Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.4, Aug. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.4.635

웹 페이지에서 사용자 입력 값 변조 방지에 관한 연구*

유 창 훈,† 문 종 섭‡
고려대학교 정보보호대학원

A Study on Protecting for forgery modification of User-input on Webpage*

Chang-hun Yu,† Jong-sub Moon‡
Center for Information Security Technologies, Korea University

요 약

인터넷을 통하여 제공되는 대부분의 웹 서비스들은 웹 브라우저를 통하여 사용자에게 제공된다. 웹 브라우저는 텍스트 형태의 웹 페이지를 서버로부터 수신하여 해석하고 사용자에게 보여준다. 웹 브라우저는 추가적으로 설치 할 수 있는 각종 기능들을 통하여 확장성을 제공한다. 하지만 추가로 설치 할 수 있는 기능들도 웹 페이지에 접근하여 내용을 위/변조 할 수 있다는 점에서 웹 브라우저를 통한 웹 서비스는 보안상 문제점을 내포할 수 있다. 웹 브라우저는 웹 페이지정보를 DOM구조의 형태로 메모리에 저장한다. 웹 페이지의 변조를 방지하기 위한 방법으로는 DOM구조의 특정 부분에 해쉬(hash)값을 적용하는 방법이 있다. 하지만 웹 페이지의 특성상 해쉬를 이용한 대응방안이 효과를 발휘할 수 없는 부분이 있다. 즉, 사용자가 직접 입력하는 부분은 정해진 입력 값이 아니기 때문에 미리 해쉬 값을 계산 해 놓을 수도 없고 따라서 임의로 변조되는 것을 막을 수 없다. 본 논문에서는 웹페이지에 입력되는 사용자 입력 값의 위조나 변조를 방지 또는 탐지하는 방안을 제안한다. 제안 방법은 사용자가 키보드를 사용하여 입력하는 입력 값을 저장 해 놓았다가 웹 브라우저가 입력 값을 전송하는 순간 저장된 입력 값과 전송되는 값을 비교하여 변조 여부를 파악한다.

ABSTRACT

Most of the web-based services are provided by a web browser. A web browser receives a text-based web page from the server and translates the received data for the user to view. There are a myriad of add-ons to web browsers that extend browser features. The browser's add-ons may access web pages and make changes to the data. This makes web-services via web browsers are vulnerable to security threats. A web browser stores web page data in memory in the DOM structure. One method that prevents modifications to web page data applies hash values to certain parts in the DOM structure. However, a certain characteristic of web-pages renders this method ineffective at times. Specifically, the user-input data is not pre-determined, and the hash value cannot be calculated prior to user input. Thus the modification to the data cannot be prevented. This paper proposes a method that both detects and inhibits any attempt to change to user-input data. The proposed method stores user-input from the keyboard and makes a comparison with the data transmitted from the web browser to detect any anomalies.

Keywords: FinancialSecurity, Memory-Hacking, Webpage, Man-in-the-browser, Online-Banking, Malware

접수일(2014년 5월 8일), 수정일(2014년 7월 24일), 게재확정일(2014년 7월 30일)
* 본 연구는 미래창조과학부 및 한국인터넷진흥원의 "2014년도 고용계약형 정보보호 석사과정 사업"의 연구 결과로 진행되었음(과제번호 H2101-14-1001)
† 주저자, dbckdgns0515@korea.ac.kr
‡ 교신저자, jsmoon@korea.ac.kr(Corresponding author)

I. 서 론

서버에 저장된 웹 페이지는 사용자PC에 설치된 웹 브라우저를 통하여 접근할 수 있다. 웹 브라우저는 서버로부터 해당 웹 서비스에 대한 텍스트 형태의 정보를 전송받아 DOM(Document Object Model) 구조의 형태로 메모리에 저장, 해석하여 사용자에게 보여 준다[1]. 웹 브라우저들은 W3C (World Wide Web Consortium)에서 권고한 DOM 표준에 맞게 각자 API를 구현하여 웹 페이지를 해석하고 있다. 이러한 API들은 웹 브라우저에 설치 된 외부프로그램에 의해 접근이 가능하다. 웹 브라우저의 API를 사용하여 이미 만들어진 DOM을 수정할 수 있으며 원래의 웹 페이지와는 다른 정보가 사용자에게 전달될 수 있다[1][2]. 악성코드가 악의적인 목적을 가지고 API를 사용하여 웹 페이지를 변조한다면 보안메커니즘을 우회하여 로그인 정보와 같은 사용자의 비밀정보 획득이 가능하다. 또한 악성코드는 DOM에 접근하여 저장된 입력 값을 변조할 수 있다. 악성코드는 사용자 입력 값을 자신이 원하는 값으로 변조하여 사용자가 의도치 않는 거래를 수행하는 공격이 가능하다[3].

