웹 어플리케이션에서의 재전송 공격 방어 알고리즘

원종선*, 손진곤
한국방송통신대학교 대학원 정보과학과
jswon7942@naver.com*, jgshon@knou.ac.kr

A Defence Algorithm against Replay Attacks in Web Applications

Jong Sun Won*, Jin Gon Shon
Dept. of Computer Science, Graduate School
Korea National Open University

요 약

회원가입 정보를 이용하는 대다수의 웹 어플리케이션은 쿠키 및 세션을 이용한 인증기법으로 로그인 인증을 한다. 그러나, 쿠키에 의존하여 로그인 인증을 하는 웹 어플리케이션은 재전송 공격(Replay Attack)에 취약하다. 재전송 공격이란 사용자가 로그인할 때 쿠키의 정보를 해커가 스니핑(Sniffing)하여 Opera 웹브라우저의 쿠키 관리자에서 강제로 재전송하여 위장할 수 있는 공격을 말한다. 본 논문에서는 쿠키와 세션 인증키를 이용하여 재전송 공격을 방어할 수 있는 알고리즘을 제안하였다. 그리고, 클라이언트와 서버간에 신뢰할 수 없는 통신으로 인한 스니핑의 문제점을 서버에만 저장된 암호화된 세션 인증키로 개선하였다.

1. 서론

현재 많은 웹 어플리케이션에서 인증을 위한 방식으로 쿠키와 세션을 이용한 인증이 폭넓게 활용되고 있다. 많은 웹 어플리케이션에 사용되는 인증방식은 패스워드 인증 방식이다[1]. 이 방식은 폭넓게 활용되고 있는 만큼 해커들에게 많은 취약점들이 노출되어 최근까지 공격의 대상이 되고 있다. 쿠키와 세션에 저장되어 인증되는 과정에 도청용 소프트웨어에 의해 쿠키와 세션 아이디가 도청되어 위장에 이용되고 있어서 문제가 심각하다. 이러한 위장 공격의 대표적인 경우가 재전송 공격이라고 할 수 있다.

재전송 공격을 방어하기 위한 대표적 방식으로 타임스탬프(Timestamp), 시도-응답(Challenge-Response), 커버로스(Kerberos) 방식들이 있다[1]. 이러한 방식들은 개발자가 개발하기에 복잡하며 보안성도 완벽하다고 볼 수 없다. 이들 방식은 상대방 상호간의 인증방식이 양방향 인증인데 클라이언트와 서버가 통신을 하는 동안 적절히 보호되지 않는다면 안전하다고 할 수 없다. 기존의 방식들이 암호화된 형태의 정보를 송수신하는 방법이지만 해커들에 의해 스니핑(Sniffing)될 수 있고 암호화된 정보가 노출된다면 더 이상 안전하다고 볼 수 없다.

안전한 서버 접속을 위한 사용자 기법 연구에서 사용자 인증정보를 저장한 후에 세션 아이디를 비교하여 재전송 공격에 방어하는 기법을 이용하였다[2]. 그러나, 세션 아이디는 일종의 쿠키 값들의 집합으로서 쿠키 값을 스니핑하여 재전송 공격에 이용될 수 있다.

또한, 웹 어플리케이션에서 인증과 세션 취약점의 개선 방안에 관한 연구에서는 보안 세션 토큰(Session Token)을 만들어 유효성을 검증하였다. 세션 토큰에 저장되어 있는 IP와 현재 접속한 IP가 같고, 클라이언트의 쿠키에 저장된 세션 토큰에 저장된 값과 서버에서 생성한 세션 토큰 값이 같을 경우에 사용자의 요청을 처리하여 안전한 세션관리를 위한 프로그램 작성방법을 제안하였다[3]. 하지만 클라이언트의 IP와 쿠키에 저장된 클라이언트의 세션토큰이 통신과정에 스니핑될 수 있어서 개선할 필요가 있다.

