전자서명 토큰을 이용한 웹 애플리케이션에서의 CSRF공격 방어 기법

이대섭, 김효종, 전문석
숭실대학교 일반대학원 컴퓨터학과
e-mail : daesupi@nate.com, weeps@hanmail.net, mjun@ssu.ac.kr

A Method for Preventing CSRF Attacks in Web Application using Digital Signature Token

Dae-Seop Lee, Hyo-Jong Kim, Moon-Seog Jun
Dept of Computer Science, Soong-Sil University

요 약

웹 애플리케이션 해킹 방법인 CSRF(Cross Site Request Forgery) 공격은 2008년 2월에 온라인 경매 사이트인 옥션에서의 1800만명의 개인정보를 유출 사고 피해를 입힌 공격이다. OWASP(Open Web Application Security Project)에서는 이 공격의 해결방안으로 동기화되고 고유한 토큰 값을 생성하여 페이지 요청 시에 이를 검증하는 시스템을 권고하고 있다. 따라서 본 논문에서는 이 공격을 방어하기 위한 방법으로 타임스탬프와 사용자 고유의 값인 전자서명을 토큰형태로 생성하여 Hidden Field에 삽입함으로써 검증하는 기법에 대해 연구하고자 한다.

1. 서론

웹 애플리케이션과 웹 기반 정보 시스템의 이용이 증가하면서 웹은 우리의 생활에서 필수적인 부분이 되었다. 하지만 이에 대한 보안 위협은 현재에도 계속 일어나는 문제로 뉴스에서도 그 사례를 찾아볼 수 있다. 국내에서는 2008년 2월에 발생한 온라인 경매 사이트인 옥션(www.auction.co.kr)에서의 1800만명의 개인 정보 유출 사고가 크로스 사이트 요청 변조(Cross Site Request Forgery, CSRF) 공격의 피해로 보고되었다. 지금까지 이 공격은 OWASP(Open Web Application Security Project)의 10대 가장 심각한 웹 애플리케이션 보안 취약점으로 상위에 있는 크로스 사이트 스크립팅(Cross Site Scripting, XSS)과 인젝션(Injection) 취약점에 비해서 많은 주목을 받지 못했다. 하지만 주로 사용자에 대한 공격으로 금전거래나 개인정보 유출 등에 활용되기 때문에 CSRF 공격으로 인한 피해는 심각할 수 있다. OWASP는 이 공격에 대한 해결방안으로 동기화되고 사용자의 고유한 토큰을 이용하는 방법을 권고하고 있다.

따라서 본 논문에서는 안전한 토큰을 생성하기 위한 방법으로 타임스탬프 값과 사용자의 고유한 전자서명 값을 이용하여 안전한 토큰을 생성하고 이를 Hidden Field에 삽입하여 사용자가 의도한 페이지를 검증하고 관리할 수 있는 방안에 대해 연구하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 CSRF와 실제 취약점 분석해보고 3장에서는 전자서명 토큰을 이용한 CSRF 공격 방어 기법을 제안하며 4장에서는 결론을 기술한다.

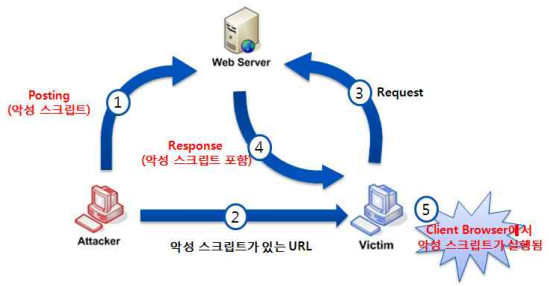
2. 관련연구

2.1 CSRF(Cross Site Request Forgery)

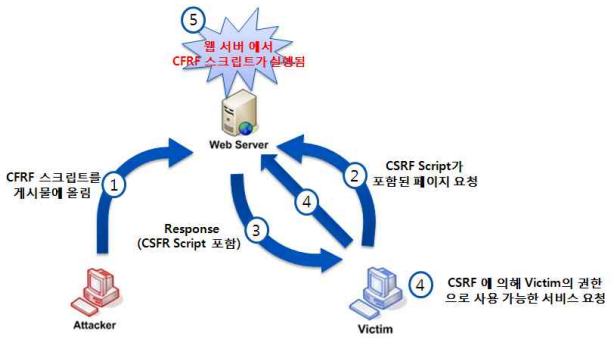
CSRF(Cross Site Request Forgery, 이하 CSRF)는 웹 애플리케이션이 정상적인 절차에 의해 발생된 요청과 그렇지 못한 요청을 구분하지 못하는 경우 발생한다. XSS(Cross Site Scripting, 이하 XSS)와 비슷한 이 공격은 적법한 사용자(희생자)로 하여금 조작된 정상 Request를 웹 애플리케이션에 전송하도

록 유도함으로써 희생자는 자신이 원하지 않았던 정상 Request를 서버로 보내고 서버에서는 희생자의 권한으로 Request에 대한 처리를 하면서 공격이 이루어진다.