본 논문의 목적은 사용자가 웹 페이지를 통해 입력한 값이 악성코드에 의해 변조되는 것을 탐지 하는데 있으며, 사용자가 의도치 않은 행위를 방지하는 것이다. 본 논문에서는 키보드를 통하여 입력된 사용자 입력 값과 DOM에 저장된 값을 비교하여 악성코드에 의한 입력 값 변조 여부를 탐지하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서 사용자 입력 데이터 변조로 인한 문제점을 서술하고, 3장에서 논문에 서술된 관련 기술들과 관련연구를 설명한다. 4장에서는 제안하는 방안을 제시하고, 5장에서 제안 방법을 검증한다. 6장에서는 결론으로 본 논문을 서술한다.

II. 문제 정의

2.1 웹 페이지 변조 과정

웹 브라우저가 웹 서버에 접속하여 서버로부터 HTML/XML 문서를 전달받으면 웹 브라우저는 메모리상에 DOM 구조를 생성한다[1][4]. 이때, 악성코드가 DOM 구조에 접근하여 내용을 변조하고 사용자의 입력과는 다른 내용을 웹 서버로 전송한다.

Fig.1.은 악성코드에 의한 웹 페이지 변조공격 과정이다.

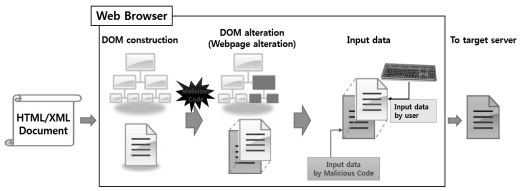


Fig. 1. Webpage Modification process

2.2 계좌이체 시 웹 페이지 변조 과정

사용자는 웹 브라우저를 통해 인터넷 뱅킹 사이트에 접속 후 사용자 인증을 거치고 나서 계좌이체 화면으로 이동한다. 웹 브라우저는 계좌이체에 해당하는 HTML/XML 문서를 전달받아 DOM 구조를 생성하고 화면에 내용을 표현한다.

국내 인터넷뱅킹 환경에서 계좌이체 과정은 예비거래, 본거래, 이체내역확인 과정으로 나눌 수 있다. 예비거래를 통하여 사용자가 입력한 입금계좌에 대한 정보(입금대상 계좌정보, 비밀번호, 이체 액수)가 세션키로 암호화 되어 서버로 전송되고 서버는 내용 확인 후, 대상 계좌의 완전한 정보(수신자 이름 추가)를 사용자에게 전송한다. 사용자는 내용을 확인 후에 본 거래를 진행한다. 본거래시 보안카드나 OTP (One-Time Password)같은 정보를 추가적으로 입력하여 서버에게 추가정보를 전송한다. 전송 시에 사용자의 개인키(공인인증서에 내재)로 서명함으로써 부인방지 역할을 한다.

인터넷 뱅킹 계좌이체 시, 문서변조 공격 수행 과정은 Fig.2.와 같다.

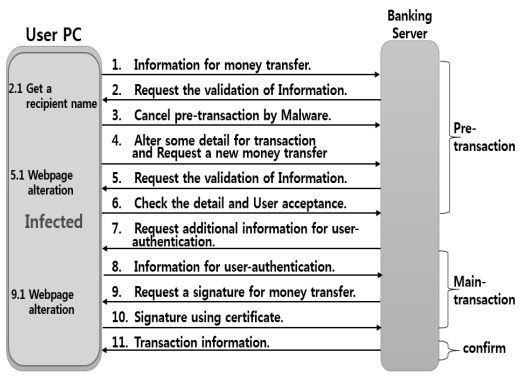


Fig.2. Money Transfer with Modified Webpage by a malicious code

악성코드가 이미 설치된 환경에서 예비거래 시, 악성코드는 사용자가 입력한 이체내용을 그대로 서버에 전송하여 수신자의 이름을 획득한다. 이후 사용자가 다시 계좌 이체를 시도할 경우 악성 코드는 작동을 시작한다. 즉, 사용자가 입력한 정보를 서버에 전송하지 않고(Fig.2.의 1, 2 그리고 3) 사전에 프로그래밍 되어있는 정보(공격자가 의도한 계좌이체정보)를 가지고 계좌이체를 진행한다. 악성코드는 DOM에 접근하여 이체정보의 입금은행, 입금계좌 및 이체금액을 악성코드가 사전에 미리 입력해놓은 공격자의 계좌번호와 입금은행 및 이체금액으로 변조하여 예비거래단계를 진행한다(4, 그리고 5). 이때 화면에는 계좌이체 확인화면이 보여 지는데 악성코드는 이 화면을 숨기고 미리 저장 해 놓은 이체 정보를 화면 상위계층으로 올려 사용자가 악의적인 행위를 인지하지 못하도록 한다. 본 거래에서도 웹페이지 화면을 조작하여 사용자가 의도한 거래처럼 사용자를 속이고 OTP나 보안카드와 같은 추가인증 정보는 그대로 이용한다[5][6][7]. 공인인증서를 이용한 부인방지용 서명과정에서도 화면을 조작하여 변조된 거래내역에 대한 서명을 유도한다[8]. 이체 내역 확인 단계에서도 화면을 조작하여 사용자는 인지하지 못하는 상황에서 거래가 성공적으로 종료된다.

