

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
Figure & Table	4
1. Introduction	7
1.1 Purpose	7
1.2 Specification.....	7
1.3 Function	8
1.4 Site Map	8
1.5 Database Structure	10
2. 보안 요구사항	11
2.1 기본 보안설정	11
2.2 OWASP 10, S/W 개발보안 47.....	12
3. Secure Coding	14
3.1 회원가입필드, 유효한 Value 만 입력하도록 적용	14
3.1.1 내용	14
3.1.2 적용방법 및 Code	15
3.2 ID 중복체크.....	16
3.2.1 내용	16
3.2.2 적용방법 및 Code	16
3.3 Password 입력 시, *로 표시	17
3.3.1 내용	17
3.3.2 적용방법 및 Code	17
3.4 Password 30일 마다 변경 알림	18
3.4.1 내용	18
3.4.2 적용방법 및 Code	18

3.5	Password 5회 잘못 입력 시 접근제한	19
3.5.1	내용	19
3.5.2	적용방법 및 Code	19
3.6	SQL Injection	20
3.6.1	내용	20
3.6.2	적용방법 및 Code	21
3.7	DB 암호화	22
3.7.1	내용	22
3.7.2	적용방법 및 Code	22
3.8	인증 및 세션관리 취약점	23
3.8.1	내용	23
3.8.2	적용방법 및 Code	23
3.9	Cross-Site Scripting, Cross-Site Request Forgery	24
3.9.1	내용	24
3.9.2	적용방법 및 Code	25
3.10	취약한 직접 객체 참조	27
3.10.1	내용	27
3.10.2	적용방법 및 Code	27
3.11	민감 데이터 노출	28
3.11.1	내용	28
3.11.2	적용방법 및 Code	28
3.12	기능 수준의 접근통제	30
3.12.1	내용	30
4.	Static Analysis	32
4.1	PMD	32
4.2	Acunetix	35

5	Project Summary.....	37
5.1	Lessons Learns.....	37
5.2	Work breakdown Structure.....	38

Figure & Table

Figure 1. 과제관리 시스템 Specification	7
Figure 2. Sitemap.....	8
Figure 3. 회원가입 입력사항	14
Figure 4. 회원가입 유효 값 체크 에러메시지	14
Figure 5. 회원가입 유효 값 체크 Code.....	15
Figure 6. ID 중복체크	16
Figure 7. 사용가능 ID 확인.....	16
Figure 8. ID 중복체크 _Join_form.jsp	Figure 9. ID 중복체크 _Join_IDCheck.jsp.....16
Figure 10. Password 입력 시, *로 표시	17
Figure 11. Password 입력 시, *로 표시_ Code	17
Figure 12. Password 30일 마다 변경 알림	18
Figure 13. Password 30일 마다 변경 알림_Code	18
Figure 14. 1.1 Password 입력 5회 잘못 입력 시 접근제한	19
Figure 15. Password 입력 5회 잘못 입력 시 접근제한_Code.....	19
Figure 16. SQL Injection 구현 전.....	20
Figure 17. SQL Injection 구현 후.....	20
Figure 18. SQL Injection Code_1	21
Figure 19. SQL Injection Code_1	21
Figure 20. DB 암호화 전	22
Figure 21. DG 암호화 적용 후.....	22
Figure 22. SHA256 적용	23

Figure 23. SHA256 적용	23
Figure 24. XSS, CSRF Test Script.....	24
Figure 25. XSS, CSRF – 구현 전.....	24
Figure 26. XSS, CSRF – 구현 후.....	24
Figure 27. XSS, CSRF - 파일형식 규제-구현 전.....	25
Figure 28. XSS, CSRF - 파일형식 규제-구현 후.....	25
Figure 29. 게시판 스크립트 입력방지.....	26
Figure 30. 게시판 업로드 파일 형식규제_Code.....	26
Figure 31. 주소입력 강제접근 에러 메시지.....	27
Figure 32. 주소입력 강제접근_Code.....	27
Figure 33. AES256 암호화	28
Figure 34. AES 256 암호화.....	28
Figure 35. AES256 복호화	29
Figure 36. AES256 복호화_Code	29
Figure 37. AES256 복호화	29
Figure 38. 기능수준 접근통제_학생.....	30
Figure 39. 기능수준 접근통제_관리자	30
Figure 40. 기능수준 접근통제_학생.....	30
Figure 41. 기능수준 접근통제_교수.....	31
Figure 42. PMD 분석결과 - 2015.11.20.....	32
Figure 43. PMD 분석결과 - 2015.12.10.....	34
Figure 44. Acunetix 분석결과_2015.11.24.....	35
Figure 45. Acunetix 분석 _ 수정사항	35
Figure 46. Acunetix 분석결과 _ 2015.12.10.....	36
Figure 47. S/W 개발보안 47	37
Figure 48. WBS.....	38

Table 1. 과제관리 시스템 Page.....	7
Table 2. Site Map - Page 별 address	9
Table 3. Database_member	10
Table 4. Database_assignment.....	10
Table 5. Databse_data_list	10
Table 6. 기본보안 적용내용	12
Table 7. OWASP 10 적용내용	13

1. Introduction

1.1 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.

1.2 Specification

[과제 관리 시스템]은 Web service를 하는 시스템으로 Client에게 Web browser를 통하여 서비스한다. 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

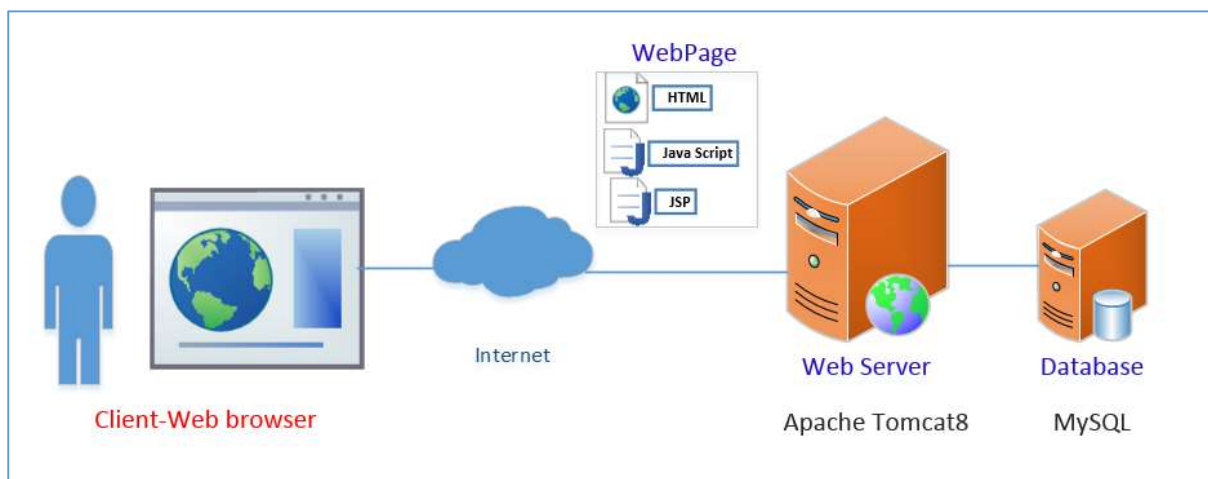


Figure 1. 과제관리 시스템 Specification

1.3 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)

1.4 Site Map

[과제 관리 시스템]의 Site Map은 아래와 같다.

