AJAX 기술과 보안

유성수, 노봉남
전남대학교 정보보호협동과정
badaro2001@lsrc.jnu.ac.kr

Security to AJAX

Song-su Ryu, Bong-nam No
Information Security, chonnam national university

요 약

AJAX(Asynchronous Javascript and XML) 이란 기존에 존재하는 기술들의 조합으로 이루어진 새로운 웹 플랫폼 기반 어플리케이션 개발 기술이다. 데스크탑 어플리케이션과 같은 기능을 웹에서 실행 가능하게 하는 가능성을 가지고 있는 기술로, 앞으로의 웹 어플리케이션의 방향성을 제시하고 있다. 본 논문에서는 Ajax에 사용되어지는 기술들을 소개하고, Ajax의 특징 및 사용 예를 살펴보고, Ajax에서의 보안문제에 대해서 소개한다.

1. 서 론

시대가 변하면서 웹 환경은 발전하고 있으며, 보다 사용자와의 상호작용을 위한 여러가지 기술들이 개발되어 왔다. 기존의 단방향의 웹환경에서, 사용자와 상호작용을 통한 웹 어플리케이션 환경으로 변화하고 있으며, 그에 따른 발전으로 기존 데스크탑 어플리케이션 환경에서 가능하다고 생각되었던 상호작용을 통한 인터렉티브한 기능들이, 이제는 웹에서도 그 가능성을 보이고 있다. 이처럼 웹에서 데스크탑 애플리케이션과 같은 환경을 제공 할 수 있다고 하는 생각이 확산되기까지는 구글의 Gmail, Google Maps의 등장이 큰 계기가 되었다. 이 두 가지 서비스는 웹 기반의 어플리케이션이지만 RIA(Rich Interface Application)로 데스크탑의 기능에 달하는 양방향성과 UI를 제공한다.

Ajax은 이미 존재하고 있는 여러기술들(XHTML, CSS, DOM, XML, XSLT, XMLHttpRequest, JavaScript 등)이 합쳐져서 이루어진 기술이다. 비동기 통신을 이용한 데이터 전송량의 감소, 그리고 JavaScript를 이용한 인터렉티브한 UI의 구현으로, 고속화 되어지는 네트워크와 고성능의 PC 환경을 바탕으로, 서버쪽에서 처리하던 데이터처리와 가공을 클라이언트로 분배함으로써 여러 리소스와 자원들을 좀 더 효율적으로 사용할 수 있게 한다.

Ajax 어플리케이션은 사용자와 서버사이에 Ajax 엔진이라는 한단계를 더 두어서 일반 웹 어플리케이션서의 화면 reload나 전체 페이지를 매번 전송해야 하는 반복적인 작업들을 제거하고, 필요한 데이터만을 전송 하여 반응성을 높이고, 서버에 요청을 보낸 후 응답을 기다리지 않고 사용자가 다른 작업을 계속 하는 것이 가능하게 해준다.

본 논문에서는 위에 나열되어 있는 기술들을 설명하고, Ajax의 특징과 서비스, 사용 예를 살펴보고, Ajax에서의 보안문제에 대해서 소개한다.

2. AJAX기술에 대한 소개

2.1 AJAX

Ajax (Asynchronous JavaScript And XML)로 JavaScript와 XML을 이용한 비동기 통신방식을 뜻한다. 2005년 Jesse James Garrett는 Ajax의 4가지 특징에 대해서 다음과 같이 말하고 있다.[8]

- XHTML(또는 HTML)과 CSS를 이용한 UI
- DOM을 이용한 동적인 화면구성과 상호작용
- XML과 XSLT를 이용한 데이터교환
- XMLHttpRequest를 이용한 비동기 데이터교환
- 그리고 이것을 합쳐 주는 JavaScript

일반 웹 어플리케이션 모델은 클라이언트가 요청을 보낸 후, 서버는 그 응답을 처리하고, 요청에 대한 응답에 대한 전체페이지 데이터를 전송, 클라이언트는 응답이 도착하기까지 다른 작업을 하지 않고 기다려야 했던 반면 Ajax 웹 어플리케이션모델은 XMLHttpRequest를 이용하여 백그라운드에서 서버와의 통신을 수행하며, 요청에 대한 응답 XML 데이터를 전체페이지가 아닌 필요한 문서 속성의 데이터에 대해서만 교체한다.

