Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.1, Feb. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.1.5

OAuth를 이용한 로그아웃 문제로 인한 취약점 방지 기법에 대한 연구*

김 진 욱,[1†] 박 정 수,[2] 응 웬 부 렁,[2] 정 수 환[2‡]
[1]디지캡, [2]숭실대학교

A Study on Vulnerability Prevention Mechanism Due to Logout Problem Using OAuth*

Jinouk Kim,[1†] Jungsoo Park,[2] Long Nguyen-Vu,[2] Souhwan Jung[2‡]
[1]DigiCAP, [2]Soongsil University

요 약

많은 웹 서비스가 OAuth 프로토콜을 이용하여 리소스 서버로부터 프로필 정보를 전달받아 사용자를 로그인시켜주는 형태로 제공되어 추가적인 회원가입의 불편함을 줄여주고 있다. 하지만, 작업을 종료한 사용자가 클라이언트 웹 서비스를 로그아웃하더라도 리소스 서버가 로그인 상태로 남아 사용자의 개인정보 및 리소스 서버 자원 유출에 대한 문제가 발생할 수 있다. 본 논문에서는 OAuth 프로토콜을 통해 로그인한 사용자가 클라이언트에서 로그아웃할 경우, 리소스 서버의 로그인 상태에 관한 알림을 통하여 사용자가 리소스 서버의 로그아웃 여부를 판단할 수 있도록 설계 및 구현하였다. 본 논문에서 제안하는 방법을 활용하여 OAuth 프로토콜을 통해 로그인한 사용자가 도서관, 백화점, 인쇄소와 같이 여러 사용자가 이용하는 공용 PC 환경에서 리소스 서버에 로그아웃 여부를 확인하지 않아 발생할 수 있는 정보 유출 문제를 줄이고 리소스 서버의 자원을 보호할 수 있다.

ABSTRACT

Many web services which use OAuth Protocol offer users to log in using their personal profile information given by resource servers. This method reduces the inconvenience of the users to register for new membership. However, at the time a user finishes using OAuth client web service, even if he logs out of the client web service, the resource server remained in the login state may cause the problem of leaking personal information. In this paper, we propose a solution to mitigate the threat by providing an additional security behavior check: when a user requests to log out of the Web Client service, he or she can make decision whether or not to log out of the resource server via confirmation notification regarding the state of the resource server. By utilizing the proposed method, users who log in through the OAuth Protocol in the public PC environment like department stores, libraries, printing companies, etc. can prevent the leakage of personal information issues that may arise from forgetting to check the other OAuth related services. To verify our study, we implement a Client Web Service that uses OAuth 2.0 protocol and integrate it with our security behavior check. The result shows that with this additional function, users will have a better security when dealing with resource authorization in OAuth 2.0 implementation.

Keywords: OAuth, Access Token, Identity Provider, Threat, Data Privacy

Received(06. 22. 2016), Modified(12. 02. 2016), Accepted(02. 01. 2017)

* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학 ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2016-H8501-16-1008) 또한, 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.R-20160222-002755, 맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안 기술 개발)

† 주저자, ouk92@ssu.ac.kr
‡ 교신저자, souhwanj@ssu.ac.kr(Corresponding author)

I. 서 론

인터넷 기술의 발전으로 다양한 형태의 애플리케이션이 등장하였고 그중 일부 애플리케이션은 외부 서비스 API (Application Programming Interface)를 이용하여 인증을 제공하는 형태로 서비스되고 있다[1,2]. 이러한 애플리케이션이 사용자 인증을 위한 외부 서비스 API를 안전하게 이용하기 위해 사용하고 있는 OAuth (Open Authorization) 프로토콜은 다양한 웹 환경에서 권한인가를 가능하게 해주는 표준으로 IETF (Internet Engineering Task Force)에서 논의되고 있다[3]. 초기, OAuth 프로토콜은 서비스를 제공하는 애플리케이션 (Client)과 사용자의 정보를 포함하고 있는 리소스 서버 (Resource Server) 간 사용자의 자격 증명 공유 없이 리소스 서버에 저장되어 있는 사용자 자원에 접근 가능한 권한을 인가하기 위하여 제안되었다. 이러한 OAuth 프로토콜을 이용하여 클라이언트 애플리케이션은 리소스 서버 API를 이용할 수 있으며 게시물 작성, 내용 공유 등의 작업이 가능하고 리소스 서버에 저장된 사진, 주소록 등의 정보에 접근할 수 있다.

하지만, 많은 클라이언트는 OAuth 2.0 프로토콜을 이용하여 페이스북, 구글, 네이버와 같은 많은 양의 사용자 정보를 보유한 IdP (Identity Provider)로부터 프로필 정보를 전달받아 사용자를 로그인시켜주는 형태로 서비스하고 있다[2,4,5]. 사용자는 클라이언트에서 로그인을 수행할 IdP를 선택하고 리다이렉트를 통하여 IdP의 로그인 페이지에서 사용자의 계정 정보를 이용한 로그인을 수행한다. 로그인 후, 클라이언트에서 요청하는 프로필 정보 (이메일, 이름 등)의 제공을 수락함으로써 클라이언트에 로그인할 수 있으며 클라이언트에서 정한 권한에 맞는 제한적인 서비스를 이용할 수 있다.

