http://dx.doi.org/10.7840/kics.2014.39B.3.162

서버와 사용자간 비밀 값을 이용한 보안성이 강화된 CSRF 방어

박 진 현◆, 정 임 영°, 김 순 자*

Enhanced CSRF Defense Using a Secret Value Between Server and User

Jin-hyeon Park◆, Im Y. Jung°, Sun-ja Kim*

요 약

Cross-Site Request Forgery(CSRF)는 오늘날 인터넷 환경에서 발생하는 악의적인 공격방식 중 하나이다. 이는 권한이 없는 공격자가 사용자의 브라우저를 통해 웹 서버에 정당한 요청을 전송 하도록 한다. 공격자에 의한 요청은 웹 서버에서 정상적인 요청으로 판단되어 사용자가 원하지 않는 결과를 가져온다. 이러한 문제는 웹 서버에서 쿠키에 포함된 정보만으로 사용자를 식별하기 때문에 발생한다. 본 논문에서는 쿠키에 포함된 정보 이외에 페이지 식별번호와 사용자 비밀번호의 해시 값을 추가하여 요청을 검증하는 보안성이 강화된 CSRF 방어를 제안한다. 이는 구현이 간단하며 기존 CSRF 대응법으로 알려진 일회성 토큰을 이용한 방식의 문제점인 토큰 노출의 문제점을 해결한다.

Key Words : Cross-Site Request Forgery, Authentication, Web Security, Browser Security

ABSTRACT

Cross-Site Request Forgery is one of the attack techniques occurring in today's Web Applications. It allows an unauthorized attacker to send authorized requests to Web Server through end-users' browsers. These requests are approved by the Web Server as normal requests therefore unexpected results arise. The problem is that the Web Server verifies an end-user using his Cookie information. In this paper, we propose an enhanced CSRF defense scheme which uses Page Identifier and user password's hash value in addition to the Cookie value which is used to verify the normal requests. Our solution is simple to implement and solves the problem of the token disclosure when only a random token is used for normal request verification.

Ⅰ. 서 론

Cross Site Request Forgery(CSRF) 공격은 웹 환경에서 발생할 수 있는 공격 중 하나이다[1-4]. 이는 사전에 웹 서버에 인증된 사용자를 대상으로 하는 공격으로, 사용자가 의도하지 않은 요청을 웹 서버로 전송하는 방법이다. 공격자는 웹 서버와 인증을 수행하지 않은 상태에서 인증된 사용자의 권한을 이용하여 웹

※ 본 연구는 2013년도 경북대학교 신임교수정착연구비에 수행되었습니다.
◆ First Author : 경북대학교 전자공학부 컴퓨터통신망 연구실, helloworld@knu.ac.kr, 학생회원
° Corresponding Author : 경북대학교 전자공학부 차세대 IT융합보안 연구실, iyjung@ee.knu.ac.kr, 종신회원
* 경북대학교 전자공학부 컴퓨터통신망 연구실, snjkim@ee.knu.ac.kr, 종신회원
논문번호 : KICS2013-12-548, Received December 23, 2013; Reviewed February 11, 2014; Accepted February 26, 2014

서버로 요청 메시지를 전송한다. 이 악의적인 요청은 사용자의 브라우저를 통해 웹 서버로 전송되기 때문에 웹 서버에서는 정상적인 사용자의 요청과 구별할 수 없다.

웹 서버에는 다양한 광고뿐 아니라 사용자가 직접 작성할 수 있는 게시판과 같은 페이지를 포함하고 있다. 공격자는 이러한 광고 또는 게시판에 악성 스크립트를 삽입하여 사용자가 접근하기만을 기다린다[5,6]. 이 스크립트는 사용자가 접근할 시 자동으로 실행되며, 공격자가 원하는 요청 메시지를 웹 서버로 전송하게 된다. 이와 같은 스크립트를 사용하는 방법 뿐 아니라 웹 서버 게시판에 링크를 삽입하여 사용자가 공격자의 사이트로 접속을 하도록 유도할 수 있다[5,7]. 사용자가 공격자의 사이트로 접속을 하게 되면 공격자가 사전에 준비한 악의적인 URL을 아래와 같이 이미지 형태로 사용자에게 보여준다.

