한국통신학회논문지 '13-04 Vol.38C No.04
http://dx.doi.org/10.7840/kics.2013.38C.4.335

전자정부 소프트웨어의 보안약점 진단도구 평가방법론

방 지 호◆, 하 란*

Evaluation Methodology of Diagnostic Tool for Security Weakness of e-GOV Software

Jiho Bang◆, Rhan Ha*

요 약

SW 개발단계에서 사이버침해사고의 주요 원인인 SW 보안약점을 진단하여 제거하면 사이버침해사고를 효과적으로 예방할 수 있다. 국내의 경우, SW 개발보안 적용이 의무화되어 SW 보안약점을 제거하는 것이 필수사항이 되었다. 효과적으로 SW 보안약점을 진단하여 제거하기 위해서는 신뢰된 SW 보안약점 진단도구의 도움이 필요하다. 따라서, 본 논문은 국내환경에 적합한 진단도구 기능 요구사항과 진단도구의 신뢰성을 보증할 수 있는 평가방법론을 제안한다. 그리고, 제안된 평가체계의 효과를 분석하기 위한 시범 적용한 결과 및 절차를 제시한다.

Key Words : Weakness, Weakness Diagnostic Tool, Vulnerability, Static Analysis, e-GOV SW

ABSTRACT

If the SW weaknesses, which are the main cause of cyber breaches, are analyzed and removed in the SW development stages, the cyber breaches can be prevented effectively. In case of Domestic, removing SW weaknesses by applying Secure SDLC(SW Development Life Cycle) has become mandatory. In order to analyze and remove the SW weaknesses effectively, reliable SW weakness diagnostic tools are required. Therefore, we propose the functional requirements of diagnostic tool which is suitable for the domestic environment and the evaluation methodology which can assure the reliability of the diagnostic tools. Then, to analyze the effectiveness of the proposed evaluation framework, both demonstration results and process are presented.

Ⅰ. 서 론

최근 사이버침해사고의 약 75%는 소프트웨어(이하 SW) 자체 보안취약점(Vulnerability)을 악용한 것으로 특정 SW의 보안취약점을 대상으로 지능화된 기법을 이용한 APT(Advance Persistent Threat) 공격 등이 전방위적으로 확산되는 추세이다[1].

보안취약점은 악의적인 목적을 갖은 사용자(예, 해커 등) 등에 의해 악용되어 중요정보 유출, 권한 상승, 보안기능 우회 등의 보안사고가 발생하는 SW 보안요구사항, 설계, 기능 관련 속성이다[2]. 보안약점(Weakness)은 SW의 결함, 오류 등으로 해킹 등 사이버공격을 유발할 가능성이 있는 잠재적인 보안취약점(Potential Vulnerability)[3]으로 보안취약점의 원인이라고 할 수 있으나 모든 보안약점이 보안취약점이 되지는 않는다[2]. 보안취약점의 원인이 되는 보안약점을 SW 개발단계에서 배제될 수 있도록 SW를 개발하면 SW 운영단계에서 나타날 수 있는 많은 보안취약점들이 제거될 수 있기 때문에 효과적이라고 할 수 있으며, 이는 개발단계에서 보안약점을 제거하는 것이 운영단계에서 보안취약점 패치 등의 보안취약점 제거비용을 30배정도 감소시킬 수 있다는 美 국

◆ 주저자 : 홍익대학교 컴퓨터공학과 실시간시스템 연구실, jhbang@kisa.or.kr, 정회원
* 홍익대학교 컴퓨터공학과 실시간시스템 연구실, rhanha@cs.hongik.ac.kr
논문번호 : KICS2013-01-057, 접수일자 : 2013년 1월 25일, 최종논문접수일자 : 2013년 4월 9일

립표준기술연구소(NIST)의 연구결과[4]와 MS社의 개발보안 적용 사례[5]를 통해 확인할 수 있다.

따라서, SW 개발단계에서 보안약점을 진단해서 제거하는 SW 개발보안 활동이 필요한데, 이는 개발자에 의한 시큐어코딩(Secure Coding)으로 쉽게 접근할 수 있다. 시큐어코딩은 JAVA, C/C++ 등 프로그래밍 언어를 기반으로 SW를 구현할 때, SQL 삽입이나 크로스사이트스크립트(XSS) 등과 같은 보안약점이 SW에 존재하지 않도록 소스코드를 코딩하는 것으로 언어별 보안약점 및 예제코드를 제시하고 있는 시큐어코딩 가이드를 참고하면 도움이 된다. 국외의 경우, 카네기멜론대학(CMU)의 SW공학연구소(SEI)에서 시큐어코딩 가이드를 개발하여 배포하고 있으며, 국내의 경우, 행정안전부가 한국인터넷진흥원(KISA)과 함께 개발하여 온라인으로 배포[1]하고 있다.

