creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

[UCI]I804:24011-200000265057
조선대학교
CHOSUN UNIVERSITY

2015年 8月
碩士學位論文

시큐어 코딩을 적용한 웹 자산관리 시스템

朝鮮大學校 産業技術融合大學院
소프트웨어融合工學科
金 東 雲

조선대학교
CHOSUN UNIVERSITY

시큐어 코딩을 적용한 웹 자산관리 시스템

Web Management System using the Secure Coding

2015年 8月 25日

朝鮮大學校 産業技術融合大學院

소프트웨어融合工學科

金 東 雲

조선대학교
CHOSUN UNIVERSITY

시큐어 코딩을 적용한 웹 자산관리 시스템

指導敎授 潘 聲 範

이 論文을 工學 碩士學位卒業 論文으로 提出함

2015年 4月

朝鮮大學校 産業技術融合大學院

소프트웨어融合工學科

金 東 雲

조선대학교
CHOSUN UNIVERSITY

金東雲의 碩士學位 論文을 認准함

위원장 조선대학교 교 수 朴 鍾 安 印

위 원 조선대학교 교 수 潘 聲 範 印

위 원 조선대학교 조교수 申 朱 玄 印

2015年 5月

朝鮮大學校 産業技術融合大學院

조선대학교
CHOSUN UNIVERSITY

목 차

표목차 iii

도목차 iv

ABSTRACT vi

제 1 장 서 론 1

제 1 절 연구 배경 1

제 2 절 연구 목적 및 방법 2

제 2 장 관련연구 4

제 1 절 시큐어 코딩 4

1. 시큐어 코딩 개념 4

2. 시큐어 코딩 보안 약점 6

제 2 절 보안 취약점 분석 방법 및 분석 도구 13

1. 보안 취약점 분석 방법 13

2. 분석 도구 14

제 3 장 시큐어 코딩을 적용한 웹 자산관리 시스템 17

제 1 절 기존 자산관리 시스템 17

제 2 절 시큐어 코딩 적용 19

1. 보안 약점 검출 결과 19

조선대학교
CHOSUN UNIVERSITY

2. 보안 약점 제거 22

제 4 장 연구 결과 37

제 5장 결 론 39

조선대학교
CHOSUN UNIVERSITY

표 목 차

표 1. 소프트웨어 개발 단계별 결함 수정비용 분석 2
표 2. OWASP Top 10 2013 7
표 3. CWE/SANS Top 25 2011 8
표 4. 소프트웨어 개발보안 적용 대상 및 범위 11
표 5. 행정자치부 보안 약점 목록 12
표 6. 소스코드 분석 방법 13
표 7. 소스코드 분석 도구 14
표 8. 스패로우 언어별 소스코드 분석 방법 15
표 9. 사용자 별 주요 제공 기능 16
표 10. 검출된 보안 약점 레벨 별 수 21
표 11. 검출된 보안 약점 목록 21
표 12. 위험 문자 인코딩 25

조선대학교
CHOSUN UNIVERSITY

도 목 차

그림 1. 보안침해사고 발생 빈도 1
그림 2. MS-SDL 적용 소프트웨어와 미적용 소프트웨어 4
그림 3. 소프트웨어 개발 순서 5
그림 4. CWSS 평가척도 그룹과 그룹별 하위 요소들 10
그림 5. 비주얼 스튜디오 기반으로 제작된 자산관리 시스템 17
그림 6. JSP 기반으로 제작된 웹 자산관리 시스템 18
그림 7. 검출된 보안 약점 수 19
그림 8. 스패로우 보안 약점 검출 20
그림 9. 검출된 보안 약점 크로스 사이트 스크립팅 23
그림 10. 시큐어 코딩 규칙 적용 후 24
그림 11. 검출된 보안 약점 크로스 사이트 요청 위조 26
그림 12. 시큐어 코딩 규칙 적용 후 27
그림 13. 검출된 보안 약점 하드코드 된 패스워드 28
그림 14. 검출된 보안 약점 경쟁조건: 검사시점과 사용시점 29
그림 15. 시큐어 코딩 규칙 적용 후 30
그림 16. 검출된 보안 약점 오류메시지를 통한 정보노출, 시스템 데이터 정보 노출 31
그림 17. 시큐어 코딩 규칙 적용 후 32
그림 18. 검출된 보안 약점 주석문 안에 포함된 패스워드 등 시스템 주요 정보 32
그림 19. 검출된 보안 약점 AUTOCOMPLETE_PASSWORD 33
그림 20. 시큐어 코딩 규칙 적용 후 34
그림 21. 검출된 문제점 35
그림 22. 문제점 해결 35
그림 23. 검출된 문제점 36
그림 24. 문제점 해결 36

조선대학교
CHOSUN UNIVERSITY

그림 25. 보안 약점 제거 결과 37
그림 26. 보안 약점 제거 결과 그래프 38

조선대학교
CHOSUN UNIVERSITY

ABSTRACT

Web Asset Management System using the Secure Coding

Kim, Dong-Un
Advisor : Prof. Pan, Sung Bum Ph.D.
Department of Software Convergence Engineering,
Graduate School of Industry Technology
Technology Convergence, Chosun University

Security breaches have recently become a social issue. Although network firewalls and user authentication programs are used to fight the problem resulting from it, most of the breaches occur not from networks or web servers but from application programs that have security vulnerability. Many studies are performed to address such breaches and secure coding is emerging as the most effective measure.

Secure coding, a safe software production technique, removes security weakness of a software source code to prevent security breaches. By doing so, it allows use of safe software that has strengthened security.

In 2002, the US enacted Federal Information Security Management Act (FISMA), making secure coding compulsory. Korea also began to develop the secure coding technique in 2009 and its Ministry of Government Administration and Home Affairs(MOGAHA) announced revision of the guideline on establishment and operation of information system in June 2012

조선대학교
CHOSUN UNIVERSITY

to make secure coding compulsory.

Most companies use asset management system to manage their assets. The asset management system, which is customized for clients' requirements and usage, provides various functions. The asset management system suggested by the paper supports all functions about internal asset management and related works. Any security breach of such asset management system could lead to loss or leak of employee information, work data and other important data. Therefore, the security should be improved.

This thesis analyzed source codes using Sparrow to eliminate security weakness of the asset management system and found 162 security weaknesses. Among them, seven were out of 47 items listed in the software security weakness diagnosis guideline provided by the MOGAHA and three were found from the security weakness items offered by CWE, CERT, OWASP and Sparrow. This thesis eliminated the security weakness by referring to the security weakness elimination techniques that the MOGAHA and Sparrow provided as way to remove the identified security weaknesses and confirmed that all security weaknesses were removed.

조선대학교
CHOSUN UNIVERSITY

제 1 장 서 론

제 1 절 연구 배경

인터넷의 발달로 인하여 기업의 업무 효율을 향상시키기 위한 방법으로 자산관리 시스템, 인사관리 시스템 등 다양한 소프트웨어가 개발되어 출시되고 있다. 이렇게 출시된 소프트웨어는 일반적으로 보안에 취약하다는 문제점을 가지고 있다. 이러한 문제점을 해소하기 위하여 네트워크 방화벽, 사용자 인증 시스템 등 다양한 보안 프로그램이 출시되어 사용되고 있지만, 여전히 보안침해의 문제점은 사회적으로 이슈가 되고 있다.

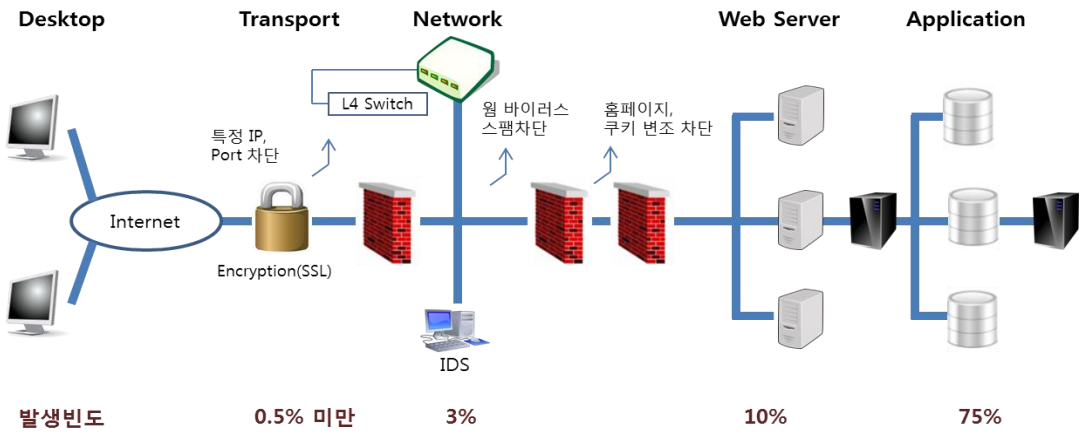

그림 1. 보안침해사고 발생 빈도

가트너(Gartner)의 보고서에 의하면 그림 1과 같이 보안침해사고의 대부분이 네트워크와 웹 서버가 아닌 취약점을 내포하고 있는 응용프로그램에서 발생되었다[1]. 또한, 개발 완료 후, 취약점을 보안하기 위한 비용이 매우 크기 때문에 개발 단계에서부터 프로그램의 안정성을 고려해야 한다.

조선대학교 CHOSUN UNIVERSITY

표 1. 소프트웨어 개발 단계별 결함 수정비용 분석

구분	설계단계	코딩단계	통합단계	베타제품	제품출시
설계과정	1배	5배	10배	15배	30배
코딩과정	-	1배	10배	20배	30배
통합과정	-	-	1배	10배	20배

표 1은 소프트웨어 개발 단계별 오류·취약점 등의 결함을 제거하는 시점에 따른 수정 비용을 분석한 결과이다[2]. 설계 과정에서 발생한 결함을 수정하게 될 경우 1배의 비용이 발생하게 되지만 제품 출시 이후에 결함을 발견하여 조치하게 될 경우 결함 수정 비용은 최대 30배의 비용이 발생하게 된다.

중요 정보를 보호하기 위하여 보안 프로그램을 이용하여 보안성을 높이는 것도 중요하지만, 프로그래머가 개발 단계에서 보안 약점을 사전에 제거하여 보안성을 높이는 것이 가장 효과적인 방법이라 할 수 있다[3].

소프트웨어의 보안 약점을 제거하여 안전한 소프트웨어를 개발하기 위한 다양한 연구가 진행되었으며, 보안침해사고를 예방하고 안전한 프로그램을 개발하고자 시큐어 코딩(Secure Coding)의 중요성이 강조되고 있다.

제2절 연구 목적 및 방법

소프트웨어는 크게 두 가지로 분류 된다.