<img src="http://example.com/img/example.jpg">

이 이미지를 클릭할 시 해당 URL은 웹 서버로 전송이 되고 공격자가 원하는 결과가 웹 서버에서 실행된다.

이와 같은 문제는 POST 요청과 GET 요청 모두 웹 서버에서 동일한 수행을 하도록 구현되어 있기 때문이며, 또한 헤더 필드에 포함된 인증 정보를 기반으로 사용자를 식별하기 때문이다. 즉, 요청의 유효성을 판단하기 위해서 Cookie에 포함된 정보만을 사용하기 때문에 문제가 발생한다. 사용자는 웹 서버에 인증을 완료한 상태이기 때문에 유효한 Cookie 값을 가지고 있다. 따라서 사용자 브라우저를 통해 웹 서버로 전송되는 요청 메시지는 검증에 필요한 값을 자동으로 포함하게 된다.

본 논문에서는 사용자를 식별하기 위해 Cookie에 포함된 세션ID 이외에 페이지를 식별하기 위한 페이지식별번호를 사용하는 방법을 제안한다. 또한 요청 순간 검증에 필요한 값을 생성하기 위해서 사용자와 웹 서버 사이의 비밀 값을 활용한다.

본 논문의 전개는, 2장에서는 CSRF 공격의 시나리오를 보여주고, 3장에서 공격에 대응하기 위한 방법을 제안한다. 4장에서 제안 기법을 평가한다. 5장에서 관련 연구를 소개한 뒤 마지막으로 6장에서 결론과 향후연구를 기술한다.

Ⅱ. CSRF 공격 시나리오

그림 1은 CSRF공격에 대한 일반적인 시나리오를 보여준다[7-9]. 사용자가 웹 서버에 인증 과정을 완료하고 서비스를 이용하는 상황에서 특정 광고 또는 게시판에 포함된 링크를 통해 공격자의 사이트에 접속하게 된다. 또는 웹 서버의 서비스를 이용하면서 동시에 웹 서핑을 통해 공격자의 사이트에 우연히 접속하게 된다. 사용자가 악의적인 사이트에 접속 시 공격자는 위조된 URL을 이미지나 링크 형태로 제시하여 사용자의 클릭을 유도한다.

공격자는 사전에 패킷 분석 툴을 이용하여 웹 서버의 요청, 응답 메시지를 분석하고 이와 같은 악의적이고 웹 서버에서 정상적으로 수행이 가능한 URL을 준비한다.

사용자가 공격자에 의해 미리 준비된 URL을 클릭하게 되면 검증에 필요한 값은 자동으로 헤더에 포함되어 웹 서버로 전송이 되고, 웹 서버는 해당 요청을 정상적인 사용자의 요청으로 판단한다.

그림 1의 각 절차는 다음과 같다.

1) 사용자는 웹 서버와 인증 절차를 수행한다.
2) 사용자와 웹 서버 사이의 정상적인 요청과 응답 과정이다. 공격자는 이 요청과 응답 메시지를 분석함으로써 악의적인 URL 생성이 가능하게 된다.
3) 사용자는 웹 서버에 포함된 광고 또는 게시판의 링크를 통해서 공격자의 사이트에 접속하게 된다.
4) 사용자는 아래와 같이 악의적인 URL을 포함하는 이미지를 클릭한다.

