DLL를 이용한 웹페이지 에서의 XSS 대응에 대한 웹 서버 성능 향상 방안

이래홍*, 이희조
*고려대학교 컴퓨터정보통신대학원
e-mail : mrlee@dreamwiz.com

A Study Web Server tuning about Preventing for XSS In Web Page using DLL

Nae-Hong Lee*, Heejo Lee
* Graduate of School Computer and Information Technology,
Korea University

요 약

웹 서비스를 기반으로 구축된 IT 환경에서 Dynamic Web page 를 동작하도록 하는 것이 CGI(Common Gateway Interface)이다. 이런 CGI 를 사용하는 Web page 에서는 XSS(Cross Site Scripting)에 취약점을 가지고 있다. XSS 의 취약점을 이용하여 Web page 의 변조, Cookie 의 가로채기 등의 악의적인 행동으로 인해 많은 피해사례가 있다. 기존의 연구들은 이러한 문제를 해결하기 위해서 게시판 입력 값을 체크하여 Meta character 를 필터링 하는 방법으로 XSS 공격을 대응하였다. 그러나 이러한 방법은 각 페이지 마다 필터링 스크립트를 사용하기 때문에 웹 서버의 성능에 많은 부하를 초래 하는 단점이 있었다. 따라서 본 논문에서는 이러한 웹 서버의 부하를 줄이기 위해 필터링 스크립트를 DLL(Dynamic link library) 화 시켜 모듈화된 함수를 각 페이지에서 호출하여 사용함으로써 웹 서버의 성능 향상을 제안 한다.

1. 서론

최근 대부분의 웹 사이트는 정적인 웹 사이트에서 벗어나 보다 다양한 경험과 정보를 사용자에게 제공하기 위해 스크립트를 사용한 동적인 웹 페이지로 구현되고 있다. 스크립트가 사용자의 웹 경험을 확대시켜준다는 장점이 있긴 하지만 다른 사람의 웹 메일을 엿보는 등의 웹 프라이버시 침해에 대한 우려가 높아지고 있다. 특히 1997년 최초로 발견된 XSS(Cross Site Scripting)는 유명한 웹 메일서버나 웹 서버를 공격할 만큼 진화된 상태이다. XSS 공격은 웹 기반의 이메일 시스템이 HTML 웹 폼으로 구성되어 있고 이메일 메시지 텍스트 부분에 스크립트 삽입이 가능하다는 것을 이용한 공격이다[1, 2].

본 논문의 구성은 다음과 같다. 제 2 장에서는 XSS의 정의 및 공격유형을 제시하고, 3 장에서는 기존 XSS 대응 연구방안과 성능 평가를 기술하였고, 마지막으로 4 장에서는 결론 및 향후 연구방향에 대하여 기술한다.

2. 관련 연구

2.1 XSS 의 정의

소프트웨어의 세큐리티홀의 하나이며, Web 사이트의 방문자의 입력을 그대로 화면에 표시하는 게시판 등의 프로그램이 악의를 가진 코드를 방문자의 브라우저에 보내는 취약성을 말한다. 악의를 가진 유저가 폼 등을 통해서 JavaScript 등의 스크립트 코드를 입력할 때에, 프로그램 쪽에 적절한 체크기능이 없으면, 그 스크립트 내용이 그대로 HTML 에 저장되어 페이지를 열람한 컴퓨터로 스크립트가 실행 되는 경우가 있다.이러한 형태로 페이지에 저장 된 스크립트는 Web 브라우저로는 페이지 작성자 이외의 사람이 저장 한 것이라고 인식할 수 없기 때문에, 브라우저쪽에서 이 문제를 방지하기 위해서는 스크립트를 사용하지 않는다는 설정으로 할 수 밖에 없으며, 스크립트를 사용하는 경우는 항상 이 문제가 일어날 수 있다.악의를 가진 코드를 직접 저장하여 실행 시키는 방법 이외에, 유저가 인식하지 않고 다른 장소에 있는 스크립트를 호출하여 실행 하도록 하는 것이 가능하기 때문에, [크로스사이트]의 이름이 남아있다.스크립트 내용에 따라서는 Cookie 데이터의 도청 등이 가능하므로, 상거래에 사용한 Cookie 를 도청하여, 상거래 본인인 것 처럼 물품의 구입을 하거나, Cookie 를 인증이나 세션관리에 사용하고 있는 사이트에 침입하거나, 보다 광범위한 심각한 피해를 입힐 가능성이 있다.