첫 번째, 네트워크 연결이 필요하지 않은 소프트웨어는 정확성, 효율성을 요구한다.

두 번째, 인터넷 연결이 필요한 소프트웨어는 정확성, 효율성, 추가로 안정성을 요구하게 된다. 인터넷 환경에서 프로그램 및 데이터를 네트워크를 통해 교환함으로 악의적인 침입자에게 악의적인 공격을 받아 데이터가 손실될 수 있기 때문에 안정성을 요구한다[3].

조선대학교
CHOSUN UNIVERSITY

이처럼 악의적인 공격을 받을 가능성이 있는 소프트웨어의 보안 약점을 사전에 제거하기 위하여 시큐어 코딩 기법이 나왔으며, 국내의 경우 2012년 12월부터 정부사업을 대상으로 적용하여 점차 강화하는 등 소프트웨어의 보안 강화의 중요성이 부각 되고 있다.

본 논문에서는 현재 기업에서 사용하는 비주얼 스튜디오 기반으로 제작된 자산관리 시스템을 기업과 협력하여 JSP 기반으로 제작하였으며, 시맨틱 기반 정적 분석도구인 스패로우를 활용하여 소스코드를 분석하고, 스패로우에서 제공하는 보안 약점 제거 기법과 행정자치부의 소프트웨어 보안약점 진단 가이드에 따라 해결방안을 제시한 후 소스코드의 보안 약점을 제거한다. 웹 자산관리 시스템의 보안 약점 제거 전과 후의 결과를 비교하여 보안이 강화된 안전한 자산관리 시스템을 제시한다.

조선대학교
CHOSUN UNIVERSITY

제 2 장 관련연구

이 장에서는 시큐어 코딩의 개념에 대하여 설명하고, 대표적인 국내·외의 보안 약점에 대하여 살펴본다. 또한, 분석 방법 및 분석 도구에 대하여 소개한다.

제 1 절 시큐어 코딩

1. 시큐어 코딩 개념

미국은 국내보다 10년 앞서 2002년 연방정보보호관리법(FISMA)을 제정해 시큐어 코딩을 의무화 하였다[4]. 국내의 경우 2012년 6월 행정자치부의 정보시스템 구축·운영 지침 개정안 행정예고를 통하여 시큐어 코딩 의무화 법안을 발표하였다.

시큐어 코딩을 적용한 소프트웨어 개발보안의 대표적인 사례는 마이크로소프트의 개발 생명주기(MS-SDL)가 있다. 자체적으로 개발한 개발 생명주기를 제품 개발 전 과정에 적용하여 보안 약점을 제거하였다.

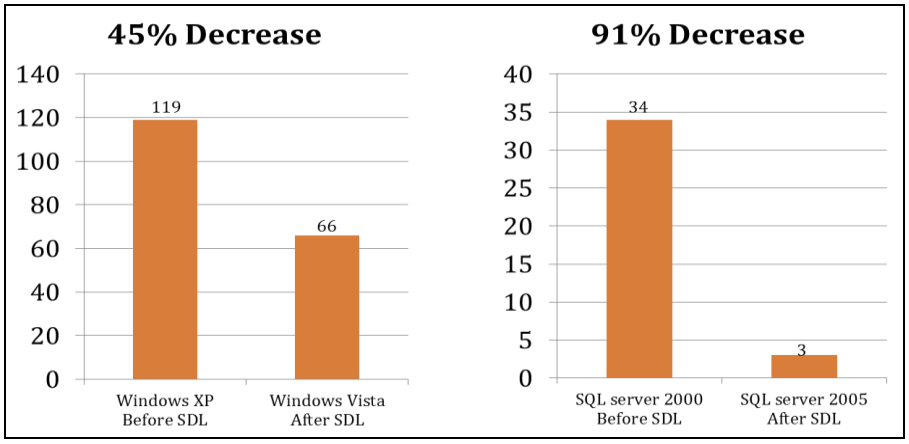

그림 2. MS-SDL 적용 소프트웨어와 미적용 소프트웨어

조선대학교 CHOSUN UNIVERSITY

그림 2는 MS-SDL 적용 전의 운영체제 및 DB 서버와 적용 후의 운영체제 및 DB 서버의 보안취약점을 분석한 것이다[5]. Windows XP와 Windows Vista를 출시 이후 1년간의 보안취약점 발생현황을 분석한 결과 보안취약점이 45% 감소하였으며, SQL server 2000과 SQL server 2005 출시 이후 3년간의 보안취약점 발생현황을 분석한 결과 91%의 보안취약점이 감소한 것을 확인할 수 있다. 보안취약점 감소로 얻을 수 있는 가장 큰 효과로는 제품 출시 후 보안패치 횟수가 감소하여 보안 패치 비용이 절감되는 효과를 가져왔다.

시큐어 코딩이란 소프트웨어 소스코드 내부에 존재할 수 있는 보안 약점을 제거하고, 보안을 고려하여 소프트웨어 개발 전 과정에서 지켜야 할 보안활동을 말한다. 물론 보안 약점이란 시큐어 코딩 기법을 적용하는 과정에서 새로운 보안 약점이 발견될 수 있어 모든 보안 약점을 제거한다는 것은 사실 불가능하다. 그렇지만 최대한 보안 약점을 제거하여 안전한 소프트웨어를 개발하는 것을 목적으로 한다[6].

그림 3. 소프트웨어 개발 순서

일반적인 소프트웨어 개발방법은 그림 3과 같이 요구사항 도출 → 설계 → 구현(개발) → 테스트 → 릴리즈 순으로 이루어진다. 시큐어 코딩은 소프트웨어 개발단계(Software Development Life Cycle : SDLC)에서 보안 약점을 제거하여 해킹의 위험성을 줄이는 방어적 프로그래밍 기법이다[7]. 중요한 점은 구현 단계에서 이 기법을 적용하게 된다. 프로그래머는 프로그램을 개발하는데 기존의 개발 방식을 버리고 새로운 개발 방식을 익히기 위해서는 상당한 시간과 노력이

조선대학교
CHOSUN UNIVERSITY

필요하다. 하지만 안전한 소프트웨어를 개발하기 위해서는 이처럼 많은 노력이 필요하다.

2. 시큐어 코딩 보안 약점

이 장에서는 국내·외 시큐어 코딩에서 다루고 있는 주요 보안 약점 목록에 대하여 설명한다.

가. 주요 보안 약점 목록

보안 약점 목록의 대표적인 사례로 CWE(Common Weakness Enumeration)를 들 수 있다. CWE의 경우 보안취약점으로 이어질 수 있는 800 개 이상의 프로그래밍 오류, 설계오류 및 아키텍처 오류에 대하여 제공하고 있다[8]. 또한, 개발 중이거나 개발된 프로그램 소스코드의 보안 약점을 제거하는 과정 중에도 새로운 보안 약점이 발생할 가능성이 높다. 이러한 문제점을 해결하기 위하여 연계 가능성이 크고 침해 시 피해가 심각한 보안 약점에 대하여 먼저 대책을 마련하기 위하여, 대표적인 주요 보안 약점목록 (Open Web Application Security Project) OWASP Top 10[9]과 CWE/SANS Top25[10]가 있다.

조선대학교
CHOSUN UNIVERSITY

표 2. OWASP Top 10 2013

구분	종류	발생원인
A1	인젝션	SQL, 운영체제, LDAP 인젝션 취약점은 신뢰할 수 없는 데이터가 명령이나 질의문의 일부로서 인터프리터로 보내질 때 발생한다.
A2	인증 및 세션 관리 취약점	세션으로 활용하는 쿠키의 일련의 값을 암호화 하지 않거나 로그인, 로그아웃해도 세션 값이 바뀌지 않고 고정될 때, 세션 ID가 URL에 노출되어 발생한다.
A3	크로스사이트 스크립팅(XSS)	XSS 취약점은 애플리케이션이 신뢰할 수 없는 데이터를 가져와 적절한 검증이나 제한 없이 웹 브라우저로 보낼 때 발생한다.
A4	취약한 직접 객체 참조	직접 객체 참조는 개발자가 파일, 디렉토리, 데이터베이스 키와 같은 내부 구현 객체를 참조하는 것을 노출시킬 때 발생한다.
A5	보안 설정 오류	배포과정이나 릴리즈 단계에서 관리가 허술할 때 발생한다.
A6	민감한 데이터 노출	데이터 암호화 부분과 관련 있는 것이다. 패스워드를 해시화 하더라도 salt없이 암호화하거나 로그인이나 계좌이체 같은 중요한 정보가 오고 가는 상황에 데이터를 암호화하지 않을 때 발생하는 취약점이다.
A7	기능 수준의 접근 통제 누락	인가되지 않은 사용자가 특정 인증이 필요한 기능에 정상적인 인증절차 없이 접근이 가능하도록 되는 취약점
A8	크로스 사이트 요청 변조(SCRF)	CSRF 공격은 로그인된 피해자의 취약한 웹 애플리케이션에 피해자의 세션 쿠키와 기타 다른 인증 정보를 자동으로 포함하여 위조된 HTTP 요청을 강제로 보내도록 하는 것이다.
A9	알려진 취약점이 있는 컴포넌트 사용	컴포넌트, 라이브러리, 프레임워크 같은 애플리케이션을 구성하는 요소 중 취약점이 있는 컴포넌트를 사용할 때 발생하는 취약점이다.
A10	검증되지 않은 리다이렉트 및 포워드	적절한 검증 절차가 없으면 공격자는 피해자를 피싱 또는 악성 코드 사이트로 리다이렉트 하거나 승인되지 않은 페이지에 접근하도록 전달할 수 있다.

조선대학교 CHOSUN UNIVERSITY

표 2 OWASP Top 10은 애플리케이션과 관련한 보안 관련 연구를 수행하고 위험 요소에 대한 목록을 제공하는 프로젝트이다. 2003년 처음 10개 항목으로 이루어진 목록을 발표하였으며 소규모 업데이트로 2004년 판과 2007년 판을 만들었고, 2010년 판과 2013년 판은 위험뿐만 아니라 발생 빈도에 우선순위를 두고 개정하였다[11]. 또한, OWASP는 이러한 위험에 대하여 예방할 수 있는 방법론으로 OWASP's, ESAPI(OWASP Enterprise Security API), OWASP XSS Prevention Cheat sheet 등을 위험별로 제시하고 있다[12].