<img src="http://www.example.com/transferCash.php?acc=Ali

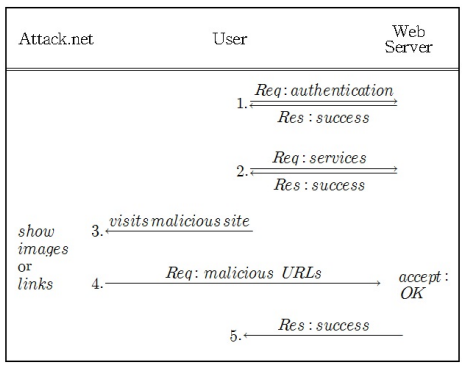

그림 1. 사이트간 위조 공격 시나리오
Fig. 1. CSRF Attack Scenario

ce&amount=999&where=Eve">

이 후 자신이 원하지 않은 요청을 웹 서버로 전송하게 된다. 사용자의 Cookie 값이 유효하다면 이와 같은 요청은 웹 서버에서 정상적인 사용자의 서비스 요청으로 판단되고, 그 결과 사용자가 의도하지 않는 작업이 웹 서버에서 실행된다.

5) 웹 서버는 요청을 정상적으로 수행 후 사용자에게 응답한다.

Ⅲ. 제안 기법

사용자는 웹 서버에 인증을 정상적으로 수행하였다고 가정한다.

사용자의 페이지 요청 시 웹 서버는 세션ID외에 페이지식별번호를 랜덤하게 생성하여 사용자에게 응답한다. 요청 페이지가 form 필드를 포함하는 경우 페이지식별번호는 페이지 body의 hidden 필드 또는 Cookie에 저장될 수 있다. form 필드를 포함하지 않는 페이지일 경우 페이지식별번호는 Cookie에 저장되어 사용자에게 전송되어야 한다.

사용자는 웹 서버로부터 페이지를 수신 받은 후, 이 페이지에 대한 응답으로 검증 값을 포함시킨다. 검증에 필요한 값은 세션ID, 페이지식별번호 그리고 사용자 비밀번호의 해시 값을 통해서 생성된다. 이 세 가지 값을 연접 후 SHA1 함수에 입력하고 함수의 결과 값을 검증에 사용한다. 사용자의 비밀번호는 웹 서버에 바로 저장되는 것이 아닌 비밀번호를 해시한 값이 저장되는데 이 값을 활용하여 검증에 필요한 값을 생성 하였다. 이 해시 값은 오직 사용자와 웹 서버만이 공유하고 있는 값이다. 세션ID와 페이지식별번호는 악성 스크립트에 의해 노출될 수 있지만 사용자와 웹 서버만이 공유하고 있는 비밀 값을 사용함으로써 공격자가 검증에 필요한 유효한 값을 생성할 수 없도록 한다.

사용자에 의한 서버로의 요청이 POST 방식일 경우 검증 값은 페이지 body의 hidden 필드 또는 Cookie에 저장한다. GET 방식일 경우 검증 값은 Cookie에 저장하여 웹 서버로 응답한다.

웹 서버는 요청을 수신한 후 세션ID를 통해 사용자를 식별한다. 유효한 세션ID로 판단이 되면 데이터베이스에서 사용자 비밀번호의 해시 값을 가져온다. 이 후 세션ID와 전송 받은 페이지식별번호 그리고 사용자 비밀번호의 해시 값을 연접하여 사용자와 동일한 방식으로 해시 값을 생성한다. 이 해시 값과 사용자가 전송한 해시 값을 비교하여 일치여부를 판단한다. 일치 시 정당한 요청으로 판단하며, 불일치하는 경우 요청은 거절된다.

그림 2는 그림 1의 CSRF 공격에 대응하는 본 논문의 제안 도식을 보여준다.

그림 2의 각 절차는 다음과 같다.

1) 사용자는 웹 서버에 특정 페이지를 요청한다.
2) 웹 서버는 페이지식별번호를 생성한 후 다음과 같이 페이지 body의 hidden 필드에 값을 포함시킨다.

<input type="hidden" name="Pageidentifier" value="MTM4Njc0NDg5NzlMRm96R1pxWGVzVk5BZ09oSjlpR3pGTmx4b0lsRElS">

또한 검증 값은 다음과 같이 공백으로 설정 한다. 이 필드의 값은 사용자에 의해 채워지는 값이다.

<input type="hidden" name="verValue" value="">

form 필드를 포함하지 않는 페이지일 경우 페이지식별번호와 검증 값은 Cookie에 저장되며, 이 때 검증 값은 공백으로 설정된다. 이 후 Session ID와 함께 사용자에게 전송한다.

