공개용 소스코드 보안약점 분석도구 개발 동향

방지호*

현재 국가 정보화사업으로 개발되는 정보시스템 소프트웨어는 개발단계부터 보안약점(보안 취약점의 원인)이 배제되도록 개발하는 '소프트웨어 개발보안' 적용이 의무화 되었다. 소프트웨어 개발보안을 적용하여 안전한 소프트웨어를 개발 및 구현하기 위해 정적분석 도구 등 다양한 자동화 도구가 개발 및 활용되고 있다. 국가 정보화사업에 대한 감리 시 감리법인이 보안약점을 진단하기 위해 정적분석 기반의 자동화 도구인 소스코드 보안약점 분석도구를 사용하는 경우 '정보보호시스템 평가 · 인증 지침'에 따라 평가 · 인증 받은 제품을 사용하는 것이 의무화 되어, 현재(2014년 5월 기준) 관련 유형의 제품에 대한 평가가 진행 중이며, 인증된 제품도 최근 2종이 출시되었다.

본 기고에서는 상용제품 이외에 손쉽게 접할 수 있는 오픈소스로 공개된 정적분석 기반의 보안약점 분석도구를 대상으로 보안약점 분석 성능을 분석하고자 한다. 보안약점 성능 분석 결과는 중소기업 등 영세한 민간기업에서 소프트웨어 개발 시 참고하여 활용할 수 있는 공개용 분석도구에 대한 정보를 제공하여 소프트웨어 개발 안전성 향상에 기여 할 수 있을 것으로 기대한다.

* 한국인터넷진흥원 보안평가팀 책임연구원(jhbang@kisa.or.kr)

Ⅰ. 서론

현재 국가 정보화사업으로 개발되는 정보시스템 소프트웨어(이하, SW)는 개발단계부터 보안취약점(Vulnerability)의 원인인 보안약점(Weakness)[1]을 배제하도록 개발하는 'SW 개발보안' 적용이 의무화 되었다. SW 개발보안 단계는 요구사항 분석 단계, 설계 단계, 구현 단계, 시험 단계, 릴리즈 단계, 운영 단계로 구분되며 다음과 같이 수행된다.

'요구사항 분석 단계'는 개발대상 SW가 다루는 정보(및 데이터)와 SW가 제공하는 서비스를 대상으로 보안위협을 도출하여, 도출된 보안위협을 대응(제거, 완화, 회피 등)하기 위한 보안요구사항을 도출하는 단계이다.

'설계 단계'는 보안위협을 기반으로 도출된 보안요구사항이 SW 기능으로 구현되도록 보안 구조 및 인터페이스, 알고리즘 등을 설계하는 단계이다.

'구현 단계'는 보안요구사항에 대한 설계 내용을 프로그래밍 언어(예, JAVA, C 등)로 보안약점이 내재되지 않도록 안전하게 구현하는 단계로, 현재 국내에서 의무화가 적용된 단계이다.

'시험 단계'는 보안요구사항에 따른 보안기능 등이 적절하게 동작하는지를 단위 모듈 · 서비스 시험, 통합 모듈 · 서비스 시험 등을 통해 확인하는 단계이다.

'릴리즈 단계'는 시험 단계를 통해 보안요구사항이 적정하게 구현된 것을 확인된 경우, SW 컴파일 등을 통한 배포본을 제작하는 단계이다.

'운영 단계'는 SW 배포본을 설치하여 정상적인 IT 서비스를 제공하는 단계로 주기적인 보안취약점을 점검하여 안전한 서비스를 제공해야 한다. 보안취약점이 발생하는 경우 관련 보안패치를 적용하여 안전한 서비스를 제공해야 한다.

안전한 SW 개발을 위해 SW 개발 단계별로 다양한 자동화 도구가 개발되고 있는데, 국내에 의무화된 구현 단계의 경우 소스코드를 대상으로 SW 에러, 버그 등 보안약점을 분석해주는 정적분석 도구가 대표적이다. 최근 안전한 SW 개발을 위해 보안약점을 분석할 수 있는 정적분석 도구가 많이 활용되고 있는데, 국내 국가 정보화사업에 대한 감리 시 감리법인이 보안약점을 진단하기 위해 정적분석 기반의 자동화 도구인 '소스코드 보안약점 분석도구'를 사용하는 경우 '정보보호시스템 평가 · 인증 지침'에 따라 평가 · 인증 받은 제품(CC인증 제

1 보안약점(Weakness)은 잠재적인 보안취약점(Potential Vulnerability)으로 해킹 등 사이버공격을 유발할 수 있는 SW 결함, 오류 등으로, 관련 보안약점 목록은 MITRE社(http://cwe.mitre.org)에서 제공하고 있음

품)을 사용하는 것이 의무화[2] 되었다.

