KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.1 pp.31~36 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.1.31

A Defense Mechanism Based on Session Status against Cookie Replay Attack in Web Applications

Jong Sun Won† · JiSu Park†† · Jin Gon Shon†††

ABSTRACT

As web accessibility has been easier, security issue becomes much more important in web applications demanding user authentication. Cookie is used to reduce the load of the server from the session in web applications and manage the user information efficiently. However, the cookie containing user information can be sniffed by an attacker. With this sniffed cookie, the attacker can retain the web application session of the lawful user as if the attacker is the lawful user. This kind of attack are called cookie replay attack and it causes serious security problems in web applications. In this paper, we have introduced a mechanism to detect cookie replay attacks and defend them, and verified effectiveness of the mechanism.

Keywords : Web Applications, Session, Cookie, Cookie Replay Attack, Security

웹 애플리케이션에서 세션 상태 기반의 쿠키 재전송 공격 방어 기법

원 종 선† · 박 지 수†† · 손 진 곤†††

요 약

웹 접근성이 보다 용이해짐에 따라 사용자 인증이 필요한 웹 애플리케이션에서 보안이 중요시 되고 있다. 웹 애플리케이션에서 쿠키는 세션으로 인한 서버의 부하를 줄이고, 사용자 정보를 효율적으로 관리하기 위해 사용한다. 그러나 사용자 정보가 저장된 쿠키는 공격자에 의해 스니핑될 수 있으며, 이렇게 스니핑된 쿠키를 이용하여 공격자는 마치 합법적인 사용자인 것처럼 사용자의 세션을 유지할 수 있다. 이러한 종류의 공격을 쿠키 재전송 공격이라 하는데, 이것은 웹 애플리케이션에서 중대한 보안 문제를 야기한다. 본 논문에서는 이러한 쿠키 재전송 공격을 탐지하고 방어할 수 있는 기법을 제안하였고 그 효과성을 검증하였다.

키워드 : 웹 애플리케이션, 세션, 쿠키, 쿠키 재전송 공격, 보안

1. 서 론

국제 웹 보안 표준기구는 2013년에 웹 애플리케이션(Web Application)의 취약점으로 가장 위험한 Top10을 발표하였다. Top10은 위험 순위별로 구분되었으며, 그 중 2위는 웹 애플리케이션의 인증과 세션관리의 취약점이다[12].

본 논문에서 웹 애플리케이션의 쿠키 재전송 공격은 취약한 인증과 세션 관리의 문제를 다룬다. 웹 애플리케이션을 이용하는 사용자들은 쿠키를 사용하여 로그인을 한다.

쿠키는 사용자 컴퓨터에 이전에 방문했던 기록을 저장하고, 사용자가 웹 애플리케이션에 다시 접속을 하였을 경우 저장된 쿠키의 정보를 로딩하여 로그인함으로써 웹 서비스를 제공받는다[1]. 쿠키는 사용자의 컴퓨터에 저장되어 관리되고 있어서 스니핑(sniffing) 즉, 도청의 위험에 노출되어 있다. 그러므로 중요한 개인정보는 쿠키에 저장되어서는 안 되며 보안 소켓을 이용하는 세션으로 관리되어야 한다. 그러나 쿠키의 많은 정보들이 모두 세션에서 관리된다면 웹 서버는 오버헤드가 발생한다. 이는 사용자의 세션이 발급되는 개수만큼 웹 서버에 부하를 주기 때문이다. 그러므로 쿠키와 세션의 적절한 상호보완 관계가 필요하며 연동을 통한 안전하고, 부하를 줄일 수 있는 정보 관리가 필요하다.

본 논문에서는 세션 상태를 이용하여 웹 애플리케이션의 강인한 인증과 세션관리를 위한 쿠키 재전송 공격의 방어 기법을 제안하고, 제안하는 방어 기법의 효과성을 검증한다.

