ISBN : 000-00-000-0000-0

웹 어플리케이션 취약점 분석
('17년 2분기)

2017. 06

KiSTi www.kisti.re.kr
한국과학기술정보연구원
Korea Institute of Science and Technology Information

웹 어플리케이션 취약점 분석
('17년 2분기)

2017. 06

부 서 : 첨단연구망센터 첨단연구망정보보호실
제출자 : 이형주 (lhj275@kisti.re.kr)

KiSTi www.kisti.re.kr 한국과학기술정보연구원 Korea Institute of Science and Technology Information

[목 차]

제1장 서론

최근 웹을 이용한 침해사고를 미연에 방지하기 위한 하나의 방법으로써 웹 취약점 분석에 관한 연구가 활발히 진행되고 있다. 네트워크 환경이 실 생활에 필수 요소로 자리 잡은 지금 웹은 모든 응용 계층 중에 가장 많이 사용하는 프로토콜이 되었다. 이러한 환경의 변화로 많은 양의 웹 응용 프로그램들이 등장하게 되었고, 이들의 취약점을 이용한 공격사례들이 증가하게 되었다.

웹 서비스는 개방된 환경에서 보안장비를 거치치 않고 사용자와 서버 간 통신이 연결되는 구조적으로 취약한 특성을 가지고 있어 이로 인해 악의적인 공격자에 의한 공격타겟이 되기 쉽다. 이러한 공격을 보호하기 위한 대책으로 보안장비 도입을 통한 실시간 모니터링, 보안정책 관리 등의 보안조치를 수행하고 있다. 하지만 이러한 보안시스템들은 웹 어플리케이션 취약점의 근원적인 문제 해소를 보장하지 못한다. 따라서 날로 지능화되는 공격기법에 대응하기 위해서는 웹 어플리케이션에 대한 지속적인 취약점 점검 및 개선조치가 반드시 필요한 실정이다.

이에, 과학기술사이버안전센터에서는 선제적인 웹 취약점 제거를 통해 보안사고를 미연에 방지할 수 있도록 자동화기반의 취약점 진단 시스템을 구축·보급하여 대상기관에서 운영중인 웹사이트의 균형적인 보안수준 향상을 도모하고 있다.

본 기술보고서에서는 17년도 2/4분기에 취약점 블랙박스 테스트를 통해 실제 탐지된 『**소스코드 내 중요정보 노출 취약점, 공개용 웹 게시판 취약점, 크로스사이트 스크립트(XSS) 취약점**』을 중심으로 취약점에 대한 상세한 설명과 취약점 개선에 필요한 조치방안을 기술하고자 한다.

제2장 관련 연구

제1절 웹 어플리케이션 취약점 유형

본 절에서는 웹 어플리케이션에서 발생하는 취약점의 정의와 주요 탐지 유형에 대하여 살펴본다. 과학기술사이버안전센터에서 정의한 17개의 취약점 유형은 아래와 같다.

1 관리자 페이지 노출 취약점

일반적으로 추측이 가능한 관리자 페이지 경로(/admin, /manager 등)를 사용하거나, 프로그램 설계상의 오류, 인증 미흡으로 인해 관 리자 메뉴에 직접 접근이 가능하며 권한인증이 가능한 취약점

2 디렉터리 나열 취약점

서버내의 모든 디렉터리 혹은 중요한 정보가 포함된 디렉터리에 대해 인덱싱이 가능하게 설정되어 중요파일 정보가 노출될 수 있는 취약점

3 시스템 관리 취약점

응용 프로그램 설치 중에 생성되는 설치·임시 파일이 존재하거나 웹상에서 윈도우 로그인 창이 노출되는 등 시스템 상 설정 미비로 인해 발생하는 취약점

4 WEBDAV 취약점

IIS 일부 버전의 취약점으로 악의적인 HTTP 요청을 이용하여 FTP나 시스템에 직접 접근하지 않고 원격에서 파일을 수정 및 처리가 가능한 취약점

5 불필요한 Method 허용 취약점

웹 서비스 제공 시 불필요한 Method(PUT, DELETE, OPTIONS 등) 허용으로 외부 공격자에 의해 악성파일을 업로드 하거나 중요파일에 대한 조작이 가능해지는 취약점

6 취약한 파일 존재 취약점

웹 루트 하위에 내부 문서나 백업파일, 로그파일, 압축파일과 같은 파일이 존재할 경우 파일명을 유추하여 파일명을 알아내고, 직접 요청하여 해킹에 필요한 서비스 정보를 획득할 수 있는 취약점

7 계정 관리 취약점

회원가입 시에 안전한 패스워드 규칙이 적용되지 않아서 취약한 패스워드로 회원 가입이 가능할 경우 무차별 대입공격을 통해 패스워드가 누출될 수 있는 취약점

8 실명인증 취약점

본인 확인 과정상에서 취약한 프로그램을 악용하여 사용자정보를 변조하는 공격으로 관리자 위장을 통해 개인정보를 수집하거나 기타 공격에 악용할 수 있는 취약점

9 전송 시 개인정보 노출 취약점

프로그램이 보안과 관련된 민감한 데이터를 평문으로 통신 채널을 통해서 송·수신 할 경우, 통신채널 스니핑을 통해 인가되지 않은 사용자에게 민감한 데이터가 노출될 수 있는 취약점

10 파일 다운로드 취약점

외부 입력값에 대해 경로 조작에 사용될 수 있는 문자를 필터링하지 않는 취약점을 악용하여 예상 밖의 접근 제한 영역에 대한 경로 문자열 구성이 가능해져 시스템 정보누출, 서비스 장애 등을 유발 시킬 수 있는 취약점

11 파일 업로드 취약점

공격자가 웹 사이트에 있는 게시판이나 자료실의 파일 업로드 기능을 이용하여 공격자가 만든 특정 공격 프로그램을 업로드하여 웹 서버의 권한 획득이 가능한 취약점

12 소스코드 내 중요정보 노출 취약점

소스코드 주석문에 민감한 정보(개인 정보, 시스템 정보 등)이 포함되어 있는 경우, 외부 공격자에 의해 패스워드 등 보안 관련정보가 노출될 수 있는 취약점

13 공개용 웹 게시판 취약점

공개용 게시판을 사용할 경우 인터넷에 공개된 각종 취약점 정보로 인해 홈페이지 변조 및 해킹 경유지로 사용될 수 있는 취약점

14 크로스사이트스크립트(XSS) 취약점

공격자가 클라이언트 스크립트를 악용하여 웹사이트에 접속하려는 일반 사용자로 하여금 공격자가 의도한 명령이나 작업을 수행하는 공격으로, 세션탈취, 웹사이트 위변조, 악성 스크립트 삽입 및 실행, 접근경로 리다이렉트 등의 다양한 공격을 유발할 수 있는 취약점

15 구문삽입(SQL-Injection) 취약점

URL 요청 또는 웹 요청에 포함되는 웹 어플리케이션에서 입력 폼 및 URL입력란에 SQL 문을 삽입하는 형태의 공격으로 시스템 내부정보를 열람 또는 조작할 수 있는 취약점

