정 보 보 호 학 회 지
제24권 제4호, 2014. 8

HTML5 차세대 웹표준 환경에서의 보안 이슈

강 석 철*, 박 정 섭**

요 약

HTML5는 차세대 웹문서 표준으로서, 텍스트와 하이퍼링크만을 표시하던 HTML(Hyper Text Markup Language)이 멀티미디어 등 다양한 어플리케이션까지 표현・제공하도록 진화한 "웹 프로그래밍 언어"이다. ActiveX 등 비표준 기술의 남용 및 이의 방치로 인해서 초래된 웹의 갈라파고스화와 접근성 취약문제를 해결하고, 글로벌 OS・플랫폼 업체에 종속될 우려가 있는 ICT 중소 벤처기업에게 돌파구를 제시하기 위해 HTML5가 등장하였다.

HTML5는 2014년 8월 경 최종 표준이 확정될 것으로 전망되나, 현재 웹브라우저 기업을 비롯한 글로벌 업체들은 오디오・비디오・그래픽 등 표준화가 많이 진행된 기능을 중심으로(특히 모바일 영역에서) HTML5를 적극 도입하여 사용 중이다.

하지만 HTML5에서는 지금까지 존재하지 않았던 다양한 웹보안 취약점들이 나타날 수 있다. 대부분의 웹사이트들은 신규 웹보안 위협에 대부분 취약할 것으로 예상되며, 이에 대한 정부차원의 대응이 필요할 것으로 보인다.

본고에서는 HTML5에서 발생 가능한 보안 위협을 설명하고 이에 대한 대응방법을 제안하고자 한다.

Ⅰ. 서 론

HTML(Hyper Text Markup Language)5는 웹문서를 제작하기 위해 사용되는 프로그래밍 언어인 HTML의 최신규격을 말한다. 기존의 HTML에서는 브라우저 자체 기능만으로 동영상 재생 및 그래픽 (RIA : Rich Internet Application)등을 구현하는데 많은 제약 사항이 따랐으며, 이를 해결하기 위해 다양한 비표준 플러그인(Flash, SilverLight, ActiveX 등)을 사용해야만 했다.

결국 이러한 형태로 만들어진 웹어플리케이션들은 플러그인에 대한 의존도가 커지게 되었고, 플랫폼에 맞게 별도로 개발해야하기 때문에 많은 개발 비용 및 시간이 소요되었다.

HTML5는 이러한 문제점에 대한 개선의 필요성에 의해서 제안되었다. HTML5의 기본적인 원칙은 "플랫폼, 장치 등에 의존하지 않는 웹어플리케이션 구현"이며, 이를 위해서는 브라우저 자체적으로 다양한 기능을 구현할 수 있어야 한다. 그래서 기존 HTML 버전과 비교해 보았을 때, 명세의 많은 부분이 추가・확장되었으며, 브라우저 제조사들 역시 이러한 기능들을 구현할 수 있는 브라우저를 개발하기 시작하였다.

audio/video를 포함한 각종 태그 및 속성, CORS, Web Storage, Web Socket, Geolocation 등이 HTML5 신규 기능의 대표적인 예이며, Chrome, Safari, FireFox, Opera, IE 등 주요 웹브라우저들도 이러한 기능 구현을 위해 지속적인 업데이트를 수행중이다.

그러나 새롭게 추가된 기술들을 통해 새로운 보안 위협이 발생할 수 있으며, 이는 지금의 웹 개발자, 사용자, 웹사이트, 단말, 웹어플리케이션 등 웹과 관련된 모든 요소들이 신규 보안 위협에 취약할 수 있음을 의미한다.

본고에서는 HTML5 차세대 웹표준 환경에서 발생 가능한 보안 위협을 시나리오 기반으로 설명하고, 이를 예방할 수 있는 대책에 대하여 살펴본다.

Ⅱ. HTML5 신규 기능별 보안 위협

2.1. HTMl5의 역사

W3C에 의해 제안된 XHTML은 웹어플리케이션을 제작하는 실무자들이 필요로 하는 부분을 충분히 반영

* 한국인터넷진흥원 (sckang@kisa.or.kr)
** 한국인터넷진흥원 (pjs@kisa.or.kr)

하지 못했으며 방향성에 대한 우려가 점점 커지게 되었다. 이러한 이유로 브라우저 제조사인 Apple, Mozila, Opera는 WHATWG(Web Hypertext Application Technology Working Group)을 설립하였으며, HTML 표준과 관련된 현실적인 문제들을 해결하기 위해 “Web Application 1.0”을 제안하게 되었다. 결국 W3C는 2009년에 이르러 XHTML2에 대한 연구를 중단하고 Web Application 1.0을 HTML5라는 이름으로 바꾸어 WHATWG와 공동으로 표준화 작업을 진행하게 되었다. HTML5의 총괄 개발자인 Lan Hickson이 WHATWG 소속이기 때문에, 사실상 관련 표준에 대한 동향은 WHATWG에 속한 브라우저 제조사들의 정책에 크게 의존하고 있는 상황이다.

HTML5에 대한 표준화 작업은 WHATWG의 주도하에 현재(14년 7월)까지도 진행 중이다. 그래서 지금까지 국내외 컨퍼런스 등을 통해 발표된 HTML5 신규 보안 위협들에 대한 해석 및 접근 방법이 지금보다 더 다양화되거나, 의미가 없어질 수도 있다.

본고에서는 표준화 작업이 거의 완료된 신규 기능들과 표준에 추가될 가능성이 있는 기능들에 대한 보안취약점 분석 결과를 소개한다.