III. 관련연구

웹 페이지 변조 공격은 사용자 웹 브라우저에 악성코드가 상주하여 사용자가 보는 웹 페이지를 변조하는 것으로 MITB(Man-in-the-browser) 형태의 공격방법이다[3]. 웹 브라우저에 악성코드가 상주하여 공격하는 것은 2005년 Augusto에 의해 처음 발표되었고, 2007년 Philipp에 의해 MITB라고 불려지게 되었다[3][9].

RSA에서는 금융거래시 웹 페이지 변조 공격에 대한 위험성에 대해 발표하였으며[10], Dougan T등의 연구에서는 MITB 공격에 사용되는 다양한 방법에 대해 설명하였다[11]. Rauti등의 연구에서는 Ajax를 이용한 웹 브라우저 환경에서의 웹 페이지 변조 공격에 대해 소개하였다[12].

MITB 공격에 대한 대응 방법으로 Entrust에서 Active-Safeguard와 Passive-Safeguard 라는 분류기준을 세워 소개하였다[13]. 또한 웹 컨텐츠를 변조하여 웹페이지를 변조 하는 공격에 대한 탐지방법으로는 "Hash"를 이용한 탐지 방법이 있다[14]. 사용자 웹 브라우저는 서버에서 고정적인 웹 컨텐츠에 대한 Hash 값을 전달받아 DOM의 구조 변경 시 또는 이벤트 완료 시 Hash값을 생성 비교함으로써 웹 컨텐츠의 변조 유무를 탐지한다. 하지만 금융권에서 보고되는 악성코드들은 DOM 구조에서 가변적인 영역 즉, 계좌번호와 같은 사용자의 입력을 통해 만들어지는 영역을 변조하고 있어 Hash를 이용한 탐지 방법으로는 한계가 있다.

3.1 문서 객체 모델(Document Object Model)

3.1.1 DOM 구성 및 형태

DOM은 W3C에서 권고하는 표준으로 플랫폼과 언어에 독립적으로 구조화된 문서를 표현하는 형식이다. 문서의 해석을 위한 표준적인 기능 정의를 제공할 뿐만 아니라 HTML/XML문서의 내용과 구조들을 조작할 수 있는 기능을 제공한다[2][4]. 또한 DOM은 관련 객체들에 대해 인터페이스를 정의한다. 사용자 PC에 설치된 웹 브라우저는 Fig.3. 과 같이 서버로부터 전송된 HTML/XML 형태의 문서를 전달받아 메모리에 트리 형태의 자료구조로 저장한다.

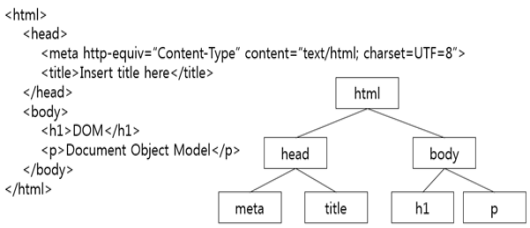


Fig.3. The Document Structure of a webpage

3.1.2 DOM을 통한 문서의 접근

브라우저는 W3C에서 제공한 DOM 표준에 맞추어 API를 구현하고 웹 페이지를 관리한다. 하지만 악의적인 목적을 가진 악성코드가 브라우저 내에 설치되면 DOM의 내용이 변조될 수 있고, 이는 보안의 취약점으로 악용 된다. Table 1. 은 DOM 인터페이스 중 트리 노드를 탐색하거나 수정하는 등 악의적으로 사용될 수 있는 기능들의 예를 간단히 소개한다.

Table 1. Web API Interfaces by W3C

Definition	Usage
Searching-Attribute	document.getElementById("id") ;
Searching-Name	document.getElementsByName ("Name");
Content modification	value=document.getElementBy Id("id"); value="Modified Content";
Attribute modification	value=document.getElementBy Id("id"); value.Setattribute=("align","ce nter");
Add node	document.Body.AppendChild(N odeName);

3.2 Keyboard 입력 처리과정

3.2.1 일반적인 처리과정

키보드는 PS/2 방식과 USB 방식이 있다. Fig.4.는 키보드 방식에 따라 인터럽트 발생 이후 데이터의 흐름을 표현한 것이다[15][16].

