# 접근 제어를 이용한 교차 사이트 스크립트 필터링

김형주[0] 예홍진
아주대학교 정보통신전문대학원
조은선
충북대학교 전기전자컴퓨터학부
{cutehjk, hjyeh}@madang.ajou.ac.kr, cmohan@hitel.net

# Malicious Cross Site Script Filtering Using ACL

Hyung-Ju Kim[0] Hong-Jin Yeh
Graduate School of Information and Communication, Ajou University
Eun-Sun Cho
Chungbuk National University

요 약

최근 웹 메일은 사용과 관리의 편리함 등으로 그 사용이 점점 더 늘어나고 있다. 그러나 웹 메일은 본래 보안 기능을 중시해서 만들지 않았다는 보안 취약점으로 인해 불법적인 해킹이나 도청 같은 악의적인 공격의 대상이 되고 있다. 특히 e-메일과 HTML, 스크립트 언어들을 함께 사용할 수 있게 된 이후로 스크립트를 이용한 인터넷 범죄가 증가하고 있다. 본 논문은 스크립트 공격 중 상호 교차된 사이트 스크립트 공격에 대해서 기술하고 이 공격의 실행을 성공적으로 막을 수 있는 대응 방안을 제시한다.

## 1. 서 론

최근 대부분의 웹 사이트는 정적인 웹 사이트에서 벗어나 보다 다양한 경험과 정보를 사용자에게 제공하기 위해 스크립트를 사용한 동적인 웹 페이지로 구현되어 있다. 스크립트가 사용자의 웹 경험을 확대시켜준다는 장점이 있긴 하지만 다른 사람의 웹 메일을 엿보는 등의 웹 프라이버스 침해에 대한 우려가 높아지고 있다.

특히 1997년 최초로 발견된 교차 사이트 스크립팅(Cross Site Scripting: CSS)기술은 현재 핫메일과 같은 유명 웹 메일을 공격하는 만큼 만연돼 있는 상태이다. CSS 기술은 웹 기반 e-메일 시스템이 HTML 웹 폼으로 구성되어 있고 e-메일 메시지 텍스트 부분에 스크립트 삽입을 허용한다는 것을 이용한 공격이다. 대부분 e-메일을 통해 사용자에게 보내는데 이 기술은 인터넷 사이트와 웹 애플리케이션에 악성 스크립트를 삽입해 브라우저를 통해 이동한 다른 사이트가 이를 인증하지 못할 때 다른 웹 페이지로 링크를 걸고 악성 코드를 실행하게 된다. [1]

본 논문에서는 구체적으로 웹 메일을 통한 CSS의 악성 행위에 대해 알아보고 그 중 자바스크립트를 이용한 CSS에 대해 중점을 두고 있다. 그리고 자바스크립트의 자체적으로 제공되는 보안 기술에 대해 논한 후, 웹 메일의 CSS 공격에 대응하는 진보된 필터링 기법에 ACL을 적용한 새로운 방안에 대해 기술하였다.

## 2. 관련 연구

### 2.1. 교차 사이트 스크립트의 악성행위

웹 메일은 웹 브라우저만으로도 별도의 프로그램 없이 e-메일을 주고 받을 수 있어 각기 다른 많은 기업들이 제공하고 있는 인기 있는 서비스이다. 전형적으로 웹 메일의 보안은 HTTP 쿠키들에 의해서 다뤄진다. 대부분의 웹 메일 사이트들은 초기 시 사용자가 이름과 패스워드를 입력하면 고유한 인증 정보를 쿠키에 담아 브라우저에게 보내어지고 그 후 접속 시에는 웹 서버와 브라우저 간에 오가는 쿠키에 의해 인증이 이루어 진다. 이 쿠키 값을 악성 스크립트를 담고 있는 웹 메일을 통해 얻어 올 수가 있다. [2] 메일을 여는 즉시 악성 스크립트가 작동하면서 악성 행위를 하는 사이트로 이동 될 수 있고 사용자의 시스템의 정보를 공격자에게 보낼 수 있게 된다.

```
<script>
  var allCookies = document.cookie
  location = "http://www.cracker.org/cgi-bin/
  collector.cgi?data=allCookies"
</script>
```

