情報保護學會論文誌
第18卷 第1號, 2008. 2

Single Sign-On 솔루션의 재전송 공격 취약점 분석*

맹 영 재[1†], 양 대 헌[1‡]
[1]인하대학교 정보보호연구실

An Analysis of Replay Attack Vulnerability on Single Sign-On Solutions*

YoungJae Maeng[1†], DaeHun Nyang[1‡]
[1]Information Security Research Laboratory, INHA University

요 약

Single Sign-On은 한 번의 로그인으로 여러 시스템에 인증된 상태로 접근할 수 있게 해주는 인증기술이다. 웹 서비스를 그룹단위로 통합하는 곳이 늘어감에 따라 이를 위해 다양한 Single Sign-On 솔루션이 개발되어 사용되고 있지만 이러한 솔루션들의 보안성은 대부분 쿠키에만 의존하기 때문에 공격자는 단순한 네트워크 도청과 재전송 공격을 통해 사용자의 세션을 가로채는 것이 가능하며 세션을 가로챈 이후 Single Sign-On이 적용된 타 사이트로도 이동할 수 있어 또 다른 보안문제를 발생시킬 수 있다. 본 논문에서는 유명 포털사이트 및 메신저에서 사용되고 있는 Single Sign-On 솔루션을 예로 들어 이러한 취약점을 분석하고 사용자의 세션을 보호하는 방법을 제안한다.

ABSTRACT

Single Sign-On is an authentication scheme that enables a user to authenticate once and then to access to the resources of multiple software systems without re-authentication. As web services are being integrated into a single groupware, more web sites are adopting for user convenience. However, these Single Sign-On services are very dependent upon the cookies and thus, simple eavesdropping enables attackers to hijject the user's session. Even worse, the attacker who hijacked one session can move to another site through the Single Sign-On. In this paper, we show the vulnerabilities of the top ranked sites regarding this point of view and also propose a way to protect a user's session.

Keywords : Single Sign-On, Cookie, Session, Replay Attack, Hypertext Transfer Protocol

I. 서 론

최근의 웹 서비스들을 살펴보면 한 번의 로그인만으로도 그 웹 서비스와 연동된 서비스들을 별도의 로그인 단계 없이 연속적으로 이용할 수 있는 곳을 쉽게 찾아볼 수 있다. 이러한 웹 서비스들은 이전의 웹 서비스들이 동일한 기업의 웹 서비스라 하더라도 사이트별로 로그인을 요구했던 것과는 다르게 웹 서버들 간의 인증정보를 공유할 수 있도록 Single Sign-On(SSO)을 적용한 것이다. SSO이 적용되면 단일 인증을 통해 다양한 서비스를 이용할 수 있기 때문에 사용자에게는 여러 번 로그인을 시도해야 하는 불편함을 덜고 기업에는 기업 내 다양한 응용시스템 도입 및 운영에 따라 복잡해질 수 있는 ID 관리를 중앙집중식으로 관리할 수 있도록 하여 인증정책과 권한설정을 용이하게 한다. 이러한 장

접수일: 2007년 6월 15일; 채택일: 2007년 11월 28일
* 이 논문은 인하대학교의 지원에 의하여 연구되었음.
† 주저자, brendig@seclab.inha.ac.kr
‡ 교신저자, nyang@inha.ac.kr

점들 때문에 금융, 포탈, 지방자치단체, 방송국, 대학 등 대부분의 웹 서비스들에 SSO이 빠르게 적용되고 있는 추세이다.

SSO이 기술적으로 의미하는 바는 인증 정보의 공유이다. 인증 정보를 공유하는 방법은 서버들의 환경, 적용범위 등에 따라 다르고 기존의 웹 어플리케이션에 SSO을 적용시키는 방식 또한 다양해질 수 있기 때문에 자체개발 또는 아웃소싱을 통하여 SSO를 이용하고 있다. 그러나 몇몇의 SSO 솔루션들은 보안성이 충분히 고려되지 않은 상태로 제작되었기 때문에 사용자가 SSO이 적용된 타 서비스로 이동시에 네트워크를 통해 전송되는 SSO 토큰을 공격자가 도청(sniffing)한 후 재전송(replay attack)하는 것만으로도 사용자의 계정으로 로그인 할 수 있었다. 이와 같은 취약점은 사용자의 세션을 관리하는데 있어서 신뢰할 수 없는 쿠키(http cookie)를 이용하기 때문이다. 쿠키를 이용하여 사용자의 세션을 관리하는 경우, 위와 같은 공격은 예전부터 알려진 취약점이지만 대부분은 이러한 사실을 간과하고서 쿠키를 사용하는 SSO을 적용시키고 있다. SSO이 적용된 사이트에서 재전송 공격이 가능할 경우에는 SSO의 특성 중 하나인 타 사이트에 대한 손쉬운 접근성을 공격자도 그대로 가지게 된다. 이에 따라 공격자는 SSO이 적용된 모든 사이트에 사용자의 계정으로 인증된 상태로 접근할 수 있게 되어 이전보다 더욱 광범위한 범위의 보안 문제를 발생시킬 수 있다는 것을 의미한다. 따라서 본 논문에서는 사용자의 세션을 도청으로부터 안전하게 보호하는 방법을 제안한다.

본문의 2장에서는 SSO과 쿠키 기반의 사용자 세션을 안전하게 설계하는데 필요한 요구사항에 대해 살펴보고 3장에서는 SSO구현 방법들에 대해 알아본다. 4장에서는 유명한 포털사이트와 국내에서 가장 많이 사용되는 N메신저 등에서 사용된 취약한 SSO솔루션들을 분석하고 5장은 웹에서 SSO과 같은 사용자 세션을 안전하게 유지하기 위한 방법을 알아본다. 6장은 결론을 담는다.

Ⅱ. 안전한 사용자 세션과 SSO 요구사항

안전한 SSO를 설계하기 위해서는 다음과 같은 요구사항을 만족해야 한다[2].

2.1. 기밀성

사용자의 로그인 세션 정보를 포함하는 SSO 토큰은 송/수신될 때 공격자가 내용을 확인할 수 없도록 암호화 되어야 한다. 암호화된 SSO 토큰이 Offline Dictionary Attack등으로 원문이 노출되었다 하더라도 정상적인 사용자의 민감한 정보가 담겨있지 않도록 SSO 토큰은 인증을 위한 정상적인 사용자의 간접적인 인증 정보만을 가지도록 설계한다.