개발자가 시큐어코딩 가이드 등을 참고하여 SW를 개발해도, 개발자의 실수로(또는 의도적으로) 보안약점이 SW에 내포될 수 있기 때문에 개발단계에서 개발자 자체적으로 보안약점 여부를 점검하여 제거하는 것이 필요하며, 검수단계에서 발주기관(또는 신뢰할 수 있는 전문기관)이 보안약점 잔존여부를 최종 점검하여 조치하도록 하는 것이 필요하다. 보안약점 진단시, 사람이 직접 소스코드를 보면서 다양한 보안약점을 찾는 것은 어렵기 때문에 짧은 시간에 보안약점을 점검해주는 자동화된 도구를 사용하면 효과적으로 SW 보안약점을 진단하여 제거할 수 있다.

최근 안전한 SW 개발을 위해 소스코드를 대상으로 보안약점을 진단할 수 있는 정적분석(Static Analysis) 도구가 많이 활용[6,7]되고 있다. 정적분석은 SW 실행을 통한 기능 동작 관점의 분석을 수행하는 동적분석(Dynamic Analysis)에 비해 SW를 실행하지 않고 소스코드를 대상으로 입력데이터 검증, 취약한 API 사용 등 다양한 보안약점을 탐지할 수 있는 장점이 있다. 그러나, 동적분석 도구(예, 웹스캐너)처럼 진단규칙에 의존적이며, 진단결과에 오탐(False Positive)이 존재할 수 있기 때문에 도구에 대한 신뢰성 보증이 필요하다.

행정안전부는 KISA와 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 진단도구(이하, 진단도구) 기반의 SW 보안약점 시범진단 서비스를 제공하였는데 SW 라인수는 평균 87만라인으로 2010년도의 경우 평균 2,400개의 보안약점을, 2011년도의 경우 평균 1,328개의 보안약점을 진단하여 조치하도록 하였다. 2012년 6월 '정보시스템 구축·운영 지침[3]'이 개정·고시되어 정보시스템 감리 대상 전자정부 SW는 개발보안을 적용하여 안전하게 개발하도록 의무화되었다. 이에 따라, 전자정부 SW를 개발하는 행정기관은 자체적으로 진단도구 및 관련 전문가 등을 활용하여 SW 보안약점 존재여부를 진단하여 조치해야 한다. 보안약점 진단시, 이를 효과적으로 지원할 수 있는 진단도구 개발을 위해 최소 기능 요구사항 도출과 신뢰할 수 있는 진단도구 선택을 위한 평가기준이 필요하다. 관련 지침에서 SW 보안약점 진단을 위해 진단도구를 사용하는 경우 국가정보원장이 CC(Common Criteria, 공통평가기준) 인증한 진단도구를 사용(2014년 1월이후) 하도록 의무화하였으나, 관련 지침의 요구사항을 만족하는 CC 평가·인증 진단도구는 아직 없으며, 국외에서도 평가·인증된 사례가 없다. 그래서, 진단도구 신뢰성 보증을 위한 평가기준 및 방법에 대한 연구가 필요하다. 기존 연구들은 대부분 SQL 삽입, XSS 등과 같은 특정 보안약점 중심의 진단도구 평가기준 및 방법[2,7,8,9,10]에 대한 연구로 SQL 삽입, XSS를 포함하여 43개 보안약점(지침[3] 별표3)에 대한 요구사항 등 SW 개발보안 적용이 의무화된 국내 환경에 적용하는데 한계가 있다.

따라서, 본 논문은 안전한 전자정부 SW 개발을 지원하는 신뢰성 있는 보안약점 진단도구 개발을 위해 진단도구 기능요구사항을 제시한다. 또한, 보안약점 진단도구의 신뢰성을 평가할 수 있는 평가기준 및 방법 등 진단도구 평가방법론을 제시하여 진단도구 개발업체 및 도입기관에서 활용할 수 있게 하였다.

본 논문은 2장에서는 진단도구 평가 관련 국외 동향을 설명하고, 3장에서는 진단도구의 기능 요구사항을 분석한다. 그리고, 4장에서는 본 논문에서 제안하는 SW 보안약점 진단도구에 대한 평가기준 및 방법과 적용사례를 설명하고, 5장에서 결론을 맺는다.

Ⅱ. 관련 동향

본 장에서는 SW 보안약점 관련 연구 및 보안약점을 진단할 수 있는 정적분석 기반의 진단도구에 대한 평가 관련 연구 동향에 대해 살펴보도록 한다.

2.1. SW 보안약점 관련 연구

정보보안 관련 연구기관인 MITRE社는 美 국토안보부(DHS, Department of Homeland Security)의 지원하에 다양한 관점으로 SW의 보안약점을 분석 및 분류하여 관련 목록인 CWE(Common Weakness Enumeration)[11] 정보를 제공하고 있다. CWE는 정보보호제품을 포함한 IT 제품에서 발견되는 보안취약점 정보를 제공하는 CVE(Common Vulnerabilities and Exposures)[12] 등을 기반으로 보안취약점의 원인을 도출하여 CWE의 보안약점 정보를 추가 및 갱신하고 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 개발시 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[13], 즉 웹 어플리케이션에 특화된 주요 보안약점을 발표하고 있다.

