FOCUS 4

크로스 사이트 스크립팅(XSS) 공격 종류 및 대응 방법

성윤기

과거 10년전에 비해 사이버 공격의 방법은 지능적이고, 미치는 범위는 급격히 확대되고 있다. 단일 취약점에 대한 스크립트 키디 수준의 공격에서, 이제 조직화된 범죄 단체 또는 국가가 후원하는 조직에서 서버뿐만 아니라 클라이언트 취약점, 스피어 피싱 공격 등 여러 취약점 공격을 자동화하는 악성코드를 배포하여 대규모 감염을 유도하는 공격 등 다양한 플랫폼을 대상으로 복합적인 형태의 공격을 통해 사이버 공격자가 원하는 중요 정보나 금전을 훔치는 공격으로 발전하고 있다.

이중 크로스 사이트 스크립팅 취약점은 취약한 웹 사이트에 악성 스크립트를 포함할 수 있는 손쉬운 방법 중 하나로 공격자들이 가장 많이 선호하는 방식 중 하나이다. XSS 취약점은 OWASP Top 10 – 2013에도 여전히 3번째 높은 위험으로 매겨져 있으며, 취약점분포도도 '매우 광범위'한 수준으로 본 취약점은 조직의 정보 자산을 보호하는데 가장 큰 위험 중 하나라고 할 수 있다.

그래서 본 문서에서는 크로스 사이트 스크립트 취약점, 공격 방법 및 근본적으로 해결할 수 있는 방안에 대해서 살펴본다.

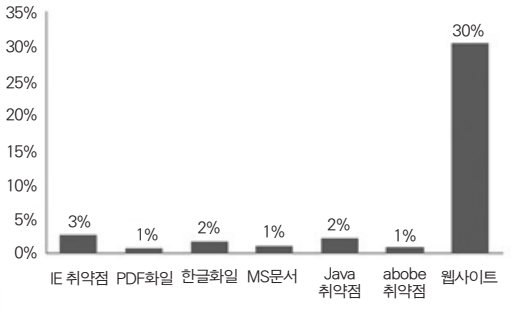
한국인터넷진흥원 종합상황대응팀 책임연구원(yune@kisa.or.kr)

Ⅰ. 개요

1. 사이버 공격 동향

과거 10년 전에 비해 사이버 공격의 방법은 광범위하고 지능적이며, 미치는 범위는 급격히 확대되고 있다. 과거 개별 시스템에 존재하는 취약점에 대한 스크립트 키디 정도의 공격에서, 이제 조직화된 범죄 단체 또는 국가가 후원하는 조직에서 서버뿐만 아니라 클라이언트 프로그램 취약점, 스피어 피싱(Spear phishing)[1] 기법 등 복합적인 공격 방법을 이용하여 사이버 공격자가 원하는 중요 정보나 금전을 훔치는 공격으로 발전하고 있다.

조직에서는 내부 시스템에 대한 비인가 접속 시도 및 공격을 차단하기 위해 다양한 보안 솔루션을 구축함에 따라, 공격자는 내부 사용자의 합법적인 인터넷 사용을 악용하는 방식으로 공격방법을 변경하였다. 즉 사용자가 정상적으로 웹 사이트를 접속할 때 또는 이메일을 읽을 때 공격자가 여기에 악성 스크립트를 포함하여 개인의 PC를 감염시키고, 조직 내부 시스템에 침투하여 원하는 정보를 탈취해 가고 있다. 공격자는 자동화 공격 악성코드를 배포하여 많은 수의 서버를 동시에 감염시키고, 하나의 시스템에 취약점이 발견되면 이를 이용해서 다른 시스템으로 2차 공격하는 등 취약점을 이용하여 넓은 범위와 빠른 속도로 피해가 확산될 수 있다.



주요 전파경로	분포
웹 사이트	286건 (30%)
IE 취약점	25건 (3%)
PDF 파일	7건 (1%)
한글 파일	16건 (2%)
MS 문서	10건 (1%)
Adobe 취약점	8건 (1%)

[그림 1] 악성코드 주요 전파경로별 분류

주: 위 표에 언급된 수치는 3,693개 악성코드에 대한 자체 분석결과로 타 기관 및 업체의 내용과 다를 수 있음
출처: 한국인터넷진흥원 수집 PC 악성코드 분석동향 ('12.1 ~ '13.3)

1) 스피어 피싱(Spear phishing) : 모르는 사람에게 무작위로 피싱 이메일을 보내는 것이 아니라 특정 조직을 공격하기 위해 공격 대상 조직의 중요 시스템 및 정보와 관련 있는 특정인을 대상으로 피싱 이메일을 보내는 방식

[그림 1]과 같이 2012년 1월에서 2013년 3월까지의 한국인터넷진흥원에서 분석한 악성코드 배포 방식을 조사한 결과 웹 사이트를 통해 이용자에게 전파된 경우가 전체의 30%로 대부분을 차지하고 있는 것으로 나타났다. 나머지 악성 문서를 통한 배포가 4%, IE 및 어도비 등 애플리케이션 소프트웨어를 통한 배포가 4%로 나타났다. 즉 웹 사이트의 취약점을 이용한 악성코드 배포가 여전히 인터넷 이용자에게 큰 위협요인이 되고 있다.