2.2. 무결성

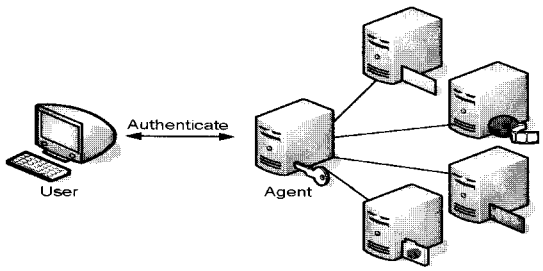
로그인 또는 SSO을 위해 사용자에게 전송되는 폼은 위조 또는 변조될 수 있다. 서버가 사용자를 인증하기 위해 폼에 저장해놓은 정보가 그대로 노출되어 악용될 수 있기 때문에 이러한 폼을 이용할 경우 사용자에게 전송된 폼의 위조 또는 변조가 불가능 하도록 하거나 위조 또는 변조되었을 경우 서버가 이러한 사실을 알아낼 수 있도록 설계해야 한다.

2.3. 재전송 공격 보안

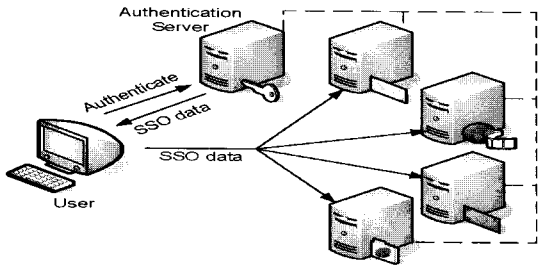
2.1절에서 언급한바와 같이 이더넷에서 동일 네트워크의 트래픽은 공유되어 있기 때문에 정상적인 사용자의 패킷은 간단하게 도청될 수 있으며 이를 통해 공격자는 재전송을 시도할 수 있다. 따라서 재전송 공격에 대해서 안전한 SSO을 설계해야 한다. 재전송 공격에 대한 취약점을 근본적으로 해결하기 위해서는 SSO 토큰을 네트워크에 노출 시키지 않아야 한다. SSO 토큰을 노출시키지 않기 위해서는 정상적인 사용자와 SSO 인증서버 사이에 SSL(Secure Socket Layer)[5] 등과 같은 암호채널을 사용하거나 안전한 키 공유를 이용한 안전한 암호 프로토콜을 이용하여 SSO 토큰을 전달해야 한다. SSO 토큰이 노출되었다 하더라도 재전송 공격에 취약하지 않도록 사용자마다 고유한 SSO 토큰을 가지지 않도록 해야 한다.

Ⅲ. SSO 구현 방법

SSO 구현 방법은 서비스 서버들의 환경과 적용 범위에 따라 다양해질 수 있으며[1,4] 사용자 인증 방식에 따라 크게 인증 대행 방식과 인증정보 전달 방식으로 나뉜다.



〔그림 1〕. 인증 대행 방식 SSO



〔그림 2〕. 인증정보 전달 방식 SSO

3.1. 인증 대행 방식

[그림 1]은 사용자의 인증 정보를 에이전트가 관리하고 사용자 대신 서비스 서버에 로그온 해주는 방식이다. 따라서 에이전트는 각각 다를 수 있는 서비스 서버들의 사용자 계정 정보를 직접 소유하고 있거나 각 서비스 서버와의 통신을 통하여 사용자 대신 인증할 수 있어야 한다. 사용자와 에이전트와의 인증이 안전하게 이루어진다면 실제로 SSO과 관련된 정보는 클라이언트를 거치지 않고 에이전트와 서비스 서버들 사이에서만 전달되기 때문에 사용자와 에이전트 사이의 채널이 보호된다고 가정하면 인증 대행 방식은 비교적 안전한 방식이라고 할 수 있다.

3.2. 인증정보 전달 방식

사용자가 인증 서버를 통해 인증에 성공하면 사용자는 인증 서버로부터 서비스 서버로 전달할 SSO 토큰을 발급받는다. 사용자가 서비스 서버로 접근할 때 SSO 토큰을 자동으로 전달하도록 하여 서비스 서버가 사용자를 확인할 수 있도록 하는 방식이다. 웹 환경에서는 HTTP의 POST, GET형식의 변수와 쿠키(Cookie)를 이용할 수 있으며 실제로 웹 환경에서의 SSO은 주로 쿠키를 이용한 모델을 채택하고 있다. 하지만 SSO 토큰이 클라이언트를 거쳐 이동하기 때문에 스니핑으로 SSO 토큰이 쉽게 노출될 수 있고 쿠키 또한 신뢰할 수 없기 때문에 재전송, Man In The Middle 공격 등에 취약점이 발생하기 쉽다[3]. 인증정보 전달 방식은 SSO을 적용할 서비스 서버들이 위치한 쿠키 도메인(Cookie Domain) 범위에 따라 구현방법이 달라진다. 서버들의 상위 도메인 주소가 같다면 동일한 쿠키 도메인에 속한 것으로 해석된다. 다시 말해 쿠키를 생성할 때 적용 범위를 상위 도메인으로 설정하면 동일한 쿠키 도메인에 속한 서비스 서버들을 이동할 때마다 HTTP request의 header에 클라이언트의 쿠키가 자동으로 전송되기 때문에 구현하기가 간편하다는 장점이 있다. 만약 서버들의 쿠키 도메인이 다르다면 POST와 GET변수를 사용하여 구현하는 방법이 있다.

3.2.1. 단일 쿠키 도메인에서의 SSO

서비스 서버들이 하나의 쿠키 도메인 내에 존재할 때는 간편하게 쿠키를 이용할 수 있다. 인증을 받은 사용자의 웹 브라우저는 SSO 토큰을 쿠키에 저장하고 지정된 쿠키 도메인 안에서 웹 브라우저는 서비스 서버에 접속할 때마다 자동으로 쿠키를 전송한다. 쿠키가 포함된 request를 받은 SSO 에이전트는 SSO 토큰으로부터 사용자를 확인하고 요청된 자원에 대해 접근 권한이 있을 경우 response한다.