16 권한인증 취약점

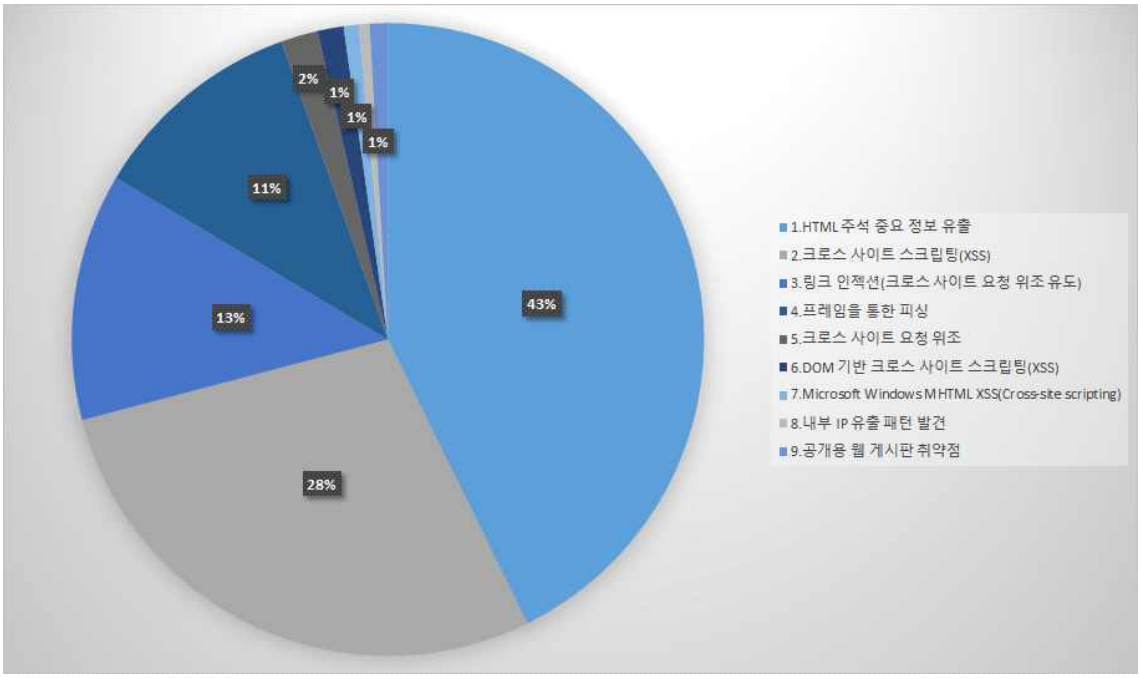
웹 어플리케이션 상에서 모든 실행 경로에 대해서 접근제어를 검사하지 않거나 불완전하게 검사하는 취약점을 이용하여 임의의 명령 실행이 가능한 악의적인 파일을 서버로 업로드하여 권한을 탈취할 수 있는 취약점

17 에러처리 취약점

웹 서버에 별도의 에러페이지를 설정하지 않은 경우, 에러 메시지를 통해 서버 데이터 정보 등 공격에 필요한 정보가 노출되는 취약점

제2절 웹 취약점 유형 별 주요 탐지현황

과학기술사이버안전센터에서는 웹 어플리케이션 분야의 취약점을 탐지하기 위하여 다수의 패턴을 보유하고 있으며, 앞서 분류된 웹 취약점 유형들이 포함하고 있는 주요 탐지패턴 현황은 아래와 같다. 이번 보고서에는 17년도 2/4분기에 주로 탐지된 11개의 취약점 패턴 분석내용을 다루도록 한다.



순번	취약점 유형	주요 탐지패턴
1	소스코드 내 중요정보 노출 취약점	[1-1] 내부 IP 유출 패턴 발견
		[1-2] HTML 주석 중요 정보 유출
2	공개용 웹 게시판 취약점	[2-1] 공개용 웹 게시판 취약점
3	크로스사이트스크립트 (XSS) 취약점	[3-1] 크로스사이트 스크립팅 (XSS)
		[3-2] 프레임을 통한 피싱
		[3-3] 링크 인젝션 (크로스 사이트 요청 위조 유도)
		[3-4] 크로스 사이트 요청 위조
		[3-5] Microsoft Windows MHTML XSS (Cross-site scripting)
		[3-6] DOM 기반 크로스 사이트 스크립팅 (XSS)
		[3-7] 호스트가 모든 도메인에서도 플래시 액세스 허용
		[3-8] Silverlight : 임의의 도메인으로부터의 액세스 허용

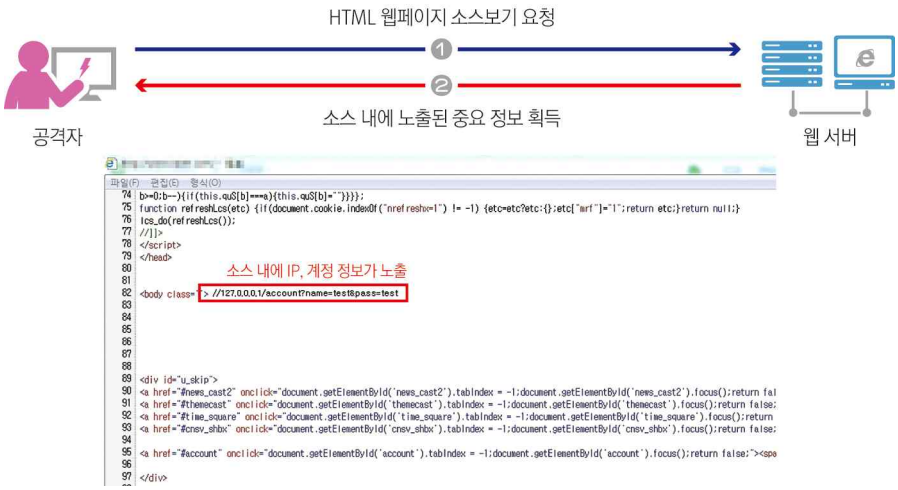
<2017년도 2/4분기 웹 취약점 유형 탐지현황>

제3장 웹 취약점 유형 별 상세 조치방안

제1절 소스코드 내 중요정보 노출 취약점

취약점 설명

웹 페이지의 소스코드 또는 주석문에 민감한 정보(개인 정보, 시스템 정보 등)가 포함되어 있는 경우, 공격자가 해당 정보를 획득하여 공격에 악용할 수 있다. 특히 웹 애플리케이션 개발 시 개발자가 편의를 위해 삽입한 코드에는 관리자 페이지 접속 링크나, 게시판 삭제용 패스워드 등이 그대로 노출 되는 경우도 존재하는데, 이러한 정보 노출로 인해 공격자는 별다른 시도 없이 손쉽게 공격이 가능하다.