본 문서에서는 이러한 합법적인 인터넷 서비스인 웹과 이메일 서비스를 통해 사용자와 조직 내부 공격에 사용할 수 있는 웹 기반 취약점 중 하나인 크로스 사이트 스크립팅(XSS)의 문제점과 해결책을 다룬다. XSS 취약점은 2013년에 OWASP[2] 국제웹보안표준기구에서 발표한 'OWASP Top 10 – 2013'에도 3번째 높은 위험으로 평가 되었으며('OWASP Top 10 – 2010'에서는 2번째 높은 위험), 취약점분포도는 '매우 광범위'한 수준으로, 이 취약점으로 인해 다양한 위험이 발생하고 있다고 할 수 있다. 최근에 이슈가 되고 있는 지능적 위협(APT) 및 워터링 홀[3] 유형의 공격에서도 XSS 취약점과 사회 공학적 기법을 결합하여 공격한다. XSS 취약점을 이용한 공격은 탐지도 어려울 뿐만 아니라, 공격이 성공하는 경우 조직 내부의 PC를 해킹하여 내부 시스템까지 접근할 수 있는 위험이 있다.

그래서 본 문서에서는 이러한 XSS 위험을 효과적으로 줄이고, 이를 활용한 지능적 사이버 공격을 예방하기 위해 XSS 취약점의 원인, 공격 방법 및 근본적인 해결책에 대해서 고찰한다.

2. HTTP 프로토콜

XSS 공격은 웹 응용에 존재하는 취약점을 기반으로 웹 서버와 클라이언트간 통신 방식인 HTTP[4] 프로토콜 동작과정 중에 발생한다. HTTP 프로토콜이란 웹 서버와 클라이언트간 서버에 저장된 웹 문서(일반적으로 HTML[5], 스크립트, 이미지(.gif, .jpeg)등을 클라이언트로 전달하기 위한 표준 규약(RFC[6] 2616)이다. 먼저 클라이언트(브라우저)는 서버로 헤더 정보를 포함하여 관련 정보를 요청(request)하면, 웹 서버는 클라이언트의 요청을 분석하여 요청한 자원 데이터 및 헤더를 포함하여 클라이언트로 응답(Response)을 보낸다.

2) OWASP(Open Web Application Security Project) : 웹 응용 및 소프트웨어 보안 문제를 해결하기 위해 전세계 전문가들로 구성된 비영리 재단. 3년에 1회 'OWASP Top 10'이라는 웹 응용 취약점 위험에 대해서 발표한다.
3) 워터링 홀 공격 : 공격대상 조직의 특징을 빅데이터로 조사하여 구성원들이 많이 방문하는 웹 사이트에 공격 코드를 삽입하여, 공격대상 조직을 감염시키고 침투하는 방법
4) HTTP(HyperText Transfer Protocol) : 인터넷으로 서버와 브라우저간 HTML 문서를 송수신할 수 있도록 요청(request)/응답(response) 통신 규약
5) HTML(HyperTest Markup Language) : 웹 문서를 작성하기 위한 문법 언어. 가장 큰 특징은 문서에 링크를 포함하여 문서간 연결 및 이동을 보장한다.
6) RFC(Request for Comments) : IETF(Internet Engineering Task Force)에서 제정한 인터넷 통신을 위한 표준을 정의한 문서명

```
1 GET /www.server.com/ HTTP/1.1
2 ACCEPT_ENCODING: gzip,deflate,sdch
3 CONNECTION: keep-alive
4 ACCEPT: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 ACCEPT_CHARSET: windows-949,utf-8;q=0.7,*;q=0.3
6 USER_AGENT: Mozilla/5.0 (X11; Linux i686) AppleWebKit/535.1 (KHTML, like Gecko)
    Chrome/13.0.782.24
7 ACCEPT_LANGUAGE: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
8 HOST: www.server.com
9 Cookie2: $Version=1; Skin=new;
```

[그림 2] HTTP 요청 헤더 양식

[그림 2]와 [그림 3]는 브라우저에서 서버(www.server.com)을 접속할 때 발생한 요청 헤더 및 응답 헤더를 보여준다. 브라우저가 서버의 데이터를 요청하고, 서버는 응답 시 헤더의 뒤에는 www.server.com의 첫 페이지 HTML 데이터가 브라우저로 전달되며, 브라우저가 이를 해석하여 사용자에게 보여준다. HTTP 프로토콜의 요청 및 응답 프로토콜 필드의 자세한 사항은 'RFC 2616'에 정의되어 있다.

```
1 HTTP/1.1 200 OK
2 Date: Fri, 08 Jul 2013 00:59:41 GMT
3 Server: Apache/2.2.4 (Unix) PHP/5.2.0
4 X-Powered-By: PHP/5.2.0
5 Expires: Mon, 26 Jul 2013 05:00:00 GMT
6 Last-Modified: Fri, 08 Jul 2013 00:59:41 GMT
7 Cache-Control: no-store, no-cache, must-revalidate
8 Content-Length: 102
9 Keep-Alive: timeout=15, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html
12 Set-Cookie2: UserID=JohnDoe; Max-Age=3600; Version=1
```

[그림 3] HTTP 응답 헤더 양식

XSS 공격은 브라우저로 전달되는 데이터에 악성 스크립트가 포함되어 개인의 브라우저에서 실행되면서 해킹을 하는 것이며, 이 공격용 악성 스크립트는 공격자가 웹 서버에 구현된 웹 애플리케이션의 XSS 취약점을 이용하여 서버 측 또는 URL에 미리 삽입을 해놓은 것이다.

