# 과제 관리 시스템

## Assignment management system

**Secure S/W 설계**

**Team1**


임혜선
조은상
양경석
이수훈
홍명기

# Table of Contents

# Introduction

- Purpose
- Specs
- Functions
- Sitemap
- Data Structure

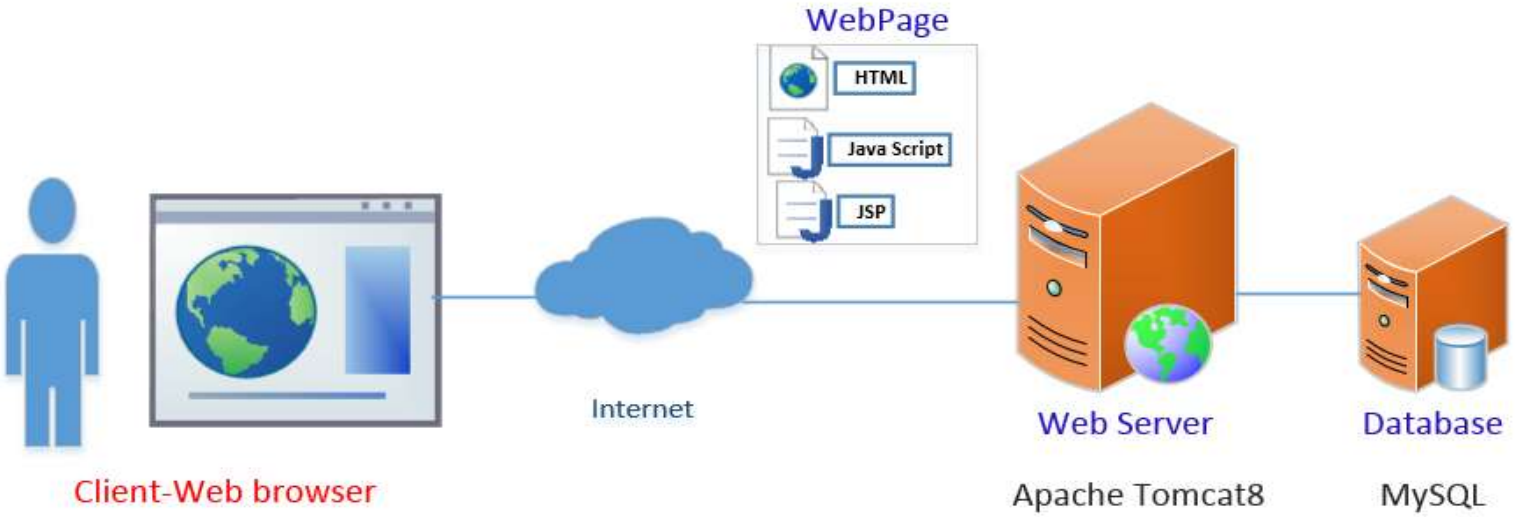
# Introduction - purpose

- 과제 관리 시스템
  - 과제 제출 및 점수 부여 및 확인하는 AIMS2 E-class의 과제 게시판 기능
  - CIA, OWASP TOP10 을 적용할 수 있는 주제로 선정
  - S/W 개발보안 47

