

1. 기사 내용

제목	네트워크의 위험 : SNMP 통한 지속적 XSS 공격, 너무 쉽다
기사 날짜	2016년 9월 8일
스크랩 담당자	손우규
요약	<ul style="list-style-type: none">• 9개 네트워크 관리 시스템 조사한 결과 13가지 취약점 발견• 네트워크 관리 시스템, 기기 대 기기 통신 잘 검사하지 않아 <p>현존하는 네트워크 관리 시스템 대부분 웹용 인터페이스를 가지고 있고, 시스템을 트래킹하고 관리하는 데에 있어 SNMP를 디폴트로 사용한다. 환경설정에 따라 기기의 SNMP 데이터를 자동으로 받을 수도 있다.</p> <p>이걸 공격자의 시각에서 보면, 한 마디로 네트워크 관리 시스템 만큼 이상적인 공격 목표가 없다. 거의 실시간으로 네트워크에 연결된 각종 요소에 대한 정보가 최신화된다는 게 가장 크다. 공격자 입장에서는 친절한 안내자이자 상세한 이정표다. 개인정보가 들어있는 인사부 서버와 같이 '당연한' 것들 외에 각종 재무표가 인쇄되고 있는 프린터처럼 간헐적으로 중요한 일이 진행되고 있는 '덜 눈에 띄는' 표적들도 공격이 가능해진다.</p> <p>네트워크 관리 시스템은 네트워크에 새롭게 등장한 기기에서 나오는 정보를 거의 무조건적으로 신뢰한다. 제대로 확인하는 절차를 거치지 않는다는 것이다. 공격자로서는 악성 기기를 쉽게 연결할 수 있게 된다. 그리고 네트워크 관리 시스템의 물려터진 인증과정을 통해 네트워크에 연결이 되면 그 기기를 통해 지속적인 XSS 페이로드를 전송할 수 있다.</p>
출처	http://www.boannews.com/media/view.asp?idx=51742&page=1&kind=4&search=title&find=xss

2. 용어

용어	설명
SNMP	간이 망 관리 프로토콜(Simple Network Management Protocol, SNMP)은 IP 네트워크상의 장치로부터 정보를 수집 및 관리하며, 또한 정보를 수정하여 장치의 동작을 변경하는 데에 사용되는 인터넷 표준 프로토콜이다. SNMP를 지원하는 대표적인 장치에는 라우터, 스위치, 서버, 워크스테이션, 프린터, 모뎀 랙 등이 포함된다.
XSS	사이트 간 스크립팅(또는 크로스 사이트 스크립팅, 영문 명칭 cross-site scripting, 영문 약어 XSS)은 웹 애플리케이션에서 많이 나타나는 취약점의 하나로 웹사이트 관리자가 아닌 이가 웹 페이지에 악성 스크립트를 삽입할 수 있는 취약점이다. 주로 여러 사용자가 보게 되는 전자 게시판에 악성 스크립트가 담긴 글을 올리는 형태로 이루어진다.

3. 관련 기술(XSS)

3.1 개요

사이트 간 스크립팅(또는 크로스 사이트 스크립팅, 영문 명칭 cross-site scripting, 영문 약어 XSS)은 웹 애플리케이션에서 많이 나타나는 취약점의 하나로 웹사이트 관리자가 아닌 이가 웹 페이지에 악성 스크립트를 삽입할 수 있는 취약점이다. 주로 여러 사용자가 보게 되는 전자 게시판에 악성 스크립트가 담긴 글을 올리는 형태로 이루어진다. 이 취약점은 웹 애플리케이션이 사용자로부터 입력 받은 값을 제대로 검사하지 않고 사용할 경우 나타난다. 이 취약점으로 해커가 사용자의 정보(쿠키, 세션 등)를 탈취하거나, 자동으로 비정상적인 기능을 수행하게 할 수 있다. 주로 다른 웹사이트와 정보를 교환하는 식으로 작동하므로 사이트 간 스크립팅이라고 한다.

SQL injection과 함께 웹 상에서 가장 기초적인 취약점 공격 방법의 일종으로, 악의적인 사용자가 공격하려는 사이트에 스크립트를 넣는 기법을 말한다. 공격에 성공하면 사이트에 접속한 사용자는 삽입된 코드를 실행하게 되며, 보통 의도치 않은 행동을 수행시키거나 쿠키나 세션 토큰 등의 민감한 정보를 탈취한다.

크로스 사이트 스크립팅이란 이름 답게, 자바스크립트를 사용하여 공격하는 경우가 많다. 공격 방법이 단순하고 가장 기초적이지만, 많은 웹사이트들이 XSS에 대한 방어 조치를 해두지 않아 공격을 받는 경우가 많다. 여러 사용자가 접근 가능한 게시판 등에 코드를 삽입하는 경우도 많으며, 경우에 따라서는 메일과 같은 매체를 통해서도 전파된다.