결과적으로 DOM(Document Object Model) security

restrictions 을 건너뛰어 명령 실행되기 때문에 어떠한 제약도 받지 않고 악의적인 행위가 가능한 것이다. DOM 은 스크립트가 동적 웹 컨텐츠에 수정을 가하는 데에 대한 개념적인 프레임워크로서 웹 브라우저의 보안 셋팅에 의해 구현되어 진다[3, 4].

2.2 XSS 공격 유형

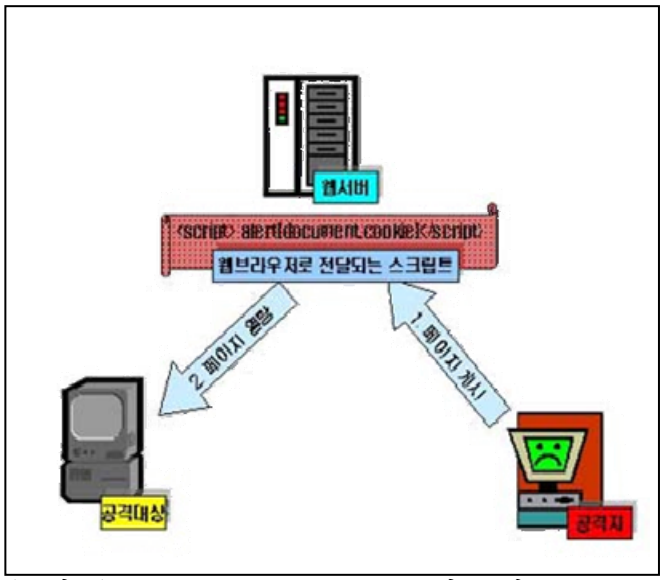
XSS 에는 client-to-client 방식과 client-to-itself 방식의 두 가지 유형이 존재한다.

2.2.1 Client-to-Client 방식

(그림 2)와 같이 한 클라이언트에서 다른 클라이언트로 악의적인 코드가 전달되는 것이다. 게시판에 글을 쓴다든지 하는 방식으로 악의적인 코드를 전달할 수 있다. BBS 나 메시징 시스템 등에 다음과 같이 스크립트가 포함된 내용을 포스팅할 경우에 이 스크립트를 필터링하지 않을 경우에 발생한다.
(그림 1)과 같은 스크립트가 HTML 태그나 스크립트에 포함되어 있는 경우 Client-to-Client 의 예이다

```
<script>
for(i=0; i<100; ++i) {
    alert("Hello, World!");
}
</script>
```

(그림 1) XSS Client-to-Client 예제 스크립트[9,10]



(그림 2) XSS Client-to-Client 공격 구성도

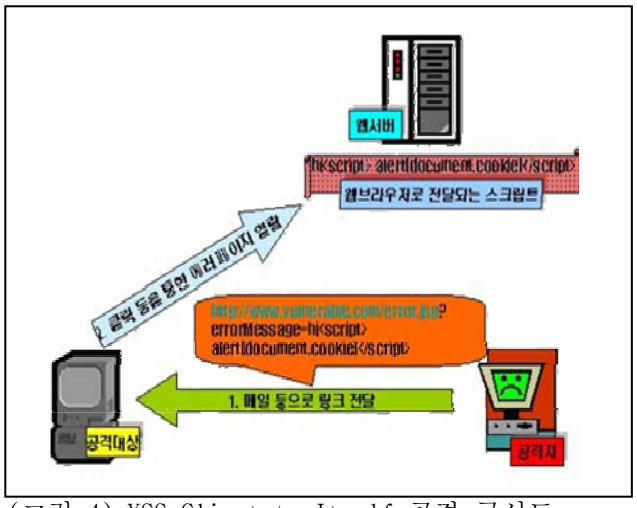
2.2.2 Client-to-Itself 방식

(그림 4)와 같이 악의적인 코드가 공격 대상이 되는 클라이언트 자신이 보내서 자신이 되돌려 받는 유형이다. 이러한 형태의 공격은 주로 이메일 이나 웹 페이지를 통해 링크를 제시하고 사용자가 그러한 링크를 클릭하도록 유도하는 방식으로 이뤄진다.
(그림 3)과 같은 스크립트가 HTML 태그나 스크립트에 포함되어 있는 경우 Client-to-Client 의 예이다.

```
<script>
document.location.replace('http://<URL>
/cookie.cgi?'+docum
ent.cookie+'&'+document.URL);
</script>
```

