한국항행학회 논문지 제15권 제4호 2011년 8월

BHO 이용한 웹 컨텐츠 변조 탐지 방법

Web contents deformation detection method by BHO

모정훈*, 정만현*, 조재익*, 문종섭*

Jeong-Hoon Mo*, Man-Hyun Chung*, Jae-Ik Cho* and Jong-Sub Moon*

요 약

최근 인터넷 서비스 기술이 발달함에 따라 웹 서비스는 사용자의 컴퓨팅 환경에 많은 변화를 주었다. 시사, 경제, 게임/오락은 물론, 개인 금융까지도 웹 페이지를 통해 처리 된다. 이 때, 웹 페이지는 텍스트 형태의 코드를 전송받아 DOM 정보로 가공되어 웹 브라우저에 의해 사용자에게 보여 진다. 하지만, 이 정보들은 다양한 경로를 통해 접근이 가능하고 악의적인 목적으로 변조되어질 수 있다. 또한, 보안 매커니즘을 우회하여 사용자의 로그인 정보나 인증서를 획득할 수도 있다. 따라서, 본 논문에서는 이러한 웹페이지 변조 행위를 탐지하기 위해 웹 브라우저 중 대표적인 MicroSoft 사의 MS Internet Explorer의 Add-On 프로그램인 BHO를 이용하여 웹 컨텐츠에 대한 무결성을 검증하는 탐지 방법을 제안한다.

Abstract

Recently, with improvement of internet service technology, web service has been affecting the environment for computing user. Not only current events, economics, game, entertainment, but also personal financial system is processed by web pages through internet. When data transmission is implemented on the internet, webpage acquire text form code and transform them to DOM information, and then shows processed display to user by web browser. However, those information are not only easily accessed by diversified route, but also easily deformed by intentional purpose. Furthermore, it is also possible to acquire logon information of users and certification information by detouring security mechanism. Therefore, this dissertation propose the method to verify integrity of web contents by using BHO which is one of the Add-On program based on MS Internet Explorer platform which is one of major web browser program designed by MicroSoft to detect any action of webpage deformation.

Key words : Information Secret, Web Forge, DOM, BHO

I. 서 론

최근 인터넷 서비스 기술이 발달함에 따라 웹 서비스는 사용자의 컴퓨팅 환경에 많은 변화를 주었다. 시사, 경제, 오락은 물론, 개인 금융까지도 웹 페이지를 통해 처리 된다. 이 때, 웹페이지는 텍스트 형태의 코드를 전송받아 DOM 정보로 가공되어 웹 브라우저에 의해 사용자에게 보여 진다. 하지만, 이 정보들은 다양한 경로를 통해 접근이 가능하고 악의적인 목적으로 변조되어질 수 있다. 또한, 보안 매커니즘을 우

* 고려대학교 정보보호대학원 (Korea University, Graduate School Of Information Security
· 제1저자 (First Author) : 모정훈
· 투고일자 : 2011년 8월 10일
· 심사(수정)일자 : 2011년 8월 10일 (수정일자 : 2011년 8월 24일)
· 게재일자 : 2011년 8월 30일

회하여 사용자의 로그인 정보나 인증서를 획득할 수 있다. 현재 사용되고 있는 웹 브라우저들은 W3C에서 권고한 DOM의 표준에 맞게 각자의 API들을 구현하여 웹페이지를 파싱하고 사용자에게 관련 웹 페이지를 제공 한다[1][3]. 하지만, W3C에서는 DOM 객체들에 대해 인터페이스만을 정의하고 구현에 대한 상세한 정보를 제공하지 않기 때문에 웹 브라우저들은 DOM 표준을 따르되 각자의 제품의 기능에 맞게 구현해 사용한다. 그리고 이러한 API들은 외부 프로그램을 통해 접근될 수도 있으며, DOM 관련 API를 사용하여 문서의 파싱에 관여하고 웹 페이지를 수정할 수도 있다. 예를 들어, Microsoft의 인터넷 익스플로러의 기능을 외부 플러그인을 이용해 확장 할 수 있도록 설계된 기술인 BHO는 웹 브라우저가 사용하는 DOM 관련 API를 모두 사용할 수 있다[4][8]. 이러한 Add-on 프로그램은 사용자에 의해 직접 설치되거나 사용자의 의도와 관계없이 설치되는 경우도 있다. 웹 브라우저의 제작 시에도 DOM 관련 API의 안전한 접근에 대한 고려가 필요하다. 따라서 본 논문에서는 2장 관련연구에서 Document Object Model의 구성과 현재 많이 사용되고 있는 Microsoft 사의 웹 브라우저인 MS Internet Explorer의 Add-On 프로그램인 BHO 설명하고 제안방법에서 사용하는 암호화 방법을 설명 한다 그리고 3장에서는 BHO를 이용해서 가능한 공격 방법을 설명하고 4장에서는 탐지 방안을 제시하고, 5장에서 결론으로 본 논문을 서술한다.

