**Secure S/W Development**

**-과제 관리 시스템**

**Instructor :**Prof. Shinhye Park

**Prepared by :**<Team 1>  
임혜선  
조은상  
양경석  
이수훈  
홍명기

# Table of Contents

[Table of Contents 2](#_Toc437531442)

[Figure & Table 4](#_Toc437531443)

[1. Introduction 7](#_Toc437531444)

[1.1 Purpose 7](#_Toc437531445)

[1.2 Specification 7](#_Toc437531446)

[1.3 Function 8](#_Toc437531447)

[1.4 Site Map 8](#_Toc437531448)

[1.5 Database Structure 10](#_Toc437531449)

[2. 보안 요구사항 11](#_Toc437531450)

[2.1 기본 보안설정 11](#_Toc437531451)

[2.2 OWASP 10, S/W 개발보안 47 12](#_Toc437531452)

[3. Secure Coding 14](#_Toc437531453)

[3.1 회원가입필드, 유효한 Value 만 입력하도록 적용 14](#_Toc437531454)

[3.1.1 내용 14](#_Toc437531455)

[3.1.2 적용방법 및 Code 15](#_Toc437531456)

[3.2 ID 중복체크 16](#_Toc437531457)

[3.2.1 내용 16](#_Toc437531458)

[3.2.2 적용방법 및 Code 16](#_Toc437531459)

[3.3 Password 입력 시, \*로 표시 17](#_Toc437531460)

[3.3.1 내용 17](#_Toc437531461)

[3.3.2 적용방법 및 Code 17](#_Toc437531462)

[3.4 Password 30일 마다 변경 알림 18](#_Toc437531463)

[3.4.1 내용 18](#_Toc437531464)

[3.4.2 적용방법 및 Code 18](#_Toc437531465)

[3.5 Password 5회 잘못 입력 시 접근제한 19](#_Toc437531466)

[3.5.1 내용 19](#_Toc437531467)

[3.5.2 적용방법 및 Code 19](#_Toc437531468)

[3.6 SQL Injection 20](#_Toc437531469)

[3.6.1 내용 20](#_Toc437531470)

[3.6.2 적용방법 및 Code 21](#_Toc437531471)

[3.7 DB 암호화 22](#_Toc437531472)

[3.7.1 내용 22](#_Toc437531473)

[3.7.2 적용방법 및 Code 22](#_Toc437531474)

[3.8 인증 및 세션관리 취약점 23](#_Toc437531475)

[3.8.1 내용 23](#_Toc437531476)

[3.8.2 적용방법 및 Code 23](#_Toc437531477)

[3.9 Cross-Site Scripting, Cross-Site Request Forgery 24](#_Toc437531478)

[3.9.1 내용 24](#_Toc437531479)

[3.9.2 적용방법 및 Code 25](#_Toc437531480)

[3.10 취약한 직접 객체 참조 27](#_Toc437531481)

[3.10.1 내용 27](#_Toc437531482)

[3.10.2 적용방법 및 Code 27](#_Toc437531483)

[3.11 민감 데이터 노출 28](#_Toc437531484)

[3.11.1 내용 28](#_Toc437531485)

[3.11.2 적용방법 및 Code 28](#_Toc437531486)

[3.12 기능 수준의 접근통제 30](#_Toc437531487)

[3.12.1 내용 30](#_Toc437531488)

[4. Static Analysis 32](#_Toc437531489)

[4.1 PMD 32](#_Toc437531490)

[4.2 Acunetix 35](#_Toc437531491)

[5 Project Summary 37](#_Toc437531492)

[5.1 Lessons Learns 37](#_Toc437531493)

[5.2 Work breakdown Structure 38](#_Toc437531494)

# Figure & Table

[Figure 1. 과제관리 시스템 Specification 7](#_Toc437531495)

[Figure 2. Sitemap 8](#_Toc437531496)

[Figure 3. 회원가입 입력사항 14](#_Toc437531497)

[Figure 4. 회원가입 유효 값 체크 에러메시지 14](#_Toc437531498)

[Figure 5. 회원가입 유효 값 체크 Code 15](#_Toc437531499)

[Figure 6. ID 중복체크 16](#_Toc437531500)

[Figure 7. 사용가능 ID 확인 16](#_Toc437531501)

[Figure 8. ID 중복체크 \_Join\_form.jsp Figure 9. ID 중복체크\_ Join\_IDCheck.jsp 16](#_Toc437531502)

[Figure 10. Password 입력 시, \*로 표시 17](#_Toc437531503)

[Figure 11. Password 입력 시, \*로 표시\_ Code 17](#_Toc437531504)

[Figure 12. Password 30일 마다 변경 알림 18](#_Toc437531505)

[Figure 13. Password 30일 마다 변경 알림\_Code 18](#_Toc437531506)

[Figure 14. 1.1 Password 입력 5회 잘못 입력 시 접근제한 19](#_Toc437531507)

[Figure 15. Password 입력 5회 잘못 입력 시 접근제한\_Code 19](#_Toc437531508)

[Figure 16. SQL Injection 구현 전 20](#_Toc437531509)

[Figure 17. SQL Injection 구현 후 20](#_Toc437531510)

[Figure 18. SQL Injection Code\_1 21](#_Toc437531511)

[Figure 19. SQL Injection Code\_1 21](#_Toc437531512)

[Figure 20. DB 암호화 전 22](#_Toc437531513)

[Figure 21. DG 암호화 적용 후 22](#_Toc437531514)

[Figure 22. SHA256 적용 23](#_Toc437531515)

[Figure 23. SHA256 적용 23](#_Toc437531516)

[Figure 24. XSS, CSRF Test Script 24](#_Toc437531517)

[Figure 25. XSS, CSRF – 구현 전 24](#_Toc437531518)

[Figure 26. XSS, CSRF – 구현 후 24](#_Toc437531519)

[Figure 27. XSS, CSRF - 파일형식 규제-구현 전 25](#_Toc437531520)

[Figure 28. XSS, CSRF - 파일형식 규제-구현 후 25](#_Toc437531521)

[Figure 29. 게시판 스크립트 입력방지 26](#_Toc437531522)

[Figure 30. 게시판 업로드 파일 형식규제\_Code 26](#_Toc437531523)

[Figure 31. 주소입력 강제접근 에러 메시지. 27](#_Toc437531524)

[Figure 32. 주소입력 강제접근\_Code 27](#_Toc437531525)

[Figure 33. AES256 암호화 28](#_Toc437531526)

[Figure 34. AES 256 암호화 28](#_Toc437531527)

[Figure 35. AES256 복호화 29](#_Toc437531528)

[Figure 36. AES256 복호화\_Code 29](#_Toc437531529)

[Figure 37. AES256 복호화 29](#_Toc437531530)

[Figure 38. 기능수준 접근통제\_학생 30](#_Toc437531531)

[Figure 39. 기능수준 접근통제\_관리자 30](#_Toc437531532)

[Figure 40. 기능수준 접근통제\_학생 30](#_Toc437531533)

[Figure 41. 기능수준 접근통제\_교수 31](#_Toc437531534)

[Figure 42. PMD 분석결과 - 2015.11.20 32](#_Toc437531535)

[Figure 43. PMD 분석결과 - 2015.12.10 34](#_Toc437531536)

[Figure 44. Acunetix 분석결과\_2015.11.24 35](#_Toc437531537)

[Figure 45. Acunetix 분석 \_ 수정사항 35](#_Toc437531538)

[Figure 46. Acunetix 분석결과 \_ 2015.12.10 36](#_Toc437531539)

[Figure 47. S/W 개발보안 47 37](#_Toc437531540)

[Figure 48. WBS 38](#_Toc437531541)

[Table 1. 과제관리 시스템 Page 7](#_Toc437531542)

[Table 2. Site Map - Page 별 address 9](#_Toc437531543)

[Table 3. Database\_member 10](#_Toc437531544)

[Table 4. Database\_assignment 10](#_Toc437531545)

[Table 5. Databse\_data\_list 10](#_Toc437531546)

[Table 6. 기본보안 적용내용 12](#_Toc437531547)

[Table 7. OWASP 10 적용내용 13](#_Toc437531548)

# Introduction

## Purpose

과제 제출, 점수부여, 점수확인을 하는 AIMS2 E-Class 과제 게시판 기능을 하는 Web Service 형태의 [과제 관리 시스템]을 개발하였다. Secure S/W 설계 과목에서 추구하는 Secure coding을 위하여 우리의 프로젝트에 CIA, OWASP Top10을 적용하고 Manual Test 및 PMD, Acunetix로 정적분석 결과를 기반으로 지적된 보안요구 사항을 적용함으로써 이미 존재하고 있거나 발생 가능한 위협과 취약점에 대한 우려사항을 해결하고자 하였다.