3.2.2. 다양한 쿠키 도메인에서의 SSO

서비스 서버들이 여러 도메인으로 분산되어 있는 멀티 도메인(Multi Domain) 환경인 경우에는 POST와 GET 변수를 이용하여 SSO 토큰을 공유한다. 인증된 사용자에게는 SSO 토큰을 발급하고 클라이언트가 이 정보를 각 서비스 서버에게 전달한다.

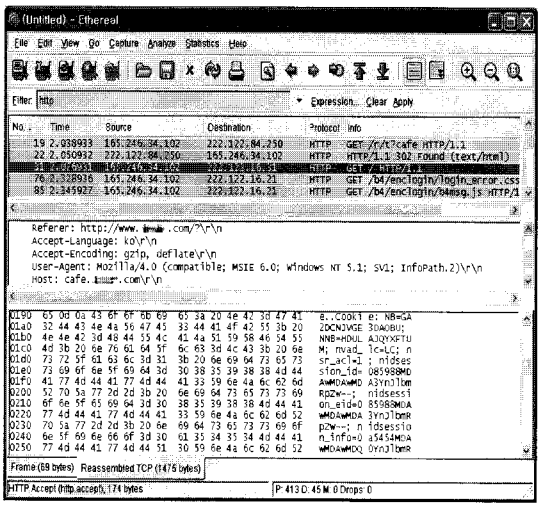
쿠키는 클라이언트에서 임의로 수정될 수 있기 때문에 SSO 인증에 있어서 신뢰할 수 없는 쿠키에 인증정보를 직접적으로 저장하는 것은 안전한 방법이 아니다. 따라서 쿠키는 클라이언트를 신뢰할 수 있도록 해주는 다른 암호기술들과 함께 사용되어야 한다. 4장에서 실제로 사용되고 있는 SSO 솔루션들의 취약점을 분석해 본다.

Ⅳ. 재전송 공격에 취약한 SSO 솔루션 분석

SSO은 거의 모든 종류의 홈페이지에 적용되고 있으며 사용된 SSO 솔루션도 다양하다. 이 절에서는 다양한 종류의 웹서비스에 적용된 SSO의 취약점을 알아본다. 네트워크를 스니핑 할 수 있는 프로그램 중 하나인 ethereal을 이용하여 네트워크를 스니핑하고 스니핑을 통해 얻어낸 사용자의 HTTP request 또는 SSO 토큰이 포함된 쿠키를 다른 컴퓨터에서 서비스 서버에 그대로 전송하여 취약성을 판단한다. 사용자 인증이 된 상태로 HTTP response가 수신되었다면 재전송 공격에 성공한 것이다.

4.1. 유명 포털사이트

대부분의 포털 사이트는 사용자의 로그인 과정에 SSL[5]을 적용하거나 공개키 방식의 프로토콜을 이용하여 사용자의 아이디와 패스워드를 보호하고 있다. 하지만 로그인 이후의 사용자 세션은 대부분 쿠키에만 의존하고 있기 때문에 공격자는 사용자의 쿠키정보를 포함한 HTTP request를 [그림 3]과 같이 도청 후, 쿠키정보를 브라우저에 강제입력(그림 5)하여 재전송하면 쿠키에 담겨있는 인증정보(그림 4)를 토대로 사용자 인증에 성공하게 되어 신분을 위장한 상태로 서비스를 이용할 수 있다. [그림 6]은 스니핑 한 쿠키정보를 opera 브라



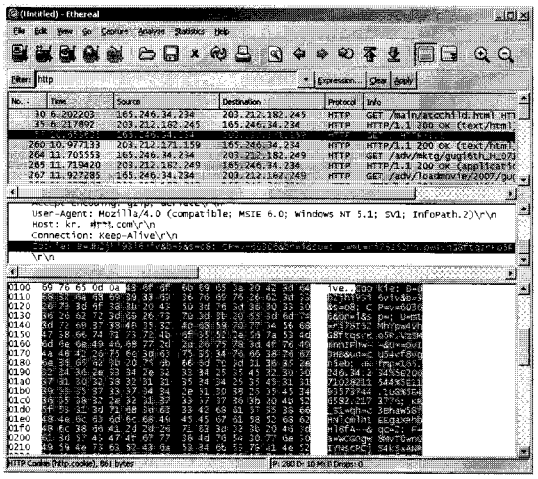
〔그림 3〕. N사 사용자의 HTTP request 도청 화면

〔그림 4〕. 도청한 쿠키정보를 opera에 모두 입력한 모습

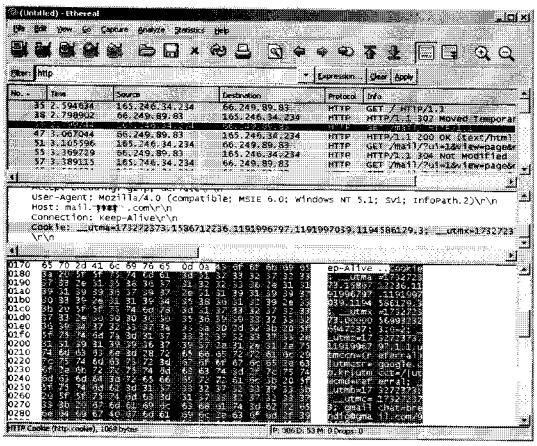
〔그림 5〕. 쿠키를 강제 입력하는 모습

〔그림 6〕. 도청한 쿠키를 다른 컴퓨터에서 재전송하여 인증된 화면

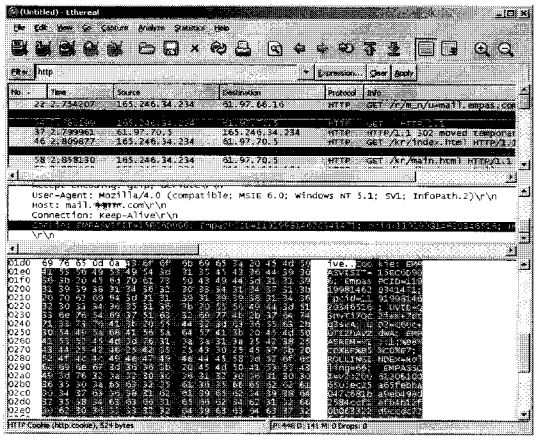
우저에 강제 입력하고 전송하여 도청한 사용자의 계정으로 접속된 화면이다. 공격이 단순히 로그인한 사용자의 HTTP request 하나를 도청하고 재전송 하는 것이지만 이를 통해 공격자는 해당 업체가 제공하는 메일, 카페, 블로그, 뉴스, 쇼핑 등 다양한 서비스의 대부분을 위장한 상태로 이용할 수 있다. 이러한 서비스들 중에는 사생활과 밀접하게 관련된 서비스들도 적지 않기 때문에 재전송을 이용한 단순한 공격이 사생활 침해로까지 번질 수 있다. 재전송 공격에 대한 취약점은 세계적으로 유명한 G사, M사, Y사, 국내에서 유명한 N사, D사, E사, C사(그림 7-12) 등 대부분의 대형 포털 및 사이트에도 적용되는 것으로 확인되었다. 이러한 사이트에서 제공하는 보안서비스는 [표 1]에서 정리해놓았다.