Ⅱ. XSS 취약점 및 공격

1. XSS 취약점 및 현황

웹 애플리케이션 보안 연구재단인 OWASP에 따르면 "크로스 사이트 스크립팅(XSS)는 애플리케이션에서 브라우저로 전송하는 페이지에서 사용자가 입력하는 데이터를 검증하지 않거나, 출력 시 위험 데이터를 무효화 시키지 않을 때 발생한다"라고 정의되어 있다. 즉 공격자가 의도적으로 브라우저에서 실행될 수 있는 악성 스크립트를 웹 서버에 입력 또는 이것을 출력 시 위험한 문자를 중성화시키지 않고 처리하는 애플리케이션의 개발 과정에서 발생한다. XSS는 일반적으로 자바스크립트에서 발생하지만, VB 스크립트, ActiveX 등 클라이언트에서 실행되는 동적 데이터를 생성하는 모든 언어에서 발생이 가능하다.

전 세계 시스템 공격 방법 통계에서도 XSS는 3위 SQL[7] 인젝션(17%)에이어 4위(6.2%)를 차지하는 등 여전히 맹위를 떨치고 있다(출처: The Web Application Security Consortium). 뿐만 아니라 2011년에 발표된 'SANS/CWE 가장 위험한 25大 소프트웨어 오류'[8]에서 4번째로 높은 위험(위험도 77.7점 기록)으로 기록되고 있다.

이와 같이 XSS 취약점은 비교적 쉽게 공격할 수 있으며 웹 애플리케이션 개발 시 제거되지 않아 매우 광범위하게 분포되고 있다고 할 수 있다. 그래서 이 취약점을 이용한 악성 스크립트 배포 및 이를 통한 악성코드 배포 및 클라이언트 프로그램 해킹 등 현재도 개인 및 조직의 보안에 큰 위협이 되고 있다.

---

7) SQL(Structured Query Language) : 관계형 데이터베이스 관리시스템(DBMS)의 데이터를 관리하기 위해 설계된 프로그래밍 언어

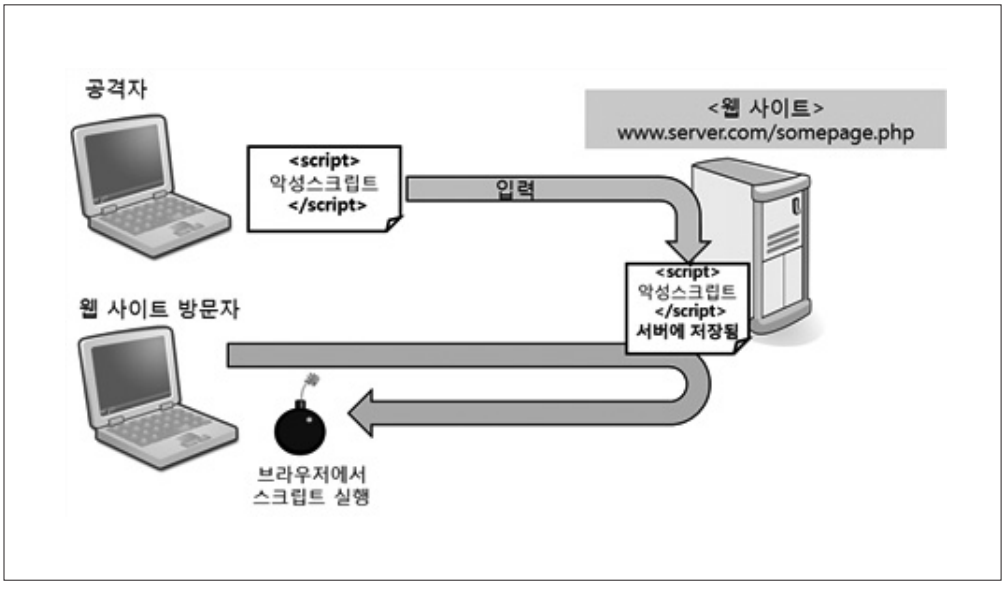
8) 보안연구소인 SANS 연구소와 취약점 데이터베이스목록인 CWE가 소프트웨어 프로그래밍 과정에서 가장 많이 존재하는 취약한 요소를 정의한 문서

2. XSS 공격 종류

그렇다면 공격자가 어떻게 XSS 취약점을 이용하여 공격하는 지 알아보자. XSS 취약점을 이용한 공격 방법은 3가지로 분류된다. 하나는 접속자가 많은 웹 사이트를 대상으로 공격자가 XSS 취약점이 있는 웹 서버에 공격용 스크립트를 입력시켜 놓으면, 방문자가 악성 스크립트가 삽입되어 있는 페이지를 읽는 순간 방문자의 브라우저를 공격하는 방식이며(저장 XSS 공격), 또 하나는 반사 XSS 공격으로 악성 스크립트가 포함된 URL을 사용자가 클릭하도록 유도하여 URL을 클릭하면 클라이언트를 공격하는 것이고(반사 XSS 공격), 마지막으로 DOM 환경에서 악성 URL을 통해 사용자의 브라우저를 공격하는 것이다(DOM 기반 XSS 공격).