국내의 경우, CWE 등의 정보에 주요 정보화사업을 대상으로 수행한 보안약점 분석결과를 추가로 고려하여 국내 환경에 적합한 보안약점을 도출[14]하였으며, 이를 전자정부 SW 개발단계에서 반드시 제거해야 하는 보안약점 기준으로 의무화[2] 하였다.

2.2. 진단도구 평가 관련 연구

진단도구 평가 관련 기존 연구결과들은 SQL 삽입 및 XSS와 같은 특정 보안약점만 구현한 시험코드를 기반으로 진단도구 평가방법[7]을 제시하거나 진단도구 설치, 설정, 보고서 등의 추가 평가기준을 기반으로 진단도구 평가방법[8]을 제시하고 있다. 그 외에 인위적으로 구현한 시험코드 외에 Wireshark 등과 같은 오픈소스 SW를 이용하여 진단도구 결과를 비교하여 평가하는 방법[9]도 있다. 그러나, 대부분의 연구가 언어별 대표적인 보안약점 1~2개(C/C++ 언어는 버퍼오버플로우, JAVA 언어는 SQL 삽입 및 XSS)에 집중되어 관련 진단결과만 제시하고 있다.

미국의 경우, 정부차원에서 진단도구 평가방법 등을 연구하는 SAMATE(Software Assurance Metrics And Tool Evaluation) 프로젝트를 진행하고 있다. SAMATE 프로젝트는 DHS NCSD(National Cyber Security Division)의 후원을 받아서 NIST에서 2004년부터 시작하여, SW 보증 관련 도구 평가방법론 개발, 보증 기술 및 도구의 효과성 측정 등 SW 보증 향상을 위한 연구를 수행하고 있다. 이에 따라, NIST는 진단도구 기능요구사항[2]과 시험방법[10]을 연구하고 있는데 진단도구가 진단해야 하는 19개(JAVA 9개, C/C++ 19개)의 필수 보안약점을 정의하고 있으며, 정의한 보안약점을 기반으로 진단도구 기능(표1)을 필수 및 선택 요구사항으로 구분하여 제시하고 관련 시험방법(표2)을 제시하고 있다. 또한, 기능 요구사항을 시험할 수 있는 SRD(SAMATE Reference Dataset)[15]와 같은 시험코드를 제시하고 있다.

표 1. 진단도구 기능요구사항(NIST)
Table 1. The functional requirements for analysis tool (NIST)

Feature	Functional Requirement
Mandatory	· Identify 19 weaknesses such as XSS despite the presence of the coding complexities · Textually report any weaknesses that it identifies · For any identified weaknesses in the classes listed such as XSS, report the class using a semantically equivalent name · For any identified weaknesses, report at least one location by providing the directory path, file name and line number · Have an acceptably low false positive rate
Optional	· Produce an XML-formatted report · Not report a weakness instance that has been suppressed · Use the Common Weakness Enumeration [CWE] number and name of the weakness class it reports

표 2. 진단도구 시험방법(NIST)
Table 2. The evaluation methodology for analysis tool(NIST)

Feature	Description
Mandatory	· Test whether to identify weaknesses and report identified weaknesses · Test for determining False Positive Rate
Optional	· Test whether to report a weakness instance that has been suppressed

그러나, NIST의 평가방법론은 국내와 다른 보안약점 기반의 기능요구사항 및 시험방법을 제시하고 있으며, SW 개발보안 의무화에 따른 SW 개발보안 적용 증적자료로 활용하기 위한 요구사항들이 반영되어 있지 않다. 따라서, NIST의 방법론을 그대로 국내환경에 적용하는데 한계가 있다.

Ⅲ. 진단도구 기능요구사항

본 장에서는 전자정부 SW의 보안약점을 진단하기 위해 사용될 수 있는 진단도구 최소 기능요구사항을

제시하기 위해, SW 개발보안 적용 의무화로 SW 개발 시 반드시 제거해야 하는 SW 보안약점 기준[3]을 진단도구 관점에서 분석하고, 이를 기반으로 진단도구 기능 요구사항을 설명한다.

3.1. 진단도구의 보안약점 진단 범위

최근 행정안전부는 '정보시스템 구축·지침'을 개정·고시하여 정보시스템 감리대상 전자정부 SW 개발 시 보안약점이 내포되지 않은 안전한 SW를 개발하도록 SW 개발보안 적용을 의무화 하였다. 해당 지침[2] 별표 3에서는 SW 개발 시 반드시 배제해야 하는 43개의 최소 보안약점 기준을 제시하였다. 해당 보안약점 항목을 정적분석 관점으로 분석하여 진단도구가 진단해야 하는 보안약점 항목을 정의하고자 한다.

3.1.1. 입력데이터 검증 및 표현

'입력데이터 검증 및 표현' 유형의 14개 보안약점 항목 중 '보호메커니즘을 우회할 수 있는 입력값 변조' 보안약점을 제외한 13개 보안약점은 데이터 유입유형과 해당 데이터를 사용하는 함수(메소드)를 정형화할 수 있어서 이를 기반으로 기본적인 진단규칙을 도출할 수 있다. JAVA 언어 관점으로 'SQL 삽입' 보안약점을 예로 들면 표 3과 같다. 그러나, '보호메커니즘을 우회할 수 있는 입력값 변조' 보안약점은 진단대상 SW에 구현된 보호메커니즘의 설계내용을 파악해야 진단규칙을 도출할 수 있다.

