http://dx.doi.org/10.14400/JDC.2014.12.8.289

Open Authorization에서의 안전한 사용자 권한 인증 방법에 관한 연구

채철주*, 이준환**, 조한진**
한국과학기술정보연구원 R&D시스템개발실*, 극동대학교 스마트모바일학과**

Secure User Authority Authentication Method in the Open Authorization

Cheol-Joo Chae*, June-Hwan Lee, Han-Jin Cho****
Dept. of R&D System Development, Korea Institute of Science and Technology Information*
Dept. of Smart Mobile, Far East University**

요 약 최근 다양한 웹 서비스와 어플리케이션들이 사용자에게 제공되고 있다. 이러한 서비스들은 인증된 사용자에 한해서 서비스를 제공하기 때문에 사용자는 매번 서비스 별로 인증을 수행해야 하는 불편함을 겪고 있다. 이러한 불편함을 해결하기 위해 3rd Party 어플리케이션이 웹 서비스에 대하여 제한된 접근 권한을 얻을 수 있게 해주는 OAuth(Open Authorization) 프로토콜이 등장하게 되었다. 이러한 OAuth 프로토콜은 사용자에게 편리하고 유연한 서비스를 제공하지만 권한 획득에 대한 보안 취약점을 가지고 있다. 그러므로 본 논문에서는 OAuth 2.0 프로토콜에서 발생할 수 있는 보안 취약점을 분석하고 이러한 보안 취약점을 보완한 방안을 제시한다.

주제어 : Open Authorization, OAuth 취약점, 권한 인증, 사용자 인증, 보안

Abstract Recently, the various web service and applications are provided to the user. As to these service, because of providing the service to the authenticated user, the user undergoes the inconvenience of performing the authentication with the service especially every time. The OAuth(Open Authorization) protocol which acquires the access privilege in which 3rd Party application is limited on the web service in order to resolve this inconvenience appeared. This OAuth protocol provides the service which is convenient and flexible to the user but has the security vulnerability about the authorization acquisition. Therefore, we propose the method that analyze the security vulnerability which it can be generated in the OAuth 2.0 protocol and secure user authority authentication method.

Key Words : Open Authorization, OAuth Vulnerability, Authorization, Authentication, Security

1. 서론

컴퓨터와 네트워크의 발전으로 다양한 웹 어플리케이션들이 개발되고 사용되고 있다. 이러한 웹 어플리케이션 중에서 가장 대표적으로 많이 사용하고 있는 웹 어플리케이션이 SNS(Social Network Service)이다. 이러한

Received 10 June 2014, Revised 15 July 2014
Accepted 20 August 2014
Corresponding Author: Han-Jin Cho (Dept. of Smart Mobile, Far East University)
Email: hanjincho@hotmail.com

ISSN: 1738-1916

SNS 웹 어플리케이션 사용자는 여러 어플리케이션과 손쉽게 연동·공유하고자 원한다. 그러나 기존의 사용자 인증은 ID / Password를 이용한 기본 인증을 기반으로 하기 때문에 보안에 취약한 구조를 가지고 있다. 그러므로 이러한 보안 취약점을 극복하기 위해 각 웹 어플리케이션 마다 별도의 인증 방법을 개발하여 사용자를 인증하였다. 이러한 웹 어플리케이션 인증 방법의 예로 Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Amazon 웹 서비스 API 등이 있다[1][2][3].

이러한 인증 방법은 각 어플리케이션 마다 상이하기 때문에 사용자는 한번만 인증하면 어플리케이션 마다 별도의 인증 절차 없이 사용할 수 있는 방법을 필요로 하게 되었다. 이러한 사용자 요구사항에 맞추어 등장한 것이 OAuth(Open Authorization) 프로토콜이다. OAuth 프로토콜은 기존의 서로 다른 인증 방식의 표준화를 목적으로 만들어졌기 때문에 어플리케이션 마다 별도의 인증 절차 없이 여러 어플리케이션을 통합 인증하여 사용 하는 것이 가능하다. OAuth는 3rd Party 어플리케이션과 사용자간 한 번의 인증만을 통해 여러 서비스 제공자들이 제공하는 서비스를 이용할 수 있는 권한을 인증해주는 프로토콜이다. 그러나 OAuth는 편의성과 확장성을 제공하지만 3rd 어플리케이션과 사용자 사이의 인증 과정에 여러 보안 취약점을 가지고 있다. 이러한 보안 취약점은 기존의 웹 어플리케이션 인증 취약점과는 달리 OAuth 프로토콜에서는 한번 인증에 성공하게 되면 다수의 서비스를 이용할 수 있기 때문에 심각한 보안 문제점을 발생시킬 수 있다.