OAuth 프로토콜을 사용자 인증으로 사용하는 것에 대하여 OAuth 커뮤니티와 OAuth 2.0 프로토콜 에디터 Eran Hammer의 우려가 있었지만[6,7], 현재 OAuth 2.0 프로토콜은 다양한 서비스 환경에서 적용 가능한 인증 솔루션으로 사용되고 있다. 이러한 OAuth를 사용자 인증에 사용하는 서비스의 증가와 함께 사용의 안전성을 분석하기 위한 연구들이 수행되었고 다양한 취약점이 발견되었다[8,9]. IETF OAuth WG은 앞선 연구를 통해 발견된 취약점에 대한 보안 고려사항을 OAuth RFC 문서에 권고하였다[3,10].

하지만, OAuth 2.0 프로토콜을 도입한 많은 웹 서비스는 OAuth 2.0 프로토콜의 최종 스펙과 보안 고려사항이 완성되기 전에 서비스를 제공하여 표준 보안 고려사항을 제대로 충족하지 못하는 경우가 발생하고 있다[11]. 또한, 최근 OAuth 2.0 프로토콜 표준에 권고된 보안 고려사항 외에도 새로운 취약점이 발견되어 현재 OAuth WG에서 해결 방안을 논의 중이다[12,13].

본 논문은 OAuth 2.0 프로토콜이 구현된 클라이언트 웹 서비스에 로그인한 사용자의 브라우저에서 로그인 과정 후 보이지 않는 리소스 서버가 지속적으로 로그인 되어있음을 확인하였다. 사용자가 리소스 서버에 로그인한 주요 목적은 클라이언트를 이용하는 것으로 사용자가 리소스 서버 상태를 인지하기 어렵다. 실제 도서관, 백화점, 인쇄소 등과 같이 많은 사용자가 사용하는 공용 PC 환경에서 OAuth 2.0 프로토콜을 이용한 로그인을 통해 클라이언트 웹 서비스를 이용할 때, 사용자 부주의로 인하여 로그인 상태의 리소스 서버를 남겨두고 자리를 떠난다면 다음 사용자에 의해 사용자 개인 정보 및 문서, 사진 등과 같은 사용자의 중요한 자원이 탈취될 수 있다. 본 논문에서는 리소스 서버의 로그인 상태로 인해 발생할 수 있는 위협을 소개하고 이러한 위협을 해결하기 위하여 클라이언트 사용자에게 메시지를 통한 알림으로 로그아웃 여부를 판단할 수 있게 제안하였으며 시스템을 구현하였다. 본 시스템은 로그인된 사용자에게 간단한 게시판을 제공할 수 있는 클라이언트 웹 서비스를 구축하였으며 Google에 클라이언트 등록 및 OAuth 2.0 라이브러리 적용을 통하여 동작이 가능하도록 하였다. 또한, 클라이언트 웹 서비스 이용이 끝난 사용자가 로그아웃을 누를 경우 팝업 형태의 메시지를 통하여 리소스 서버의 로그아웃을 제어할 수 있다.

본 논문의 구성은 다음과 같다. 뒤이어 2장에서는 OAuth 2.0 프로토콜의 네 가지 형태와 특징에 대한 소개와 OAuth 2.0 프로토콜을 이용한 웹 기반 인증에 대하여 소개한다. 그리고 3장에서 OAuth 2.0 프로토콜을 통한 클라이언트 웹 서비스 로그인 시 발생할 수 있는 위협을 소개하고, 그 다음 4장에서 위협에 대한 해결방안 논의 및 구현 결과에 대한 설명을 한다. 끝으로 5장에서 결론을 맺는다.

II. 관련 연구

2.1 OAuth 2.0 프로토콜

OAuth 2.0 프로토콜은 서비스 간 계정 정보공유 없이 사용자의 리소스 서버 자원에 접근할 수 있는 권한인가를 지원하는 하는 표준이다[3]. 이러한 OAuth 2.0 프로토콜은 권한인가를 위한 네 가지 타입의 Authorization Grant를 지원하며, 이 외에도 추가로 정의해서 사용할 수 있도록 하였다. 다음의 내용에서 네 가지 Authorization Grant의 특징을 설명한다.

2.1.1 Authorization Code

Confidential 클라이언트가 사용하는 권한인가 방식으로 기밀성이 필요한 클라이언트 서비스에서 주로 이용된다. 본 방식은 권한인가 서버에서 사용자 인증 및 요청하는 권한에 대한 수락 후, 클라이언트에게 Authorization Code를 발급한다. 클라이언트는 발급받은 Authorization Code를 권한인가 서버에 전달과 함께 client_id/client_secret을 이용한 클라이언트 인증을 수행한다. 클라이언트가 제공한 client_id/client_secrect의 값이 올바른 경우, 클라이언트에게 Access Token을 발급하여 리소스 서버의 API를 이용할 수 있도록 한다. 본 방식은 OAuth 2.0 Protocol이 지원하는 네 가지 타입 중 가장 널리 사용되는 방식이다.