따라서, 진단도구는 다음 표 4처럼 'SQL 삽입' 등 13개 보안약점에 대한 기본적인 진단규칙을 보유하는 것이 필요하며, '보호메커니즘을 우회할 수 있는 입력값 변조' 보안약점에 대한 진단규칙은 SW 설계내용을 기반으로 사용자가 진단규칙을 정의할 수 있도록 하는 것이 필요하다. 참고로, NIST[2]는 'SQL 삽입' 등 4개 보안약점에 대한 진단기능만 요구하고 있다.

표 3. 'SQL 삽입' 보안약점 관련 예제
Table 3. Example related to 'SQL Injection' weakness

Feature	Description
Inflow Type	· Web(Servlet, Cookies, Parameter etc.), Config(Consol, Environment, Property etc.) etc.
Method	· execute(), executeBatch(), executeUpdate(), executeQuery() etc.
Code	data = buffread.readLine(); … Boolean bResult = sqlstatement.execute("insert into users (status) values ('updated') where name='"+data+"'");

표 4. 입력데이터 검증 및 표현
Table 4. Input data validation and expression

Weakness	CWE No.	Diagnostic	NIST[2]
SQL Injection	89	Mandatory	Mandatory
Resource Injection	99	Mandatory	Mandatory
XSS	80	Mandatory	Mandatory
OS Command Injection	78	Mandatory	Mandatory
Unrestricted Upload of File with Dangerous Type	434	Mandatory	-
URL Redirection to Untrusted Site	601	Mandatory	-
XQuery Injection	652	Mandatory	-
XPath Injection	643	Mandatory	-
LDAP Injection	90	Mandatory	-
CSRF	352	Mandatory	-
Path Traversal	22	Mandatory	-
HTTP Response Splitting	113	Mandatory	-
Integer Overflow	190	Mandatory	-
Reliance on Untrusted Inputs in a Security Decision	807	Recommend	-

3.1.2. 보안기능

'보안기능' 유형의 보안약점은 특성상 SW 설계 내용 기반의 기능동작으로 보안약점 존재 여부를 확인하는 것이 더 효과적일 수 있다. '취약한 패스워드 허용' 보안약점이 대표적인 예로, 진단도구를 통해 패스워드 설정 규칙의 적절성을 진단하기는 어렵기 때문에 진단대상 SW의 패스워드 보안정책을 파악하고 관련 소스코드를 확인하는 것이 필요하다. 이와 같이 '취약한 패스워드 허용' 등 7개의 보안약점은 진단대상 SW에 구현된 보안속성 및 기능 설계내용을 파악해야 진단규칙을 도출할 수 있으며, 해당 소스코드에 대한 리뷰가 수반되는 것이 필요하다. 그러나, '취약한 암호화 알고리즘 사용' 등 9개의 보안약점은 MD5와 같이 취약한 암호화 알고리즘 등 일반적으로 통용되어 정의할 수 있는 보안속성 및 기능 함수(메소드)가 존재하기 때문에 진단규칙을 정형화 할 수 있다.

따라서, 진단도구는 다음 표 5처럼 '취약한 암호화 알고리즘 사용' 등 10개 보안약점에 대한 기본적인 진단규칙을 보유하는 것이 필요하며, '적절한 인증 없는 중요기능 허용' 등 6개의 보안약점에 대한 진단규칙은 SW 설계내용을 기반으로 사용자가 진단규칙을 정의하고 사용자가 관련 소스코드에 대한 리뷰를 수행하도록 하는 것이 필요하다. 참고로, NIST[2]는 '하드코드된 패스워드' 등 1개 보안약점에 대한 진단기능만 요구하고 있다.

표 5. 보안기능
Table 5. Security features

Weakness	CWE No.	Diagnostic	NIST[2]
Missing Authentication for Critical Function	306	Recommend	-
Improper Authorization	285	Recommend	-
Incorrect Permission Assignment for Critical Resource	732	Recommend	-
Use of Broken or Risky Crypto. Algorithm	327	Mandatory	-
Missing Encryption of Sensitive Data	311	Recommend	-
Use of Hard-coded Password	259	Mandatory	Mandatory
Insufficient Key Size	310	Mandatory	-
Use of Insufficiently Random Values	330	Mandatory	-
Plaintext Storage of Password	256	Mandatory	-
Use of Hard-coded Crypto. Key	321	Mandatory	-
Weak Password Requirements	521	Recommend	-
Info. Through Persistent Cookies	539	Mandatory	-
Sensitive Cookie in HTTPS Session without Secure Attribute	614	Mandatory	-
Info. Exposure Though Comments	615	Mandatory	-
Use of an One-Way Hash without a Salt	759	Mandatory	-
Download of Code Without Integrity Check	494	Recommend	-

표 6. 시간 및 상태 등 5개 유형의 보안약점
Table 6. 5 types weakness, such as Time and Status etc.