2.2. HTML5 신규 기능 및 보안 위협

〈표 1〉 HTML5 신규 기능 요약

주요 기능	설 명
신규 요소	audio, video, source, canvas, embed와 같은 신규 태그와 각 태그에 대한 세부 기능을 설정할 수 있는 속성 들이 추가되었다.
CORS(Cross Origin Resource Sharing)	CORS는 브라우저가 XHR 을 이용하여 Cross-Origin 요청을 가능하게 한다. 해당 기술은 서로 다른 도메인(사이트)간 자원을 공유할 수 있게 해주며, 특정 조건들이 만족될 경우 동일출처정책(SOP : Same Origin Policy)에 의한 제한을 완화하는데 사용될 수 있다.
Web Storage	Web Storage는 브라우저에서 구현되는 약 5MB 크기의 저장 공간이며 각 도메인에 따라 별도로 제공 된다. 웹어플리케이션은 정보 유지 및 관리를 위해 해당 공간에 다양한 정보들을 저장할 수 있다.
Web Socket	Web Socket 은 웹 서버와 User Agent간 지속적인 연결을 기반으로 하는 전이중 통신을 가능하게 해준다. 채팅, 주식 차트와 같은 실시간 통신이 필요한 어플리케이션에 활용할 수 있다.
Geolocation	Geolocation은 사용자(단말기)의 위치정보를 브라우저 단에서 측정할 수 있는 기능이다. 해당 기술은 모바일 장치에서 유용하게 사용된다.

2.2.1 HTML5 신규 요소

자바스크립트를 이용하여 자원에 접근하거나 제어할 수 있는 기술들만이 HTML5의 전부는 아니다. 태그 및 속성 부분에서도 많은 기술들이 제안되었다. HTML5 이전 기술로 제작된 웹어플리케이션들은 클라이언트 측 기능을 구현할 때 대부분 자바스크립트나 기타 플러그인에 의존했었다. 하지만 HTML5의 신규 태그 및 속성들이 이러한 부분을 자체적으로 처리하는 것이 가능해지면서, 웹어플리케이션 개발이 용이해졌으며, 사용자 입장에서도 부가적인 플러그인 설치가 필요 없어졌다.

예를 들어, HTML5 이전 버전 브라우저에서 비디오・오디오를 재생하기 위해서는, 이러한 영상을 처리해주는 플러그인을 설치해야 했지만, HTML5에서는 <video>, <audio> 태그들이 이러한 부분을 스스로 처리할 수 있다.

(1) HTML5 신규 요소 관련 보안 위협

HTML5의 신규 태그 및 속성들은 보안의 관점에서 보면 공격자들이 악용할 수 있는 공격 범위를 더 넓게 만들었다. 예를 들어 onfocus, autofocus 등의 속성들은 기존 XSS(Cross-Site Scripting) 필터에 포함되지 않으면서 자바스크립트를 실행하는데 사용될 수도 있다. [그림 1]은 신규 태그 및 속성을 이용한 XSS 공격 예이다.

```
<video><source onerror="alert(1)"></source></video>
<audio><source onerror="alert(1)"></source></audio>
<select autofocus onfocus="alert(1)">
<textarea autofocus onfocus="alert(1)">
```

〔그림 1〕 HTML5 신규 태그 및 속성을 이용한 자바스크립트 실행 코드(예)

2.2.2 CORS(Cross Origin Resource Sharing)

최근에는 웹서비스 업체들이 제공하는 각종 콘텐츠와 서비스를 가져와 새로운 서비스를 재창출하는 매시업(Mash-Up) 형태의 서비스가 활발하게 개발되고 있다. 이는 기존에 만들어진 자원(또는 서비스)을 재사용하면 개발 비용 및 시간이 절약되기 때문이다. 하지만 동일 출처 정책(SOP : Same Origin Policy)에 의한 제한은 이러한 유형의 매시업 서비스 개발을 어렵게 만들었다. 동일 출처 정책(SOP)이란, 웹브라우저들이 보안을 위해 자바스크립트로 다른 도메인의 웹페이지에 접근하는 것을 못하도록 막아놓은 제약조건이다. 즉 호스트나 응용계층의 프로토콜, 포트번호가 일치하지 않는 곳으로부터의 요청 발생을 제한하는 정책이다.

예를들어 www.foo.com의 자원을 사용하기 위해서는 아래 www.foo.com과 동일한 호스트, 프로토콜, 포트로부터(그림 2) 요청이 발생한 경우만 가능하다.

기존에는 SOP를 우회하기 위해 Server-Side Proxy, JSONP 등의 다양한 편법들이 사용되기도 했지만 웹서비스 구현이 복잡해지고 성능에 영향을 준다는 단점을 가지고 있었다.

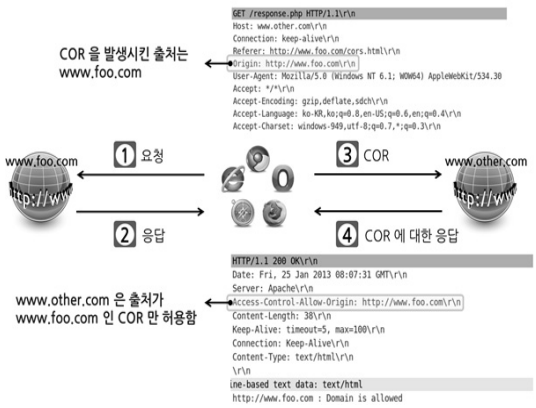
Compared URL	Outcome	Reason
http://www.foo.com/dir/page.html	Success	호스트, 프로토콜, 포트가 동일함
http://www.foo.com/dir2/other.html	Success	호스트, 프로토콜, 포트가 동일함
http://www.foo.com:81/dir/other.html	Failure	호스트, 프로토콜은 동일하나 포트가 다름
https://www.foo.com/dir/other.html	Failure	포트, 호스트는 같으나 프로토콜이 다름
http://en.foo.com/dir/other.thml	Failure	프로토콜, 포트는 같으나 호스트가 다름
http://foo.com/dir/other.html	Failure	프로토콜, 포트는 같으나 호스트가 다름
http://www.foo.com:80/dir/other.html	Don't use	브라우저에 따라 다름

〔그림 2〕 동일 출처 정책(SOP) 설명

CORS는 웹페이지상에서 자바스크립트를 이용하여 XHR(XMLHttpRequest)을 다른 도메인으로 발생 시킬 수 있도록 해주는 기능을 가지고 있다. 해당 기술은 XHR 기반 Cross-Origin 요청을 이용하여 자원을 공유해야 하는 브라우저와 서버 사이의 대화 방법을 정의한다.