## Specification

[과제 관리 시스템]은 Web service를 하는 시스템으로 Client에게 Web browser를 통하여 서비스 한다. Web Service로 Apache Tomcat8, DB Server로 My SQL을 사용하였다.

|  |  |
| --- | --- |
| **Home page** | <http://210.107.197.133:8080/secure/> |
| **FTP Server** | <ftp://210.107.197.133:21> |
| **Database** | 210.107.197.133/myadmin/ |

Table 1. 과제관리 시스템 Page

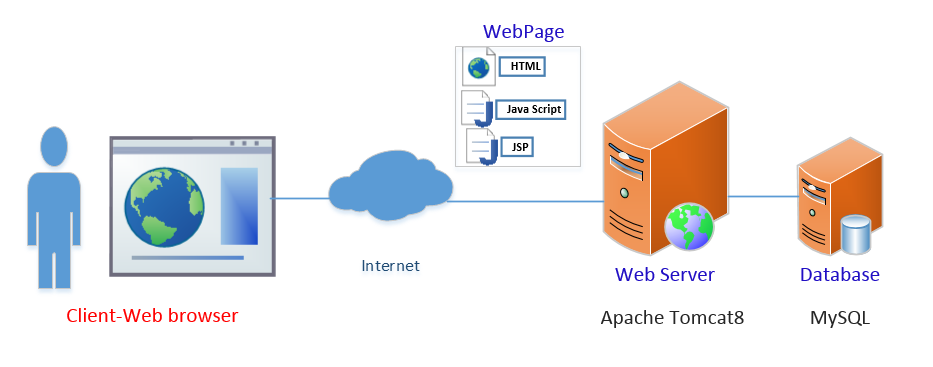


Figure 1. 과제관리 시스템 Specification

## Function

[과제 관리 시스템]에서 제공하는 서비스에 포함된 기능은 아래와 같다.

1. 회원가입, 회원정보 수정, 회원관리, Log-in, Log-out
2. 게시판: 과제등록, 등록과제 수정, 과제제출, 제출된 과제확인, 제출과제 수정, 점수부여, 점수확인, 점수수정, 과제파일 업로드
3. Access control: 사용자 권한에 따른 page, data 접근통제 (관리자, 교수, 학생)

## Site Map

[과제 관리 시스템]의 Site Map은 아래와 같다.

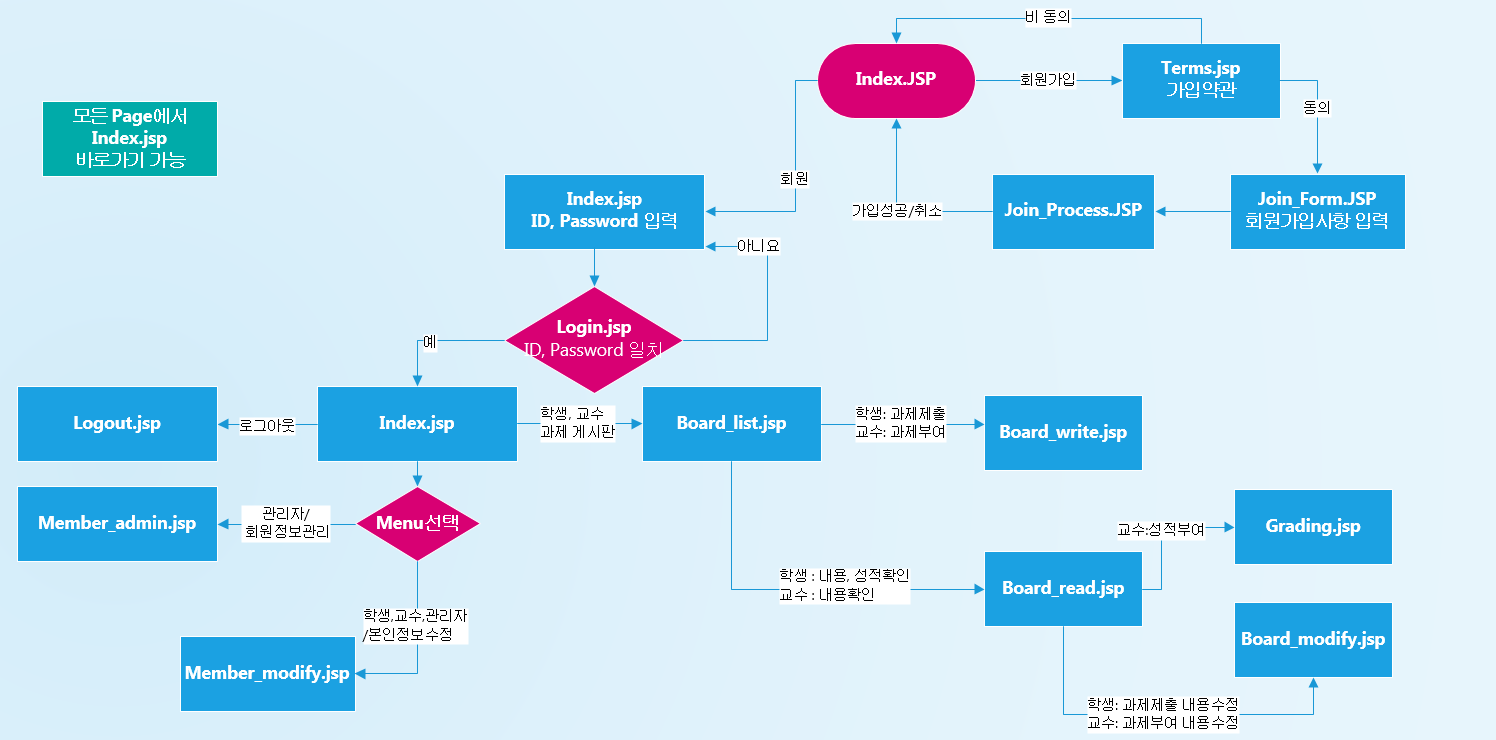


Figure 2. Sitemap

|  |  |
| --- | --- |
| **Main Page** | <http://210.107.197.133:8080/secure/> |
| **회원가입 약관동의** | <http://210.107.197.133:8080/secure/register_notice.jsp> |
| **회원가입 Form** | <http://210.107.197.133:8080/secure/register_form.jsp> |
| **회원정보 수정** | <http://210.107.197.133:8080/secure/member_modify.jsp> |
| **사용자 권한변경** | <http://210.107.197.133:8080/secure/user_info.jsp> |
| **과제목록** | <http://210.107.197.133:8080/secure/board_list.jsp> |
| **과제 상세보기** | <http://210.107.197.133:8080/secure/board_read.jsp> |
| **게시글 작성, 과제등록** | <http://210.107.197.133:8080/secure/write_index.jsp> |
| **과제 내용 수정** | <http://210.107.197.133:8080/secure/board_modify.jsp> |
| **점수부여** | <http://210.107.197.133:8080/secure/grading.jsp>  <http://210.107.197.133:8080/secure/sub_grading.jsp> |
| **회원정보관리(관리자)** | <http://210.107.197.133:8080/secure/member_admin.jsp> |

Table 2. Site Map - Page 별 address

## Database Structure

* Table 1 : member

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **value** | **column** | **value** |
| No | 회원가입번호 | Prof | 교수인지 확인 |
| Id | ID | Email | E-mail |
| Pw | 비밀번호 | Phone | 전화번호 |
| Name | 이름 | Register\_date | 회원가입일 |
| Student\_no | 학번 | Password\_changed\_date | 비밀번호 변경일 |
| Fail\_flag | 로그인 실패횟수 |  |  |

Table 3. Database\_member

* Table 2 : assignment

|  |  |  |  |
| --- | --- | --- | --- |
| **column** | **value** | **column** | **value** |
| submit\_no | 제출번호 | student\_no | 학번 |
| assignment\_no | 과제번호 | Title | 제목 |
| content | 내용 | grade | 학점 |
| prof | 교수인지 확인 | Submit\_yesno | 제출여부 |
| Assignment\_register\_date | 제출일 |  |  |

Table 4. Database\_assignment

* Table 3 : data\_list

|  |  |  |  |
| --- | --- | --- | --- |
| **column** | **value** | **column** | **value** |
| Data\_no | 제출번호 | Board\_no | 과제번호 |
| File\_name | 과제첨부파일 | User\_id | 제출자 ID |
| New\_file\_name | 첨부파일명 변환 | User\_no | 제출자 학번 |
| publisher | 제출자 이름 |  |  |

Table 5. Databse\_data\_list

# 보안 요구사항

과제관리 시스템에서 적용된 보안 요구사항을 정리한 항목으로써, 이에 대한 자세한 내용은 3. Secure coding에서 다룬다. 표 내의 blue font는 S/W 개발보안 47개 항목을 구현된 OWASP10과 기본 보안설정에 Match시켜 확인하였다.