본 기고에서는 손쉽게 접할 수 있는 오픈소스로 공개된 정적분석 기반의 보안약점 분석도구를 대상으로 보안약점 분석 성능을 분석하고자 한다. 보안약점 성능 분석 결과는 중소기업 등 영세한 민간기업에서 SW 개발 시 참고하여 활용할 수 있는 공개용 분석도구에 대한 정보를 제공하여 소프트웨어 개발 안전성 향상에 기여 할 수 있을 것으로 기대한다.

본 기고는 국내외 개발보안 동향과 공개용 소스코드 보안약점 분석도구 시험기준을 기반으로 성능시험 결과를 분석하고, 결론을 맺는다.

Ⅱ. 본론

1. 국내 · 외 개발보안 동향

1) 국외 개발보안 동향

미국 국토안보부(이하, DHS)는 국방부(이하, DoD) 및 국립표준기술연구소(이하, NIST)와 함께 SW 보증을 위한 다양한 방법론을 연구하고 있다. NIST의 경우, DHS의 지원을 받아 2004년부터 SW 보증을 통한 SW 보안성을 향상시키기 위한 다양한 연구를 수행하는 SAMATE(Software Assurance Metrics And Tool Evaluation) 프로젝트를 수행하고 있다. SAMATE 프로젝트는 SW 보안성을 지원하는 도구 평가 방법, 도구와 개발보안 기법의 차이 및 효과 분석 등을 수행하고 있는데, 소스코드 보안약점 분석도구와 관련하여 분석도구 성능 평가 방법론 및 테스트베드를 제공하고 있다. 성능평가기준으로 소스코드 보안약점 분석도구가 탐지해야 하는 최소 보안약점 기준 항목을 JAVA 언어는 9개, C/C++ 언어는 19개로 정의하고 있다. 성능평가를 위한 테스트베드로 CWE를 기반으로 소스코드에 다양한 보안약점이 포함되도록 구현한 JAVA 및 C/C++ 개발언어 기반의 Juliet 코드[3] 등을 제공하고 있다. SAMATE 프로젝트는 정적분석 기반의 상용 분석도구를 대상으로 2008년부터 매년 상용 분

2 현재(2014년 5월 기준) 관련 유형의 제품에 대한 평가가 진행 중이며, 인증 제품도 출시되었음

3 (참조 사이트: Juilet 코드 등 다양한 예제코드 제공) http://samate.nist.gov/SARD/testsuite.php

석도구 개발업체의 참여하에 SATE(Static Analysis Tool Exposition)를 개최하고 있다. SATE는 참여를 희망하는 상용 분석도구 개발업체를 대상으로 NIST에서 제시하는 공개용 오픈소스를 시험코드로 하여 각 진단도구 특징 및 분석 성능을 시험한다. 2010년에는 공개용 오픈소스 외에 CVE에서 리포팅한 정보를 기반으로 취약한 공개용 오픈소스와 보안패치된 공개용 오픈소스를 한 쌍의 시험코드로 사용하였다. 2012년에는 단순 공개용 오픈소스는 시험코드에서 제외하고, 대신 Juliet 코드를 시험코드에 포함하였다. 2012년 이전까지는 분석도구의 성능보다 SATE에 참여한 진단도구의 특성 분석 및 비교에 초점을 두었으나, 2012년에는 3차례의 SATE 진행 경험을 토대로 SATE에 참여한 분석도구의 성능(정탐, 오탐 등) 관점으로 SATE 진행 결과를 리포트 하였다.