3) 사용자는 CryptoJS.SHA1 함수를 이용하여 검증 값을 생성한다. 함수의 입력 값은 Session ID, Pageidentifier 그리고 자신의 비밀번호의 해시 값(a)을 연접한 값이다. 생성된 값 b'는 다음과 같이 설정한다.

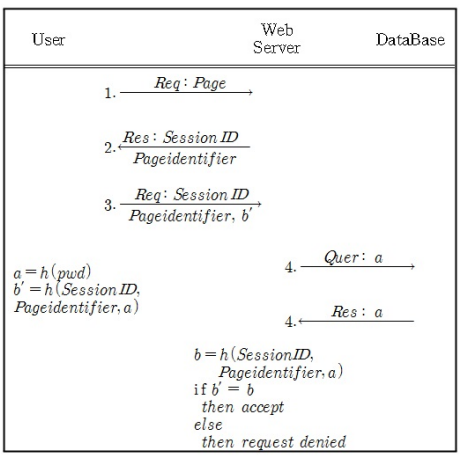

그림 2. 비밀번호와 페이지식별자 그리고 세션ID를 이용한 CSRF 방어
Fig. 2. Proposed CSRF Defense Using a Password, Pageidentifier and SessionID

<input type="hidden" name="verValue" value="b'">

form 필드를 포함하지 않는 경우 이 값은 Cookie에 설정된다. 이 후 웹 서버로 응답한다.

4) 웹 서버는 세션의 유효성을 판단하고 페이지 body의 hidden 필드 또는 Cookie에서 필요한 값을 가져온다. 그리고 데이터베이스에서 사용자 비밀번호의 해시 값을 가져온다. 값을 받은 후 사용자와 동일한 방식으로 b값을 생성한다. 수신한 b'값과 비교하여 일치 시 요청을 수행하며, 불일치 시 요청을 거부한다.

Ⅳ. 평 가

이 장에서는 제안 방식의 보안성과 성능을 평가하기 위해서 실제 구현을 통해 실험을 수행 하였다. 4.1절에서 실험환경을 기술하고, 4.2절에서 성능 및 오버헤드에 대해서 분석한다. 이를 통해 웹 서버에 추가로 삽입되는 루틴으로 인해 성능에 어떤 변화가 있는지 나타낸다. 4.3절에서는 일회성 토큰을 사용한 방식과 비교하여 더 나은 보안성을 제공한다는 것을 보여준다.

4.1 실험 환경

실습에 사용한 컴퓨터는 32비트로 동작하며, Intel Core i3 3.20GHz CPU를 사용하고 메모리는 4GB이다. 가상머신을 이용해 리눅스를 설치한 후 리눅스에 웹 서버를 구동 하였다. 웹 서버의 버전은 Apache-2.0.54이며, 사용하는 데이터베이스는 MySQL-4.1.20을 사용 하였다. 사용한 웹 기술은 php이며, IE10을 통해서 웹 서버에 접속 테스트를 하였다. 또한 임의의 수많은 요청을 전송하기 위해서 java.net 패키지를 이용하여 클라이언트를 구현하였다.

웹 서버는 사용자의 POST요청을 받은 후 데이터베이스에 입력을 수행하고 사용자에게 응답한다. 이 때 일회성 토큰을 이용한 방식과 제안 방식의 각 단계는 아래에 나타나 있다. 이 두 경우 각각에 대해서 사용자의 입력 요청(POST 요청)에 대한 웹 서버의 응답 시간을 자바 클라이언트를 이용하여 측정하였다.

일회성 토큰을 이용한 방식

1) 사용자는 form 필드를 포함하는 페이지를 요청한다.
2) 웹 서버는 랜덤 토큰 값을 생성하여 페이지의 body에 포함하고, 동시에 Cookie에도 토큰 값을 저장한다.
3) 사용자는 데이터 입력 후 웹 서버에 데이터베이스 입력 수행을 요청한다.(POST 요청)
4) 웹 서버는 요청 페이지의 body에서 토큰 값을 추출하고, 동시에 Cookie에서 토큰 값을 가져와서 두 값을 비교한다.
5) 토큰 값이 동일하면 데이터베이스 입력을 수행하고 사용자에게 결과를 응답한다.