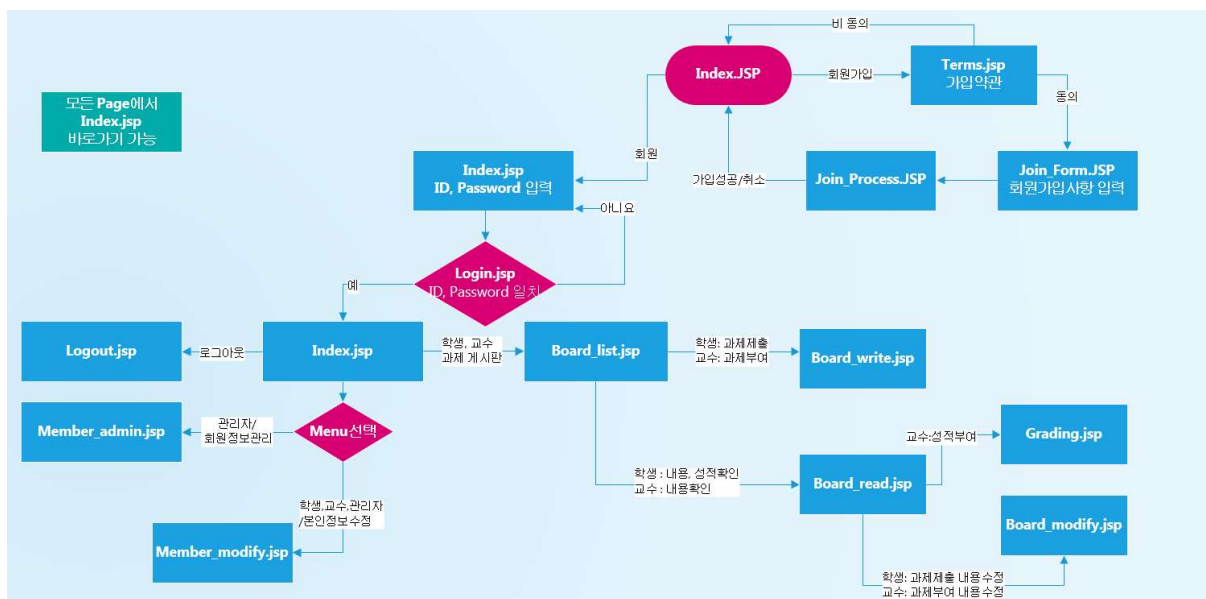


Figure 2. Sitemap

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
과제 목록	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

1.5 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

2. 보안 요구사항

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

2.1 기본 보안설정

내 용	적용사항
회원가입 필드, 유효한 value만 입력	회원가입 시, 사용자 입력 value를 유효한 value만 입력 하도록 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] [하드코딩된 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

2.2 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 <i>[오류 메시지를 통합 정보노출-S/W47]</i> <i>[오류상황대응부재 - S/W47]</i> <i>[부적절한 예외처리 - S/W47]</i>	error.html로 이동하도록 조치. web.xml에 코드 삽입 Sorry, An Error Has Occurred An Error has occurred during the current request go back reload
A2. 인증 및 세션관리 취약점	로그인 과정 암호화(SSL: Secure Socket Layer)	로그인이나 회원가입시 ID는 256bit의 키 (AES256) Password는 160bit(SHA256)의 Hash로 암호화하여 사용 (암호화, 복호화)
<i>[잘못된 세션에 의한 정보노출 - S/W47]</i>	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 <i>[크로스사이트 스크립팅, 신뢰되지 않은 URL 주소로 자연접속 연결, 크로스사이트 요청 위조-S/W47]</i>	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 < 등으로 치환 => 태그실행이 되지 않고, 스크립트 자체가 텍스트로 출력.
	게시판 file upload 시 upload file 형식규제 <i>[위험한 형식 파일 업로드 - S/W47]</i>	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 체크 하여 권한 확인, 주소 입력 접속차 단 <i>[경로조작 및 자원삽입 -S/W47]</i>	userid 세션 없고 , assignment_no 없으면 index.jsp로 강제 리다이렉트 or error message 출력
A6. 민감 데이터노출	중요 Data 암호화 <i>[중요정보 평문저장, 중요정보 평문전송, 충 분하지 않은 키 길이 사용, 적절하지 않은 난수값 사용, 하드코드 된 암호화 키- S/W47]</i>	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 <i>[적절한 인증 없는 중요기능 허용 -S/W47] [부적절한 인가 - S/W47] [중요한 자원에 대 한 잘못된 권한 설정-S/W47]</i>	관리자, 교수, 학생 권한으로 나뉘어 해당하 는 기능만 사용할 수 있도록 함

Table 7. OWASP 10 적용내용

3. Secure Coding

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

3.1 회원가입필드, 유효한 Value 만 입력하도록 적용

3.1.1 내용

과제관리 시스템은 회원가입 시, 아래의 항목을 입력하도록 되어있다.

회원가입

별표시(*)는 필수 입력사항입니다.

아이디 *	<input type="text"/>	중복확인	[영문/숫자 5자이상 15자이하입니다]
성명 *	<input type="text"/>		
학번 *	<input type="text"/>		
비밀번호 *	<input type="password"/>		[영문+숫자+특수문자 5자 이상 ~ 20자 이하입니다]
비밀번호 확인 *	<input type="password"/>		
이메일 *	<input type="text"/>		
휴대폰 *	<input type="text"/> - <input type="text"/> - <input type="text"/>		

Figure 3. 회원가입 입력사항

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

비밀번호: 5~20자 영문대소문자, 숫자, 특수문자 혼합하여 사용
 학번: 숫자 4~10자 사용가능.
 이메일: 유효한 이메일주소를 넣어주세요..
 전화번호: 유효한 전화번호를 넣어주세요.