정보보안 관련 연구기관인 MITRE社의 경우, DHS의 지원을 받아 SW의 보안약점 목록인 CWE(Common Weakness Enumeration) 정보를 제공하고 있다. CWE의 보안약점 정보는 정보보호제품을 포함한 IT 제품에서 발견되는 보안취약점 정보를 제공하는 CVE(Common Vulnerabilities and Exposures) 등을 기반으로 도출된 보안취약점의 원인을 기반으로 추가 및 갱신되고 있다. 2008년 9월 CWE 버전 1.0을 시작으로 2012년 10월에 CWE 버전 2.3을 발표하여 총 909개의 항목을 제공하고 있으며, SANS社와 함께 SW를 위험하게 만드는 25개의 CWE 항목을 매년 발표하여 SW 개발 시 참고하도록 하고 있다. 그리고 웹 어플리케이션에 대한 보안을 연구하여 공유하고 있는 OWASP(The Open Web Application Security Project)는 매년 전문가 의견을 수렴하여 가장 위험한 10개의 보안위험, 즉 웹 어플리케이션에 특화된 주요 보안약점을 발표하고 있다.

2) 국내 개발보안 동향

안전행정부는 한국인터넷진흥원(이하, KISA)와 함께 2009년부터 SW 개발보안 제도 도입 및 정착을 위해 SW 개발보안 관련 가이드를 개발하여 배포하고 있으며, 정보화사업 담당 공무원 및 개발자 등을 대상으로 SW 개발보안 교육과정을 운영하고 있다. 또한, 전자정부지원사업으로 개발되는 SW를 대상으로 KISA가 자체 개발한 SW 보안약점 분석도구를 기반으로 SW 보안약점 시범진단 서비스를 제공하였는데 SW 라인수는 평균 87만라인으로 2010년도의 경우 평균 2,400개의 보안약점을, 2011년도의 경우 평균 1,328개의 보안약점을 진단하여 조치하도록 하였다.

안전행정부와 KISA의 지속적인 SW 개발보안 관련 정책 추진에 따라, 2012년 6월 '정보시스템 구축 · 운영 지침'이 개정 · 고시되어 정보시스템 감리 대상 전자정부 SW는 개발보안을

적용하여 안전하게 개발하도록 의무화되었다. 이에 따라, 전자정부 SW를 개발하는 행정기관은 자체적으로 분석도구 및 관련 전문가 등을 활용하여 SW 보안약점 존재여부를 진단하여 조치해야 한다. 보안약점 진단 시, 이를 효과적으로 지원할 수 있는 분석도구 개발을 위해 최소 기능 요구사항 도출과 신뢰할 수 있는 분석도구 선택을 위한 평가기준이 필요하다. 관련 지침에서 SW 보안약점 진단을 위해 분석도구를 사용하는 경우 국가정보원장이 정보보호시스템 보안성 평가 · 인증(Common Criteria, 공통평가기준)한 분석도구를 사용(2014년 1월 이후) 하도록 의무화하여, 현재(2014년 5월 기준) 인증된 분석도구 2종이 출시되었다. 최근 2013년 8월 해당 지침이 일부 개정되어 '행정기관 및 공공기관 정보시스템 구축 · 운영 지침'[4]이 고시되면서 안전한 SW 개발을 위해 필수 진단하여 제거해야 하는 SW 보안약점 기준이 개정되어 43개 항목에서 47개 항목으로 조정되었다.

2. 공개용 소스코드 보안약점 분석도구 유형 및 시험기준

'행정기관 및 공공기관 정보시스템 구축 · 운영 지침'에 따라, 정보시스템 감리 시 사용하는 소스코드 보안약점 분석도구는 CC인증 제품 사용이 의무화 되었으나 SW 개발이나 자체 보안약점 진단을 위해 사용하는 경우 CC인증 제품 사용에 대한 제약이 없다. 따라서, 본 기고에서는 국가 정보화사업 SW 개발 시 많이 사용되는 프로그래밍 언어인 JAVA 언어 기반의 분석도구를 대상으로 손쉽게 접할 수 있는 공개용 분석도구 개발 현황 및 국내 기준 부합 수준을 분석하여 개발자 등에 관련 정보를 제공하고자 한다.

NIST SAMATE 프로젝트는 SW의 품질 및 보안성 개선을 위해 정적분석 기반의 보안약점 분석도구(상용 · 공개용)를 포함하여 SW 개발단계별 사용할 수 있는 다양한 도구 목록을 제공하다. 소스코드 보안약점 분석도구 중 공개용 분석도구는 〈표 1〉과 같으며, JAVA 언어로 구현된 소스코드의 보안약점을 분석할 수 있는 공개용 분석도구는 FindBugs, FindSecurityBugs, JLint, LAPSE+, PMD, Yasca 등 6종이다.