1) 저장 XSS 공격

저장 XSS 공격은 [그림 4]와 같이 웹 애플리케이션 취약점이 있는 웹 서버에 악성 스크립트를 영구적으로 저장해 놓는 방법이다. 이 때 웹 사이트의 게시판, 사용자 프로필 및 코멘트 필드 등에 악성 스크립트를 삽입해 놓으면, 사용자가 사이트를 방문하여 저장되어 있는 페이지에 정보를 요청할 때, 서버는 악성 스크립트를 사용자에게 전달하여 사용자 브라우저에서 스크립트가 실행되면서 공격한다.



[그림 4] 저장 XSS 공격 방법

가장 일반적인 방법은 게시판 같은 곳에 HTML 문서에 〈script〉를 이용하여 이 스크립트 태그 안에 악성 스크립트를 저장하는 방식이다. 즉 텍스트만 표시되도록 설계된 어떤 게시판에 〈script〉 "악성 스크립트" 〈/script〉과 같은 태그를 포함한다. 이 경우에 게시판에는 태그가 나타나지 않으며 사용자는 확인할 수가 없다. [그림 5]는 브라우저의 쿠키 값을 보여주는 간단한 스크립트이며, 어떤 웹 페이지에〈script〉alert(document.cookie)〈/script〉가 포함되어 있는 어떤 페이지를 사용자가 읽을 때 마다, 브라우저는 이 스크립트를 실행하면서 쿠키 값을 보여주게 된다. 공격자는 'alert(document.cookie)' 스크립트 대신 정교한 공격용 코드를 포함하여 다양한 공격을 수행할 수 있다.

```
〈script〉alert(document.cookie)〈/script〉
```

[그림 5] XSS 용 자바스크립트

저장 XSS는 공격자 입장에서 사용자들이 많이 방문하는 사이트가 공격 대상으로 가장 적합한 곳이다. 즉 유명 온라인 게시판, 웹 기반 이메일 및 사용자 프로필 등에 악성 스크립트를 포함하면, 다른 방문자들이 해당 페이지를 읽어보는 즉시 악성 스크립트가 브라우저에서 실행되면서 감염되므로 효과적이다.

2) 반사 XSS 공격

반사식 XSS 공격은 웹 애플리케이션의 지정된 변수를 이용할 때 발생하는 취약점을 이용하는 것으로, 검색 결과, 에러 메시지 등 서버가 외부에서 입력받은 값을 받아 브라우저에게 응답할 때 전송하는 과정에서 입력되는 변수의 위험한 문자를 사용자에게 그대로 돌려주면서 발생한다.

일반적으로 서버에 검색 내용을 입력하면, 검색 결과가 있는 경우에는 결과 값을 사용자에게 전달하지만, [그림 6]과 같이 서버에서 정확한 결과가 없는 경우 서버는 브라우저에 입력한 값을 [그림 7]과 같이 그대로 HTML 문서에 포함하여 응답한다. 이 경우 HTML 페이지에 포함된 악성 스크립트가 브라우저에서 실행이 된다.

```
http://www.server.com/search/?q=〈script〉alert(document.cookie)〈/script〉&x=0&y=0
```

[그림 6] 반사 XSS 공격용 URL

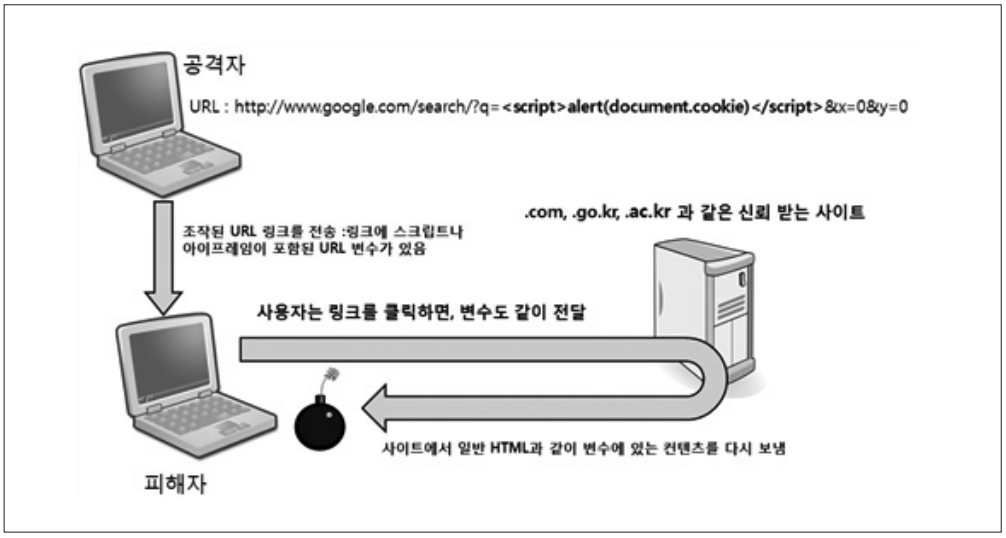
```
〈html〉
〈body〉
〈div id="pageTitleTxt"〉
〈h2〉〈span class="highlight"〉Search Results〈/span〉〈br /〉
Search: "〈script〉alert(document.cookie)〈/script〉"〈/h2〉
〈/body〉
〈/html〉
```

[그림 7] 서버가 반사한 HTML 데이터

즉 사용자가 서버로 입력 한 값을, 서버는 요청한 사용자의 브라우저로 악성스크립트를 반사시킨다. 반사 XSS 공격은 주로 사용자에게 악성 URL을 배포하여 사용자가 클릭하도록 유도하여 클릭한 사용자를 바로 공격한다. 즉 사용자는 악성 스크립트가 포함된 링크를 클릭한 순간 바로 악성 스크립트가 사용자의 브라우저에서 실행된다. 반사 XSS 공격은 이메일 메세지 또는 다른 웹 사이트와 같이 다양한 경로로 피해자 시스템에게 전달된다.