† 준 회 원 : 한국방송통신대학교 정보과학과 석사과정
†† 정 회 원 : 고려대학교 정보창의교육연구소 연구교수
††† 종신회원 : 한국방송통신대학교 컴퓨터과학과 교수
Manuscript Received : September 03, 2014
First Revision : October 13, 2014; Second Revision : November 25, 2014
Accepted : November 27, 2014
* Corresponding Author : Jin Gon Shon(jgshon@knou.ac.kr)

2. 재전송 공격 방어 연구

2.1 세션 토큰

세션 토큰(session tokens)의 암호화 알고리즘인 SHA는 1993년에 미국 NIST에 의해 개발되어 가장 많이 사용되는 해쉬 함수이다. 해시 함수는 해시 결과 값으로 입력 값을 계산하는 것은 어려운 특징을 가지므로 정보보안 강도가 높고 안전하다[11]. 세션 토큰은 송신자가 일회성 암호를 해시 함수로 계산하여 송신하면, 수신자가 같은 계산을 수행하여 두 값이 일치하는 경우에 로그인이 성공된다[5, 6, 7, 10].

2.2 순서번호

각 메시지에 대한 순서번호(sequence number)를 부여하여 번호가 순차적일 경우에만 수신한다. 메시지 송신을 위해 회선교환과 같이 고정적(물리적) 회선을 사용하는 경우는 연속되는 전송 데이터의 도달 순서가 바뀌는 일은 없다. 그러나 패킷교환의 데이터그램 서비스에서는 패킷의 전송 루트가 일정하지 않으므로 패킷에 순서번호를 부여하여 번호순으로 패킷 정보를 순서화 한다[9].

2.3 시도-응답

웹 애플리케이션에서 인증시 사용자 각각은 신뢰된 메시지를 기대한다. 사용자가 웹 서버로 인증을 시도하여 수신된 메시지에 정확한 값이 포함할 것을 요구한다[2]. 시도-응답(challenge-response) 방식은 네트워크 사용자 인증을 위해 웹 서버에서 사용된다. 사용자 인증을 위해 서버에서 사용자에게 비밀번호 요구를 시도하고 사용자로부터 응답을 받아 비밀번호가 정확하면 인증된다. 이 방식에서는 기본 인증과 다이제스트 인증의 두 가지를 사용한다. 기본 인증은 비밀번호를 평문으로 사용하기 때문에 도용 문제가 있다. 이러한 단점을 보완한 다이제스트 인증 방식은 메시지를 해시 함수로 반복 적용하여 축약된 일정한 길이의 비트열로 만들어 표현함으로써 비밀번호가 네트워크에 평문으로 전송되지 않게 한 것이다[8].

2.4 타임 스탬프

수신자가 인증된 메시지를 수신하여 송신자가 보낸 시간이 적절한 타임스탬프의 허용범위 내에 있을 경우만 정상적인 인증으로 판단하고 메시지를 수신한다[2, 3, 4, 10]. 이 방법을 적용하기 위해서는 송수신자들이 동기화된 클럭를 가져야 한다. 그러나 다양한 지역에서 접속한 사용자들의 로컬 시간이 동기화되어야 하기 때문에 적용하기 어렵다는 단점이 있다.

2.5 일회용 패스워드

OTP(one-time passwords) 방식은 사용된 패스워드를 폐기하여 재사용 할 수 없도록 하는 방식으로 온라인 뱅킹 시스템에서 널리 활용되고 있는 방식이다[10]. 그러나 사용자가 OTP 기기를 구입하여야 하는 비용적인 문제가 있다.

2.6 기존의 쿠키 재전송 공격 방어기법과 제안기법의 비교

Microsoft Internet Explorer 버전 6 서비스 팩 1 이상에서는 사이트 간 스크립팅을 통해 발생할 수 있는 쿠키 도난 문제를 줄이는 데 도움이 되는 HttpOnly 쿠키 속성을 사용할 수 있다. 도난된 쿠키는 사이트에서 사용자를 식별하는 중요한 정보를 포함할 수 있으며 공격자가 해당 사용자를 가장하거나 중요한 정보를 알아내는 데 악용될 수 있다. 호환되는 브라우저에서 HttpOnly 쿠키를 받으면 클라이언트 측 스크립트에서 해당 쿠키에 액세스할 수 없다[13].

그러나 기존 쿠키 스니핑을 봉쇄하는 기법인 HttpOnly 쿠키는 호환되는 브라우저에서만 사용되므로, 모든 웹 브라우저를 완벽히 원천봉쇄하는 것은 불가능하다. 따라서 쿠키가 스니핑되었을 경우를 대비한 방어 기법이 필요하다.