Figure 4. 회원가입 유효 값 체크 에러메시지

3.1.2 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}$/;      //아이디: 5~15 자 영문소문자, 숫자, 특수문자 _ 사용가능
var reg_pw = /^(?=.*[a-zA-Z])(?=.*[!@#$%^*+=-]).*[0-9]).{5,20}$/;
                                     //비밀번호: 5~20 자 영문대소문자, 숫자, 특수문자 혼합하여 사용
var reg_number = /^[0-9]{4,10}$/;      //학번: 숫자 4~10 자 사용가능.
var reg_email = /^[0-9a-zA-Z_@-]+\@[0-9a-zA-Z_-]+(\.[0-9a-zA-Z_-]+)*$/;
                                     //이메일: 유효한 이메일 주소를 넣어주세요.
var reg_phone = /^Wd{2,3}W-Wd{3,4}W-Wd{4}$/;
                                     //전화번호: 유효한 전화번호를 넣어주세요.
```

아래 그림은 실제로 구현된 Code를 보여준다.

```
function validid(field) {
    if(reg_id.test(field)) {
        return "아이디: 5~15자 영문소문자, 숫자, 특수문자 _ 사용가능\nWr";
    }
    return "";
}

function validpw(field1, field2) {
    if(field1 != field2) { //비밀번호 입력한 내용과 비밀번호chk에 입력한 내용이 같은지 체크
        return "비밀번호와 비밀번호 확인의 내용이 다릅니다\nWr";
    } else if(reg_pw.test(field1)) {
        return "비밀번호: 5~20자 영문대소문자, 숫자, 특수문자 혼합하여 사용\nWr";
    }
    return "";
}

function validnumber(field) {
    if(reg_number.test(field)) {
        return "학번: 숫자 4~10자 사용가능\nWr";
    }
    return "";
}

function validemail(field) {
    if(reg_email.test(field)) {
        return "이메일: 유효한 이메일주소를 넣어주세요.\nWr";
    }
    return "";
}

function validphone(field1, field2, field3) {
    if(reg_phone.test(field1+'-'+field2+'-'+field3)) {
        return "전화번호: 유효한 전화번호를 넣어주세요.\nWr";
    }
    return "";
}
```

Figure 5. 회원가입 유효 값 체크 Code

3.2 ID 중복체크

3.2.1 내용

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

회원가입

Figure 6. ID 중복체크

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

Figure 7. 사용가능 ID 확인

3.2.2 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.

```
function check_id() {
    //alert("중복아이디 체크");
    var id = document.getElementById("id_text").value;

    //if(var_id=="") //id값이 없을 경우
    retString = validId(var_id);
    if(retString == "") {
        document.getElementById("hidden_id").value = var_id;
        frm.target = "por"; // iframe의 이름
        frm.action = "join_IDcheck.jsp";
        frm.submit();
    } else {
        //alert(retString); //메세지 경고창을 띄운 후
        document.getElementById("id_text").focus(); // id 텍스트박스에 커서를 위치
        return retString;
    }
    return "";
}

function check_id_result(result) {
    //ID Check 결과가 저장됨
    result_idcheck = result;
    if(result_idcheck == true) { //아이디 중복
        alert("사용 가능한 아이디입니다.");
    } else {
        alert("이미 중복된 아이디가 존재합니다.");
    }
}

<frame width=800 name="por" width="0" height="0" frameborder="0" scrolling="no"></frame>
<form name="frm" method="post" action="">
<script type="text/javascript">
</script>
<input type="hidden" name="check_id" id="hidden_id" value="ttest">
```

Figure 8. ID 중복체크 _Join_form.jsp

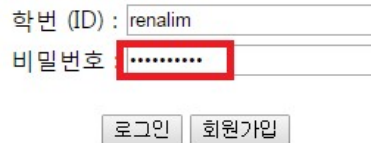
```
if(rs.next()) { //중복 아이디가 있을 경우
    %><script type="text/javascript">
    console.log("중복있음 <%=userid%>");
    parent.check_id_result(false);
    self.close();
    </script>
    %>
} else { //중복 아이디가 없을 경우
    %><script type="text/javascript">
    console.log("중복없음 <%=userid%>");
    parent.check_id_result(true);
    self.close();
    </script>
    %>
```

Figure 9. ID 중복체크 _Join_IDCheck.jsp

3.3 Password 입력 시, *로 표시

3.3.1 내용

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



학번 (ID) :

비밀번호 :

Figure 10. Password 입력 시, *로 표시

3.3.2 적용방법 및 Code

Input type 을 password 로 하여 비밀번호를 보호하였다.

```
<Form method="POST" name="inform" AUTOCOMPLETE="off" action="login.jsp" or
<p>&nbsp;</p>
<table>
<tr>
<td>학번 (ID) :</td>
<td><input type="text" name="id"></td>
</tr>
<tr>
<td>비밀번호 :</td>
<td><input type="password" name="pw"></td>
</tr>
</table>
<br />
<input type="submit" value="로그인">
<input type="button" value="회원가입" onClick="javascript:window.locatio
</form></center>
```

Figure 11. Password 입력 시, *로 표시_ Code

3.4 Password 30일 마다 변경 알림

3.4.1 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.

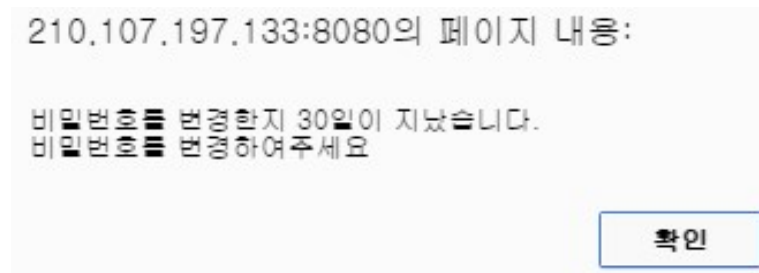


Figure 12. Password 30일 마다 변경 알림

3.4.2 적용방법 및 Code

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

```
String now=(String)session.getAttribute("dateString");
String pwc=(String)session.getAttribute("password_changed_date");

int year_now, month_now, day_now;
int year_chg, month_chg, day_chg;
int cha;
year_now=Integer.parseInt(now.substring(0,4));
year_chg=Integer.parseInt(pwc.substring(0,4));

month_now=Integer.parseInt(now.substring(4,6));
month_chg=Integer.parseInt(pwc.substring(4,6));

day_now=Integer.parseInt(now.substring(6));
day_chg=Integer.parseInt(pwc.substring(6));

//총 날 년도 차이*365+월차이*30+일 차이
cha = (year_now-year_chg)*365+(month_now-month_chg)*30+(day_now-day_chg);

%>

비밀번호 변경한지 <%out.print(cha); %>일 지났습니다.<br><br>
```

Figure 13. Password 30일 마다 변경 알림_Code

3.5 Password 5회 잘못 입력 시 접근제한

3.5.1 내용

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

비밀번호 인증 실패 횟수가 5번이 넘었습니다.
관리자에게 연락하여 재인증 받으세요.
관리자(홍명기)
- Mobile : 010-3333-3333
- email : hmk@gmail.com

Figure 14. 1.1 Password 입력 5회 잘못 입력 시 접근제한

3.5.2 적용방법 및 Code

로그인 실패 카운터 (fail_flag)를 DB 에 갱신하여 5 회 이상 실패 시, 메시지 출력한다.