일반적인 반사 XSS 공격 단계는 다음과 같다.

1) 공격자는 먼저 A사이트에 XSS 취약점이 있는 것을 발견한다.
2) 민감한 정보를 획득할 수 있는 공격용 악성 URL을 생성한다.
3) 공격자는 이 URL을 이메일 메시지에 포함하여 배포한다.
4) 피해자가 URL을 클릭하면, 바로 공격 스크립트가 피해자로 반사되어 A 사이트에 관련된 민감한 정보(ID/패스워드, 세션 정보)를 공격자에게 전송한다.



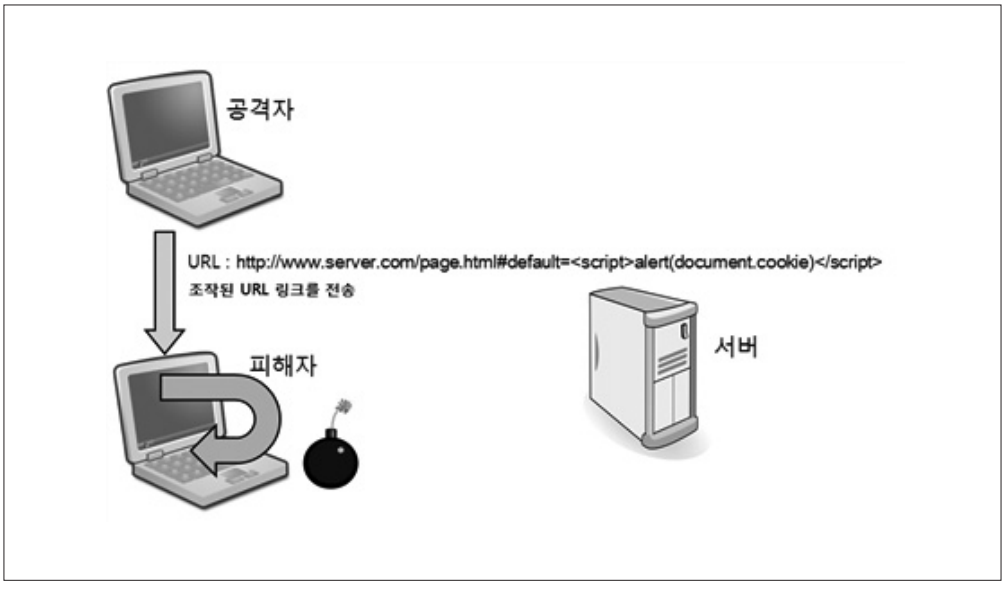
[그림 8] 반사 XSS 공격 방법

3) DOM 기반 XSS 공격

DOM(Document Object Model)이란 W3C 표준으로 HTML 및 XML 문서에 접근방법을 표준으로 정의하는 문서객체모델이다. W3C[9]에서는 DOM을 '프로그램 및 스크립트가 문서의 컨텐츠, 구조 및 형식을 동적으로 접근 및 업데이트할 수 있도록 하는 언어 중립적인 인터페이스이다'라고 정의되어 있다. DOM은 HTML 문서를 계층적으로 보면서 컨텐츠를 동적으로 변경할 수 있다.

DOM 기반 XSS 공격은 2005년에 처음으로 아밋 클라인(Amit Klein)이 관련 취약점 논문을 발표하면서 알려졌다. 피해자의 브라우저가 HTML 페이지를 구문 분석할 때마다 공격 스크립트가 DOM 생성의 일부로 실행되면서 공격한다. 페이지 자체는 변하지 않으나, 페이지에 포함되어 있는 브라우저측 코드가 DOM 환경에서 악성코드로 실행된다.

앞에서 다룬 저장 XSS 및 반사 XSS 공격의 악성 페이로드가 서버 측 애플리케이션 취약점으로 인해, 응답 페이지에 악성 스크립트가 포함되어 브라우저로 전달되면서 공격하는 것인 반면, DOM 기반 XSS는 서버와 관계없이 브라우저에서 발생하는 것이 차이점이다.



[그림 9] DOM 기반 XSS 공격 방법

9) W3C : World Wide Consortium 으로 웹의 창시자 팀버너스 리가 웹 표준을 제정하기 위해 만들 비영리 재단

일반적으로 DOM 기반 XSS 취약점 있는 브라우저를 대상으로 조작된 URL을 이메일을 통해 사용자에게 전송하면, 피해자는 이 URL 링크를 클릭하는 순간 공격 피해를 입게 된다.

```
<HTML>

<TITLE>Welcome!</TITLE>

Hi

<script>
 var pos=document.URL.indexOf("name=")+5;
 document.write(document.URL.substring(pos,document.URL.length));
</script>

<br>

Welcome to our system

This demo borrowed from http://www.webappsec.org/projects/articles/071105.shtml

</HTML>
```