제안 방식

1) 사용자는 form 필드를 포함하는 페이지를 요청한다.
2) 웹 서버는 페이지식별자를 생성 후 페이지의 body에 포함하여 응답한다.
3) 사용자는 데이터 입력 후 검증 값을 생성한다.
4) 생성한 검증 값을 페이지의 body에 포함하여 웹 서버로 전송한다.(POST 요청)
5) 웹 서버는 세션ID를 통해 사용자를 식별하고 데이터베이스에 접속하여 사용자의 비밀 값을 가져온다. 마지막으로 사용자와 동일한 방식으로 검증 값을 생성 후 사용자가 전송한 검증 값과 비교한다.
6) 검증 값이 동일하면 데이터베이스 입력을 수행하고 결과를 응답한다.

4.2 성능 및 오버헤드

CSRF 공격을 방지하기 위해 암호 기법을 사용하는 것은 웹 서버에 오버헤드가 추가될 수 있다. 본 논문의 제안 방식은 암호 기법을 사용하며, 이는 웹 서버의 성능에 영향을 미친다. 암호 기법을 사용하지 않는 방식 중 현재 사용되고 있으며 권장되는 방식은 시도-응답 방식이다[10]. 여기에는 One-time Token, CAPTCHA가 있다. 이러한 방식은 서버에서 임의로 생성한 값을 요청에 포함된 값과 비교하여 검증하는 방식으로 서버에서의 역할이 동일하다.

따라서 추가되는 오버헤드를 측정하기 위해 시도-응답 방식 중 One-time Token 방식과 본 논문에서 제안한 방식의 웹 서버 응답시간을 측정하여 성능을 비교하였다. 이렇게 함으로써 암호 기법을 사용한 방식과 사용하지 않는 방식의 웹 서버 성능 차이를 알 수 있다.

실험을 수행하기 전 데이터베이스에 100명의 사용자를 등록한 후, 자바 클라이언트에서 이 100명의 사용자 각각에 대해서 100개의 POST 요청 메시지를 생성하여 서버로 전송하고 이 요청에 대한 응답 시간을 측정 하였다.

그림 3은 제안 방식과 One-time Token을 구현하여 수행한 결과 클라이언트의 입력 요청 횟수에 따른 응답 시간을 나타낸다. 빨간색 그래프는 제안 방식을 수

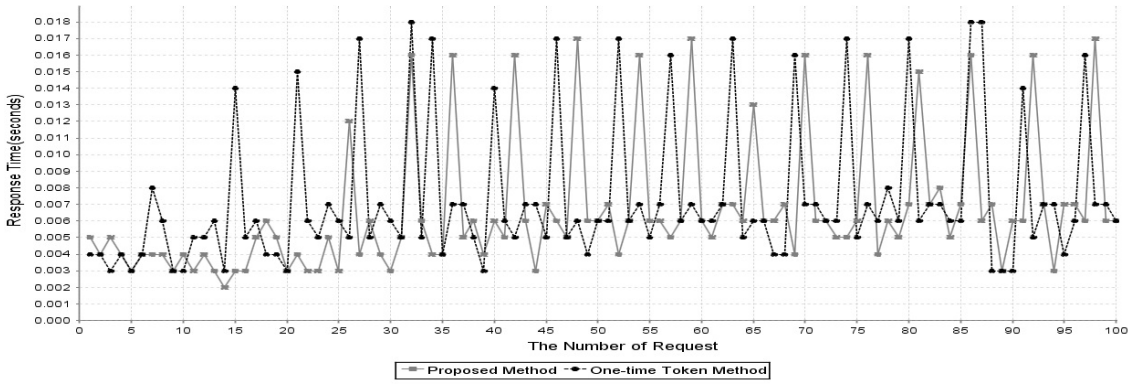

그림 3. 웹 서버 응답 시간 비교
Fig. 3. Comparison of Web Server Response Time

행한 결과이며, 파란색 그래프는 One-time Token 방식을 수행한 결과이다. y축의 단위는 초 이고, x축은 요청 횟수를 나타낸다.