CORS는 2004년 3월 미국의 통신회사인 Tellme Networks사의 엔지니어들이 처음 제안했으며, 차후 W3C와 주요 브라우저 제조사들이 참여하여 표준화 작업을 진행하고 있다.

2013년 1월 29일 W3C가 후보 표준안을 확정한 상태이다.



〔그림 3〕 CORS 동작 방식

(1) CORS 관련 보안 위협

CORS의 근본적인 보안문제는 XHR을 다른 도메인으로 발생시킬 때, 사용자에게 허가 요청을 받지 않는다는 것이다. 이러한 문제는 사용자의 세션을 오용한 접근제어와 관련된 보안 문제를 야기할 수 있다. 공격자가 사용자의 세션을 오용할 수 있게 된다는 것은 접근제어를 우회하여 허가되지 않은 리소스에 접근할 수 있다는 의미이다.

〈표 2〉 CORS 관련 보안 위협

관련 보안 위협	설 명
접근제어 우회	사용자가 속해있는 내부 네트워크에 존재하는 웹사이트가"Access-Control-Allow-Origin" 응답 헤더를 잘못 정의했을 경우 공격자는 외부에서 직접 접근할 수 없는 내부 네트워크의 웹사이트에 접근할 수 있다.
CORS와 Web Socket을 이용한 원격 쉘	XHR과 Web Socket 기술을 이용하면 사용자의 브라우저 세션을 탈취하고 행동을 제어할 수 있다. 대표적인 예로 세션 하이재킹 공격이 있다.
CORS와 Web Socket을 이용한 웹 기반 봇넷	CORS와 Web Socket 기술을 이용하면 사용자의 브라우저들을 제어하여 다양한 정보를 수집하거나 명령을 내릴 수 있다.
CORS와 Web Worker를 이용한 DDoS 공격	CORS와 Web Worker를 이용하면 보다 효과적으로 DDoS 공격을 수행할 수 있다.

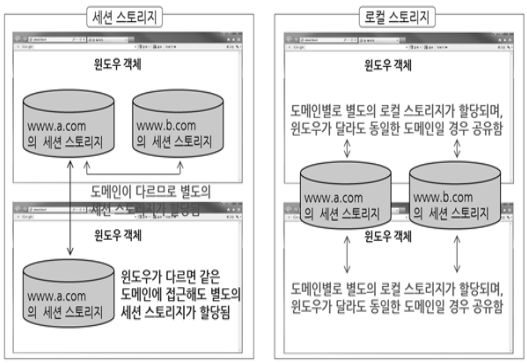
2.2.3 Web Storage

HTML5 이전 버전에서는 웹어플리케이션 관련 콘텐츠를 클라이언트 측에 저장할 때 주로 쿠키를 사용하였다. 하지만 쿠키의 경우 몇 가지 문제점을 가지고 있다. 첫 번째는 도메인(사이트) 별로 20개까지만 쿠키를 생성할 수 있으며 각 쿠키는 4KB 크기를 넘을 수 없다는 것이다. 두 번째는 HTTP 요청이 발생할 때마다 쿠키 값이 포함되어 서버로 전송된다는 것이다. 이는 불필요한 오버헤드 및 웹세션 탈취 관련 문제점을 야기한다. Web Storage는 이러한 문제점들을 보완하기 위해서 고안된 기술이다.

Web Storage는 웹사이트와 관련된 콘텐츠를 사용자의 컴퓨터에 저장하고 이후에 자바스크립트를 이용하여 접근하거나 통제할 수 있다. 컴퓨터에 저장할 수 있는 데이터의 용량은 브라우저의 종류에 따라 다르나 일반적으로 도메인당 5MB 정도의 크기를 사용할 수 있다.

〈표 3〉 Web Storage 설명

종 류	설 명
Session Storage	기본적으로 Local Storage와 유사하나 브라우저 혹은 탭을 종료할 경우 데이터가 삭제된다.
Local Storage	해당 영역에는 모든 텍스트 값을 저장할 수 있다. 각 아이템들은 Key Name - Value 쌍으로 구분되며, 존재하는 기존 아이템은 Key Name을 통해 접근할 수 있다. 사용자 혹은 웹어플리케이션에 의해서 생성/삭제될 수 있으며, 브라우저를 종료해도 데이터들은 삭제되지 않는다.



〔그림 4〕 Session Storage Vs Local Storage

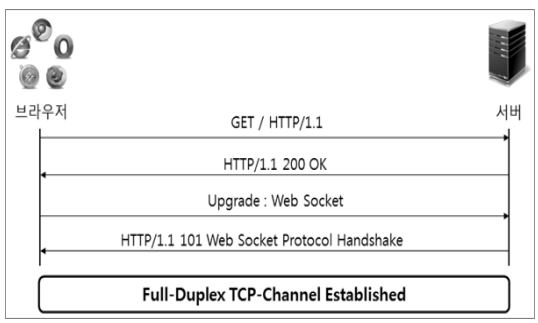
(1) Web Storage 관련 보안 위협

Web Storage와 관련된 가장 큰 보안 이슈는 해당 영역에 저장되어 있는 데이터에 대한 불법 적인 접근과 이를 사용자 측에서 인지할 수 없다는 것이다. Web Storage에 대한 모든 접근 및 제어는 자바스크립트를 통해 이루어지며, 특정 도메인이 XSS 등 스크립트 기반 취약점을 가지고 있을 경우 공격자는 사용자의 브라우저에 있는 모든 Web Storage 데이터들을 사용자 모르게 조작하거나 가져올 수 있다. 해당 취약점은 HTML5 Web Storage의 자체적인 취약점으로는 볼 수 없지만, Cookie에 비하여 저장 공간이 크며, 다양한 정보가 저장되기 때문에 공격자가 탈취할 수 있는 정보 역시 많다고 볼 수 있기에, 개발자는 해당 기술을 구현할 때 중요 정보를 암호화하여 저장하거나 취급하지 않아야 할 것이다.