PS/2 port 또는 USB bus를 통해 키보드 입력 값이 접수 되면 운영체제는 K/B Stack 또는 USB Driver Stack을 통해서 키보드 입력 값을 임시 저장(버퍼)하게 된다. PS/2 방식에서는 선택사항으로 K/B Filter Driver를 통해 키보드 입력 값을 검사하기도 하며 USB 방식에서는 HID Class Driver에서 USB 장치가 입력장치(키보드/마우스)인지를 확인하고 제어한다. 또한 USB방식의 K/B Mapper Driver에서는 PS/2 키보드 입력 값으로 번역된다. K/B Class Driver에서는 키보드 입력 값의 메시지 변환 및 분기가 이루어지며 User- Interface 에서는 키 배열 설정에 따라 가상키 값으로 번역된다. System Message Queue와 Thread Message Queue 에서는 윈도우 메시지 처리과정을 통해서 키 입력 등 event를 알리기 위한 처리과정을 수행한다. 최종적으로 Application Programs에서는 키보드 입력 값을 접수하게 된다[16].

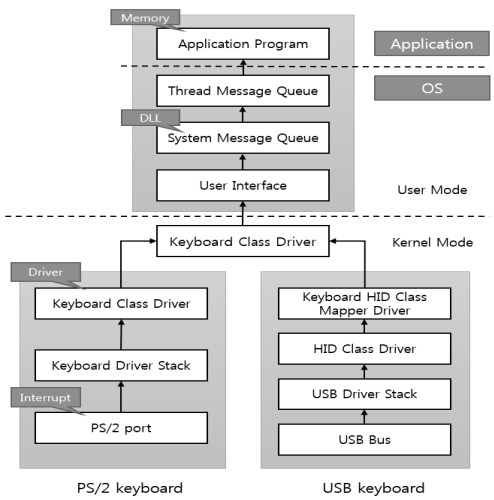


Fig.4. Process of transmitting keyboard input value

3.2.2 키 입력 데이터의 공격구간 및 키보드 보안

키 입력 데이터의 전송 흐름에서 키 입력을 가로채는 구간은 키보드 인터럽트 핸들러의 교체를 이용한 인터럽트 가로채기, 해킹용 키보드 드라이버를 개발해 기존 드라이버 대체나 앞선 수행을 통한 드라이버 해킹, 키보드 입력 값을 처리하는 실행코드의 메모리 영역에 키보드 입력 값을 빼내는 해킹코드를 기록한 후 실행시키는 DLL Injection, BHO (Browser Helper Object)나 메모리덤프를 이용한 메모리 저장 값 유출 등으로 다양하게 존재한다[16].

국내에 키보드 보안프로그램들은 Fig.4.의 각 구간들에 대해서 구현 방법에 따라 키로거와 같은 행위를 하는 공격으로부터 키 입력 데이터를 보호하고 있다[17]. Fig.5.는 키보드 보안 프로그램 설치에 따른 키보드 입력 정보 처리 절차이다[16][18]. 사용자가 키보드 보안프로그램이 적용 된 입력창을 클릭하면 키보드 보안프로그램이 동작하게 된다. 이때, 설치 된 '보안 키보드 드라이버'가 '기존 키보드 드라이버' 대신 호출되어 입력 값 전달 흐름을 제어한다. 사용자가 키보드를 입력하면 보안 키보드 드라이버내 버퍼에 입력 값이 저장된다. 동시에 Null 값이 System Message Queue에 저장되어 user level에서 동작하는 키로거에 의한 입력값 탈취를 차단한다. 또한 보안 키보드 드라이버와 보안입력창 제어부간의 직접적인 통신을 통해 입력창에 특수문자(*)를 표시하도록 지시한다. 이후 확인 버튼이 눌러지면 보안 키보드 드라이버 버퍼에 저장된 입력값을 SEED나 AES 암호

화를 통하여 암호화 한다. 암호화된 값을 직접 보안 입력 창 제어부로 전송하고 입력창제어부에서 복호화 하여 응용프로그램에 전달한다[16][18][19]. 그러나 사용자가 화면을 통해 입력 값을 확인해야 하는 계좌 번호와 같은 정보는 키보드 보안을 거치더라도 DOM에 입력되기 전에 평문으로 복호화 되어 저장되며 DOM 구조를 통해 직접 확인가능하고 웹 브라우저 내에서 접근하여 변조가 가능하다.

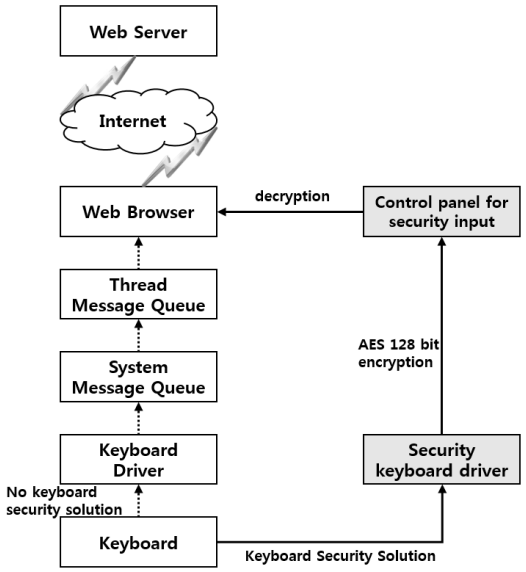


Fig.5. Transmission Process of Keyboard input value

3.3 Keyboard Event & Internet Explorer Event

웹 브라우저 및 DOM 에서는 W3C의 권고에 의하여 모든 키보드 관련 이벤트가 정의되어 있으며 스크립트 수준에서 이벤트를 사용할 수 있다[20]. Table 2.와 같이 웹 브라우저는 다양한 이벤트를 제공한다.