Ⅱ. 관련연구

2-1 문서 객체 모델 (DOM; Document Object Model)

DOM은 객체 지향 모델로써 구조화된 문서를 표현하는 형식이다. 이는 플랫폼과 언어에 독립적으로 구조화된 문서를 표현하는 W3C의 표준으로 문서의 파싱을 위한 표준적인 기능의 정의를 제공하고, HTML문서와 XML문서의 내용과 구조들을 조작할 수 있는 기능을 제공한다[1]. 하지만 DOM은 관련 객체들에 대해 인터페이스는 정의하지만, 구현에 대해서는 어떤 내용도 제공하지 않는다.

1) DOM을 통한 문서의 표현

웹 브라우저는 서버로부터 전송된 HTML/XML의 텍스트 형태의 문서를 트리 형태의 자료형으로 파싱하여 메모리에 저장한다. 다음 표 1은 간단한 HTML 코드로 작성된 문서의 예이다.

표 1. HTML 문서의 예
Table 1. Example of HTML Document

```
<html>
  xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="en">
    <head>
        <title>Apple</title>
    </head>
    <body>
        <h1>Apple</h1>
        <p title="memt">Apple List.</p>
        <ul id="list">
            <li>Apple</li>
            <li>Peach</li>
            <li>Tomato</li>
        </ul>
    </body>
<html>
```

위에서 언급한 바와 같이, HTML 문서는 다음 그림 1과 같이 트리 형태의 구조로 파싱되어 메모리에 표현되며, 원하는 요소(Element)에 대하여 트리의 노드로써 검색, 수정, 삭제가 용이하며 사용자가 보게 되는 화면을 동적으로 구성할 수 있다. 이와 같이 요소에 접근을 편리하게 해주는 오브젝트 모델로서 HTML과 XML에 대한 API를 제공한다[1][2].

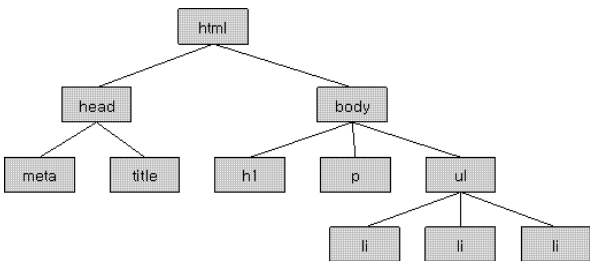

그림 1. 웹페이지 문서 구조
Fig. 1. Webpage Document Structure

2) DOM 인터페이스를 이용한 접근

많은 웹 브라우저들이 W3C에서 권고한 DOM의 표준에 맞게 각자의 API들을 구현하여 웹 페이지를 파싱한다. 하지만, 이러한 API들이 외부 프로그램에서 사용되어 웹 브라우저의 파싱에 관여한다면 웹 페이지 보안의 취약점으로 악용될 수 있다. 다음은 DOM 인터페이스 중 트리의 노드를 탐색 또는 수정하는 등 악의적으로 사용될 수 있는 기능들의 예를 간단히 소개한다.

a) 노드(Element)의 탐색

다음 표 2와 표 3과 같이 HTML 태그의 이름이나 속성을 이용하여 노드를 탐색할 수 있다.

표 2. HTML 태그 이름을 이용해 탐색
Table 2. Search using HTML tag name

```
document.GetElementsByTagName("태그이름");
```

표 3. HTML 태그의 속성을 이용해 탐색
Table 3. Searching using HTML tag attribute

```
document.GetElementById("ID 이름");
```

b) 노드(Element)의 수정

다음 표 4와 표 5와 같이 탐색된 노드의 내용이나 속성을 수정할 수 있다.

표 4. 노드의 내용을 수정
Table 4. Modify content of node

```
Element = document.GetElementById("ID 이름");
Element.innerHTML = "수정할 내용";
```

표 5. 노드의 속성을 수정
Table 5. Modify attribute of node

```
Element = document.GetElementById("ID 이름");
Element.SetAttribute("속성", "수정할 내용");
```

c) 노드(Element)의 추가

다음 표 6과 같이 노드를 새롭게 추가할 수 있다.