표 3. CWE/SANS Top 25 2011

순위	점수	ID	발생원인
[1]	93.8	CWE-89	SQL 명령에 사용된 특별한 엘리먼트를 무효화 하지 않는 것(SQL 삽입)
[2]	83.3	CWE-78	OS 명령에 사용된 특별한 엘리먼트를 무효화 하지 않은 것(OS 인젝션)
[3]	79.0	CWE-120	입력의 크기를 확인하지 않고 버퍼를 복사하는 것 (버퍼 오버플로우)
[4]	77.7	CWE-79	웹 페이지 생성 도중 입력 값을 무효화하지 안는 것 (XSS)
[5]	76.9	CWE-306	핵심 기능에 대해서 인증을 누락하는 것
[6]	76.8	CWE-862	인가 기능을 누락 하는 것
[7]	75.0	CWE-798	인증 데이터 코드에 직접 기록하는 것
[8]	75.0	CWE-311	민감한 데이터의 암호화 누락하는 것
[9]	74.0	CWE-434	위험한 유형의 파일 업로드를 제한하지 않는 것
[10]	73.8	CWE-807	신뢰할 수 없는 입력 정보에 의존하여 보안성 결정 하는 것
[11]	73.1	CWE-250	불필요한 권한으로 실행하는 것
[12]	70.1	CWE-352	사이트 간 요청 위조 (CSRF)
[13]	69.3	CWE-22	금지된 디렉토리에 접근을 제한하지 않는 것 (경로 추적)

조선대학교 CHOSUN UNIVERSITY

[14]	68.5	CWE-494	무결성 검사 없이 코드를 다운로드 하는 것
[15]	67.8	CWE-863	올바르지 않은 인가
[16]	66.0	CWE-829	신뢰할 수 없는 제어영역에서 온 기능을 포함하는 것
[17]	65.5	CWE-732	중요한 장원에 잘못된 권한을 할당하는 것
[18]	64.6	CWE-676	잠재적으로 위험한 함수를 사용하는 것
[19]	64.1	CWE-327	해동되었거나 위험한 암호화 알고리즘을 사용하는 것
[20]	62.4	CWE-131	버프 크기 잘못 계산하는 것
[21]	61.5	CWE-307	과도한 인증 시도를 제안하지 않는 것
[22]	61.1	CWE-601	신뢰할 수 없는 사이트로 URL 접속지 변경하는 것
[23]	61.0	CWE-134	형식 문자열 (Format String)을 통제하지 않는 것
[24]	60.3	CWE-190	정수 오버플로우 또는 랩 어라운드
[25]	59.9	CWE-759	솔트 없이 일방향 해쉬를 사용하는 것

표 3 CWE/SANS Top 25는 소프트웨어의 취약성을 초래할 수 있는 치명적인 오류를 정리한 목록이다. 2009년 25개의 주요 보안 약점을 선발하여 발표하였으며, 가장 최근에는 2011년 버전이 발표되었다. 또한, 2011년에는 보안 약점 선별을 위하여 중요도 정량 평가 방법인 CWSS(Common Weakness Scoring System)[13]가 사용되었으며, 2010년에 사용되었던 확산도 및 중요도 요인 이외에 악용 가능성도 평가하였다.

조선대학교
CHOSUN UNIVERSITY

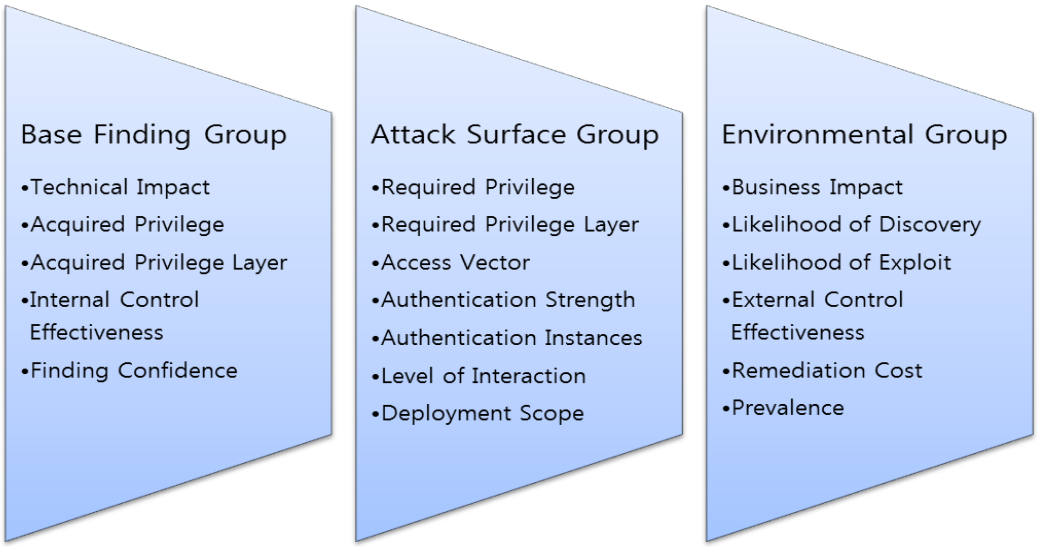

그림 4. CWSS 평가척도 그룹과 그룹별 하위 요소들

그림 4는 CWSS 평가척도 그룹과 하위 요소를 나타내고 있으며, 약점 자체의 심각성(Base Finding Metric Group), 공격 측면의 심각성(Attack Surface Matric Group), 환경적 측면의 심각성(Environment Matric Group) 3개 그룹으로 평가척도를 분류하였으며 보안 약점의 심각성을 평가하기 위한 기준으로 18가지의 평가척도를 포함하고 있으며, 각 평가척도별 1점 만점의 점수를 기반을 둔 100점 만점의 중요도 점수 산술식을 제공하고 있다[14].

나. 행정자치부 소프트웨어 개발 보안

시큐어 코딩은 국내에서 최초 사용하거나 만든 단어가 아니다. 미국은 이미 지난 2002년 연방정보보안관리법(FISMA: Federal Information Security Management ACT) 제정을 통해 시큐어 코딩을 의무화하였다. 국내의 경우 2009년 처음 시큐어 코딩 기법 개발을 시작하였으며, 2012년 6월 행정자치부에서 정보시스템 구축·운영 지침 개정안을 발표하여 시큐어 코딩 의무화를 시행하였다[15].

조선대학교 CHOSUN UNIVERSITY

표 4. 소프트웨어 개발보안 적용 대상 및 범위

구분	내용	비고
대상	▶정보시스템 감리대상 정보화 사업 → 2012년 12월 행정기관 등에서 추진하는 개발비 40억 원 이상 정보화 사업에 소프트웨어 개발 보안 적용을 의무화 → 2014년 1월 개발비 20억 원 이상의 정보화 사업에 적용 → 2015년 감리대상 전 정보화 사업에 소프트웨어 개발 보안 적용	단계적 확대
범위	▶소스코드(신규개발 전체, 유지보수로 변견된 부분)	상용 소프트웨어 제외
기준	▶소프트웨어 보안 약점 기준(SQL 삽입 등 47개 항목)	소프트웨어 보안 약점 진단
	▶공통평가기준(Comon Criteria, CC)	진단도구 평가 · 인증
	▶진단원 자격기준	진단원 자격 부여

표 4는 행정자치부의 소프트웨어 개발보안 적용 대상 및 범위이다. 2012년 12월 정보시스템 감리대상 정보화 사업 40억 원 이상의 대상으로 시행하였으며, 2014년 1월 개발비 20억 원 이상의 정보화 사업에 적용, 2015년 감리대상 전 정보화 사업으로 의무화 대상을 확대하였다. 또한, 시큐어 코딩 의무화를 통하여 다양한 분석 도구들이 출시되고 있다. 하지만 출시되는 모든 분석 도구를 사용할 수 있는 것이 아닌 공통평가기준 인증을 획득한 제품만을 사용해야 한다.

조선대학교
CHOSUN UNIVERSITY

표 5. 행정자치부 보안 약점 목록

유형	발생원인	개수
입력 데이터 검증 및 표현	프로그램 입력 값에 대한 부적절한 검증 등으로 인해 발생할 수 있는 보안 약점 예) SQL 삽입, 자원삽입, 크로스 사이트 스크립팅 등	15
보안기능	인증, 접근제어, 권한 관리 등을 적절하지 않게 구현 시 발생할 수 있는 보안 약점 예) 적절한 인증 없는 중요 기능 허용, 부적절한 인가, 중요 자원에 대한 잘못된 권한설정 등	16
시간 및 상태	멀티프로세스 동작환경에서 부적절한 시간 및 상태 관리로 발생할 수 있는 보안 약점 예) 경쟁조건: 검사시점과 사용시점(TOCTOU), 제어문을 사용하지 않는 재귀함수	2
에러 처리	불충분한 에러 처리로 중요정보가 에러정보에 포함되어 발생할 수 있는 보안 약점 예) 오류메시지 통한 정보노출, 오류상황 대응 부재, 적절하지 않은 예외 처리	3
코드 오류	개발자가 범할 수 있는 코딩 오류로 인해 유발되는 보안 약점 예) 널 포인터 역 참조, 부적절한 자원 해제, 해제된 자원 사용, 초기화되지 않은 변수 사용	4
캡슐화	불충분한 캡슐화로 인가되지 않은 사용자에게 데이터가 노출될 수 있는 보안 약점 예) 잘못된 세션에 의한 데이터 정보노출, 제거되지 않고 남은 디버그 코드, 시스템 데이터 정보노출 등	5
API 오용	부적절하거나, 보안에 취약한 API사용으로 발생할 수 있는 보안 약점 예) DNS lookup에 의존한 보안결정, 취약한 API 사용	2

조선대학교 CHOSUN UNIVERSITY

표 5 행정자치부의 보안 약점 목록은 7개 유형 47개의 보안 약점 항목으로 구성되어 있으며, 보안 약점 목록 47개 항목은 CWE라는 데이터베이스에서 47개 항목만을 추출하여 사용하고 있으며, OWAPS Top 10의 항목 중 A9을 제외한 모든 항목을 포함하고 있다[16].

제 2 절 보안취약점 분석 방법 및 분석 도구

1. 보안취약점 분석 방법

소스코드 분석 방법은 정적 분석(Static Code Analysis) 방법과 동적 분석(Dynamic Code Analysis) 방법으로 분류된다. 본 논문에서는 소스코드를 분석하기 위한 도구로 정적 화이트 박스 분석 도구 스패로우를 사용한다.

표 6. 소스코드 분석 방법

	정적 분석	동적 분석
블랙 박스	소스코드를 참조하거나 직접 실행하지 않고 분석하는 방법	소스코드 없이 실행만으로 분석하는 방법 실행 결과가 예상결과와 일치하는지를 비교한다
화이트 박스	소스코드 만으로 분석하는 방법 정적 분석 도구를 이용한다	소스코드를 실행함과 동시에 소스코드를 참조하여 분석하는 방법 디버깅 및 단위 시험등이 화이트 박스 분석에 해당한다.