    : 회원가입, 로그인, 게시판

# Introduction - Specs

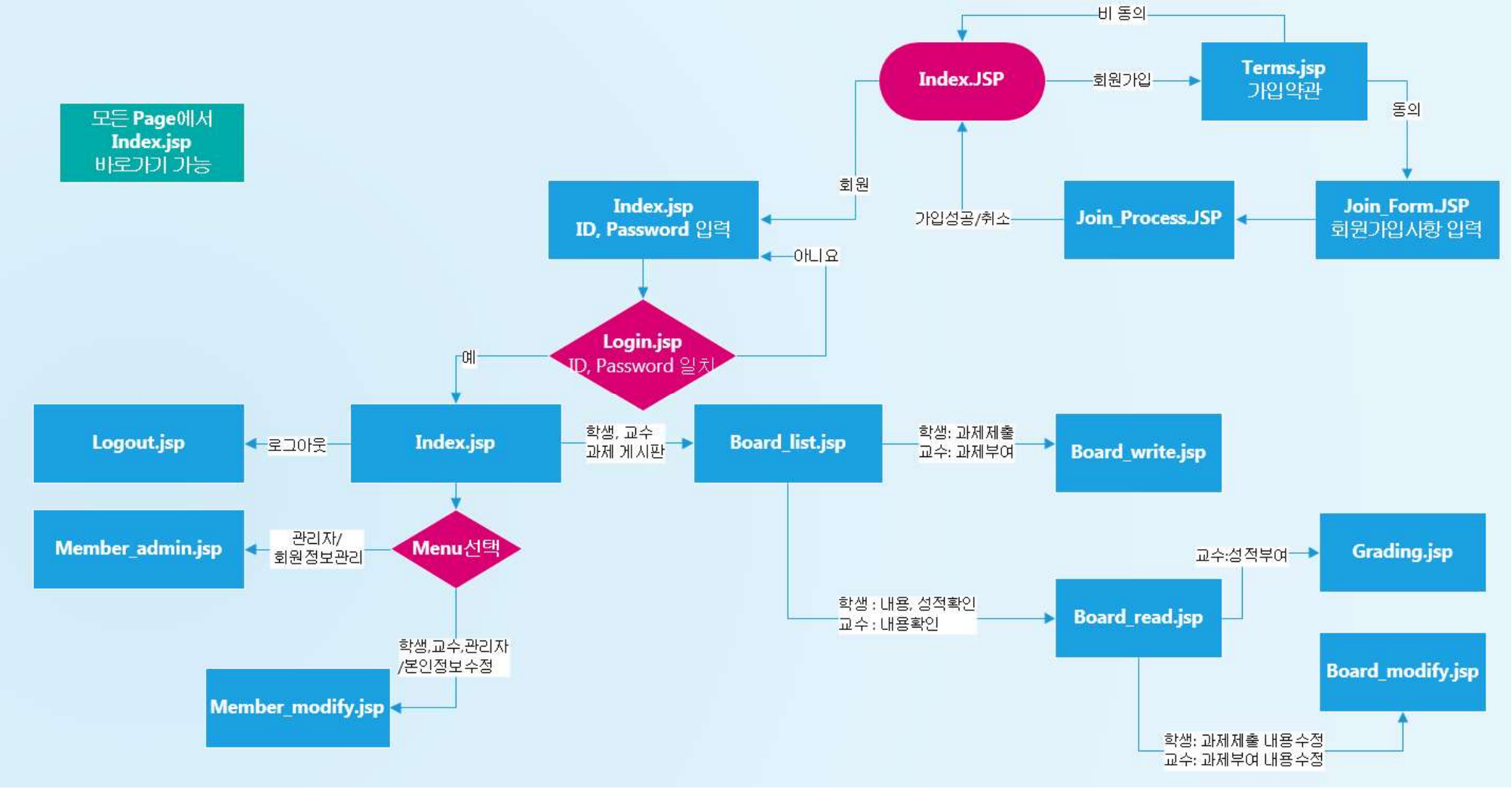
- Architecture with specs



# Introduction - Functions

- 회원가입, 회원관리, Log-in / Log-out
- 게시판 :
  과제등록, 등록과제 수정, 과제제출, 제출된 과제 확인
  제출과제 수정, 점수부여, 점수확인, 점수수정
  과제파일 Upload / Download,
  점수부여 / 확인
- Access Control :
  사용자 권한에 따른 page, data 접근 통제
  관리자, 교수, 학생

# Introduction – Site Map



# Introduction – Site Map – Page 별 Address

| | |
|---|---|
| **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<br>http://210.107.197.133:8080/secure/sub_grading.jsp |
| 회원정보관리(관리자) | http://210.107.197.133:8080/secure/member_admin.jsp |

# Introduction – Database Structure

Table 1 : Members

| 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 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 | | |

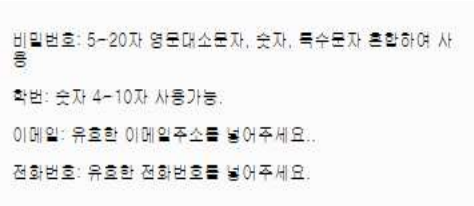
Table 3 : Data_lis

| column | value |
|---|---|
| Data_no | 제출번호 |
| File_name | 과제첨부파일 |
| New_file_name | 첨부파일명 변환 |
| publisher | 제출자 이름 |
| Board_no | 과제번호 |
| User_id | 제출자 ID |
| User_no | 제출자 학번 |

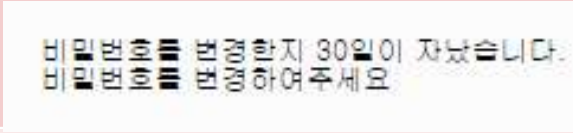
- 보안 요구사항 적용
  - 기본 보안 적용
  - **OWASP 10**
    - **+ S/W** 개발보안 **47**

# 기본보안 적용

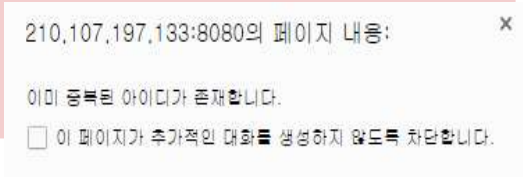
| No. | 내용 | 적용사항 |
|---|---|---|
| 1 | 회원가입 필드, 유효한 value만 입력 | 회원가입 시, 사용자 입력 value를 유효한 value만 입력하도록 적용. *[포맷스트링 삽입-S/W47]* |
| 2 | Password 입력 시, * 로 표시 | 입력 한 password대로 보이지 않도록 처리 |
| 3 | Password 30일 마다 변경 알림 | 사용자 password 설정 후, 30일 경과 시, 아래 message box 출력 |
| 4 | Password 전송 시, 암호화 전송 *[중요정보 평문전송-SW47] [하드코드 된 비밀번호-SW47]* | 암호화된 정보를 데이터베이스에 저장, 데이터베이스에서 암호화 된 정보를 가져와 복호화 하여 사용 |







# 기본보안 적용

| No. | 내용 | 적용사항 |
|---|---|---|
| 5 | Password 설정 시, 입력조건 적용 | 5~20자 영문대소문자, 숫자, 특수문자 혼합하여 사용, 그 외 회원가입 필드 조건에 대해 조건에 맞지 않을 경우, Message box 출력<br>*[취약한 비밀번호 허용 – S/W47]* |
| 6 | 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<br>*[반복된 인증시도 제한 기능부재 – S/W47]*<br>  |
| 7 | ID 중복체크 | 회원가입 시, ID 중복체크, 중복된 ID 입력 시, 아래 Message 출력<br>  |