<소스 내에 포함된 민감정보 노출>

사전예방조치 방안

* 개발 시 주로 사용되던 소스코드 및 주석처리부분 등을 확인하여 삭제
* 소스 노출이나 DB정보 노출 등의 문제가 발생할 수 있는 웹서버상의 불필요한 백업 파일 모두 삭제
* 중요 정보가 포함된 설정파일(예 : dbconnet.inc) 등의 권한 설정을 확인하여 홈페이지 상에 노출이 되지 않도록 설정

[1-1] 내부 IP 유출 패턴 발견

◎ 개요

웹 애플리케이션 내 웹 페이지 소스에서, 주석 구문 및 처리구문에 직접적으로 IP정보나 파일경로 등 시스템 내부의 중요 정보가 포함되어 외부로 노출되는 취약점으로, 공격자는 노출된 중요 정보를 통해 추가 공격을 시도할 수 있음

<소스 내에 포함 된 IP 및 계정정보 노출>

◎ 조치방안

* 주석구문에 중요 정보 삭제
* IP 및 접속 계정 등 중요 정보는 별도의 파일로 관리하며 난독화 처리를 통해 외부 노출이 불가능하도록 설정

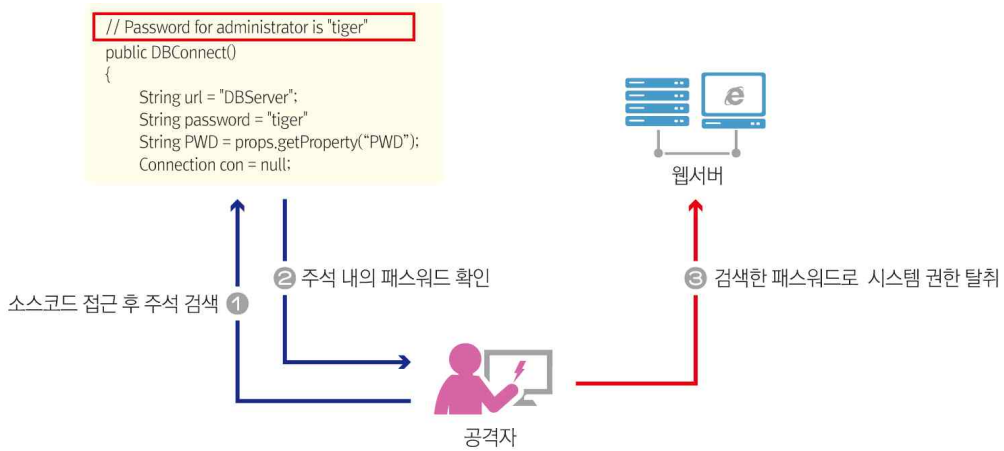
안전한 코드 예제

```
Properties props = new Properties();
....
.....
String IP = props.getProperty("IP");
String ID = props.getProperty(“ID”);
String PWD = props.getProperty(“PWD”);
....
....
Class.forName(“com.mysql.jdbc.Driver”);
conn=DriverManager.getConnection(IP,ID,PWD);
```

[1-2] HTML 주석 중요정보 노출

◎ 개요

웹 페이지 소스 내의 HTML 주석을 통해 SQL 코드, ID/Password, 내부IP, 디버깅 정보 등과 같은 시스템 구성에 관한 중요 정보가 노출될 경우, 공격자에게 내부 정보를 제공할 위험성이 존재하는 취약점



<주석문 안에 포함된 시스템 주요정보>

◎ 조치방안

* 기밀정보가 포함된 html 주석 삭제
* 중요한 로직 및 주석에 대한 처리는 웹서버에서 구현되는 언어(Server Side)와 같이 처리되도록 작성

· Client Side 언어 : HTML, JAVA Script, Virtual Basic Script 등

· Server Side 언어 : ASP, JSP, PHP, Perl 등

안전한 코드 예제

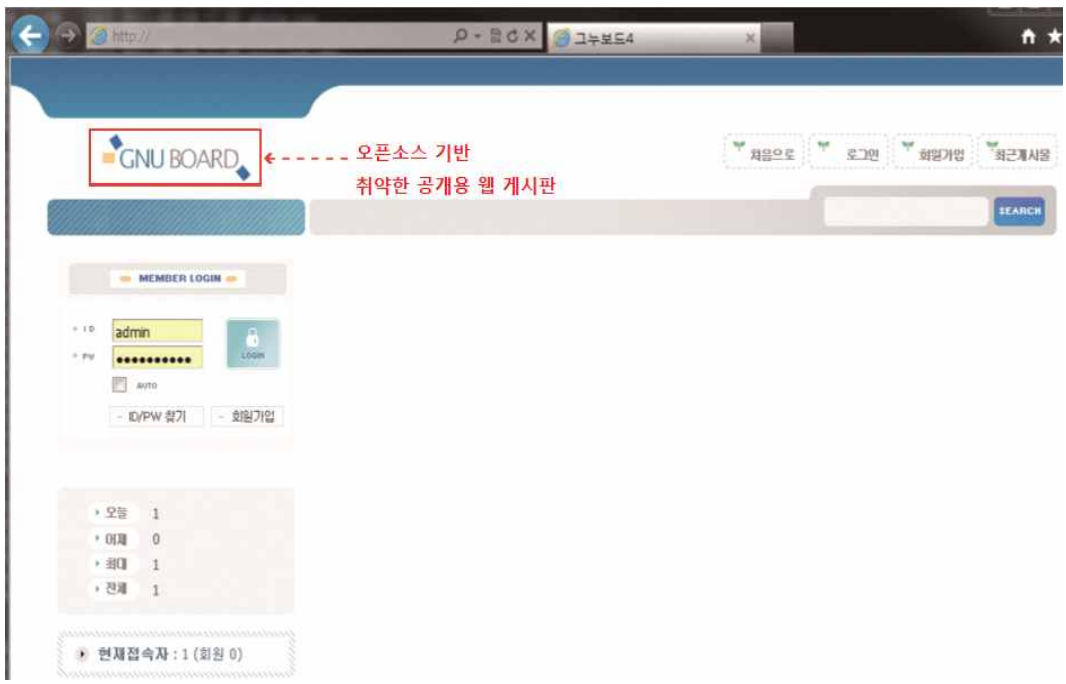
```
……
//디버깅 등의 용도로 소스 주석에 적어놓은 패스워드 삭제
 public Connection DBConnect(String id, String Password) {
 String url = "DBServer";
 Connection conn = null;

    try {
     String CONNECT_STRING = url+":"+id+":"+ Password;
     InitialContext ctx = new InitialContext();
     DataSource datasource = (DataSource) ctx.lookup(CONNECT_STRING);
     conn = datasource.getConnection();
     } catch (SQLException e) {……}
     return conn;
     }
```

제2절 공개용 웹 게시판 취약점

취약점 설명

공공기관에서는 웹서버의 구축 시 금전적, 시간적인 부담으로 인해 공개용 게시판(제로보드, 그누보드 등)을 많이 이용하고 있다. 오픈소스 기반으로 제작 된 공개용 웹 게시판을 사용할 경우 각종 취약점에 대한 정보가 인터넷에 공개되는 특성이 있는데 공격자가 이러한 정보를 수집하게 되면 웹 게시판 별로 유발되는 취약점을 악용하여 홈페이지 위·변조나 해킹 경유지 사용, 파일 업로드 등 다양한 공격이 가능해진다.