표 6은 소스코드 분석 방법을 나타낸 표이다. 첫 번째 정적 프로그램 분석은 프로그램을 실행하지 않고 분석하는 방법으로 자동화된 도구를 이용해서 분석하는 방법을 말한다[17]. 두 번째 동적 프로그램 분석은 프로그램을 실행해 보고 분석하는 방법으로 결과를 바탕으로 판단을 내리는 방법이다.

조선대학교 CHOSUN UNIVERSITY

2. 분석 도구

소스코드 보안 약점 제거를 위하여 연구기관과 기업 등에서 많은 연구가 이루어졌다. 표 7은 소스코드 분석 도구이다. 무료 버전, 유료 버전 등 다양한 분석 도구들이 존재하지만 본 논문에서 사용하는 소스코드 분석 도구 스패로우는 프로그램의 실행 의미 분석을 통해 버퍼 오버런(Buffer Overrun), 메모리 누수(leak) 등의 치명적인 메모리 오류를 검출해 주는 실행 의미 기반 프로그램 오류 자동 분석기이며, 분석된 오류에 대한 분석 실행 시간, 발생 경로, 메모리 상태 등의 정보를 제공한다[18].

표 7. 소스코드 분석 도구

분석 도구	분석 방법
MOPS[19]	· 버클리 대학교에서 개발한 모델 검사기 · 보안 취약요소를 프로퍼티로 정의, 유한오토마타를 이용한 정형화 · 적은 분석 비용으로 모델링된 취약점을 모두 검사 가능 · 자료흐름분석을 하지 않음으로 인한 한계 존재
스패로우[20]	· C/C++, Java 프로그램 분석 도구 · Sematic, Syntactic, Comment 분석 방법 지원 · 신속한 수정을 위한 오류 분석 보고서와 상세 분석가이드 제공
Coverity Prevent[21]	· 소스코드에 대한 정적 분석도구 · 전체 코드에서 발견된 취약점을 목록으로 표시 · 각각의 목록은 취약점이 발생한 소스코드 내의 위치와 취약점이 발생된 원인을 포함함
Findburgs[22]	· 매릴랜드 대학에서 개발한 정적 분석 도구 · JRE에서 동작되며, 이클립스 플러그인으로 개발환경에서 사용 가능하고 Java

조선대학교 CHOSUN UNIVERSITY

	프로그램 분석 도구
PMD[23]	· 공개 프로젝트로 진행된 보안약점 분석 도구 · Java 프로그램 분석 도구 · 개선된 버전에서 jps, xsl, ecmascript, xml 분석 가능
Fortify 360[24]	· Fortify에서 개발한 취약점 탐지 도구 · C/C++, Java 등 12개 언어지원 · 정적&동적 분석 기법 모두 사용한 취약점 분석 도구

표 8. 스패로우 언어별 소스코드 분석 방법

항목	C/C++	Java
분석파일	컴팡리러 전 처리 파일 및 소스코드	소스코드 및 클래스
분석 파일 수집 시점	빌드 시점	빌드 후
분석방법	Syntactic Semantic	Syntactic Semantic
체커 수	약 300개	약 500개

표 8은 스패로우에서 지원하는 언어별 소스코드 분석 방법이다. 스패로우는 C/C++, Java로 개발된 소스코드에 대해 정적 분석을 지원하며, CWE, CERT, OWASP, 안전행정부 지정 보안취약점, 국가정보원 보안 적합성 검증 기준 소프트웨어 취약점 분류 기준이 제시하는 품질 및 보안 이슈 검출 기준을 충족하는 소스코드 보안 약점 검출 도구이다. 스패로우의 분석 방법은 Syntactic 분석 방법과 Semantic 분석 방법으로 나누어진다.

Syntactic 분석은 프로그램의 생김새나 구조만 관찰하는 분석 기술, 간단한 구문 검사만으로 쉬운 패턴의 오류 검출이 가능하다[20]. Semantic 분석은 실제 프로그램이 실행될 때의 실행 의미를 이해하는 분석기술, 구문 검사만으로는

조선대학교 CHOSUN UNIVERSITY

파악하기 어려운 오류까지 검출하고 허위 경보가 적은 정교한 분석이 가능하다[20].

표 9. 사용자 별 주요 제공 기능

사용자 유형	주요기능
소프트웨어 품질 관리자	▶전체 소프트웨어 품질 정책 및 전체 소스코드 품질 관리 ▶소프트웨어의 잠재적인 품질/보안취약점 및 사용자 지정 (특정 코딩 스타일 등) 검출 규칙 설계 ▶설정한 검출 규칙 전파 ▶전체 검출 이슈 관리 ▶전체 소스코드 검출 추이 파악
소프트웨어 개발자	▶개별 소스코드 분석 및 품질 관리 ▶소프트웨어 잠재적인 품질/보안취약점 및 사용자 지정 (특정 코딩 스타일 등) 검출 ▶검출 이슈 처리 ▶이슈 검출 추이 파악

표 9는 사용자별 주요 제공 기능이다. 스패로우 사용자는 크게 두 가지로 분류된다.

첫 번째, 소프트웨어 품질관리자는 스패로우를 이용해 소프트웨어의 품질 정책 및 전체 소스코드의 품질을 관리하고 소프트웨어의 잠재적인 품질/보안취약점 및 사용자 지정 검출 규칙을 설계한다. 또한, 설계한 규칙을 기반으로 소프트웨어 개발자가 해당 규칙을 이용하여 소스코드를 분석하도록 유도한다.

두 번째, 소프트웨어 개발자는 품질 관리자가 설계한 규칙에 따라 자신의 개발 환경에서 소스코드를 분석하고 분석 결과를 확인할 수 있다.

품질관리자는 분석 결과를 바탕으로 전체 소스코드의 품질을 관리할 수 있으며, 개발자는 검출된 이슈를 바탕으로 소스코드의 품질 및 보안 수준을 높일 수 있다.

조선대학교
CHOSUN UNIVERSITY

제 3 장 시큐어 코딩을 적용한 웹 자산관리 시스템

제 1 절 기존 자산관리 시스템

자산관리를 하는 이유는 자산의 구매정보를 기준으로 자산의 위치는 어디이며, 누가 사용하고 있는지 확인할 수 있다는 점이다. 또한, 자산에 대한 유지보수 기간, 장애 발생 횟수 등을 파악하여 자산에 대해 운영을 할 수 있으며, 자산번호 부여 및 자산별 이력 관리를 통해 자산의 구매일, 고장 횟수, 사용자 변동, 위치 변동 등을 검색 관리할 수 있다[25]. 이외에도 기업의 요구 사항과 쓰임에 맞춰 제작되기 때문에 다양한 기능을 지원하는 자산관리 시스템이 출시되어 사용되고 있다. 그림 5는 기업에서 사용하는 비주얼 스튜디오 기반으로 제작된 자산관리 시스템으로 네트워크 연결이 불가능한 자산관리 시스템이다.

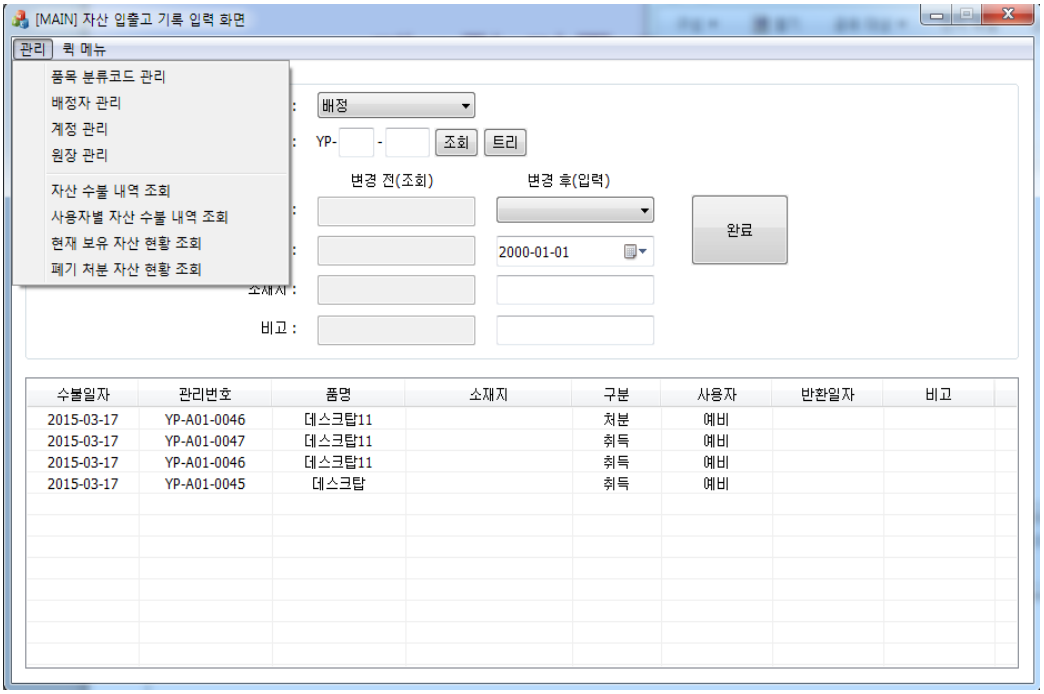

그림 5. 비주얼 스튜디오 기반으로 제작된 자산관리 시스템

조선대학교
CHOSUN UNIVERSITY

비주얼 스튜디오 기반으로 제작한 자산관리 시스템의 경우 정확성과 효율성을 위주로 제작된 자산관리 시스템이다. 네트워크 연결이 불가능하며 하나의 PC에서만 사용할 수 있으며 PC가 고장이 나게 되면 복원이 힘들다는 문제점이 있다. 이러한 문제점을 해결하고 업무의 효율을 높이기 위하여 JSP 기반의 웹 자산관리 시스템을 개발하였다.

그림 6. JSP 기반으로 제작한 웹 자산관리 시스템

그림 6은 JSP 기반으로 제작한 웹 자산관리 시스템은 인터넷 연결이 가능한 소프트웨어이다. 인터넷 연결이 가능한 소프트웨어는 시간적, 위치적, 기기적인 제약을 받지 않아 실시간 업무 처리가 가능해 업무 효율성이 매우 높다. 또한, 데이터를 PC에 저장하는 방식이 아닌 웹 서버에 저장하여 네트워크를 통해 교환함으로 사용하던 PC가 고장 나더라도 데이터가 손실되지 않는다.