Table 2. Keyboard event Defined by W3C

Event Type	Attribute	Description
Keydown	onkeydown	Event is trigge- red when the user presses a key.
Keypress	onkeypress	Event is trigge- red when the user input cha- racter
Keyup	onkeyup	Event is trigge- red when the user releases a key.

특히 keydown, keyup, keypress 이벤트를 통하여 키보드 입력에 반응하며 키보드 입력 값을 가져올 수 있다.

MS Internet Explorer는 Table 3.과 같이 DOM해석 시점을 기준으로 다양한 이벤트를 제공한다[21]. 특히 웹 브라우저의 윈도우나 프레임 셋 구성 요소가 특정 웹사이트로의 이동이 이루어지기 전에 발생하는 BeforeNavigate 와 DOM 해석이 완료된 후 발생하는 DocumentComplete 이벤트에 대한 핸들러를 사용자가 정의하면 웹 브라우저는 DOM이 변경되었을 때마다 사용자가 정의해 놓은 핸들러를 호출 한다.

Table 3. Event handlers of MS Internet Explorer

Event	Description
BeforeNavigate	Fires before naviga- tion occurs in the given object.
DocumentComplete	Fires when a docu- ment is completely loaded and initializ- ed
WindowStateChanged	Fires when the visibility state of a content window, such as the browser window or a tab, changes.

프로그래머는 keybd_event()(Winuser.h에 정의)함수[22]를 통하여 키보드 입력을 조정하는 것이 가능하다. Keybd_event() 함수는 사용자가 키보드를 눌러 이벤트를 발생시켰을 때와 동일한 이벤트를 발생시키는 함수이다. Keybd_event() 함수를 사용하여 프로그래밍 하면 윈도우에서 제공하는 화상 키보드를 구현할 수 있고, 원격에서 인터넷 뱅킹이 가능하도록 가상 키보드 구현이 가능하다.

IV. 키보드입력 데이터 변조 탐지에 대한 제안

4.1 제안방법 개요

웹 페이지 변조를 통한 사용자 입력값 변조 공격은 사용자가 키보드를 통해 입력한 값과 서버에 제출되는 DOM 구조에 들어있는 값이 다름으로써 발생하는 문제이다. 본 논문에서는 키보드를 통한 사용자 입력 값과 서버로 제출되는 DOM에 저장되어있는 값을 비교

함으로써 사용자 입력값 변조 유무를 판단한다.

4.2 제안방법 원리

웹 브라우저는 키보드로부터 값을 수신하였을 때 이벤트를 발생시키고 수신된 입력 값을 획득 할 수 있는 인터페이스를 제공한다. 이를 통해 키보드로부터 수신된 입력 값을 특정 메모리에 임시로 저장할 수 있다. 또한 브라우저는 DOM에 접근할 수 있는 인터페이스를 제공하므로 특정 시점에서 DOM에 저장된 값을 확인하여 비교할 수 있다.

수집된 키보드 입력 값과 DOM에 저장된 값을 비교하는 시점은 웹 페이지 문서가 서버로 전송되기 바로 전 단계인 제출 이벤트가 발생하는 단계가 가장 적절하다. 제안하는 기법의 순서도는 Fig.6. 과 같다.

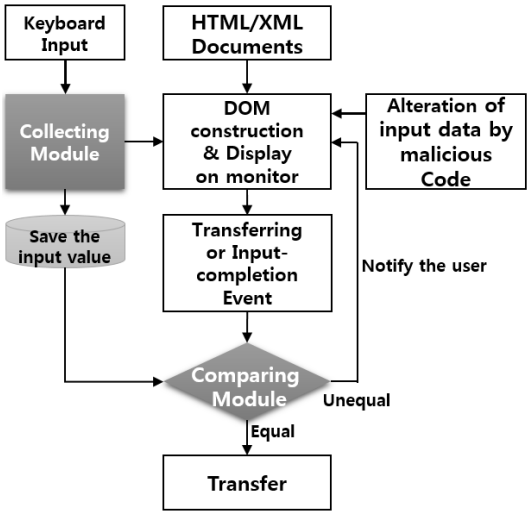


Fig.6. Flowchart of the suggested method

Fig.6. 에서 웹 브라우저가 문서를 수신한 이후 각 use case의 역할은 다음과 같다.