표 6. 노드를 추가
Table 6. Append node

```
E = document.CreateElement("추가할노드");
document.Body.AppendChild( E );
```

3) Browser Helper Object (BHO)

BHO란 웹 브라우저가 자체적으로 제공하지 못하는 기능을 지원하는 플러그인과 같은 개념이다[5][8]. BHO는 금융 관련 사이트 접속 시 보안 프로그램을 실행하는 등의 유용한 목적으로 사용되기도 하지만, 애드웨어나 스파이웨어처럼 상업적으로나 악의적인 용도로 사용되기도 한다[6]. 다음 그림 2은 MS 인터넷 익스플로러에 설치된 다양한 BHO 툴바 프로그램의 예이다.

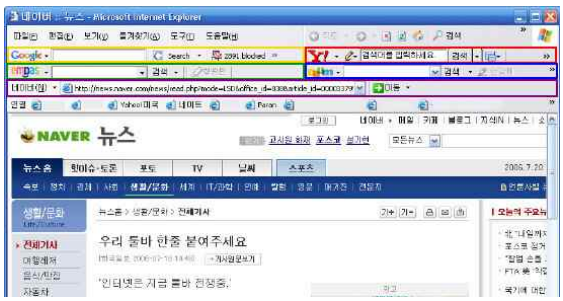

그림 2. 다양한 BHO가 설치된 MS 인터넷 익스플로러
Fig. 2. MS Internet Explorer that is installed various BHO

a) BHO를 이용한 DOM 접근

웹 브라우저는 서버로부터 전송된 텍스트 형태의 HTML/XML 코드를 파싱하여 메모리상에 트리구조

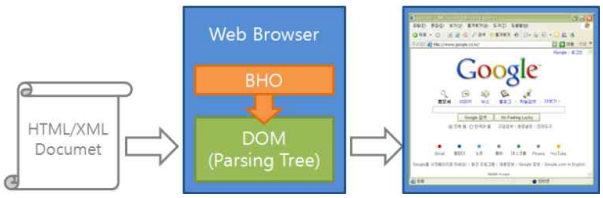

그림 3. HTML/XML 파싱 구조
Fig. 3. HTML/XML parsing structure

의 DOM으로 표현하고, 이러한 내용은 가공되어 사용자에게 UI로 출력 된다[2]. 이 때, BHO는 다음

그림 3과 같이 웹 브라우저에서 제공되는 DOM 관련 API들을 이용하여 브라우저의 HTML/XML 파싱에 관여하거나 이미 파싱된 내용을 수정하여 사용자에게 원본과 다른 내용을 보여줄 수 있다.

4) 암호화 방법

a) Diffie-Hellman 키 교환

Diffie-Hellman 키 교환 방식에서는 양쪽 통신주체가 KDC(Key Distribution Center)없이 대칭 세션 키를 생성한다. 대칭 키를 만들기 전에 양쪽은 두 개의 수 p와 q를 선택해야 한다. 여기서 p는 매우 큰 소수로서 300자리가 넘는 십진수(1024비트)이다. 두 번째 수인 g는 군 <Zp* , ×>의 원소로서 위수가 p-1인 생성자이다. 이 두 가지 정보는 비밀로 간직할 필요가 없다. 이 두 값을 인터넷을 통해서 전송한다. 다시 말해서 공개되어도 무방하다. 절차는 표 7과 같다[9].

표 7. Diffie-Hellman 키 교환 절차
Table 7. Diffie-Hellman Key exchange process

(1) A는 임의의 큰 수 x를 0≤x≤p-1 안에서 선택하고 R1 = qx mod p 를 계산한다.
(2) B는 다른 임의의 큰 수 y를 0≤y≤p-1 안에서 택하고 R2 = qy mod p 를 계산한다.
(3) A는 R1을 B에게 보낸다. 여기서 A는 x값을 보내는 것은 아니다. 오직 R1만 보낸다.
(4) B는 R2를 A에게 보낸다. 여기서도 B는 y값을 보내는 것이 아니고 오직 R2만 보낸다.
(5) A는 K=(R2)x mod p 를 계산한다.
(6) B는 K=(R1)y mod p 를 계산한다.
(7) 위의 계산으로 얻은 대칭키 K는 qxy mod p이다.

b) HMAC(Hash Message Authentication code) - SHA - 1

SHA-1(Secure Hash Algorithm) 은 1993년 미국 표준기술 연구소(NIST)에 의해 FIPS PUB 180 으로 출판되었으며, SHA -1은 최대 2^{64} 비트의 메시지로부터 160 비트의 해쉬값을 만들어 내며, MD4 알고리즘에 기반을 두고 있다. HMAC은 송신자와 수신자사이에 공유된 비밀 키를 이용하여 MAC을 산출하려는 메시지와 조합하여 MAC을 생성하는 알고리즘이다[11][12].

$$\backslash HMAC_K(m) = h((K \oplus opad) \parallel h((K \oplus ipad) \parallel m))) \quad (1)$$

HMAC-SHA-1은 MAC을 생성하기 위해 SHA-1 해쉬 함수를 사용한 것으로, 데이터의 무결성과 인증을 제공하는 알고리즘이다.