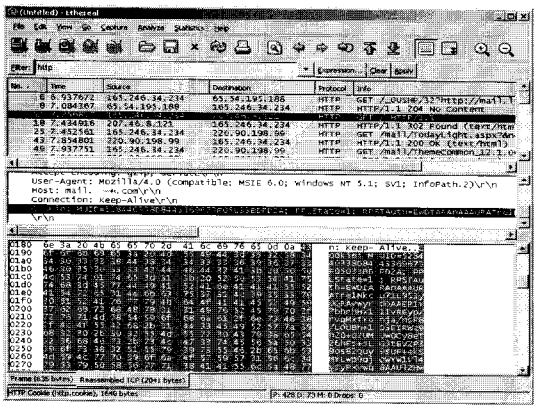
〔그림 9〕. Y사 이용자의 HTTP request 도청 화면



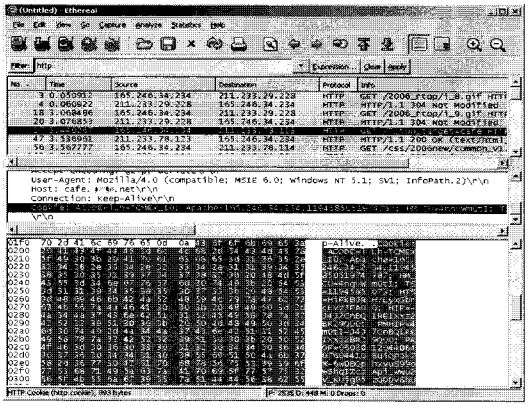
〔그림 7〕. G사 이용자의 HTTP request 도청 화면



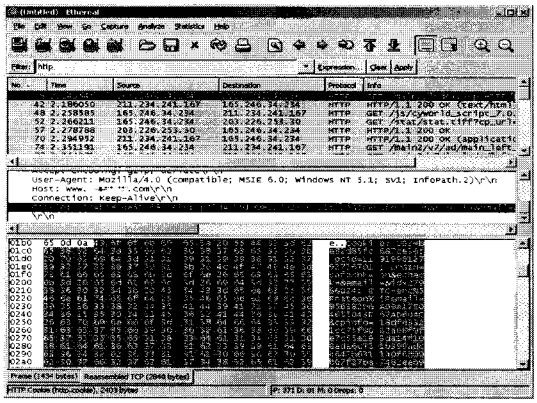
〔그림 10〕. E사 이용자의 HTTP request 도청 화면



〔그림 8〕. M사 이용자의 HTTP request 도청 화면



〔그림 11〕. D사 이용자의 HTTP request 도청 화면



〔그림 12〕. C사 이용자의 HTTP request 도청 화면

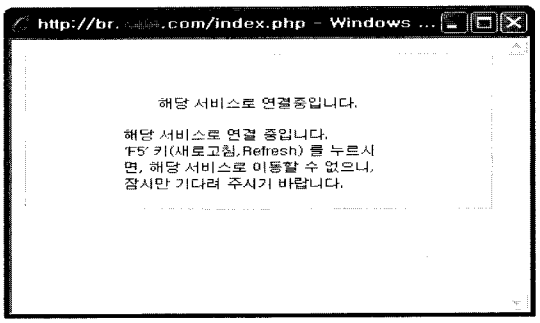
〔표 1〕. 각종 사이트에서 제공중인 보안 서비스와 제안하는 기법의 보안 서비스 비교

	ID/PW 보호	재전송 공격 보호	내용 암호화	URL 요청 인증	입력 값의 무결성 제공
G사	O				
M사	O				
Y사	O				
N사	O				
D사	O				
E사	O				
C사	O				
N메신저	O				
K방송사	O				
I대학					
제안기법	O	O	O	O	O

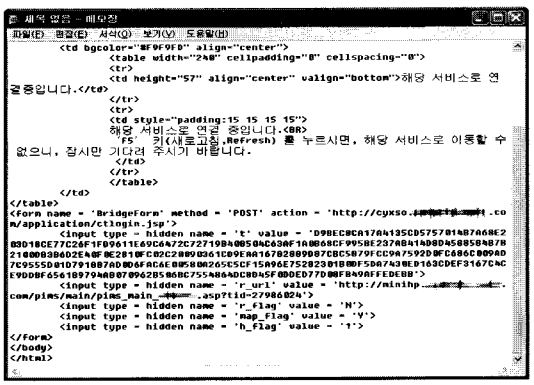
사용자의 아이디와 패스워드는 보호하고 있지만 대부분이 사용자의 세션을 보호하고 있지 않음을 알 수 있다. 특히, N사는 IP보안을 적용하여 재전송 공격에 대비하고자 하였으나 IP Spoofing이 가능한 공격자에 대해서는 여전히 재전송 공격에 취약하다.