4.2.1 웹 브라우저

웹 브라우저는 웹 페이지 문서 수신 후 DOM 구조를 생성하고 화면에 내용을 표현한다. 이후 웹 페이지에 표현된 입력 폼에 사용자가 키보드를 통해 값을 입력한다. 키보드 입력 값에 대한 가상키 값으로 변환되고 키 입력 이벤트가 발생한다.

4.2.2 수집모듈

키 입력 이벤트 발생 시 이벤트를 감지한다. 키 입력 이벤트 감지 시 키보드 입력 값을 수집하여 저장한다.

4.2.3 악성코드

악성코드가 사용자 입력 값이 저장된 DOM에 접근하여 미리 프로그래밍 되어 있는 값으로 변조한다. 악성코드는 자신의 행위를 감추기 위해 사용자가 보는 웹 페이지 화면을 변조하거나 특정 웹 페이지로 이동하기 위한 제출 이벤트를 발생 시킨다.

4.2.4 비교모듈

웹 브라우저는 악성코드나 사용자에 의해 DOM 변경 이벤트나 제출 이벤트를 발생시킨다. 이벤트 감지 시 현재 DOM 구조 입력 폼에 저장되어 있는 값을 수집한다. 수집모듈에서 수집한 키보드 입력 값과 비교모듈에서 수집한 DOM 에 저장되어 있는 값을 비교한다. 값이 같으면 키보드 입력 값과 DOM 에 저장된 값이 일치하므로 변조가 일어나지 않음을 확인 할 수 있다.

4.3 수집모듈과 비교모듈

본 논문에서는 제안하는 기법을 역할에 따라 두 가지 모듈로 나누었다. 키보드를 통한 사용자 입력 값을 수집하는 수집모듈과 실제 DOM에 저장된 값을 가져와 키보드 입력 값과 비교하는 비교모듈 이다.

수집모듈은 키보드 입력 이벤트를 감지하고 이벤트 발생 시 키보드 입력 값을 메모리에 저장하여 보관한다. 저장된 입력 값은 비교모듈에서 가져갈 수 있다. 비교모듈은 사용자의 입력이 끝난 후 웹 서버로의 전송관련 이벤트를 탐지한다. 이벤트가 탐지되면 수집모듈에서 저장했던 키보드 입력 값에 접근하여 값을 가져와 DOM구조에 저장된 값과 비교를 수행한다. Fig.7.은 본 논문에서 제안하는 수집모듈과 비교모듈을 표현한 것이다.

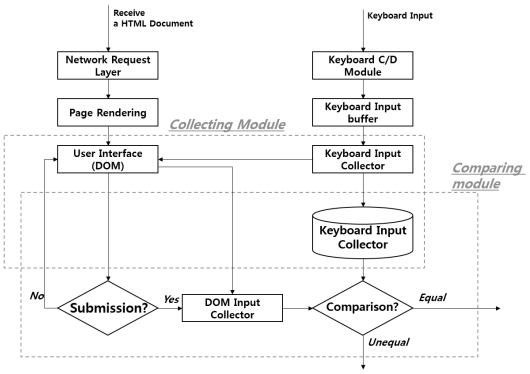


Fig.7. Keyboard input collection module and Value comparing module

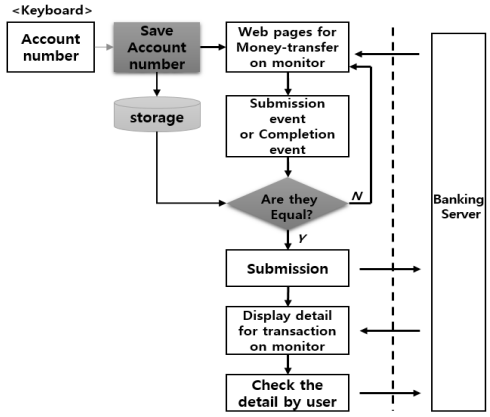


Fig.8. Experimental Simulation

V. 제안방법에 대한 검증

본 장에서는 논문에서 제안한 방법을 구현하고 실제 웹 브라우저에서의 실험을 통해 웹 페이지 변조 방지가 이루어지는 것에 대한 검증을 수행하였다. 실험 환경은 Windows 7 32bit 운영체제와 Intel Core i7-3770 CPU, DDR3 4G 램의 환경에서 실험하였다.

각 모듈의 구현은 html 문서의 head 부분에 자바스크립트를 직접 구현하는 방법과 사용자 스크립트를 통하여 브라우저에 스크립트를 삽입하는 방법, 또는 서버에서 ActiveX나 BHO를 통하여 사용자 컴퓨터에 설치되는 방법 등이 있다. 본 논문에서는 html 문서에 직접 스크립트를 삽입하고 사용자 스크립트를 추가 하는 구조로 구현하였다.