[그림 10] DOM 스크립트

[그림 10]과 같이 개발된 DOM 페이지의 name 변수에 아래와 같이 변수를 입력하면 (http://www.server.com/page.html?name=David) 정상적으로 동작한다. 하지만, DOM 기반 XSS 공격을 위해서 [그림 11]와 같이 입력을 하면, 브라우저에서 <script> 의 내용이 실행되게 된다.

```
http://www.server.com/page.html?name=<script>alert(document.cookie)</script>
```

[그림 11] DOM 기반 XSS 공격 URL

아래 [그림 12]는 DOM 기반 XSS 공격을 예방기법을 회피하기 위한 방법으로 브라우저에서 '#' 문자 뒤에 있는 값을 서버로 전송하지 않는 것을 이용하지만, 브라우저에서는 동일하게 실행이 된다.

```
http://server/page.html?name=David#<script>alert(document.cookie)</script>
```

[그림 12] DOM 기반 XSS 공격 예방 우회 URL

3. XSS 공격의 피해

본 고에서 언급한 <script>alert(document.cookie)</script> 스크립트는 쿠키 값을 출력하는 악성 스크립트이지만, 본 스크립트 대신 다양한 공격용 코드로 대체하면 쿠키 정보 및 세션 정보 획득, 클라이언트 프로그램 해킹 등 다양한 방법으로 클라이언트 시스템을 공격할 수 있다.

1) 쿠키 정보/세션 ID 획득

쿠키란 웹 서버가 HTTP 헤더 중 Set-Cookie 필드로 브라우저에게 보내는 4KB 이하의 작은 텍스트 파일이며, 사용자가 웹 사이트를 이용하는 동안 사용자 브라우저에 저장된다. 사용자가 웹 사이트의 페이지를 클릭할 때마다 브라우저는 웹 서버에게 사용자의 상태를 다시 알려준다. 사용자 상태를 기록하기 위해 쿠키 값에 로그인, 버튼 클릭 등에 대한 정보를 저장한다.

세션 쿠키는 사용자가 웹사이트를 읽거나 방문하는 동안에만 임시로 메모리에 존재하는 쿠키이다. 쿠키 생성 시 쿠키 만료시기 또는 유효성 기간이 설정되어 있지 않은 경우에 세션 쿠키가 만들어 진다. 브라우저에서는 사용자가 브라우저를 종료하면 세션 쿠키를 삭제한다.

웹 애플리케이션이 세션 ID를 쿠키에 포함하는 경우 XSS 공격을 통해, 클라이언트의 합법적인 세션 ID를 획득하여 불법적으로 정상 사용자로 가장할 수 있다.

2) 시스템 관리자 권한 획득

XSS 취약점을 이용하여 사용자 브라우저 취약점을 공격하여 PC를 완전히 통제할 수도 있다. 공격자는 XSS 취약점 있는 웹 서버에 다양한 악성 데이터를 포함시켜 놓은 후, 사용자의 브라우저가 악성 데이터를 실행하는 경우 자신의 브라우저 있는 제로데이 취약점 또는 패치되지 않은 취약점을 공격하는 공격 코드가 실행되면서 사용자 시스템을 완전히 통제할 수 있다. 회사 등 조직의 개인의 PC가 해킹되는 경우, 조직의 내부 시스템으로 이동하여 내부의 중요 정보를 탈취하는 공격으로 이어질 수 있다.

3) 악성코드 다운로드

XSS 공격은 악성 스크립트 자체로만으로는 악성 프로그램을 다운로드 할 수 없지만, 사용자가 악성 스크립트가 있는 URL을 클릭하도록 유도하여 악성 프로그램을 다운로드 받는 사이트로 리다이렉트(redirect) 하거나, 트로이목마 프로그램을 다운로드하여 설치할 수 있다.

Ⅲ. XSS 취약점 예방 기술

XSS 취약점은 쉽게 악용될 수 있으며, 공격 효과도 커 공격자들이 자주 이용하는 기술이다. 많은 조직에서 XSS 공격을 대응하기 위해 웹 방화벽(WAF)을 도입하여 방어를 하고 있으나, 대부분의 웹 방화벽은 시그너쳐 기반의 XSS 공격만을 탐지하고 있다. 하지만 특정 문자열을 탐지하는 기술은 쉽게 우회가 가능하여 방어가 효과적이지는 못하다. 또한 웹 애플리케이션 개발자는 〈script〉 태그 등 브라우저에서 실행되는 위험한 문자를 중성화 하여 XSS 취약점을 예방하고 있다.

하지만 웹 개발자들이 일일이 수동적으로 위험한 문자를 필터링 및 인코딩 하는 것은 현실적으로 불가능하며, 취약점이 잔존하게 된다. 이러한 방식으로 인해 중소 및 대형 웹 사이트 등에서 XSS 취약점이 지속적으로 발견되고 있다.

1. 입 · 출력 값 검증 및 무효화

XSS 취약점을 근본적으로 제거하기 위해서는 스크립트 등 해킹에 사용될 수 있는 코딩에 사용되는 입력 및 출력 값에 대해서 검증하고 무효화시켜야 한다. 입력 값에 대한 유효성 검사는 데이터가 입력되기 전에 가능하면, 입력 데이터에 대한 길이, 문자, 형식 및 사업적 규칙 유효성을 검사해야 한다.