그러므로 본 논문에서는 OAuth 프로토콜에서 발생할 수 있는 보안 취약점에 대해 분석하고 이를 극복할 수 있는 3rd 어플리케이션과 사용자 인증 방법을 제안한다. 논문의 구성은 다음과 같다. 2장에서는 OAuth 프로토콜에 대해 기술하고 3장에서는 OAuth 프로토콜에서 발생할 수 있는 보안 취약점에 대해 기술한다. 그리고 4장에서는 3장에서 분석한 보안 취약점을 극복할 수 있는 인증 방법을 제안하고 5장에서 결론을 맺는다.

2. OAuth 프로토콜

OAuth는 사용자의 ID/PW를 3rd Party 어플리케이션에 노출하지 않고 인증을 할 수 있으며 서비스 제공자는 인증한 API에 대해 권한을 부여하고 이에 따라 API를 제공할 수 있다. OAuth는 2007년 OAuth 1.0을 발표하고 2008년 6월 OAuth 1.0의 보안 문제를 수정한 OAuth 1.0a를 발표하여 2010년 IETF 표준으로 등록되었다. 그 후 2012년 OAuth 2.0 draft를 발표하고 2012년 10월 OAuth 2.0이 표준으로 등록되었다.

2.1 OAuth 1.0a 프로토콜

OAuth 1.0a 프로토콜은 Service Provider, Consumer, User로 구성되어 있으며 서비스 요청을 위해 Request Token, Access Token, Consumer Key를 사용한다. <Table 1>은 OAuth 1.0a 프로토콜에서 사용하는 구성요소와 역할을 보여주고 있다[4][5].

〈Table 1〉 OAuth 1.0a Protocol Definitions

Symbol	Meaning
Service Provider	- A web application that allows access via OAuth.
Consumer	- A website or application that uses OAuth to access the Service Provider on behalf of the User.
User	- An individual who has an account with the Service Provider.
Request Token	- A value used by the Consumer to obtain authorization from the User, and exchanged for an Access Token.
Access Token	- A value used by the Consumer to gain access to the Protected Resources on behalf of the User, instead of using the User's Service Provider credentials.
Consumer Key	- A value used by the Consumer to identify itself to the Service Provider.

OAuth 1.0a 프로토콜의 일반적인 동작 절차는 다음과 같다. 먼저 어플리케이션은 Service Provider로부터 사용 허가를 받았음을 의미하는 Consumer Key와 Consumer Secrete을 미리 발급 받는다. 어플리케이션은 Service Provider에게 Consumer Key, Consumer Secrete을 이용하여 인증 요청과 Request Token 발급을 요청한다. Service Provider는 Consumer Key, Consumer Secrete를 이용하여 어플리케이션 인증을 수행하고 인증에 성공하게 되면 어플리케이션이 요청한 Request Token을 발

급한다. 어플리케이션은 Request Token을 이용하여 User를 Service Provider로 Redirect한 후 User 인증을 요청한다. Service Provider는 User 인증을 완료한 후 수신한 Request Token과 동일한 값을 어플리케이션에게 Redirect하여 인증을 요청한 User인지 확인한다. 그리고 어플리케이션은 서비스에 접근하기 위해 Service Provider에게 Access Token 발급을 요청하고 Access Token을 이용하여 서비스에 접근할 수 있다.

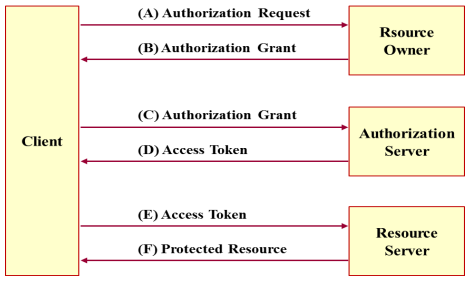
2.2 OAuth 2.0 프로토콜

OAuth 1.0a 프로토콜과 OAuth 2.0 프로토콜이 서로 호환 되지는 않지만 프로토콜의 일반적인 동작 절차는 동일하다. <Table 2>는 OAuth 2.0 프로토콜에서 사용하는 구성 요소와 역할을 보여주고 있다[6-8].

〈Table 2〉 OAuth 2.0 Protocol Definitions

Symbol	Meaning
Resource Owner	- An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.
Client	- An application making protected resource requests on behalf of the resource owner and with its authorization.
Resource Server	- The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.
Authorization Server	- The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

OAuth 2.0 프로토콜의 일반적인 동작 절차로 먼저 Client는 자원을 이용하기 위한 접근 권한을 의미하는 Access Token을 Resource Owner에게 요청한다. Authorization Server는 Client와 User 정보를 인증한 후 자원에 접근할 수 있는 권한을 발급하고 이를 이용하여 Client는 User의 자원에 접근할 수 있다. [Fig. 1]은 OAuth 2.0 프로토콜의 일반적인 동작 절차를 보여주고 있다.