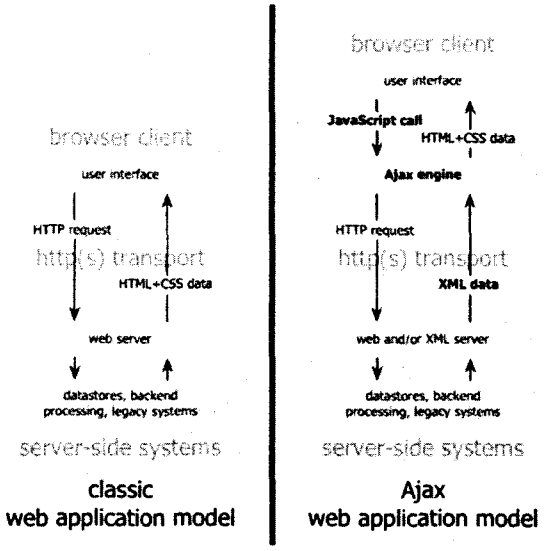

그림 1: 기존 웹어플리케이션과 Ajax 웹 어플리케이션 동작 방식[8]

2.2 JavaScript

JavaScript는 Netscape사에서 클라이언트 쪽에서 독립적으로 실행되는 프로그램을 작성하기 위한 스크립트 언어로 개발 되었다. JavaScript는 객체 지향 스크립트 언어로서 프로그램 코드는 HTML 문서 안에 포함되어서 컨트롤러, 이벤트를 이용해 동적인 웹페이지를 만들 수 있게 해주는 역할을 한다.[3]

JavaScript에 대한 표준화는 ECMA내에 있는 TC39 위원회에서 이루어 지고 있고, ECMAScript 라고 불리우며 현재 3번재 표준명세인 ECMA-262가 발표 되어있다.[7]

2.3 XML과 DOM

XML(Extensible Markup Language)은 데이터를 기술하는데 사용할 수 있는 마크업 언어이다. 연결된 시스템간의 데이터 공유를 위한 목적으로 1996년 W3C에서 제안한 것으로서, 웹상에서 구조화된 문서를 전송 가능하도록 설계된 표준언어이며, XML을 기반으로한 언어로는 RDF, RSS, MathML, XHTML, SVG등이 있다.

DOM(Document Object Model)은 객체 지향 모델로써 구조화된 문서를 표현하는 형식이다. DOM은 HTML 문서의 구성요소를 조작하기 위해 웹 브라우저에서 처음 지원됐다. 구조화된 문서는 DOM을 사용하여 트리구조로 변환 될 수 있으며, 이런 문서의 내용, 구조, 스타일에 접근하고 변경하는 수단으로 사용되어 진다.

2.4 XMLHttpRequest

XMLHttpRequest(XHR)은 HTTP를 사용하여 클라이언트와 서버간의 XML데이터 전송하기 위한 전송 기술로 비동기방식의 전송이 Ajax 기술의 핵심적인 부분을 담당한다. XMLHTTP는 Microsoft가 제안한 기술로 Internet Explorer 5.0부터 ActiveX Object로 구현되었지만, Mozilla Project는 이와 호환되는 Native Object를 XMLHttpRequest라는 이름으로 Mozilla 1.0에서 발표하고, 이외의 다른 브라우져들이 XMLRequest를 반영함에 따라 사실상 XMLHttpRequest가 표준으로 자리 잡게 되었다.

3. Ajax기술의 특징과 서비스의 예

3.1 XML과 XSLT를 이용한 데이터교환

XML은 구조화된 자료를 전송 가능하도록 설계된 표준언어이고, XSLT는 선언적인 방법을 이용해서 필요한 데이터만을 뽑아내고 재구성하여 새로운 데이터를 만들 수 있도록 해주는 역할을 한다.[4]

3.2 XMLHttpRequest를 이용한 비동기 통신

XMLHttpRequest를 이용한 비동기 통신을 통하여 사용자가 응답을 기다리지 않고도 다른 작업을 가능하게 해준다. 클라이언트에서 보낸 요청에 대한 서버의 응답이 도착하면 콜백 함수를 통해서 응답이 왔을 때 지정한 작업이 실행되기 때문에, 사용자는 응답을 기다릴 필요 없이 다른 작업을 계속 진행 할 수 있어서, 작업의 효율성이 증가하게 된다.[2]