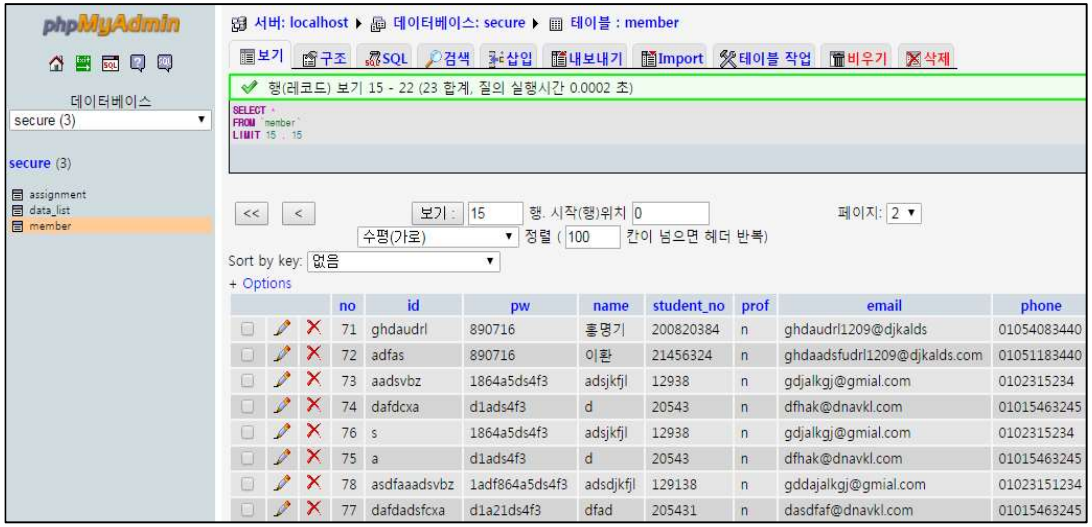
# OWASP 10

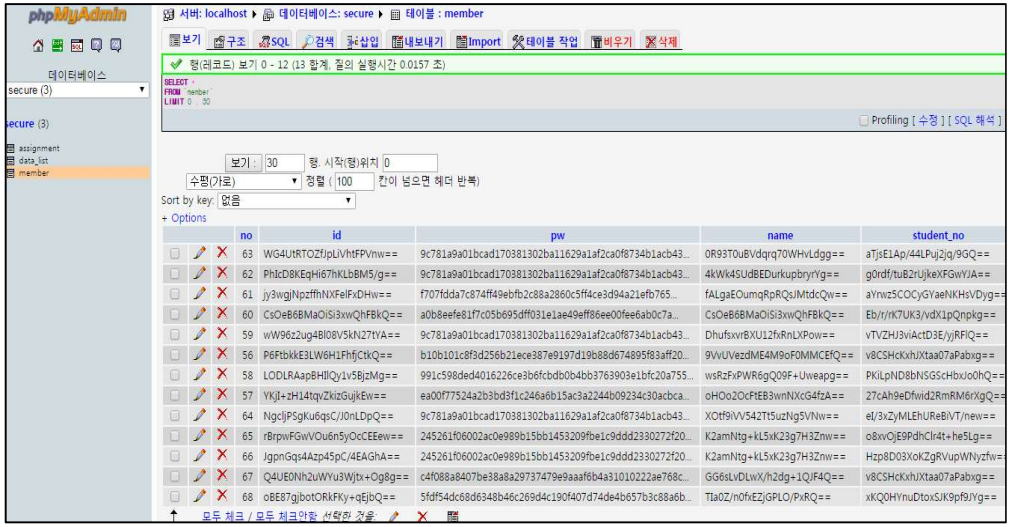
| OWASP | 내 용 |
|---|---|
| **A1. SQL Injection** | ID, Password 입력 창에서 query 문 입력 통제 (입력한 데이터 Filtering) |
| | 페이지 오류 시 Error Message 차단<br>*[오류 메시지를 통합 정보노출-SW47] [오류상황대응부재 – S/W47]*<br>*[부적절한 예외처리 – S/W47]* |
| **A2.** 인증 및 세션관리 취약점<br>*[잘못된 세션에 의한 정보노출 – S/W47]* | 로그인 과정 암호화(SSL: Secure Socket Layer) |
| | 로그인 후, 10분간 활동 없을 시에 강제 로그아웃 |
| | IE 창 닫으면 자동 로그아웃 |
| **A3. XSS, A8. CSRF**<br>*[크로스사이트 스크립팅 S/W47]*<br>*[신뢰되지 않은URL 주소로 자연접속 연결 –SW47][크로스사이트 요청 위조- SW47]* | 입력한 데이터 Filtering (스크립트 실행 금지) |
| | 입력 값 검증 및 치환(게시판) |
| | 게시판 file upload 시 upload file 형식규제 *[위험한 형식 파일 업로드 – S/W47]* |
| **A4.** 취약한 직접 객체 참조 | 오류 Page, Message -> Redirection |
| | 웹 페이지 강제 입력 시 세션 체크하여 권한 확인, 주소 입력 접속차단 *[경로조작 및 자원삽입 –S/W47]* |
| **A6.** 민감 데이터노출 | 중요 Data 암호화 *[중요정보 평문저장, 중요정보 평문전송, 충분하지 않은 키 길이 사용, 적절하지 않은 난수 값 사용, 하드코드 된 암호화 키- S/W47]* |
| | Web Page에서 DB Connection 시 ID, PW 암호화 |
| **A7.** 기능 수준의 접근통제 | 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]* |

# Secure Coding

# Database 암호화

- 내용 : Database 접근에 암호화 및 복호화
- 적용방법 : AES256, SHA256



# A6. 민감 데이터 노출

- 내용 : Web Page에서 DB Connection 시 ID, PW 암호화
- 적용방법 : AES256 Cipher를 dbUser, dbPass로 사용 후 connection시 디코딩 하여 사용

| 적용 전 | 적용 후 |
| --- | --- |
| String jdbcDriver = "jdbc:mysql://localhost:3306/secure";<br><br>//평문 DB아이디, 패스워드<br>String dbUser = "root";<br>String dbPass = "1";<br><br>String query1 = "select * from member";<br>String query2= " where id='" + enId + "'";<br>String query3= "and pw='" + enPw + "'";<br><br>//데이터베이스 커넥션 생성<br>conn = DriverManager.getConnection(jdbcDriver, dbUser, dbPass); | String jdbcDriver = "jdbc:mysql://localhost:3306/secure";<br><br>//AES256 Encoding된 DB아이디, 패스워드<br>String dbUser = "K2amNtg+kL5xK23g7H3Znw==";<br>String dbPass = "CPnv7eGM8oJo4GvYbu3ySQ==";<br><br>String query1 = "select * from member";<br>String query2= " where id='" + enId + "'";<br>String query3= "and pw='" + enPw + "'";<br><br>//데이터베이스 커넥션 생성<br>conn = DriverManager.getConnection(jdbcDriver, a256.AES_Decode(dbUser), a256.AES_Decode(dbPass)); |