4.2. N 메신저

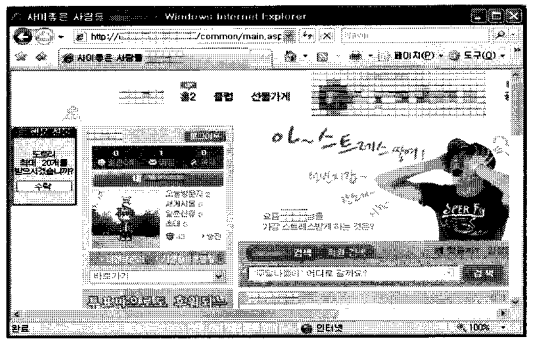
N 메신저는 현재 한국에서 가장 많이 사용되는 메신저이다. 이 메신저는 다양한 서비스와 연동되어 있어 N 메신저에 로그인한 사용자라면 별도의 로그인 단계 없이 자신의 계정으로 로그인된 상태로 이러한 서비스들을 이용할 수 있다. 이러한 서비스들 중 메신저에 내장된 서비스가 아닌 웹 서비스로 연결되는 경우에는 클라



〔그림 13〕. N 메신저와 다른 웹 서비스의 연동



〔그림 14〕. POST 형식의 변수 "t"에 저장된 SSO 토큰



〔그림 15〕. 재전송을 통해 로그인 된 화면

이언트와 웹 서비스 서버와의 인증이 추가적으로 요구되는데 웹 서비스 연동의 사용자 인증은 인증정보 전달 방식의 SSO(그림 13)이며 SSO 토큰은 [그림 14]와 같이 HTTP request의 POST를 통해 평문으로 전송되기 때문에 서비스 서버로 넘겨지는 SSO 토큰은 네트워크 스니핑으로 간단하게 얻어낼 수 있다. 다른 인증과정이 없기 때문에 얻어낸 SSO 토큰을 서비스 서버에 단순히

재전송 하는 것만으로도 공격자는 다른 사용자의 계정으로 로그인 되어 서비스를 이용할 수 있다.

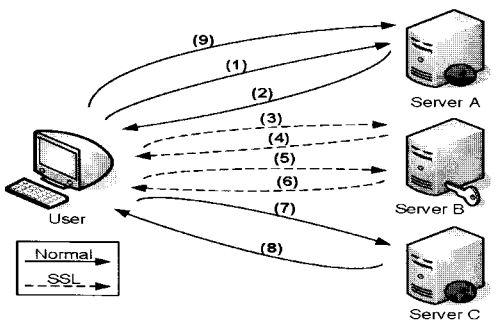
[그림 15]는 위의 과정을 이용하여 C홈페이지에 접속한 화면이다. C의 미니홈피는 국내 최다 사용자들이 찾는 개인 홈페이지 서비스이고 이 서비스는 주로 사생활과 연관되어 있다. 다양한 서비스도 포함하고 있기 때문에 공격자가 이러한 서비스들을 악용할 경우 사생활 침해뿐만 아니라 다른 보안 문제도 불러올 수 있다.

N메신저의 SSO 취약점은 SSO의 취약점 문제가 웹 환경에 국한된 것이 아니라 복합된 서비스를 제공하는 다양한 환경에서 발생할 수 있다는 것을 보여주는 좋은 예이다.

4.3. 방송사 홈페이지

모 방송사 홈페이지에서 사용된 NETS*SSO[9] 솔루션은 사용자의 아이디, 비밀번호와 SSO 토큰을 보호하기위해서 로그인 인증단계의 트래픽에 SSL을 적용하였다. 하지만 사용자가 서비스를 이용할 때는 클라이언트의 쿠키만을 확인하기 때문에 사용자 인증 단계에서 SSL을 이용하여 SSO 토큰을 보호하는 것이 의미가 없는 셈이다. 이 솔루션의 인증과정은 [그림 16]과 함께 아래에 설명하며 [그림 16]에서의 실선은 일반 트래픽, 점선은 SSL로 암호화되어 도청이 불가능한 트래픽을 뜻한다.

1. 사용자가 A서버에 로그인 폼을 요청한다.
2. 서버 A는 사용자에게 로그인 폼을 전송한다.
3. 사용자는 수신 받은 로그인 폼에 아이디와 비밀번호를 입력하고 submit을 클릭하면 사용자 인증을 요구하는 HTTP request가 SSL을 사용하는 서버 B로 안전하게 전송된다.
4. 서버 B는 사용자의 아이디와 비밀번호를 확인하여 사용자 인증에 성공하면 SSO 토큰을 생성하고 이 토큰을 폼에 넣어 사용자에게 전송한다. SSL채널을 이용하기 때문에 SSO 토큰은 기밀성을 유지한다.
5. 사용자에게 수신된 SSO 토큰을 포함하는 폼은 서버 B로 자동 submit된다.
6. 서버 B는 수신된 SSO 토큰이 유효할 경우, 쿠키를 사용자에게 전송한다. 이때 전송하는 HTTP response는 클라이언트가 서버 C로 자동으로 이동하도록 설정되어있다. 쿠키정보를 받은 클라이언트는 쿠키를 생성한다.
7. 클라이언트는 서버 C로 자동으로 이동한다. 6에서 생성된 쿠키는 쿠키 도메인에서 모두 유효하기 때문에 서버 C에도 이를 전송한다.
8. 서버 C는 클라이언트에 HTTP header에 서버 A로 이동하라는 명령이 담겨있는 HTTP response를 전송한다.
9. 클라이언트는 서버 A로 쿠키정보와 함께 HTTP request하여 인증된 상태로 서비스를 이용할 수 있다.

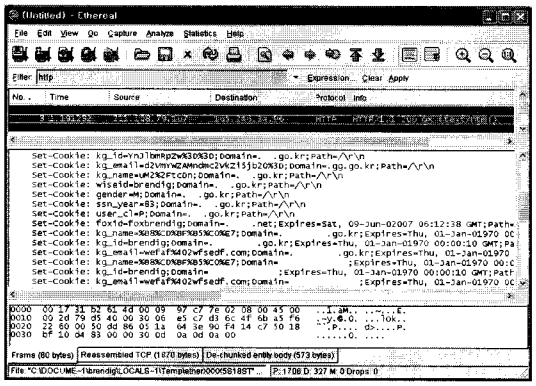


[그림 16]. 모 방송사의 홈페이지에 적용된 SSO 솔루션의 인증과정

이 솔루션의 사용자 인증과정은 SSL을 이용하여 비교적 안전하게 이루어지지만 사용자 인증을 거치고 난 후에 생성된 쿠키정보는 네트워크에 그대로 노출되어 공격자가 쉽게 얻어낼 수 있다. 사용자 인증 후 클라이언트를 신뢰하는 방법은 쿠키에만 의존하기 때문에 인증된 사용자의 HTTP request를 도청하고 HTTP request에 포함된 쿠키 값을 브라우저에 강제로 입력한 다음, 홈페이지에 접속하면 인증된 상태로 서비스를 이용할 수 있다.