또한, NIST SAMATE 프로젝트는 JAVA 기반의 보안약점 분석도구의 성능을 측정할 수 있는 시험코드인 Juliet 코드를 제공하고 있는데 국내에서 의무화한 47개 보안약점 중 24개 보안약점에 대한 시험코드가 포함되어 있다.

4 행정기관 및 공공기관 정보시스템 구축·운영 지침(안전행정부고시 제2013-36호, 2013.8.27.)

〈표 1〉 국외 공개용 소스코드 보안약점 분석도구 현황

[O : 해당사항 있음, — : 해당사항 없음]

도구명		분석대상 언어			
		JAVA	C	C++	기타
1	ABASH	–	–	–	Bash
2	BOON	–	O	–	–
3	Clang Static Analyzer	–	O	–	Objective–C
4	Closure Compiler	–	–	–	JavaScript
5	Cppcheck	–	O	O	–
6	CQual	–	O	–	–
7	Csur	–	O	–	–
8	FindBugs	O	–	–	Groovy, Scala
9	FindSecurityBugs	O	–	–	Groovy, Scala
10	Flawfinder	O	O	–	
11	Jlint	O	–	–	–
12	LAPSEO	–	–	–	
13	PHP–Sat	–	–	–	PHP
14	Pixy	–	–	–	PHP
15	PMD	O	–	–	–
16	pylint	–	–	–	Python
17	RATS (Rough Auditing Tool for Security)	–	O	O	Perl, PHP, Python
18	Smatch	–	O	–	–
19	Splint	–	O	–	–
20	UNO	–	O	–	–
21	Yasca	O	O	O	JavaScript, ASP, ColdFusion, PHP, COBOL, .NET 등

※ 출처 : NIST SAMATE, 〈http://http://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html〉

분석도구 성능분석을 위한 성능결과 판정 기준은 다음 〈표 2〉에 서술한 기준을 기반으로 보안약점이 내포된 Bad 코드에서 취약한 함수(또는 메소드) 등의 위치를 정확하게 탐지한 경우 '정탐(True Positive)'으로 판정하며 그렇지 못한 경우 '미탐(False Negative)'으로 판정한다. 또한, 보안약점이 해결되어 보안약점이 존재하지 않는 Good 코드에서 보안약점이 존재한다고 진단하는 경우 '오탐(False Positive)'으로 판정한다. '정탐'과 '미탐'은 서로 역(逆) 관계에 있기 때문에 본 기고에서는 '정탐'과 '오탐'에 대해서만 논하기로 한다.

〈표 2〉 시험코드 기반 분석결과 판정기준

구분	코드유형	판정기준
정탐	Bad	· 보안약점이 내재된 함수 등의 위치를 정확하게 탐지한 경우
오탐	Good	· 보안약점이 해결된 함수 등을 탐지하는 경우
미탐	Bad	· 보안약점이 내재된 함수 등을 탐지 못하는 경우

3. 공개용 소스코드 보안약점 분석도구 성능 분석

본 절에서는 JAVA 프로그래밍 언어 기반의 공개용 분석도구 6종(〈표 3〉참조)을 대상으로 2절의 분석기준을 적용한 성능 분석 결과를 설명한다.

〈표 3〉 JAVA 프로그래밍 언어 기반의 공개용 분석도구

구분	버전	진단대상	설치 · 운영
FindBugs[27]	2.0.2	바이트코드	Eclipse 플러그인
FindSecurityBugs[28]	1.1.0	바이트코드	FindBugs 모듈
JLint[29]	3.0	바이트코드	CLI
LAPSE+[30]	2.8.1	자바코드	Eclipse 플러그인
PMD[26]	4.0.2	자바코드	Eclipse 플러그인
Yasca[31]	2.1	자바코드	CLI

Juliet 코드는 다양한 유형을 고려하여 개발되었기 때문에 보안약점별 여러 종류의 세부 시험코드가 존재한다. 따라서, 분석도구가 보안약점별 세부 시험코드 중 한 개라도 보안약점을 정탐하는 경우 다양한 유형의 진단규칙 보유가 미흡하나 해당 보안약점을 점검할 수 있는 메커니즘은 보유하고 있다고 할 수 있기 때문에 점검 가능하다고 판정(○ 표시)하였다. 만약 한 개의 정탐도 없는 경우 점검할 수 없다고 판정(× 표시) 하였으며, 보안약점이 포함된 소스코드를 탐지하였어도 분석도구 탐지규칙에 적합하지 않는 경우 오탐으로 판정(× 표시)하였다. 해당 보안약점에 대한 Juliet 코드가 존재하지 않는 경우 분석대상에서 제외(– 표시)하였다.