2.1.2 Implicit

Public 클라이언트가 사용하는 권한인가 방식으로 브라우저 기반의 서비스나 모바일 서비스에서 이용된다. 이 방식은 권한인가 서버에서 로그인을 통한 사용자 인증 후 클라이언트의 인증 절차 없이 클라이언트의 Return URL 값에 해당하는 경로로 Access Token을 바로 발급한다.

2.1.3 Resource Owner Password Credentials

클라이언트에 사용자의 아이디/패스워드와 같은 계정 정보를 저장하고, 이러한 정보를 이용하여 Access Token을 요청하는 방식이다. 클라이언트를 신뢰할 수 없을 때 이용하는 것은 바람직하지 않으며, API 서비스의 공식 애플리케이션이나 신뢰할 수 있는 클라이언트에 한해서 사용한다.

2.1.4 Client Credentials

클라이언트가 권한인가 서버에서 클라이언트의 client_id, client_secret, return URL 등과 같은 자격 증명 정보를 이용하여 클라이언트 인증을 수행하고 Access Token을 발급받는 방식이다. Client 자격 증명 방식은 사용자와의 상호작용 없이 Background에서 동작하며, 클라이언트의 방문자 수 조회 등과 같이 클라이언트에 대한 작업을 수행할 때 이용할 수 있다.

2.2 OAuth 2.0을 이용한 웹 기반의 인증

OAuth 프로토콜은 서비스 간 계정정보 공유 없이 자원에 접근할 수 있는 권한인가를 위하여 제안되었다. 하지만 최근 많은 웹 서비스는 사용자 인증을 위하여 OAuth 2.0 프로토콜을 기반으로 SSO (Single Sign-On) 기능을 이용하고 있다[4,5,8,9]. OAuth 2.0 프로토콜을 사용자 인증 및 권한인가에 사용할 경우의 프로세스는 [Fig. 1]과 같다. OAuth 프로토콜을 사용자 인증 및 권한인가로 사용할 때, 두 가지 모두 공통 프로세스를 거치며 Access Token 발급 과정 이후 차이가 있다. 권한인가를 위해 사용될 경우, 클라이언트는 Access

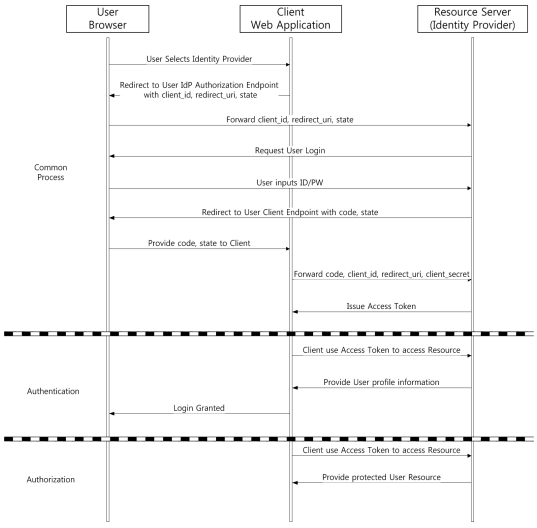

Fig. 1. OAuth 2.0 Protocol Authentication & Authorization Process

Token을 이용해 리소스 서버 자원 API를 이용하여 사진 및 주소록 다운로드, 게시물 작성 및 공유 등의 작업이 가능하다. 반면 사용자를 인증하기 위해 사용할 경우, 리소스 서버 사용자 정보를 조회한 뒤 프로필 정보를 이용하여 로그인시켜준다. OAuth 2.0 프로토콜 기반의 SSO 기능을 이용하는 클라이언트는 다음과 같은 이점이 있다[14].

① 추가적인 회원가입 과정 없이 리소스 서비스의 회원이 클라이언트의 회원이 될 수 있음
② 클라이언트는 복잡한 개인정보를 관리 할 필요가 없음
③ 계정정보 분실, 사용자 문의 등 회원관리를 위한 인증 비용 (SMS, ARS 등)은 리소스 서버가 부담함

III. 사용상의 위협

본 장에서는 웹 서비스를 이용한 서비스 이용 종료 후 발생할 수 있는 위협을 설명한다[15]. 사용상 발생 가능한 위협을 설명하기 위해 OAuth 프로토콜을 이용하여 클라이언트 웹 서비스에 로그인하는 사용자의 관점에 관해 설명한다. 그러고 나서 도서관, 백화점, 인쇄소와 같이 공용 PC 환경에서 발생할 수 있는 재로그인 문제와 리소스 서버 자원 노출 문제에 관하여 설명한다.

3.1 사용자의 관점

사용자는 웹 서비스 이용을 위해 회원가입을 통해 생성한 계정 정보로 로그인 과정이 요구된다. 하지만, 최근 많은 클라이언트 웹 서비스는 회원가입이 필요한 일반적인 로그인과 추가적인 등록 없이 다른 서비스의 계정 정보를 이용하여 클라이언트 웹 서비스에 로그인할 수 있는 방법을 지원한다. 클라이언트 웹 서비스에 등록을 원하지 않는 사용자는 서비스가 지원하는 IdP 중 원하는 것을 선택한다. 그리고 사용자의 브라우저는 IdP 로그인 페이지로 이동하며 사용자가 로그인할 수 있다. 사용자는 IdP에서 로그인 후 클라이언트의 프로필 정보 조회 요청을 수락함으로써 IdP에 프로필을 클라이언트 웹 서비스로 전송한다. 이러한 사용자의 권한 허가 이후 사용자의 브라우저는 클라이언트 웹 서비스로 이동하며 클라이언트 웹 서비스에 로그인된다. 이러한 과정을 통하여 클라이언트 웹 서비스는 회원가입 등의 추가적인 등록과정 없이 IdP의 계정 정보를 이용하여 로그인한 사용자에게 서비스를 제공한다.