3. 쿠키 재전송 공격 방어 기법

3.1 쿠키 재전송 공격 방법

웹 애플리케이션에서 쿠키 정보를 탈취하는 방법으로 다양한 기법을 사용한다. 그 중 한 가지로 웹 애플리케이션의 주소 경로에 자바스크립트 파라미터(parameter)를 입력하여 탈취하는 방법이 있다. 자바스크립트 언어에서 사용하는 스크립트 문자들과 특수기호를 혼합하여 주소경로에 입력 후 실행하여 쿠키정보를 탈취한다.

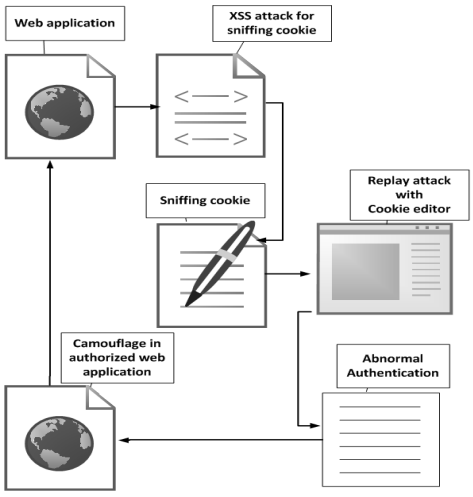


Fig. 1. Replay attack with cookie editor

또한 XSS(Cross Site Scripting) 공격을 실행하여 쿠키를 탈취하는 방법과 공격자가 사용자들 간에 송수신되는 패킷을 스니핑하여 탈취하는 방법 등이 있다. 이렇게 탈취한 쿠키를 편집하여 재전송 공격을 실행하여 개인정보를 조회하거나 악용할 수 있다. 쿠키 편집기인 쿡시 툴바를 이용하면 탈취한 다른 사용자의 쿠키를 자신의 쿠키 파일로 편집하여 웹 서버로 전송함으로써 쿠키 재전송 공격이 가능하다. 즉, 쿡시 툴바는 웹 애플리케이션의 사용자 계정을 편집(위장)하거나 쿠키 재전송 공격의 도구로 악용된다.

그림 1은 다른 사용자의 쿠키를 XSS 공격으로 탈취하고, 쿡시 툴바로 편집하여 쿠키 재전송 공격을 실행하는 과정을 나타낸 것이다. 공격자는 스니핑된 쿠키를 편집하고 재전송하여 비정상적으로 로그인 인증허가를 받는다. 그리고 쿠키 재전송 공격이 성공한 이후에 공격자는 합법적인 사용자 권한으로 정상적인 이용이 가능하다.

3.2 세션 상태 확인 알고리즘

웹 애플리케이션에서는 세션 토큰을 발급함으로써 합법적인 사용자 인증을 한다. 즉, 사용자가 웹 애플리케이션에 로그인을 할 때마다 세션 토큰은 암호화되어 발급된다. 그러나 세션 하이재킹(Session Hijacking) 즉, 정당한 사용자가 인증을 수행한 후 공격자에 의해 세션을 가로채는 보안공격에 취약할 수 있다[9]. 따라서 인증을 세션 토큰만 가지고 수행한다면 다른 사용자의 세션 토큰을 스니핑하여 재전송 공격을 할 수 있다. 따라서 본 논문에서는 세션 상태 정보를 데이터베이스에 저장하여 재전송 공격을 방어한다.

그림 2는 세션 유지 시간과 로그인 시간의 변수를 나타낸 것이다. 사용자가 정상적(normal)으로 로그아웃하는 경우의 세션 종료시간은 $\Delta Tend1$이다. 이 경우에 식(1)에서와 같이 세션 종료시간은 로그아웃한 시간과 같으며 세션은 로그아웃하기 전까지 유지된다.

$$\Delta Tend1 = \Delta Tlogout \quad (1)$$

$\Delta Tend1$: 세션의 종료시간 (정상 종료)

$\Delta Tlogout$: 로그아웃 시간

그림 2에서 $\Delta Tlogout$은 세션의 유지시간 동안에 사용자가 정상적으로 로그아웃을 한 경우의 세션 종료 시간이다. 다음 식(2)는 사용자가 웹 브라우저의 닫기 버튼을 클릭하거나 사용자의 컴퓨터가 강제 종료되는 경우와 같이 비정상적(abnormal)으로 로그아웃되었을 때의 세션 종료시간($\Delta Tend2$)을 식으로 나타낸 것이다.