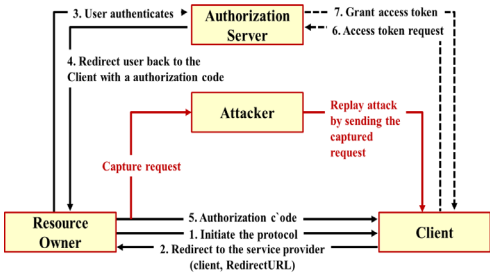
[Fig. 1] OAuth 2.0 Protocol Flow

3. OAuth 프로토콜 보안 취약점

OAuth 프로토콜의 경우 자원 접근 권한을 가지는 Access Token을 탈취할 경우에는 여러 어플리케이션을 이용하여 자원에 접근할 수 있다는 보안 취약점이 발생한다. 이러한 Access Token을 탈취할 수 있는 방법으로는 일반적인 네트워크 보안 취약점인 재전송 공격, 피싱 공격, 위장 공격이 있다. 본 장에서는 OAuth 2.0 프로토콜에서 재전송 공격, 피싱 공격, 위장 공격을 이용하여 권한을 탈취할 수 있는 보안 취약점에 대해 기술한다[9-11].

3.1 재전송 공격

OAuth 2.0 프로토콜에서는 Client가 Resource Owner에 접근 허용 코드로 authorization code를 사용한다. 그러므로 authorization code는 한번만 사용되어야만 한다. 그렇지 않으면 authorization code를 이용한 재전송 공격이 가능하다. [Fig. 2]는 OAuth 2.0 프로토콜에서 authorization code를 이용한 재전송 공격 방법을 보여주고 있다.

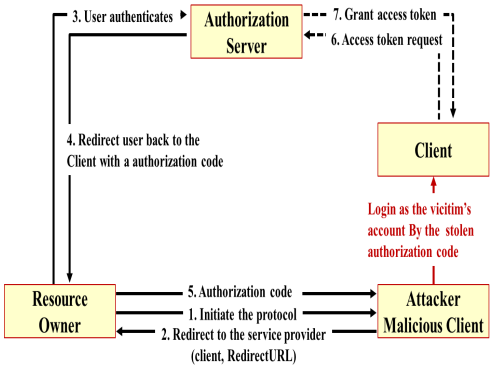


[Fig. 2] Replay Attack

재전송 공격을 재전송 공격을 위해 공격자는 Resource Owner와 Client 사이의 authorization code를 캡처한다. 그리고 공격자는 캡처한 authorization code redirection request를 이용하여 authorization code와 관련된 Resource Owner의 계정으로 로그인하기 위해 Client에게 요청을 재전송할 수 있다. 공격자는 이러한 재전송 공격을 통해 Resource Owner에 대한 정보를 획득한 후 Resource Server에 접근할 수 있는 권한을 획득할 수 있다.

3.2 피싱 공격

피싱 공격은 공격자가 쉽게 공격할 수 있으면서도 효과적인 공격 방법 중하나이다. OAuth 2.0 프로토콜에서 [Fig. 3]과 같은 방법으로 피싱 공격을 시도할 수 있다.

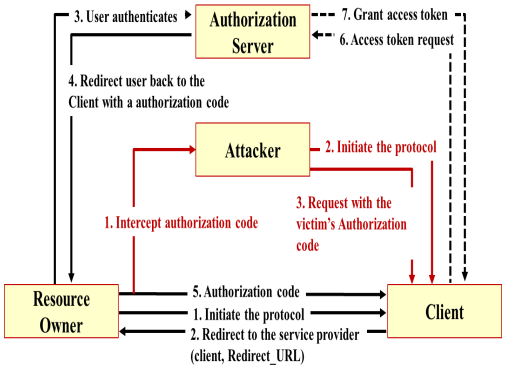


[Fig. 3] Phishing Attack

Client가 Resource Owner의 정보를 이용하기 위해서는 Authorization Server의 인증을 거쳐야만 한다. 이 과정에서 공격자는 악의적인 Client를 생성하여 인증에 필요한 Resource Owner ID / Password를 탈취하는 것이 가능하다. 공격자는 이러한 피싱 공격으로 획득한 Resource Owner의 ID / Password를 이용하여 로그인한 후 Resource Server에 접근할 수 있는 권한을 획득할 수 있다.

3.3 위장 공격

공격자는 authorization code를 인터셉트하여 위장 공격을 시도할 수 있다. OAuth 2.0 프로토콜에서 [Fig. 4]와 같은 방법으로 위장 공격을 시도할 수 있다. 위장 공격을 시도하기 위해 공격자는 먼저 authorization code를 도청하여 인터셉트한다. 그리고 공격자는 인터셉트한 authorization code를 유지하기 위하여 실제 authorization code 요청을 차단한다. 그리고 공격자는 Client와 초기 세션을 시작하고 세션이 시작되면 인터셉트한 authorization code를 이용하여 Resource Server에 접근할 수 있는 권한을 획득할 수 있다.