# A6. 민감 데이터 노출

- 내용 : 중요 Data 암호화
- 적용방법 : AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 암호화

```
public class AES256Cipher {

    public static volatile AES256Cipher INSTANCE;

    final static String secretKey = "12345678901234567890123
    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, NoS
            InvalidKeyException, InvalidAlgorithmParameterEx
        byte[] keyData = secretKey.getBytes();

        SecretKey secureKey = new SecretKeySpec(keyData, "AE

        Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding"
        c.init(Cipher.ENCRYPT_MODE, secureKey, new IvParamete

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

```
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 modifyFlag = request.getHeader("referer");          //Member Modify의 경우

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 저장 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);
```

# A6. 민감 데이터 노출

- 내용 : 중요 Data 암호화
- 적용방법 : AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 복호화

**회원정보 관리**

| 순번 | 아이디 | 이름 | 학번 | 신분 | 비고 |
|---|---|---|---|---|---|
| 62 | PhIcD8KEqHi67hKLbBM5/g== | 4kWk4SUdBEDurkupbryrYg== | g0rdf/tuB2rUjkeXFGwYJA== | 학생 | e0/6e/anQHKEkNDX/OjaiCjFvTqSy |
| 61 | jy3wgjNpzffhNXFelFxDHw== | fALgaEOumqRpRQsJMtdcQw== | aYrwz5COCyGYaeNKHsVDyg== | 교수 | YodOZ6ae3HCMb6Ef+w |
| 60 | CsOeB6BMaOiSi3xwQhFBkQ== | CsOeB6BMaOiSi3xwQhFBkQ== | Eb/r/rK7UK3/vdX1pQnpkg== | 관리자 | Ql4dMmzEi3sYYzKij+xt9v5yq/TZY |
| 59 | wW96z2ug4Bl08V5kN27tYA== | DhufsxvrBXU12fxRnLXPow== | vTVZHJ3viActD3E/yjRFlQ== | 학생 | Tv9/yKMgCea00Iuk+FVaKdIMV/pp |
| 56 | P6FtbkkE3LW6H1FhfjCtkQ== | 9VvUVezdME4M9oF0MMCEfQ== | v8CSHcKxhJXtaa07aPabxg== | 학생 | xrUMZx89d5ePYZb7uEi1vCEUyqLT |
| 58 | LODLRAapBHIlQy1v5BjzMg== | wsRzFxPWR6gQ09F+Uweapg== | PKiLpND8bNSGScHbxJo0hQ== | 학생 | Twr8xqhFnrBzE7oNuV |

# A6. 민감 데이터 노출

- 내용 : 중요 Data 복호화
- 적용방법 : AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 복호화

```
// 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");
}
```

# A6. 민감 데이터 노출

- 내용 : 중요 Data 복호화
- 적용방법 : AES256 암호화(양방향 암호화)를 사용하여 회원의 중요 Data를 복호화

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

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

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

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

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


    while (rs.next()) {  //아이디와 비밀번호 일치하는 경우: re.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);
%>
```

# A2. 인증 및 세션관리 취약점

- 내용 : 로그인 과정 암호화(SSL: Secure Socket Layer)
- 적용방법 : 로그인, 회원가입 시 ID는 256bit의 키(AES256)
  Password는 160bit(SHA256)의 Hash로 암호화하여 사용

```java
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;
    }
}
```

- SHA256은 Hash 암호 알고리즘이며 파일 값이 약간만 바뀌어도 값이 천차만별로 변동

# A2. 인증 및 세션관리 취약점

- 내용 : 로그인 과정 암호화(SSL: Secure Socket Layer)
- 적용방법 : 로그인, 회원가입 시 ID는 256bit의 키(AES256)
  Password는 160bit(SHA256)의 Hash로 암호화하여 사용

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

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

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

- 복호화가 불가능하여 주로 패스워드 같은 인증 관련 값에 적용

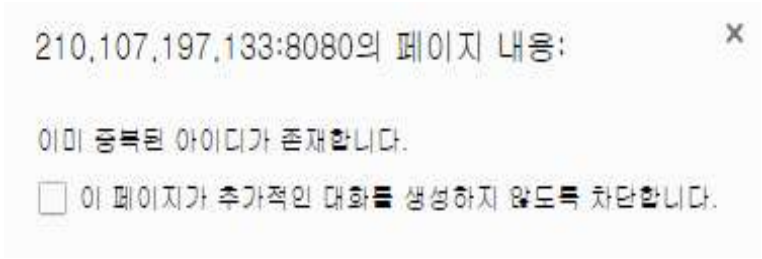
# 회원가입 필드, 유효한 Value만 입력

- 내용 : 회원가입 시, 유효한 Value만 입력 가능
- 적용방법 : 각 필드 별로 입력 유효한 값을 정규식 으로 정의하고 유효성 검사

```
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.
```

```
function validId(field) {
        if(!reg_id.test(field)) {
                return "아이디: 5~15자 영문소문자, 숫자, 특수문자 _ 사용가능₩n₩r";
        }
        return "";
}
function validpw(field1, field2) {
        if(field1 != field2) { //비밀번호 입력한 내용과 비밀번호Chk에 입력한 내용이 같은지 체크
                return "비밀번호와 비밀번호 확인의 내용이 다릅니다.₩n₩r";
        } else if(!reg_pw.test(field1)) {
                return "비밀번호: 5~20자 영문대소문자, 숫자, 특수문자 혼합하여 사용₩n₩r";
        }
        return "";
}
function validnumber(field) {
        if(!reg_number.test(field)) {
                return "학번: 숫자 4~10자 사용가능.₩n₩r";
        }
        return "";
}
function validemail(field) {
        if(!reg_email.test(field)) {
                return "이메일: 유효한 이메일주소를 넣어주세요..₩n₩r";
        }
        return "";
}
function validphone(field1, field2, field3) {
        if(!reg_phone.test(field1+'-'+field2+'-'+field3)) {
                return "전화번호: 유효한 전화번호를 넣어주세요.₩n₩r";
        }
        return "";
}
```