3.3 JavaScript를 이용한 UI구현

JavaScript를 이용해서 UI와 다른 구성요소들을 합쳐주는 역할을 한다. 특정 플랫폼에 종속적이지 않고 웹 브라우저만 있으면 실행 할 수 있으며, CSS와 XHTML과 같은 기법과 함께 사용되어 데스크탑 어플리케이션과 같은 역동적인 UI의 구현과 XMLHttpRequest의 비동기 통신과 함께 비동기적인 작업을 가능하게 한다.

3.4 Ajax 기반 서비스의 예

Google Maps는 Ajax기술을 이용해서 만들어진 웹기반 어플리케이션으로, ActiveX나 Desktop 기반 어플리케이션에서 구현되어 지던 기능을 Ajax 기술로 대처할 수 있는 가능성을 보여줬다는 점에서 중요한 위치를 차지한다. ActiveX를 이용한 지도서비스 들이 있었지만 이 같

은 경우는 Microsoft사의 Internet Explorer에서만 실행이 가능하고 설치과정을 거쳐야 하는 반면, Ajax기술을 이용해서 구현된 Google Maps는 Javascript와 XML을 이용해서 구현되었기 때문에 운영체제에 종속적이지 않으면서 웹브라우저만 갖춰져 있는 환경이라면 어디에서나 실행이 가능하다는 장점을 가지고 있다는 점에서 그 의미가 크다고 할 수 있다.

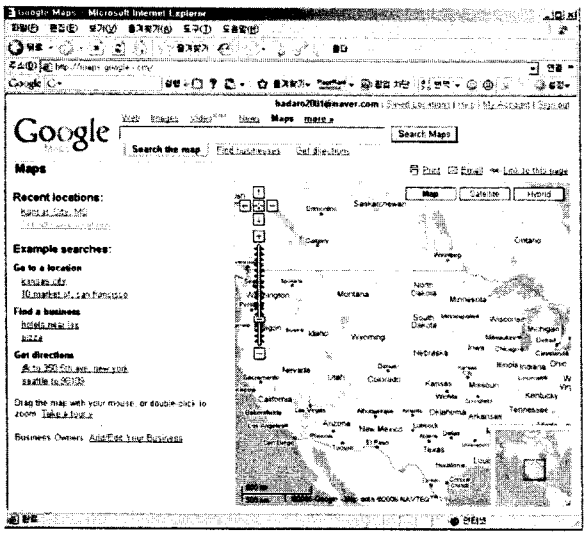

그림 2: Ajax기술로 구현된 Google Maps

4. Ajax 보안

4.1 크로스사이트 스크립팅 취약점을 이용한 공격

일반적인 웹 어플리케이션에서는 서버의 응답을 기다리고, 화면에 나타나기까지 기다려야 했던 것과는 달리, Ajax 어플리케이션은 비동기통신을 이용해서 서버에 요청을 보낸 후 응답을 기다리면서 화면상에서 다른 작업을 하는 것이 가능하다. 하지만 이런 편리성이 악의적인 용도로 사용된다면 새로운 위협으로 다가 올 수 있다. 예를 들면, 크로스사이트 스크립팅 취약점을 이용해서 공격자는 사용자에게 악의적인 공격이 담긴 Ajax 어플리케이션을 실행하게 할 수 있으며, Ajax의 특성상 사용자는 공격당하고 있다는 사실을 알지 못한 채 공격코드가 백그라운드에서 실행되면서, 사용자 정보를 가로 챈 다던가 사용자의 친구목록을 이용해 공격적인 코드를 전송하는 등의 공격들을 실행 할 수 있다. 물론 웹브라우저에서 스크립트 실행에 대한 설정을 제공하고 있지만 그 기능이 간단한 수준에 머물고 있고, 크로스사이트 스크립팅 취약점을 이용해서 신뢰된 사이트로부터 전송되었다고 웹브라우저는 생각하게 되기 때문에, 공격적인 스크립트는 사용자의 민감한 정보에 접근할 수 있고[5], Ajax기술은 이런 악의 적인 스크립트공격에도 향상된 기능을 제공할 수 있다.