3.2 공격 형태

3.2.1 스크립트 태그

방법 : 스크립트 태그로 자바스크립트를 실행한다.

예제 : `<script>alert('XSS');</script>`

설명 : 스크립트 태그의 스크립트를 실행시킨다. 안타깝게도, 매우 정직한 방법이라 대부분의 사이트에서 막는 경우가 많다. 브라우저단에서 필터링 해주는 경우도 있다. 물론 예외도 있다. 하지만 만들어진지 몇십 년 이상 되었거나, 무작정 막지도 않는 경우도 있다.

3.2.2 자바스크립트 태그

방법 : 링크 태그로 자바스크립트를 실행한다.

예제 : `XSS`

설명 : 브라우저에서 about: 링크와 같이, javascript: 로 시작하는 링크는 스크립트를 실행시킨다. 스크립트 태그와 같이, javascript: 를 필터링하는 경우가 많아 많은 사이트에서 막는다.

3.2.3 이벤트 속성

방법 : 이벤트 속성을 사용한다.

예제 : ``

설명 : 이벤트 속성으로 스크립트를 실행할 수 있다. 주로 on 으로 시작하는 속성이 이벤트 속성이다. 자주 사용되는 이벤트 속성으로는 onload onerror onclick 등이 있다. 물론, 이 방법 역시 '자바스크립트 링크' 방법만큼 많이 막혔다.

3.2.4 블랙리스트 우회

방법 : 알려지지 않은 태그와 속성들을 사용한다.

예제 : `<ruby oncopy="alert('XSS')">XSS</ruby>`

설명 : 블랙 리스트 방식으로 막는 사이트에 사용할 수 있다. 이벤트 속성 목록을 참고하자. 위의 방법들 보다는 적게 막혔으나, 여전히 최근 웹사이트들에선 화이트리스트 방식 차단이 대부분이라, 막혔을 가능성이 높다.

3.2.5 내용 난독화

방법 : 따옴표로 감싸는 문자열 사이에 공백 문자들을 넣고, HTML 인코드를 하여 난독화한다.

예제 : `<a`

`href="javas
cript
:
alert
('XSS')">XSS`



설명 : 일부 브라우저에서 javascript: 링크 사이에 공백 문자가 들어갈 수 있고, HTML 인코드를 해도 디코드된 내용이 출력된다는 점을 이용한다. 여기에서는 '자바스크립트 링크' 방법과 사용하였지만, 당연히 다른 방법과 함께 사용할 수 있다.

3.2.6 스크립트 난독화

방법 : <http://utf-8.jp/public/aaencode.html>에서 자바스크립트 난독화.

예제 : <https://namu.wiki/w/XSS/aaaenocde> 참조.

설명 : 스크립트를 일본어를 사용한 이모티콘들로 난독화한다. 스크립트 실행은 가능하지만, document.cookie 와 같은 단어를 막을 경우 사용하면 된다.

3.3 공격 원인

웹페이지가 사용자에게 입력받은 데이터에 대해 적절한 인코딩이나 필터링 없이 동적으로 생성된 웹페이지에 포함하여 사용자에게 재전송 할 때 발생하며, 성공할 경우 공격자는 클라이언트 측에서 실행되는 악성 스크립트 코드를 실행할 수 있게 허용하여 세션 가로채기, 웹 사이트 변조, 악의적 콘텐츠 삽입, 피싱 등의 공격을 할 수 있는 위험이 존재한다.

HTTP 요청에 대해 동적으로 HTML을 생성하는 어플리케이션(CGI)이 공격대상이 된다. 정적인 홈페이지는 이 문제가 발생하지 않는다.

3.4 보완 방법(해결방안)

XSS는 데이터를 입력할 때와 출력할 때, 모두 필터링하고, 클라이언트에도 막을 수 있을만한 수단을 구성해놓는 것이 좋다. 보안은 한없이 덧대도 끝이 없고, 아래 서술한 방식으로 구성해도 기어코 뚫어버리는 사람들이 존재하기 때문이다.

3.4.1 입력 필터

자바는 적당한 XSS 필터를 만든 뒤, web.xml에 선언하여 모든 파라미터가 해당 필터를 거치도록 하는 것 만으로도 좋은 효과를 볼 수 있다.

PHP는 입력을 처리할 때, 정규식을 이용하는 preg_replace를 사용하거나, 노드를 이용하는 DOMDocument를 사용할 수 있다. 속도는 preg_replace가 더 빠르지만, 정규식의 경우 예외와 오류가 종종 발생하기 때문에, 더 안전한 DOMDocument를 사용하는 것이 권장된다.