$$\Delta Tend2 = \Delta Twas + \Delta TlastAccess \quad (2)$$

$\Delta Tend2$: 세션의 종료시간 (비정상 종료)

$\Delta Twas$: 웹 애플리케이션 서버의 세션 설정 시간

$\Delta TlastAccess$: 마지막 웹 애플리케이션 조회 시간

$\Delta TlastAccess$는 사용자가 마지막으로 웹 애플리케이션을 조회한 시간이다. 비정상 종료인 경우 사용자가 웹 애플리케이션에 다시 접속했을 때 로그아웃된 상태이나 세션은 웹 서버에서 세션 유지시간 동안 유지된다. 따라서 사용자가 마지막으로 웹 애플리케이션을 조회한 시간인 $\Delta TlastAccess$가 강제로 로그아웃한 경우의 세션 시작시간이 된다. 그리고 세션 유지시간은 사용자가 마지막으로 웹 애플리케이션을 조회한 시간인 $\Delta TlastAccess$에서 $\Delta Twas$를 더한 $\Delta Tend2$ 시간까지 연장된다.

표 1은 세션 상태 확인 알고리즘이다. 사용자가 웹 애플리케이션을 사용하는 동안 지속적으로 세션 상태를 확인한다. 세션 상태는 세션이 유지되는 active 상태와 세션이 종료된 destroy 상태로 구분된다.

Table 1. Algorithm of session status verification

```
1  //ΔTend is equal to ΔTend1 or ΔTend2
2  sessionStatus(Timestamp T) {
3    If(ΔTstart <= T and T <= ΔTend) {
4       Status = active;
5    } Else {
6       Status = destroy;
7    }
8    return Status;
9  }
```

최종 접속 시간인 $\Delta TlastAccess$와 세션의 유지시간 동안에 세션 토큰은 active 상태이다.

이 시간 동안에 사용자는 '새로고침'이나 화면 이동의 동작을 한다. 그러나 세션이 파기되면 destroy 상태로 바뀌며, 파기된 세션 토큰을 재전송을 하여도 제안 3.3 절의 방어 기법에 따라 방어된다.

3.3 세션 상태 기반 쿠키 재전송 공격 방어 기법

세션 상태 기반 쿠키 재전송 공격의 방어 기법은 세션 상태 확인 알고리즘의 sessionStatus 메소드(method)를 이용하여 세션의 상태정보를 리턴받아 세션의 유지 여부를 조회

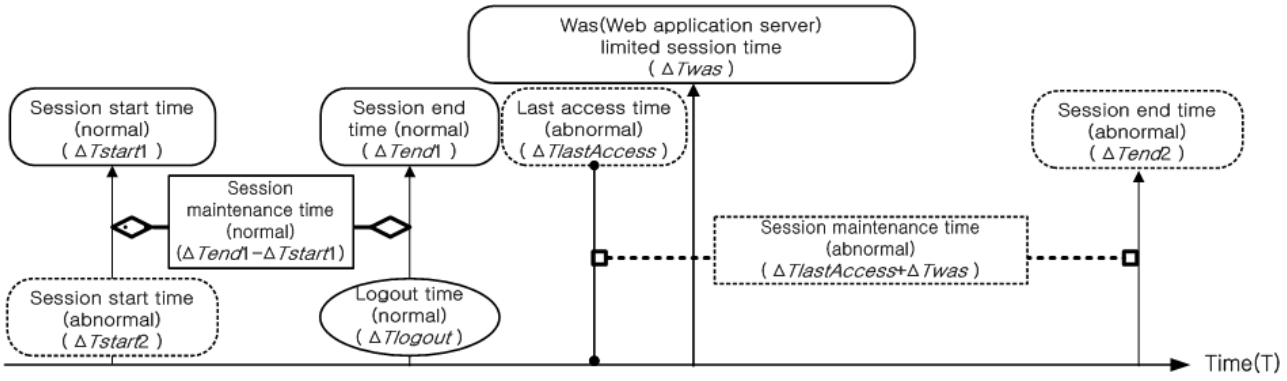


Fig. 2. Session maintenance time in web server

한 후에 방어한다.