Fig.8.은 공인인증서를 사용한 인터넷 뱅킹의 계좌이체 과정에서 본 논문에서 제안하는 기법을 적용한 것이다. 실험은 사용자가 계좌번호와 비밀번호를 입력하여 서버로 전송하는 Fig.8. 과 동일한 구조로 html문서를 구현하였고, 수집모듈과 비교모듈은 스크립트 형태로 구현하였다. 악성코드의 동작 구현은 개발자도구를 이용하여 DOM구조의 계좌번호 입력란에 접근, 변조하였다. 계좌이체정보 입력과정에서 수집모듈에 의해 사용자가 키보드를 통해 입력한 수신자 계좌번호가 저장된다. 모든 이체정보 입력이 끝나고 이체실행 버튼을 누르면 비교 모듈이 동작하여 DOM 구조에 저장된 변조된 수신자 계좌번호와 수집모듈에 의해 저장된 키보드를 통해 입력된 수신자 계좌번호를 비교한다.

수집모듈 구현은 웹 브라우저에서 키보드입력에 대한 이벤트 감지를 위해 keypress 이벤트를 사용하였다. 특정 입력 폼에 keypress 이벤트와 스크립트 함수로 구현된 핸들러에 연결하여 이벤트 발생 시 입력키 값을 메모리에 배열형태로 저장하도록 하였다. 비교모듈 구현은 전송버튼 객체에 onclick 이벤트와 스크립트 함수로 구현된 핸들러를 연결하였다. 이벤트 발생시 DOM구조에 접근하여 특정 입력 폼에 저장된 값을 가져와 수집모듈에 의해 저장된 입력값과 비교한다.

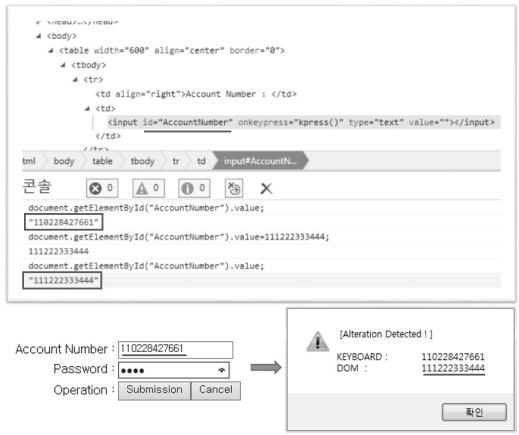


Fig.9. Verification of purposed method

사용자가 키보드 입력을 통해 계좌번호를 입력하였고, 악성코드의 동작처럼 DOM구조에 접근하여 계좌번호를 변조하였다. 논문에서 제안한 기법을 통해 키보드에서 입력 받은 값과 DOM 구조에 저장된 값이 다름을 확인하여 웹 페이지 문서 변조를 탐지할 수 있었다.

VI. 결론 및 향후 연구

본 논문에서는 웹 페이지 변조를 통한 사용자 입력 값 변조 공격 방법을 설명하고, 방지 대책에 대해 제안하였다. 최근 웹 페이지 변조공격을 통한 범죄가 증가하면서 사용자의 안전한 인터넷 환경을 위협하고 있다. DOM 구조의 특성상 접근과 변조가 용이하지만 악의적인 접근에 대해 판단이 어렵다는 점을 악용하여 공격이 시도 되고 있다. 또한 사용자가 정상이라고 인지하고 있는 상황에서 사용자 몰래 악의적인 행위를 한다는 점에서 이전 공격보다 고도화되고 교묘해졌다.

실제 서버에 전송되는 DOM에 담겨있는 정보는 컴퓨터 메모리에 존재하는 것으로 악성코드가 쉽게 변조가 가능하다. 특히 웹 브라우저에서 플러그인 형태로 제공되는 BHO나 ActiveX형태의 모듈들은 Internet Explorer 프로세스와 쉽게 연동되어 접근이 용이하다. 이러한 메모리 접근의 취약성을 이용한 공격 기술이 금융권이나 민감한 정보를 담고 있는 웹 페이지를 대상으로 할 경우 그 파급 효과는 심각할 것이다. 따라서 웹 페이지 변조를 방지하기 위한 방안은 중요 웹 페이지 보호를 위해 반드시 필요한 항목이다.

본 논문에서는 클라이언트에서 키보드 입력과 DOM에 저장된 값을 비교하는 방법을 통하여 웹 페이지 변조여부를 탐지하는 방안을 제안하였지만 향후에는 서버 단에서 웹 페이지 변조 여부를 탐지하는 방안에 대한 연구를 수행할 것이다.

References

[1] Joe Marini, The Document Object Model: Processing Structured Docume -nts, McGraw-Hill/Osborne, 2002.

[2] Lauren Wood, "Programming the Web : the W3C DOM specification," Inter- net Computing, IEEE, vol. 3, no 1, pp. 48-54, Jan. 1999.