Ⅲ. 공격시나리오

인터넷 서비스를 제공하는 프로그램들은 기본적으로 사람이 읽을 수 있는 텍스트 기반으로 통신한다. 또한 프로그램에 사용되는 통신 프로토콜의 구조가 공개되어 있어 클라이언트와 서버 사이에 오가는 패킷을 감시함으로써 통신내용을 알 수 있다. 이러한 방법으로 민감한 정보들을 유출하여 악용할 수 있다. 현재의 개인 보안 매커니즘 또한 데이터를 암호화 하여 전송하는 등의 네트워크의 보안이나 클라이언트의 후킹에 의한 데이터 유출 방지에 초점이 맞춰져있다. 최근 인터넷 포털 사이트는 BHO로 인한 후킹을 방지하기 위해 FLASH나 ActiveX 등을 이용하여 후킹을 방지하는 보안 기능을 사용한 사용자 인증 폼을 제공한다[13][15]. 하지만 웹 페이지 내의 사용자 인증 폼을 삽입하는 구문 자체를 교체해 버리면 이러한

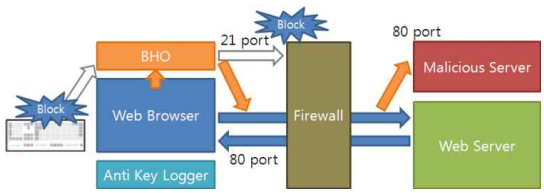

그림 4. BHO 보안 매커니즘을 우회 방법
Fig. 4. BHO Security mechanism bypass method

보안 기능이 쓸모없게 된다. 이처럼 BHO로 악의적인 행위들을 정상 서비스인 것처럼 가장하면 현재의 보안 매커니즘들을 우회할 수 있다.

그림 4와 같이 HTML 코드를 수정하여 사용자가 입력한 정보를 가로채는 방법은 웹 페이지 내에서 직접 정보를 얻음으로써 메시지 후킹 방지에 대한 보안 매커니즘을 우회할 수 있으며, 이렇게 가로챈 정보들을 웹 페이지를 통해 정보를 다른 곳으로 전송하는

방법은 특정 포트의 통신만을 허용하는 방화벽을 통과할 수 있다. 가능한 공격을 컨텐츠 변경 및 개인정보 유출 두 가지로 분류하여 소개한다.

3-1 컨텐츠 변경

웹 브라우저는 서버로부터 전송된 텍스트 형태의 코드를 파싱하고 이를 가공하여 사용자에게 보여준다. 이 때 악의적인 코드들은 파싱된 내용을 사용자가 인식하기 이전에 수정하여 사용자에게는 원본과 다른 내용을 보여 지도록 할 수 있다.

이러한 컨텐츠 변경은 다음과 같은 공격들로 예를 들 수 있다. 이미지나 쇼크웨이브 플래시 파일과 같은 컨텐츠의 HTML 삽입 코드를 변경함으로써 사용자에게 보이는 컨텐츠를 전혀 다른 내용으로 교체하거나 추가 또는 삭제하는 방법, 또는 사용자에게 주기적으로 특정 스팸 메시지 또는 광고를 보여주거나 인터넷 검색엔진의 검색 결과에 대한 하이퍼링크들을 변경할 수도 있다. 만약 이와 같은 방법으로 사용자를 피싱 사이트로 유도할 경우, 공인된 인터넷 검색엔진의 검색 결과로써 크게 의심하지 않고 피싱 사이트에 접속할 가능성이 크다. 이러한 웹 페이지의 변조는 정상적인 웹 서비스를 가장하여 다양한 공격으로 활용될 수 있다.

3-2 인증을 위한 개인 정보 유출

웹 서비스를 이용하면서 사용자를 인증하는 방법은 일반적으로 사용자 ID와 패스워드를 입력 받고 이를 검증하는 방법이다. 사용자가 입력하는 ID와 패스워드는 HTML 코드의 FORM태그와 INPUT 태그를 이용한 웹 폼을 통해 전달된다. 하지만 이렇게 입력된 정보들은 안전하지 않다. 악의적인 코드에 의해 웹 페이지의 입력 폼 구문을 수정하여 사용자가 입력한 정보를 가로챌 수 있다. 또한, 이러한 취약성을 보완하기 위해 FLASH나 ActiveX 형태의 입력 폼을 호출하여 사용하는 웹 페이지라도 이를 호출하는 구문 자체를 변경하여 보안 기능 자체를 무력화할 수 있다. 외부의 서버로부터 작성된 입력 폼을 해당 부분에 삽입하거나 구문 자체를 수정하여 사용자를 속일 수 있다. 변경된 입력 폼은 사용자가 입력한 인증정보를 다른 서버로 보냄으로써 인증정보를 가로챌 수 있다.