<취약한 웹 게시판 예시>

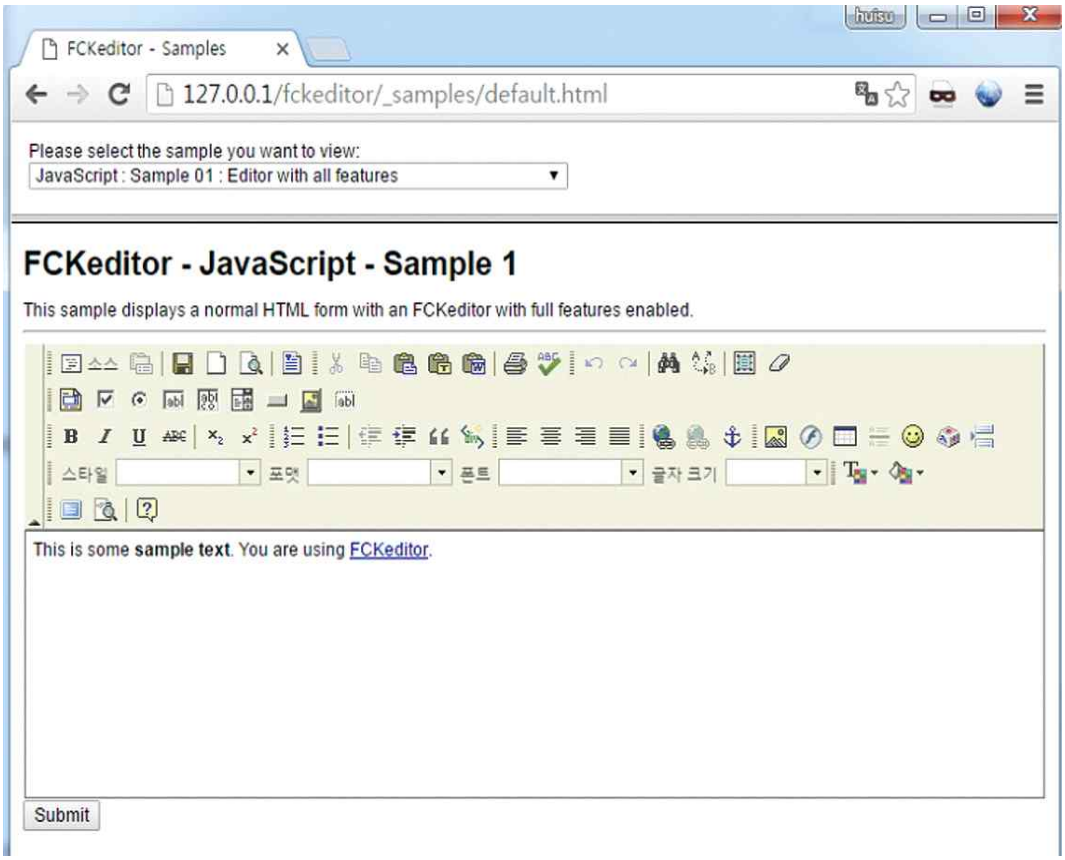
사전예방조치 방안

* 웹서버에는 공개용 웹 게시판(제로보드, 테크노트, 세팔보드, 그누보드 등) 사용 지양
* 필요에 의해 사용해야 하는 경우는 보안 패치 또는 최신 버전 제품으로 설치
* 정기적으로 게시판 배포 사이트에 방문하여 보안 취약점 정보 확인

[2-1] 공개용 웹 게시판 취약점

◎ 개요

오픈소스 기반으로 제작된 공개용 웹 게시판은 취약점이 외부로 공개되는 특성으로 인해 공격자가 해당정보를 활용하여 홈페이지 변조 및 시스템 장악 등 공격시도에 악용될 수 있으며 대표적인 웹 에디터로는 제로보드, FCKeditor, WordPress, 그누보드, 텍스트큐브 등이 있음



<공개용 게시판의 파일업로드 취약점 예시>

◎ 조치방안

* 구축되어 있는 웹 게시판의 종류 확인 후 공식 배포 사이트를 통한 보안패치를 최신 버전으로 유지
* 웹 게시판의 배포 버전 및 정보 확인
* 웹서버 내에 설치파일 검색 후 불필요 시 삭제

· UNIX, Linux 검색 예시

제로보드 검색

find/[웹서버디렉터리] -name "license.txt" -exec ls -alt {} ₩; -exec grep "배포버전:" {}₩;

테크노트 검색

find/[웹서버디렉터리] -name "config.cgi" -exec ls -alt {} ₩; -exec grep "배포버전:" {}₩;
find/[웹서버디렉터리] -name "main.cgi" -exec ls -alt {} ₩; -exec grep "배포버전:" {}₩;

· 브라우저를 통한 샘플페이지 검색 예시

제로보드 검색

http://홈페이지주소/bbs/license.txt
http://홈페이지주소/zb/license.txt
http://홈페이지주소/zeroboard/license.txt

테크노트 검색

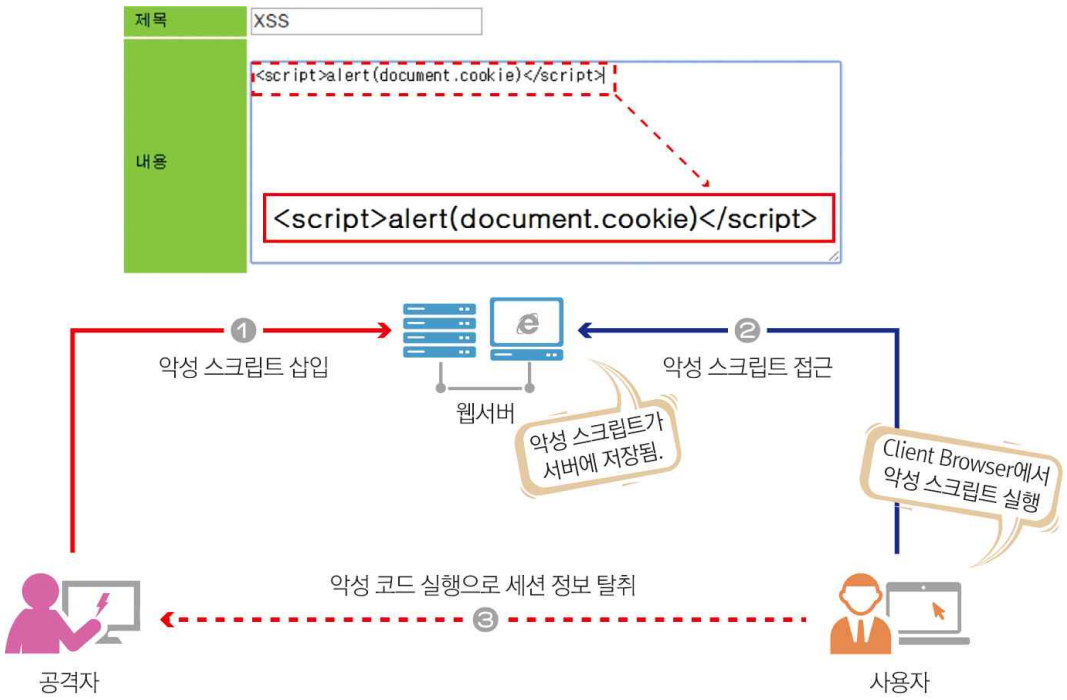
http://홈페이지주소/게시판 디렉터리/config.cgi
http://홈페이지주소/게시판 디렉터리/main.cgi