조선대학교
CHOSUN UNIVERSITY

인터넷 연결이 가능한 웹 자산관리 시스템은 정확성, 효율성 추가로 안정성을 요구하게 된다. 기업에서는 안정성을 높여 보안침해사고를 예방하는 방법으로 네트워크 방화벽, 사용자 인증 프로그램 등 보안 프로그램을 사용하고 있다. 하지만 보안침해사고 대부분은 네트워크, 웹 서버 등이 아닌 애플리케이션 자체의 보안취약점에서 일어나고 있다. 이처럼 소프트웨어의 보안취약점을 사전에 예방하는 방법으로 소스코드 내부에 존재할 수 있는 잠재적인 보안 약점을 제거하는 시큐어 코딩 기법을 적용하고자 한다.

제 2 절 시큐어 코딩 적용

1. 보안 약점 검출 결과

웹 자산관리 시스템의 보안 약점을 검출하기 위하여 시맨틱 기반의 정적 분석 도구인 스패로우를 이용하여 보안 약점을 분석, 검출하였다. 그림 7과 같이 검출 결과 162개의 보안 약점이 검출되었다.

	언어	신규 이슈	총 이슈	분석자	소스 파일 수	빌드 라인 수	상태
▸	Java / Others	162	162	admin1(Super administrator)	60	16,798	Completed

그림 7. 검출된 보안 약점 수

스패로우는 소프트웨어 개발자가 더욱 손쉽게 보안 약점을 제거할 수 있도록 그림 8과 같이 검출된 보안 약점을 표로 정의하고 있다.

조선대학교
CHOSUN UNIVERSITY

		ID	신규 여부	▲위험도	체커	경로	함수	라인	이슈 상태
☐		2.2.64	신규	Level 1	ScpoeForInVariable	C:\project\ypitgoods\WebContent\dist\jstree.js		1,981	미열람
☐		2.2.65	신규	Level 1	ScpoeForInVariable	C:\project\ypitgoods\WebContent\dist\jstree.js		2,007	진행
☐		2.2.66	신규	Level 1	GlobalVariable	C:\project\ypitgoods\WebContent\js\jquery.form.js		552	미열람
☐		2.2.67	신규	Level 1	GlobalVariable	C:\project\ypitgoods\WebContent\js\rexpert.js		1,730	미열람
☐		2.2.68	신규	Level 1	XSS	C:\project\ypitgoods\WebContent\WEB-INF\jsps\inoutView.jsp			미열람
☐		2.2.69	신규	Level 1	XSS	C:\project\ypitgoods\WebContent\WEB-INF\jsps\modifyGoods.jsp			미열람
☐		2.2.70	신규	Level 2	USING_DYNAMIC_CLAS	C:\project\ypitgoods\src\ypit\dao\GoodsDAO.java			미열람
☐		2.2.71	신규	Level 2	IMPROPER_CHECK_FOR	C:\project\ypitgoods\src\ypit\dao\GoodsDAO.java			미열람
☐		2.2.73	신규	Level 2	HARD_CODE_USER_NAM	C:\project\ypitgoods\src\ypit\dao\GoodsDAO.java			미열람
☐		2.2.74	신규	Level 2	AUTOCOMPLETE_PASS	C:\project\ypitgoods\WebContent\WEB-INF\jsps\login.jsp			미열람
☐		2.2.75	신규	Level 3	PASSWORD_IN_COMME	C:\Project\ypitgoods\WebContent\js\jquery.form.js			미열람
☐		2.2.76	신규	Level 4	STRING_CONVERSION_W	C:\Project\ypitgoods\src\ypit\util\MoneyString.java			미열람

그림 8. 스패로우 보안 약점 검출

스패로우는 검출된 보안 약점에 대하여 각각의 ID 값을 부여하여 관리하고, 신규 보안 약점인지 기존의 제거되지 않은 보안 약점인지에 대하여 신규 여부를 분류하고 있으며, 보안 약점에 대하여 보안침해사고 시 연계 가능성이 크고 피해 정도가 심각한 순으로 위험도를 분류하고 있다. 또한, 체커는 각각의 보안 약점의 이름에 대하여 기술하고 있으며, 해당 보안 약점의 자세한 위치를 표시하기 위하여 파일 경로와 세세한 라인 위치까지 표시하여 소프트웨어 개발자의 편의를 제공하고 있다.

스패로우를 이용하여 검출한 162개의 보안 약점을 항목별로 분류한 결과 행정자치부에서 제공하는 소프트웨어 보안 약점 진단 가이드 47개 항목에서는 크로스 사이트 스크립팅, 사이트 요청 위조 등 7개의 보안 약점 항목이 검출되었으며, CWE, CERT, OWASP, 스패로우 등에서 제공하는 보안 약점 항목에서는 AUTOCOMPLETE_PASSWORD, Consistent Return 등 3개의 보안 약점 항목이 검출되었다.

조선대학교 CHOSUN UNIVERSITY

표 10. 검출된 보안 약점 레벨 별 수

보안 레벨	보안 약점 수
Level 1	96
Level 2	32
Level 3	5
Level 4	28
Level 5	0

스패로우는 표 10과 같이 보안 약점에 대하여 5단계로 분류하고 있으며, Level 1 매우 위험, Level 2 약간 위험 등의 순서로 위험도를 분류하고 있다. 또한, 위험도가 높은 순서에 의하여 보안 약점을 제거할 것을 권고하고 있으며, 이는 보안 약점 위험도가 높은 보안 약점일수록 새로운 보안 약점이 생성될 수 있기 때문이다.

표 11. 검출된 보안 약점 목록

항목	검출 유형
크로스 사이트 스크립팅	검증되지 않은 외부 입력 값에 의해 브라우저에서 악의적인 코드가 실행되는 보안 약점
크로스 사이트 요청 위조	검증되지 않은 외부 입력 값에 의해 브라우저에서 악의적인 코드가 실행되어 사용자 요청 정보가 서버에 보내지는 보안 약점
하드코드 된 패스워드	소스코드 내에 비밀번호를 하드코딩 함에 따라 관리자 비밀번호가 노출되거나, 주기적 변경 등 수정(관리자 변경 등)이 용이하지 않는 보안 약점
경재조건: 검사시점과 사용시점(TOCTOU)	멀티프로세스 상에서 자원을 검사하는 시점과 사용하는 시점이 달라서 발생하는 보안 약점
오류메시지 통한 정보 노출	개발 시 활용을 위한 정보의 출력 메시지를 배포될 버전의 소프트웨어에 포함시킬 때 발생하는 보안 약점

조선대학교 CHOSUN UNIVERSITY

시스템 데이터 정보 노출	End user가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되는 보안 약점
주석문 안에 포함된 패스워드 등 시스템 주요 정보 노출	소스코드 내의 주석문 안에 포함된 주요 정보 password 등이 노출되는 보안 약점
Scope For In Variable	For 문 내에서 변수를 사용할 시 Scope를 고려해야 한다.
AUTOCOMPLETE_PASSWORD	Html Form의 password 필드가 autocomplete로 지정되는 경우 검출한다.
STRING_CONVERSION_WITH_CONSTRUCTOR	기본 자료형을 문자열로 전환할 때 객체 생성을 하여 전환했을 경우 검출한다.

표 11은 웹 자산관리 시스템의 검출된 보안 약점 목록이다. 본 논문에서는 보안 약점 검출 기준으로 CWE, CERT, OWASP, 행정자치부 지정 보안취약점, 국가정보원 보안 적합성 검증 기준이 제시하는 품질 및 이슈 검출 기준에 한해서 검출된 보안 약점을 검출하였다.

2. 보안 약점 제거

이 장에서는 행정자치부의 소프트웨어 보안 약점 진단 가이드와 스패로우에서 제공하는 보안 약점 제거 기법을 참조하여 검출된 보안 약점을 제거한다.

조선대학교
CHOSUN UNIVERSITY

가. 1.3 크로스 사이트 스크립팅(Cross-Site Scripting: XSS)

크로스 사이트 스크립팅 취약점은 취약한 웹 사이트에 악성 스크립트를 포함할 수 있는 손쉬운 방법 중 하나로 공격자들이 가장 많이 선호하는 방식 중 하나이다. 크로스 사이트 스크립팅 취약점은 OWASP Top 10에서도 3번째로 높은 위험으로 매겨져 있으며(CWE/SANS Top 25에서는 4번째로 높은 위험), 취약점 분포도도 '매우 광범위' 한 수준으로 본 취약점은 조직의 정보 자산을 보호하는데 가장 큰 위험 중 하나라고 할 수 있다. 최근에 이슈가 되고 있는 지능적 위협(APT) 및 워터링 홀 유형의 공격에도 크로스 사이트 스크립팅 취약점과 사회 공학적 기법을 결합하여 공격한다. 크로스 사이트 스크립팅 취약점을 이용한 공격은 탐지도 어려울 뿐만 아니라 공격이 성공하는 경우 조직 내부의 PC를 해킹하여 내부 시스템까지 접근할 수 있다[26].

```
1  String goodsNo = request.getParameter("goods_no");
2  String lastState = request.getParameter("last_state");
3  String lastID = request.getParameter("last_id");
4  String laserUser = request.getParameter("last_user");
5  String laserUserName = request.getParameter("last_user_name");
```

그림 9. 검출된 보안 약점 크로스 사이트 스크립팅

(1) 발생유형

크로스 사이트 스크립팅은 게시판 같은 곳의 HTML 문서에 <script>를 이용하여 스크립트 태그 안에 악성 스크립트를 저장하는 방식이다. 이 경우 게시판에는 태그가 나타나지 않으며 사용자는 확인할 수가 없다[26].

그림 9의 1, 2, 3, 4, 5번 라인과 같은 소스코드를 포함하고 있는 게시판 등의 웹 페이지에 <script>alert(document.cookie)</script> 가 포함되어 있는 어떤 페이지를 사용자가 읽을 때마다 브라우저는 이 스크립트를 실행하면서 쿠키 값을 보여주게 된다.