2.2.4 Web Socket

현재 웹에서 주로 사용되는 HTTP는 브라우저가 서버에게 요청을 해야만 서버에서 응답을 해주는 구조로 설계되어 있다. 이러한 구조는 채팅이나 주식시장 차트처럼 실시간 정보 전달을 요구하는 어플리케이션을 개발하는데 제한적이다. 네트워크 세션형성/종료에 많은 오버헤드를 가져오기 때문이다. 결국 한 번의 네트워크 세션 형성을 통해 전이중 통신이 가능한 기술에 대한 개발자들의 요구사항이 커지게 되었고, 이러한 필요에 의해 HTML5의 Web Socket이 제안되었다.

Web Socket은 12년 12월 후보권고안(Candidate Recommendation)으로 확정된 상태이다.



〔그림 5〕 Web Socket 동작 원리

(1) Web Socket 기술 관련 보안 위협

사용자의 허가 없이 다른 도메인으로 연결 요청을 발생시킬 수 있으며, 요청이 발생했다는 사실을 사용자에게 알려주지 않는다. 공격자는 사용자의 브라우저에 임의의 자바 스크립트 코드를 실행하게 하여 사용자 모르게 브라우저를 통해 다른 도메인에 있는 시스템과 통신 채널을 형성한 후 데이터 송수신이 가능하다.

2.2.5 Geolocation

모바일 기기를 통한 인터넷 이용률 증가는 위치정보 기반의 서비스들이 급증하는데 큰 역할을 하였다. 대표적인 예가 SNS(페이스북 등)를 이용하여 자신의 위치를 공유하는 기능이다.

과거에는 이러한 기술을 구현하기 위해서는 부가적인 플러그인을 설치를 해야만 가능했다. 하지만 HTML5의 Geolocation API가 등장하면서 브라우저 자체 기능만으로 단말의 위치정보 계산이 가능해졌다.

〔그림 6〕 Geolocation API를 이용한 위치정보 확인

일반적으로 스마트 단말의 경우 GPS(Global Positioning System), 3G, 4G네트워크 정보 등을 기반으로 위치정보가 수집되며, GPS가 없는 단말기의 경우에는 IP 주소 정보를 통해 위치 정보를 계산한다.

Geolocation는 13년 10월 표준안 확정이 된 상태이다.

(1) Geolocation 기술 관련 보안 위협

대부분의 브라우저들은 Geolocation API를 사용할 때 사용자에게 동의를 구하도록 설계가 되어 있지만, 동일 도메인에서는 단 한 번의 동의를 통해 사용자의 위치정보를 계속 수집할 수 있는 경우가 있다.

이 경우 사용자가 공격자의 위치정보 수집 악성 링크를 클릭하고 무심결에 동의를 누를 경우 자신의 위치정보를 계속적으로 공격자에게 전달할 수 있다. 이는 기존 Native-App에서도 발생 가능한 공격 기법이지만, Geolocation을 적용한 HTML5 Web App의 경우 단순 사이트 방문만으로도 사용자의 위치정보가 노출될 수 있기 때문에, 기존 설치된 앱에서의 사용자 위치추적보다 위험성이 크다고 볼 수 있다. Geolocation 기술 활용에 대한 개발자 및 사용자의 세심한 주의가 필요하다.

Ⅲ. HTML5 기반 공격 시나리오 및 대응방안

3.1. 신규 태그/속성을 이용한 Tabnabbing 공격

3.1.1 공격기법 이해를 위한 배경 기술

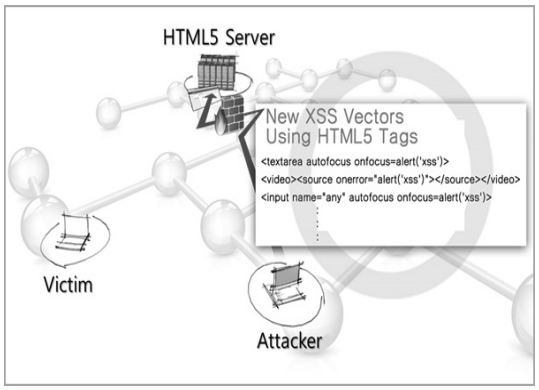
Tabnabbing은 자바스크립트를 이용한 자동화된 피싱 공격 기법이다. 기존의 피싱 기법은 사용자의 클릭을 유도하여 웹브라우저의 탭을 피싱 사이트로 이동시키는 형태이지만, 해당 공격은 사용자의 브라우저 탭에서 아무런 행위를 하지 않아도 악성 콘텐츠를 삽입하거나 브라우저 탭을 피싱 사이트로 이동시킬 수 있다. 물론 사용자가 해당 탭을 사용하고 있을 때 조작을 하게 되면 사용자가 의심할 수 있으므로, 희생자가 해당 탭을 사용하지 않고 다른 탭을 사용할 때 공격 행위가 이루어진다. Tabnabbing 공격은 XSS 취약점을 이용하여 구현된다. 하지만 많은 웹사이트들이 XSS 공격을 막기 위한 필터링을 설정해두었기 때문에, 이를 우회하기 위해 본 시나리오에서는 HTML5 신규 태그/속성을 이용한다.

3.1.2 공격 시나리오

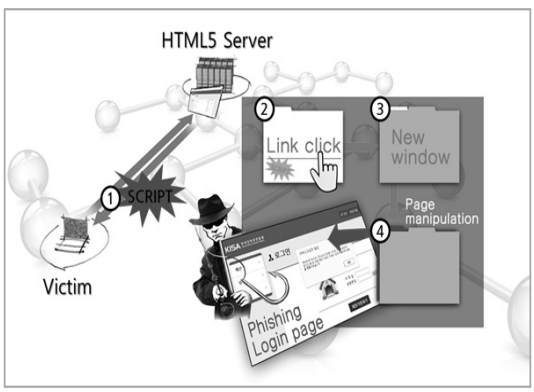
① 공격자는 공격 대상 웹서버에 XSS 공격을 시도하지만 기존의 잘 알려진 대부분의 태그/속성을 이용한 XSS 공격은 대부분 차단되어 있는 상태이다.