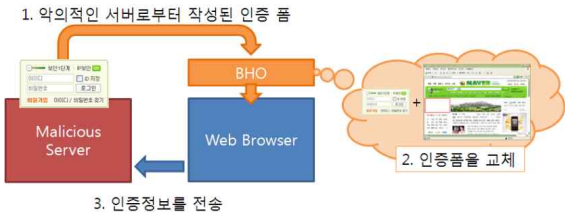

그림 5. 외부에서 작성된 입력 폼으로 교체하여 입력된 정보를 유출
Fig. 5. Leak information by replacing input form made from outside

그리고 안전한 사용자 인증을 필요로 하는 인터넷 사이트는 사용자의 공인 인증서 정보를 요구하기도 한다. 제 3의 공인된 기관에서 사용자에 대한 공인 인증서를 발급하여 서비스 제공자와 사용자를 서로 인증해주는 방식이다. 이 때, 웹 브라우저와 연동되는Add-On 프로그램이 인증서를 관리하고 인증정보를 안전하게 전송한다. 이러한 Add-On 프로그램은 HTML 코드 내의 JavaScript 에 의해 호출되는데 다음 그림 6과 같이 프로그램을 호출하는 JavaScript 구문을 변경하면 위조된 프로그램을 실행하게 할 수 있다. 위조된 프로그램은 정상 프로그램과 사용법과 모양이 동일하게 제작된다면 사용자가 직관적으로 구별할 수 없으며 실행 시 사용자의 인증서와 입력 받은 패스워드를 외부로 전송할 수 있다[15].

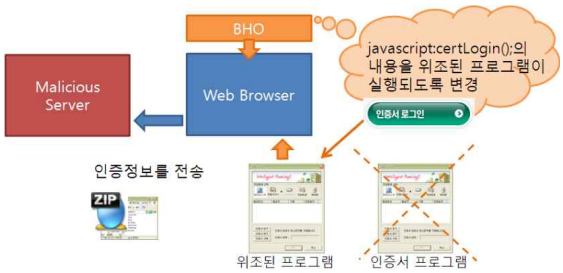

그림 6. 위조된 프로그램 실행
Fig. 6. Execute counterfeit program

Ⅳ. 제안 방법

4-1 웹 페이지 변조 탐지 방법

웹 페이지에는 다양한 웹 컨텐츠가 텍스트 형태의 코드로 구성되어 있으며 이들 중 개인 인증 정보를 입력받거나 인증서를 호출하는 코드와 같이 반드시 보호되어야 할 정보들도 포함되어 있다. 하지만, 위의 3장에서 살펴본 바와 같이 웹 페이지는 악성코드가 포함된 BHO나 JavaScript 코드 또는 프록시 서버를 이용한 설정 등의 다양한 방법으로 변조가 가능하다. 본 논문에서는 일반적으로 많이 사용되고 있는 마이크로소프트사의 인터넷 익스플로러의 Add-On 프로그램인 BHO를 이용해 웹 페이지의 변조를 탐지하기 위해 다음과 같은 방법을 제안 한다.

본 논문에서는 무결성과 가용성을 보장하기 위해 반드시 보호되어야 할 영역을 선정하여 각각의 컨텐츠 마다 {보호 대상 영역: 검증값} 형태의 데이터를 구성하고 해당 컨텐츠가 웹 페이지에 표시되는 1점에 이를 서버와 클라이언트가 안전한 방법으로 공유한다. 클라이언트인 웹 브라우저는 보호되어야 할 영역을 지속적으로 감시하며 해당 컨텐츠의 무결성을 검증한다. 이 때, 통신에 이용되는 데이터 형식으로는 JSON, XML을 사용하였다[17][18].

표 8. 컨텐츠 관리 데이터 구조
Table 8. Content manage data structure

JSON Ex) { key1:value1, key2:value2, … } XML Ex)<xml><key1>value1</key1><key2>value2</key2> … </xml>

본 논문의 제안 방법은 HMAC을 위한 키교환, 보호 컨텐츠의 HMAC 생성, 컨텐츠 변조 탐지 3단계로 구성한다. 1단계는 사용자의 웹 페이지 요청이 발생할 때 시작되는 단계로 BHO와 Web Server간의 안전한 통신을 위해 Key 교환을 하고, 2단계는 웹 브라우저에서 주요 객체들의 변조 여부를 판별하기 위한 HMAC을 생성하고 데이터 송수신을 하는 단계이며, 3단계에서 수신한 검증 데이터를 바탕으로 웹 페이지의 변조 여부를 판단한다.