3.2 발생 가능한 위협

사용자는 OAuth 프로토콜을 이용하여 등록절차 없이 클라이언트 웹 서비스의 서비스를 받을 수 있다. 그리고 작업을 종료한 사용자의 로그아웃 요청으로 클라이언트 웹 서비스는 토큰 폐기 요청을 함으로써 Access Token을 폐기할 수 있다[16].

작업을 종료한 사용자는 클라이언트 서비스 로그아웃 후 PC를 떠날 수 있지만, 사용자가 클라이언트 웹 서비스를 로그아웃하더라도 리소스 서버는 로그인 상태로 남아있다. 처음 리다이렉트 및 팝업 등의 브라우저 이동으로 로그인 페이지에서 로그인한 사용자는 로그인 후 브라우저에서 리소스 서버에 대한 정보를 확인할 수 없으므로 로그인 상태로 남아 있다는 사실을 파악하기 어렵다.

공용 PC 환경에서 사용자가 리소스 서버를 로그아웃하지 않고 자리를 떠난다면, [Fig. 2]와 같은 문제가 발생할 수 있다. 다음 사용자가 클라이언트 웹 서비스를 이용하기 위하여 같은 IdP로의 인증을 희망할 경우, 로그인된 리소스 서버 상태 때문에 다음 사용자는 리소스 서버의 로그인 절차 없이 클라이언트 웹 서비스에 로그인할 수 있다. 그리고 기존 사용자의 클라이언트 웹 서비스 히스토리 등의 이용 정보를 탈취할 수 있으며, 클라이언트 웹 서비스를 이용하는 것이 아닌 리소스 서버를 이용할 목적으로 리소스 서버 웹 페이지에 방문할 경우, [Fig. 3]과 같

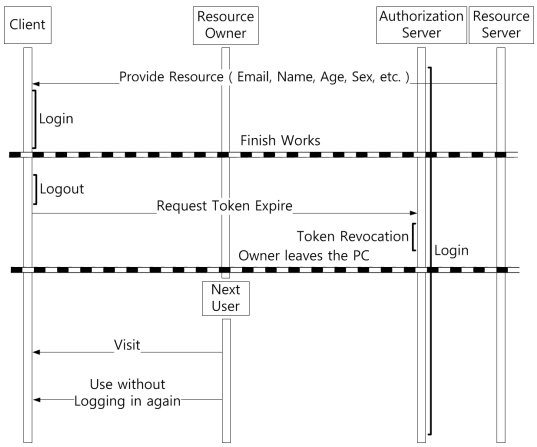

Fig. 2. Next User can access Client without authentication

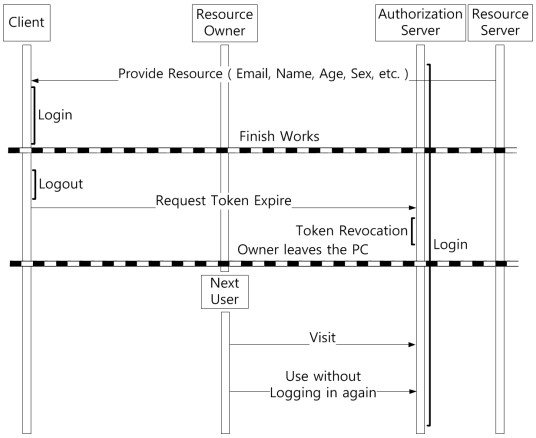

Fig. 3. Next User can access Resource Server

이 이미 로그인 되어있는 리소스 서버의 기존 사용자 정보와 중요 자원을 탈취당할 수 있다.

IV. 제안하는 시스템

본 논문에서는 클라이언트 웹 서비스의 로그아웃 후, 로그인 상태의 리소스 서버로 인하여 클라이언트 웹 서비스 재로그인 및 개인정보 탈취 문제 발생 위협을 해결하기 위하여 알림 기법을 제안한다. 이러한 내용은 OAuth 프로토콜의 취약점 및 안전성에 제안되어 있지 않은 새로운 취약점으로 추후 클라이언트 측에서 개발, 적용이 필요한 내용이다. 사용자의 로그아웃 요청 시 리다이렉트 및 팝업 형태의 정보를 알림으로써 리소스 서버의 상태를 사용자가 직접 판단할 수 있도록 하였다. 본 장에서는 사용자의 판단 없이 리소스 서버를 자동 로그아웃시켰을 때 발생 가능한 문제에 대해 논의한 후, 자동 로그아웃이 아닌 사용자에게 리소스 서버의 로그인 상태 확인을 촉구시키며 선택적으로 리소스 서버를 로그아웃시킬 수 있는 방법에 대하여 제안한다.