1) 입력데이터 검증 및 표현

JAVA 언어 기반 Juliet 코드는 입력데이터 검증 및 표현 유형에 대해 10개 보안약점(SQL 삽입, 경로 조작 및 자원 삽입, 크로스사이트스크립트(XSS), 운영체제 명령어 삽입, 신뢰되

지 않는 URL 주소로 자동 접속 연결, XPath삽입, LDAP 삽입, HTTP 응답분할, 정수형 오버플로우, 포맷 스트링 삽입)에 대한 시험코드를 제공하고 있다. 6개의 분석도구 중 PMD를 제외한 5종의 분석도구는 '입력데이터 검증 및 표현'유형 중 1~6개의 보안약점을 탐지하였다. FindBugs, FindSecurityBugs, LAPSE+의 경우, 'HTTP Response Splitting' 진단규칙으로 '신뢰되지 않은 URL 주소로 자동 접속 연결' 보안약점을 식별하여 오탐으로 처리하였다. Yasca의 경우, 'Process Control' 진단규칙으로 'OS 명령어 삽입' 보안약점을 식별하였으나 해당 진단규칙은 절대경로 지정 없이 임의의 라이브러리를 로드하였을 때 발생할 수 있는 보안약점이기 때문에 오탐으로 처리하였다.

〈표 4〉 분석도구 성능분석 : 입력데이터 검증 및 표현

보안약점	CWE	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
SQL 삽입	89	O	O	×	O	×	O
경로 조작 및 자원 삽입	22/99	O	O	×	O	×	×
크로스사이트 스크립트	80	O	O	×	O	×	×
운영체제 명령어 삽입	78	×	O	×	O	×	×
위험한 형식 파일 업로드	434	–	–	–	–	–	–
신뢰되지 않는 URL 주소로 자동접속 연결	601	×	×	×	×	×	×
XQuery 삽입	652	–	–	–	–	–	–
XPath 삽입	643	×	×	×	O	×	×
LDAP 삽입	90	×	×	×	×	×	×
크로스사이트 요청 위조	352	–	–	–	–	–	–
HTTP 응답분할	113	O	O	×	O	×	×
정수형 오버플로우	190	×	×	O	×	×	×
보안기능 결정에 사용 되는 부적절한 입력값	807	–	–	–	–	–	–
메모리 버퍼 오버플로우	119	–	–	–	–	–	–
포맷 스트링 삽입	134	×	×	×	×	×	×

2) 보안기능

JAVA 언어 기반 Juliet 코드는 보안기능 유형에 대해 7개 보안약점(취약한 암호화 알고리즘 사용, 중요정보 평문전송, 하드코드된 비밀번호, 적절하지 않은 난수값 사용, 하드코드된 암호화 키, 주석문 안에 포함된 패스워드 등 시스템 주요정보, 솔트없이 일방향해쉬 함수 사용)에 대한 시험코드를 제공하고 있다. 6개의 분석도구 중 FindBugs, FindSecurityBugs, Yasca는 '보안기능' 유형 중 1~2개의 보안약점을 탐지하였다. FindBugs, FindSecurityBugs는

'Hardcoded constant database password' 진단규칙으로 '사용자 중요정보 평문저장 및 전송' 보안약점을 식별하여 오탐으로 처리하였다.

〈표 5〉 분석도구 성능분석 : 보안기능

보안약점	CWE	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
적절한 인증 없는 중요기능 허용	306	–	–	–	–	–	–
부적절한 인가	285	–	–	–	–	–	–
중요한 자원에 대한 잘못된 권한 설정	732	–	–	–	–	–	–
취약한 암호화 알고리즘 사용	327	×	O	×	×	×	O
중요정보 평문저장	312	–	–	–	–	–	–
중요정보 평문전송	319	×	×	×	×	×	×
하드코드된 비밀번호	259	O	O	×	×	×	×
충분하지 않은 키 길이 사용	310	–	–	–	–	–	–
적절하지 않은 난수 값 사용	330	×	O	×	×	×	×
하드코드된 암호화 키	321	×	×	×	×	×	×
취약한 비밀번호 허용	521	–	–	–	–	–	–
사용자 하드디스크에 저장되는 쿠키를 통한 정보 노출	539	–	–	–	–	–	–
주석문 안에 포함된 시스템 주요정보	615	×	×	×	×	×	×
솔트 없이 일방향 해쉬함수 사용	759	×	×	×	×	×	×
무결성 검사없는 코드 다운로드	494	–	–	–	–	–	–
반복된 인증시도 제한 기능 부재	307	–	–	–	–	–	–