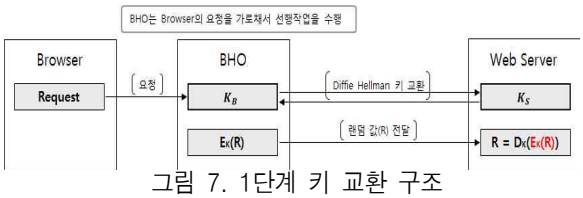

그림 7. 1단계 키 교환 구조
Fig. 7. first step: key exchange structure

1 단계 : 웹 브라우저가 Web Server에 웹 페이지에 대한 Request를 보내면, BHO는 웹 브라우저의 Request 이벤트를 가로채고, Web Server와 세션을 생성하고 키를 교환한다. 이때 교환된 키는 웹 페이지의 변조 여부를 판단하기 위한 검증 값 생성에 사용된다. 키 교환에서는 Diffe-Hellman 키 교환 방식을 이용하여 BHO와 Web Server 간에 비밀키를 교환하고 데이터 통신의 안전성을 확보한다. 키 교환 후 BHO는 웹 페이지의 보호 대상 영역에 대한 HMAC 생성에 Key로 사용될 Random Number를 생성하여 Web Server 에 전송한다.

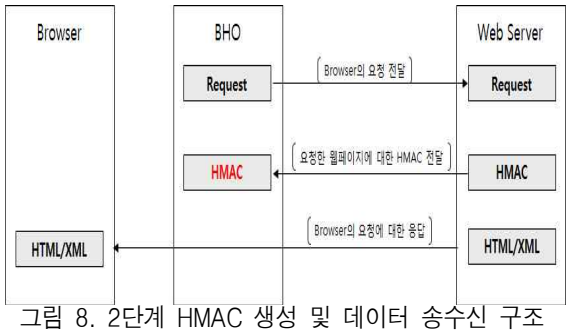

그림 8. 2단계 HMAC 생성 및 데이터 송수신 구조
Fig. 8. Second step: HMAC generation and data transmission structure

2단계 : BHO는 1단계에서 생성한 Random Number를 Web Server에 전송하는 동시에, 웹 브라우저가 요청한 웹 페이지의 보호 대상 영역의 정보를 Web Server에 요청한다. Web Server는 해당 웹 페이지의 미리 정의된 보호 대상 영역에 대해 HMAC을 생성하여 BHO로 전송한다. HMAC 생성에 사용되는 Key는 BHO가 생성한 Random Number이며, Json 또는 XML 데이터 포맷을 사용하여 통신한다.

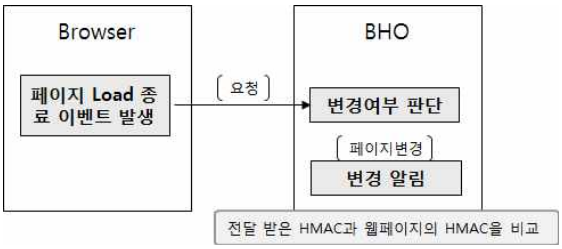

그림 9. 3단계 웹 페이지 변조 판단 구조
Fig. 9. Third step: web page modulation judgement structure

3단계 : 웹 브라우저와 BHO는 각각 Web Server로부터 실제 웹 페이지에 대한 문서와 이를 검증하기 위한 HMAC를 전달 받았다. 웹 브라우저는 수신된 웹 페이지를 파싱하여 DOM을 생성하고 Load 완료 이벤트를 발생한다. BHO는 웹 브라우저의 완료 이벤트를 감지하여 웹 브라우저가 로드한 DOM을 통해 보호 대상 영역을 탐색하여 1단계에서 생성한 Random Number를 이용해 HMAC을 구한다. 그리고 Web Server로부터 수신된 HMAC값과 비교하여 웹 페이지 변조 여부를 판단 한다.