4.4. 지방 자치단체 홈페이지

모 지방 자치단체의 포털 홈페이지에 적용된 Singler Sing-On[10]역시 다양한 서비스와 연동되어있다. 이 솔루션은 다수의 쿠키 도메인에 SSO이 적용되도록 하기 위해 사용자 인증 후, 적용되는 도메인이 다른 다수의 쿠키를 설정한다. 하지만 쿠키가 안전하지 않은 채널로 전달되기 때문에 다양한 서비스로 연결할 수 있는 다수



(그림 17). 다양한 쿠키 도메인에 적용되는 SSO 토큰

의 SSO 토큰들이 공격자에게 그대로 노출 될 수 있다 (그림 17).

4.5. 대학교 포털 홈페이지

모 대학교 홈페이지에 적용된 ezSSO[11]는 SSO 토큰은 [그림 18], [그림 19]와 같이 GET형식의 변수로 전달되기 때문에 스니핑에 간단하게 노출된다. ezSSO의 SSO 토큰은 사용자에 따라 고유한 값을 가지고 시간이 지나도 바뀌지 않기 때문에 사용자의 토큰이 한번만 노출되어도 언제 어디서든 이 토큰을 이용하여 대학교 홈페이지에 접근할 수 있다. 아이디/비밀번호에 기반을 두는 인증은 비밀번호가 노출되었더라도 사용자가 비밀번호를 자주 바꾸면 임시적으로 보안을 유지할 수 있지만 ezSSO에서 사용된 고정된 SSO 토큰은 사용자가 임의로 바꾸는 것이 불가능하기 때문에 사용자의 편의를 위해 사용한 SSO이 더 큰 보안 문제를 불러온 셈이다.

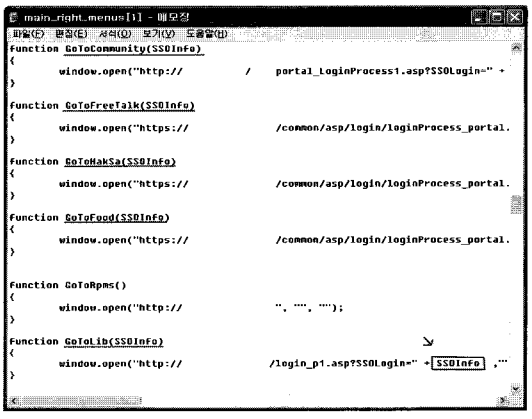
4.6. 취약점의 원인

지금까지 실제로 사용되고 있는 몇 개의 SSO 솔루션의 취약점을 분석해 보았다. 놀랍게도, 대부분의 SSO 솔루션에서 재전송 공격이 가능하였고 이 취약점은 SSO이 적용된 시스템의 규모를 가리지 않았다. 본 논문에서 시도하지 않은 스푸핑(spoofing) 공격까지 시도하면 대부분의 SSO 솔루션에서 취약점이 발생할 것으로 사료된다. 다행히도 최근의 네트워크 장비들은 스푸핑을 방지하기 위한 기법들이 적용되고 있어 공격자로부터 비교적 안전할 수 있지만 스니핑에 대해서는 근본적인 대책이 없는 상태이다. 도청을 예방하기 위해선 SSL과 같은 안전한 채널을 이용하여 인증정보를 교환해야 하지만 비용문제로 실제로는 사용되기 어려운 단점이 있다.

쿠키가 무분별하게 악용되는 것을 막기 위한 하나의 방안으로 국제 웹 표준화 기구인 World Wide Web Consortium(W3C)은 Platform for Privacy Preference (P3P)[8]를 제안하여 웹 브라우저의 설정에 따라 다른 도메인 간 쿠키 공유를 불가능 하게 하였지만 근본적인 해결방안은 되지 못한다.

웹 환경에서 사용자의 세션을 안전하게 디자인 하는 것이 어려운 이유는 웹 서버와 웹 브라우저사이의 통신 채널을 신뢰할 수 없고 신뢰할 수 없는 통신 채널 위에서 안전한 채널을 생성하기 위한 키 공유 또한 불가능

(그림 18). 고유한 값을 가지는 SSO 토큰



(그림 19). URL을 통해 전송되는 SSO 토큰

하기 때문이다. 웹 페이지 위에서 동작하는 Client-Side Script(e.g. Javascript)는 다음 페이지로 이동할 때, 이전 페이지의 변수를 기밀성을 유지한 상태로 공유할 수 없고 사용자의 세션이 유지되는 시간동안 유일하게 데이터를 유지할 수 있는 쿠키는 HTTP request/response 패킷에 매번 평문으로 송/수신되기 때문에 기밀성을 기대할 수 없다. 이와 같은 이유로 본 논문에서는 안전한 세션을 유지하는데 필요한 데이터를 웹 브라우저에서 유지하는 방법들을 알아보았다. 다음 장에서는 이러한 방법들을 이용하여 HTTP 상에서 사용자의 세션을 안전하게 유지하기 위한 방법을 알아본다.

V. 키 공유를 통한 안전한 세션 관리 기법

5.1. 웹 클라이언트 측의 안전한 세션 유지

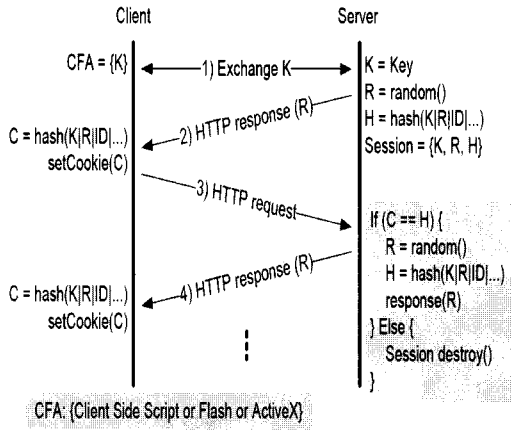
일반적으로 웹 브라우저에서 웹 페이지가 바뀌더라도 어떠한 데이터를 그대로 유지하는 방법은 쿠키가 유일하다고 알려왔다. 하지만 쿠키는 HTTP 헤더에 자동 첨부되어 평문으로 송/수신되기 때문에 쿠키에 비밀키와 같은 민감한 데이터를 저장하기에는 좋은 방법이 아니다. [7]에서는 쿠키를 안전하게 사용하기 위한 방법들에 대해서 쿠키에 IP를 저장하는 방법, 사용자의 패스워드를 해시하여 쿠키에 저장하는 방법, 전자 서명에 기반을 둔 방법, 쿠키의 무결성을 제공하기 위해 쿠키의 MAC을 저장하는 방법 등을 제시하였지만 이러한 방법들을 이용하더라도 공격자는 아이피를 속이기 위한 스푸핑(IP Spoofing)이나 offline dictionary attack등을 이용한 공격이 가능하기 때문에 사용자의 세션을 보호하기에는 여전히 부족하다. [6]에서는 쿠키의 유지시간을 짧게 설정하고 SSL을 사용하도록 요구하였지만 쿠키의 짧은 유지시간 때문에 제공하는 서비스에 제한이 생길 수 있고 모든 사용자와의 통신 채널에 SSL을 적용하는 것은 비용의 문제로 적용하기 어려운 문제가 있다.