4.1 리소스 서버 자동 로그아웃 시나리오

본 논문에서 제시한 위협을 해결하기 위하여 사용자의 클라이언트 웹 서비스 로그아웃 요청으로 리소스 서버를 함께 로그아웃시키는 방법을 고안할 수 있다. 클라이언트 서비스에서 지원하는 OAuth 프로토콜을 통하여 사용자가 리소스 서버를 로그인한 주요 목적은 클라이언트 웹 서비스에 로그인하기 위한 것이며, 사용자는 리소스 서버가 로그인 상태로 남는 것을 원하지 않을 수 있다. 따라서 클라이언트를 통하여 리소스 서버로 접근한 경우, 특정 값을 포함하여 Access Token을 발급하고 클라이언트의 로그아웃으로 인한 Access Token 폐기 요청 시 Token의 값을 확인하여 리소스 서버를 자동으로 로그아웃시킬 수 있다. 하지만, 리소스 서버를 자동으로 로그아웃시킬 경우 사용자가 다음과 같은 두 가지 경우에서 불편함을 겪을 수 있다.

4.1.1 여러 클라이언트 웹 서비스 이용

사용자는 하나의 리소스 서버 계정 정보로 여러 개의 클라이언트 웹 서비스를 이용할 수 있다. 〔Fig. 4〕에서 클라이언트 웹 서비스 1과 2는 리소스 서버로부터 받아온 사용자의 프로필 정보를 이용하여 사용자를 로그인시켜주는 클라이언트다. 그리고 클라이언트 3은 리소스 서버에 글을 게시할 수 있는 API 접근권한을 부여받아 게시물 작성 및 공유를 할 수 있다.

클라이언트 웹 서비스 1의 로그아웃으로 인하여 자동 로그아웃될 경우 클라이언트 2는 이미 받아온 프로필 정보를 이용하여 사용자를 로그인시킨 것으로 서비스 이용이 가능하다. 하지만, 클라이언트 3은 리

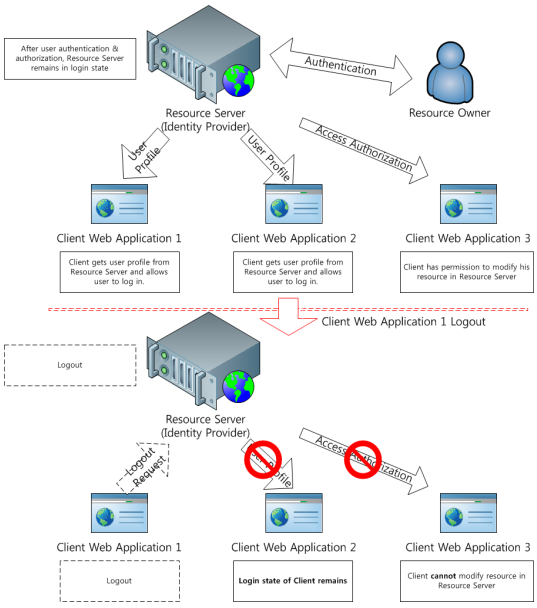

Fig. 4. Client 3 becomes unused when user logs out of Client 1

소스 서버 로그아웃으로 더는 게시물을 업로드 할 수 없다. 따라서 클라이언트 웹 서비스 3을 이용에 지장이 있으며 게시물 업로드 등의 클라이언트 웹 서비스 3이 지원하는 서비스를 받기 위해 리소스 서버 재로그인 절차가 필요하다.

4.1.2 리소스 서버 이용

사용자는 클라이언트와 함께 리소스 서버를 이용할 수 있으며 클라이언트 이용종료 후에도 리소스 서버 이용을 계획할 수 있다. 하지만, 〔Fig. 5〕와 같이 클라이언트 로그아웃으로 인하여 리소스 서버가 자동으로 로그아웃될 경우, 사용자는 리소스 서버 이용에 영향을 받아 재 로그인이 필요하다.

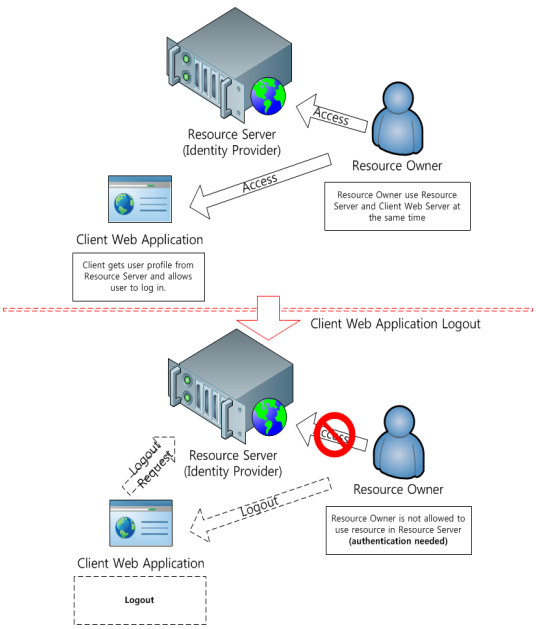

Fig. 5. User needs to re-log in to use resource in Resource Server

4.2 사용자 알림을 통한 로그아웃

사용자의 클라이언트 로그아웃으로 인하여 리소스 서버가 자동 로그아웃될 경우 앞서 소개한 두 가지 상황에서 문제가 발생할 수 있다. 이러한 인터넷 사용자의 계획과 상황은 시스템이 판단할 수 없다. 따라서 클라이언트 로그아웃 이후 리소스 서버를 로그아웃할지 계속 이용할지에 대한 문제는 사용자가 판단하는 것이 효율적이다.