Weakness		CWE No.	Diagnostic	NIST[2]
Time & Status	TOCTOU Race Condition	367	Mandatory	Mandatory
	Uncontrolled Recursion	674	Mandatory	-
Errors	Information Exposure through an error message	209	Mandatory	-
	Detection of Error Condition Without Action	390	Mandatory	-
	Improper Check for Unsual or Exceptional Conditions	754	Mandatory	-
Code Quality	NULL Pointer Dereference	476	Mandatory	Mandatory
	Improper Resource Shutdown or Release	404	Mandatory	-
Encapsulation	Exposure of Data Element to Wrong Session	488	Mandatory	-
	Leftover Debug Code	489	Mandatory	Mandatory
	Info. Leak of System Data	497	Mandatory	-
	Private Array-Typed Field Returned From A Public Method	495	Mandatory	-
	Public Data Assigned to Private Array-Typed Field	496	Mandatory	-
APIAbuse	Reliance on DNS Lookups in a Security Decision	247	Mandatory	-

3.1.3. 시간 및 상태 등 5개 유형의 보안약점

‘시간 및 상태’ 등 5개 유형의 13개 보안약점 항목은 데이터 유입유형과 해당 데이터를 사용하는 함수(메소드)를 정형화할 수 있어서 이를 기반으로 기본적인 진단규칙을 도출할 수 있다. 그러나, ‘시스템 데이터 정보노출’ 보안약점은 부적절한 에러 처리시(예, e.printStackTrace() 메소드) 시스템 데이터 정보가 노출될 수 있는데 이는 ‘오류메시지를 통한 정보노출’ 보안약점 발생 시나리오와 유사하기 때문에 관련 소스코드로 명확하게 구분하기가 쉽지 않다.

따라서, 진단도구는 다음 표 6처럼 ‘경쟁조건: 검사시점과 사용시점(TOCTOU)’ 등 13개 보안약점에 대한 기본적인 진단규칙을 보유하는 것이 필요하며, 이때 ‘시스템 데이터 정보노출’ 보안약점은 ‘오류메시지를 통한 정보노출’ 보안약점과 통합하거나, 시스템 데이터를 별도 정의하여 ‘시스템 데이터 정보노출’ 보안약점 진단규칙을 정교화 하는 것이 필요하다. 참고로, NIST[2]는 ’널(Null) 포인터 역참조’ 등 3개 보안약점에 대한 진단기능만 요구하고 있다.

3.2. 진단도구 기능 요구사항

전자정부 SW의 보안약점 진단하기 위해 사용되는 진단도구는 다음 표 7과 같이 진단도구 및 진단대상 SW 식별 정보를 제공할 수 있어야 한다. 그리고, 3.1절에서 분석하여 제시한 SW 보안약점 항목을 기본 진단 항목으로 해서 진단할 수 있어야 하며, 진단결과의 오탐율은 진단도구 도입기관에서 수용할 수 있는 수준이어야 한다. 2010년 및 2011년 전자정부지원사업 SW를 대상으로 보안약점 시범진단을 한 결과, SW의 평균 라인수가 87만라인 정도였으므로 최소 90만라인 정도의 SW는 진단 가능해야 한다. SW의 보안약점을 조치하기 위한 식별한 보안약점 위치정보 및 조치방안 등의 정보를 제공해 주어야 하며, 조치방안의 경우 SW 개발보안 가이드 내용과 일관성이 있어야 한다. SW 개발사업 감리 및 검수시 SW 보안약점 조치여부에 대한 증빙자료로 활용될 수 있는 진단결과를 보고서로 생성할 수 있어야 하며, 오탐으로 판정된 내용이 보고서에 포함되지 않도록 해야 한다. 보안약점 발생 유형은 계속 복잡・다양해지기 때문에 관련 정보가 반영된 최신 진단

규칙이 적용될 수 있도록 진단규칙 업데이트 기능 및 수정 기능을 제공해야 한다.

표 7. 진단도구 기능요구사항
Table 7. The functional requirements for analysis tool

Feature	Functional Requirement
Basic (Identification)	· Provide basic informations of tool (ex, tool name·version, support language etc.) · Provide basic information of target SW (ex, SW name·version, implementation language, diagnostic time, total line etc.)
Diagnostic (Capability)	· Have diagnostic rules for weaknesses listed in Chapter 3.1 · Identify weaknesses listed in chapter 3.1 · Have an acceptably low false positive rate (ex, agency adopted tools determine, within 30~40%, etc.) · Can analyze lots of files (ex, more than 0.9 million line etc.)
Report (Result)	· For any identified weaknesses, provide the weakness name(Guidelines[2]) and the CWE number · For any identified weaknesses, report location informations (ex, directory path, file name, line number etc.) · For any identified weaknesses, provide a definition and modification methods(Refer to the Guide[3]) · Produce a result report
Config (Management)	· A verdic record for diagnostic results · A suppress configuration for false positive · A rule management(insert, modification etc.) · A rule update(automatic/manual)

Ⅳ. 진단도구 평가기준 및 방법론

본 장에서는 본 논문에서 제안하는 진단도구 평가기준 및 방법론을 제시하고, 진단도구를 평가하기 위해 사용된 시험코드 및 시범 적용한 결과를 설명한다.