조선대학교
CHOSUN UNIVERSITY

```
1   String goods = request.getParameter("goods_no");
2   if(goodsNo != null){
3       goodsNo = goodsNo.replaceAll("<", "&lt;");
4       goodsNo = goodsNo.replaceAll(">", "&gt;");
5       goodsNo = goodsNo.replaceAll("&", "&amp;");
6   } else {
7       return;
8   }
9
10  String lastState = request.getParameter("last_state");
11  if(lastState != null){
12      lastState = lastState.replaceaAll("<", "&lt;");
13      lastState = lastState.replaceaAll(">", "&gt;");
14      lastState = lastState.replaceaAll("&, "&amp;");
15  } else {
16      return;
17  }
18
19  String lastId = "request.getParameter("last_id");
20  if(lastId != null){
21      lastId = lastId.replasceAll("<", "&lt;");
22      lastId = lastId.replasceAll(">", "&gt;");
23      lastId = lastId.replasceAll("&", "&amp;");
24  } else {
25      return;
26  }
27
28  String laserUser = request.getParameter("last_user");
29  if(laserUser != null){
30      laserUser = laserUser.replaceAll("<", "&lt;");
31      laserUser = laserUser.replaceAll(">", "&gt;");
32      laserUser = laserUser.replaceAll(">", "&amp;");
33  } else {
34      return;
35  }
36
37  String laserUserName = request.getParameter("last_user_name");
38  if(laserUserName != null){
39      laserUserName = laserUserName.replaceAll("<", "&lt;");
40      laserUserName = laserUserName.replaceAll(">", "&gt;");
41      laserUserName = laserUserName.replaceAll("&", "&amp;");
42  } else {
43      return;
44  }
```

그림 10. 시큐어 코딩 규칙 적용 후

조선대학교 CHOSUN UNIVERSITY

(2) 해결방안

그림 10은 시큐어 코딩 규칙을 적용한 소스코드이다. 크로스 사이트 스크립팅 취약점을 제거하기 위해서는 스크립트 등 해킹에 사용될 수 있는 코딩에 사용되는 입력 및 출력값에 대해서 검증하고 무효화시켜야 한다.

크로스 사이트 스크립팅 공격은 기본적으로 <script> 태그를 사용하기 때문에 공격을 차단하기 위해 외부 입력 문자열에 replaceAll() 메소드를 사용하여 <, >, &, “, ” 같이 스크립트 생성에 사용되는 문자열을 문자 참조(HTML entity)로 필터링하고, <, >, &, &quto; 등으로 변경하면 파라미터에 악성코드가 포함되더라도 스크립트가 실행되지 않는다.

HTML 문자 참조란 ASCII 문자를 동일한 의미의 HTML 문자로 변경하는 과정이다. HTML 엔터티는 대부분 인터프리터(특히, 브라우저)에서 특수한 의미를 가지지 않으며, 단순한 문자로 처리된다. 이렇게 인코딩하면 사용자는 <script>가 정상적으로 보이지만 HTML 문서에서는 <script>로 나타나서 브라우저에서 일반 문자로 인식하고 스크립트로 해석되어 실행되지 않는다.

표 12. 위험 문자 인코딩

ASCII 문자	참조문자	ASCII 문자	참조문자
&	&	“	&quto;
<	<	‘	'
>	>	/	/
{	(	)	*

표 12는 HTML 문서에서 악성 스크립트에 포함되어 브라우저에서 실행될 수 있는 문자와 대체 문자를 정리한 것이다. 악성 스크립트는 HTML 태그 안에 포함을 할 수 있으므로 반드시 표 12에 있는 위험 문자의 경우 출력값을 이스케이핑 해야 한다[26]. 해당 보안 약점은 CWE의 79, 80번 행정자치부

조선대학교 CHOSUN UNIVERSITY

소프트웨어 보안 약점 진단 가이드 1.3 크로스 사이트 스크립팅으로 관리되고 있다.

나. 1.10 크로스 사이트 요청 위조(Cross-site Request Forgery: CSRF)

웹 애플리케이션에서 정상적인 경로를 통한 요청과 비정상적인 경로를 통한 요청을 서버가 구분하지 못할 경우 공격자가 스크립트 구문을 이용하여 정상적인 사용자로 하여금 조작된 요청을 전송하도록 하여 게시판 설정 변경 및 자동 댓글, 회원 등급 변경 등의 문제가 발생할 수 있는 취약점이다. 크로스 사이트 요청 위조는 공격을 당한 사용자의 권한을 그대로 사용하게 되므로 사용자의 권한 수준에 따라 그 범위가 달라질 수 있다[27].

```
1  <form method="get" action="<%=request.getContextPath() %>/dropGoods.ypit">
2      <table align="center" boarder="1">
3          <tr align="center" height="30" bgcolor="#dedede">
4              <td><input type="text" id="text_date1" name="date1"/></td>
5              <td><input type="text" id="text_date2" name="date2"/></td>
6              <td><input type="submit" value="조회" style="padding: 3px 25px;"/></td>
7              <td><input type="button" value="출력" style="padding: 3px 25px;"
9                          onclick="javascript:dropGoodsPrint()" /></td>
10         </tr>
11     </table>
12 </from>
```

그림 11. 검출된 보안 약점 크로스 사이트 요청 위조

(1) 발생 유형

그림 11 크로스 사이트 요청 위조는 사용자가 인증한 세션에 공격자가 특정 동작을 수행하여도 세션이 계속 유지되어 정상적인 요청과 비정상적인 요청을 구분하지 못하는 점을 악용하여 공격을 수행한다. 웹 응용프로그램에 요청을 전달할 경우 해당 요청의 적법성을 입증하기 위하여 전달 값이 고정되어 있고, 이러한 자료가 GET 방식으로 전달된다면 공격자가 이를 쉽게 알아내어 원하는

조선대학교
CHOSUN UNIVERSITY

요청을 보냄으로써 위험한 작업을 요청할 수 있게 된다[15].

```
1  <form method="post" action="<%=request.getContextPath() %>/dropGoods.ypit">
2      <table align="center" boarder="1">
3          <tr align="center" height="30" bgcolor="#dedede">
4              <td><input type="text" id="text_date1" name="date1"/></td>
5              <td><input type="text" id="text_date2" name="date2"/></td>
6              <td><input type="submit" value="조회" style="padding: 3px 25px;"/></td>
7              <td><input type="button" value="출력" style="padding: 3px 25px;"
9                      onclick="javascript:dropGoodsPrint()" /></td>
10         </tr>
11     </table>
12 </from>
```

그림 12. 시큐어 코딩 규칙 적용 후

(2) 해결방안

그림 11의 입력 화면 폼에 GET 방식을 사용하였다. GET 방식은 폼 데이터를 URL 뒤에 덧붙여서 전송하기 때문에 전달 값이 노출되어 크로스 사이트 요청 위조 공격에 쉽게 노출될 수 있다.

문제점을 해결하기 위하여 그림 12 시큐어 코딩 규칙을 적용 후의 소스코드는 GET 방식이 아닌 POST 방식을 사용하였다. 또한 입력 화면 폼과 해당 입력을 처리하는 프로그램 사이에 토큰을 생성하여, 공격자의 직접적인 URL 사용이 동작하지 않도록 처리하였다. 중요한 기능에 대해서는 사용자 세션 검증과 재인증을 유도하여 보안 약점을 제거하였다. 해당 보안 약점은 CWE의 352번 행정자치부 소프트웨어 보안 약점 진단 가이드 1.10 크로스 사이트 요청 위조로 관리되고 있다.

조선대학교
CHOSUN UNIVERSITY

다. 2.6 하드코드 된 패스워드

프로그램 내부에 하드코드 된 암호 키를 포함하는 경우 의미하며 만약 백업파일 등의 형태로 외부에 노출되는 경우 관리자의 정보가 노출될 수 있어 위험하다[28]. 또한, 코드 내부에 하드코드 된 패스워드가 인증 실패하는 경우, 시스템 관리자가 그 실패 원인을 파악하기 쉽지 않은 단점이 있다.

```
1    puabic void deleteInoutView(InOutModel modelParam) {
2        try{
3            Class.froName(JDBC_DRIVER);
4            connection = DriverManager.getConnection(DB_URL, DB_ID, DB_PASSWD);
5
6            pstmt = connection prepareStatement("DELETE FROM in_out WHERE id=?");
7
8            pstmt.setInt(1, ModelParam.getInOutNo());
9
10           pstmt.executeUpdate();
11       }
12       catch(Exception e){
13           e.printStackTrace();
14       }finally{
15           close(rs, pstmt, connection);
16       }
17   }
```

그림 13. 검출된 보안 약점 하드코드 된 패스워드

그림 13의 검출된 보안 약점은 URL 주소, IP 주소, PASSWD를 스패로우에서는 하드코드 된 문자열로 인식하였으며, 이를 노출의 위험성을 가지고 있다고 판단하여 검출하였다. 하지만 해당 소스코드는 PASSWD라는 문자를 사용하였을 뿐 하드 코딩된 문자열이 아니므로 이 보안 약점은 무시해도 된다. 해당 보안 약점은 CWE의 259번 행정자치부의 소프트웨어 보안 약점 진단 가이드 2.6 하드코드 된 패스워드로 관리되고 있다.

조선대학교
CHOSUN UNIVERSITY

라. 3.1 경쟁조건: 검사시점과 사용시점(TOCTOU)

동시 또는 거의 동시에 수행을 지원하는 병렬 시스템, 하나 이상의 프로세스가 동작하는 환경에서 시간 및 상태를 부적절하게 관리하여 발생할 수 있는 보안 약점이다[28].

```
1   protected void doPost(HttpServletRequest request, HttpServerletResponse respor
2
3       ServletContext Context = getServletContext();
4       String uploadPath = context.getRealPath("/images/goods");
5                                  ·
                                   ·
                                   ·
6               File orgFile = new File(context.getRealPath("/images/goods/"+file));
7               file movFile = new file(context.getRealPath("/images/goods/"+ext));
8
9               if(movFile.exists())      moveFile.delete();
10              if(orgFile.exists())      orgFile.renameTo(moveFile);
```

그림 14. 검출된 보안 약점 경쟁조건: 검사시점과 사용시점

(1) 발생 유형

그림 14의 검출된 보안 약점은 소스코드가 존재를 확인하는 부분과 실제로 파일을 사용하는 부분을 실행하는 과정에서 시간차가 발생하는 경우, 파일에 대한 삭제가 발생하여 프로그램이 예상하지 못하는 형태로 수행될 수 있다. 이는 시간차를 이용하여 파일을 변경하는 등의 공격에 취약할 수 있다.

공유자원(예: 파일)을 여러 프로세스가 접근하여 사용할 경우, 동기화 구분을 사용하여 한 번에 하나의 프로세스만 접근 가능하도록(synchronized, mutex 등) 하며, 성능에 미치는 영향을 최소화하기 위해 임계 코드 주변만 동기화 구문을 사용한다[15].

조선대학교
CHOSUN UNIVERSITY

```
1   protected void doPost(HttpServletRequest request, HttpServerletResponse respo
2
3       ServletContext Context = getServletContext();
4       String uploadPath = context.getRealPath("/images/goods");
5       int maxSize = 10 * 1024 * 1024;
6       synchronized (SYNC) {
                                  .
                                  .
                                  .
7               File orgFile = new File(context.getRealPath("/images/goods/"+file));
8               file movFile = new file(context.getRealPath("/images/goods/"+ext));
9
10              if(movFile.exists())     moveFile.delete();
11              if(orgFile.exists())     orgFile.renameTo(moveFile);
```

그림 15. 시큐어 코딩 규칙 적용 후

(2) 해결방안

그림 14의 경쟁조건: 검사 시점과 사용 시점의 보안 약점을 제거하기 위하여 그림 15와 같이 synchronize(SYNC)를 사용하여 여러 프로세스가 한 번에 하나의 프로세스만 접근할 수 있도록 하여 보안 약점을 제거하였다. 해당 보안 약점은 CWE-367과 행정자치부의 소프트웨어 보안 약점 진단 가이드 3.1 경쟁조건 검사로 관리하고 있다.