# ID 중복체크




- 내용 : 회원가입 시, ID 중복체크
- 적용방법: 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 전달

Join_form.jsp

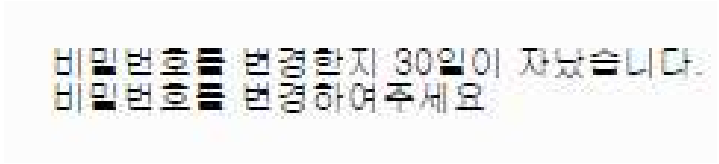
```
function check_id()
        var_id = document.getElementById("id_text").value;

        //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.");
        }
}
```

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

Join_IDCheck.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>
```


# Password 입력




- 내용 : Password 입력 시, *로 표시
- 적용방법 : Input type을 password 비밀번호 입력 보호

```
<Form method="POST" name="inform" AUTOCOMPLETE="off" action="login.jsp" or
<p> </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>
```

- 내용 : Password 30일마다 변경 알림
- 적용방법 : DB에 비밀번호 변경일자를 저장, 30일 후 알림

```
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>
```

# Password 입력


- 내용 : Log-in Password 5회 잘못 입력 시, 접근불가 – 관리자에게 재 인증
- 적용방법 : 로그인 실패 카운터 (fail_flag)를 db에 갱신하여 5회 이상 실패 시, 메시지 출력
  - 로그인 성공 시 0으로 카운터 reset

```
rs = stmt.executeQuery(query1+query2);
if (rs.next()) {
        //String failFlag = rs.getString("fail_flag");
        failFlag = Integer.parseInt(rs.getString("fail_flag"));
        //아이디는 있지만 비번실패 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;
        }

}
```

# A4. 취약한 직접객체 참조

- 내용 : 웹 페이지 강제 입력 시 세션 체크하여 권한 확인, 주소 입력 접속차단
- 적용방법 : userid 세션 & assignment_no 없으면,
  index.jsp로 강제 리다이렉트 or error message 출력

```
//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");
}
```

- 페이지 별 특성에 따른 직접 URL입력 방지
  - Join_form.jsp : 가입 시에만 필요한 약관 열람 부분이므로 로그인 중에 진입 금지
  - 로그인이 필요한 페이지 : **로그인 세션**을 체크
  - 파라미터가 필요한 페이지 : **필요 파라미터를 체크**하여
    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) { //받아온 과제 넘버가
있을때
        assignment_no = Integer.parseInt(request.getParameter("
        assignment_no")); // 과제 넘버값을 저장
} else {
        response.sendRedirect("index.jsp");
}
```

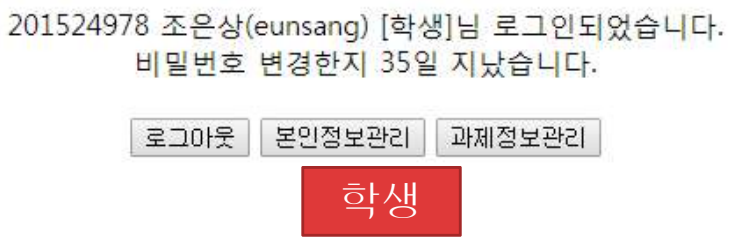
# A7. 기능 수준의 접근통제

- 내용 : 사용자 권한에 따른 페이지 접속 여부
- 적용방법 : 관리자, 교수, 학생 권한으로 나뉘어 해당하는 기능만 사용할 수 있도록 함
- 현재 Login한 sessionid를 확인하여 아래와 같이 Role에 맞은 권한을 부여함

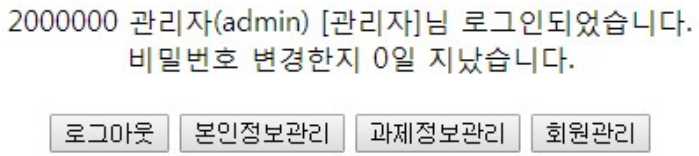
| 페이지 | 설명 | 학생 | 교수 | 관리자 |
|---|---|---|---|---|
| Index.jsp | 메인 화면 | 회원관리 버튼 안보임 | | [회원관리 버튼] |
| Board_list.jsp | 과제목록 | 본인 제출 여부 확인 | 과제 제출 인원 확인<br>[과제 등록 버튼] | 과제 제출 인원 확인<br>[과제 등록 버튼] |
| Board_read.jsp | 과제 내용 | 과제 점수 확인<br>[과제 제출 버튼] | [제출된 과제 확인 버튼] | [제출된 과제 확인 버튼] |
| Grading.jsp | 제출된 과제 목록 | 진입불가 | [학점입력/수정 버튼] | [학점입력/수정 버튼] |

# A7. 기능 수준의 접근통제

- 관리자에게만 보이는 회원 관리 버튼




학생




관리자

- 같은 페이지에서 학생권한에 필요한 내용과 교수 권한에 필요한 내용을 다르게 표시


학생


교수

# A1. SQL Injection

- 내용 : ID, Password 입력 창에서 query 문 입력 통제 (입력한 데이터 Filtering)
- 적용방법 : 정규식을 이용하여 가용된 문자 이외 사용하지 못하도록 구현

```
<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
        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;
        }

}
```