3.4.2 필터 제작

단순히 텍스트만 입력시키거나 출력하는 데 필터를 제작할 때, 주로 필터되는 것이 <와 >이다. 각각 < 와 > 처럼 HTML 문자로 바꾸어서 HTML 코드가 아닌 단순 문자로 인식하게 하는 것이다. 하지만, 이 경우는 모든 HTML 태그를 막아버리기 때문에, 각종 스타일을 적용시켜야하는 사이트에서는 맞지 않을 수도 있다.



3.4.3 라이브러리 제작

각종 스타일을 적용시켜 글을 꾸며야 하는 사이트의 경우, 스크립트 태그와 이벤트 속성, javascript: 링크만 제대로 막아주어도 상당수를 막을 수 있을 것이다. 하지만, EMBED 태그를 이용하여 저런 기법을 사용하지 않고도 XSS를 먹이는 경우가 있으므로, 다른 것들도 모두 막아주어야 한다.

이 때, 일부 태그 및 속성만 막기 보다는, 반대로 일부 태그 및 속성만 허용하게 하는 것이 좋다. 몇몇 태그만 막는다면 HTML 태그나 속성이 추가될 수록 주기적으로 필터를 수정해줘야 하는데, 이럴 경우 HTML 표준을 주기적으로 꼼꼼히 챙겨보지 않는 한은, 업데이트를 하는 사이에 이미 해커가 침투해놓고 유유자적하게 빠져나갔을 수도 있다.

3.4.4 라이브러리 이용

기존에 있던 라이브러리를 가져다 사용해도 좋다. 개인이나 소규모 단체가 만든 것 보다는, 전문적인 보안 업체나 거대 기업에서 만든 것이 공신력있고 편하기 때문이다. 사용할만한 것으로는 OWASP Antisamy이나 NAVER Lucy XSS Filter, ESAPI 등을 이용하는 방법이 있다.

3.4.5 BBCode 이용

글을 꾸며야 하지만, 따로 필터를 만들긴 어려운 경우, BBCode를 사용할 수도 있다. 만들기도 다른 방법에 비해 편하고, 안전성도 높은 편이기 때문에 가장 추천되는 방법이다. 또한 <나 >같은 HTML 코드를 단순 문자로 바꿀 수 있기 때문에 XSS가 실행될 염려가 적다. 하지만 이 때, 정규식과 같이 단순히 문자열 치환으로 수정할 경우, XSS가 발생할 수도 있다.

3.4.6 출력 필터

HTML5부터 iframe의 sandbox 옵션으로 iframe 내의 자바스크립트, 폼과 같은 것의 제한이 가능해졌으므로, 위의 방법을 실행한 뒤 2차적으로 써보는 것도 좋다. 단, 어디까지나 2차로만 사용하는 것이 좋다. 구버전의 웹 브라우저라면 호환이 되지 않아 스크립트가 그대로 실행되거나, 설령 그게 아니더라도 웹 브라우저에서 취약점이 발견될 경우, 이런 방법이 뚫릴 수 있기 때문이다.

iframe은 단순히 다른 웹사이트를 불러오는 것만이 아니라, 부모 창에서 마음대로 수정할 수 있기 때문에, 무식하게 다른 페이지를 불러오며 AJAX나 PJAX 등을 포기하며 사용하지 말자. 물론, 기존부터 라이브러리를 사용하지 않고 웹 페이지를 개발하는 사람은 물론이고, jQuery같은 라이브러리를 사용하는 사람들도 조금만 검색해보면 간단하게 iframe의 내용을 수정할 수 있는 방법이 나오니 도전해보도록 하자.

3.4.7 쿠키의 보안 옵션 사용

쿠키 생성시 '보안 쿠키'라는 파라미터를 지정하면 TLS 상에서만 사용하게 할 수 있으며, 'HTTP ONLY'라는 파라미터로 웹 브라우저상에서만 쓸 수 있게 할 수도 있다. 물론 완전히 방어가능한 건 아니니 위의 해결법과 병행하는 것이 좋다.

3.4.8 기타

위 방법들과는 별개로 일반 사용자가 스크립트를 잘 알지 못하는 것을 악용하여, 사용자가 브라우저 콘솔에 악성 스크립트를 삽입해 실행되도록 속하기도 하는데, 이를 스스로 하는 XSS라는 뜻의 Self XSS라고 부른다. 이는 사용자가 스스로 실행한 것이니만큼 위 방어 방법으론 막기 어려우니, 잘 알지 못하는 스크립트 등은 절대로 실행해서는 안 된다.



4. 참고 자료

- <https://namu.wiki/w/XSS>
- https://ko.wikipedia.org/wiki/%EC%82%AC%EC%9D%B4%ED%8A%B8_%EA%B0%84_%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8C%85