제안 방식의 웹 서버 응답시간은 One-time Token을 이용한 방식과 비교해서 성능상의 차이가 크게 나타나지 않았다. 즉, 해시 함수(SHA1)의 사용과 데이터베이스 접속으로 인한 성능상의 문제는 나타나지 않는다.

4.3 보안성

일회성 토큰을 이용한 방식은 토큰 값이 스크립트에 의해 노출되면 공격자에 의해 사용될 수 있다는 취약점이 있다[11-13]. 이와 같은 상황은 사용자가 form 필드를 포함하는 페이지를 수신 후 오랫동안 웹 서버로 응답을 하지 않을 경우 발생한다. 이는 사용자가 잠시 다른 작업을 하고 있거나, 입력 데이터가 많은 경우 시간이 오래 걸리기 때문이다. 공격자가 데이터를 사전에 준비하고 있는 상황에서 토큰 값이 추출되면 아래와 같이 요청을 구성하여 공격에 이용할 수 있다. 여기서 'token' 값이 노출된 토큰 값이다.

 <img src="http://img.example.net/image/image.jpg" border="0">

제안 방식에서 세션ID와 페이지식별번호는 헤더와 메시지의 body에 포함되어 있어서 스크립트를 이용하면 쉽게 노출될 수 있다. 그러나 사용자의 비밀번호는 스크립트를 통해서 알 수 없다. 또한 비밀번호의 해시 값은 데이터를 웹 서버로 전송하는 순간 생성되기 때문에 그 이전에 공격자에 의한 공격 시도는 검증에 필요한 유효한 값을 자동으로 포함하지 않게 된다. 이 검증 값은 아래와 같이 hidden 필드에 공백으로 되어 있으며, generate함수에서 이 값을 설정한다.

<input type="hidden" name=" result"value=""> <input type="submit"value="Register"onclick="generate();">

공격자가 공격에 성공하기 위해선 검증 값과 세션 ID 그리고 페이지식별번호를 통해서 사용자 비밀번호를 추출해야하는 어려움에 직면한다. 세션ID는 세션마다 바뀌며, 페이지식별번호는 페이지 요청마다 바뀐다. 즉, 공격자가 세션ID, 페이지식별번호, 그리고 검증 값을 수집하여 분석하여도 값의 일관성이 없어 소용이 없게 된다. 또한 단방향 함수(one-way function)의 사용은 출력 값을 통해서 입력 값을 찾는 것을 불가능하게 만든다. 따라서 제안 방식의 안전성은 사용자의 비밀번호의 안전함에 의존한다.

그러나 이와 같은 비밀번호 관리와 해시 함수 호출과 같은 작업을 수행하기 위해 사용자 브라우저에 추가 확장 프로그램이 필요하다. 확장 프로그램으로는 기존의 ID 관리 또는 패스워드 매니저 프로그램이 있으며, 이 프로그램에 제안 방식을 적용할 수 있어야 한다.

V. 관련 연구

CSRF 공격을 방지하기 위한 많은 방식이 제안되었다. Encrypted Token Pattern 방식은 서버에서 임의

의 값을 암호화 하여 페이지의 body에 포함한 후 사용자에게 전송한다[1]. 사용자는 암호화된 값을 추출 후 header에 포함시키고 서버로 응답한다. 그러나 이와 같은 방식은 페이지 body에서 값을 추출 후 header에 포함시켜 응답하기 때문에 값의 암호화가 소용이 없다. 즉, 검증 값을 생성하기 위한 절차가 클라이언트 측에서 없기 때문에 공격자에 의해 값이 추출되면 이 값은 검증에 사용될 수 있다.

