 과정에서 적용된 기준들은 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 11〕은 이들 기준을 바탕으로 본 논문의 제안 메트릭과 다른 메트릭과의 특성을 비교한 것이다.

본 논문의 제안 메트릭은 소스코드의 취약성을 탐지하기 위한 정적분석 결과를 이용하므로 보안지향성은 강한편이다. 등급의 구분 역시 명확하다고 판단된다. 또한 선택성에 있어서도 소스코드 수준 뿐 아니라 자료 중요도와 서비스 중요도를 추가로 반영할 수 있도록 두 단계의 산출 방식을 제공함으로써 융통성을 높일 수 있다. 참고로, 보안지향성과 관련하여, CC의 경우 높은 사이클로매틱 복잡도는 프로그램의 유지보수를 어렵게 하는 것으로 알려져 있다〔표 12〕. 그러한 이유로 프로그램의 복잡도 메트릭은 공통평가기준에서 활용되기도 하지만 CC 복잡도와 보안성의 상관관계는 명확히 설명되지 못하고 있다.

〔표 12〕 CC 복잡도의 수준

CC	프로그램 종류	위험성
1~4	단순한 프로그램	낮음
5~10	잘 구조화되고 안정된 프로그램	낮음
11~20	다소 복잡한 프로그램	중간
21~50	상당히 복잡하며, 조심해야할 프로그램	높음
〉50	극히 복잡하여 문제를 일으킬 수 있는 불안정한 프로그램	매우 높음

등급의 구체성과 관련하여 본 논문의 제안 메트릭은 소스코드 수준과 정보중요도 수준에서 구체적인 등급 구분을 제시하고 있다. 통합인증 프레임워크와 유사한 등급 계산과정을 제시하는 국가사이버안전 매뉴얼〔12〕의 경우 등급 분류에 있어 수행업무의 중요도(E1~E2) 및 이의 가중치 (W1~W2), 정보통신망 및 정보 중요도(E3~E4) 및 이의 가중치(W3~W4), 피해분석 정도(E5~E6) 및 이의 가중치(W5~W6)을 반영한 총점 T를 계산하며, 이의 평가 수준을 〔표 12〕에 준하여 산정한다.

$$T = \sum_{i=1}^{7} E_i \times W_i$$

이 경우 3개 등급의 분류기준으로 제시된 점수 구간 값은 일반적으로 받아들여져 사용되고 있다. 본 논문에서 제시한 〔표 2〕의 취약성 구분표는 CERT C에

〔표 11〕 메트릭의 특성비교

메트릭	목표	보안 지향성	이해 용이성	확장성	등급	표준화	선택성
CVSS	실행취약성	●	○	⊙	○	●	●
CC 및 IFC	모듈복잡도	⊙	●	○	○	⊙	○
통합인증 프레임워크	정보시스템 보안성	⊙	●	○	⊙	●	●
국가사이버안전 메뉴얼	기관등급분류	⊙	●	○	⊙	●	○
제안 메트릭	소스코드 취약성	●	⊙	⊙	●	○	●

●: 강함, ⊙: 중간, ○: 약함, ×: 없음

서 제시된 함수별 취약점 수준을 참고하여 구성되었으며, [표 6]의 소스코드 위험성 판정등급표 역시 Very High와 High요소에 대해서는 보수적인 관점에서 접근하도록 위험 수준을 설정하였다.

[표 13] 국가사이버안전 메뉴얼의 보안성 수준 구분

평가수준	분류 기준
가급	$31 \leq T \leq 39$
나급	$21 < T \leq 30$
다급	$13 < T \leq 20$

V. 결 론

본 논문에서는 소스코드의 보안수준을 평가하기 위한 목적의 소스코드 보안성 평가 메트릭을 소개하였다. 소스코드 보안성 평가 메트릭 적용은 두 부분으로 구성된다. 첫 째, 소스코드 자체에 내재되어 있는 보안 약점(security weakness)에 관한 메트릭 산출이다. 이는 정적분석을 통해 얻어진 결과와 각 함수별 위험성과 침투성 정보를 바탕으로 계산된다. 둘 째, 이 결과는 프로그램이 처리하는 자료의 보안 특성인 무결성, 기밀성, 가용성의 요소를 반영하는 정보 중요도와 서비스 중요도와 함께 최종 소스코드의 보안수준 평가에 반영이 된다.