출력 값을 무효화하기 위해서는 XSS 공격은 기본적으로 〈script〉 태그를 사용하기 때문에 XSS 공격을 차단하기 위해 태그 문자(〈, 〉) 등 위험한 문자 입력 시 문자 참조(HTML entity)로 필터링하고, 서버에서 브라우저로 전송 시 문자를 인코딩하는 것이다. HTML 문자 참조란 ASCII 문자[10]를 동일한 의미의 HTML 문자로 변경하는 과정이다. 예를 들어, 문자 "〈"는 동일한 의미의

10) ASCII(American Standard Code for Information Interchange)문자는 컴퓨터에서 사용되는 문자의 코딩 값이다.

HTML “<”로 변경한다. HTML 엔터티는 대부분의 인터프리터(특히, 브라우저)에서 특수한 의미를 가지지 않으며, 단순한 문자로 처리된다. 이렇게 인코딩하면 사용자는 〈script〉가 〈script〉로 보이지만 HTML 문서에서는 <script>로 나타나서 브라우저에서 일반 문자로 인식하고 스크립트로 해석되어 실행되지는 않는다.

ASCII 문자	참조 문자	ASCII문자	참조 문자
&	&amp;	"	&quot;
〈	&lt;	'	&#x27;
〉	&gt;	/	&#x2F;
(	&#40;	)	&#41;

[그림 13] 위험 문자 인코딩

[그림 13]은 HTML 문서에서 악성 스크립트에 포함되어 브라우저에서 실행될 수 있는 문자와 대체 문자를 정리한 것이다. 악성 스크립트는 많은 HTML 태그안에 포함을 할 수 있으므로 반드시 [그림 13]에 있는 위험문자의 경우 출력 값을 이스케이핑 해야 한다.

CSS의 DIV 엘리먼트에 다음과 같이 악성 자바 스크립트를 숨길 수 있다.

〈DIV STYLE="background-image: url(javascript:alert(document.cookie))"〉

그래서 HTML 속성(attributes)에 들어가는 값도 &#xHH (HH는 16진수 값)으로 반드시 인코딩 해야 한다.

아이프레임의 경우 다음과 같이 악성 URL을 포함시켜 악성 자바스크립트가 포함된 페이지를 사용자가 읽으면서 브라우저를 공격할 수 있다.

〈iframe src="악성 URL" width="0" height="0" frameborder="0"〉〈/iframe〉

[그림 14]는 악성 스크립트 등 악성 데이터를 포함되어 브라우저를 공격할 수 있는 다양한 방법의 HTML 태그를 정리해 놓은 것이다. 여기에 있는 ‘악성 데이터’. ‘악성 URL’ 및 ‘악성 HTML’ 악성 자바 스크립트 등이 포함이 될 수 있는 곳이다. 즉 웹 애플리케이션 개발자들은 웹 문서를 코딩 시 [그림 14]에 있는 태그에 포함될 악성 데이터에 대해서 반드시 입 · 출력 값을 검증해야 한다.

데이터 형식	콘텍스트	코딩 예	방어책
String(문자열)	HTML Body	〈span〉 악성 데이터 〈/span〉	● HTML Entity 인코딩
String(문자열)	안전한 HTML Attributes	〈input type="text" name="fname" value="악성 데이터"〉	●HTML Entity 인코딩 ●악성 데이터를 화이트리스트 방식 또는 안전한 속성에만 위치 ●background, id 및 name과 같은 안전하지 않은 속성을 철저히 검증
String(문자열)	GET 파라미터	〈a href="/site/search?value=악성 데이터"〉clickme〈/a〉	●URL 인코딩
String(문자열)	SRC 또는 HREF 속성에 악성 URL	〈a href="악성 URL"〉clickme〈/a〉 〈iframesrc="악성 URL" /〉	●입력 값 정규화 ● URL 검증 ● URL 안전성 확인 ● 화이트 리스트된 http 및 https URL 만 허용(새로운 창을 열기 위해 자바스크립트 프로토콜 사용금지) ● 속성 인코더 사용
String(문자열)	CSS 값	〈div style="width: 악성 데이터;"〉Selection〈/div〉	● 엄격하게 구조적 검증 ● CSS 16진수 인코딩 ● CSS 기능 설계 강화
String(문자열)	자바스크립트 변수	〈script〉var currentValue='악성 데이터';〈/script〉 〈script〉함수('악성 데이터');〈/script〉	● 자바스크립트 변수를 문자화 ● 자바스크립트 헥스 인코딩 ● 자바스크립트 16진수 인코딩 ● 자바스크립트 유니코드 인코딩 ● 백슬래쉬(\", \' 또는 \\)사용금지
HTML	HTML Body	〈div〉악성 HTML〈/div〉	HTML 검증 (JSoup, AntiSamy, HTML Sanitizer)
String(문자열)	DOM XSS	〈script〉document.write("악성 입력 값: " + document.location.hash);〈script/〉	●DOM 기반 XSS 예방

[그림 14] XSS 예방 기법

하지만 애플리케이션 개발자가 많은 태그의 입력 문자를 검증하기 위해 코딩 시 일일이 작업하는 것은 많은 노력과 자원이 소모된다. 또한 인코딩 방식을 통해 방어기술을 무력화할 수 있으므로 애플리케이션 개발자 직접 모두 처리하는 것은 근본적으로 불가능하다. 그래서 입 · 출력 값을 자동적으로 검증해주는 라이브러리를 사용하면 좀 더 효과적으로 대응할 수 있다.