그림 1 악성 사이트로 이동하는 예제

또 다른 악성 행위의 예를 들면, e-메일에 포함된 하이퍼링크를 클릭했을 경우 스크립트가 작동하도록 하여 개인 또는 네트워크의 하드드라이브를 손상시키는 행위를 할 수 있게 된다. 이와 같은 웹 메일을 통한 CSS 공격은 첨부 파일 없이 메일을 여는 것만으로 시스템을 감염시키거나 서비스 거부 공격 생성, 개인 정보 유출, 의도치 않는 웹사이트 접근 같은 악성 행위가 이루어질 수 있다.

```
<a  onmouseover=" window.open(http://malicious-site)"
href= " Safe-site"  target=" _blank" >
Here </a>
```

그림 2 서비스 거부 공격 예제

### 2.2. 악성 자바스크립트 객체와 메소드, 속성

주로 악성 행위에 쓰이는 자바스크립트의 객체와 메소드를 정리하면 다음과 같다.

- window 객체의 open(), alert(), setTimeout() 메소드와 ' onLoad' 이벤트 핸들러는 주로 일정한 시간 간격으로 새로운 창을 계속 열어주는데 사용된다.
- document 객체의 ' href' 속성과 ' onClick' 의 이벤트 핸들러는 문서의 URL을 알아내는데 사용 된다.
- form 객체의 ' mailto' 속성과 ' submit' 메소드는 form에 입력된 내용을 서버에 전달하는데 사용된다.
- link객체의 ' onMouseDown' ,' onMouseOver' 이벤트 핸들러 는 마우스를 눌렀을 때 악성 행위를 하는 사이트로 연결되게 하는 행위를 할 수 있다.

## 3. 필터링 기법을 통한 CSS 공격에 대한 대응 방안

앞에서 살펴 보았던 CSS의 공격에 대한 대응책을 살펴보겠다. 우선 자바스크립트 자체적으로 가지고 있는 보안 정책을 알아본다. 그리고 보다 유연성 있는 CSS 공격에 대한 대응 방안을 위해 LiveConnect 기술을 이용하여 구성한 접근 제어 리스트(Access Control Lists) 필터링 기법에 대해서 제시한다.

### 3.1. 자바스크립트의 보안 기법

자바 스크립트는 클라이언트 측에서 해석되는 스크립트 언어로 Netscape에서 개발하였다. Netscape는 ' policy' 라는 용어 아래 보안 메커니즘을 가지고 있는데 이는 브라우저나 스크립트 언어로 인한 잠재적인 문제와 침입 요청을 다룰 수 있는 방법을 의미한다. Netscape는 두 가지의 다른 보안 정책인 동일 출처 정책과 서명된 스크립트 정책이 있다.

#### 3.1.1. 동일 출처 정책(same origin Policy)

동일 출처 정책은 Navigator 2.0부터 디폴트 정책이었다. 한 출처로부터 문서를 적재할 때는, 다른 출처에서 로드 된 스크립트는 윈도우나 프레임, 객체(그림, 레이어, 주소, 창, 문서) 에서 속성을 가져오거나 설정할 수 없다. [3]

#### 3.1.2. 서명된 스크립트 정책(signed script policies)

Navigator 4.0부터 새로 나온 서명된 스크립트 정책은 새로운 자바 보안 모델인 객체 서명을 토대로 한다. 자바스크립트의 새로운 정책을 사용하려면, 새로운 자바 보안 클래스를 사용하고 자바스크립트로 만든 스크립트들에 서명한다. [3]

#### 3.1.3. 진보된 보안 기법

자바 스크립트 1.2에서 주어진 스크립트에 대한 신뢰는 동일 출처 정책 (Same Origin Policy)에 의해 통제가 된다. 하지만 만약 같은 회사에 속한 두 도메인들이 서로 신뢰관계를 성립시키려면 동시에 불가능하다. 그러므로 보다 유연한 정책을 위해서 ACL(Access Control List: ACL)을 이용하면 스크립트에 명시된 어떤 도메인이 사용자의 컴퓨터를 접근하려고 하는 지를 알 수 있고 악성 코드를 다운로드 받거나 문제를 일으킬 소지가 있는 행위를 제한시킬 수 있다.