## 기본 보안설정

|  |  |
| --- | --- |
| **내 용** | **적용사항** |
| 회원가입 필드,  유효한 value만 입력 | 회원가입 시, 사용자 입력 value를 유요한 value만 입력하도록 적용. *[포맷스트링 삽입- S/W47]* |
| Password 입력 시, \* 로 표시 | 입력 한 password대로 보이지 않도록 처리 |
| Password 30일 마다 변경 알림 | 사용자 password 설정 후, 30일 경과 시, 아래 message box 출력 (password\_changed\_date) |
| Password 전송 시, 평문으로 전송  -> 암호화 전송  *[중요정보 평문전송-SW47] [하드코드된 비밀번호-SW47]* | 암호화된 정보를 데이터베이스에 저장, 데이터베이스에서 암호화된 정보를 가져와 복호화하여 사용 |
| Password 설정 시,  영문+숫자+특수문자 5-20자 조건 | 5~20자 영문대소문자, 숫자, 특수문자 혼합하여 사용, 그 외 회원가입 필드 조건에 대해 조건에 맞지 않을 경우, 아래 Message box 출력 *[취약한 비밀번호 허용 – S/W47]* |
| Log-In Password 5회 잘못 입력 시, 해당 정보로 접근 막고, 관리자 연락처와 조치사항 알림 | 로그인 실패 카운터 (fail\_flag)를 db에 갱신하여 5회 이상 실패 시 로그인 불가, 관리자 연락 메시지 출력  *[반복된 인증시도 제한 기능부재 – S/W47]* |
| ID 중복체크 | 회원가입 시, ID 중복체크, 중복된 ID 입력 시, 아래 Message 출력 |
| DB 접근 : 아이디/비밀번호 하드코딩 -> 별도 로그인 기능 추가 | DB 접근: 아이디/비밀번호 하드코딩 -> 별도 로그인 기능 추가 |

Table 6. 기본보안 적용내용

## OWASP 10, S/W 개발보안 47

|  |  |  |
| --- | --- | --- |
| **OWASP** | **내 용** | **적용사항** |
| A1. SQL Injection | ID, Password 입력 창에서 query 문 입력 통제 (입력한 데이터 Filtering) | 정규식을 이용하여 가용된 문자 이외 사용하지 못하도록 구현 |
| 페이지 오류 시 Error Message 차단 *[오류 메시지를 통합 정보노출-SW47]*  *[오류상황대응부재 – S/W47]*  *[부적절한 예외처리 – S/W47]* | error.html로 이동하도록 조치. web.xml에 코드 삽입 |
| A2. 인증 및 세션관리 취약점  *[잘못된 세션에 의한 정보노출 – S/W47]* | 로그인 과정 암호화(SSL: Secure Socket Layer) | 로그인이나 회원가입시 ID는 256bit의 키(AES256) Password는 160bit(SHA256)의 Hash로 암호화하여 사용 (암호화, 복호화) |
| 로그인 후, 10분간 활동 없을 시에 강제 로그아웃 | 페이지 이동 시 로그인한 시간부터 체크해서 10분 경과 시 Message box 출력 후,  로그아웃(세션삭제) |
| IE 창 닫으면 자동 로그아웃 | 브라우저 자체적으로 지원됨. 다만, 크롬은 세션삭제가 안됨. 브라우저 문제로 알려짐. |
| A3. XSS  A8. CSRF *[크로스사이트 스크립팅,  신뢰되지 않은 URL 주소로 자연접속 연결,  크로스사이트 요청 위조- SW47]* | 입력한 데이터 Filtering (스크립트 실행 금지) | 게시판 내용-CKEDITOR의 소스 버튼 제거 |
| 입력 값 검증 및 치환(게시판) | 제목-<. >, ', " 등 xss에 사용될 수 있는 특수문자 &lt 등으로 치환 => 태그실행이 되지 않고, 스크립트 자체가 텍스트로 출력. |
| 게시판 file upload 시 upload file 형식규제  *[위험한 형식 파일 업로드 – S/W47]* | 파일 확장자가 .zip이 아닌 경우에는 경고메시지 출력하고 전송하지 않음(글 작성/수정 시) |
| A4. 취약한 직접 객체 참조 | 오류 Page, Message -> Redirection | "페이지 오류 시 Error Message 차단"에서 완료 |
| 웹 페이지 강제 입력 시 세션 체크하여 권한 확인, 주소 입력 접속차단 *[경로조작 및 자원삽입 –S/W47]* | userid 세션 없고 , assignment\_no 없으면 index.jsp로 강제 리다이렉트 or error message 출력 |
| A6. 민감 데이터노출 | 중요 Data 암호화  *[중요정보 평문저장, 중요정보 평문전송, 충분하지 않은 키 길이 사용, 적절하지 않은 난수값 사용, 하드코드 된 암호화 키- S/W47]* | AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 암호화 |
| Web Page에서 DB Connection 시 ID, PW 암호화 | AES256 Cipher를 dbUser, dbPass로 사용 후 connection시 디코딩하여 사용 |
| A7. 기능 수준의  접근통제 누락 | 사용자 권한에 따른 페이지 접속 여부  *[적적한 인증 없는 중요기능 허용 –S/W47]*  *[부적절한 인가 – S/W47] [중요한 자원에 대한 잘못된 권한 설정-S/W47]* | 관리자, 교수, 학생 권한으로 나뉘어 해당하는 기능만 사용할 수 있도록 함 |

Table 7. OWASP 10 적용내용

# Secure Coding

이 장에서는 과제관리 시스템에 적용한 보안요구사항을 항목별로 구현내용과 함께 Coding 부분을 담았다.

## 회원가입필드, 유효한 Value 만 입력하도록 적용

### 내용

과제관리 시스템은 회원가입 시, 아래의 항목을 입력하도록 되어있다.



Figure 3. 회원가입 입력사항

회원가입 시, 각 필드 별로 정해진 조건에 맞지 않을 경우, 아래 메시지를 통해 조건에 맞는 값을 입력하도록 하고 있다.

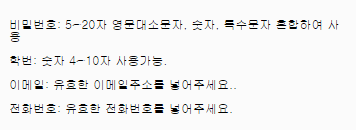


Figure 4. 회원가입 유효 값 체크 에러메시지

### 적용방법 및 Code

각 필드 별로 입력 유효한 값을 정규식으로 정의하고 유효성을 검사하도록 적용하였다. 아래는 각 필드별 정의된 값을 보여준다.

|  |
| --- |
| var reg\_id = /^[a-z0-9\_]{5,15}$/; //아이디: 5~15자 영문소문자, 숫자, 특수문자 \_ 사용가능  var reg\_pw = /^(?=.\*[a-zA-Z])(?=.\*[!@#$%^\*+=-]|.\*[0-9]).{5,20}$/;  //비밀번호: 5~20자 영문대소문자, 숫자, 특수문자 혼합하여 사용  var reg\_number = /^[0-9]{4,10}$/; //학번: 숫자 4~10자 사용가능.  var reg\_email = /^[0-9a-zA-Z\_\-]+@[0-9a-zA-Z\_-]+(\.[0-9a-zA-Z\_\-]+)\*$/;  //이메일: 유효한 이메일 주소를 넣어주세요.  var reg\_phone = /^\d{2,3}\-\d{3,4}\-\d{4}$/;   //전화번호: 유효한 전화번호를 넣어주세요. |

아래 그림은 실제로 구현된 Code를 보여준다.