본 논문에서 제안 하는 방법은 DOM의 동적인 구조와 웹 브라우저가 발생하는 이벤트를 이용한다. DOM의 동적 구조는 웹 브라우저가 웹 서버에서 수신 받은 웹 문서를 DOM 구조로 작성하고 이를 메모리에 적재한다. 이 과정이 완료되면 웹 브라우저로부터 이벤트가 발생하며 BHO는 이 이벤트를 감지 후 DOM에 접근한다. 만약, BHO를 통해 웹 페이지의 검증을 이미 마친 이후 악의적인 다른 BHO#2 또는 기타 다른 방법을 이용해 DOM을 변조 하여도, 웹 브라우저는 다시 DOM에 대한 이벤트를 발생시켜 BHO의 DOM 검증 기능을 작동시킨다. 따라서, HMAC의 검증 시점에 대한 시간차를 두고 DOM을 접근하는 경우라도 DOM의 내용이 변경될 때 마다 완료 이벤트가 발생되고 BHO#1가 다시 이를 탐지하기 때문에 본 논문의 제안된 탐지 방법을 우회 할 수 없다.

4-2 탐지 방법의 효율성 검증

본 논문에서 제안하는 탐지 방법은 사용자의 웹

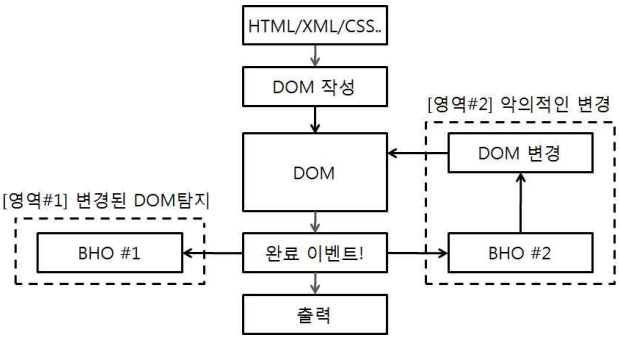

그림 10. DOM 동적 구조
Fig. 10. DOM dynamic structure

브라우저에 BHO를 설치 후 사용자가 요청한 웹 페이지의 보호 대상 영역의 변조 여부를 탐지하는 것으로 기존의 웹 페이지 처리 순서에 웹 페이지 변조 여부를 판단하기 위한 HASH 값 요청과 그리고 키 교환, 복호화, 탐지 등의 기능이 추가 되므로 기본 브라우저의 웹 컨텐츠 처리 속도를 측정 비교할 필요성이 있다. 웹 컨텐츠 처리 속도를 비교하기 위하여 DynaTrace Software에서 제작한 AJAX Edition을 사용하였다. AJAX Edition은 Web 2.0 Application을 최적하기 위한 분석도구로 Add-on 방식으로 웹 브라우저에 설치되어 웹 브라우저에서 웹 컨텐츠의 기능 및 속도 그리고 함수 기능을 분석 할 수 있다[19].

본 논문에서는 두 대의 동일한 성능의 PC에 익스플로러 8버전을 설치한 후 효율성 검증을 위해 한 대의 컴퓨터에 제안 방법을 적용한 BHO 프로그램을 설치하였다. 그리고 각각의 PC에서 동일한 웹 페이지를 50회 접속하여 속도에 대한 평균을 구해 보았다. BHO 프로그램이 설치된 PC에서는 키 교환 로직, 페이지 검증 로직을 사용할 때의 두 단계로 나누어 설치되지 않은 환경과 비교하였다. 다음 표9는 제안 방법을 사용한 웹 브라우저와 일반 웹 브라우저를 비교 분석할 결과 키 교환에 약 0.25 sec, 페이지 변조 탐지에 약 0.06sec의 차이가 발생하며, 검사 영역이 늘어남에 따라 키교환과 검증 로직 자체가 소비하는 시간은 큰 차이가 발생하지 않음을 확인할 수 있다. 이는 사용자가 체감하기는 힘든 속도 차이다.

표 9 제안방법의 효율성 검증
Table 9. efficiency verification of purposed method

상태		키교환 로직 작동	검증 로직 작동
미설치		1.33592 sec	0.28836 sec
BHO 설치	1개의 검사영역	1.58716 sec	0.34914 sec
	2개의 검사영역	1.61222 sec	0.34627 sec
	3개의 검사영역	1.57924 sec	0.34922 sec
	4개의 검사영역	1.58100 sec	0.36001 sec
	5개의 검사영역	1.58371 sec	0.35790 sec