(그림 3) XSS Client-to-Itself 예제 스크립트

이런 Client-to-Itself 방식을 통해 사용자의 쿠키를 가로채거나 웹 페이지를 변조하는 피해를 준다[4].



(그림 4) XSS Client-to-Itself 공격 구성도

3. 기존 XSS 대응 연구방안 과 성능평가

3.1 기존 XSS 대응 연구방안

사용자 입력으로 사용 가능한 문자들을 정해놓고, 그 문자들을 제외한 나머지의 모든 문자들을 필터링 하여 XSS 를 예방한다.

<표 1> Meta Character 의 Encoding 함수[8]

Meta Character	Encoding
<	< or <
>	> or &#62
&	& or &#38
“	" or &#34
‘	&#39
(	&#40
)	&#41
#	&#35
%	&#35
;	%#59
+	&#43
-	&#45

또한 게시판에서 HTML 포맷을 사용할 수 없도록 설정하는 방법과 필요한 경우 모든 HTML 을 사용하지 못하게 설정 후 필요한 HTML tag 만 쓸 수 있도록 설

정한다[6, 7].

```
job_sum = upload_frm("job_sum")
if job_sum <> "" then
job_sum = replace(job_sum,"'","''")
job_sum = Replace(job_sum,"<","&lt;")
job_sum = Replace(job_sum,">","&gt;")
job_sum = Html2Text(upload_frm("job_sum"))
end if
```

(그림 5) ASP Meta Character 필트링 스크립트[5]

(그림 5) 는 <표 1> 을 참조로 XSS 공격 가능한 Meta Character 를 필터링 하는 ASP 스크립트 예제이다. 기존의 연구들은 (그림 5) 같은 방법으로 각 페이지마다 XSS 필터링 스크립트를 사용하여 웹 서버의 성능에 많은 부하를 초래하였다.

3.2 성능 평가

웹 페이지마다 스크립트를 수행하는 부분을 필터링하여 응답속도가 지연되고 웹 서버의 성능 저하 문제점을 해결하고자 본 논문에서는 DLL 를 이용하여 모듈화된 함수를 각 페이지에서 호출하여 사용함으로써 XSS 의 취약점을 해결함과 동시에 웹 서버 성능의 향상을 제안 하고자 한다.

특정 게시판이 있는 웹 페이지의 필터링 하는 (그림 5) 스크립트를 추가하여 응답속도를 측정하고,다시 (그림 5) 스크립트를 DLL 화 시켜 (그림 6) 과 같이 함수를 호출하는 방식으로 응답속도를 측정한 값의 결과를 비교한다.

3.2.1 대상 시스템

. O/S : Windows2000 Advance Server
. 웹 서버: IIS 5.0
. CPU : 2.8GHz*2
. Memory : 2GByte
. Application : ASP, Visual studio 6
. Test Tool : WAS (Web Application Stress)

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0
'NotAnMTSObject
END
Attribute VB_Name = "Class1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Private Sub Call_Replace()
```

```
    job_sum = upload_frm("job_sum")
    If job_sum <> "" Then
    job_sum = Replace(job_sum, "'", "''")
    job_sum = Replace(job_sum, "&amp;lt;",
"&lt;")
    job_sum = Replace(job_sum, "&amp;gt;",
"&gt;")
    job_sum =
Html2Text(upload_frm("job_sum"))