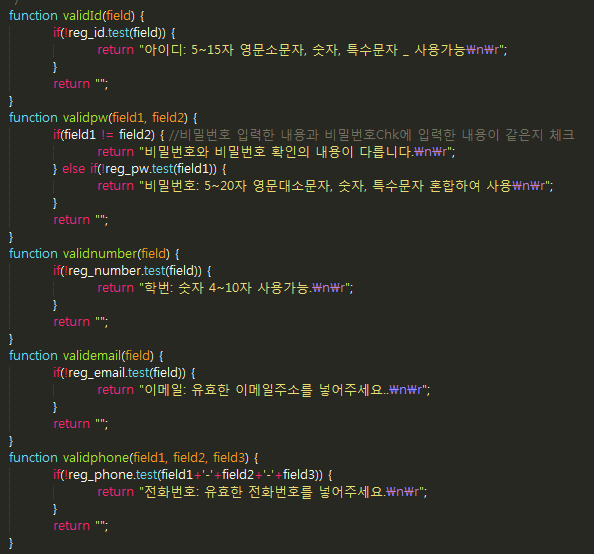


Figure 5. 회원가입 유효 값 체크 Code

## ID 중복체크

### 내용

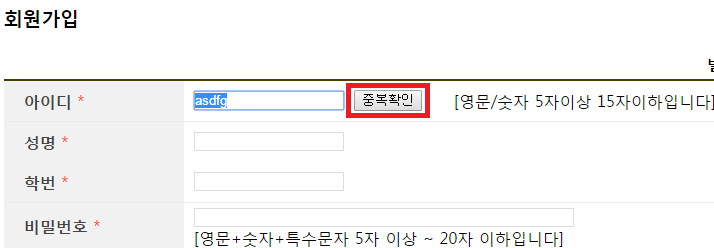
회원가입 시, 가입정보를 입력하는 Field에서 ID 중복체크를 하는 기능으로써, 사용자의 고유성을 확보하기 위한 수단이다.  


Figure . ID 중복체크

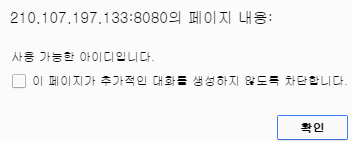
중복확인 후, 사용가능 한 ID인 경우 아래의 메시지로 사용할 수 있음을 알린다  


Figure . 사용가능 ID 확인

### 적용방법 및 Code

Hidden POST방식으로 중복 체크할 아이디를 join\_IDCheck.jsp로 전달하여 DB의 아이디를 확인하여 중복체크를 하고, 중복 결과를 호출한 부모페이지(Join\_form.jsp)의 함수를 호출하여 결과 전달하여 구현하였다.

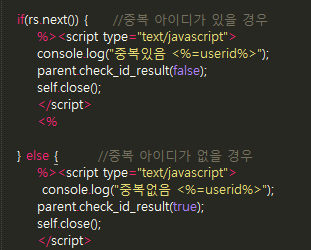
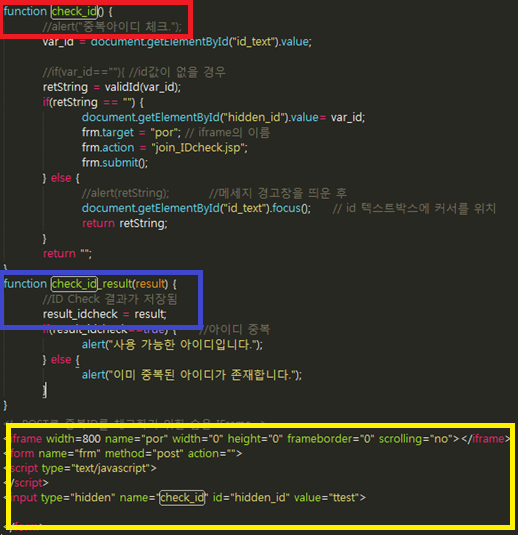


Figure . ID 중복체크 \_Join\_form.jsp Figure . ID 중복체크\_ Join\_IDCheck.jsp

## Password 입력 시, \*로 표시

### 내용

로그인 화면에서 사용자 Password 입력 시, \*로 표시하여 비밀번호 노출되지 않도록 구현하였다.

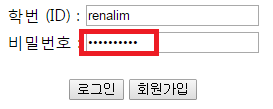


Figure . Password 입력 시, \*로 표시

### 적용방법 및 Code

Input type을 password 로 하여 비밀번호를 보호하였다.

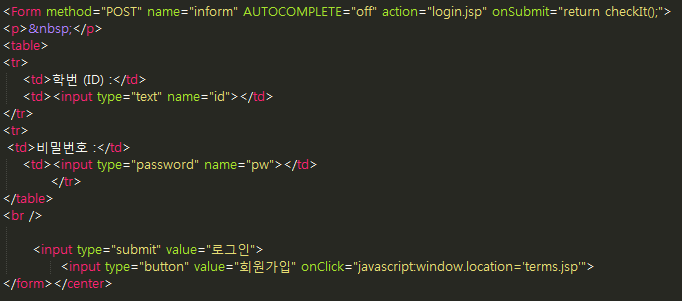


Figure . Password 입력 시, \*로 표시\_ Code

## Password 30일 마다 변경 알림

### 내용

사용자 비밀번호를 주기적으로 변경하도록 함으로써, 비밀번호의 노출에서 조금 더 안전하도록 구현하였다. 사용자가 비밀번호 변경 후 30일이 지나면, 메시지 창을 통하여 이를 알리고 비밀번호를 변경하도록 하였다.

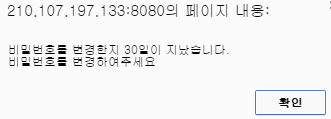


Figure . Password 30일 마다 변경 알림

### 적용방법 및 Code

Database에 비밀번호 변경일자를 저장하고 30일 후 알림 메시지 창을 노출한다.

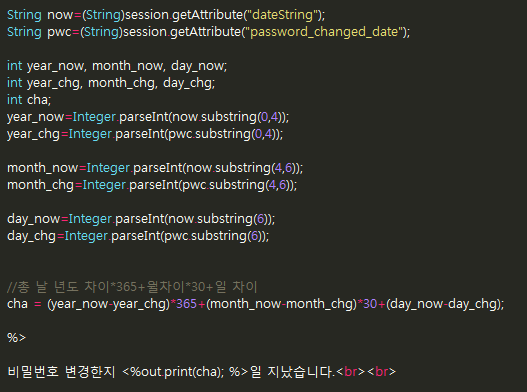


Figure . Password 30일 마다 변경 알림\_Code

## Password 5회 잘못 입력 시 접근제한

### 내용

사용자가 Password 입력 시, Password를 입력하여 5회 이상 잘못된 password를 입력하면, 접근을 차단하고 관리자에게 재 인증을 받도록 구현하였다.



Figure . 1.1 Password 입력 5회 잘못 입력 시 접근제한

### 적용방법 및 Code

로그인 실패 카운터 (fail\_flag)를 DB에 갱신하여 5회 이상 실패 시, 메시지 출력한다.

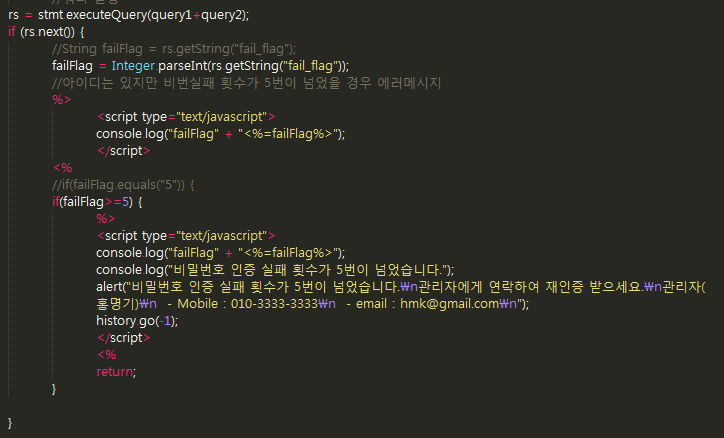


Figure . Password 입력 5회 잘못 입력 시 접근제한\_Code

## SQL Injection

### 내용

ID, Password 입력 창에서 query 문 입력을 통제하여 구현하였다. SQL Injection 구현 전, 회원로그인 창에서 ID에 ' or 1=1 # 입력하면, 아래 그림처럼, 로그인이 되었다.



Figure 16. SQL Injection 구현 전

보안상 취약점이 발견되어 SQL Injection 구현 후, 로그인이 되지 않고 error message를 출력함으로써 보안취약 사항을 개선하였다.