사용자의 세션을 안전하게 유지하는 방법으로 국내의 금융권 사이트 등에서는 사용자의 브라우저에 ActiveX를 설치하여 사용자와의 키를 설정하고 중요한 데이터를 암호화 하는 방식을 택하였다. 인터넷 익스플로러만이 실행 가능한 ActiveX는 컴파일 된 형태로 배포되기 때문에 코드가 단순히 노출되는 Client Side Script에 비해서 비교적 안전하다고 볼 수 있지만 사용자의 접근 환경을 제한하므로 호환성이 좋은 방법은 아니다.

ActiveX를 사용하지 않고서 웹 브라우저에 데이터를 안전하게 유지하는 방법이 존재한다면 웹 서버와 사용자의 웹 브라우저 세션을 보호하는 것과 동시에 브라우저 호환성이 높은 프로토콜을 설계하는 것이 가능할 것이다. 웹 브라우저에 데이터를 안전하게 보관하는 데에는, 쿠키를 제외 한다면 크게 두 가지 방법이 있다. 페이지가 바뀌더라도 값을 유지하는 웹 브라우저의 특별한 DOM(Document Object Model) 변수(예를 들어, window.name[12])를 사용하는 방법이 있고, 이것이 제안하는 방법들 중 가장 유연한 방법이라고 할 수 있다. 하지만 타 사이트로 이동할 경우 이동한 사이트에서도 이 변수를 접근할 수 있기 때문에 보안상 취약하다고 할 수 있으며 인증 프로토콜 내에서 이를 해결해야 할 것이다. 또 다른 방법으로는 플래시의 LSO(Local Shared Objests)[13]를 이용하는 것이다. 플래시는 Flash Plug-In 설치를 요구하지만 이미 대부분의 브라우저(Macromedia에 따르면 98%)에서 설치되어 사용되고 있는 만큼 부담이 적으며 대부분의 브라우저를 지원하기 때문에 호환성도 높아서 다른 Plug-In에 비해서 부담이 적다고 할 수 있다. 사용 가능한 LSO의 기본 크기는 100kB이며 LSO도 웹에서의 쿠키와 같이 도메인에 따라서 데이터 접근을 제한하는 것이 가능하다.

5.2. Challenge-Response 방식의 세션 인증기법

제안하는 기법은 위에서 언급된 저장소(그림 20에서의 CFA를 이용한 저장소)들을 이용하고, 사용자가 웹 서버에 로그인을 할 때 Diffie-Hellman, RSA 등을 이용한 키 동의(Key Agreement) 또는 키 전송(Key Transport)을 통해서 웹 서버와 웹 브라우저가 키를 공유한다고 가정한다. 이러한 가정은 사용자 로그인 과정에 RSA 등을 이미 적용하고 있으므로(e.g. Naver 보안 로그인) 키를 공유하는 것은 어렵지 않을 것으로 사료된다. 공유된 키를 이용하여 웹 서버는 웹 브라우저가 HTTP request할 때마다 공유된 키에 대한 MAC (Message Authentication Code)을 함께 요구하는 것으로 HTTP request가 유효한 것인지를 검증하고, 검증된 HTTP request에 대해서만 서비스 응답하는 방법을 통해 사용자의 세션을 안전하게 유지할 수 있다. 한편, MAC을 재전송 공격으로부터 보호하기 위해서는 Challenge-Response 프로토콜을 이용해야 한다.



[그림 20]. Challenge-Response 프로토콜을 적용한 HTTP.

Challenge-Response 프로토콜을 HTTP에 적용하면 [그림 20]에서와 같이 HTTP response는 challenge (Challenge-Response의 challenge), HTTP request는 response(Challenge-Response의 response)의 형태가 된다. 아래에서 Challenge-Response 프로토콜을 적용한 HTTP를 보인다.

1. 사용자가 로그인을 할 때, Diffie-Hellman, RSA 등을 이용한 키 동의(Key Agreement) 또는 키 전송(Key Transport)을 통해서 웹 서버와 웹 브라우저가 키 K를 공유한다.
2. 웹 서버는 Random Nonce인 R을 생성하고 이것을 HTTP response에 담아 웹 브라우저에 전송한다. 웹 서버가 전송하는 HTTP response에는 클라이언트가 해시값을 생성
3. 웹 브라우저는 웹 서버가 전송한 R과 공유되어 있는 키 K를 Hash하고 그 결과값을 HTTP request에 담아 웹 서버에 전송한다.
4. 웹 서버는 자신이 전송했던 R과 공유되어 있는 키 K를 Hash한 값과 웹 브라우저가 전송한 Hash값이 같다면 웹 브라우저가 K를 가지고 있다는 것을 확인한 것으로 인증에 성공한 것이고 비교한 값이 다르다면 인증에 실패한다.

위와 같은 Challenge-Response방식의 프로토콜을 이용하면 웹 서버는 Challenge R에 대한 MAC을 요구하고 이를 검증하여 요청한 서비스에 대해서 응답하기 때문에 재전송 공격으로부터 안전하다.

5.3. 제안하는 기법의 보안 서비스

4장의 예제에서 보였듯이 로그인과정에서 SSL 또는 공개키를 이용한 프로토콜을 개발하고 적용하여 사용자의 아이디와 패스워드는 보호하고 있지만 그 이후의 사용자의 세션은 단순히 쿠키에만 의존하는 것이 일반적이다. 쿠키만으로 사용자의 세션을 인증하면 재전송 공격에 무방비 하지만 제안하는 기법은 IP Spoofing이 가능한 공격자에 대해서도 사용자의 세션을 보호할 수 있다. 제안하는 기법의 가정으로부터 사용자의 아이디와 패스워드는 도청으로부터 보호되고 Challenge- Response 방식의 프로토콜을 이용한 MAC 검증 후 서비스를 응답하기 때문에 재전송 공격으로부터 안전하다. 또한 공유된 키를 응용하면 다음과 같은 보안서비스들이 가능하다.