XSS와 CSRF 간의 가장 큰 차이점은 XSS는 공격자가 작성한 악성 스크립트가 클라이언트 브라우저에서 실행되는 반면에 CSRF는 적법한 사용자가 정상정인 Request를 보내 서버에서 실행된다는 점이다. 다음은 XSS 공격과 CSRF 공격절차의 그림이다.



[그림 1] XSS 공격절차



[그림 2] CSRF 공격절차

2.2 취약점 분석

2.2.1 OWASP 예제

Alice가 Bob에게 계좌이체 하는 경우 다음과 같은 GET 메소드 방식의 URL 매개변수를 이용하여 웹 서버에게 전달한다.

```
GET    http://bank.com/transfer.do?acct=BOB&amount=100  HTTP/1.1
```

공격자인 Maria는 Alice가 계좌이체 할 때 위와 같은 Request로 만들어지는 것을 착안하여 다음과 같은 코드를 E-mail에 html 코드로 숨겨서 Alice에게 전송할 수 있다.

Alice가 아래 링크를 클릭하는 순간 Alice는 자신이 의도하지 않은 '계좌이체 Request'가 만들어져 은행 서버에게 보내면 공격자인 Maria에게 100000 달러의 현금을 계좌이체 시킨다. 결과적으로 계좌이체 Request를 받은 은행 서버는 이 Request가 Alice가 정당하게 보낸 것인지 아니면 공격자인 Maria에 의해 보낸 것인지 판단하지 못하기 때문에 계좌이체 Request에 대한 처리를 한다.

```
<a href="http://bank.com/transfer.do?acct=MARIA&amount=100000">
View my Pictures!</a>
```

희생자인 Alice가 공격에 대해 알아차리지 못하게 하기 위해서 다음과 같은 코드로 대체할 수 있다.

```
<img src="http://bank.com/transfer.do?acct=MARIA&amount=100000"
width="1" height="1" border="0">
```

이 이미지 태그로 인해 Alice는 Maria에게 계좌이체 하는 Request를 볼 수 없지만, Alice의 브라우저는 계좌이체 Reqeust를 은행서버에게 보내게 된다.

2.2.1 D 포털사이트 까페의 권한상승 예제

실제 D 포털사이트의 까페에서 CSRF를 이용한 권한상승 취약점을 분석하였다. 공격절차는 다음과 같다.

1) 공격자가 포털사이트 까페에 가입
2) 공격자는 관리자의 권한으로 권한을 상승시키는 Request를 분석
3) 분석된 Request를 스크립트로 게시판에 게제
4) 관리자는 해당 게시글을 읽음과 동시에 공격자의 권한을 상승시키는 CSRF 스크립트를 자신도 모르게 페이지를 요청
5) 공격자의 까페 등급이 상승

```
<img src="http://cafe363.daum.net/_c21_/updateLvl_action?grpid=1Kpl7&emoticon=&mgrpid=&fldid=&dataid=-
MI45kwI4lQ0&page=&prev_page=&firstbbsdepth=&lastbbsdepth=&content=&contentval=&espam1=&viewcount1=&re
gdt1=&mode=7&num=&pardataname=&parbbsdepth=&e_paruserid=&pardatatype=&parregdt=&move=&userlist=-
MI45kwI4lQ0&issueregyn=&changerolecode=25&callCount=1&page=%2F_c21_%2Ffounder_member_admin_v2%3Fgrpid
%3D1Kpl7&httpSessionId%3D1=&c0-scriptName=WIChgMemberInfoService&c0-methodName=changeMemberLevel&c0-
id=0&c0-param0=string%3A1Kpl7&c0-param1=number%3A1&c0-param2=string%3A25&c0-e1=string%3A-
MI45kwI4lQ0&c0-param3=Array%3A%5Breference%3Ac0-
e1%5D&batchId=3&cmttype=member&comment_view=%B5%EE%B7%CF%B5%C8+%B4%F1%B1%DB%C0%CC+%BE%F8%
BD%C0%B4%CF%B4%D9.+%B4%F1%B1%DB%C0%BB+%B5%EE%B7%CF%C7%D8+%C1%D6%BC%BC%BF%E4.+&comment
_view_open=%C8%B8%BF%F8%C0%BA%B4%F1%B1%DB%C0%BB+%BE%B5%BC%F6+%BE%F8%BD%C0%B4%CF%B4%D9.
&emoticon=&nickname=&grpcode=&act=&seq=&return_url=&parseq=&comment= HTTP/1.1" width=0 height=0>
동업해주세요
```

[그림 3] D 포털사이트 까페의 권한상승 패킷분석

위 그림과 같이 실제 분석결과를 통해서 관리자의 요청패킷은 여러 가지의 매개변수로 이루어져 있다는 사실을 확인할 수 있었다. 하지만 많은 매개변수 중에서 특정 매개변수의 용도는 까페의 권한을 변경하는 것으로서 매개변수의 이름을 통해 알 수 있었다. 공격자는 위와 같은 분석 자료를 통해서 다음과 같은 D 포털사이트의 권한에 따른 관련 매겨변수의 값을 알아낼 수 있다.