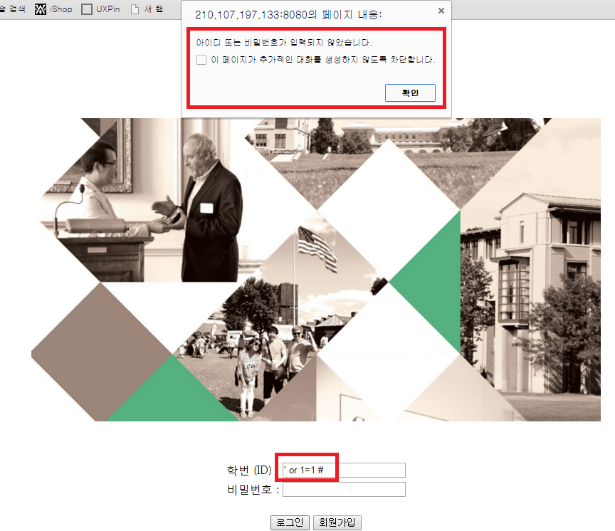


Figure 17. SQL Injection 구현 후

### 적용방법 및 Code

SQL Injection은 정규식을 이용하여 가용된 문자 이외에는 사용하지 못하도록 구현하였다. 실제 구현된 Code는 아래와 같다.



Figure 18. SQL Injection Code\_1

Submit 시 Figure8의 CheckIt() 함수에서 정규식을 이용하여 입력 데이터를 filtering 한다.

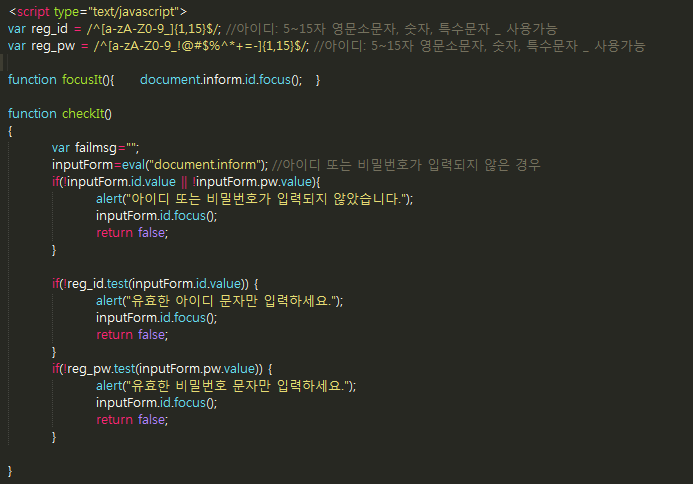


Figure 19. SQL Injection Code\_1

## DB 암호화

### 내용

Database 접근에 AES256, SHA256를 사용하여 암호화 및 복호화 알고리즘을 적용하였다.

해당 내용은 OWASP10의 [A2. 인증 및 세션관리 취약점]과 [A6. 민감 데이터 노출] 부분에서 적용하였으므로 자세한 내용은 해당 섹션에서 살펴보기로 한다.

### 적용방법 및 Code

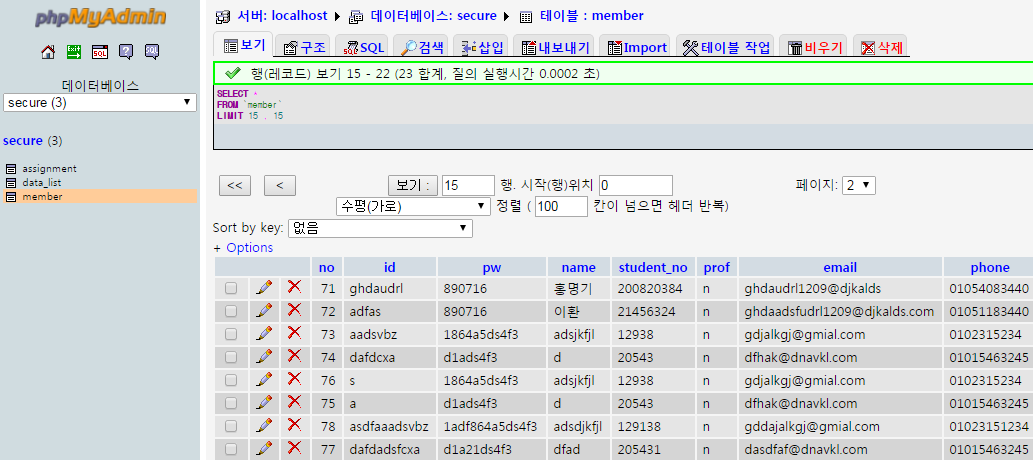


Figure . DB 암호화 전

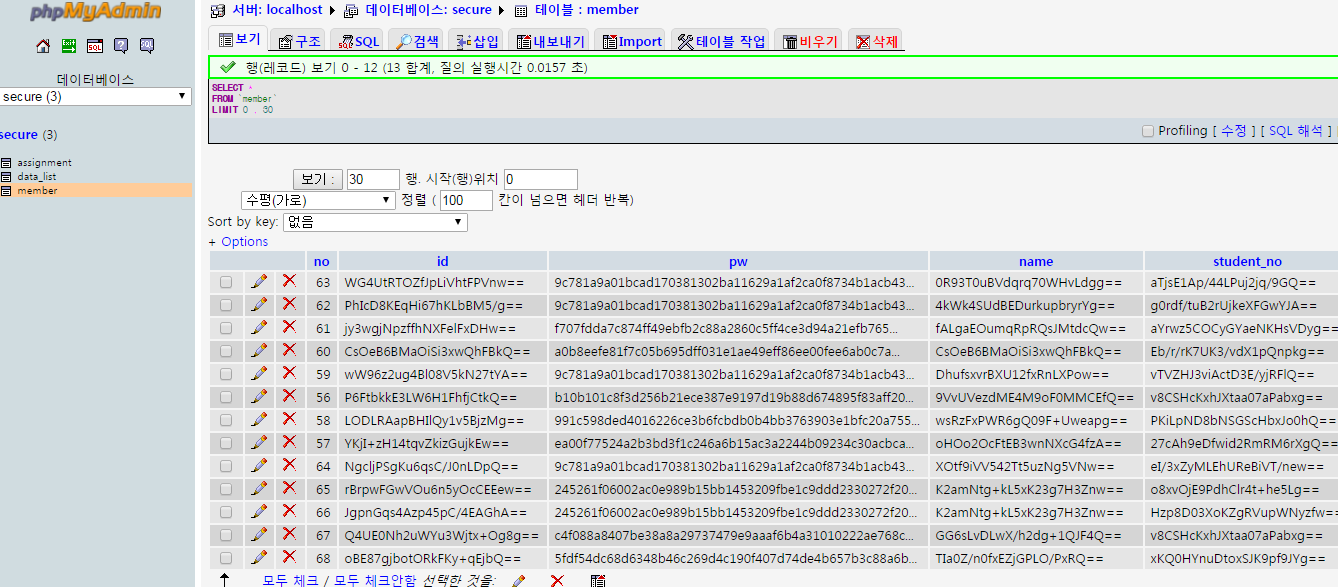


Figure . DG 암호화 적용 후

## 인증 및 세션관리 취약점

### 내용

사용자가 로그인, 회원가입 시 정보를 암호화 한다. (SSL: Secure Socket Layer)

### 적용방법 및 Code

로그인, 회원 가입 시 ID 는 256bit의 키(AES256) Password는 160bit(SHA256)의 Hash로 암호화하여 사용하였다.

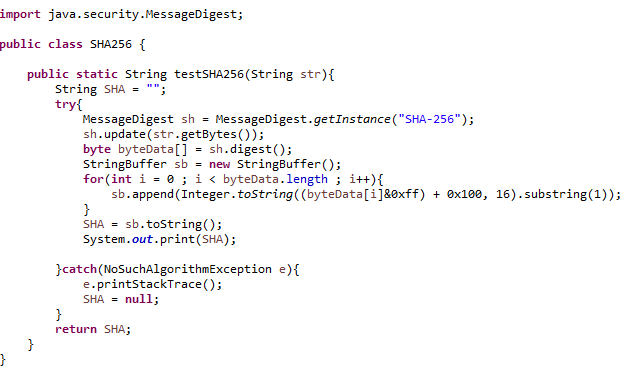


Figure . SHA256 적용

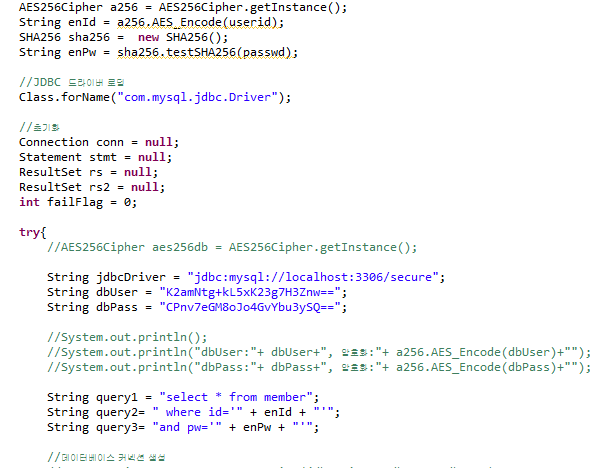


Figure . SHA256 적용

SHA256은 Hash 암호 알고리즘이며 파일 값이 약간만 바뀌어도 값이 천차만별로 변동한다. 복호화가 불가능하여 주로 패스워드 같은 인증관련 값에 적용한다.