3) 시간 및 상태

JAVA 언어 기반 Juliet 코드는 시간 및 상태 유형에 대해 1개 보안약점(종료되지 않는 반복문 또는 재귀 함수)에 대한 시험코드를 제공하고 있다. 6개의 분석도구 중 FindBugs, FindSecurityBugs는 'An apparent infinite recursive loop' 진단규칙으로 '제어문을 사용하지 않는 재귀함수' 보안약점을 탐지하였다.

〈표 6〉 분석도구 성능분석 : 시간 및 상태

보안약점	CWE	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
경쟁조건: 검사 시점과 사용 시점(TOCTOU)	367	–	–	–	–	–	–
종료되지 않는 반복문 또는 재귀 함수	674/835	O	O	×	×	×	×

4) 에러처리

JAVA 언어 기반 Juliet 코드는 에러처리 유형에 대해 2개 보안약점(오류메시지를 통한 정보노출, 오류상황 대응 부재)에 대한 시험코드를 제공하고 있다. 6개의 분석도구 중PMD만 'EmptyCatchBlock' 및 'AvoidPrintStackTrace' 진단규칙으로 2개의 '에러처리' 유형 보안약점을 탐지하였다.

〈표 7〉 분석도구 성능분석 : 에러처리

보안약점	CWE	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
오류 메시지를 통한 정보 노출	209	×	×	×	×	O	×
오류 상황 대응 부재	390	×	×	×	×	O	×
부적절한 예외 처리	754	–	–	–	–	–	–

5) 코드오류

JAVA 언어 기반 Juliet 코드는 코드오류 유형에 대해 2개 보안약점(Null Pointer 역참조, 부적절한 자원 해제)에 대한 시험코드를 제공하고 있다. 6개의 분석도구 중 FindBugs, FindSecurityBugs는 'Null pointer dereference' 진단규칙으로, JLint는 'Data Flow' 진단규칙으로 '널 포인터 역참조' 보안약점을 탐지하였다.

〈표 8〉 분석도구 성능분석 : 코드오류

보안약점	CWE	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
Null Pointer 역참조	476	O	O	O	×	×	×
부적절한 자원 해제	404	×	×	×	×	×	×
해제된 자원 사용	416	–	–	–	–	–	–
초기화되지 않은 변수 사용	457	–	–	–	–	–	–

6) 캡슐화

JAVA 언어 기반 Juliet 코드는 캡슐화 유형에 대해 2개 보안약점(제거되지 않고 남은 디버그 코드, 시스템 데이터 정보노출)에 대한 시험코드를 제공하고 있다. 6개의 분석도구 중 Yasca만'Authorization: 'Debug' Parameter Found' 진단규칙으로 '제거되지 않고 남은 디버그 코드' 보안약점을 탐지하였다.

〈표 9〉 분석도구 성능분석 : 캡슐화

보안약점	CWE	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
잘못된 세션에 의한 데이터 정보 노출	488	–	–	–	–	–	–
제거되지 않고 남은 디버그 코드	489	×	×	×	×	×	O
시스템 데이터 정보노출	497	×	×	×	×	×	×
Public 메소드부터 반환된 Private 배열	495	–	–	–	–	–	–
Private 배열에 Public 데이터 할당	496	–	–	–	–	–	–

7) API오용

JAVA 언어 기반 Juliet 코드는 API 오용 유형의 보안약점(DNS lookup에 의존한 보안결정, 취약한 API 사용)에 대한 시험코드를 제공하고 있지 않기 때문에 해당 보안약점에 대한 분석은 생략한다.