4.1. 평가기준 및 방법

전자정부 SW에 대한 보안약점 내포여부를 진단하여 이를 제거할 수 있도록 지원하는 자동화된 진단도구의 신뢰성을 평가하기 위해서는 제3장에서 서술한 진단도구 기능요구사항의 구현여부를 점검하고, 진단기능 동작에 따른 진단결과의 적절성 및 정확성을 점검하는 것이다.

진단도구의 기본(식별) 기능을 평가(표8-①~②)하기 위해서는 진단도구의 진단결과에 진단도구의 기본정보 및 진단대상 SW 기본정보가 정확하게 표시되어 있는지 확인해야 한다. 진단도구의 진단(능력) 기능을 평가(표8-③~⑤)하기 위해서는 먼저 진단도구의 진단규칙 정보를 확인하여 제3.1절에서 제안한 SW 보안약점 항목에 대한 진단규칙이 모두 있는지 확인하고, 진단결과를 분석(정탐, 오탐)하여 진단규칙 및 진단결과의 적절성을 확인한다. 이때, 시험코드 기준으로 필수 진단항목은 모두 진단 가능하고, 각 진단항목별 세부 유형은 70% 이상 진단시 적절하다고 판정한다. 전자정부 SW의 경우 소스코드 규모가 크기 때문에 최소 90만라인 이상의 대단위 파일을 분석할 수 있는지 확인해야 한다. 진단도구의 보고(결과) 기능을 평가(표8-⑥~⑨)하기 위해서는 진단결과에서 식별된 보안약점의 위치정보가 정확한지, 각 보안약점별 보안대책 등의 정보를 정확하게 제공하는지 등의 여부를 확인하고 보고서에 누락되지 않게 모두 포함하여 생성 및 출력하는지 확인해야 한다. 진단도구의 설정(관리) 기능을 평가(표8-⑩~⑪)하기 위해서는 보안약점 진단결과 수정(예, 오탐 표시) 기능을 시험하여, 결과보고서에 오탐 설정된 진단결과가 제외되었는지 확인해야 한다. 마지막으로, 보안약점 진단규칙을 추가하거나 수정할 수 있는 관리기능이 존재하는지 여부를 확인해야 한다.

표 8. 진단도구 평가기준
Table 8. The evaluation criteria for analysis tool

Feature	Major Evaluation Criteria
Basic (Identification)	① Determine whether to provide basic informations of tool ② Determine whether to provide basic informations of target SW correctly
Diagnostic (Capability)	③ Determine whether to have diagnostic rules ④ Determine whether to identify weaknesses correctly ⑤ Determine whether to be able to analyze lots of files(more than 0.9 million line)
Report (Result)	⑥ For any identified weaknesses, determine whether to provide weakness name and CWE number ⑦ For any identified weaknesses, determine whether to provide location informations correctly ⑧ For any identified weaknesses, determine whether to provide a definition and modification methods(refer to Guide[3]) ⑨ Determine whether to produce a report
Config (Management)	⑩ Determine whether to be able to suppress weakness instances for false positive ⑪ Determine whether to be able to modify rules

따라서, 진단도구 평가기준은 다음 표 8과 같이 진단도구 및 진단대상 SW에 대한 식별정보 제공 여부 및 정확성을 평가하는 기본항목과 SW 보안약점 진단결과의 적절성을 평가하는 진단항목 등으로 구성된다. 각

항목들은 보안약점을 진단할 수 있는 시험코드를 기반으로 진단도구가 분석한 내용으로 평가를 한다. 주로 진단도구 메뉴 및 동작화면, 진단결과 보고서 등을 점검하여 평가하는데, 시험코드는 각 항목을 평가하기 위한 필수 요소이다. 관련 시험코드는 다음 절에서 설명한다.

4.2. 시험코드

진단도구를 평가하기 위한 시험코드는 NIST의 SAMATE 프로젝트의 일환으로 개발된 SRD의 Juliet 시험세트(v1.0)[15]를 기반으로 개발되었다. 본 논문에서는 간략하게 개발된 시험코드를 설명한다.

시험코드는 보안약점별로 다음 표 9와 같이 한 쌍의 Bad 코드와 Good 코드로 구성하였다.

표 9. 시험코드 구성
Table 9. Test code configure

Type	Description
Bad	Source Code existed Weaknesses
Good	Source Code removed Weaknesses

보안약점은 외부로부터 유입되는 데이터 또는 내부 데이터에 대한 적절한 검증 없이 사용할 경우 발생할 수 있기 때문에 Bad 코드는 내·외부 데이터 유입 부분과 유입된 데이터를 사용하는 부분으로 구성되어 있으며, Good 코드는 적절한 검증된 데이터가 유입되거나 기정의된 데이터가 유입되어 사용될 수 있도록 구성되어 있다. 내·외부 데이터의 유입 유형은 다음 표 10과 같이 다양화 하였다.