## Cross-Site Scripting, Cross-Site Request Forgery

### 내용

1. Cross-Site Scripting, Cross-Site Request Forgery 취약점을 개선하기 위해, 스크립트를 입력할 수 있는 게시판에서 태그실행이 되지 않고, 스크립트 자체가 텍스트로 출력되도록 처리하였다.  
   게시판에서 제목, 본문에 각각 Script를 삽입하면,

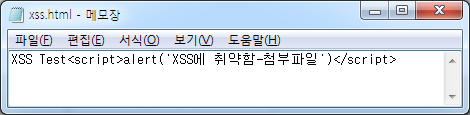


Figure 24. XSS, CSRF Test Script

Script가 실행되어 취약함을 확인할 수 있다.

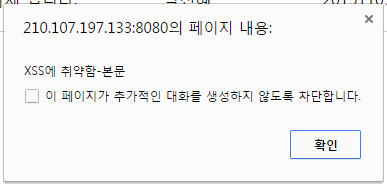
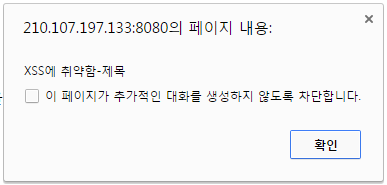


Figure 25. XSS, CSRF – 구현 전



Figure 26. XSS, CSRF – 구현 후

1. 게시판 file upload 시 upload 파일의 형식을 규제하여 위험한 파일의 실행을 차단하였다. 파일 확장자가 .zip이 아닌 경우에는 경고 메시지를 출력하고 파일을 전송하지 못하도록 하였다.  
   본 취약점 수정 전, 게시판에 figure5의 스크립트가 추가된 xss.html 파일을 업로드하고 파일을 다운로드 받을 경우, 해당 파일이 실행되어 보안에 취약점이 드러났다.

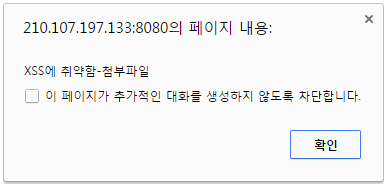


Figure 27. XSS, CSRF - 파일형식 규제-구현 전

아래와 같이 취약점을 개선하였다.

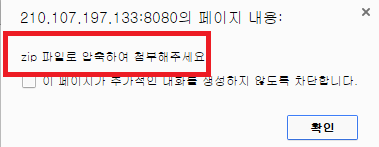
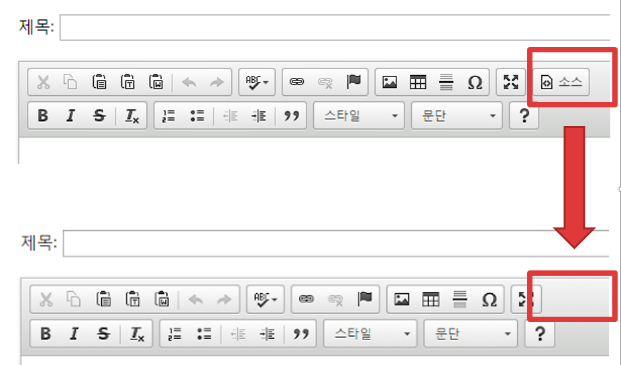


Figure 28. XSS, CSRF - 파일형식 규제-구현 후

### 적용방법 및 Code

1. 입력한 데이터 Filtering (스크립트 실행 금지) 부분은 게시판에서 CKEDITOR의 소스버튼을 제거하여 적용하였다.



* ckeditor의 config.js에 코드 추가 :

|  |
| --- |
| config.removeButtons = 'Source'; |

* write\_index.jsp의 코드에 config.js파일이 반영되도록 수정

|  |
| --- |
| CKEDITOR.replace(‘text\_content’,{toolbar: ‘’}); |

1. 게시판에서 입력하는 값을 검증 및 치환하는 부분은, 아래의 방법으로 구현 하였다.

* 게시판 제목에도 스크립트 삽입이 가능한 것을 방지
* - <. >, ', " 등 xss에 사용될 수 있는 특수문자 &lt 등으로 치환
* 제목에 입력된 <, > 등의 특수 문자가 특수문자의 기능을 발휘하지 못하고 텍스트로만 기능하도록 함

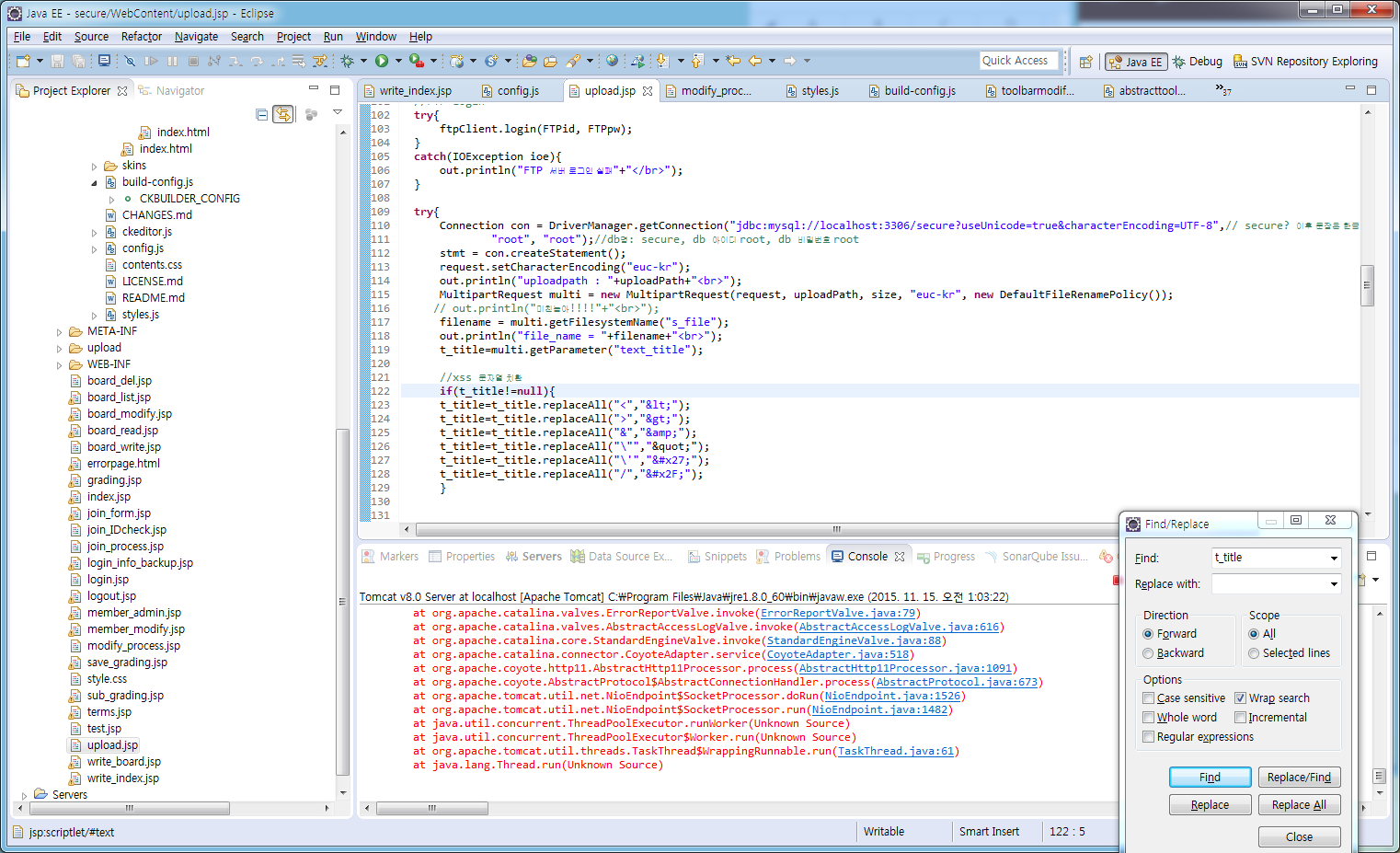


Figure . 게시판 스크립트 입력방지

1. 게시판에서 File upload 시 upload file의 형식을 규제하는 부분은 아래의 방법으로 구현 하였다.