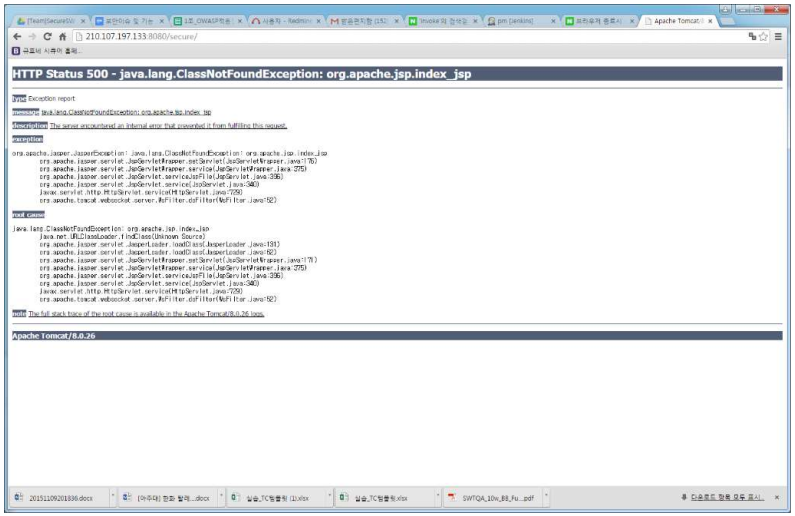
```
<center><body onLoad="focusIt()">

        <Form method="POST" name="inform" AUTOCOMPLETE="off" action="login.jsp" onSubmit="return checkIt();">
        <p> </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>
```

Submit시 checkIt()함수에서
정규식을 이용하여
입력 데이터 Filtering

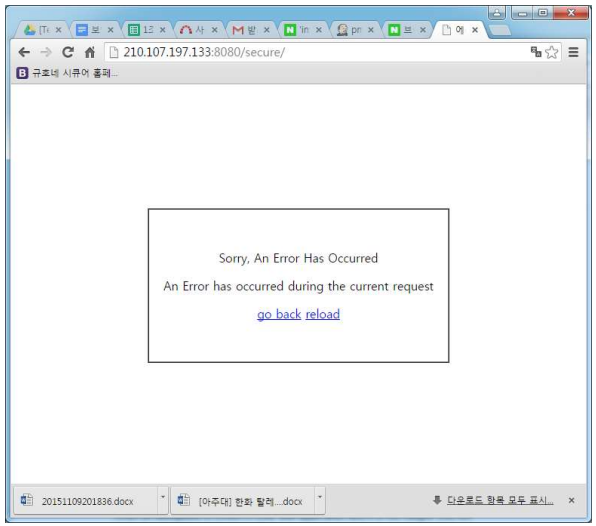
# A1. SQL Injection

- 내용 : 페이지 오류 시 Error Message 차단
- 적용방법 : /WEB-INF/web.xml 에 아래의 코드를 삽입하여 error.html로 이동하도록 조치

```
<error-page>
    <location>/errorpage.html</location>
</error-page>
```

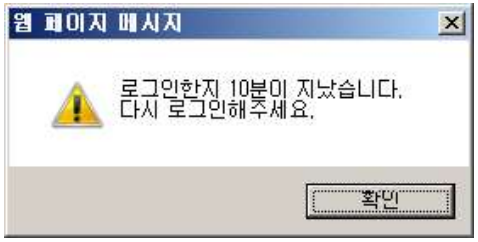



적용 전




적용 후

# A2. 인증 및 세션관리 취약점




- 내용 : 로그인 후, 10분간 활동 없을 시에 강제 로그아웃
- 적용방법 : 페이지 이동 , 시 로그인한 시간부터 체크해서 10분 경과 시
  Message box 출력 후, 로그아웃(세션삭제)
- 매 페이지마다 체크하기 위하여 세션으로 관리

```
<%//세션 생성시간 체크
long lasttime=session.getLastAccessedTime();
session.setAttribute("lasttime", lasttime);

long createdtime=session.getCreationTime();
session.setAttribute("createdtime",createdtime);

long time_used=(lasttime-createdtime)/60000;///60000; /1000하면 초단위로 나옴. 테스트때는 1000정도로 해서 테스트
session.setAttribute("time_used",time_used);

if(time_used>10){//로그인한지 10분이 지난 경우
%>
<script>alert("로그인한지 10분이 지났습니다.\n다시 로그인해주세요.");
location.href="logout.jsp";</script><%}
else{
```

# A2. 인증 및 세션관리 취약점

- 내용 : Internet Explore 창 닫으면 자동 로그아웃
- 적용방법 : 브라우저 자체적으로 지원 됨.

  - Internet Explorer
    - 자동 로그아웃 지원 됨
    - 브라우저 종료 시 자동으로 세션 삭제되어 로그아웃이 됨
  - Chrome
    - 자동 로그아웃 지원 안 함
    - 브라우저 자체에서 지원하지 않음(이미 알려진 문제)

# A3. 크로스 사이트 스크립팅
# A8. 크로스 사이트 변조요청

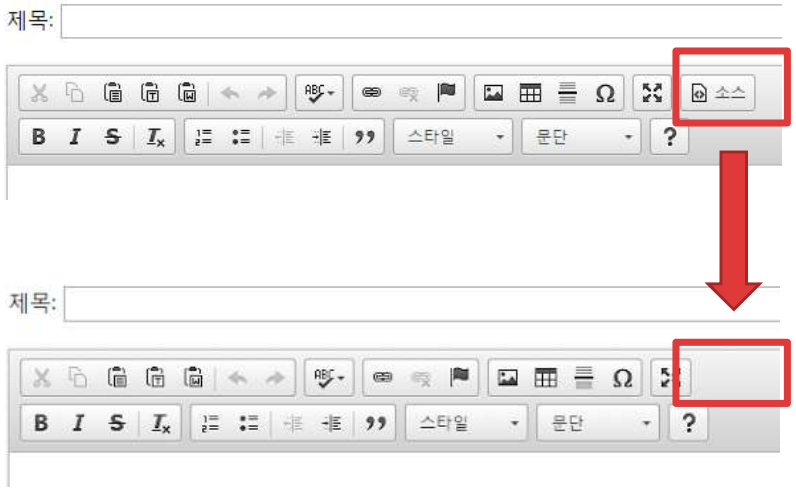
- 내용 : 입력한 데이터 Filtering (스크립트 실행 금지)
- 적용방법 : 게시판 내용-CKEDITOR의 소스 버튼 제거