Secret cookie를 이용한 방법은 Cookie에 포함되는 정보를 암호화하는 방식이다[7]. 그러나 공격자에 의한 위조된 요청 메시지는 자동으로 헤더에 이 비밀 값을 포함하고 웹 서버로 전송되기 때문에 공격자는 이 값을 알 필요가 없다. 이 방식은 웹 서버에서 정당한 사용자의 요청과 공격자에 의한 요청을 구별할 수 없다.

웹 사이트 방문 흔적을 이용한 방식은 사용자의 프라이버시가 노출된다는 단점이 있으며, 스크립트를 이용한 조작의 가능성이 있다[8].

CAPTCHA를 이용한 방식은 컴퓨터는 식별할 수 없는 사용자만이 식별 가능한 문자를 이용한 방식이다[14]. 따라서 스크립트를 이용해 값을 추출할 수 없다. 이 문자는 매번 바뀌며 사용자는 데이터 전송 시 이 문자와 함께 웹 서버로 전송한다. 웹 서버는 수신한 문자의 유효성을 판별한 후 요청의 정당성을 판단한다. 이와 같은 방식은 비록 안전하지만 사용자의 불편함을 초래한다.

One-time Token 방식은 임의로 생성한 토큰을 HTML body에 포함시켜 사용자에게 전송한다[15]. 사용자는 웹 서버로 데이터 전송 시 이 토큰 값을 body에 포함 시킨다. 이 토큰은 웹 서버에서 검증을 위해서 사용되며 최초 한번 사용 후 폐기한다. 검증에 필요한 값이 body에 포함되어 있기 때문에 공격자의 악의적인 요청 메시지는 이 값을 자동으로 포함 시키지 않는다. 그러나 스크립트를 이용해 값을 추출한 후 공격에 이용할 수 있다[11-13]. 만약 사용자에 의해 토큰이 사용되기 전 공격자에 의해 토큰이 추출되고 공격에 사용되면 웹 서버는 정상적인 요청으로 판단하게 된다. 공격자에 의해 토큰이 사용되면 사용자의 정당한 요청은 웹 서버에서 거부된다.

Ⅵ. 결론 및 향후 연구

웹 서버는 요청을 받을 시 Cookie 정보를 통해서 해당 요청의 유효성을 판단하게 된다. 정당한 요청과 악의적인 요청은 이 Cookie 값이 동일하기 때문에 웹 서버에서는 이를 구분할 수 없다.

본 논문에서는 CSRF 공격에 대한 대응방안으로 세션ID와 페이지식별번호 그리고 사용자 비밀번호의 해시 값을 이용하여 요청 순간 검증에 필요한 값을 생성 하도록 하였다.

검증에 필요한 값은 사용자와 웹 서버만이 알고 있는 비밀 값을 활용하여 생성되기 때문에 서버는 이를 구현하기 위해 많은 변경이 필요하지 않다. 그리고 간단한 해시 함수(SHA1)의 사용은 웹 서버의 성능에 영향을 미치지 않는다.

제안 방식에서 비밀번호 관리와 검증 값 생성을 안전하게 수행하기 위해서 패스워드 매니저와 같은 전용 프로그램이 필요하다. 또한 이러한 프로그램이 다양한 웹 서버와 호환이 가능하도록 할 필요가 있다. 호환이 이루어지지 않는다면 이와 같은 프로그램은 클라이언트 측에서 많이 필요하게 되며, 이는 클라이언트에 부담이 될 수 있다. 웹 서버와의 호환 뿐 아니라 다양한 브라우저와 호환이 가능한 프로그램의 개발이 필요하다.

References

[1] OWASP, *The Ten Most Critical Web Application Security Risks*(2013), Retrieved Dec., 30, 2013, from https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[2] J. H. Bang and R. Ha, “Evaluation methodology of diagnostic tool for security weakness of eGOV software,” *J. KICS*, vol. 38, no. 4, pp. 335-343, Apr. 2013.

[3] J. H. Bang and R. Ha, “Validation test codes development of static analysis tool for secure software,” *J. KICS*, vol. 38, no. 5, pp. 420-427, May 2013.

[4] S. H. Lee, Y. J. Maeng, D. H. Nyang, and K. H. Lee, “Possibility of disclosure of user information in internet explorer,” *J. KICS*, vol. 38, no. 12, pp. 937-943, Dec. 2013.