```
rs = stmt.executeQuery(query1+query2);
if (rs.next()) {
    //String failFlag = rs.getString("fail_flag");
    failFlag = Integer.parseInt(rs.getString("fail_flag"));
    //아이디는 있지만 비밀번호 실패 횟수가 5번이 넘었을 경우 에러메시지
    %>
    <script type="text/javascript">
        console.log("failFlag" + "<%=failFlag%>");
    </script>

    <%
    //if(failFlag.equals("5")) {
    if(failFlag>=5) {
        %>
        <script type="text/javascript">
            console.log("failFlag" + "<%=failFlag%>");
            console.log("비밀번호 인증 실패 횟수가 5번이 넘었습니다.");
            alert("비밀번호 인증 실패 횟수가 5번이 넘었습니다.\n관리자에게 연락하여 재인증 받으세요.\n관리자(
홍명기)\n - Mobile : 010-3333-3333\n - email : hmk@gmail.com\n");
            history.go(-1);
        </script>
        <%
        return;
    }
}
```

Figure 15. Password 입력 5회 잘못 입력 시 접근제한_Code

3.6 SQL Injection

3.6.1 내용

ID, Password 입력 창에서 query 문 입력을 통제하여 구현하였다. SQL Injection 구현 전, 회원 로그인 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

3.6.2 적용방법 및 Code

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

```
<center><body onLoad="focusIt()">

  <Form method="POST" name="inform" AUTOCOMPLETE="off" action="login.jsp" onSubmit="return checkIt();">
  <p>&nbsp;</p>
  <table>
  <tr>
    <td>학번 (ID) :</td>
    <td><input type="text" name="id"></td>
  </tr>
  <tr>
    <td>비밀번호 :</td>
    <td><input type="password" name="pw"></td>
  </tr>
  </table>
  <br />

  <input type="submit" value="로그인">
  <input type="button" value="회원가입" onClick="javascript:window.location='terms.jsp'">
</form></center>
```

Figure 18. SQL Injection Code_1

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

```
<script type="text/javascript">
var reg_id = /^[a-zA-Z0-9_]{1,15}$/; //아이디: 5~15자 영문소문자, 숫자, 특수문자 _ 사용가능
var reg_pw = /^[a-zA-Z0-9_!@#%$^*+=-]{1,15}$/; //아이디: 5~15자 영문소문자, 숫자, 특수문자 _ 사용가능

function focusIt(){ document.inform.id.focus(); }

function checkIt()
{
  var failmsg="";
  inputForm=eval("document.inform"); //아이디 또는 비밀번호가 입력되지 않은 경우
  if(!inputForm.id.value || !inputForm.pw.value){
    alert("아이디 또는 비밀번호가 입력되지 않았습니다.");
    inputForm.id.focus();
    return false;
  }

  if(!reg_id.test(inputForm.id.value)) {
    alert("유효한 아이디 문자만 입력하세요.");
    inputForm.id.focus();
    return false;
  }

  if(!reg_pw.test(inputForm.pw.value)) {
    alert("유효한 비밀번호 문자만 입력하세요.");
    inputForm.id.focus();
    return false;
  }
}
```

Figure 19. SQL Injection Code_1

3.7 DB 암호화

3.7.1 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.

3.7.2 적용방법 및 Code

Figure 20 shows the phpMyAdmin interface for the 'secure' database, 'member' table. The table has 22 records. The columns are: no, id, pw, name, student_no, prof, email, and phone. The 'pw' column contains hashed passwords.

no	id	pw	name	student_no	prof	email	phone
71	ghdaudrl	890716	홍명기	200820384	n	ghdaudrl1209@djkalds	01054083440
72	adfas	890716	이환	21456324	n	ghdaadsfudrl1209@djkalds.com	01051183440
73	aadsvbz	1864a5ds4f3	adsjckfj	12938	n	gdjalkgj@gmail.com	0102315234
74	dafdcxa	d1ads4f3	d	20543	n	dfhak@dnavl.com	01015463245
76	s	1864a5ds4f3	adsjckfj	12938	n	gdjalkgj@gmail.com	0102315234
75	a	d1ads4f3	d	20543	n	dfhak@dnavl.com	01015463245
78	asdaaadsbvz	1adf864a5ds4f3	adsjckfj	129138	n	gddajalkgj@gmail.com	01023151234
77	dafdadscxa	d1a21ds4f3	dfad	205431	n	dasdfaf@dnavl.com	01015463245

Figure 20. DB 암호화 전

Figure 21 shows the phpMyAdmin interface for the 'secure' database, 'member' table after applying DG encryption. The table has 12 records. The columns are: no, id, pw, name, and student_no. The 'pw' column contains encrypted passwords.

no	id	pw	name	student_no
63	WG4UtrTOZfjpLhVFPVnw==	9c781a9a01bcad170381302ba11629a1af2ca0f8734b1ac343...	0R93T0uBVdqrq70WHVldgg==	aTjsE1Ap/44LPuj2jq/9GQ==
62	PhlCD8EqHi67hKLbBM5/g==	9c781a9a01bcad170381302ba11629a1af2ca0f8734b1ac343...	4kKw45UdBEDurkupbryrYg==	g0rdrf/tu2RjUjkeXGwYJA==
61	jy3wgiNpzffhNXFeLxHw==	f707fdda7c874ff49ebfb2c88a2860c5ff4ce3d94a21efb765...	fALgaEOumqRpRQJmtdcQw==	aYrvz5COCyGaeNKHsVDyg==
60	CsOeB6BMAOiSi3xwQhFbkQ==	a0b8eeef81f7c05b695df031e1ae49eff86ee00fee6ab0C7a...	CsOeB6BMAOiSi3xwQhFbkQ==	Eb/r/rK7UK3/vdXlpQnpkg==
59	wW96z2ug4BIO8V5KN27iYA==	9c781a9a01bcad170381302ba11629a1af2ca0f8734b1ac343...	DhufsvxrXU12fRnLXPow==	vTVZHj3ivAct3E/yjRFIq==
56	P6FtbkE3LW6H1fhfCkQ==	b10b101c8f3d256b21ece387e9197d19b88d674895f83aff20...	9vVUvezdME4M9oFOMMCEq==	v8CSHCckXhXtaa07aPabxg==
58	LODLRAapBHIQy1v5BjzMG==	991c598ded4016226ce3b6fcbdb0b4bb3763903e1bfc20a755...	wsRzFvPWRegQ09f+Uweapg==	PKlPND8bNSGScHbxlo0hQ==
57	YKjI+2H14tqvZkizGujKEw==	ea00f77524a2b3bd3f1c246a6b15ac3a2244b09234c30acba...	oHOo20cFIEB3wnNXcG4fzA==	27cAh9eDfwid2RmRM6XgQ==
64	NgclP5gKu6qCjJOnLdpQ==	9c781a9a01bcad170381302ba11629a1af2ca0f8734b1ac343...	XOth9Vv542T15uzNg5VnW==	el/3xZyMLEhUReBIVT/new==
65	RbrpwFGwVOu6n5yOCCEEw==	245261f06002ac0e989b15bb1453209fbc1c9ddd2330272f20...	K2amNtg+KL5kK23g7H3Znw==	o8xvOjE9PdhtClr4t+he5Lg==
66	JgnpGqs4Azp45pC/4EAghA==	245261f06002ac0e989b15bb1453209fbc1c9ddd2330272f20...	K2amNtg+KL5kK23g7H3Znw==	Hzp8D03XoKZgRVupWVnyzf==
67	Q4UEONh2uWYU3Wjtx+Og8g==	c4088a8407be38a8a29737479e9aaaf6b4a31010222ae768c...	GG6sLvDLwX/h2dg+1QJf4Q==	v8CSHCckXhXtaa07aPabxg==
68	oBE87gibotORKFky+qEjbQ==	5fdf54dc68d6348b46c269d4c190f407d74de4b67b3c88a6b...	TJaoZ/nOfEZjGpLO/PxRQ==	xK0HYNuDtox5JK9pf9Jvg==

Figure 21. DG 암호화 적용 후

3.8 인증 및 세션관리 취약점

3.8.1 내용

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

3.8.2 적용방법 및 Code

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