마. 4.1 오류메시지 통한 정보노출, 6.3 시스템 데이터 정보 노출

에러를 처리하지 않거나, 불충분하게 처리하여 에러 정보에 중요 정보(시스템 내부정보 등)가 포함될 때, 발생할 수 있는 취약점으로서 시간 및 상태를 부적절하게 관리하여 발생하는 보안 약점이다[29].

응용프로그램이 실행환경, 사용자, 관련 데이터에 대한 민감한 정보를 포함하는 오류 메시지를 생성하여 외부에 제공하는 경우 공격자가 고의적으로 예외 오류메시지를 발생시켜 공격에 필요한 정보 획득에 악의적으로 사용될 수 있다.

조선대학교
CHOSUN UNIVERSITY

```
1    puabic void deleteInoutView(InOutModel modelParam) {
2       try{
3            Class.froName(JDBC_DRIVER);
4            connection = DriverManager.getConnection(DB_URL, DB_ID, DB_PASSWD);
5
6            pstmt = connection prepareStatement("DELETE FROM in_out WHERE id=?");
7
8            pstmt.setInt(1, ModelParam.getInOutNo());
9
10           pstmt.executeUpdate();
11      }
12      catch(Exception e){
13         e.printStackTrace();
14      }finally{
15           close(rs, pstmt, connection);
16      }
17   }
```

그림 16. 검출된 보안 약점 오류메시지를 통한 정보노출, 시스템 데이터 정보 노출

(1) 발생유형

그림 16 검출된 보안 약점은 일부 개발자의 경우 예외 상황 발생 시 시스템 메시지 등의 정보를 화면에 출력하도록 하는 경우가 많다. 응용프로그램이 실행환경, 사용자 관련 데이터에 대한 민감한 정보를 포함하는 오류 메시지를 생성하여 외부에 제공하는 경우, 공격자의 악성 행위를 도와줄 수 있다. 예외상황 발생 시 예외 이름이나 스택 트레이스를 출력하는 경우, 프로그램 내부 구조를 쉽게 파악할 수 있으므로 시스템의 내부 정보가 화면에 출력하지 않도록 개발해야 한다[29].

조선대학교
CHOSUN UNIVERSITY

```
1    puabic void deleteInoutView(InOutModel modelParam) {
2       try{
3            Class.froName(JDBC_DRIVER);
4            connection = DriverManager.getConnection(DB_URL, DB_ID, DB_PASSWD);
5
6            pstmt = connection prepareStatement("DELETE FROM in_out WHERE id=?");
7
8            pstmt.setInt(1, ModelParam.getInOutNo());
9
10           pstmt.executeUpdate();
11      }
12      catch(Exception e){
13        System.out.println("연결 예외 발생");
14      }finally{
15           close(rs, pstmt, connection);
16      }
17   }
```

그림 17. 시큐어 코딩 규칙 적용 후

(2) 해결 방안

그림 17 시큐어 코딩 규칙 적용 후의 소스코드는 사용자에게 민감한 정보를 포함하는 오류를 출력하지 않도록 미리 정의된 메시지를 제공하게 하여 공격자가 이용할 수 있는 예외 정보 스택 트레이스를 출력하지 않게 변경하여 보안 약점을 제거하였다. 해당 보안 약점은 CWE-209, CWE-497, 소프트웨어 보안 약점 진단 가이드 4.1 오류 메시지 통한 정보 노출, 6.3 시스템 데이터 정보 노출로 관리되고 있다.

바. 2.14 주석문 안에 포함된 패스워드 등 시스템 주요 정보

소스코드 주석 문에 민감한 정보(개인 정보, 시스템 정보 등)가 포함되어 있는 경우, 외부 공격자에 의해 패스워드 등 관련 정보가 노출될 수 있다.

```
[ { name: 'username', value: 'jresig' }, { name: 'password', value: 'secret' } ]
```

그림 18. 검출된 보안 약점 주석문 안에 포함된 패스워드 등 시스템 주요 정보

조선대학교
CHOSUN UNIVERSITY

(1) 발생유형

그림 18 검출된 보안 약점은 소프트웨어 개발자의 편의를 위해서 주석 문에 패스워드 등 중요한 정보를 적어둔 경우, 소프트웨어가 완성된 후에는 해당 주석을 제거하는 것이 매우 어렵게 된다. 또한, 공격자가 소스코드에 접근할 수 있다면, 아주 쉽게 시스템에 침입할 수 있다[28].

(2) 해결방안

프로그램 개발 시 주석문 등에 남겨놓은 사용자 계정이나 패스워드 등의 정보는 개발 완료 후 삭제한다. 해당 보안 약점은 소프트웨어 보안 약점 진단 가이드 2.14 주석문 안에 포함된 패스워드 등 시스템 주요 정보로 관리하고 있다.

사. AUTOCOMPLETE_PASSWORD

AUTOCOMPLETE_PASSWORD는 폼 필드 안에 패스워드 자동완성 기능에 대하여 OFF 설정이 되어 있지 않을 경우 사용자의 비밀번호 정보가 노출될 수 있는 보안 약점이다.

```
1   <form id="slick-login" name="<%=request.getContextPath() %>/Login.ypit" method="post" c
2       <p align="center"><font color="white" size="5">자산관리 시스템<br/></font></p>
3           <label for="username">아이디</label>
4           <input type="text" name="a_id" class="placeholder" placeholder="ID">
5           <label for="password">비밀번호</label>
6           <input type="password" name="a_pwd" class="placeholder" placeholder="Passwd">
7           <input type="submit" value="Log In">
9   </from>
```

그림 19. 검출된 보안 약점 AUTOCOMPLETE_PASSWORD

조선대학교
CHOSUN UNIVERSITY

(1) 발생유형

그림 19 검출된 보안 약점은 HTML 폼 필드 중 password 타입의 필드의 autocomplete 속성 기능이 설정되어 있지 않을 경우 password 자동 완성 기능이 작동하여 사용자의 비밀번호 정보가 노출될 수 있다.

```
1   <form id="slick-login" name="<%=request.getContextPath() %>/Login.ypit" method="post" onSubmit="return ch
2       <p align="center"><font color="white" size="5">자산관리 시스템<br/></font></p>
3           <label for="username">아이디</label>
4           <input type="text" name="a_id" class="placeholder" placeholder="ID">
5           <label for="password">비밀번호</label>
6           <input type="password" name="a_pwd" class="placeholder" placeholder="Passwd" autocomplete="off"/>
7           <input type="submit" value="Log In">
9  </from>
```

그림 20. 시큐어 코딩 규칙 적용 후

(2) 해결방안

그림 20의 HTML의 from 필드를 보면 input 타입의 필드 속성에 autocomplete 속성을 off 하였다. autocomplete를 off로 지정함으로써 password가 자동으로 생성되는 문제점을 해결하였다.

아. 문제점 검출

이 장에서는 보안 약점 이외에 소스코드 내부에 보안 약점이 될 수 있는 문제점에 대하여 도출하여 해당 문제점을 제거한다.

(1)Scope For In Variable

조선대학교 CHOSUN UNIVERSITY

for 문에서 변수를 사용할 시 scope를 고려해야 한다. ‘var’ 를 사용하지 않고 변수를 사용할 시 외부의 변수를 사용하기 때문에 의도치 않은 변수의 변경이 발생할 수 있다.

```
1   for(i in df) {
2       if(df.hasOwnProperty(i)) {
3           tmp.state[i] = df[i];
4       }
5   }
```

그림 21. 검출된 문제점

그림 21의 for 문 내에 scope를 고려하여 'var'를 사용하여 의도치 않은 변수가 발생할 수 있는 문제점을 해결하였다.

```
1   for(var i in df) {
2       if(df.hasOwnProperty(i)) {
3           tmp.state[i] = df[i];
4       }
5   }
```

그림 22. 문제점 해결

(2) Global Variable

단순히 ‘var’ 선언문을 사용하지 않았을 경우 검출한다. 전역변수의 사용은 디버깅을 어렵게 만들기 때문에 최소한으로 사용해야 한다.

(3) STRING_CONVERSION_WITH_CONSTRUCTOR

그림 23은 기본 자료형을 문자열로 전환할 때 객체 생성을 하여 전환했을 경우 검출한다.

조선대학교
CHOSUN UNIVERSITY

```
생성자를 사용하여 문자열 변환을 수행하지 않도록 합니다.
String str = new Integer(money).toString();
```

그림 23. 검출된 문제점

그림 24는 새로운 객체를 생성하지 않고 문자열로 변환하도록 하여 문제점을 해결하였다.

```
생성자를 사용하여 문자열 변환을 수행하지 않도록 합니다.
String str = Integer(money).toString();
```

그림 24. 문제점 해결

조선대학교
CHOSUN UNIVERSITY

제 4 장 연구 결과

보안 약점을 검출하기 위하여 CWE, CERT, OWASP, 행정자치부 지정 보안 취약점, 국가정보원 보안 적합성 검증 기준이 제시하는 품질 및 이슈 검출 기준에 적합한 스패로우를 이용하여 소스코드를 분석, 보안 약점을 검출하였다.

보안 약점을 제거하기 위한 순서로 연계 가능성이 크고 피해가 심각한 순으로 보안 약점을 Level 별로 분류하였다. 분류 결과 Level 1은 96개, Level 2는 33개 Level 3은 5개, Level 4는 28개, Level 5에서는 0개의 보안 약점이 검출되었다.