- 사용자에게 서비스되는 내용 중, 민감한 내용을 암호화하고 이에 대한 MAC도 함께 전송하는 것으로 서비스되는 내용에 대한 기밀성 및 무결성 제공
- HTTP request를 전송하기 직전에 사용자가 요청하는 URL을 MAC에 포함하여 계산하고 전송하는 것으로 요청 URL에 대한 인증
- 사용자가 입력한 값들을 공유한 키를 이용하여 암호화하고 이 암호문에 대한 MAC을 함께 전송하여 서버로 전송되는 내용에 대한 기밀성 및 무결성 제공

[표 1]에서는 각종 사이트에서 제공하는 SSO솔루션과 제안하는 기법의 보안성을 비교하였다. 이미 사용 중인 SSO솔루션들은 아이디와 패스워드만을 보호하고 있으며 그 외의 보안서비스는 제공하지 않고 있는 것으로 밝혀졌다. 웹을 활용한 서비스와 이러한 서비스들의 이용도는 날로 증가하는 반면에 보안 서비스만은 그러한 증가속도에 비해 한참 뒤쳐져 있다는 것으로 풀이된다. 제안하는 기법은 모든 내용이 의무적으로 암호화되는 SSL과는 다르게, 민감한 정보만을 선택적으로 암호화하는 것이 가능하기 때문에 웹 서버로 하여금 암호화에 소요되는 비용 또한 가볍게 유지할 수 있어 웹 서비스에 보안을 적용하기 위한 부담을 크게 줄일 수 있다. 제안하는 기법이 웹 브라우저 호환성이 높고 다양한 보안서비스를 제공할 수 있다는 것을 고려하면, 제안하는 기법은 서비스 접근성에 민감한 웹 환경의 보안연구에 다양하게 활용될 수 있을 것으로 예상한다.

Ⅵ. 결 론

SSO은 그 자체가 하나의 사용자 인증을 의미하는 만큼 보안성이 충분이 고려된 후 제작되어야 한다. 하지만 현재 서비스되고 있는 SSO 솔루션들은 단순한 재전송 공격에 취약하며 홈페이지 간의 연동을 위한 SSO뿐만 아니라 메신저와 다른 웹 서비스들을 연동할 때 쓰이는 SSO 또한 취약하였다. 이처럼 SSO 솔루션들이 취약한 쿠키만을 이용하여 계속 적용된다면 SSO은 보안에서의 또 다른 불안요소로 자리 잡을 것으로 보인다.

본 논문에서는 재전송 공격에 취약한 SSO 솔루션들과 취약점의 원인을 분석하고 이를 보완하기 위해서 사용자의 세션을 안전하게 유지하는 기법을 제시하였다. 제안하는 프로토콜은 사용자의 세션을 안전하게 보호할 수 있지만 실제 서비스에 적용되기에는 사용자의 행동패턴 또는 네트워크 환경에 따라 공유된 R이 비동기화 될 가능성이 있기 때문에 제안하는 프로토콜이 정상적으로 동작하지 않을 수 있다. 이에 따라 제안하는 프로토콜을 개선하여 stateless 환경을 고려한 Fault-Tolerant 프로토콜을 디자인 하는 것은 좋은 연구거리가 될 것이다.

앞으로의 유비쿼터스 환경에서는 각종 서비스가 통합되고 서비스 접근 매체 또한 다양해지는 만큼 SSO은 현재보다 더욱 자주 쓰일 것이라 예상된다. 따라서 SSO을 단순히 업무의 효율과 사용자의 편의에 목적을 두기 전에 보안을 고려하여 디자인하고 SSO이 적용되는 시스템 또한 보안상에 빈틈이 없어야 사용자에게 신뢰할 수 있는 서비스를 제공할 수 있을 것이다.

참고문헌

[1] Jan De Clercq, "Single Sign-On Architectures", Proceedings of the International Conference on Infrastructure Security, pp. 40-58, 2002

[2] Gary Ellison, Jeff Hodges, Susan Landau, "Security and Privacy Concerns of Internet Single Sign-On", Liberty v1.6, September 2002

[3] B. Pfitzmann, B. Waidner, "Token-based web Single Signon with Enabled Clients", *IBM Research Report RZ 3458 (#93844)*, November (2002)

[4] Andreas Pashalidis, Chris J. Mitchell, "A Taxonomy of Single Sign-On Systems", Proceedings of the 8th Australasian Conference, LNCS 2727, 2003

[5] Eric Rescorla, "SSL and TLS", *Addison-Wesley,* Reading, Massachusetts, 2001

[6] V. Samar, "Single Sign-On Using Cookies for Web Applications," *WET ICE'99*, June 1999.

[7] J Park and R. Sandhu, "Secure Cookies on the Web," *IEEE Internet Computing*, Aug. 2000.

[8] http://www.w3.org/P3P/

[9] http://www.nets.co.kr

[10] http://www.sebitsoft.com

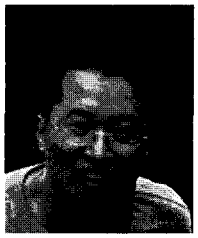
[11] http://www.kaoni.com

[12] http://www.boutell.com/newfaq/creating/scriptpass.html

[13] http://www.adobe.com/support/flash/action_scripts/local_shared_object/

〈著者紹介〉

맹 영 재 (YoungJae Maeng) 정회원
2006년 8월 : 인하대학교 컴퓨터공학과 졸업
2006년 9월~현재 : 인하대학교 정보통신대학원 석사
<관심분야> 인터넷 보안, 네트워크 보안



양 대 헌 (DaeHun Nyang) 정회원
1994년 2월 : 한국과학기술원 과학기술대학 전기 및 전자공학과 졸업
1996년 2월 : 연세대학교 컴퓨터과학과 석사
2000년 8월 : 연세대학교 컴퓨터과학과 박사
2000년 9월~2003년 2월 : 한국전자통신연구원 정보보호연구본부 선임연구원
2003년 2월~현재 : 인하대학교 정보통신대학원 조교수
<관심분야> 암호이론, 암호프로토콜, 인증프로토콜, 무선 인터넷 보안