```
import java.security.MessageDigest;

public class SHA256 {

    public static String testSHA256(String str){
        String SHA = "";
        try{
            MessageDigest sh = MessageDigest.getInstance("SHA-256");
            sh.update(str.getBytes());
            byte byteData[] = sh.digest();
            StringBuffer sb = new StringBuffer();
            for(int i = 0 ; i < byteData.length ; i++){
                sb.append(Integer.toString((byteData[i]&0xff) + 0x100, 16).substring(1));
            }
            SHA = sb.toString();
            System.out.print(SHA);

        }catch(NoSuchAlgorithmException e){
            e.printStackTrace();
            SHA = null;
        }
        return SHA;
    }
}
```

Figure 22. SHA256 적용

```
AES256Cipher a256 = AES256Cipher.getInstance();
String enId = a256.AES_Encode(userid);
SHA256 sha256 = new SHA256();
String enPw = sha256.testSHA256(passwd);

//JDBC 드라이버 로드
Class.forName("com.mysql.jdbc.Driver");

//초기화
Connection conn = null;
Statement stmt = null;
ResultSet rs = null;
ResultSet rs2 = null;
int failFlag = 0;

try{
    //AES256Cipher aes256db = AES256Cipher.getInstance();

    String jdbcDriver = "jdbc:mysql://localhost:3306/secure";
    String dbUser = "K2amNtg+kL5xK23g7H3Znw==";
    String dbPass = "CPnv7eGM8oJo4GvYbu3ySQ=";
```

Figure 23. SHA256 적용

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

3.9 Cross-Site Scripting, Cross-Site Request Forgery

3.9.1 내용

- a. 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,

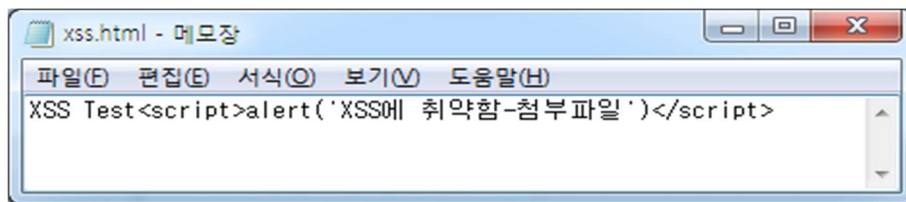


Figure 24. XSS, CSRF Test Script

Script가 실행되어 취약함을 확인할 수 있다.

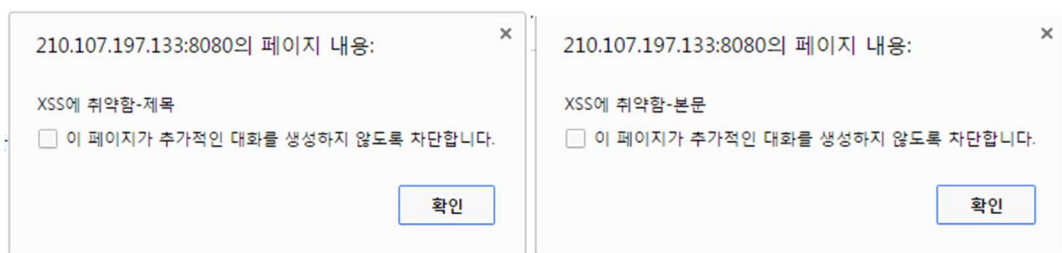


Figure 25. XSS, CSRF – 구현 전

제출한 과제			
No.	과제명	제출자	작성일
3	XSS Test<script>alert('xss 에 취약함 제 목')</script>	임학생	20151201
xss Test<script>alert('xss 에 취약함 본문')</script>			

Figure 26. XSS, CSRF – 구현 후

- b. 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.

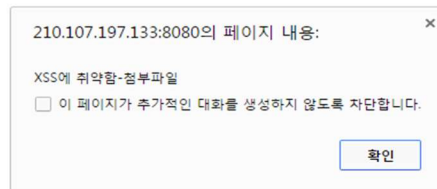


Figure 27. XSS, CSRF - 파일형식 규제-구현 전

아래와 같이 취약점을 개선하였다.

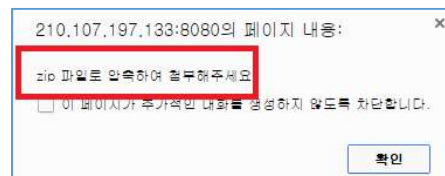
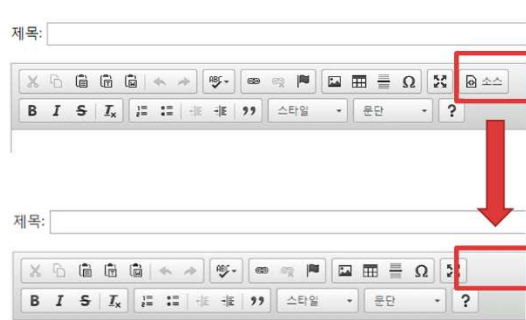


Figure 28. XSS, CSRF - 파일형식 규제-구현 후

3.9.2 적용방법 및 Code

- a. 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: ''});
```

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

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

```
//xss 문자열 치환
if(t_title!=null){
t_title=t_title.replaceAll("<","&lt;");
t_title=t_title.replaceAll(">","&gt;");
t_title=t_title.replaceAll("&","&amp;");
t_title=t_title.replaceAll(""","&quot;");
t_title=t_title.replaceAll("'","&#x27;");
t_title=t_title.replaceAll("/","&#x2F;");
}
```

Figure 29. 게시판 스크립트 입력방지

c. 게시판에서 File upload 시 upload file 의 형식을 규제하는 부분은 아래의 방법으로 구현 하였다.