- ckeditor의 config.js에 코드 추가

```
config.removeButtons = 'Source';
```

- write_index.jsp의 코드에 config.js파일이 반영되도록 수정

```
CKEDITOR.replace('text_content',{toolbar: ''});
```




- 글 작성시 태그가 Text로 출력됨

제출한 과제

| No. | 과제명 | 제출자 | 작성일 | 파일 |
|---|---|---|---|---|
| 1 | xss test(본문태그제거) | 김소특 | 20151115 | |

<script>alert('xss');</script>

# A3. 크로스 사이트 스크립팅
# A8. 크로스 사이트 변조요청

- 내용 : 입력 값 검증 및 치환(게시판)
- 적용방법
  - 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 인식 하도록 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;");
}
```

# A3. 크로스 사이트 스크립팅
# A8. 크로스 사이트 변조요청

- 내용 : 게시판 file upload 시 upload file 형식규제
- 적용방법 : 첨부파일을 zip file만 upload 가능하게 하여
  웹 브라우저에서 코드 실행 방지(form submit시 파일명 체크)

```
function fileCheck(fileValue)
{
    //확장자 체크
    var src = getFileType(fileValue);
    if(!src){//파일이 첨부되지 않은 경우
        //alert("파일이 첨부되지 않았습니다.");
        document.frm.submit();//파일 없는 채로 그냥 submit
        }
    else{//파일이 첨부된 경우
        if(!(src.toLowerCase() == "zip")){
            alert("zip 파일로 압축하여 첨부하세요.");
            return;//파일첨부창으로 유지
        }
        else{
            //alert("zip파일이 맞습니다.");
            document.frm.submit();//파일이 있는 경우 zip파일인 경우에만 submit
            }
    }
}
```

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

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

# Static Analysis

- PMD
- Acunetix

# PMD

# PMD

- PMD 분석 결과 : 2015. 11. 20

Violations Overview

| 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 |

- JspEncoding
- NoinlineScriptlet
- NoScriptlets

Modify_process.jsp

- NoLongScripts
- JspEncoding
- NoInlineScript
- NoScriptlets

Join_Idcheck.jsp


- NoHtmlComments
- JspEncoding
- NoScriptlwets

Upload.jsp

# PMD_ 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

# PMD_ PMD Rule 분석

- 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

# PMD_ PMD Rule 분석

- 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

# PMD_ PMD Rule 분석

- 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

# PMD

- 항목별 수정사항

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

# PMD

- PMD 분석 결과 : 2015. 12.10