② 공격자는 HTML5에 새롭게 추가된 태그/속성을

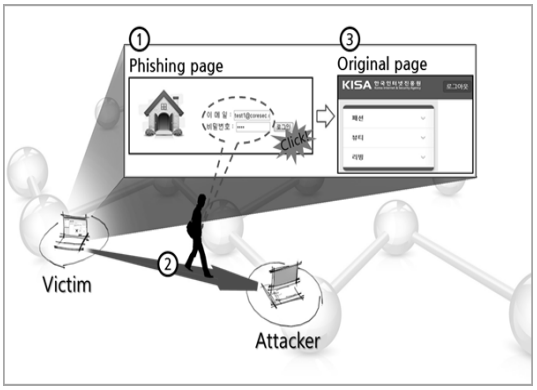
통해 XSS 필터를 우회하여, Tabnabbing 공격을 수행하는 악성코드를 게시판 등을 통해 삽입하고, 사용자는 해당 악성 게시글을 클릭한다.



③ 게시글의 내용 중 정상적인 페이지로 이동시키는 링크(예. 뮤직비디오 재생 페이지)를 클릭할 경우, 희생자 브라우저에서 새로운 탭이 생성된다. 이때, 희생자가 사용하고 있던 탭(게시글 페이지)에서는 tabnabbing을 수행하는 악성코드가 실행되며, 기존 탭을 피싱 사이트(기존 탭과 유사한 사이트의 로그인 페이지)로 강제 이동시킨다.



④ 다른 탭에서 뮤직비디오 시청을 완료한 희생자는 기존 탭의 로그인 페이지를 보고, 아무 의심 없이 로그인 정보를 입력함으로써 공격자 서버로 로그인 정보가 전송된다.



3.1.3 대응방안

『신규 속성/태그를 이용한 Tabnabbing 공격』 시나리오에서 Tabnabbing 공격은 주로 자바스크립트를 사용한다. 그래서 서버 측에서 XSS 취약점에 대한 대응을 한다면 공격을 완화할 수 있다. 또한 해당 공격은 HTML5에서 새롭게 추가된 속성 및 태그를 사용하기 때문에 XSS 필터도 이에 맞는 조건들을 추가해 줄 필요가 있다.

3.2. CORS를 이용한 사용자 개인정보 탈취

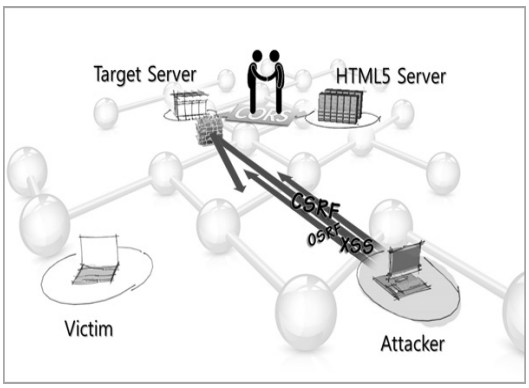
3.2.1 공격기법 이해를 위한 배경 기술

CORS를 이용하면 서로 다른 도메인간(웹사이트 서버) 자원 공유가 가능하며, 이는 서버 설정 중 'Access-Control-Allow-Origin:domain'을 통해 설정이 가능하다. 예를 들어 www.foo.com에서 www.other.com에 CORS로 자원을 요청할 경우 www.other.com 설정은 'Access-Control- Allow-Origin : www.foo.com 혹은 *(모든 도메인)'으로 설정이 되어있어야만 한다. CORS 보안 취약점 예방을 위해서는 신뢰할 수 있는 사이트(서버)에 대해서만 접근을 설정을 해야 한다(Access-Control-Allow-Origin : 신뢰할 수 있는 사이트).

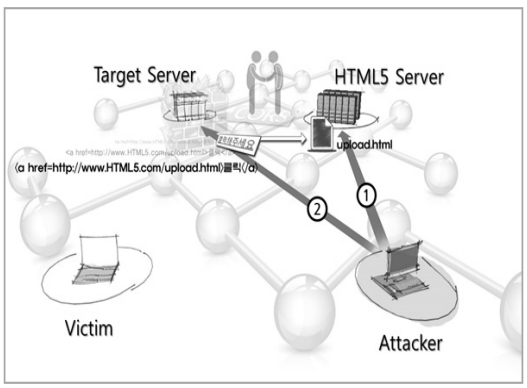
3.2.2 공격 시나리오

① 공격자(Attacker)는 공격 대상 서버(Target Server)의 CSRF 취약점을 이용하여 희생자의 개인 정보를 탈취하려고 하지만, 공격 대상 서버는 XSS 및

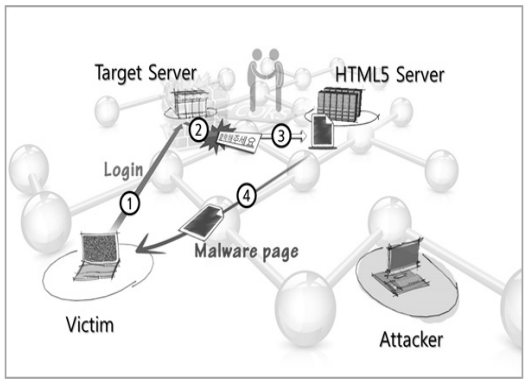
CSRF에 취약하지 않다. 또한 시도하는 모든 공격이 웹 방화벽에 의해서 차단되어 있는 상황이다. 하지만 공격 대상 서버는 특정 웹 서버(HTML5 Server)와 CORS를 위한 신뢰 관계가 형성되어 있음을 확인했다.



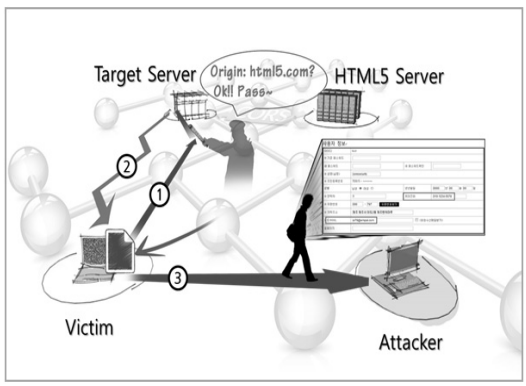
② 공격자는 특정 웹 서버(HTML5 Server)에 악성 스크립트가 담긴 파일을 업로드한다. 그리고 공격 대상 서버에는 특정 웹 서버(HTML5 Server)에 업로드 되어 있는 파일을 링크하는 게시글을 업로드한다.