웹 서버에서 세션이 유지되고 있다면 정적 변수의 메소드에 저장된 사용자 토큰을 이용하여 방어한다. 사용자 토큰이 널(null)일 경우와 널이 아닌 경우로 나눠서 방어된다. 반면 세션이 만료된다면 로그아웃을 실행한다.

세션 토큰은 사용자의 IP와 로그인한 시간의 타임스탬프 그리고 특수문자를 조합하여 해시 함수를 이용하여 생성하고 로그인 이력 테이블에 저장한다.

Table 2. Algorithm of defense mechanism (login, loginProc)

```
1  //input data1 : userid (user id value)
2  //input data2 : password (user password value)
3  //output data1 : session_token (session value)
4  //output data2 : user_token (static user value)
5
6  login() {
7      Setting a value to null user_token;
8  }
9  loginProc(userid, password) {
10     Create of session_token;
11     Setting a value in the user_token;
12 }
```

Table 3. Algorithm of defense mechanism (Detect, Defense)

```
1  //input data1 : active (session status value)
2  //input data2 : session_token (session value)
3  //input data3 : user_token (static user value)
4
5  //return value of sessionStatus() [active | destroy]
6  Call sessionStatus();
7  //Verification (for defense of cookie replay attack)
8  If(session_token == null) {
9      Execute log out;
10 } Else If(active && user_token == null) {
11     Detect for attack without authentication;
12     Defense for attack without authentication;
13 } Else If(active && session_token != user_token) {
14     Detect for attack after authentication;
15     Defense for attack after authentication;
16 }
```

사용자 토큰은 세션 토큰과 동일한 규칙으로 생성되며, 공격자와 공격당한 사용자를 구분할 때 사용한다. 사용자 토큰은 정적 변수로 웹 서버에 저장된다. 또한 사용자 토큰은 GET이나 POST로 전송되지 않으므로 스니핑할 수 없다. 그리고 우회 공격을 시도하여 강제로 수정할 수 없다. 웹 서버의 정적 변수에 저장된 사용자 토큰을 공격자가 수정할 수 없기 때문이다.

표 2는 쿠키 재전송 공격 방어를 목적으로 제안한 로그인 프로세스 알고리즘이다. login은 사용자가 로그인하기 이전에 실행되는 메소드이다. login 메소드에서 사용자 토큰은 널로 설정되어 정적 변수에 저장된다(7번 라인). 저장된 값은 전역변수로 메소드에 저장되어 모든 웹 애플리케이션에서 공격을 탐지하기 위해 사용된다. 정상적인 로그인 프로세스가 이뤄지면 loginProc 메소드가 실행되며 이 메소드에서 세션 토큰을 생성하고 사용자 토큰에 값을 설정한다(10번 라인, 11번 라인).

표 3은 쿠키 재전송 공격 방어 알고리즘이다. 맨 처음 sessionStatus 메소드를 호출하여 세션이 유지되고 있는지 확인하여 세션이 유지되는 상태이면 공격이 가능하므로 방어 기법의 알고리즘으로 방어한다(6번 라인, 12번 라인, 15번 라인). 쿠키 재전송 공격이 실행되면 탐지는 모든 웹 페이지에서 이뤄진다(11번 라인, 14번 라인). 공격이 탐지되면 방어는 자동으로 이뤄진다(12번 라인, 15번 라인). 세션 토큰에 값이 없으면 로그아웃이 자동 실행된다(9번 라인).

제안한 알고리즘에서 쿠키 재전송 공격은 두 가지 시나리오로 실행되며 이에 따른 방어도 다르게 이뤄진다.

첫 번째, 공격자가 정상 로그인 이후에 공격하는 경우로 공격자의 토큰(user_token)과 스니핑한 세션 토큰이 서로 다르므로 방어된다(15번 라인). 정상적인 로그인을 한다면 세션 토큰과 사용자 토큰은 서로 같다.

두 번째, 공격자가 정상인증을 하지 않고 로그인 화면에서 공격할 경우로 공격자의 토큰(user_token)이 널로 초기화되므로 방어된다(12번 라인). login 메소드는 사용자가 웹 애플리케이션을 접속하였을 때 로그인 인증을 하지 않은 상태이면 최초로 로딩되는 메소드로 사용자 토큰의 값을 널로 초기화한다.