※ 공개용 게시판은 패치가 될 때까지 지속적인 취약점에 노출되므로 사용을 지양해야 함

제3절 크로스사이트스크립트(XSS) 취약점

취약점 설명

취약한 URL 매개변수나 홈페이지 내의 입력란에 사용자가 임의의 값을 입력할 때, 입력 값에 대하여 검증되지 않은 상태로 저장이나 실행 될 경우, 공격자가 웹 페이지에 악의적인 스크립트를 삽입시킬 수 있으며, 해당 페이지로 접근하는 사용자를 대상으로 악성 스크립트를 실행시킴으로써 개인정보 탈취, 홈페이지 위·변조, 악성코드 감염 등의 다양한 공격을 유발할 수 있다.



<악성 스크립트 실행을 통한 사용자 세션정보 탈취>

사전예방조치 방안

* 웹서버에서 입력 값에 정의된 문자 길이를 검증하여 악성 스크립트 명령이 삽입되지 않도록 수정
* 사용자 입력 폼(로그인 폼, 검색 폼, URL 등)을 대상으로 특수문자, 특수구문 필터링 규칙 적용

* Secure Coding

DotNet

· HTML 인코딩 : URL 파라미터로 전달되는 데이터에 대한 html 인코딩 처리

```
예시 1)
String subject = row[“subject”];
subject = subject.replace(“<“,“&lt;”);
subject = subject.replace(“>”,“&gt;”);

예시 2)
String subject = row[“subject”];
subject = Server.HtmlEncode(subject); -- HtmlEncode API 사용
```

· DotNet Request Validator 처리 : Inside HTML 공격방어

```
<%@ Page Language="C#" AutoEventWireup="true“
CodeFile="Default.aspx.cs" Inherits="_Default" ValidateRequest=“true" %>
```

· 입력값 이스케이프 처리

```
public static String validation(String string) {
 return string
 .replaceAll("(?i)<script.*?>.*?</script.*?>", "") // case 1
 .replaceAll("(?i)<.*?javascript:.*?>.*?</.*?>", "") // case 2
 .replaceAll("(?i)<.*?₩₩s+on.*?>.*?</.*?>", ""); // case 3
 }
```

JAVA/JSP

· HTML 인코딩 : URL 파라미터로 전달되는 데이터에 대한 html 인코딩 처리

```
예시 1)
String subject = rs.getString(“subject”);
subject = subject.replaceAll(“<“, “&lt;”);
subject = subject.replaceAll(“>”, “&gt;”);

예시 2)
import org.apache.commons.lang.StringEscapeUtils;
String subject = rs.getString(“subject”);
subject = StringEscapeUtils.escapeHtml(subject); -- HtmlEncode API 사용
```

· 입력값 이스케이프 처리

```
public static String validation(String string) {
return string
 .replaceAll(“(?i)<script.*?>.*?</script.*?>”, “”) // case 1
 .replaceAll(“(?i)<.*?javascript:.*?>.*?</.*?>”, “”) // case 2
 .replaceAll(“(?i)<.*?₩₩s+on.*?>.*?</.*?>”, “”); // case 3
 }
```

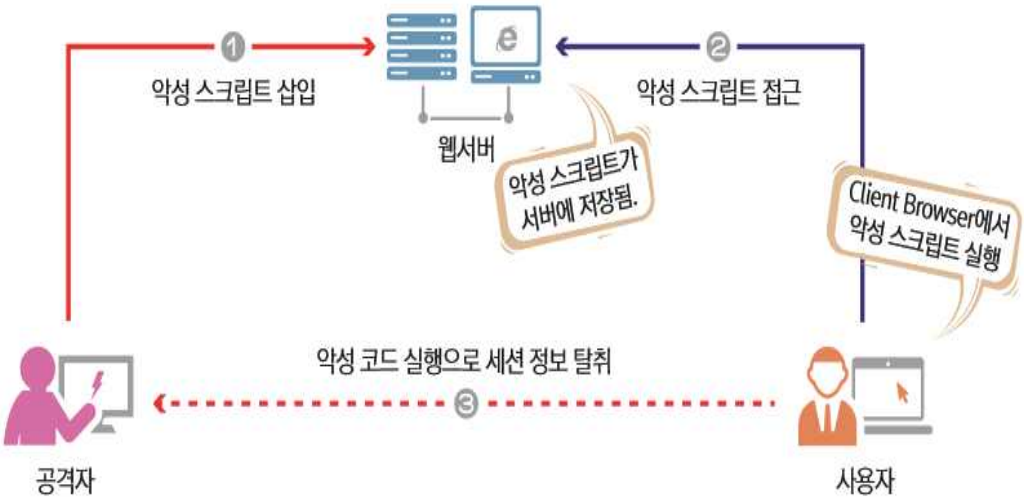
· JSTL 처리 : JSTL에서 제공하는 <c:out escapeXML=”true” value=”값”/> 사용

```
<c:out escapeXml=”true” value=”${value}” />
```

[3-1] 크로스 사이트 스크립팅(XSS)

◎ 개요

공격자가 클라이언트 스크립트를 악용하여 웹사이트에 접속하려는 일반 사용자로 하여금 공격자가 의도한 명령이나 작업을 수행하는 공격으로, 세션탈취, 웹사이트 위변조, 악성 스크립트 삽입 및 실행, 접근경로 리다이렉트 등의 다양한 공격을 유발할 수 있는 취약점