표 10. 데이터 유입 유형
Table 10. Data inflow type

Type	Description
Basic	Data defined in SW internal
DB	Data from DBMS, filesystem
Config	Data from console, environment, property etc.
Web	Data from servlet, cookies, parameter, URL etc.
Network	Data through TCP protocol

Juliet 시험코드는 SW 보안약점 기준 항목과 비교해 봤을 때, 다음 표 11과 같이 일부 항목만 참조가 능하고 대부분 추가 개발이 필요하여 Juliet 시험코드 구조를 기반으로 표 10과 같은 데이터 유입 유형을 고려하여 시험코드를 개발하였다. 43개 보안약점에 대한 JAVA 언어 관련 시험코드는 308개의 기본코드로 구성되어 있으며, 다양한 데이터 및 제어 흐름 등을 기본코드에 반영한 시험코드는 약 9,000개를 개발하였다. 그리고, C/C++ 언어 관련 시험코드는 약 15,000개를 개발하였다.

표 11. SW 보안약점 기준 항목과 관련된 Juliet 코드(예, JAVA)
Table 11. Juliet code related to SW security weakness criteria(ex, JAVA)

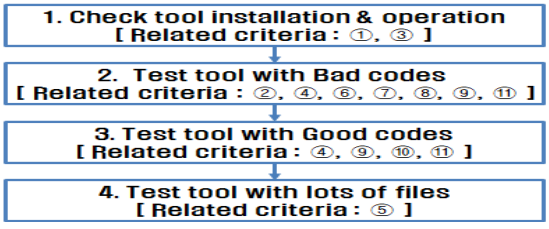
Weakness	Related Juliet Code
Input Validation & Representation (14)	10 (SQL Injection, XSS etc.)
Security Feautres (16)	10 (Use of Broken or Risky Crypto. Algorithm, Use of Hard-coded Password etc.)
Time & State (2)	2 (TOCTOU Race Condition etc.)
Errors (3)	2 (Information Exposure through an error message etc.)
Encapsulation (2)	2 (NULL Pointer Dereference etc.)
Code Quality (5)	2 (Leftover Debug Code etc.)
API Abuse (1)	-

4.3. 적용결과

본 논문에서 제안하는 평가기준을 기반으로 데이터 및 흐름제어, 의미분석 등 정적분석 도구의 기본기능을 갖추고 JAVA 언어기반의 SW 소스코드를 분석할 수 있는 진단기능 및 규칙을 보유하고 있는 국내외 상용 진단도구 7종에 적용해 보았다. 그러나, 적용한 결과가 각 도구업체에 영향을 미칠 수 있기 때문에 본 논문에서는 보안약점 진단결과를 중심으로 간략한 결과만 설명하고, 세부 적용결과는 각 도구업체에 개별적으로 제공하였다. 세부 적용결과에 따라 각 도구업체는 진단규칙 등을 보완하고 있으며, 보완된 진단도구를 대상으로 재검검을 진행할 예정이다.

진단도구는 4.2절에서 설명한 시험코드를 기반으로 다음과 같은 절차(그림1)로 평가를 진행하였다.

그림 1. 평가절차
Fig. 1. Evaluation Process



먼저, 진단도구를 평가하기 위해 진단대상 도구를 설치하고 정상적으로 설치 되었는지 기본기능을 확인한다. 시범적용시, 설정 등의 문제로 진단도구간 충돌이 발생되어 기능 동작이 안되는 경우가 발견되었다. 따라서, 진단도구를 단일 PC(서버) 또는 가상환경(예, VMware 등)에 단독으로 운영환경을 구성하는 것이 필요하다.

진단도구 설치가 완료되면, 제4.2절에서 설명한 시험코드를 기반으로 진단도구를 평가한다. Bad 코드를 대상으로 진단도구가 진단한 결과를 기반으로 진단도구 평가항목(제4.1절)의 대부분을 평가한다. Good 코드 및 대용량 파일은 진단도구 오탐 관련 항목과 대용량 파일 진단능력 항목 평가를 위해 사용한다. Bad 코드 및 Good 코드를 이용한 진단결과 판정시 다음과 같은 기준(표 12)으로 진단도구의 진단결과를 판정하였다.

표 12. 시험코드 기반 진단결과 판정기준
Table 12. The verdict criteria for test results based on Test code

Verdic	Code Type	Criteria
True Positive	Bad	When to report a function etc. correctly where a weakness is present
False Positive	Good	When to report a function etc. where no weakness is present
	Bad	When to report code line except weakness related a rule
False Negative	Bad	When not to report a function etc. where a weakness is present

시험적용 결과, 43개 보안약점 중 평균 20개 정도의 보안약점을 진단 가능 하였으며, 모든 보안약점을 진단하는 도구는 없었다. 그리고, 일부 진단도구는 다음과 같은 보안약점이 소스코드 구현시 사용되는 함수(메소드) 등의 유사성 때문에 해당 보안약점을 구분하지 못하는 경우가 있었다. 이는 해당 보안약점과 관련된 진단규칙명은 서로 상이하지만 관련 진단규칙이 동일하기 때문에 발생하였다.