4. 검 증

쿠키 재전송 공격 방어 기법을 검증하기 위한 실험 환경을 구축하였다. 실험환경의 웹 서버는 운영체제를 WindowsXP로 하였으며 WAS(Web Application Server)를 톰캣(Tomcat)으로 구성하였다. 검증을 위한 프로그램은 Java, 스프링(Spring) 프레임워크를 이용하여 개발하였다. 데이터베이스는 MySQL을 이용하여 구축하였다. 웹 서버는 공인 IP를 사용하는 1EA로 구축하여 공격의 방어 가능 여부를 테스트해 보았다. 그리고 공격용 클라이언트는 노트북을 사용하였다. 클라이언트에서 웹 서버로 접속하기 위해서 WiFi를 이용하였으며 클라이언트에서 공격의 가능여부를 실험하였다.

그림 3과 그림 4는 제안한 기법을 검증하기 위해 개발한 프로그램의 데이터 흐름도이며, 각 번호는 흐름순서(seq) 번

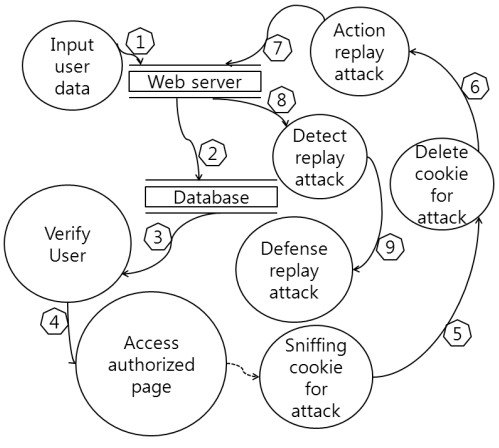


Fig. 3. Data flow diagram of replay attack after authentication

호이다. 그림 3은 사용자가 로그인 정보를 입력하여 웹 서버로 정보를 전송하고, 데이터베이스에서 정보를 확인하여 인증처리를 한다. 정상 인증이 완료되면, 웹 애플리케이션의 서비스를 사용한다(seq 1~4). 그림 3에서는 3.3 절의 쿠키 재전송 공격의 첫 번째 시나리오인 공격자가 정상로그인을 한 상태의 공격을 방어할 수 있다(seq 5~9). 이 경우는 로그인 인증을 수행한 후에 공격이므로 3.3절의 표 3의 13번 라인에서 세션 토큰과 공격자의 토큰(user_ token) 값이 다르므로 방어된다.

그림 4는 공격자가 로그인 인증을 하지 않고 쿠키를 스니핑하여 재전송 공격을 시도하였을 경우로 두 번째 시나리오의 공격 방법이다. 이 경우는 로그인 인증을 하지 않으므로 3.3절의 표 3의 10번 라인에서 공격자의 토큰(user_ token) 값이 널이므로 방어된다.

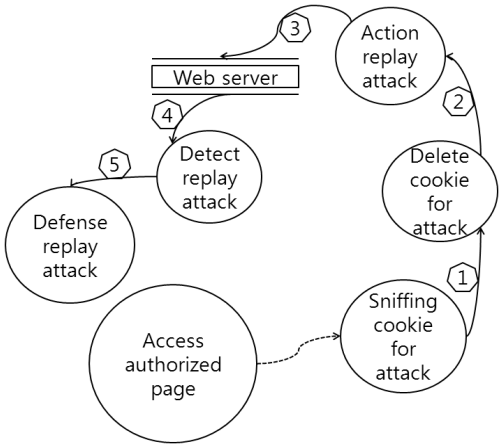


Fig. 4. Data flow diagram of replay attack without authentication

표 4는 정상 인증 이후의 검증 실험 결과를 나타낸 것으로 test1, test2로 IE에서 실험한 결과의 데이터들이다. test1과 test2는 데이터 흐름도의 seq 1~9번까지 실험한 결과를 표 4의 Row1로 정리하였다. Row1은 test1로 로그인하여 test2의 쿠키를 스니핑하여 공격했다.