본 논문에서는 OAuth 프로토콜을 통해 로그인한 사용자가 클라이언트를 로그아웃할 경우, 〔Fig. 6〕과 같이 클라이언트 서비스가 사용자에게 리소스 서버의 상태를 환기할 수 있는 알림을 통하여 위협을 예방할 것을 제안한다.

〔Fig. 6〕에서 사용자의 클라이언트 작업이 종료되었을 경우, 사용자는 클라이언트에서 로그아웃을 요청한다. 그리고 클라이언트는 인증 서버에 토큰을 파기 요청과 동시에 사용자에게 리소스 서버의 로그인 상태 종료 여부를 사용자에게 요청하여 응답을 전달받아 로그아웃 여부를 결정하게 된다. 리소스 서버에 대한 알림을 받은 사용자는 상황을 고려하여 리소스 서버를 로그인 상태로 남기고 클라이언트 서비스만 로그아웃할 수 있다. 그리고 사용자가 리소스 서버의 이용을 원하지 않을 경우 잊지 않고 리소스 서버의 로그인 상태를 확인할 수 있으며 리소스 서버가 로그아웃 API를 제공할 경우, 클라이언트 웹 서비스에서 리소스 서버를 로그아웃시킬 수 있다.

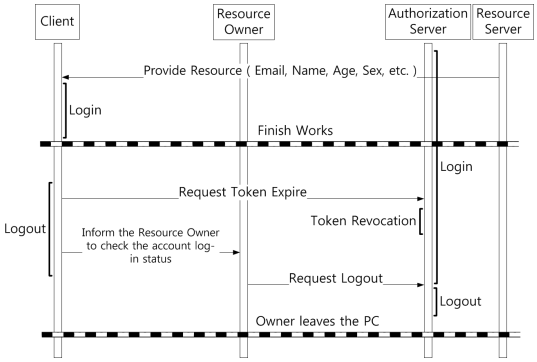

Fig. 6. Notification to User after Client Logout

4.3 제안하는 시스템 구현

본 논문에서는 OAuth 2.0 프로토콜을 통한 로그인으로 클라이언트 웹 서비스를 이용한 사용자가 로그아웃을 요청할 때, 사용자에게 알림을 제공함으로써 리소스 서버의 상태를 판단할 수 있게 하였다. 클라이언트 웹 서비스의 사용자가 로그아웃 요청을 할 경우의 과정을 〔Fig. 7〕과 함께 설명한다.

〔Fig. 7〕에서는 사용자가 로그아웃을 하게 될 경우 클라이언트 측에서는 OAuth를 이용하여 로그인한 유저인지 판단을 하게 되고 맞으면 리소스 서버를

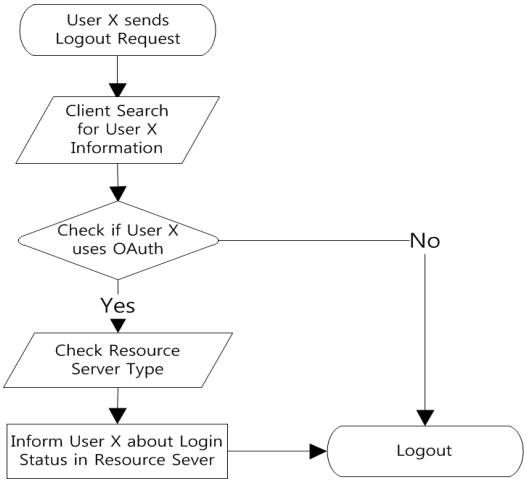

Fig. 7. Notification Flow Chart

확인하여 리소스 서버 로그아웃에 대한 요청을 하게 된다. 만약 일반 로그인 사용자라면 바로 로그아웃을 하게 처리한다. 다음 내용에서 구현의 결과를 그림과 함께 설명한다.

① 〔Fig. 8〕은 본 논문에서 제안하는 OAuth 프로토콜을 테스트하기 위해 구현한 별도 클라이언트 웹 서비스의 메인 화면이다. 해당 클라이언트 웹 서비스를 이용하기 위하여 로그인이 필요하며, 사용자는 "Login with Google" 버튼을 누름으로써 Google의 로그인 페이지로 이동하여 로그인을 수행하게 된다.

② 〔Fig. 9〕는 Google 계정을 통한 로그인이 완료된 상태이다. 그림 상단에 Google로부터 넘겨받은 사용자 정보를 확인할 수 있다. 본 서비스에서 제공하는 두 메뉴에 접근할 수 있으며 사용 종료를 원하면 "Logout" 버튼으로 로그아웃할 수 있다.