### 3.2. 제안하는 대응 기법

일반적으로 사용자들이 할 수 있는 악성 자바스크립트의 대응 방안으로는 웹 메일 서비스 옵션에서 '자바스크립트 허용여부' 등을 선택할 수 있다면 허용하지 않는 것으로 설정하는 것이다. 그러나 사용자가 자바 스크립트를 해제시키더라도 이 기능을 해제하지 않은 다른 사용자가 e-메일을 받을 경우에는 자바스크립트의 악성 행위의 효력을 발휘한다.
또 다른 웹 메일의 사이트에서 할 수 있는 기본 대책은 수신된 웹 메일의 본문에 포함된 스크립트 언어 소스의 ' <' 와' >' 태그를' &lt' 와' &gt' 로 인코딩하여 HTML 태그의 기능을 해제시키는 것이다. 하지만 그림 3과 같이' a href' 값에 스크립트가 삽입된 경우 사용자의 의도와 달리 악성 데이터를 다른 사이트를 통해 전송하게끔 하는 HTML 코드는 실행이 되어버린다.

```
<a href =<script src=' http://malicisious-
site/steal_password.js' ></script>">
Click here! </a>
```

그림 3 대응 원리

#### 3.2.1. 제안된 대응 원리

위와 같은 취약성을 보완하기 위해 본 연구에서는 그림 4와 같은 2단계의 대응 방안을 제안한다. 이는 서버 단계와 클라이언트 단계로 구성되어 있다.

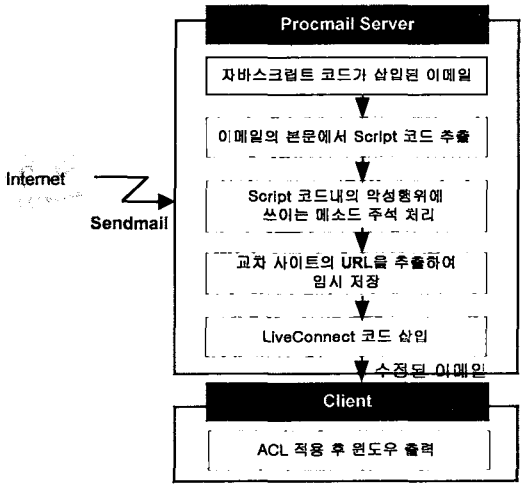


그림 4 시스템 구성도

● 서버 단계

이 단계는 인터넷을 통해 메일을 프락메일(Procmail) 서버에서 받았을 때 클라이언트에게 보내기 전에 메일의 내용을 검사하고 필터링에 필요한 코드를 삽입한다. 프락메일은 메시지의 헤더와 본문에서 정의한 정보(필터)를 이용해서 적절한 조치를 취할 수 있는 기능을 제공한다. [8] 이 단계에서 프락메일 서버를 이용하는 주된 이유는 보다 e-메일의 Body 추출이 쉽고

html의 에러율이 낮으며 메일처리에 자유로움을 가져다 주기 때문이다. 또한 Sendmail server에서의 overhead를 줄일 수 있다는 장점을 가지고 있기 때문이다. 서버에서 받은 e-메일의 메시지 부분 소스에서' <script' 와' /script>' 부분만을 추출하여 그림 5과 같이 주석처리를 하고 자바스크립트 기능을 해제시킨다.

```
<script language="javascript">
<!--
function open_window()
{
   {}<!--window.open("http://malicious-site/",
     width=50,height=50); //-->
}
//-->
</script>
```

그림 5 주석 처리

그리고 2장에서 언급한 CSS 공격에 쓰일 수 있는 자바스크립트의 메소드, 속성, 이벤트 핸들러를 찾아서 URL 부분을 추출하여 임시 저장한다. 그 다음에 LiveConnect을 이용한 ACL을 적용한다.

```
<APPLET code="LiveConnect.class" name=liveapplet
code=LiveApplet MAYSCRIPT width=250 height=100>
</APPLET>
```

그림 6 LiveConnect에서의 애플릿 설정

LiveConnect 사용을 위해 NAME 태그를 줘야 자바 스크립트에서 애플릿 객체를 알 수 있고 MAYSCRIPT는 자바 스크립트에게 이 애플릿에 접근할 권한을 준다.

```
<FORM NAME= form1
onsubmit=document.liveapplet.setURL(document.form1.str.value); return false;>
<INPUT TYPE=text SIZE=28 Name=str value="자바스크립트의 URL-Path">
<INPUT TYPE=button VALUE=URL
onclick=document.liveapplet.setURL(document.form1.str.value)>
</FORM>
```