Table 4. Result data of experiments (attack after authentication)

	Seq	User id	Verification of session_token and user_token
Row1	1	test1	N/A
	2	test1	N/A
	3	test1	TRUE
	4	test1	TRUE
	5	test1	TRUE
	6	test1	TRUE
	7	test1	TRUE
	8	test2	FALSE
	9	N/A	N/A

표 5는 정상 인증 이전의 공격에 대한 검증 실험 결과를 나타낸 것으로 IE에서 실험한 결과의 데이터들이다. 데이터 흐름도의 seq 1~5번까지 실험한 결과를 표 5의 Row2로 정리하였다. 로그인 이전에 test1의 쿠키를 스니핑하여 공격했다.

Table 5. Result data of experiments (attack without authentication)

	Seq	User id	Verification of session_token and user_token
Row2	1	N/A	N/A
	2	N/A	N/A
	3	N/A	N/A
	4	test1	FALSE
	5	N/A	N/A

세션 토큰은 웹 서버에서 세션 유지 시간이 만료되거나 사용자가 로그아웃하기 전까지 남아있게 된다. 검증 결과의 데이터들 중에서 Row1의 공격은 seq 7에서 시작되어 seq 9에서 탐지되어 방어가 된다. 웹 서버에 스니핑한 쿠키를 재전송하기 전인 seq 7에서는 세션에 저장된 사용자 아이디와 세션 토큰은 공격전과 동일한 값으로 유지된다. 그러나 seq 8에서 공격을 당한 웹 서버의 세션에 저장된 사용자 아이디와 세션 토큰은 스니핑하여 재전송된 쿠키에 의해서 변경된다. 하지만 공격자의 토큰(user_token)은 변경되지 않는다. seq 9에서 세션 토큰과 공격자의 토큰(user_token)을 비교한다. 비교한 값이 다르므로 공격은 탐지되고 방어가 이뤄진다.

Row2에서는 로그인을 하지 않은 공격자가 URI로 웹 애플리케이션을 접속하여 공격했을 때의 실험결과이다. 3.3 절의 표 2의 login 메소드를 실행하지만 loginProc 메소드를 실행하지 않으므로 공격자의 토큰(user_token)은 널이다. 따라서, Row1의 경우와 마찬가지로 세션 토큰과 공격자 토큰(user_token)의 확인 결과는 FALSE가 된다.

해당 검증을 통해 쿠키 재전송 공격에 대한 두 가지 시나리오 모두 방어를 할 수 있다는 것을 보았다.

5. 결 론

사용자 정보가 저장된 쿠키는 공격자에 의해 스니핑될 수 있으며, 이렇게 스니핑된 쿠키를 이용하여 공격자는 마치 합법적인 사용자인 것처럼 사용자의 세션을 유지할 수 있다. 또한 세션이 유지되고 있는 상태이면 쿠키 재전송 공격으로 인해 개인 정보 침해에 대한 위험에 노출이 된다. 서버와 클라이언트 간에 통신시 서버측 정적 변수에 저장된 사용자 토큰 값은 스니핑되지 않는다. 쿠키나 세션에 사용자 토큰을 저장한다면 스니핑될 수 있으며 공격에 취약하다. 따라서 세션 정보가 웹 서버에 유지되고 있는지를 확인하기 위한 세션 상태 확인 알고리즘을 제안하였다. 또한 웹 애플리케이션에서 세션 토큰과 공격자가 스니핑을 하거나 수정할 수 없는 웹 서버의 정적 변수인 사용자 토큰을 이용

하여 쿠키 재전송 공격을 방어하는 기법을 제안하였다. 그리고 제안한 기법의 실험을 위해 IE에서 쿠키 재전송 공격을 시도하였고, 쿠키 재전송 공격의 데이터 흐름 단계마다 세션 토큰과 사용자 토큰을 확인하여 효과성을 검증 하였다. 하지만 정적 변수의 활용은 소스 코드를 삽입하여 웹 서버의 데이터를 공격할 수 있는 코드 인젝션 공격에 취약할 수 있다.

따라서 향후 연구에서는 제안한 방어기법에 대한 코드 인젝션 공격의 취약점을 분석한다. 코드 인젝션 공격에 안전한 방어 기법을 추가로 제안하여 이러한 문제점도 해결하고자 한다. 또한 제안한 기법이 모바일 환경에서도 안전하게 쿠키 재전송 공격을 방어할 수 있는지의 여부를 검증한다.