[3] Philipp Gühring, "Concepts against Man-in-the-Browser Attacks," http://www.cacert.at/svn/sourcerer/CAcert/SecureClient.pdf

[4] Suhit Gupta, "Dom-based Content Extraction of HTML Documents," WWW-2003, May. 2003.

[5] N. Haller, C. Metz, P. Nesser and M. Straw, "A one-time password system," IETF RFC 2289, Feb. 1998.

[6] H.W.Sim, W.J.Kang and H.Y.Park, "An one time password based e-finan -cial transaction verification protocol," TTAK.KO-12.0167, Dec. 2011.

[7] Alain Hiltgen, Thorsten Kramp and Thomas Weigold, "Secure Internet Banking Authentication," IEEE Security and Privacy, vol. 4, no 2, pp. 21-29, April. 2006.

[8] Young-Jae Maeng, Dong-Oh Shin, Sung-Ho Kim, Dae-Hun Nyng and Mun-Kyu Lee, "A Vulnerability Analy- sis of MITB in Online Banking Trans- actions in Korea," Internet and Information Security vol. 1(2), pp. 101-118, Nov. 2010.

[9] Gerson Paes de Barros and Augusto Paes de Barros, "O futuro dos backdoors - O pior dosmundos," http://www.paesdebarros.com.br/backdoors.pdf

[10] RSA, "Making sense of man-in-the-browser attacks:Treate analysis and mitigation for financial institution," http://www.julesbartow.com/Downloads/Making_Sense_of_Man-in-the-Browser_Attacks.pdf

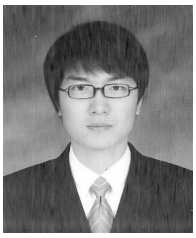
[11] Timothy Dougan and Kevin Curran, "Man in the Browser Attacks," International Journal of Ambient Computing and Intelligence, vol. 4, no. 1, pp. 29-39, Jan-Mar. 2012.

[12] Rauti Sampsa and Lappanen Ville, "Browser Extension-Based Man-in-the--Browser Attacks against Ajax Applic -a-tions with Countermeasures," Proceeding of the 12th International Conference on Computer Systems and Technologies, pp. 251-258, ACM, 2012.

[13] Entrust, "Defeating Man-in-the-Browser Malware : How to prevent the latest malware attacks against consumer and corporate banking," http://entrust.wpengine.netdna-cdn.com/wp-con-

tent/upoads/2014/03/WP_Entrust-MITB_March2014.pdf
〔14〕 Jeong-Hoon Mo, Man-Hyun Chung, Jae-Ik Cho and Jong-Sub Moon "Web contents deformation detection method by BHO," Journal of Advanced Navigation Technology, vol. 15(4), pp. 655-663, Aug. 2011.
〔15〕 Shin-Bum Kang and Hyun-Chul Jung, "Countermeasure and type of hacking on Internet banking," Korea Institute of Information Security & cryptology Conference, vol. 15(4), pp. 28-37, Aug. 2008.
〔16〕 KF/ISAC, "technique of keyboard hacking and Analysis of the countermeasure," Korea Financial Information Sharing & Analysis Center, Nov. 2005.
〔17〕 Muzammi Baig and Wasim Mahmood, "A Robust Technique of Anti Key- Logging using Key-Logging Mechani- sm," Digital EcoSystems and Technol- ogies Conference `07. Inaugural IEEE- IES, pp. 314-318, Feb. 2007.
〔18〕 INKA-Internet, "K.R. Patent No. 10-2004-0009575: Hacking prevention of key stroke data," Jan. 2004.
〔19〕 Soft Camp "K.R. Patent No. 2002-0048313," Mar. 2002.
〔20〕 World Wide Web Consortium, "Document object model level-3 events specification," http://www.w3.org/TR/DOM-Level-3-Events/
〔21〕 Internet Explorer Event, http://msdn.microsoft.com/en-us/library/aa768400(v=vs.85).aspx
〔22〕 keydb_event, http://msdn.microsoft.com/en-us/library/windows/desktop/ms646304(v=vs.85).aspx

〈저자소개〉



유 창 훈 (Chang-hun Yu) 학생회원
2010년 2월: 안동대학교 정보통신공학과 공학사
2013년 3월 ~ 현재: 고려대학교 정보보호대학원 금융보안학과 석사과정
〈관심분야〉 시스템보안, 금융보안, 정보보호

문 종 섭 (Jong-sub Moon) 종신회원
1981년 1월: 서울대학교 계산통계학과 졸업
1983년 1월: 서울대학교 대학원 계산통계학과 석사
1991년 5월: Illinois Insitute of Technology 전산학 박사
2002년 3월 ~ 현재: 고려대학교 전자 및 정보공학과 교수
〈관심분야〉 정보보호, 전자공학, 통신공학