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

```
function fileCheck(fileValue)
{
    //파일명 체크
    var src = getFileType(fileValue);
    if(!src){
        //alert("파일이 업로드되지 않았습니다.");
        document.frm.submit();
    }
    else{
        if(!src.toLowerCase() == "zip"){
            alert("zip 파일로 압축하여 업로드하세요.");
            return;
        }
        else{
            //alert("zip파일이 아닙니다.");
            document.frm.submit();
        }
    }
}

function getFileType(filePath)
{
    var index = -1;
    index = filePath.lastIndexOf('.');
    var type = "";
    if(index != -1)
    {
        type = filePath.substring(index+1, filePath.length);
    }
    else
    {
        type = "";
    }
    return type;
}
```

Figure 30. 게시판 업로드 파일 형식규제_Code

```
<td align="center"><input type="button" value="업로드" onclick="fileCheck(document.frm.s_file.value)"/>
<input type="button" value="뒤로가기" onclick="javascript:history.back()"/></td>
```

3.10 취약한 직접 객체 참조

3.10.1 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 31. 주소입력 강제접근 에러 메시지.

3.10.2 적용방법 및 Code

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

- a. Join_form.jsp : 가입시에만 필요한 약관 열람 부분이므로 로그인 중에 진입 금지

```
//로그인 된 상태라면 접근금지
String user_id= (String)session.getAttribute("userid");
if(user_id!=null) {
    response.sendRedirect("index.jsp");
}
String term_val= (String)session.getAttribute("term");
if(term_val == null) {
    response.sendRedirect("index.jsp");
}
```

Figure 32. 주소입력 강제접근_Code

- b. 로그인이 필요한 페이지 : 로그인 세션을 체크

파라미터가 필요한 페이지 : 필요 파라미터를 체크하여 파라미터가 없다면 진입 금지
파라미터가 없다면 진입 금지 → 파라미터를 POST방식으로 보내므로 URL 강제 연결 방지

```
//로그인 안된 상태라면 접근금지
String user_id= (String)session.getAttribute("userid");
if(user_id==null) {
    response.sendRedirect("index.jsp");
}
int assignment_no=0; //과제 넘버
if(request.getParameter("assignment_no")!=null) { //받은 과제 넘버가 있을때
    assignment_no = Integer.parseInt(request.getParameter("assignment_no")); // 과제 넘버값을 저장
} else {
    response.sendRedirect("index.jsp");
}
```

3.11 민감 데이터 노출

3.11.1 내용

과제정보 시스템에서 중요한 data를 암호화 하였다.

3.11.2 적용방법 및 Code

- a. AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 암호화

```
public class AES256Cipher {
    public static volatile AES256Cipher INSTANCE;

    final static String secretKey = "12345678901234567890123456789012"; // 32bit
    static String IV = ""; // 16bit

    public static AES256Cipher getInstance() {
        if (INSTANCE == null) {
            synchronized (AES256Cipher.class) {
                if (INSTANCE == null)
                    INSTANCE = new AES256Cipher();
            }
        }
        return INSTANCE;
    }

    public AES256Cipher() {
        IV = secretKey.substring(0, 16);
    }

    // 암호화
    public static String AES_Encode(String str)
        throws java.io.UnsupportedEncodingException, NoSuchAlgorithmException, NoSuchPaddingException,
        InvalidKeyException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException {
        byte[] keyData = secretKey.getBytes();

        SecretKey secureKey = new SecretKeySpec(keyData, "AES");

        Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
        c.init(Cipher.ENCRYPT_MODE, secureKey, new IvParameterSpec(IV.getBytes()));

        byte[] encrypted = c.doFinal(str.getBytes("UTF-8"));
        String enStr = new String(Base64.encodeBase64(encrypted));
        System.out.print(enStr);
        return enStr;
    }
}
```

Figure 33. AES256 암호화

```
String userid = request.getParameter("id_text"); //input의 name에서 받아옴
String name = request.getParameter("mem_name");
String student_no = request.getParameter("mem_number");
String passwd = request.getParameter("mem_pass");
String passwd_chk = request.getParameter("mem_passChk");
String email = request.getParameter("mem_email");
String phone = request.getParameter("mem_phone01")+"-"+request.getParameter("mem_phone02")+"-"+request.getParameter("mem_phone03");
String modify_date = request.getParameter("mem_modify_date"); //member_modify_date

AES256Cipher a256 = AES256Cipher.getInstance();
String enuserid = a256.AES_Encode(userid);
String enname = a256.AES_Encode(name);
String enstudent_no = a256.AES_Encode(student_no);

SHA256 sha256 = new SHA256();
String enpasswd = sha256.testSHA256(passwd);
String enpasswd_chk = sha256.testSHA256(passwd_chk);

String enemail = a256.AES_Encode(email);
String enphone = a256.AES_Encode(phone);

//현재 날짜를 String 형으로 출력 YYYYMMDD
java.util.Date dt = new java.util.Date();

SimpleDateFormat dateFormatter = new SimpleDateFormat("yyyyMMdd");
String dateString = dateFormatter.format(dt);

String register_date = dateString;
String password_changed_date = dateString;

String enregister_date = a256.AES_Encode(register_date);
String enpassword_changed_date = a256.AES_Encode(password_changed_date);
```

Figure 34. AES 256 암호화

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

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

회원정보 관리

순 번	아이디	이름	학번	신분	비고
62	PhIcD8EqHi67hKLbBMS/g==	4kWk4SUdEDurkupbryrYg==	g0rdf/tuB2rUjkeXFGwYJA==	<input type="button" value="학생"/>	e0/6e/anQHKEkNDX/OjaiCjFvTqSyI
61	jy3wgjNpzffhNXFelxDHw==	fALgaEOumqRpRQsIMtdcQw==	aYrwz5COCyGYaeNKHsVDyg==	<input type="button" value="교수"/>	YodOZ6ae3HCMb6Ef+n
60	CsOeB6BMaOiSi3xxQhFBkQ==	CsOeB6BMaOiSi3xxQhFBkQ==	Eb/r/rK7UK3/vdXlpQnpkg==	관리 자	QL4dMmzEi3sYYzKij+xt9v5yq/TZYI
59	wW96z2ug4BIO8V5kN27tYA==	DhufxvrBXU12fxRnLXPow==	vTVZHJ3viActD3E/yjRFIQ==	<input type="button" value="학생"/>	Tv9/yKMgCea00Iuk+FVaKdIMV/pp
56	P6FtbkkE3LWGh1FhfjCtkQ==	9VvUVezdME4M9oF0MMCEfQ==	v8CSHckKhJXtaa07aPabxg==	<input type="button" value="학생"/>	xrUMZx89d5ePYZb7uEi1vCEUyqLTe
58	LODLRAapBHllQy1v5BjzMg==	wsRzFxPWR6gQ09f+Uweapg==	PKilpND8bNSGScHbxJo0hQ==	<input type="button" value="학생"/>	Twr8xqhFnrBzE7oNuVI