V. 결 론

웹 브라우저에 의해 파싱된 HTML/XML 등의 문서는 DOM 정보들로 가공되어 사용자에게 보여 진다. 이 DOM 정보들은 다양한 경로를 통해 접근이 가능하며 악의적인 목적으로 변조될 수 있다. 이는 인터넷 주소를 다른 곳으로 강제로 이동 시키거나 사용자에게 잘못된 정보를 제공하여 정상적인 서비스를 방해할 수 있다. 특히, 인터넷 상거래나 금융 서비스에 대한 웹 페이지의 위조/변조는 서비스 제공자와 사용자에게 직접적으로 경제인 피해를 줄 수 있으며, 개인정보 등과 같이 민감한 정보들을 유출시킬 수도 있다. 또한, 다수의 사용자 PC에 설치되어 조직적으로 공격자의 명령을 실행할 수 있으며 이 때, 사용자는 자신도 모르게 악의적인 행위에 동참하게 될 수도 있다. 이러한 문제점으로 인해, 본 고 에서는 웹 서비스의 중요 컨텐츠를 미리 선별하여 HMAC으로 저장하고, 이를 기준으로 DOM 정보의 지속적인 무결성을 확인하는 웹 브라우저의 Add-on 프로그램의 설치를 통해 웹 페이지의 변조 행위를 탐지할 수 있다.

참 고 문 헌

[1] Johnny Stenback, "Document Object Model (DOM) Level2 HTML Specificatoin", *W3C Recommendation 09 January 2003*

[2] Suhit Gupta, "Dom-based Content Extraction of HTML Documents", *WWW2003, May 20-24,* 2003, Budapest, Hungary.

[3] L Wood, "Programming the Web: the W3C DOM specification", *Internet Computing, IEEE, 3(1),* 48 - 54, Jan, 1999

[4] Behrouz A. Forouzan, "Cryptography and Network Security", Pages 467-468

[5] MicroSoft corp. "http://www.microsoft.com/"

[6] AhnLAB, "http://kr.ahnlab.com/info/securityinfo/"

[7] Rolf Oppliger, "Internet security,firewalls and beyond", *Communications of the ACM archive, Volume 40, Issue 5 (May 1997),* Pages 92-102, 1997, ISSN:0001-0782

[8] Browser Helper Objects: The Browser the Way You Want It, Available: http://msdn.microsoft.com/en-us/library/bb250436(VS.85).aspx, Oct. 5, 2009.

[9] Diffie W., Hellman M., "New directions in cryptography", *Information Theory, IEEE, 22(6),* 644 - 654, Nov, 1976

[10] R.L. Rivest. The MD5 message-digest algorithm, Request for Comments (RFC 1320), *Internet Activities Board, Internet Privacy Task Force*, 1992.

[11] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC:Keyed-Hashing for Message Authentication," *RFC 2104,* February 1997.

[12] D. Eastlake and P. Jones, "RFC3174: US Secure Hash Algorithm 1(SHA1)", Available at http://www.faqs.org/rfcs/rfc3174.html, 2001.

[13] JooBeom Yun, Youngjoo Shin, HyoungChun Kim and Hyunsoo Yoon,"Miguard : Detecting and Guarding against Malicious Iframe through API Hooking", *IEICE Electronics Express,* Vol.8, No.7, 460-465, 2011

[14] Kangbin Yim, "Keyboard Security,"Workshop on Ubiquitous Information Security, May 2008

[15] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna. "Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis". *In Proceeding of the Network and Distributed system Security Symposium (NDSS'07),* February, 2007.

[16] Kyungroul Lee, Kwangjin Bae, Kangbin Yim, "Hardware Approach to Solving Password Exposure Problem through Keyboard Sniff," *ACADEMIC SCIENCE RESEARCH,* proceedings ofWASET2009, pp.23-25, Oct 2009

[17] JSON, "http://www.json.org/js.html"

[18] XML, "http://www.w3.org/XML/"

[19] DynaTrace, www.dynatrace.com/

모 정 훈 (牟禎勳)

2006년 2월 : 목포대학교 컴퓨터공학과
2006년 3월 ~ 현재 : 고려대학교 정보보호대학원 석사과정
관심분야 : 패턴인식, 시스템 보안, 네트워크 보안, 웹 보안

조 재 익 (趙宰翊)

2005년 2월 : 동국대학교 컴퓨터학과 학사
2008년 2월 : 고려대학교 정보경영공학전문대학원 석사
2008년 3월 ~ 현재 : 고려대학교 정보보호대학원 박사과정
관심분야 : 네트워크 모델링, 패턴인식

정 만 현 (鄭晩鉉)

2006년 2월 : 동국대학교 컴퓨터학과 학사
2009년 2월 : 고려대학교 정보경영공학전문대학원 석사
2010년 3월 ~ 현재 : 고려대학교 정보보호대학원 박사과정
관심분야 : 패턴인식, 시스템 보안, 네트워크 보안

문 종 섭 (文鍾燮)

1981년 2월 ~ 1985년 : 금성 통신 연구소 연구원
1991년 : Illinois Institute of technology 졸업(전산학 박사)
1993년 ~ 현재 : 고려대학교 전자 및 정보공학부 교수
관심분야 : 생체인식, 침입탐지, 운영체제