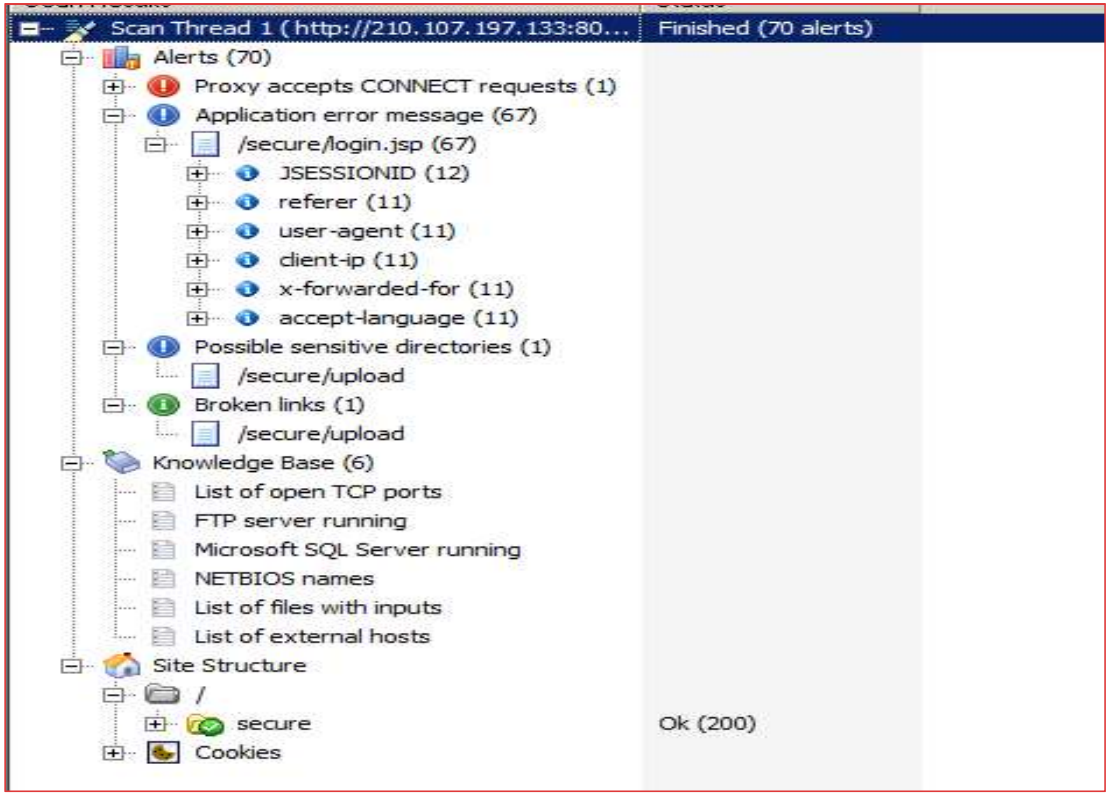
Violations Overview | Console | SVN Repositories

| 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 |

# Acunetix 6.0

# Acunetix

- Acunetix 분석 결과 : 2015. 11. 24



# Acunetix

- 항목별 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. |

# Acunetix

- Acunetix 에러에 관한 보고 사항


에러페이지 출력 메시지

**Application error message** Severity 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**

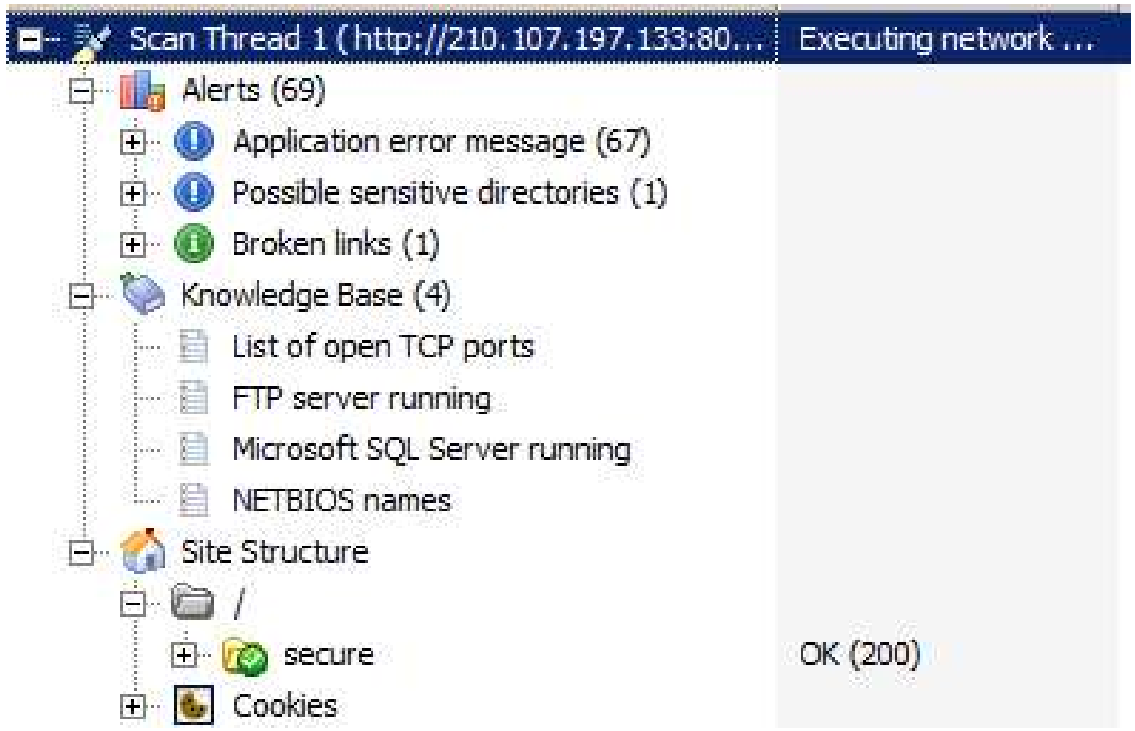
Review the source code for this script.

**Web references**

- Acunetix

# Acunetix

- Acunetix 분석 결과 : 2015. 12. 10



# Demo

http://210.107.197.133:8080/secure/index.jsp

# Project Summary

# 적용된 보안 요구사항

- 일반적인 보안 요구사항
- OWASP 10
- S/W 보안요구사항 47

| | |
|---|---|
| 1 | SQL 삽입 |
| 2 | 경로 조작 및 자원 삽입 |
| 3 | 크로스사이트 스크립트 |
| 4 | 운영체제 명령어 삽입 |
| 5 | 위험한 형식 파일 업로드 |
| 6 | 신뢰되지 않은 URL 주소로 자동접속 연결 |
| 7 | XQuery 삽입 |
| 8 | XPath 삽입 |
| 9 | LDAP 삽입 |
| 10 | 크로스사이트 요청 위조 |
| 11 | HTTP 응답분할 |
| 12 | 정수형 오버플로우 |
| 13 | 보안기능 결정에 사용되는 부적절한 입력값 |
| 14 | 메모리 버퍼 오버플로우 |
| 15 | 포맷 스트링 삽입 |
| 16 | 적절한 인증 없는 중요기능 허용 |

| | |
|---|---|
| 17 | 부적절한 인가 |
| 18 | 중요한 자원에 대한 잘못된 권한 설정 |
| 19 | 취약한 암호화 알고리즘 사용 |
| 20 | 중요정보 평문저장 |
| 21 | 중요정보 평문전송 |
| 22 | 하드코드된 비밀번호 |
| 23 | 충분하지 않은 키 길이 사용 |
| 24 | 적절하지 않은 난수값 사용 |
| 25 | 하드코드된 암호화 키 |
| 26 | 취약한 비밀번호 허용 |
| 27 | 사용자 하드디스크에 저장되는 쿠키를 통한 정보노출 |
| 28 | 주석문 안에 포함된 시스템 주요정보 |
| 29 | 솔트없이 일방향 해쉬함수 사용 |
| 30 | 무결성 검사 없는 코드 다운로드 |
| 31 | 반복된 인증시도 제한 기능 부재 |
| 32 | 경쟁조건 : 검사시점과 사용시점(TOCTOU) |

| | |
|---|---|
| 33 | 종료되지 않는 반복문 또는 재귀함수 |
| 34 | 오류메시지를 통한 정보노출 |
| 35 | 오류 상황 대응 부재 |
| 36 | 부절절한 예외처리 |
| 37 | Null Pointer 역참조 |
| 38 | 부적절한 자원 해제 |
| 39 | 해제된 자원사용 |
| 40 | 초기화되지 않은 변수 사용 |
| 41 | 잘못된 세션에 의한 데이터 정보노출 |
| 42 | 제거되지 않고 남은 디버그 코드 |
| 43 | 시스템 데이터 정보노출 |
| 44 | Public 메소드로부터 반환된 private 배열 |
| 45 | Private 배열에 Public 데이터 할당 |
| 46 | DNS lookup에 의존한 보안결정 |
| 47 | 취약한 API 사용 |

# Lesson Learns

- 웹 보안에 대한 경각심 고취
- Secure Coding Experience

  Vulnerabilities Analysis, design, development

  Static Analysis, Mitigation of identified risks

# Q & A