Figure 35. AES256 복호화

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

```
// 복호화
public static String AES_Decode(String str)
    throws java.io.UnsupportedEncodingException, NoSuchAlgorithmException, NoSuchPaddingException,
        InvalidKeyException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException {
    byte[] keyData = secretKey.getBytes();
    SecretKey secureKey = new SecretKeySpec(keyData, "AES");
    Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
    c.init(Cipher.DECRYPT_MODE, secureKey, new IvParameterSpec(IV.getBytes("UTF-8")));

    byte[] byteStr = Base64.decodeBase64(str.getBytes());

    return new String(c.doFinal(byteStr), "UTF-8");
}
```

Figure 36. AES256 복호화_Code

```
String query1 = "select * from member";

//String query2= " where id='"+ userid + "' and pw='"+ passwd + "'";

//데이터베이스 연결
conn = DriverManager.getConnection(jdbcDriver, dbUser, dbPass);

//Statement 생성
stmt = conn.createStatement();

//쿼리 실행
rs = stmt.executeQuery(query1);

while (rs.next()) { //아이디와 비밀번호 일치하는 경우는 rs.next()는 ResultSet에서 select의 값이 존재하는 경우 true를 리턴하고, 존재하지 않는 경우에는 false를 리턴한다.
    String no = rs.getString("no");
    String id = rs.getString("id");
    String name = rs.getString("name");
    String student_no = rs.getString("student_no");
    String prof = rs.getString("prof");
    String email=rs.getString("email");
    String phone=rs.getString("phone");

    String desid = a256.AES_Decode(id);
    String desname= a256.AES_Decode(name);
    String desstudent_no = a256.AES_Decode(student_no);
    String desemail = a256.AES_Decode(email);
    String desphone = a256.AES_Decode(phone);
}
```

Figure 37. AES256 복호화

3.12 기능 수준의 접근통제

3.12.1 내용

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

a. 관리자에게만 보이는 회원 관리 버튼

학생의 경우,

201524978 조은상(eunsang) [학생]님 로그인되었습니다.
비밀번호 변경한지 35일 지났습니다.

로그아웃 본인정보관리 과제정보관리

Figure 38. 기능수준 접근통제_학생

관리자의 경우,

2000000 관리자(admin) [관리자]님 로그인되었습니다.
비밀번호 변경한지 0일 지났습니다.

로그아웃 본인정보관리 과제정보관리 회원관리

Figure 39. 기능수준 접근통제_관리자

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

학생의 경우,

과제 목록

No.	제목	작성자	작성일	제출 여부
1	a	박신혜	20151127	미제출
2	1차 과제 입니다.	박신혜	20151128	미제출
3	2차과제	박신혜	20151130	미제출

Figure 40. 기능수준 접근통제_학생

교수의 경우,

과제 목록

No.	제목	작성자	작성일	제출 학생 수
1	a	박신혜	20151127	10/10명
2	1차 과제 입니다.	박신혜	20151128	2/10명
3	2차과제	박신혜	20151130	2/10명

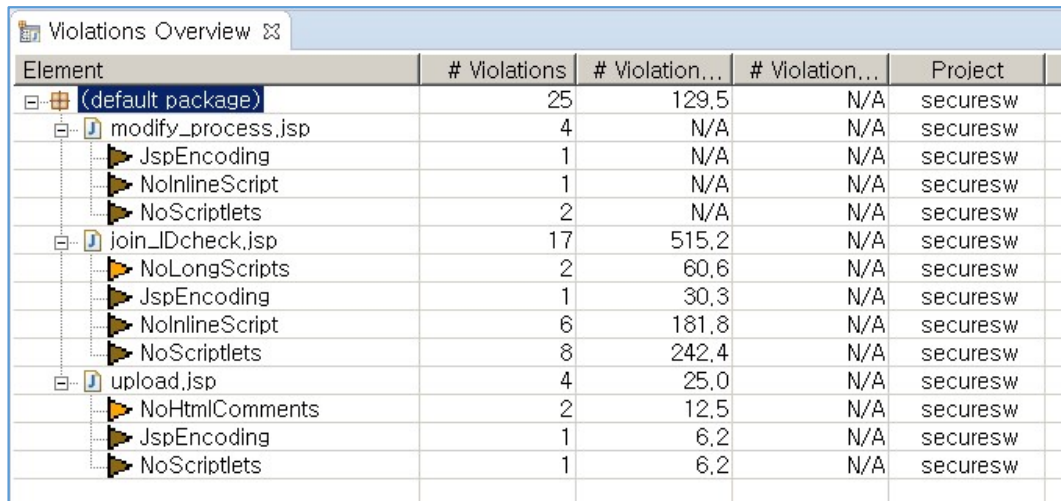
과제 등록

Figure 41. 기능수준 접근통제_교수

4. Static Analysis

4.1 PMD

(a) 2015년 11월 20일 PMD 분석결과



Element	# Violations	# Violation...	# Violation...	Project
(default package)	25	129.5	N/A	securesw
modify_process.jsp	4	N/A	N/A	securesw
JspEncoding	1	N/A	N/A	securesw
NoInlineScript	1	N/A	N/A	securesw
NoScriptlets	2	N/A	N/A	securesw
join_IDcheck.jsp	17	515.2	N/A	securesw
NoLongScripts	2	60.6	N/A	securesw
JspEncoding	1	30.3	N/A	securesw
NoInlineScript	6	181.8	N/A	securesw
NoScriptlets	8	242.4	N/A	securesw
upload.jsp	4	25.0	N/A	securesw
NoHtmlComments	2	12.5	N/A	securesw
JspEncoding	1	6.2	N/A	securesw
NoScriptlets	1	6.2	N/A	securesw

Figure 42. PMD 분석결과 - 2015.11.20

(b) 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
- ArraysStoredDirectly
 - 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

- Scriptlet은 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

(c) 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>

(d) 2015년 12월 10일 PMD 분석결과



The screenshot shows the 'Violations Overview' window of the PMD tool. It contains a table with the following data:

Element	# Violations	# Violations/KLOC	# Violations/Method	Project
▶ AES256Cipher.java	27	N/A	N/A	JspProject
▶ modify_process.jsp	3	N/A	N/A	PMD_JspProject