순번 ▸	8	7	6	5	4	3	2	1
분석 시작 일시	2015-03-23 16:41	2015-03-23 16:38	2015-03-23 09:13	2015-03-18 09:14	2015-03-17 11:29	2015-03-17 11:14	2015-03-17 09:47	2015-03-12 15:15
분석 종료 일시	2015-03-23 16:42	2015-03-23 16:39	2015-03-23 09:14	2015-03-18 09:15	2015-03-17 11:29	2015-03-17 11:15	2015-03-17 09:48	2015-03-12 15:15
분석 시간	00h 00m 41s	00h 00m 41s	00h 01m 04s	00h 01m 19s	00h 00m 41s	00h 00m 41s	00h 00m 45s	00h 00m 42s
총 이슈	0	6	65	65	65	65	97	162
신규 이슈	0	0	0	0	5	0	32	162
기존 이슈	0	6	65	65	60	64	65	0
제거된 이슈	6	59	0	0	5	33	96	0
소스 파일 수	60	60	60	60	60	60	60	60
모든 파일 수	60	60	60	60	60	60	60	60
빌드 라인 수	16,834	16,834	16,834	16,830	16,830	16,807	16,798	16,798
밀도	0	0.36	3.86	3.86	3.86	3.87	5.77	9.58
상태	Completed	Completed	Completed	Completed	Completed	Completed	Completed	Completed

그림 25. 보안 약점 제거 결과

그림 25는 보안 약점 제거 과정별 제거 결과이다. 시큐어 코딩을 적용하기 위하여 분석 도구를 사용하는 이유로 그림 25의 신규 이슈 부분을 보면 알 수 있다. 1번째 신규 이슈 162개가 검출되었다. 최초 검출하였기 때문에 신규 이슈로 등록되었다. 2번째, 4번째 검출 결과를 보면 각각의 신규 이슈 32개, 5개가 검출되었다. 이는 기존의 보안 약점을 제거하는 과정 중 새로운 보안 약점이

조선대학교
CHOSUN UNIVERSITY

생성될 수 있음을 보여준다. 그러나 분석 도구를 이용하면 새롭게 생성되는 보안 약점 또한 검출할 수 있다.

보안 약점을 제거하기 위하여 총 6차례에 걸쳐 보안 약점을 분석, 검출하여 제거하였다. Level 위험도가 높은 순서로 보안 약점을 제거하였으며, 최종적으로 보안 약점 제거 과정 중에 생기는 신규 보안 약점을 포함한 총 199개의 보안 약점을 제거하였다. 그림 26은 보안 약점 제거 결과를 그래프로 나타낸 것이다.

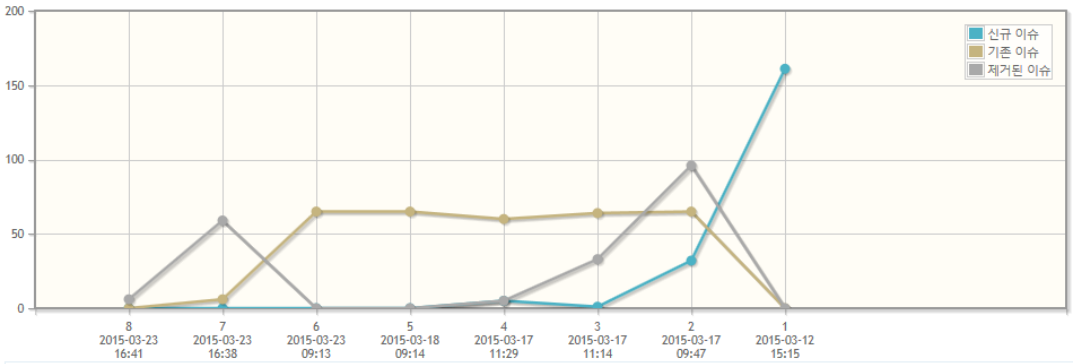

그림 26. 보안 약점 제거 결과 그래프

보안 약점 제거 후 스패로우를 이용하여 보안 약점을 분석, 검출한 결과 보안 약점이 검출되지 않았다. 또한, 공공기관 사업 참여 시 필수로 요구되는 공통평가 기준 인증을 획득한 스패로우를 이용하여 보안 약점을 검출, 제거하였기 때문에 공공기관 등에서도 사용 가능하다. 이로써 보안성과 안전성이 강화된 시큐어 코딩을 적용한 웹 자산관리 시스템을 제시한다.

조선대학교
CHOSUN UNIVERSITY

제5장 결 론

본 논문에서는 기업에서 사용하고 있는 웹 자산관리 시스템의 보안침해사고를 예방하기 위해 시큐어 코딩 기법을 적용하였다. 시큐어 코딩은 소프트웨어 개발 단계에서 소스코드 내부에 존재하는 보안 약점을 제거함으로써 보안침해사고를 예방하는 방어적 프로그래밍 기법이다.

보안 약점을 검출하기 위하여 시맨틱 기반의 소스코드 정적 분석 도구인 스패로우를 이용하여 소스코드를 분석하였다. 분석결과 총 11개 항목 162개의 보안 약점이 검출되었다. 스패로우에서 제공하는 보안 약점 항목에 대한 설명과 행정자치부의 소프트웨어 보안 약점 진단 가이드를 바탕으로 각 항목에 대하여 보안 약점을 제거하였다. 보안 약점 검출과 제거하기 위한 기준으로는 CWE, CERT, OWASP, 행정자치부지정 보안취약점, 국가정보원 보안 적합성 기준에 한하여 보안 약점을 검출, 제거하였다.

시큐어 코딩 규칙 적용 후, 웹 자산관리 시스템의 소스코드에서 보안 약점이 제거되었는지 확인하기 위해 스패로우로 재분석한 결과 보안 약점이 모두 제거된 것을 확인하였다. 보안 약점을 제거하는 과정 중 새로운 보안 약점이 생성될 수 있다. 그러나 소스코드 분석 도구를 이용하면 새롭게 생성되는 보안 약점 또한 검출할 수 있다. 보안 약점 최종 분석결과 신규 보안 약점을 포함한 총 199개의 보안 약점이 제거되었다. 이러한 연구를 바탕으로 보안성과 안전성이 강화된 안전한 시큐어 코딩을 적용한 웹 자산관리 시스템 사용이 가능하다.

조선대학교
CHOSUN UNIVERSITY

참고문헌

[1] Gartner, http://gartner.com, Nov. 2005.

[2] NIST, “The Economic Impacts of Inadequate Infrastructure for Software Testing," May 2002.

[3] 손윤식, 오세만, “자바 시큐어 코딩,” 정보과학학회지, 제 28권, 제 2호, pp. 55-62, 2010.

[4] 위키백과, “소프트웨어 개발 보안,” http://ko.wikipedia.org/wiki/소프트웨어_개발_보안/, 2013.

[5] 마이크로소프트, http://www.microsoft.com/security/.

[6] 정다혜, 최진영, 이송희, “시큐어 코딩 중심으로 본 원자력 관련 소프트웨어,” 정보보호학회논문지, 제 23권, 제 2호, pp. 243-250, 2013.

[7] A. John, R. Peter, "Electric Communication Development," Communications of the ACM, Vol. 40, pp.71-79, May. 1997.

[8] CWE (Common Weakness Enumeration), http://cwe.mitre.org/.

[9] 안준선, 이은영, 창병모, “SW 개발보안을 위한 보안약점 표준목록 연구,” 정보보호학회지, 제 25권, 제 1호, pp. 7-11, 2015.

[10] 2010 OWASP Top 10, http://www.owasp.org/index.php/top_10_2010.

[11] 2011 CWE/SANS Top25 Most Dangerous Software Error, http://cwe.mitre.org/top25/.

[12] OWASP 코리아 챕터, http://www.owasp.or.kr/.

[13] CWSS(Common Weakness Scoring System), http://cwe.mirte.org/cwss/.

[14] 안준선, 방지호, 이은영, “소프트웨어 보안약점의 중요도에 대한 정량 평가 기준 연구,” 정보보호학회논문지, 제 22권 제 6호, pp. 1407-1417, 2012.

조선대학교
CHOSUN UNIVERSITY

[15] 행정자치부, “소프트웨어 보안약점 진단가이드,” 2012.

[16] 한지혜, “시큐어코딩을 적용한 학습관리시스템 분석 및 보안 개선방안 연구: blackboard와 modle을 중심으로,” 석사학위논문, 고려대학교, 2015.

[17] 위키백과, “정적 프로그램 분석,” http://ko.wikipedia.org/wiki/정적_프로그램_분석/, 2013.

[18] 김우연, 박지우, 손윤식, 이창환, 구현우, 김영자, 오세만, “안전한 자바 코딩 규약에 기반한 취약성 분석기의 설계,” 한국컴퓨터종합학술대회, 제 36권, 제 1호, pp. 589-593, 2009.

[19] H. Chen and D. Wagner, "MOPS: an Infrastructure for examining Security Properties of Software," Proceedings of the 9th ACM conference on Computer and Communications Security, pp. 235-244, 2002.

[20] Sparrow, http://www.fasoo.com/site/fasoo/sourcecodeanalysis/sparrow.do/.

[21] Coverity, Coverity Static Analysis, http://www.coverity.com/products/static-analysis.html/, 2009.

[22] FindBugs, http://findburgs.sourceforge.net/, 2012.

[23] PMD, http://pmd.sourceforge.net/pmd-5.0.0/, 2012.

[24] Fortify Software, Fortify Source Code Analysis(SCA), http://www.fortify.com/products/sca/, 2009.

[25] 김재생, 신화성, “웹기반 IT 자산관리 시스템의 구축,” 한국디지털정책학회, 제 10권 8호, pp. 193-200, 2012.

[26] 성윤기, “크로스 사이트 스크립팅(XSS) 공격 종류 및 대응 방법,” http://www.kisa.or.kr/uploadfile/201312/201312161355109566.pdf/, 2013.

[27] 한국 인터넷 진흥원, “정보시스템 개발·운영자를 위한 홈페이지 취약점 진단·제거 가이드," 2013.

조선대학교
CHOSUN UNIVERSITY

[28] 행정자치부, “소프트웨어 개발보안 가이드,” 2012.

[29] 최경철, 김태한, “웹 모의해킹 및 시큐어 코딩 진단 가이드,” 시큐북, pp. 437-587, 2014.

조선대학교
CHOSUN UNIVERSITY

감사의 글

대학원 석사과정 동안 저를 이끌어 주시고 본 논문의 연구계획부터 완성에 이르기 까지 지도하여 주신 반성범 교수님께 깊은 감사를 드립니다. 논문 심사과정을 통해 본 논문을 지도하여 주시고 심사하여주신 박종안 교수님과 신주현 교수님께 감사를 드립니다.

대학원 생활 및 연구에 큰 도움이 되어준 문해민, 김민구, 고경리에게 감사하고, 실험실 생활을 즐겁게 해준 재상, 강효, 원웅, 세현, 규호, 진수에게 감사의 뜻을 전합니다. 또한 논문 연구를 위하여 스패로우를 지원해주신 ㈜파수닷컴의 이재용 부장님, 김경철 선임님, 웹 자산관리 시스템과 석사학위 취득을 위하여 여러 방면으로 지원해주신 ㈜유풍정보기술의 최영문 사장님, 박성운 전무님, 김현구 대리님께 감사를 드립니다. 마지막으로, 항상 저에게 큰 힘이 되어주신 부모님과 동생 그리고 항상 곁에서 격려해주고 도움이 되어준 수나에게 이 논문을 드립니다.

2015년 8월

김동운 올림