4.2 Client-side 보안필요성의 증가

일반적인 웹 어플리케이션 환경에서는 Server side에서 여러 작업을 처리하지만, Ajax에서는 서버에서 필요한 데이터만을 받고, 화면에 표시될 페이지는 클라이언트의 Ajax엔진이 처리 하는 등 클라이언트에서 담당하는 비중이 커지게 되었다. 전달해야 하는 데이터의 통신량이 줄어들고, 서버의 처리 부담이 줄어든다는 것은 분명 커다란 장점 이지만, 클라이언트의 역할이 커진다는 건 서버보안 뿐만이 아니라 클라이언트 보안에도 신경을 써야 한다는 걸 의미한다. 또한 클라이언트의 역할증가로 인한 클라이언트 코드의 비대화, 운제 발생 시 서버와 클라이언트의 두 곳의 코드를 수정해야 하는 등의 관리 부담의 증가도 가져올 것이다. 이와 같이 Ajax환경에서는 관리의 보안문제발생시의 관리의 대상이 서버에서 서버와 클라이언트로 확장되므로 그에 따른 관리가 힘들어진다는 걸 의미한다.

4.3 JavaScript 소스코드의 누출

JavaScript는 Ajax에서 클라이언트 UI와 Ajax engine에 사용되어 진다. 하지만 JavaScript 코드는 HTML안에 포함되어 소스보기를 통해서 누구나 볼 수 있으며, 이것은 클라이언트의 동작방식이나 클라이언트와 서버가 주고 받는 동작방식 등을 누구나 알 수 있다는 것이다. 물론 설계할 때 보안에 관련된 기능이나 중요한 기능들은 서버측에서 구현되도록 해야 하겠지만, 클라이언트의 코드를 숨길 수 없다는 건, 공격자가 클라이언트의 코드를 수정하거나, 비슷한 동작방식의 새로운 어플리케이션을 만드는 등의 위험성을 가지고 있다고 할 수 있다. 예를 들어, 입력값 검증을 위한 클라이언트의 코드를 공격자가 분석할 수 있으므로, 다른 방법으로 인자를 조작하지 않더라도 코드를 분석해서 조작된 인자를 보내기 위한 어플리케이션을 제작, 이를 우회하는 것이 가능하다. 물론 JavaScript 코드를 알아보기 어렵게 해주는 encoder들을 이용할 수도 있지만, 키 값을 이용한 encoding등을 이용한 방법으로 키 값에 대한 관리 문제가 있으며, 컴파일을 통한 실행코드만을 제공하는 것과 같은 경우에 비해서, 소스코드를 숨기는 작업이 힘들다. 이런 이유로 JavaScript로 만들어진 클라이언트의 어플리케이션의 민

감한 정보나 동작 방식이 포함 되어있을 경우 그 정보들이 보호되어지기 어렵다.

4.4 Ajax 어플리케이션에서의 인증부분

일반적인 웹 어플리케이션 환경에서는 요청에 대한 응답을 할 때 사용자를 확인 한 후 응답을 보내게 된다. 하지만 Ajax에서는 비동기 통신을 이용해서 필요한 데이터를 받고, 그 데이터를 이용해서 페이지 중간의 적절한 문서의 속성을 업데이트 시켜서, 페이지 전체를 리로드 하지 않고도 원하는 속성의 데이터만을 바꾸는 방식을 사용한다. Desktop 어플리케이션의 경우 정해진 어플리케이션을 통해서 통신이 이루어지지만, Ajax 어플리케이션의 특징상 어플리케이션 보다는 특정한 자료를 활용한 여러 가지 어플리케이션이 등장 할 수 있기 때문에, Ajax 어플리케이션에서 개인정보와 같은 중요한 데이터들을 다루는 경우를 생각 한다면, XML기반의 데이터에 따른 각각의 자료에 대한 사용자 인증과정을 거친다거나 각각의 자료에 대한 권한설정을 한다거나 하는 필요가 생길 수도 있으며, 인증상태를 확인하는 부분에 대한 보안문제 또는 인증상태를 어느 시점까지 유지해야 하는가에 대한 문제들이 발생 할 수도 있을 것이다. 기존의 페이지 단위의 자료에서 각 컴포넌트들에 쓰이는 자료로 세분화가 됨에 따른 각 자료의 보다 세분화된 권한의 세부설정이나 컴포넌트 단위의 권한 설정이 필요하게 될 수도 있을 것이다.