<악성 스크립트 실행을 통한 세션정보 탈취>

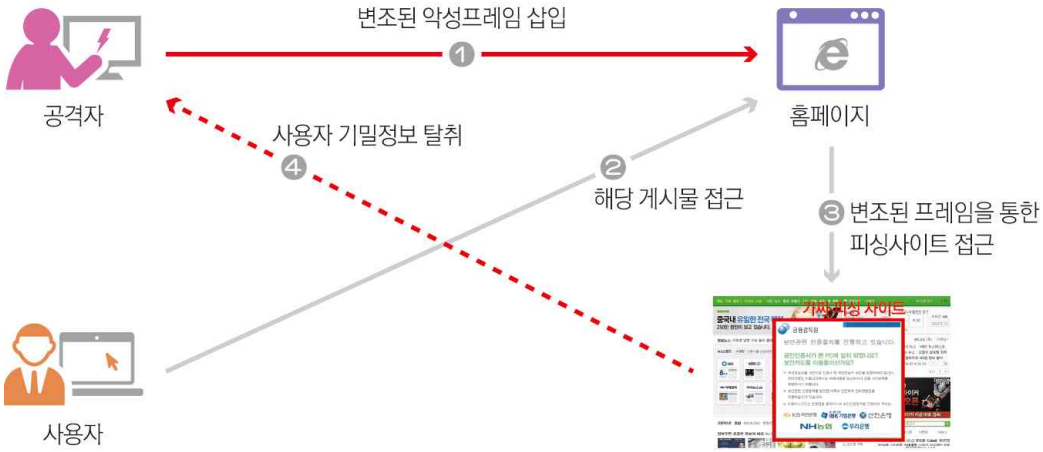
◎ 조치방안

* 악의적인 스크립트가 실행되지 않도록 주요 공격구문에 대하여 HTML 인코딩 적용
* **7절 크로스사이트스크립트(XSS) 취약점 '사전예방조치 방안'** 내용참고

[3-2] 프레임을 통한 피싱

◎ 개요

공격자가 클라이언트 스크립트를 악용하여 웹사이트에 접속하려는 일반 사용자로 하여금 공격자가 의도한 명령이나 작업을 수행하는 공격으로, 세션탈취, 웹사이트 위변조, 악성 스크립트 삽입 및 실행, 접근경로 리다이렉트 등의 다양한 공격을 유발할 수 있는 취약점



<프레임 변조를 통한 피싱사이트 접근 유도>

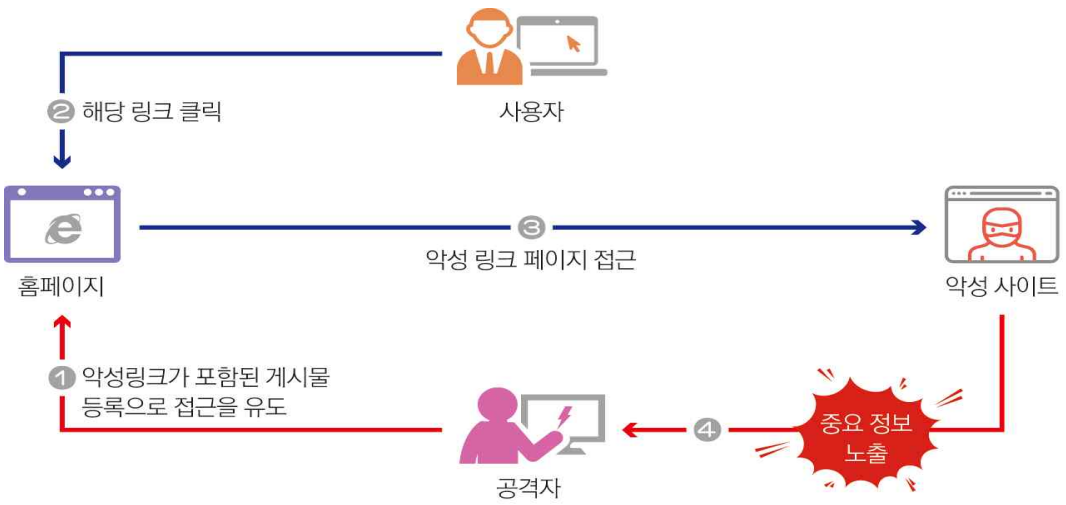
◎ 조치방안

* 악의적인 스크립트가 실행되지 않도록 주요 공격구문에 대하여 HTML 인코딩 적용
* **7절 크로스사이트스크립트(XSS) 취약점 '사전예방조치 방안'** 내용참고

[3-3] 링크 인젝션(크로스 사이트 요청 위조 유도)

◎ 개요

사용자의 접근을 유도하기 위한 목적으로 악의적인 스크립트가 포함된 게시물 또는 링크를 정상적인 것처럼 위조하여 전달할 경우, 이에 접근한 사용자의 민감정보(사용자 명, ID/ Password 등)가 유출될 수 있는 취약점으로, 공격자는 사용자의 세션과 쿠키 조작이 가능하며, 웹 페이지 및 스크립트 파일등을 조작할 수 있는 위험이 존재함



<악성링크 삽입을 통한 비정상사이트 접근 유도>

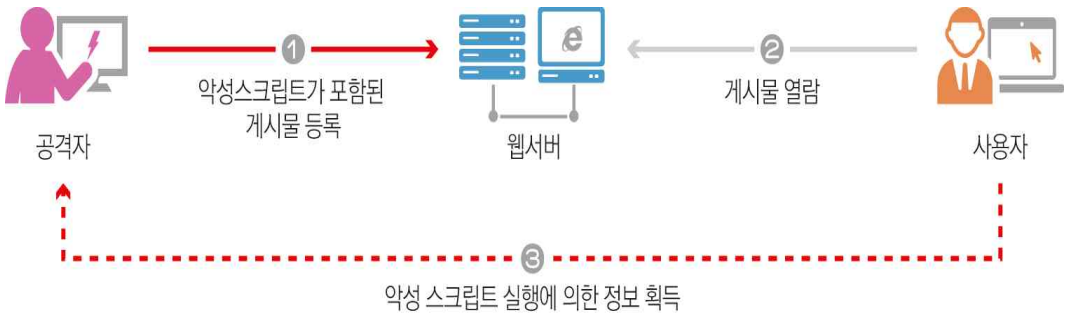
◎ 조치방안

* 악의적인 스크립트가 실행되지 않도록 주요 공격구문에 대하여 HTML 인코딩 적용
* **7절 크로스사이트스크립트(XSS) 취약점 '사전예방조치 방안'** 내용참고

[3-4] 크로스사이트 요청 위조

◎ 개요

사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 하는 취약점으로, 사용자가 웹사이트에 로그인한 상태에서 사이트 간 요청 위조 공격 코드가 삽입된 페이지를 열면, 공격 대상이 되는 웹사이트는 위조된 공격 명령이 믿을 수 있는 사용자로부터 발송된 것으로 판단하게 되어 공격에 노출됨