③ 희생자는 공격 대상 서버(Target Server)에 접근하여 로그인을 수행한 후 공격자가 ②에서 업로드했던 게시물의 링크를 클릭한다.(물론 해당 링크는 클릭을 유도하는 내용을 담고 있다.) 사용자의 웹브라우저 탭은 특정 웹 서버(HTML5 Server)에 업로드된 악성 스크립트를 가져온 후 실행하게 된다.



④ 악성 스크립트에 노출된 희생자의 웹브라우저는 자신의 개인정보를 공격자에게 전송하게 된다. 실제 CSRF를 통한 A사용자의 개인정보 탈취 트래픽을 발생시킨 것은 B서버이며, A서버와 (Access-Control-Allow-Origin : www.B.com)으로 신뢰관계를 맺고 있기에 이 공격이 가능하다.



3.2.3 대응방안

『CORS를 이용한 사용자 개인정보 탈취』 시나리오는 공격 대상 사이트의 파일 업로드 취약점을 이용하는 것으로 시작된다. 때문에 웹 서버의 파일 업로드 취약점에 대한 대응 조치를 한다면 공격을 완화할 수 있다. 그리고 서버는 Cross-Origin 요청을 모두 허용할 수 있는 “Access-Control-Allow-Origin : *” 과 같은 설정을 피하는 것이 좋다.

3.3. Web Storage 정보 탈취

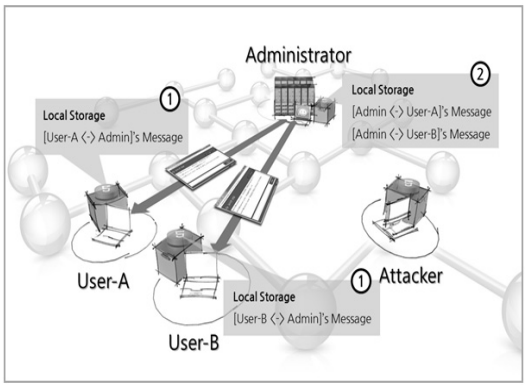
3.3.1 공격기법 이해를 위한 배경 기술

Web Storage에는 웹어플리케이션에서 이용되는 다양한 정보가 평문 혹은 암호문으로 저장될 수 있으며, 저장 공간에 대한 모든 접근 및 제어는 자바스크립트를 통해 이루어진다. 만약 특정 웹사이트가 XSS 등 스크립트 기반 공격에 취약하다면, 공격자는 사용자의 브라우저에 있는 모든 Web Storage 데이터를 사용자 모르게 조작하거나 가져올 수 있다.

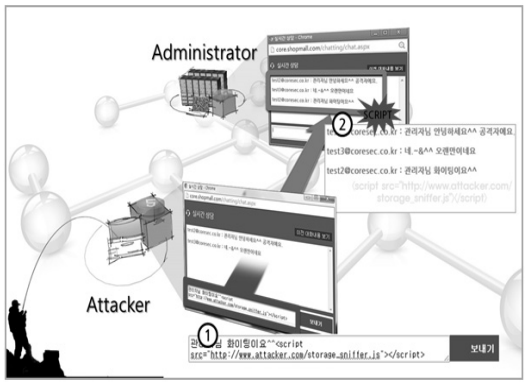
해당 취약점은 Web Storage 자체적인 문제이기 보다는 XSS 취약점과 개발자의 잘못된 정보 취급(평문, 주요 정보 저장 등)으로 인한 취약점이지만, 기존 Cookie에 비하여 저장되는 정보의 양이나 질이 높기 때문에 공격자의 입장에서는 공격 정보 수집을 위한 유용한 대상이 될 것이다.

3.3.2 공격 시나리오

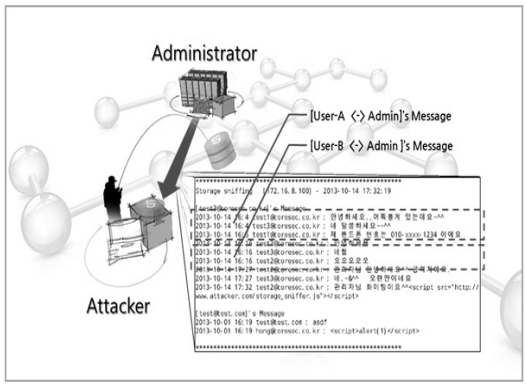
① 공격자는 (공격대상)HTML5 웹사이트의 관리자 ⇆ (사용자)채팅 내용들이 각 사용자 및 관리자의 Local Storage에 저장되며 스크립트 기반 공격(XSS, CSRF 등)이 가능하다는 것을 확인 후 이 내용들을 탈취하고자 한다.



② 공격자는 관리자에게 대화를 걸어, 관리자가 일반 사용자들과 대화한 내용들을 모두 탈취하는 악성 스크립트를 입력한다.



③ 악성 스크립트를 해석한 관리자의 웹브라우저는 자신의 Local Storage에 있는 모든 대화 내용을 공격자 서버로 전송한다.



3.3.3 대응방안

『Local Storage 에 저장된 사용자 개인정보 탈취』 시나리오는 공격 대상 사이트의 XSS 취약점을 이용하는 것으로 시작된다. 때문에 XSS 에 대한 대응조치(웹 방화벽 혹은 보안코딩 등)가 되어 있다면 공격을 완화할 수 있다. 그리고 웹어플리케이션 개발자는 평문 상태로 데이터를 저장하는 Local Storage에 사용자의 민감정보를 저장하지 않도록 하거나 암호화를 적용시켜 저장하도록 구현함으로써 정보 관리에 유의하도록 한다.