2. 관련연구

재전송 공격을 방어하기 위한 대표적 방식들은 타임스탬프, 시도-응답, 커버로스 방식이 있다. 또한, 웹 어플리케이션의 클라이언트-서버 모델에서 일반적으로 가장 많이 사용하고 있는 방식으로 패스워드 인증 방식, 변형 패스워드 인증 방식이 있다[1].

2.1 재전송 공격방어를 위한 대표적 방식

양방향 인증이라고도 불리며 통신 실체 상호간에 인증을 제공하여 통신함에 있어서 상대방이 안전한 실체임을 판단하기 위해 사용된다. 가장 많이 사용되는 부분은 세션키 교환이다.

①타임스탬프(Timestamp) : 사용자 A는 메시지가 현재 시간에 대하여 충분히 밀접하다고 판단되는 타임스탬프를

포함할 때만 메시지를 신뢰하도록 한다. 이 방법을 적용하기 위해서는 각 참여자들이 동기화된 시계를 가져야 한다는 조건이 필요하다.

②시도-응답(Challenge-Response) : A는 B로부터 신뢰된 메시지를 기대하고, B에게 B에 대한 시도를 보낸 다음 B로부터 수신되는 응답이 정확한 시도 값을 포함할 것을 요구한다.

③커버로스(Kerberos) : 분산 인증서비스 상에서 무결성과 기밀성을 제공한다[1].

④세션 토큰(Session Token) : 사용자의 인증확인을 위해서 사용자 인증과 요청 시 클라이언트에서 악의적인 데이터 변조가 일어났는지 상호 대조할 수 있도록 사용자의 정보를 이용하여 암호화된 세션 토큰을 만들어 클라이언트와 서버에 각각 저장하여 요청시 이를 비교하여 토큰의 유효성을 확인해야 한다[3][4][5].

2.2 사용자 인증 기법

회원정보를 관리하는 웹 어플리케이션의 대다수는 패스워드 인증방식을 이용해서 개발된다.

①패스워드 인증 방식 : 클라이언트-서버 모델에서 일반적으로 가장 많이 사용하고 있는 방식으로 접근 통제를 위해 사용자 자신의 아이디에 대해 패스워드를 입력함으로써 인증을 받게 된다. 그러나, 이 방식은 패스워드 전송 노출, 패스워드 재전송, 서버 인증 정보 공격, 그리고 만약 사용자의 비밀통신을 수행하고자 할 경우 별도의 키를 생성해야 하는 번거로움이 발생하는 등의 문제점이 있다.

②변형 패스워드 인증 방식 : 이 방식은 패스워드 방식의 문제점을 해결하기 위하여 일방향 해쉬함수를 도입하였다. 즉, 사용자의 인증 정보(아이디, 패스워드)를 해쉬함수를 이용하여 전송함으로써, 제3자에 의한 도청을 방지하여 전송 노출을 막고 있다. 또한 패스워드 재전송 공격을 방지하기 위해 랜덤 값을 사용하고 서버인증 정보 공격을 막기 위해 해쉬된 정보를 그대로 저장함으로써 안전성을 획득하고 있다[1].

3. 암호화된 세션 키 알고리즘

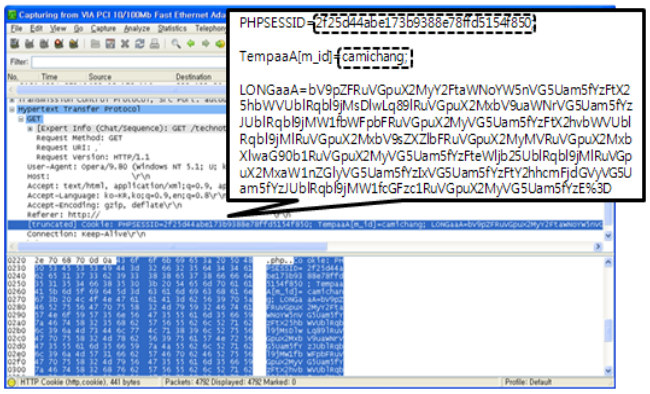
웹 어플리케이션에서 보안에 취약한 패스워드 인증기법을 분석하여 재전송 공격의 과정을 파악하였다. 본 연구에서 재전송 공격을 방어하기 위해서 사용자 아이디를 변형하고 DES(Data Encryption Standard)로 암호화하여 인증키를 생성하였다. 그리고, 인증키는 세션에 저장하여 재전송 공격을 할 때 데이터베이스에 저장된 정보와 일치여부를 검사하는 알고리즘을 제안하였다.