[표 1] D 포털사이트 까페의 권한에 따른 매개변수의 값

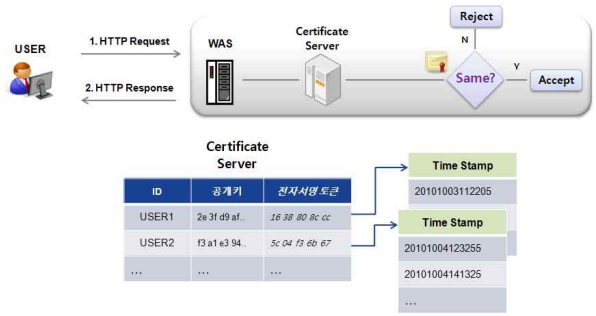
회원 등급	관련 매개변수
준회원	changerolecode=23
정회원	changerolecode=25
우수회원	changerolecode=28
특별회원	changerolecode=30

이와 같이 CSRF의 취약점은 희생자로 하여금 자신도 모르게 의도하지 않은 요청패킷을 서버로 전송하도록 유도하는 것이다.

따라서 본 논문에서는 안전한 토큰을 생성하기 위한 방법으로 타임스탬프 값과 사용자의 고유한 전자서명 값을 이용하여 안전한 토큰을 생성하고 이를 Hidden Field에 삽입하여 사용자가 의도한 페이지를 검증하고 관리할 수 있는 방법을 제시한다.

3. 제안하는 기법

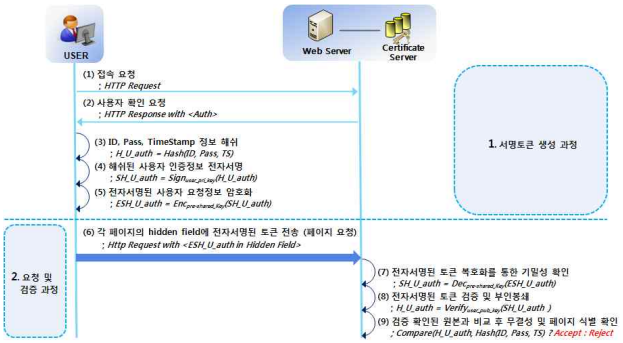
3.1 전체 구성도



[그림 4] 전자서명 토큰을 이용한 전체 검증과정 흐름도

사용자는 계정 생성 시 발급한 토큰을 HTTP Request의 Hidden Field에 삽입하여 웹 애플리케이션에게 패킷을 요청한다. 웹 애플리케이션 서버는 검증서버에게 패킷을 전송하고 검증서버는 자신의 원본 테이블의 값과 비교과정을 통해 패킷의 수용여부를 판단한다. 검증서버의 테이블에는 위와 같이 ID, 공개키, 전사서명 토큰 등의 정보가 존재하고 각 ID에 대한 유동적인 타임스탬프 값을 기록한다.

3.2 프로토콜 흐름도



[그림 4] 전자서명 토큰을 이용한 프로토콜 흐름도

페이지 검증 및 식별 프로토콜은 크게 두 가지로 나눌 수 있다. 첫 번째는 전자서명 토큰을 생성하는 과정이고 두 번째는 그 토큰을 이용하여 통신하는 과정에서 Certificate Server의 검증을 받는 과정이다.

서명토큰을 생성하는 과정은 다시 5가지로 분류되며, 다음과 같은 단계로 진행된다.

과정 1 : 사용자가 접속요청을 하는 단계이다. 사용자는 HTTP Request를 통해 웹 서버에게 접속을 요청한다.

과정 2 : 접속 요청을 받은 웹 서버는 HTTP Response를 통해 사용자의 신원을 요청한다.