<악성 링크가 포함된 게시글 열람을 통한 스크립트 실행>

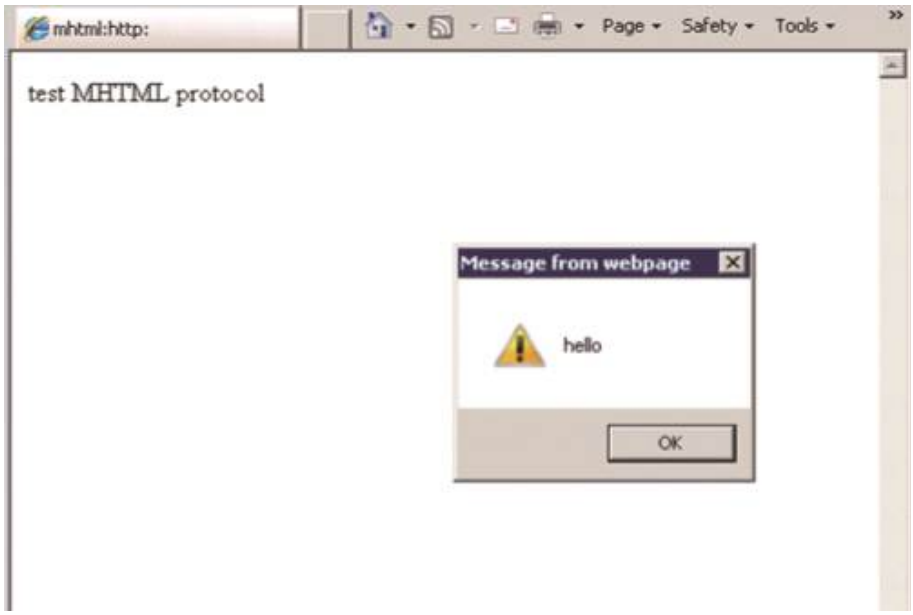
◎ 조치방안

* 정상적인 사용자 여부를 판단하는 프로세스 설계
  - 로그인시 사용자에게 세션 값을 발급(유효기간 설정 및 외부에서 유추할 수 없는 유일한 값)
  - 이후의 모든 서비스 요청 시 발급받은 세션 값을 바인딩 하여 웹서버에 요청
  - 세션 값을 기반으로 정상적인 사용자 요청여부 판별
* 별도의 사용자 인증 체계 도입(전화번호 메시지 승인, 이메일 승인 등)
* **7절 크로스사이트스크립트(XSS) 취약점 '사전예방조치 방안'** 내용참고

[3-5] Microsoft Windows MHTML XSS(Cross-site scripting)

◎ 개요

MHTML은 HTML 웹 페이지가 참고 하는 별도의 그림, 음성파일 등의 동적 콘텐츠를 인코딩하여 HTML 웹페이지에 출력될 수 있도록 제공 되는 기능으로 프로토콜 핸들러에 대한 설정 결함 시, 공격자는 악성 스크립트 실행을 통한 정보 유출 및 사용자의 세션 정보 탈취 등 부가적인 공격에 악용될 수 있는 취약점



<MHTML XSS 실행화면>

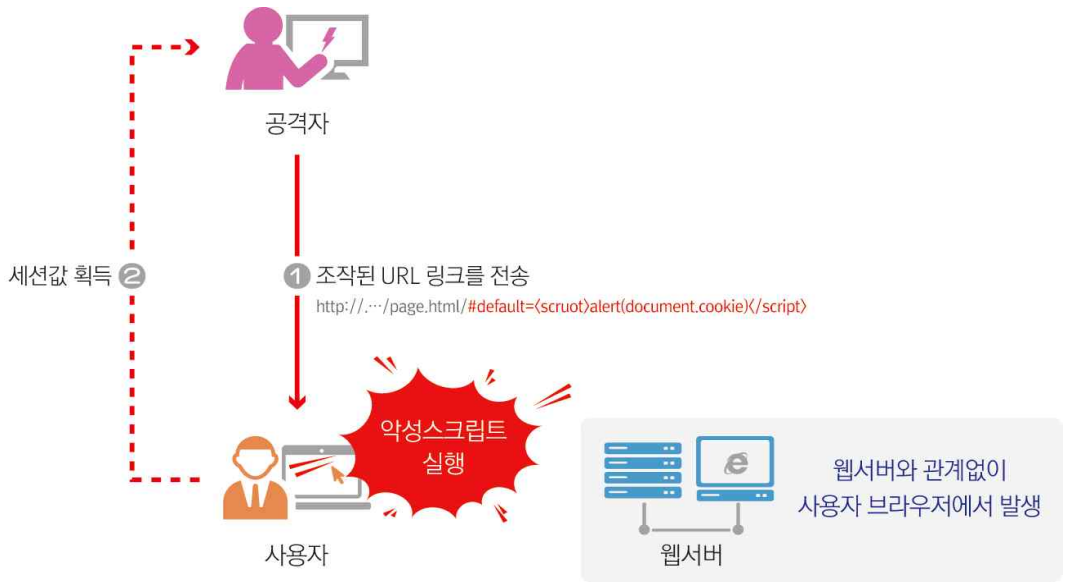
◎ 조치방안

* MS에서 배포하는 최신 버전의 FixIt Tool 설치
* **7절 크로스사이트스크립트(XSS) 취약점 '사전예방조치 방안'** 내용참고

[3-6] DOM 기반 크로스 사이트 스크립팅(XSS)

◎ 개요

DOM(Document Object Model)은 HTML 및 XML 문서를 처리하는 문서객체 모델로, DOM 기반 XSS 취약점이 존재하는 브라우저를 대상으로 변조된 URL을 전송할 경우, 이에 접근하는 사용자가 XSS 공격 피해를 받음



<DOM 기반 크로스사이트 스크립팅>

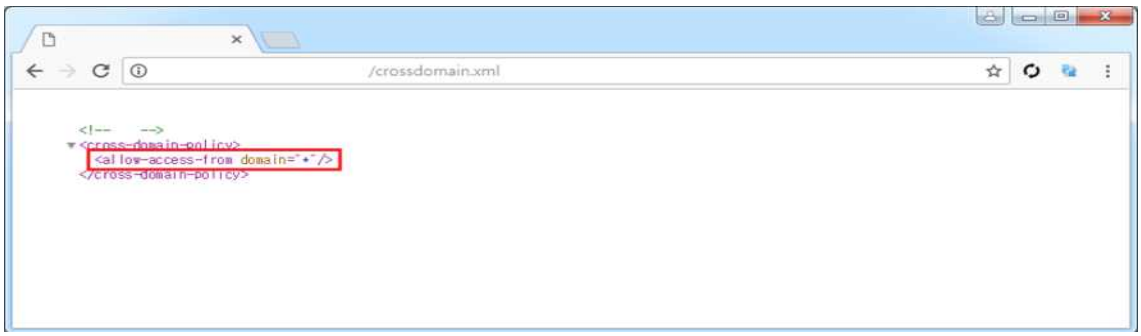
◎ 조치방안

* 민감 정보는 서버 측, 결과 값은 클라이언트 측에서 출력되도록 구현 권장
* **7절 크로스사이트스크립트(XSS) 취약점 '사전예방조치 방안'** 내용참고

[3-7] 호스트가 모든 도메인에서도 플래시 액세스 허용

◎ 개요

crossdomain.xml 파일은 웹페이지의 플래시 애플리케이션에 대해 다른 도메인의 접근을 제어하는 정책 파일이다. 그러나 플래시 애플리케이션에서 모든 도메인의 사이트로부터 리소스 접근을 허용할 경우 합법적인 사용자로 위장한 공격자에게 세션 및 쿠키 탈취를 위한 크로스 사이트 요청 위조 또는 크로스 사이트 추적 등 다양한 변종공격이 가능하다.