3.1 패스워드 인증기법 분석

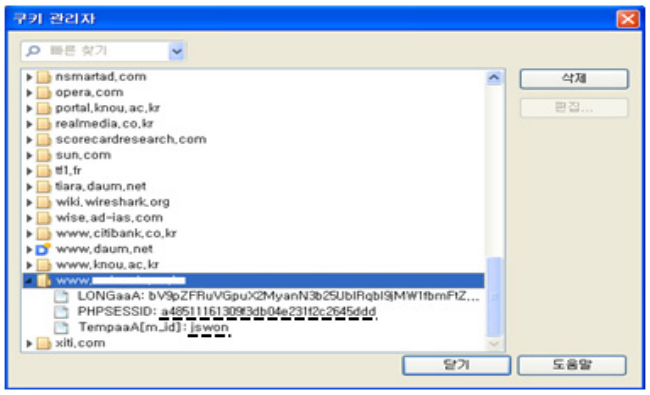
신뢰할 수 없는 쿠키의 사용에 의한 재전송 공격에 대해 패스워드 인증기법의 취약점을 분석하기 위해서 스니핑할 수 있는 프로그램인 wireshark-win32-1.4.7을 이용하였다. (그림 1)과 같이 쿠키 값을 도청하여 쿠키를 다른 컴퓨터에서 재전송하여 인증되는 테스트를 하였다. 스니핑한 쿠키 정보를 Opera 웹브라우저에 입력하여 전송한 후 도청한 사용자 계정으로 위장이 가능하였다. (그림 1)은 재전송 공격하기 전에 정상적으로 인증된 프로세스를 스니핑한 쿠키 값이다.

(그림 2)는 Opera 웹브라우저를 이용하여 쿠키관리자 화면에서 쿠키 정보를 웹브라우저에 강제 입력한 화면이다. 이때 강제 입력할 쿠키 값은 wireshark-win32-1.4.7를 이용하여 재전송 공격하기 전에 미리 텍스트 편집기에 저장한다. (그림 2)와 같이 쿠키관리자에 값을 강제입력한 후 Opera 웹브라우저를 '새로고침'한다. '새로고침'한 후에 웹브라우저에서 사용자의 계정이 재전송 공격 후와 같은 값으로 변경된 것을 확인할 수 있다.

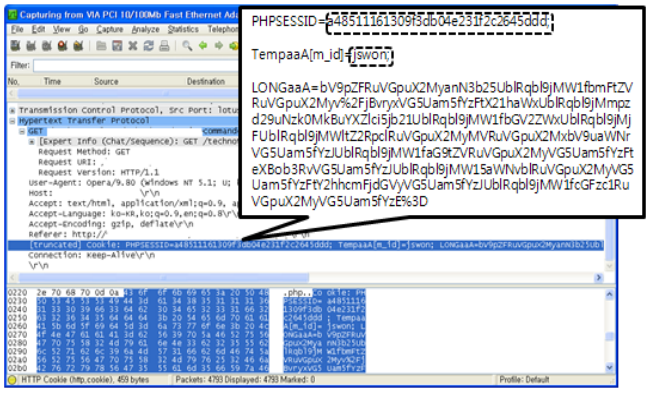
(그림 3)은 위장한 계정으로 로그인 정보가 수정된 것을 wireshark-win32-1.4.7에서 확인한 화면이다.



(그림 1) 정상 인증



(그림 2) 재전송 공격



(그림 3) 재전송 공격 후

분석 및 실험을 통하여 알아낸 사실은 패스워드 인증 방식인 쿠키에 의존한 인증기법을 사용하는 웹 어플리케이션들은 재전송 공격에 취약하다는 점이다.