과정 3 : 사용자 확인 요청을 받은 사용자는 ID, 패스워드, 타임스탬프 정보를 연접해서 해쉬함수를 실행하여 메시지 다이제스트를 생성한다. 이 과정은 아래 (식 3-1)과 같다.

$$H_U_auth = Hash(ID,\ Pass,\ TS) \qquad (식\ 3\text{-}1)$$

과정 4 : 과정 3에서 생성한 메시지 다이제스트(H_U_auth)를 사용자의 개인키(user private key)로 서영한다. 이 과정은 아래 (식 3-2)와 같다.

$$SH_U_auth = Sign_{user_pri_key}(H_U_auth) \qquad (식\ 3\text{-}2)$$

과정 5 : 계정 등록 시에 생성한 사전 공유키(pre-shared key)로 서명된 데이터(SH_U_auth)를

암호화한다. 이 과정은 아래 (식 3-3)과 같다.

$ESH_U_auth = Enc_{pre\text{-}shared_Key}(SH_U_auth)$ *(식 3-3)*

첫 번째 과정인 서명토큰 생성과정을 거치면 사용자는 타임스탬프 값과 자신의 개인키로 서명한 서명토큰이 생성된다. 사용자는 이 서명토큰을 자신의 Hidden Field에 삽입하고 모든 요청 패킷은 이 토큰을 포함하여 웹 서버에게 전달된다.

요청 및 검증과정은 4가지로 분류되며, 다음과 같은 단계로 진행된다.

과정 6 : 최종적으로 구성된 서명토큰을 HTTP Request의 Hidden Field에 삽입하여 웹 서버에게 전달한다.

과정 7 : 웹 서버는 사용자의 전자서명된 토큰(ESH_U_auth)이 포함된 패킷을 수신하고 복호화하여 기밀성이 유지되었는지 확인한다. 이 과정은 아래 (식 3-4)와 같다.

$SH_U_auth = Dec_{pre\text{-}shared_Key}(ESH_U_auth)$ *(식 3-4)*

과정 8 : 복호화된 정보를 사용자의 공개키로 검증한다. 이는 신원을 확인하고 제 3자로부터의 악의적인 공격을 방지하며 사용자의 부인을 봉쇄한다. 이 과정은 아래 (식 3-5)와 같다.

$H_U_auth = Verify_{user_pub_key}(SH_U_auth)$ *(식 3-5)*

과정 9 : 검증 후 추출된 원본 데이터(ID, Pass, TS)를 요청 시 실행했던 해쉬 함수로 메시지 다이제스트를 생성하고, 수신된 다이제스트와 비교하여 무결성을 확인한다. 만약 비교결과가 일치하지 않으면 제 3자에 의한 개입 및 조작으로 판단하여 해당 정보를 수용하지 않는다. 이 과정은 아래 (식 3-6)와 같다.

$Compare(H_U_auth, Hash(ID, Pass, TS)\ ?\ Accept : Reject$ *(식 3-6)*

4. 결론 및 향후 계획

웹 애플리케이션 보안은 현재에도 계속 나타나는 문제로 전자상거래 보안, 개인정보 유출 등을 통해 최근 이슈화되고 있다. OWASP에서는 웹 해킹을 조사하고 분석하여 상위 10개의 웹 해킹 종류를 선정하고 발표한다. 그 중에서 XSS와 CSRF 공격은 항상 상위에 랭크되어있는 공격으로 웹 해킹 공격의 많은 부분을 차지할 뿐만 아니라 그 피해도가 심각하다. 하지만 여전히 완벽한 해결방안 및 완화기법은 제시되지 않은 상황이다. 기존에 CSRF 공격 완화기법은 POST 방식, HTTP Referrer 헤더 감사 방식, 세션 ID 파라메터 확장으로 나눌 수 있다. POST 방식은 단지 GET 메소드를 POST로 바꾸어 img scr 공격정도를 완화할 수 있다. 또한 HTTP Referrer 헤더 검사 방식은 공격자가 임의로 그 값을 변경할 수 있기 때문에 안전하지 않다. 마지막으로 세션 ID 파라메터 확장 방식은 기존의 쿠키방식의 취약점을 그대로 내포하고 있다. 따라서 본 논문의 제안기법은 동기화되고 고유한 전자 서명된 토큰을 사용함으로써 기존방식의 무결성, 부인봉쇄 등의 문제를 보완할 수 있다.

향후에는 이 시스템을 웹 서버 구축을 통해 구현하여 웹 환경에서 해쉬, 전자서명을 효율적으로 사용할 수 있는 방안에 대해 연구할 예정이다.

참고문헌

[1] 선수림, AOP를 이용한 웹 애플리케이션에서의 XSRF 공격 방어 기법, 2009

[2] OWASP, 'TOP Ten most Critical Web Application Security Vulnerabilities", http://www.owasp.org, 2010

[3] Cross-site ruqeust forgerty, http://en.wikipedia.org/wiki/Cross_site_request_forgery

[4] C.Shiflett, "Foiling Cross-Site Attacks", http://shiflett.org/articles/foiling-cross-site-attacks, 2003