그림 7 자바 스크립트에서 자바 애플릿 메소드 호출하는 부분

```
JSObject win  = null;
win  = JSObject.getWindow(this);
if (URL-Path == ACL)
win.eval("alert(₩"This is a unsafed URL!!₩");");
else
getAppletContext().showDocument (URL, "_blank");
```

그림 8 자바 애플릿에서 자바 스크립트 함수를 호출하는 부분

● 클라이언트 단계

이 단계는 실질적으로 수정된 메일을 클라이언트가 볼 수 있는 단계이다. e-메일은 그림 9와 같이 e-메일 Body부분에 LiveConnect 를 이용한 코드가 삽입되어 만약 스크립트 태그 내의 URL 패스 값이 ACL로 이루어진 데이터베이스의 리스트와 일치한다면 사용자에게 제한된 사이트라는 경고창이 뜰 것이고 그렇지 않다면 이 URL 값을 가진 창이 새로이 뜰 것이다.

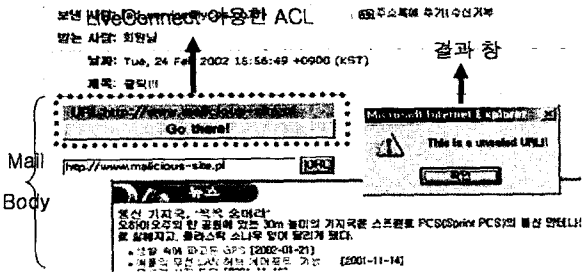


그림 9 구현 결과

## 4. 결론 및 향후 발전 방향

인터넷 상의 가장 많은 통신 수단인 e-메일이 보안 기능이 잘 갖춰져 있다면 가장 신뢰성 있는 통신수단으로 많은 업무들을 전자우편을 통해서 처리할 수 있게 될 것이다. 우리의 생활 필수품이 되어가고 있는 e-메일의 보안을 위해 본 연구에서는 웹 메일에서 발생할 수 있는 악성 CSS 공격에 대한 대응 방안을 제시하였다. 또한 HTML 인코딩이 아닌 필터링 기법을 써서 자바스크립트 기능을 해제시켰고 LiveConnect 기술을 사용하여 자바와 자바스크립트간의 통신을 가능하게 하였다. 그리고 접근 제어 리스트(Access Control Lists)을 사용함으로써 자바스크립트의 보안 정책에 보다 유연성 있는 접근을 가져오게 했다.

향후 과제로서는 자바스크립트뿐만 아니라 CSS 공격을 할 수 있는 악성 코드로 확대해 나가는 것이다. 본 논문의 구현을 통해 LiveConnect 기술이 CSS로 인한 악성 행위를 같은 행위를 할 수 있음을 확인함으로써 LiveConnect에 관한 보안에 대해서도 연구해 보고자 한다.

## 5. 참고 문헌

[1] 코드 라인 하나로 사이트 전체를 해킹한다?, http://korea.internet.com/channel/content.asp?cid=216&nid=16342, Sep 2001

[2] CERTCC-KR 권고문: KA-2000-41-악성코드에 의한 HTTP Cookie 유출 문제점 및 대책

[3] IT 보안 해설서 - 응용 프로그램 보안, http://members.tripod.lycos.co.kr/hayonia/it13-applications.html#Heading99

[4] Netscape, "LiveConnect", Netscape web site, http://home.netscape.com/navigator/v3.0/liveconnect.html, 1997

[5] Vinod Anupam, Alain Mayer, Secure Web Scripting

[6] Roger A. Grimes, Malicious Mobile Code, O' Reilly

[7] Flavio De Paoli, Andre L. Dos Santos and Richard A. Kemmerer, "Vulnerability of "Secure" web browsers", to appear in 20th National Information Systems Security Conference, October 1997

[8] 프락메일(Procmail)를 이용한 E-mail 보안, http://www.certcc.or.kr/cvirc/Alert/54/procmail.htm