3.2 회원가입시 발급되는 인증키 생성 알고리즘

회원가입시 발급되는 인증키 생성 알고리즘은 회원가입일과 회원아이디의 인덱스를 조합한 후에 블록 암호법인 DES로 암호화한다. 암호화되기 이전 단계에서 회원가입일, 회원아이디를 데이터베이스의 매핑 테이블과 비교하여 인증키를 생성하는 알고리즘이다.

매핑할 때는 (그림 4)와 같이 회원아이디를 배열 루프로 하고 중첩 루프로 매핑테이블과 64번씩 비교하여 아이디의 매핑 인덱스를 검색하게 된다. 64개의 문자는 영문대문자 26개, 영문소문자 26개, 0에서 9까지 숫자 10개, 언더스코어, 마침표로 구성하였다. 이때 중첩 루프가 수행되게 되는데 프로세스의 수행성능을 저하시킨다면 데이터베이스의 스토어드 프로시저와 같은 프로그램으로 모듈화하여 성능을 개선할 수도 있을 것이다. 생성된 배열의 인덱스와 회원가입일을 조합하여 회원의 인증키를 생성하는 알고리즘의 의사코드는 (그림 4)와 같다.

```
Algorithm encryptAuthKey(userId)

//userId : 회원아이디
//userId_len : 회원아이디 길이
//Arr : 데이터베이스 매핑테이블에서 인덱스를 로딩하여 생성된 배열
//userId_charAt : 회원아이디에서 잘라낸 한개의 문자
//tempUserPk : 회원아이디를 이용하여 매핑테이블에서 로딩한 임시 키
//tempChar2Str : 문자를 치환한 문자열
//RandArr : 매핑테이블 배열의 난수
//RandArr_len : 인덱스 로딩한 난수배열의 길이
//MemberTable : 데이터베이스의 회원관리 테이블
//userPk : 회원 인증키
//getToday : 회원가입일
//DES : 암호화 알고리즘

begin
        //회원아이디 길이 초기값
        i = 0;
        //인덱스 테이블 배열의 초기값
        j = 0;

        //매핑테이블 배열 난수발생
        RandArr = Randomize(Arr);

        while (i<userId_len)
        {
                while(j<RandArr_len)
                {
                        if (userId_charAt[i] == RandArr[j])
                        {
                                tempUserPk += tempChar2Str;
                        }
                        j++;
                }
                i++;
        }

        userPk = getToday + tempUserPk;
        The userPk is encrypted by DES;

        Insert into MemberTable values of the RandArr and userPk;
end
```

(그림 4) 회원 인증키 생성 알고리즘의 의사코드

3.3 회원 인증키를 이용한 인증 알고리즘

인증 알고리즘은 로그인할 때 GET공격에 방어할 수 있는 알고리즘을 포함하고 있다. 웹 전송방식인 GET방식과 POST방식을 검사하여 GET방식은 로그인 실패로 처리하며, GET으로 계속 공격할 때 블랙리스트 IP로 처리하여 관리한다. POST로 회원의 아이디와 비밀번호가 전송되면 데이터베이스에 회원아이디와 비교하여 해당 아이디의 존재유무를 검사하며, 아이디가 존재하면 비밀번호를 검사하여 로그인 인증을 완료하고 암호화된 세션 인증키를 생성한다.

인증이 완료된 경우 인증키는 세션에 저장하여 세션이 종료될 때까지 로그인이 유지되도록 한다. 그 외의 회원정보는 기존의 방식처럼 쿠키로 관리하여 시스템의 자원 낭비를 줄였다. 또한, 회원마다 한 개의 세션만 발급한 후 인증키로서 활용함으로써 시스템의 부하를 감소시켰다.