4.5 JavaScript Security Policy

JavaScript에서는 한 개의 특정서버의 스크립트가 다른 서버에서 보내온 문서의 속성에 접근하여 개인정보나, 구조, 사용자세션 등을 가져오는 것을 막는데 Same Origin Policy를 사용한다. Same Origin Policy는 특정 URL의 substring을 Origin으로 정의한다. A라는 Origin으로부터 온 문서를 가져오고, B라는 Origin으로부터 스크립트와 문서를 가져 왔을 때, B에서 가져온 스크립트는 A에서 가져온 문서에 대한 HTML Object의 특정 속성을 설정하거나 정보를 얻을 수 없다. 이는 JavaScript의 보안을 목적으로 하며, JAVA에서 제공하는 sandbox 보안 모델과 비슷한 방식을 취하고 있다.

4.6 웹 플랫폼

Ajax 웹 어플리케이션은, 일반 웹 어플리케이션이 가지고 있는 보안취약점들을 가지고 있다.
OWASP에서 말하는 웹 어플리케이션의 10대 취약점은 다음과 같다.[6]

- 입력값 검증 부재
- 취약한 접근 통제
- 취약한 인증 및 세션 관리
- XSS 취약점
- 버퍼 오버플로우
- 삽입 취약점
- 부적절한 에러 처리
- 취약한 정보 저장 방식
- 서비스 방해 공격
- 부적절한 환경 설정

Ajax 웹 어플리케이션 역시 웹 플랫폼이기 때문에 이런 취약점에 대해서 자유로울 수 없다. 거기에 Ajax의 특징인 XML을 이용한 데이터 교환, XMLHttpRequest를 이용한 비동기 통신 그리고 JavaScript를 이용한 클라이언트 어플리케이션 등의 특징에서 오는 여러 가지 보안문제들도 더해질 것이다.

5. 결론

본 논문 에서는 Ajax에 사용되는 기술과 특징 그리고 보안에 대해서 설명하였다. Ajax이 주목받는 이유는, 새로운 어플리케이션이나 서비스들을 만들 수 있다는 것 보다는, 기존의 특정 플랫폼에 종속적이던 여러 서비스, 어플리케이션들을 어디서나 실행할 수 있는 웹 플랫폼으로 끌어올릴 수 있는 가능성을 제시했다는 데에 있다고 볼 수 있다. 하지만 이런 가능성들은 일반적인 웹 어플리케이션의 보안문제에, Ajax의 특징으로 인한 보안문제들이 추가로 발생 할 수 있다는 것을 의미한다. 기술의 발전으로 주변의 컴퓨팅 환경이 발전할수록 Ajax을 이용한 웹 어플리케이션은 증가하게 될 것이고, 데스크탑 어플리케이션과 일반 웹 어플리케이션의 자리를 대처하게 될 것이다. 아직은 보안이나 민감한 정보를 다루는 Ajax 어플리케이션들이 선보이지는 않고 있지만 기술의 발전과 사용자의 요구로 인해서 그런 어플리케이션들이 등장하게 될것이고, Ajax이 여러 기술들이 합쳐져 새로운 방향을 제시하고, 사용 환경을 웹 환경으로 끌어올린 만큼, 그에 따른 보안 관련 연구도 진행되어야 될 것이다.

6. 참고문헌

[1] Professional Ajax, 2006, Nicholas C. Zakas
[2] Head Rush Ajax, 2006, Brett McLaughlin
[3] JavaScript:The Definitive Guide, 2001, David Flanagan
[4] Learning XML, 2003, Erik T. Ray

[5] 웹 해킹 패턴과 대응, 2005, 황순일, 김광진
[6] OWASP, www.owasp.org
[7] ECMA, http://www.ecma-international.org
[8] Jesse James Garrett, http://adaptivepath.com
[9] Google Maps, http://maps.goolgle.com