Figure 43. PMD 분석결과 - 2015.12.10

4.2 Acunetix

(a) 2015년 11월 24일 결과

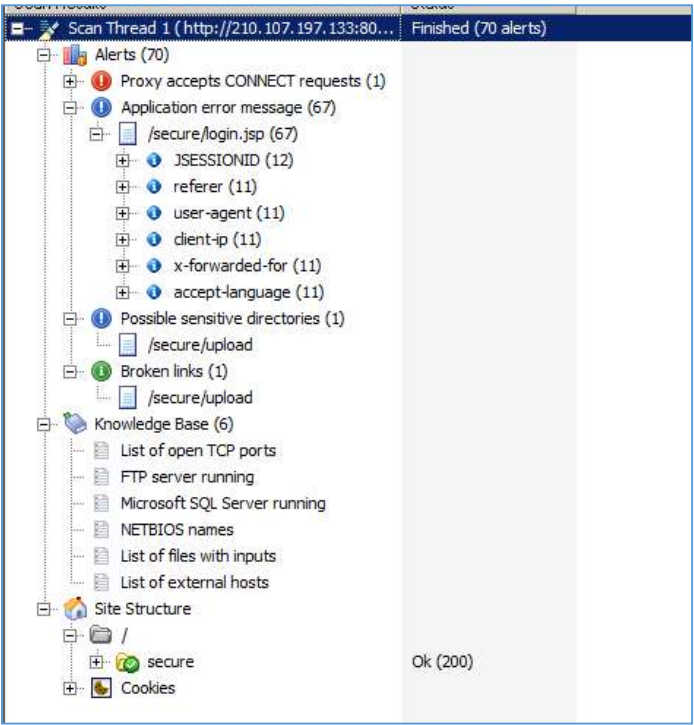


Figure 44. Acunetix 분석결과_2015.11.24

(b) 분석결과 수정사항

Log	How to Fix
Proxy Accept CONNECT request	80번 포트를 사용 중이었던 apache 서비스를 stop 시켜 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 45. Acunetix 분석 _ 수정사항

(c) 에러에 관한 보고사항

Sorry, An Error Has Occurred

An Error has occurred during the current request

[go back](#) [reload](#)

Application error message Security LOW

Vulnerability description

This page contains an error/warning message that may disclose the sensitive information. The message can also contain the location of the file that produced the unhandled exception.

This may be a false positive if the error message is found in documentation pages.

This vulnerability affects **/secure/login.jsp**.

The impact of this vulnerability

The error messages may disclose sensitive information. This information can be used to launch further attacks.

Attack details

The HTTP header **user-agent** has been set to **-268435455**.

- ⌵ [View HTTP headers](#)
- ⌵ [View HTML response](#)
- ⊙ [Launch the attack with HTTP Editor](#)
- ⊙ [Mark this alert as a false positive](#)

How to fix this vulnerability

Review the source code for this script.

Web references

- [Acunetix](#)

(d) 2015년 12월 10일 결과

The screenshot shows the Acunetix Scan Thread 1 interface. The top bar indicates the target URL is http://210.107.197.133:80... and the status is 'Executing network ...'. The left sidebar lists the following categories and counts:

- Alerts (70)
 - Proxy accepts CONNECT requests (1)
 - Server
 - Application error message (67)
 - Possible sensitive directories (1)
 - Broken links (1)
- Knowledge Base (4)
 - List of open TCP ports
 - FTP server running
 - Microsoft SQL Server running
 - NETBIOS names
- Site Structure
 - /
 - secure
 - Cookies

The right pane shows a large grey area with the text 'OK (200)' at the bottom, indicating a successful connection to the target.

Figure 46. Acunetix 분석결과 _ 2015.12.10

5 Project Summary

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

1 SQL 삽입	17 부적절한 인가	33 종료되지 않는 반복문 또는 재귀함수
2 경로 조작 및 자원 삽입	18 중요한 자원에 대한 잘못된 권한 설정	34 오류메시지를 통한 정보노출
3 크로스사이트 스크립트	19 취약한 암호화 알고리즘 사용	35 오류 상황 대응 부재
4 운영체제 명령어 삽입	20 중요정보 평문저장	36 부적절한 예외처리
5 위험한 형식 파일 업로드	21 중요정보 평문전송	37 Null Pointer 역참조
6 신뢰되지 않은 URL 주소로 자동접속 연결	22 하드코딩된 비밀번호	38 부적절한 자원 해제
7 XQuery 삽입	23 충분하지 않은 키 길이 사용	39 해제된 자원사용
8 XPath 삽입	24 적절하지 않은 난수값 사용	40 초기화되지 않은 변수 사용
9 LDAP 삽입	25 하드코딩된 암호화 키	41 잘못된 세션에 의한 데이터 정보노출
10 크로스사이트 요청 위조	26 취약한 비밀번호 허용	42 제거되지 않고 남은 디버그 코드
11 HTTP 응답분할	27 사용자 하드디스크에 저장되는 쿠키를 통한 정보노출	43 시스템 데이터 정보노출
12 정수형 오버플로우	28 주석문 안에 포함된 시스템 주요정보	44 Public 메소드로부터 반환된 private 배열
13 보안기능 결정에 사용되는 부적절한 입력값	29 솔트없이 일방향 해쉬함수 사용	45 Private 배열에 Public 데이터 할당
14 메모리 버퍼 오버플로우	30 무결성 검사 없는 코드 다운로드	46 DNS lookup에 의존한 보안결정
15 포맷 스트링 삽입	31 반복된 인증시도 제한 기능 부재	47 취약한 API 사용
16 적절한 인증 없는 중요기능 허용	32 경쟁조건 : 검사시점과 사용시점 (TOCTOU)	

Figure 47. S/W 개발보안 47

5.1 Lessons Learns

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

5.2 Work breakdown Structure

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

구분	수행 활동	상세업무-Task	담당자	일정	9월		10월				11월				12월			
					3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주
	OWASP	Database의 Error Message 차단	양경석															
		로그인 과정 암호화(SSL: Secure Socket Layer)	홍명기															
		로그인 후, 10분간 활동 없을 시에 강제 로그아웃	양경석															
		IE 창 닫으면 자동 로그아웃	양경석															
		입력한 데이터 Filtering (스크립트 실행 금지)	이수훈															
		입력 값 검증 및 치환(게시판)	이수훈															
		게시판 file upload 시 upload file 형식규제	이수훈															
		주소 직접 입력 통합 관리자 페이지 접속 차단 및 일반적인 관리자 페이지 열 사용 금지, Post 방식사용	조은상															
		오류 Page, Message -> Redirection	조은상															
		웹 페이지 강제 입력 시 세션 체크하여 권한 확인	조은상															
테스트	단위테스트	중요 Data -> Password, e-mail, 전화번호 암호화	홍명기															
	통합테스트	사용자 권한에 따른 페이지 접속 여부	양경석															
발표			All															
		보고서 최종발표	임해선															
			All															

Figure 48. WBS