인증 알고리즘의 의사코드는 아래 (그림 5)와 같다. (그림 5)에서 키는 세션으로 저장되어 각 페이지를 이동하는 경우 정보를 유지하게 되며 세션이 강제 종료되거나 회원이 로그아웃할 때 자동으로 세션이 종료되도록 하였다.

```
Algorithm loginProcessAction()

//requestId : 클라이언드가 POST방식으로 입력한 아이디
//requestPwd : 클라이언드가 POST방식으로 입력한 비밀번호
//DBId : 데이터베이스에서 검색한 아이디
//DBPasswd : 데이터베이스에서 검색한 비밀번호
//cookieFile : requestId의 쿠키파일
//sessionAuthKey : 세션에 등록된 인증키
//authKey : 데이터베이스의 매핑테이블과 연동된 인증키

begin
        success = FALSE;

        if(DBId == requestId) {
                if(DBPasswd == requestPwd) {
                        success = TRUE;

                        //ID만 Cookie파일생성
                        Create cookieFile from requestId;

                        //authKey는 세션정보로 유지
                        Create sessionAuthKey from authKey;
                } else {
                        success = FALSE;
                }
        } else {
                success = FALSE;
        }

        if(success == TRUE) {
                login process is allowed;
        } else {
                login process is failed;
        }
end
```

(그림 5) 인증 알고리즘의 의사코드

4. 검증

웹 어플리케이션에서 인증기법으로 이용되고 있는 신뢰할 수 없는 쿠키에 의존한 인증기법은 재전송 공격에 취약함을 확인하였다. 본 논문에서 분석에 이용된 웹 어플리케이션과 비슷한 환경으로 웹 어플리케이션을 개발했다. 제안한 재전송 공격에 방어할 수 있는 알고리즘으로 구현한 웹 어플리케이션을 로컬 호스트에서 테스트하여 재전송 공격에 방어 가능 여부를 검증하였다.

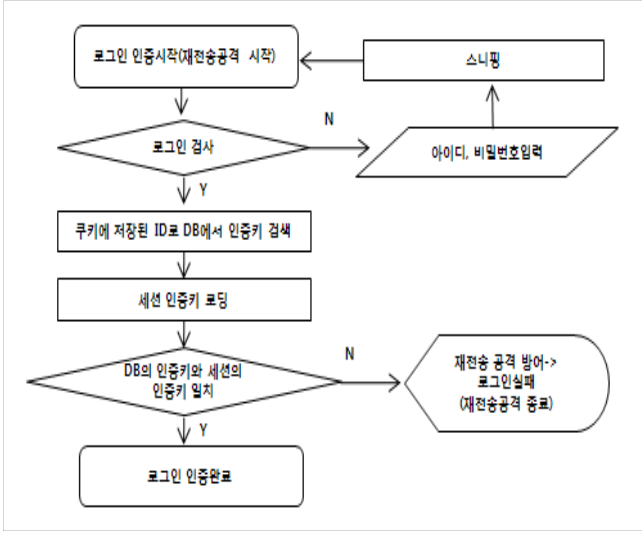
톰캣, Java JDK 1.6.0_05를 이용하여 웹서비스를 구성하였으며 웹 프로그래밍 언어는 JSP, 웹서버 운영체제는 Windows XP, 데이터베이스는 MySQL 환경으로 구성하

여 실험을 실시하였다.

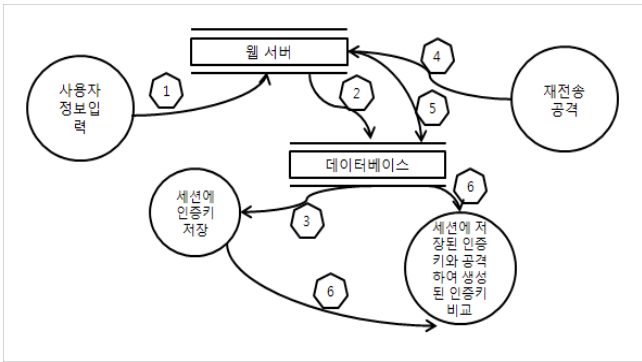
실험의 프로세스는 스니핑을 통해 도청한 다른 사용자의 쿠키 값을 Opera 웹브라우저에서 강제로 입력하여 재전송 공격을 시도하는 방식이다. 재전송 공격을 시도할 때 클라이언트의 쿠키에 저장된 회원 아이디로 데이터베이스에서 인증키를 검색하여 인증할 때 세션에 저장하였던 인증키와 일치 여부를 검사하였다. 즉, 재전송 공격에 이용된 회원 아이디와 매핑된 데이터베이스의 인증키가 로그인할 때 세션에 저장된 인증키와 일치하지 않으면 인증 실패로 처리하여 재전송 공격을 방어할 수 있었다.