End Sub
```

```
myobject.dll
Set myO = Server.CreateObject("myObject")
```

(그림 6) DLL 변환 스크립트

3.2.2 실험 방법

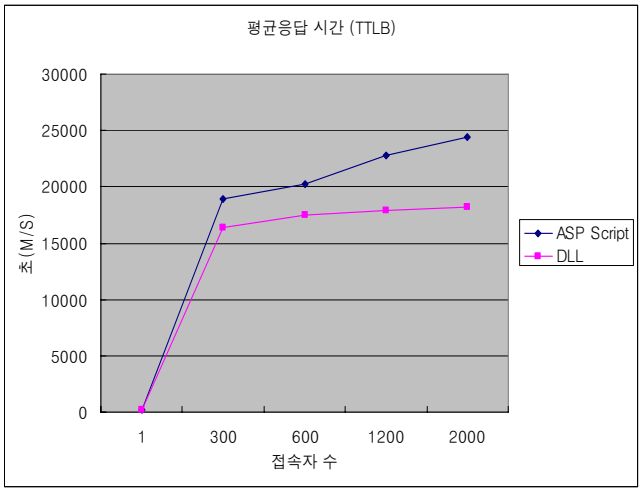
특정 게시판이 있는 웹 페이지를 WAS TOOL 를 이용하여 동시 접속자를 1 명,300 명,600 명,1200 명,2000 명이 동시에 웹 페이지에 30 초 동안 반복 접속했을 때의 응답속도를 측정한다.

3.2.3 실험 결과

<표 2>

RunLevel (접속자수)	반복당 평균 응답 시간(TTLB Avg) 밀리초	
	ASP Script	DLL
1	197	196
300	18899	16389
600	20187	17442
1200	22740	17902
2000	24448	18250

3.2.4 실험 결과



(그림 7)

(그림 7) 은 <표 2> 를 도식화 한 것이다. (그림 7) 에서도 알 수 있듯이 Test 페이지의 기존의 대응방안인 각 페이지마다 ASP 필터링 스크립트 실행했을 때의 응답속도와 DLL 를 사용하여 모듈화된 함수를 호출하였을 때 의 응답속도를 비교해보니 DLL 을 사용하여 모듈화된 함수를 호출 하여 사용하였을 때의 응답속도가 빨라졌음을 확인 할 수 있다.

4. 결론 및 향후 연구

본 논문에서는 웹 페이지에서 XSS 에 대응하는 방법으로 필터링 스크립트 DLL 로 만들어 모듈화된 함수를 각 페이지에서 호출하여 사용함으로써 웹 서버의 성능을 향상 시키는 방법을 살펴보았다. 웹 페이지마다 스크립트를 수행하는 부분을 필터링하는 방법으로 XSS 를 예방할 수 있었고 필터링 패턴이 포함된 DDL 을 사용하여 호출함으로써 처리속도가 빨라짐을 확인할 수 있었다.하지만, 필터링 하는 방법으로 XSS 의 공격으로부터 예방은 할 수 있으나 몇 가지 문제점을 내포 하고 있었다. DLL 을 사용하여 속도의 향상은 있었으나 근본적인 웹 서버의 성능저하 문제와 새로운 패턴의 스크립트 공격에 대해 대비할 수 없는 점, 그리고 DLL 을 사용하면 응용프로그램이 비 독립적이라는 단점이 있다. 이러한 문제점들은 지속적으로 연구 되어야 할 부분이다.

참고 문헌

[1] 이영석 서정석 조상현 “웹 어플리케이션 보안 기술 동향” 한국정보보호학회 pp.83-89 Vol.15 No.1 [2005]
[2] 홍기융 홍기완 박종운 이규호 “웹 서비스 보안 기술 표준화 동향” 한국정보보호학회 pp.19-26 Vol.14 No.4 [2004]
[3] 김성락 “상호연관성 분석을 이용한 웹서버 보안 관리 시스템” 한국컴퓨터정보학회 pp.157-165 Vol.9 No.4 [2004]
[4] 황순일 “웹 해킹 패턴과 대응” 사이텍미디어 [2005]
[5] 한국정보보호진흥원 “웹 어플리케이션 보안 템플릿” 연구보고서 2006 년 6 월
[6] 한국정보보호진흥원 “홈페이지 보안 가이드” 연구보고서 2005 년 4 월
[7] 한국정보보호진흥원 “웹 서버 보안관리 가이드”연구보고서 2003 년 9 월
[8] Johns, M. "SessionSafe: Implementing XSS Immune Session Handling"Vol.- No.4189 [2006]
[9] XSS laps, http://ha.ckers.org/xss.html
[10] XSS laps, http://www.owasp.org/index.php /Cross_Site_ Scripting