본 연구는 소스코드가 갖는 특징 중 보안성 관련 특징에 국한하여 소스코드의 특성을 정량화시키고자 노력하였다. 정량화된 소스코드의 취약성 수준은 이를 사용하는 이해 당사자들에게 다음과 같은 이점을 제공한다. 먼저, 개발자는 소스코드로부터 발생 가능한 취약성을 코딩 단계에서부터 통제함으로써 운영 시 발생하는 취약성 문제를 사전에 차단할 수 있다. 또한 버전이 바뀌어 릴리즈되는 프로그램에 대해 보안성이 개선되는 과정을 정량적으로 추적할 수 있다. 발주자는 의뢰한 시스템이 원하는 수준의 보안성을 제공하고 있는지를 객관적으로 판단할 수 있다. 측정된 소스코드 취약성 메트릭에 근거하여 보안 취약성을 개선하고자 하는 계획을 수립할 경우 자원과 일정을 배정하는 우선순위 계획에 활용할 수 있다.

본 논문에 소개된 판정 등급의 구간 설정은 응용 시스템의 복잡도 특성, 구현 언어별 특성, 요구되는 보안 수준에 따라 튜닝이 될 필요가 있다. 이에 대한 근거 데이터는 현재 수집 중에 있으며, 향후 누적된 데이터 분석을 통해 정밀한 구간 값의 설정이 가능해지리라 기대한다.

참고문헌

[1] The Department of Homeland Security, Practical Measurement Framework for Software Assurance and Information Security, http:// buildsecurityin.us-cert.gov/, Oct, 2008.

[2] G. McGraw, Software Security: Building Security In, Addison Wesley, pp. 83-86, Feb. 2006.

[3] J. West, Secure Programming with Static Analysis, Addison-Wesley, pp. 11-13, Jun. 2007.

[4] L. Laird and M Brennan, Software Measurement and Estimation: A Practical Approach, Wiley Inter-Science, pp. 58-67, Jun. 2006.

[5] Common Criteria, version 3.1, Part 3, http://commoncriteriaportal.org, 2006.

[6] National Vulnerability Database Version 2.2, http://nvd.nist.gov/

[7] Common Vulnerability Score System version 2.0, http://nvd.nist.gov/ cvss.cfm, 2007.

[8] Common Vulnerability and Exposures, http://cve.mitre.org/index.html.

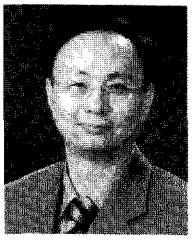
[9] R. Seacord, The CERT C Secure Coding Standard, Addison Wesley, pp. 25-27, Oct. 2008.

[10] DACS, Enhancing the Development Life Cycle to Produce Secure Software, pp. 149-177, http://www.thedacs.com, Oct. 2008.

[11] 통합인증프레임워크 가이드, 행정안전부, p. 24, 2009년 8월.

[12] 국가 사이버안전매뉴얼, 5장 보안관리수준 평가, 국가정보원 pp.97-99, 2005년 10월.

〈著者紹介〉



서 동 수 (Dongsu Seo) 정회원
1986년 6월: 중앙대학교 컴퓨터공학과 졸업
1991년 6월: University of Manchester, Dept. of Computation 석사
1994년 6월: University of Manchester, Dept. of Computation 박사
1998년 2월~현재: 성신여자대학교 IT 학부 교수
〈관심분야〉 정보보호, 소프트웨어공학