웹 어플리케이션의 클라이언트는 서버에 클라이언트의 정보를 보낸다. 서버는 클라이언트에게 응답메시지를 보내어 인증한다. 인증될 때 세션 인증키를 생성하여 재전송 공격에 방어할 수 있도록 한다. 인증을 실시한 후에 Opera 웹브라우저의 쿠키 관리자를 이용하여 강제로 쿠키 값을 입력한다. 재전송 공격이 이뤄질 것 같지만 이미 세션에 저장된 세션 인증키와 위장될 계정의 인증키를 비교하였을 때 일치하지 않으므로 재전송 공격을 방어할 수 있다(그림 6 참조).

검증을 위해 구현된 프로그램의 인증내부 동작은 (그림 7)과 같은 인증 흐름도로 나타낼 수 있다. <표 1>은 테스트 프로그램의 인증 흐름도의 결과로 나타난 쿠키 및 인증키를 보여준 것이다.



(그림 6) 테스트 프로그램의 순서도



(그림 7) 테스트 프로그램의 인증 흐름도

<표 1> 테스트 프로그램의 쿠키 및 인증키

순서	쿠키에 저장된 아이디	데이터베이스에서 로딩한 인증키	세션에 저장된 인증키
1	test	N/A	N/A
2	test	N/A	N/A
3	test	609389540de8f0271245d257dd570b29c03d3eb17e6c25f4	609389540de8f0271245d257dd570b29c03d3eb17e6c25f4
4	scientist	N/A	N/A
5	scientist	609389540de8f0271245d257dd570b29c03d3eb17e6c25f4	609389540de8f0271245d257dd570b29c03d3eb17e6c25f4
6	scientist	609389540de8f0277dafbd199909ad01cba48e096166a891380e026b8e39978	609389540de8f0271245d257dd570b29c03d3eb17e6c25f4

5. 결론

본 논문에서는 쿠키인증 과정에 스니핑된 정보를 이용하여 재전송 공격을 할 때 방어 가능한 암호화된 세션 인증키 알고리즘을 제안하였다.

제안한 알고리즘은 클라이언트와 서버간에 신뢰할 수 없는 통신으로 인한 스니핑의 문제점을 서버에만 저장된 암호화된 세션 인증키로 개선하였다. 그리고, 해커가 재전송 공격을 감행할 때 세션 인증키와 데이터베이스의 매핑 테이블에서 로딩된 인증키를 서로 비교하여 재전송 공격을 방어할 수 있었다.

본 논문에서는 알고리즘의 테스트를 로컬 서버에서 수행하였으므로 실제 운영환경에서 테스트하여 일반화할 필요가 있다.

참고문헌

[1] 김동희, 최진탁, Single Sign-On 기반의 효율적인 인증 관리 기법에 관한 연구, 한국정보기술학회논문지, 제4권, 제3호, pp.55-63, 2009

[2] 황희태, 안전한 서버 접속을 위한 사용자 인증 기법, 단국대학교, 석사학위논문, 2008

[3] 김종섭, 웹 어플리케이션에서 인증과 세션 취약점 개선방안에 관한 연구, 고려대학교, 석사학위논문, 2005

[4] http://en.wikipedia.org/wiki/Replay_attack

[5] 맹영재, 양대헌, Single Sign-On 솔루션의 재전송 공격 취약점 분석, 정보보호학회논문지, 제18권, 제1호, pp.103-114, 2008