[5] P. De Ryck, L. Desment, T. Heyman, F. Piessens, and W. Joosen, “CsFire: Transparent client-side mitigation of malicious cross-domain requests,” in *Eng. Secure Software and Syst.*, vol. 5965, pp. 18-34, Berlin Heidelberg, Germany, Feb. 2010.

[6] X. Lin, P. Zavarsky, R. Ruhl, and D. Lindskog, “Threat modeling for CSRF

attacks," in *Int. Conf. Computational Sci. and Eng.*, vol. 3, pp. 486-491, Aug. 2009.

[7] Z. Mao, N. Li, and I. Molloy, "Defeating cross-site request forgery attacks with browser-enforced authenticity protection," in *Financial Cryptography and Data Security*, vol. 5628, pp. 238-255, Berlin Heidelberg, Germany, Feb. 2009.

[8] A. Barth, C. Jackson, and J. C. Mitchell, "Robust defenses for cross-site request forgery," in *Proc. ACM Conf. Comput. Commun. Security*, pp. 75-88, New York, USA, Oct. 2008.

[9] S. Khandelwal, P. Shah, M. K. Bhavsar, and D. S. Gandhi, "Frontline techniques to prevent web application vulnerability," *Int. J. Advanced Research in Comput. Sci. Electron. Eng.*, vol. 2, no. 2, p. 208, Feb. 2013.

[10] J. H. Park, I. Y. Jung, and S. J. Kim, "CSRF defense using page identifier and sessionID," *UCWIT(2013)*, Daegu, Korea, Dec. 2013.

[11] A. Czeskis, A. Moshchuk, T. Kohno, and H. J. Wang, "Lightweight server support for browser-based CSRF protection," in *Proc. Int. Conf. World Wide Web*, pp. 273-284, Geneva, Switzerland, May 2013.

[12] E. Y. Chen, S.Gorbaty, A. Singhal, and C. Jackson, "Self-exfiltration: The dangers of browser-enforced information flow control," in *Proc. Workshop of Web 2.0 Security and Pricacy 2012*, vol. 2, San Francisco, USA, May 2012.

[13] M. Heiderich, M. Niemietz, F. Schuster, T. Holz, and J. Schwenk, "Scriptless attacks - stealing the pie without touching the sill," in *Proc. ACM Conf. Comput. Commun. Security*, pp. 760-771, New York, USA, Oct. 2012.

[14] J. Blatz, *CSRF: Attack and Defense*(2013), Retrieved Dec. 30, 2013, from http://www.foundstone.com.au/uk/resources/white-papers/wp-csrf-attack-defense.pdf.

[15] Y. C. Sung, M. C. Y. Cho, C. W. Wang, C. W. Hsu, and S. W. Shieh, "Light-weight CSRF protection by labeling user-created contents," *Int. Conf. Software Security and Reliability*, pp. 60-69, Gaithersburg, USA, Jun. 2013.

박 진 현 (Jin-hyeon Park)

2009년 2월 : 안동대학교 컴퓨터공학과 졸업
2013년 2월 : 경북대학교 전자공학과 석사
2013년 3월~현재 : 경북대학교 전자공학과 박사과정

<관심분야> System and Network Security

정 임 영 (Im Y. Jung)

1993년 2월 : 포항공과 대학교 화학과 졸업
1999년 2월 : 서울대학교 전산과학과 졸업
2001년 2월 : 서울대학교 컴퓨터공학부 석사
2010년 8월 : 서울대학교 컴퓨터공학부 박사

<관심분야> 데이터 및 시스템 보안, 스토리지 시스템, 분산컴퓨팅시스템, 클라우드 컴퓨팅,

김 순 자 (Sun-ja Kim)

1975년 2월 : 경북대학교 수학교육과 졸업
1977년 2월 : 경북대학교 수학교육과 석사
1988년 2월 : 계명대학교 이학박사

<관심분야> 정보보호 응용기술, 전자상거래 보안