2. 보안 라이브러리

1) AntiXSS

AntiXSS 라이브러리는 마이크로소프트사에서 개발한 공개용 XSS 취약점 예방 라이브러리이다. AntiXSS 라이브러리는 ASP.net 애플리케이션 개발환경에서 사용되며, 현재 ASP.net 4.5 이후 버전에만 사용이 가능하다.

이 라이브러리는 입력 값을 검증하여 서버로 악성 스크립트로 입력되지 못하는 기능과 위험한 문자를 인코딩하는 함수를 제공한다. [그림 15]은 AntiXSS에서 제공하는 위험한 데이터를 출력 시 인코딩해주는 메소드이다.

메소드	사용처	코딩 예
HtmlEncode	신뢰할 수 없는 데이터가 HTML 바디 부분에 출력되는 경우	〈span〉[악성 데이터]〈/span〉 〈p〉Hello [악성 데이터]〈/p〉
HtmlAttrEncode	HTML 속성(attributes)에 신뢰할 수 없는 데이터를 추가해야 하는 경우	〈input type="text" name="fname" value="[악성 데이터]"〉 〈p id="[악성 데이터]"〉〈/p〉
UrlQueryEncode	URL 내에 쿼리 문자열 값에 신뢰할 수 없는 데이터를 추가해야 하는 경우	〈a href="/site/search?value=[악성 데이터]"〉 clickme〈/a〉 JavaScriptEncode
JavaScriptEncode	신뢰할 수 없는 데이터를 자바스크립트 변수에 지정할 경우	〈script〉var currentValue='[악성 데이터]';〈/script〉 〈script〉someFunction('[악성 데이터]');〈/script〉
XmlEncode	XML 데이터 부분에 신뢰할 수 없는 데이터를 출력하는 경우	〈name〉[악성 데이터]〈/name〉
XmlAttrEncode	XML 속성 값에 신뢰할 수 없는 데이터를 추가해야 하는 경우	〈name firstName="[악성 데이터]"〉〈/name〉
GetSafeHtml	HTML 바디에 신뢰할 수 없는 HTML을 출력하는 경우	〈div〉[악성 HTML]〈/div〉

[그림 15] AntiXSS의 XSS 예방 인코딩 메소드

2) OWASP ESAPI 라이브러리

OWASP는 포괄적인 애플리케이션 보안을 위해 웹 응용 취약점을 대응할 수 있는 오픈소스 ESAPI 라이브러리를 개발하여 제공하고 있다. ESAPI에는 총 14개의 API가 있으며, 이 중 XSS 취약점을 예방하기 위해 API는 validator와 encoder가 있다. validator는 입력 값을 필터링하는 기능을 하고 있으며, encoder는 출력 값을 인코딩 및 디코딩 기능을 가지고 있다.

이 라이브러리는 자바, PHP, .NET, ASP, 자바스크립트 및 파이썬 등 다양한 애플리케이션 개발 언어를 지원하고 있다.

3. 브라우저 확장 프로그램

애플리케이션 개발 시 이용하는 라이브러리 이외에도 사용자가 XSS 공격을 예방할 수 있는 프로그램도 있다. NoScript는 파이어폭스 등 모질라 기반의 브라우저에서 실행되는 오픈 소스 확장 프로그램으로, 화이트 리스트 기반으로 신뢰된 사이트의 자바스크립트, 플래쉬, 실버라이트 및 액티브X 등 동적 스크립트만 브라우저에서 실행하도록 하여 XSS 공격을 예방할 수 있다.

Ⅳ. 결론

본 문서에서는 많은 웹 사이트에서 발견되고 있고, 활발하게 공격에 이용되는 XSS 취약점, 공격 방법 및 취약점 제거 방법에 대해서 살펴보았다. XSS 취약점은 문제가 되고 있는 악성 URL 배포를 통해 PC를 해킹하는 데 자주 사용될 뿐만 아니라, 피싱 문제, 인증 데이터 획득 등으로 사용자 및 조직에 심각한 위험을 초래할 수 있다.

개발단계에서 XSS 취약점을 제거하지 않고서는 다양한 XSS 형태의 공격과 우회 공격을 방어하기는 힘들다. XSS 취약점을 근본적으로 해결하기 위해서는 애플리케이션 개발단계에서 위험한 데이터 입 · 출력을 검증하고 인코딩하는 방법을 선택해야 한다. 또한 각 조직에서는 XSS 취약점을 효과적으로 제거하기 위해 웹 애플리케이션 및 소프트웨어 개발 시 공개된 다양한 오픈소스 라이브러리를 조직에 맞게 사용하여 XSS 취약점을 근본적으로 제거하는 노력과 투자가 필요하다.

참고문헌

심재홍, "금융 소비자를 위협하는 악성코드 위협사례 분석", 인터넷시큐리티 포커스 5월호, 한국인터넷진흥원 2013. 5월

OWASP Top 10 2013 –가장 심각한 웹 애플리케이션 보안 위험 10가지, OWASP 재단, 2013년 8월

웹 해킹 사고 통계, The Web Application Security Consortium, 2013년 11월

SANS 연구소/MITRE, SANS/CWE 가장 위험한 25大 소프트웨어 오류, 2010년 6월

OWASP 재단, XSS (Cross Site Scripting) Prevention Cheat Sheet

마이크로소프트, Anti-Cross Site Scripting Library V4.2, 마이크로소프트

OWASP 재단, OWASP Enterprise Security API