3.4 Web Socket을 이용한 내부 네트워크 정보 수집

3.4.1 공격기법 이해를 위한 배경 기술

Web Socket API는 일반적으로 Well-Known 포트들에 대한 접근이 기본적으로 차단되도록 구성되어있지만, 각 시스템들의 Up/Down 여부 확인이 가능하며 이는 공격자 입장에서 유용한 정보가 될 수 있다.

```
[Constructor(in DOMString url, in optional DOMString protocol)]
interface WebSocket {
  readonly attribute DOMString URL;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSED = 2;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;

  // networking
          attribute Function onopen;
          attribute Function onmessage;
          attribute Function onclose;
  boolean send(in DOMString data);
  void close();
};
WebSocket implements EventTarget;
```

〔그림 7〕 Web Socket 구조

Web Socket은 연결하려는 객체의 상태정보를 의미하는 세가지 값(Connecting, Open, Closed)을 가지고 있다. 최초 Web Socket이 생성되면 해당 객체의 ReadyState값은 0(Connecting)으로 설정되어 있으며, 스캐닝 대상의 상태에 따라 1(Open) 혹은 2(Closed)로 값이 변화한다. 즉, Web Socket의 ReadyState 값인 0(Connecting)은 원격 호스트의 포트 상태에 따라 지속시간이 달라진다고 말할 수 있다.

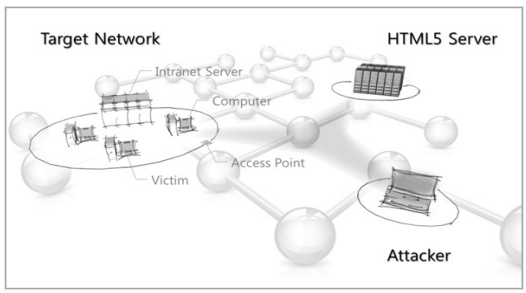
〈표 4〉 Web Socket을 이용한 시스템 포트 오픈 여부 확인

원격 시스템 포트 상태 및 필터링 유무	0값의 지속시간
원격 시스템의 포트가 열려있는 경우(Open)	<100ms (0.1초 미만)
원격 시스템의 포트가 닫혀있는 경우(Closed)	~1000ms (1초)
패킷이 필터링 되었을 경우	>30000ms (30초 초과)

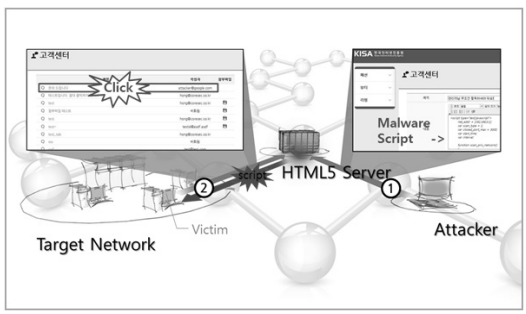
Web Socket 객체 ReadyState속성의 0값 지속시간이 수 초 이상 지나도 1이나 2로 변하지 않는다면 대상 시스템은 동작중이지 않을 확률이 높다고 판단할 수 있다. 해당 공격 시나리오 역시 이와 같은 논리를 바탕으로 동작한다.

3.4.2 공격 시나리오

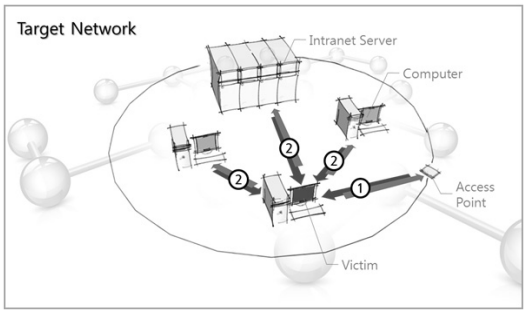
① 희생자는 회사 네트워크(192.168.5.X)를 통해 업무를 수행중이며, 공격자는 외부 인터넷망에서 희생자의 내부 시스템을 해킹하기 위해 내부 네트워크 정보를 수집하려 한다.



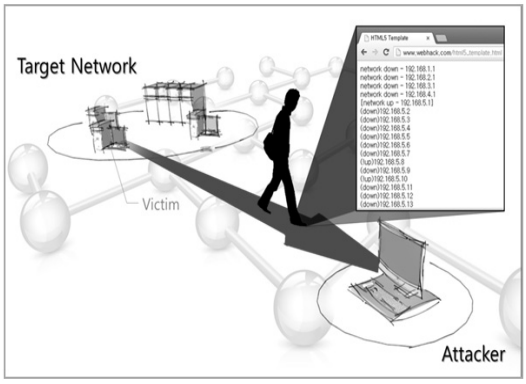
② 공격자는 특정 게시판에 사용자의 내부 네트워크를 스캐닝하는 악성 게시글을 업로드한다.



③ 희생자가 웹서핑 중 해당 게시글을 클릭하게 되면, 사용자의 웹브라우저는 Web Socket 객체를 생성하여, 자신이 속한 네트워크의 정보를 수집하기 시작한다.



④ 192.168.5.X 대역을 모두 스캐닝한 사용자의 웹 브라우저는 결과를 공격자 서버에 전송한다.



3.4.3 대응방안

『Web Socket API 를 이용한 사설 네트워크 정보수집』 시나리오는 공격 대상 사이트의 XSS 취약점을 이용하는 것으로 시작된다. 때문에 XSS 에 대한 대응조치(웹방화벽 혹은 보안코딩 등)가 되어 있다면 공격을 완화할 수 있다. 만약 희생자를 공격자의 사이트로 유도한 후 해당 시나리오를 구현하는 경우라면 대응할 수 없다.