③ 〔Fig. 10〕은 사용자가 클라이언트 웹 서비스의 로그아웃을 요청한 경우의 모습이다. 로그아웃 요청에 알림 메시지가 팝업되며 사용자의 계정 정보와

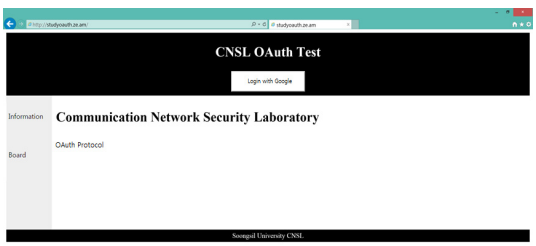

Fig. 8. Client Mainpage

Fig. 9. Client after login

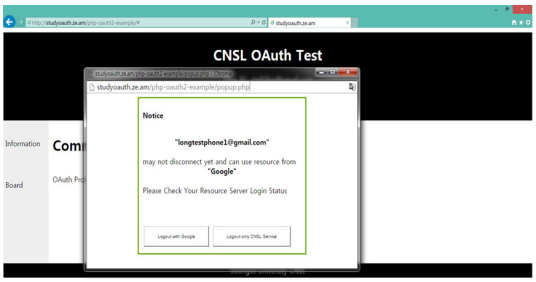

Fig. 10. After User requests Logout, Notification will be delivered to User

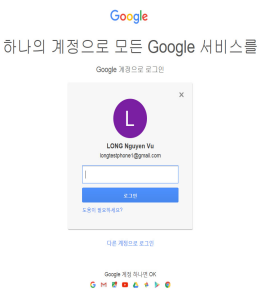

Fig. 11. Identity Provider logs out

리소스 서버 정보가 나타난다. 그리고 사용자는 클라이언트 웹 서비스만 로그아웃할지, 구글 서비스도 함께 로그아웃할지 선택할 수 있다.

④ 〔Fig. 11〕은 사용자가 구글 서비스도 함께 로그아웃할 경우의 모습이다. 구글에서 제공하는 로그아웃 API를 통해 구글도 함께 로그아웃할 수 있으며 사용자는 안전하게 자리를 떠날 수 있다.

4.4 제안하는 시스템의 장점

제안하는 시스템의 경우, 기존의 OAuth의 취약점에서 분석하지 못한 새로운 취약점에 대한 대응책

Table 1. Compare OAuth and Proposed Scheme

	OAuth	Proposed Scheme
Client Logout	○	○
Resource Server Logout	Manual	Auto Logout after user's Judgment
Resource Server Logout Condition Check	Manual	Auto

을 제시하였다. OAuth 프로토콜을 사용자 인증으로 활용하였을 경우, 클라이언트 서비스의 사용을 종료한 사용자가 로그아웃을 요청하면 클라이언트는 로그아웃 되지만 개인정보를 보관하는 리소스 서버는 로그인 상태로 유지되는 문제가 있었다. 본 논문에서 제안하는 시스템과 기존 OAuth 프로토콜의 비교 내용은 〔Table 1〕과 같다.

OAuth 프로토콜을 이용하여 클라이언트 서비스를 이용한 사용자는 클라이언트의 사용 종료 후 리소스 서버의 로그아웃 상태를 직접 확인해야 하지만 제안하는 프로토콜은 리소스 서버가 로그인 상태로 남아있을 수 있음을 환기시킨다. 또한, OAuth 프로토콜은 리소스 서버의 로그아웃을 위하여 직접 리소스 서버를 방문해야 하지만 제안하는 프로토콜은 사용자가 리소스 서버의 추가적인 이용 여부 등을 판단하여 클라이언트에서 리소스 서버가 즉각 로그아웃이 되도록 설정할 수 있어 기존 OAuth보다 효율적이고 안전하게 개인정보를 보호할 수 있다.

V. 결 론

최근 웹 서비스는 OAuth 2.0 프로토콜을 이용하여 사용자의 로그인을 지원하고 있다. OAuth 2.0 프로토콜을 이용한 클라이언트 웹 서비스 사용자의 로그아웃 후 리소스 서버가 로그인되어 있지만, 처음 로그인 과정 이후 리소스 서버를 확인하지 못한 사용자는 이를 인지하기 어렵다. 만약 도서관, 백화점, 인쇄소 등과 같이 여러 사용자가 사용하는 PC 환경에서 리소스 서버를 로그인 상태로 두고 자리를 떠날 경우, 그다음 사용자는 계정정보 없이 클라이언트 웹 서비스에 로그인할 수 있을 뿐만 아니라 리소스 서버 중요 자원에 접근할 수 있다.

본 논문에서 리소스 서버의 사용자 자원 보호를 위해 사용자의 클라이언트 웹 서비스 로그아웃 요청 시, OAuth 2.0 프로토콜을 이용하여 서비스에 로그인한 사용자를 대상으로 리소스 서버의 로그아웃 여부 확인을 요구하는 방법을 제안 및 구현하였다. 제안하는 방법을 사용하게 된다면 로그아웃 이후 사용자가 자신의 리소스 서버에 로그아웃 유무를 쉽게 판단할 수 있어 자신이 추가로 이를 이용한 웹 서비스를 사용하려 한다면 유지하고 만약 사용하지 않고 종료하여야 하는 경우에는 로그아웃하여 공공장소와 같은 PC환경에서 자신의 로그인 상태를 쉽게 확인할 수 있다는 점에서 강점을 가지고 있다. 또한, 본 논문에서 제안하는 방법을 이용하여, 회원가입 없이 OAuth 프로토콜을 이용하여 로그인한 사용자의 리소스 서버 자원을 보호할 수 있다.