* 첨부파일을 zip파일로 압축하여 업로드 하게 하여 웹 브라우저에서 코드 실행 방지(form submit시 파일명 체크)

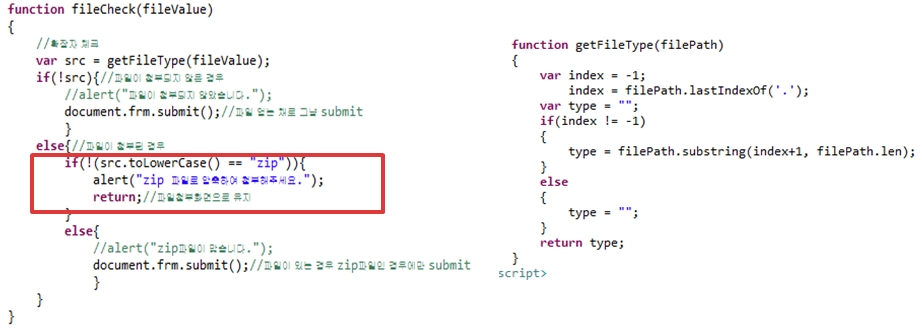


Figure . 게시판 업로드 파일 형식규제\_Code



## 취약한 직접 객체 참조

### 내용

웹 페이지 주소를 강제로 입력하여, 접근하는 취약점을 보완하기 위하여 웹 페이지 주소 강제 입력 시 Session을 체크하여 권한을 확인하여 주소입력을 통한 접속을 차단하였다. 로그인 하지 않은 상태로 웹 페이지 주소 입력 시, 아래와 같이 에러 메시지를 출력한다.

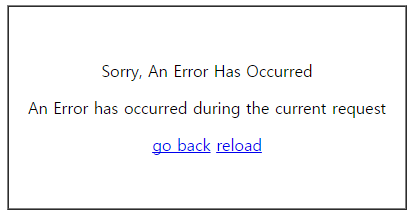


Figure . 주소입력 강제접근 에러 메시지.

### 적용방법 및 Code

userid 세션 & assignment\_no 없으면, index.jsp로 강제 리다이렉트 하거나 error message 출력하고, 페이지 별 특성에 따라 아래 코드로 구현하였다.

1. Join\_form.jsp : 가입시 에만 필요한 약관 열람 부분이므로 로그인 중에 진입 금지

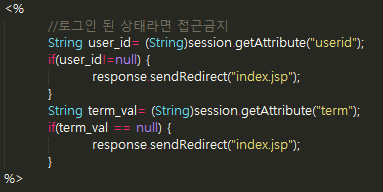
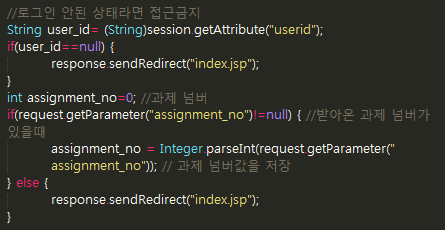


Figure . 주소입력 강제접근\_Code

1. 로그인이 필요한 페이지 : **로그인 세션**을 체크  
   파라미터가 필요한 페이지 : **필요 파라미터를 체크**하여 파라미터가 없다면 진입 금지 파라미터가 없다면 진입 금지 🡺 파라미터를 POST방식으로 보내므로 URL 강제 연결 방지



## 민감 데이터 노출

### 내용

과제정보 시스템에서 중요한 data를 암호화 하였다.

### 적용방법 및 Code

1. AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 암호화

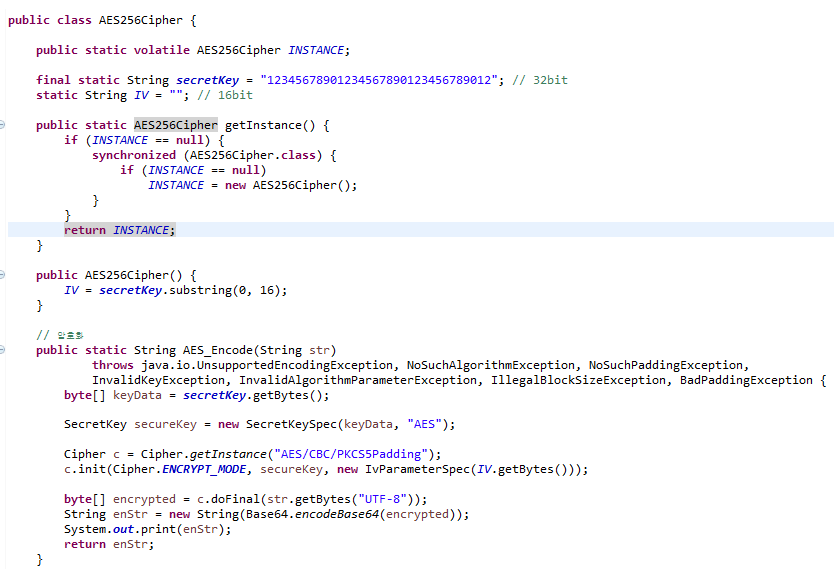


Figure . AES256 암호화



Figure . AES 256 암호화

1. AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 복호화 하였다.

중요정보를 암호화 하여 DB에 저장하니, 관리자의 “회원정보 관리” 페이지에서 암호화된 키로 표시가 되었다.



Figure . AES256 복호화

암호화 된 정보를 웹 페이지에서 인가된 사용자가 접근 시에는 복호화 하여 표시되도록 구현하였다.

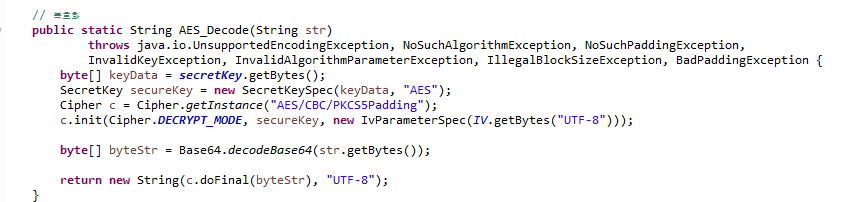


Figure . AES256 복호화\_Code



Figure . AES256 복호화

## 기능 수준의 접근통제

### 내용

과제 관리 시스템에의 사용자는 관리자, 교수, 학생으로 구분할 수 있으며 사용자에 따라 권한이 달라지도록 구현하였다.

1. 관리자에게만 보이는 회원 관리 버튼

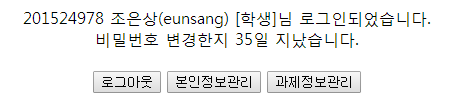
학생의 경우,   


Figure . 기능수준 접근통제\_학생

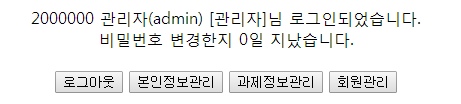
관리자의 경우,   


Figure . 기능수준 접근통제\_관리자

1. 같은 페이지라도 학생과 교수 권한에 따라 필요한 내용을 다르게 표시 함.