〈표 10〉 분석도구 성능분석 : API오용

보안약점	CWE	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
DNS lookup에 의존한 보안결정	247/350	–	–	–	–	–	–
취약한 API 사용	676	–	–	–	–	–	–

Ⅲ. 결론

본 기고에서는 소스코드 보안약점 47개 항목 중 Juliet 코드를 통해 시험할 수 있는 24개 보안약점을 대상으로 공개용 분석도구 6종에 대한 진단규칙 보유 관점의 성능시험을 수행하

였다. 성능분석 결과(〈표 11〉참조), 모든 보안약점을 분석할 수 있는 규칙 및 성능을 보유한 분석도구는 존재하지 않았다. 따라서, 소스코드 보안약점을 분석하기 위해서는 '정보보호시스템 평가 · 인증 지침'에 따라 인증된 분석도구를 사용하는 것이 필요하며, 공개용 분석도구를 사용해야 하는 경우 본 기고의 분석결과를 기반으로 여러 개의 도구를 같이 사용하여 분석범위를 넓히고 분석도구가 분석하지 못하는 보안약점에 대한 규칙을 추가하는 것이 필요하다. 또한, 국가 정보화사업 등과 같이 대규모의 SW 분석 시 분석도구의 메모리 용량 등이 중요하므로 공개용 분석도구 사용시 힙 메모리를 충분히 설정하여 분석하는 것이 필요하다.

본 기고의 분석도구 성능 분석 결과는 중소기업 등 영세한 민간기업에서 소프트웨어 개발 시 참고하여 활용할 수 있는 공개용 분석도구에 대한 정보를 제공하여 소프트웨어 개발 안전성 향상에 기여 할 수 있을 것으로 기대한다.

본 기고에서 분석도구 성능분석을 위해 사용된 Juliet 코드는 인위적으로 개발된 예제코드로 실제 적용 결과와 차이가 존재할 수 있기 때문에 향후에는 오픈소스 SW 등 다양한 코드를 대상으로 성능분석을 수행할 예정이다. 또한, C 프로그래밍 언어 기반의 공개용 소스코드 보안약점 분석도구에 대한 성능분석을 수행하고, 퍼징도구와 같은 동적분석 도구로 성능분석 영역을 확장할 예정이다.

〈표 11〉 분석도구 성능분석 결과

보안약점 유형	분석 대상	FindBugs	FindSecurityBugs	JLint	LAPSE+	PMD	Yasca
입력데이터 검증 및 표현(15개)	10	4	5	1	6	–	1
보안기능(16개)	7	1	3	–	–	–	1
시간 및 상태(2개)	1	1	1	–		–	–
에러처리(3개)	2	–	–	–	–	1	–
코드오류(4개)	2	1	1	1		–	–
캡슐화(5개)	2	–	–	–	–	–	1
API 오용(2개)	–	–	–	–	–	–	–
계(47개)	24	7	10	2	6	1	3

참고문헌

미래창조과학부 (2013). '정보보호시스템 평가인증 지침'(미래창조과학부고시 제2013-52호).

방지호, 하란 (2013). '소프트웨어 보안약점 기반의 오픈소스 보안약점 진단도구 분석', 한국정보과학회 2013 한국컴퓨터종합학술대회: 753-755.

안전행정부 (2013). 「소프트웨어 개발보안 가이드」.

안전행정부 (2013). '행정기관 및 공공기관 정보시스템 구축 · 운영지침'(안전행정부고시 제2013-36호).

행정안전부 (2012). 「소프트웨어 보안약점 진단가이드」.

DHS, Build Security In(BSI), 〈https://buildsecurityin.us-cert.gov〉

FindBugs, 〈http://findbugs.sourceforge.net〉

FindSecurityBugs, 〈http://h3xstream.github.io/find-sec-bugs〉

JLint, 〈 http://sourceforge.net/projects/jlint〉

LAPSE+, 〈https://www.owasp.org/index.php/Category:OWASP_LAPSE_Project〉

MITRE, Common Vulnerabilities and Exposures, Retrieved June, 20, 2012, from http://cve.mitre.org.

MITRE, Comon Weakness Enumeration V2.4, Retrieved Feb., 21, 2013, from http://cwe.mitre.org.

NIST SAMATE, 〈http://samate.nist.gov〉

OWASP, OWASP Top Ten 2013 rc1, 〈http://www.owasp.org.〉

PMD, 〈http://pmd.sourceforge.net〉

V. Okun 외, Report on the Static Analysis Tool Exposition(SATE) IV(NIST Special Publication 500-297), 2013.1.

Yasca, 〈http://www.yasca.org〉