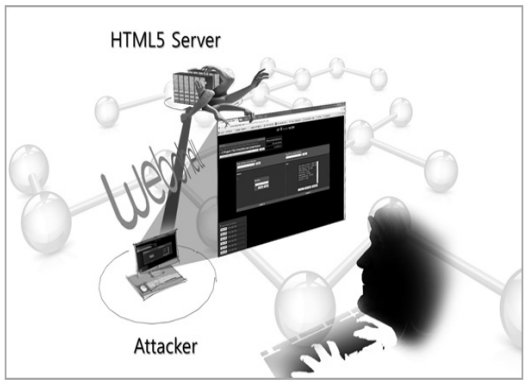
3.5. Geolocation을 이용한 사용자 위치 추적

3.5.1 공격기법 이해를 위한 배경 기술

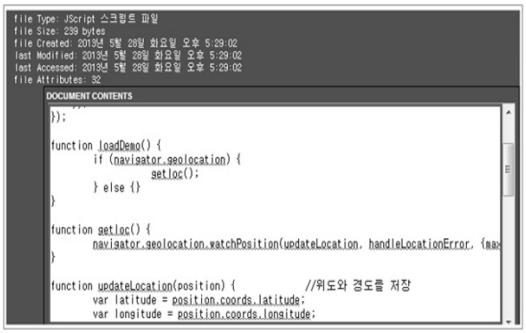
브라우저가 공격대상 서버에서 Geolocation API를 처음 사용하기 위해서는 사용자의 자발적인 동의(위치정보 수집 허용)를 얻어야 하는데, 한번 사용허가를 받은 코드가 추후 Geolocation API를 사용할 경우 사용자에게 다시 묻지 않는다. 이는 악성 자바스크립트가 사용자의 위치정보 수집을 위해 Geolocation API를 지속적으로 사용한다 해도 사용자는 알 수 없다는 의미이다.

3.5.2 공격 시나리오

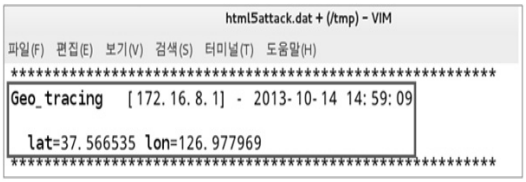
① 공격자는 희생자의 위치정보 수집을 위해 공격대상 서버에 위치 정보를 수집하는 악성 자바스크립트를 삽입할 계획이다. 공격자는 대상 서버가 가진 파일 업로드 취약점을 이용하여 WebShell을 업로드한다. 해당 WebShell은 공격 대상 시스템의 웹어플리케이션 소스코드를 조작할 수 있다.



② 공격자는 WebShell을 이용하여 웹 서버의 메인 페이지 프레임에 새로운 프레임을 덧씌우고 새로운 프레임 내부에 악성 스크립트를 삽입한다. 이는 해당 페이지를 방문한 사용자가 브라우저 탭을 닫지 않는 이상 스크립트 실행을 지속적으로 유지할 수 있기 때문이다.



③ 웹 서버를 방문한 사용자의 모바일 기기에 있는 웹브라우저들은 악성 스크립트를 지속적으로 실행시키면서 희생자의 위치 정보를 공격자에게 주기적으로 전송한다.



3.5.3 대응방안

『Geolocation API 를 이용한 사용자 위치정보 추적』 시나리오는 공격 대상 사이트의 XSS 취약점을 이용하는 것으로 시작된다. 때문에 XSS에 대한 대응조치(웹방화벽 혹은 보안코딩 등)가 되어 있다면 공격을 완화할 수 있다. 하지만 사용자를 공격자의 사이트로 유도한 후 해당 시나리오를 구현하는 경우라면 대응할 수 없다. 사용자는 가급적 자신의 기기에 대한 위치정보 접근을 제한하거나 필요한 경우 신뢰하는 서비스에 대해서만 허용을 해야 한다.

Ⅳ. 결 론

현재까지 HTML5에 대한 표준화 작업이 완료되지 않았음에도 불구하고 국내를 포함한 전 세계의 많은 웹사이트들이 HTML5 기술을 이용하여 재구축되고 있으며, 각종 장치에 포함되는 어플리케이션들 또한 HTML5를 기반으로 개발되기 시작했다. HTML5의 기본적인 디자인 원칙은 “플랫폼, 장치 등에 의존하지 않는 웹어플리케이션 구현”이다. 그래서 기존 버전과 대조해 보았을 때 명세의 많은 부분이 확장 및 추가되었다. video/audio를 포함한 각종 태그 및 속성 그리고 CORS, Web Storage, Web Worker, Web Socket, Geolocation 등이 대표적인 예이다. 이렇게 추가된 기능들로 인해 브라우저 자체적으로 동영상 재생, 그래픽 구동 등 다양한 멀티미디어 기능 구현이 가능하게 되었다.

하지만 이러한 기술들을 통해 다양한 이점을 얻을 수 있지만 동시에 악의적인 용도로 남용할 수 있는 공격의 범위도 앞서 설명한 것처럼 증가하게 되었다. 결국 HTML5를 기반으로 하는 모든 웹사이트 및 어플리케이션들은 새로운 보안 위협에 직면하게 될 것이며 이와 관련된 공격을 예측하고 대응방법을 개발하는 연구가 필요하다.

참 고 문 헌

[1] 미래창조과학부, 인터넷 글로벌 경쟁력 강화를 위한 차세대 웹 표준(HTML5) 확산 추진계획, 2012.7
[2] Daniel Nilsson, HTML5 Web application security with OWASP top ten 2013, 2013
[3] Enisia, A Security Analysis Of Next Generation Web Standards, 2011
[4] Juan Galiana Lara, HTML5 Security Cheat Sheet, 2013
[5] Martin Johns, HTML5 Security GPN 2012, 2012
[6] Michael Schmidt, HTML5 Web Security, 2011
[7] Nick Doty, Privacy Issues of the W3C Geolocation API, 2010
[8] Robert McArdle, A Look at HTML5 Attack Scenarios, 2011
[9] Shreeraj Shah, DOMJacking Attack Exploit and Defense, 2012
[10] Shreeraj Shah, Stealth Attacks and Silent Exploits, 2012

〈저 자 소 개〉

강 석 철 (Kang Sok-Chul)
2013년 7월 : 성균관대학교 정보보호학과 석사
2010년 7월~현재 : 한국인터넷진흥원 주임연구원
관심분야 : 정보보호

박 정 섭 (Park Jeong-Seop)
2001년 2월 : 한국외대 MIS 석사
2000년 10월~현재 : 한국인터넷진흥원 팀장
관심분야 : 보안산업, 정보보호