따라서, 43개 보안약점에 대한 상세한 분석을 통한 진단규칙 도출 및 정교화가 필요하며, 개발자에 의해 신규로 정의되는 함수(메소드) 등이 사용자 정의 진단규칙으로 반영될 수 있도록 진단도구의 개선이 필요하다. 그리고, 진단도구 평가의 신뢰성 및 객관성 보장을 위해 공개용 오픈소스 등을 분석하여 다양한 보안약점 구현 유형을 도출하고, 이를 시험코드에 반영하는 것이 필요하다.

V. 결 론

본 논문에서는 SW 보안약점 진단도구의 최소 기능 요구사항과 진단도구 신뢰성을 보장할 수 있는 평가기준 및 방법을 제시하였다. 그리고, 제안한 평가기준 및 방법을 기반으로 상용진단도구에 시범 적용하여 해당 진단도구가 개선될 수 있도록 유도하였다.

본 논문에서 제시하는 평가기준 및 방법을 진단도구 개발업체에서 자체적으로 적용하면 보다 신뢰성 있는 진단도구를 개발할 수 있으며, 진단도구를 도입하고자 하는 기관에서 평가기준 및 방법을 기관의 요구사항에 맞게 수용하여 적용하면 보다 신뢰성 있는 진단도구를 도입하는데 도움이 될 수 있을 것으로 기대한다. 또한, CC 평가시, 본 논문에서 제시하는 평가기준 및 방법을 적용할 수 있다. 만약, 자체적으로 시험코드를 개발하여 본 논문에서 제안한 평가방법을 적용하고자 하는 경우, 제3.1절 및 4.2절 내용을 기반으로 각 진단항목별 최소 2개 이상의 유형을 포함하여 진단필수 항목과 관련된 시험코드를 모두 개발하는 것이 필요하다.

향후, 본 논문에서 제안한 평가기준 및 방법을 기반으로 시범 적용한 결과를 분석하여 SW 보안약점 기준항목에 대한 보다 효과적인 진단규칙을 제안할 예정이다.

참 고 문 헌

[1] MOPAS, "A guide to secure software development," *Publication No.11-1311000-00030-10*, Retrieved May 2012, from http://www.mopas.go.kr

[2] P. E. Black, M. Kass, M. Koo, and E. Fong, "Source code security analysis tool functional specification version 1.1," *NIST Special Publication 500-268*, Feb. 2011.

[3] MOPAS, "Guidelines on building and operating Information Systems," *MOPAS Notification No.2012-25*, June 2012

[4] G. Tassey, "The economic impacts of inadequate infrastructure for software testing," *NIST*, May 2002.

[5] Microsoft, Inc., *SDL helps build more secure software*, retrieved Apr., 12, 2013, from http://www.microsoft.com/security/sdl/learn/measurable.aspx

[6] B. Chess and C. McGraw, "Static analysis for

security," *IEEE Security & Privacy,* vol. 2, no. 6, pp. 76~79, Nov.-Dec. 2004

[7] M. Johns and M. Jodeit, "Scanstud: a methodology for systematic, fine-grained evaluation of static analysis tools," in *Proc. IEEE 4th ICSTW*, pp. 523~530, Berlin, Germany, Mar. 2011

[8] T. Hofer, "Evaluation static source code analysis tools," M.S. Thesis, School Compt. Commun. Sci., École Polytechnique Fédérale de Lausanne, Mar. 2010

[9] R. K. McLean, "Comparing static security analysis tools using open source software," *IEEE 6th Int. Conf. SW Security Reliability Companion (SERE-C)*, pp. 68~74, Gaithersburg, U.S.A., June 2012.

[10] NIST, "Source code security analysis tool test plan Version 1.1," *NIST Special Publication 500-270*, July 2011

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

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

[13] OWASP, *OWASP Top Ten 2013 rc1,* Retrieved Feb. 2013, from http://www.owasp.org.

[14] J. Bang, R. Ha, J. Park, and P. Kang, "Minimum standard of weakness in development of reliable e-GOV software," in *Proc. KICS Int. Conf. Commun. (KICS ICC 2012)*, vol. 48, pp. 127-128, Jeju Island, Korea, June 2012

[15] NIST, Juliet Test Suite, Retrived Apr., 13, 2013, from http://samate.nist.gov/SRD/testsuite.php

방 지 호 (Jiho Bang)

1996년 2월 홍익대학교 컴퓨터공학과 졸업
2001년 8월 홍익대학교 컴퓨터공학과 석사
2004년 3월~현재 홍익대학교 컴퓨터공학과 박사과정
2001년 7월~2009년 7월 한국정보보호진흥원 선임연구원
2009년 7월~현재 한국인터넷진흥원 책임연구원
<관심분야> 모바일/SW보안, 모바일컴퓨팅, 실시간 시스템 등

하 란 (Rhan Ha)

1987년 2월 서울대학교 컴퓨터공학과 졸업
1989년 2월 서울대학교 컴퓨터공학과 석사
1995년 4월 University of Illinois at Uubana-Champaign 전산학 박사
1989년 3월~1990년 7월 한국통신 전임연구원
1995년 9월~현재 홍익대학교 컴퓨터공학과 교수
<관심분야> 모바일컴퓨팅, 실시간시스템, SW보안 등