<crossdomain.xml 비정상적인 접근 허용 예시>

◎ 조치방안

* 허용한 도메인 또는 IP만 접근할 수 있도록 웹서버 루트 디렉터리 하위에 crossdomain.xml 파일의 allow-access-from 도메인 속성을 수정

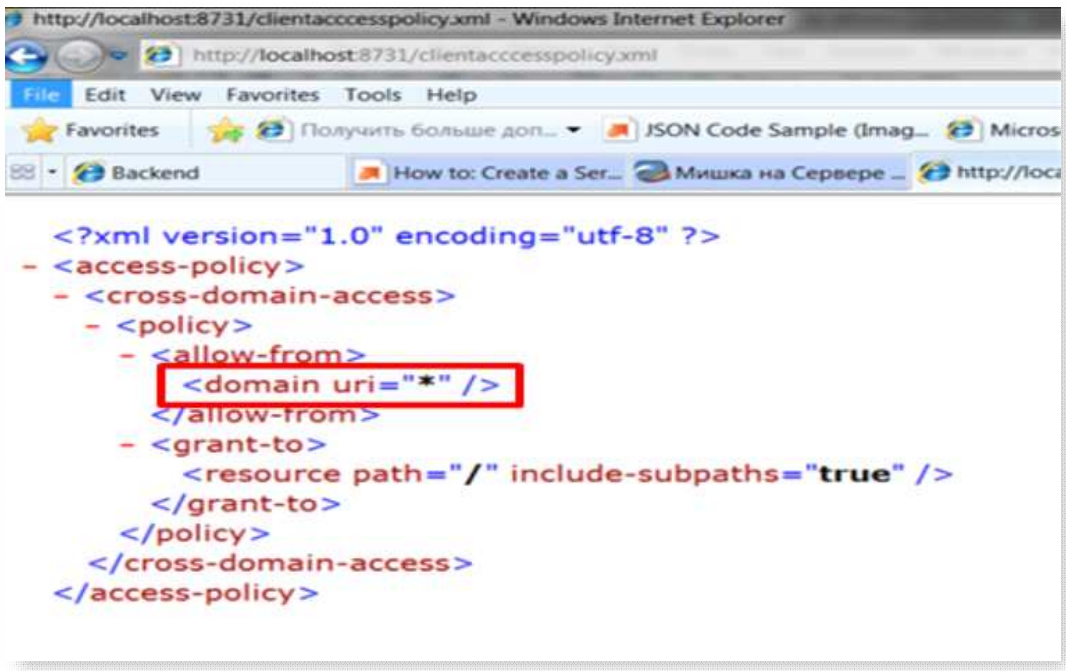
crossdomain.xml

```
<?xml version="1.0"?>
<cross-domain-policy>
    <allow-access-from domain="*.example.com" />
    <allow-access-from domain="www.friendOfExample.com" />
    <allow-access-from domain="192.0.34.166" />
</cross-domain-policy>
```

[3-8] Silverlight : 임의의 도메인으로부터의 액세스 허용

◎ 개요

clientaccesspolicy.xml 파일은 Silverlight를 사용하는 웹 페이지 자원에 대해 다른 도메인의 접근을 제어하는 정책 파일이다. 그러나 애플리케이션에서 모든 도메인의 접근을 허용할 경우 비인가자에 의한 CSRF 및 XSS 공격과 같은 공격에 악용될 수 있다.



<clientaccesspolicy.xml 비정상적인 접근 허용 예시>

◎ 조치방안

* CSRF 및 XSS 공격을 방지하기 위해 인가된 도메인만 접근할 수 있도록 clientaccesspolicy.xml 파일의 allow-from http-request-headers 도메인 속성을 수정

clientaccesspolicy.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <access-policy>
     <cross-domain-access>
         <policy>
             <allow-from http-request-headers="*">
                 <domain uri="http://www.example.com"/>
                 <domain uri="https://www.example.com"/>
             </allow-from>
             <grant-to>
                 <resource path="/" include-subpaths="true"/>
             </grant-to>
         </policy>
     </cross-domain-access>
 </access-policy>
```

제4장 결론

'17년도 2분기에는 외부 사용자가 웹 사이트 입력란이나 URL 매개변수를 통해 입력하는 값의 검증체계가 미흡하여 악성 스크립트의 실행이나 저장을 가능하게 하는 크로스사이트스크립트(XSS) 취약점이 주로 탐지되었다. 해당공격은 외부 공격자에 의해 태그나 공격구문 삽입을 통한 피싱, 세션정보 유출, 악성코드 감염유도 등 다양한 유형의 공격으로 활용할 수 있다. 따라서 보안의식이 낮은 일반 사용자를 대상으로 정상적인 공격으로 위장한 사회공학적인 공격기법과 결합한다면 더욱 치명적인 공격에 악용될 위험성이 높아진다. 이에 홈페이지 운영자는 해당구문이 삽입되지 않도록 필터링을 포함한 차단정책 적용 등 시스템 내·외부 환경에 대한 보호조치를 해야 하며, 사용자도 의심되는 컨텐츠에 접근하지 않도록 각별한 주의가 필요하다.

그 밖에도 비용이나 편의성을 고려하여 공개용 웹 게시판을 사용하는 경우, 외부로 공개된 취약점 정보로 인해 샘플 입력 폼이나 첨부파일 미검증 취약점을 악용하여 시스템 침투까지 허용할 수 있는 취약점이 일부 탐지되었다. 부득이하게 공개용 웹 게시판을 사용할 경우 샘플파일 경로를 반드시 삭제해주어야 하며 불필요한 경우 업로드 기능을 원천차단 하는 것이 바람직하다.

마지막으로 웹 서버의 페이지를 구성하고 있는 소스 내에서 해당 서버와 연계된 내부 IP정보나, 관리자 정보, 접속 정보 등 시스템 침투에 유용한 내부 정보를 유출하는 취약점도 다수 발견되었는데, 이는 초기 웹페이지 구축 시 관리자가 접속정보나 연계 DB 접속정보가 소스에 노출되지 않도록 조금만 관심을 기울이면 충분히 예방할 수 있을 것이다. 이처럼 안전한 웹 환경을 구축하기 위해서는 보안에 관한 지속적인 관심이 요구된다.

참고 자료

[1] http://cwe.mitre.org/data/definitions/434.html

[2] http://cwe.mitre.org/data/definitions/615.html

[3] https://www.owasp.org/index.php/DOM_Based_XSS

[4] MS, https://technet.microsoft.com/library/security/2501696

[5] https://www.asp.net/web-api/overview/security/preventing-cross-site-request-forgery-csrf-attacks

[6] APACHE, https://tomcat.apache.org/tomcat-6.0-doc/config/http.html#SSL_Support