References

[1] WonTae Sim, YoHan Choi, HeeSuk Seo, and BongNam Noh, “A Storage Method to Enhance Cookie File Security”, Journal of the Korea Society for Simulation, Vol.20, No.1, pp.29-37, 2011.

[2] DongHee Kim and JinTak Choi, “A Study on The Efficient Authentication Management Technique of SSO Foundation”, Journal of the Korea Institute of Information Technology, Vol. 4, No.1, pp.55-63, 2009.

[3] Aziz Baayer, Noudding Enneya, and Mohammed Elkoutbi, “Enhanced Timestamp Discrepancy to Limit Impact of Replay Attacks in MANETs”, Journal of Information Security, pp.224-230, 2012.

[4] D. E. Denning and G. M. Sacco, “Timestamps in Key Distribution Protocols”, Magazine Communications of the ACM, Vol.24, No.8, 1981.

[5] Leiba, Barry and Huawei Technologies, “OAuth Web Authorization Protocol”, IEEE Internet Computing, Vol.16, No.1, pp.74-77, 2012.

[6] John Trammel, Umit Yalcinalp, Andrei Kalfas, James Boag, and Dan Brotsky, “Device Token Protocol for Persistent Authentication Shared across Applications”, First European Conference, ESOCC, pp.230-243, 2012.

[7] Mojtaba Ayoubi Mobarhan, Mostafa Ayoubi Mobarhan, and Asadollah Shahbahrami, “Evaluation of Security Attacks on UMTS Authentication Mechanism”, International Journal of Network Security & Its Applications(IJNSA), Vol.4, No.4, pp.37-52, 2012.

[8] Item Dictionary, “Telecommunications Technology Asscociation”, 2014, http://word.tta.or.kr/terms/terms.jsp (Accessed: 27 August 2014).

[9] BokJae Cha, “Analysis of ICT terminology”, 2014, http://www.ktword.co.kr/abbr_view.php (Accessed: 27 August 2014).

[10] Wikipedia, “Replay attack - Wikipedia”, 2014, http://en.wikipedia.org/wiki/Replay_attack (Accessed: 27 August 2014).

[11] Activation of password-KISA, “Hash function-KISA”, 2014, http://seed.kisa.or.kr/iwt/ko/intro/EgovHashFunction.do (Accessed: 28 August 2014).

[12] OWASP, “Category:OWASP Top Ten Project”, 2014, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project (Accessed: 31 August 2014).

[13] MSDN, “HttpCookie.HttpOnly”, 2014, http://msdn.microsoft.com/ko-kr/library/system.web.httpcookie.httponly(v=vs.110).aspx (Accessed: 11 October 2014).

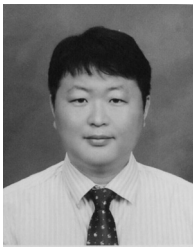
원 종 선

e-mail : whitehacker.jswon@gmail.com

1999년 충북대학교 농공학과(학사)

2011년~현 재 한국방송통신대학교 정보과학과 석사과정

관심분야 : 정보보안, 컴퓨터통신망, 웹 표준



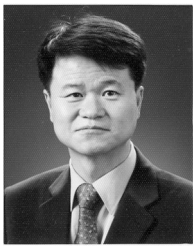
박 지 수

e-mail : bluejisu@korea.ac.kr

2013년 고려대학교 컴퓨터교육(박사)

2013년~현 재 고려대학교 정보창의교육연구소 연구교수

관심분야 : 분산 시스템, 클라우드, 모바일 클라우드 컴퓨팅, e-Learning



손 진 곤

e-mail : jgshon@knou.ac.kr

1991년 고려대학교 전산학전공(이학박사)

1991년~현 재 한국방송통신대학교 컴퓨터과학과 교수

1997년~1998년 State University of New York (Stony Brook) Visiting Professor

2000년~현 재 ISO/IEC JTC1/SC36 Korea Delegate

2005년~2007년 국무조정실 이러닝산업발전실무위원회 위원

2009년~현 재 한국정보처리학회 영문지(JIPS) 편집위원

2009년~현 재 이러닝학회 부회장

2010년 한국정보처리학회 부회장

2013년~2014년 Indiana University Visiting Professor

관심분야 : 컴퓨터통신망, 분산시스템, 그리드 컴퓨팅, e-Learning, 정보기술 표준화