학생의 경우,   


Figure . 기능수준 접근통제\_학생

교수의 경우,



Figure . 기능수준 접근통제\_교수

# Static Analysis

## PMD

1. 2015년 11월 20일 PMD 분석결과

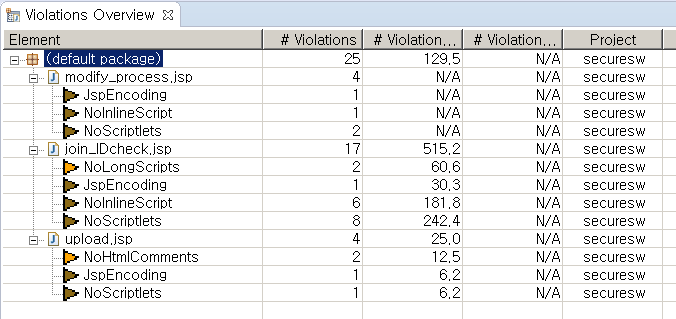


Figure . PMD 분석결과 - 2015.11.20

1. PMD rule 분석

* PMD In Security Rule.
* Secure Coding Guideline for Java SE(published by Oracle)
* MethodReturnInternalArray
* 코드 내부의 배열을 직접적으로 수정하거나 접근 X
* 복사본이나 쓰이는 변수를 캡슐화 시킬 것
* ArrayIsStoredDirectly
* 생성자나 배열을 받는 함수는 복사본을 인자로 받거나 저장해야 한다.
* 후에 바뀔 수 있는 코드에서 내부 에러 방지
* NoScriptlets
* Scriptlet은 태그 라이브러리나 JSP 선언을 이용하라는 룰
* Scripltet은 태그라이브러리가 태동한 이후 많은 단점을 양산함

|  |  |
| --- | --- |
| Resuability | reuse 불가능 |
| Replaceability | abstract로 만들 수 없음 |
| Debuggability | 만약 Exception이 발생시 얻을 수 있는 것은 빈 페이지 밖에 없음 |
| Testability | Unit 테스트 불가능 |
| Maintainability | 섞이거나 중복된 코드를 유지 보수하는데 시간이 걸림 |

* JSPEncoding
* JSP 문자 인코딩 🡪 JSP 파일 JAVA로 변경할 때 호환성을 위해 사용
* 같은 인코딩 설정 값으로 통일 charset=UTF-8 다른 언어와 호환성 유지
* NoInlineScript
* Inlining html 스크립트를 피하라는 룰
* 페이지 performance 관련됨 🡪 중복된 스크립트 정보 로드로 인한 오버헤드
* NolongScript
* PMD 3.6에서 정의된 Rule
* HTML Script Line이 10줄 이상은 되어야 함.
* 코드 가독성 관련 이슈
* NoHtmlComment
* 코드 유지 보수를 위한 가독성 관련 이슈

1. 분석 결과 수정사항

|  |  |
| --- | --- |
| NoLongScripts | Script 줄 사이의 간격을 최소화함. |
| JspEncoding | 태그에 charset=UTF-8 요소 추가 |
| NoInlineScript | Jsp 내부에 script를 넣지 않을 경우 페이지 구현에 차질이 생기므로 Rule 무시 |
| NoScriptlets | <% %> 🡪 <jsp:scriptlet> </jsp:scriptlet> |

1. 2015년 12월 10일 PMD 분석결과

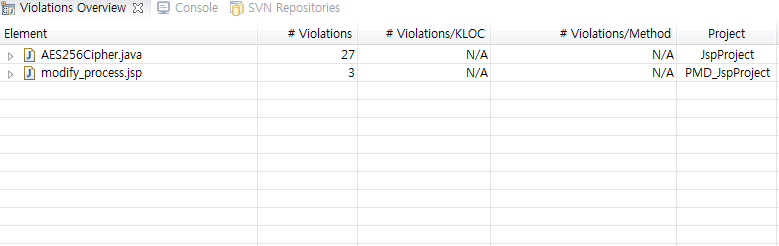


Figure . PMD 분석결과 - 2015.12.10

## Acunetix

1. 2015년 11월 24일 결과

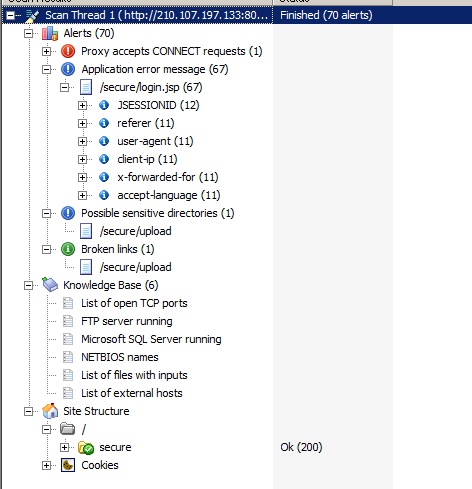


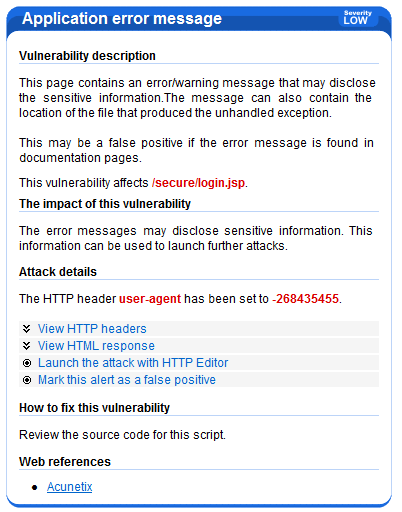
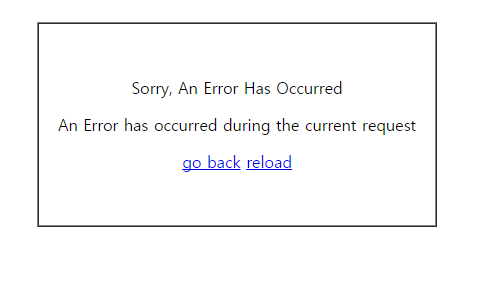
Figure . Acunetix 분석결과\_2015.11.24

1. 분석결과 수정사항

|  |  |
| --- | --- |
| Log | How to Fix |
| Proxy Accept CONNECT request | 80번 포트를 사용 중이었던 apache 서비스를 stop시켜 해결 |
| Application error message | 코드 문법적, 논리적 에러 존재 시, 기존 Apache 에러 디폴트 메시지를 띄우지 않고 에러 페이지로 출력 하게 함. |
| Possible sensitive directories | 코드 문법적 논리적 에러 시, 기존 Apache 에러 디폴트 메시지를 띄우지 않고 에러 페이지로 출력 하게 함. |
| Broken links | 인자가 넘어와야 access 가능한 페이지라서 acunetix 에서 출력시킴 |

Figure . Acunetix 분석 \_ 수정사항

1. 에러에 관한 보고사항



1. 2015년 12월 10일 결과

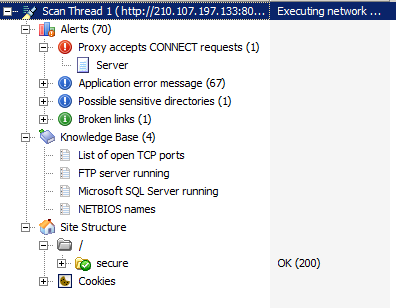


Figure . Acunetix 분석결과 \_ 2015.12.10

# Project Summary

과제관리 시스템에는 기본 보안요구사항, OWASP 10 그리고 S/W 개발보안 사항 47개를 Mapping하여 31개의 보안사항이 적용되었음을 확인 하였다.

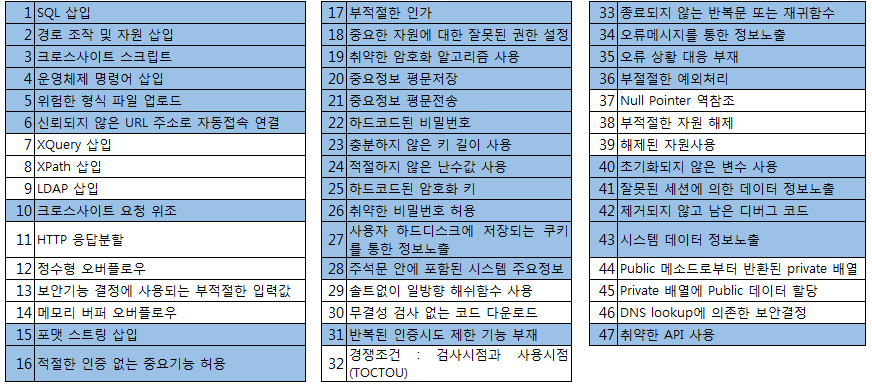


Figure . S/W 개발보안 47

## Lessons Learns

* 웹 보안에 대한 경각심 고취
* Secure Coding Experience
* Vulnerabilities Analysis, design, development
* Static Analysis
* Mitigation of identified risks

## Work breakdown Structure

2015년 9월부터 12월까지 약 12주간의 기간 동안 WBS로 일정을 계획, 조정, 구현하였다.



Figure . WBS