References

〔1〕 I. Faynberg, H. Lu, and H. Ristock, "On Dynamic Access Control in Web 2.0 and Beyond: Trends and Technologies," BELL LABS Technical Journal, vol. 16, no. 2, pp. 199-218, Sep. 2011.

〔2〕 D. Fett, R. Kusters, and G. Schmitz, "SPRESSO: A Secure, Privacy-Respecting Single Sign-On System for the Web," In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1358-1369, Oct. 2015.

〔3〕 D. Hardt, Ed., "The OAuth 2.0 Authorization Framework," Internet Engineering Task Force (IETF) RFC 6749, Oct. 2012.

〔4〕 R. Wang, S. Chen, and X. Wang, "Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-
On Web Services," In Proceedings of the IEEE Symposium on Security and Privacy, pp. 365-379, May. 2012.

〔5〕 C. Bansal, K. Bhargavan, and S. Maffeis,

"Discovering Concrete Attacks on Website Authorization by Formal Analysis," In Proceedings of the IEEE 25th Computer Security Foundations Symposium, pp. 247-262, Jun. 2012.

〔6〕 OAuth Community Reports, "User Authentication with OAuth 2.0," http://oauth.net/articles/authentication (accessed June 9, 2016).

〔7〕 Eran Hammer, "OAuth 2.0 and the Road to Hell," https://hueniverse.com /2012/07/26/oauth-2-0-and-the-road-to-hell (accessed June 9, 2016).

〔8〕 R. Wang, Y. Zhou, Ed., and Y. Gurevich, "Explicating SDKs: Uncovering Assumptions Underlying Secure Authentication and Authorization," In Proceedings of the 22nd USENIX Security Symposium, pp. 399-414, Aug. 2013.

〔9〕 S. Sun and K. Beznosov, "The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems," In Proceedings of ACM Conference on Computer and Communications Security, pp. 378-390, Oct. 2012.

〔10〕 T. Lodderstedt, Ed., "OAuth 2.0 Threat Model and Security Considerations," Internet Engineering Task Force (IETF) RFC 6819, Jan. 2013.

〔11〕 E. Ferry, and J. O Raw, "Security evaluation of the OAuth 2.0 framework," Information & Computer Security, vol. 23, no. 1, pp. 73-101, 2015.

〔12〕 D. Fett, R. Kusters, and G. Schmitz, "A Comprehensive Formal Security Analysis of OAuth 2.0," Technical Report, pp. 1-90, May 2016.

〔13〕 M Jones, Ed., "OAuth 2.0 Mix-Up Mitigation," Internet Engineering Task Force (IETF) draft-jones-oauth-mix-up-mitigation-01, Jan. 2016.

〔14〕 NaverLogin Guide. "Sign in with NAVER," http://developer.naver.com/ wiki/pages/NaverLogin (accessed June 9, 2016).

〔15〕 J. Kim, J. Park, L. Nguyen_Vu and S. Jung, "An Incomplete Logout problem after using OAuth access token for system Login," In Proceedings of APIC-IST 2016, Jun, 2016.

〔16〕 T. Lodderstedt, Ed., "OAuth 2.0 Token Revocation," Internet Engineering Task Force (IETF) RFC 7009, Aug, 2013.

〈저자소개〉

김 진 욱 (Jinouk Kim) 학생회원
2014년 2월: 숭실대학교 평생교육원 컴퓨터공학과 졸업
2016년 8월: 숭실대학교 정보통신공학과 석사
2016년 9월~현재: (주)디지캡 플랫폼개발부 사원
〈관심분야〉 클라우드 보안, IoT 보안, 사용자 및 디바이스 인증

박 정 수 (Jungsoo Park) 학생회원
2013년 2월: 숭실대학교 정보통신전자공학부 졸업
2015년 2월: 숭실대학교 전자공학과 석사
2015년 3월~현재: 숭실대학교 융합SW공학과 박사과정
〈관심분야〉 클라우드 보안, 모바일 보안, 네트워크 보안, 사용자 및 디바이스 인증

응웬부렁 (Long Nguyen-Vu) 학생회원
2008년 9월~2012년 9월: Vietnam National University of Information Technology
2016년 2월: 숭실대학교 정보통신공학과 석사
2016년 3월~현재: 숭실대학교 융합SW공학과 박사과정
〈관심분야〉 클라우드 보안, IoT 보안, 사용자 및 디바이스 인증

정 수 환 (Souhwan Jung) 종신회원
1985년 2월: 서울대학교 전자공학과 졸업
1987년 2월: 서울대학교 전자공학과 석사
1996년 6월: University of Washington 박사
1988년~1991년: 한국통신 전임 연구원
1997년~현재: 숭실대학교 전자정보공학부 교수
〈관심분야〉 클라우드 보안, 모바일 보안, 네트워크 보안