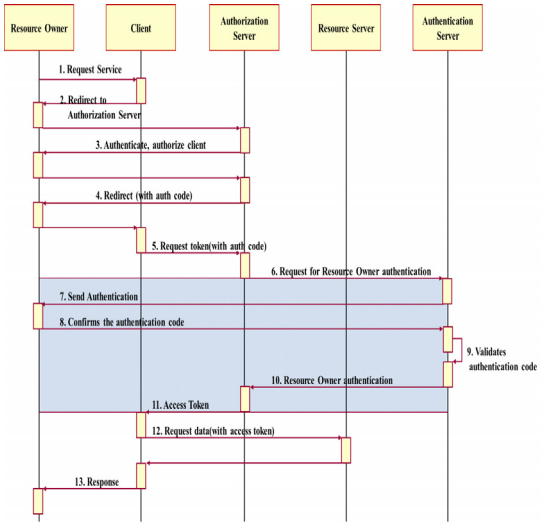
[Fig. 4] Impersonation Attack

4. OAuth 프로토콜에서 안전한 사용자 권한 인증 방법

본 논문에서는 OAuth 2.0 프로토콜에서 발생할 수 있는 권한 인증에 대한 보안 문제점을 해결하기 위한 방법으로 Resource Server에 접근할 수 있는 권한인 Access Token을 발급하기 전, Resource Owner를 인증하여 안전하게 Access Token 발급을 할 수 있는 방법을 제안한다. [Fig. 5]는 본 논문에서 제안하는 안전한 사용자 권한 부여를 위해 확장된 OAuth 2.0 프로토콜 절차를 보여준다. 제안 방법에서 사용자 권한 인증 절차는 다음과 같다.

- 1단계: Resource Owner는 Client에 접속하여 3rd Party어플리케이션 서비스를 요청한다.
- 2단계: Client는 Resource Owner와 인증하기 위해 Resource Owner를 Authorization Server 로 Redirect 한다.
- 3단계: Authorization Server는 Client와 Resource Owner를 인증한다.

· 4단계: Authorization Server는 authorization code와 함께 Resource Owner를 Client로 Redirect한다.
· 5단계: Client는 authorization code를 이용하여 Authorization Server에게 Resource Server에 접근할 수 있는 권한을 가지는 Access Token 발급을 요청한다.
· 6단계: Authorization Server는Client의 Access Token 발급 요청이 Resource Owner에 의한 요청인지 인증하기 위해 Authentication Server에게 Resource Owner 인증을 요청한다.
· 7 단계: Authentication Server 는 Resource Owner에게 인증코드를 전송한다.
· 8 단계: Resource Owner는 Authentication Server로부터 수신한 인증코드를 확인 한 후 Authentication Server로 전송한다.
· 9 단계: Authentication Server는 Resource Owner로부터 수신한 인증코드를 검증한다.
· 10단계: 인증코드가 검증되면 Authentication Server는 Authorization Server에게 Resource Owner가 승인되었음을 알린다.
· 11단계: Authorization Server는 Client에게 Access Token을 발급한다.
· 12단계: Client는 Access Token을 이용하여 Resource Server에 접근하여 자원을 요청한다.



[Fig. 5] Proposed Authorization Authentication in OAuth 2.0

제안 방법에서는 외부의 Authentication Server를 이용하여 인증코드를 검증을 사용하기 때문에 재전송 공격, 피싱 공격, 위장 공격을 통한 권한 인증 취약점을 해결할 수 있다. 먼저 재전송 공격의 공격자는 Access Token을 발급하기 위해서는 Authentication Server로부터 인증코드를 발급 받고 검증받아야하기 때문에 authorization code를 이용한 재전송 공격을 예방할 수 있다. 피싱 공격의 경우에도 Resource Owner와 Authentication Server와 인증코드를 상호 인증하기 때문에 피싱 공격과 위장 공격을 예방할 수 있다.

5. 결론

OAuth 프로토콜은 기존의 서로 다른 인증 방식의 표준화를 목적으로 만들어졌기 때문에 어플리케이션 마다 별도의 인증 절차 없이 여러 어플리케이션을 통합 인증하여 사용 하는 것이 가능하다. 이러한 OAuth 프로토콜은 편의성과 확장성을 제공하지만 3rd 어플리케이션과 사용자 사이의 인증 과정에 여러 보안 취약점을 가지고 있다. 본 논문에서는 OAuth 프로토콜에서 발생할 수 있는 보안 취약점을 극복하기 위해 외부 Authentication Server를 이용하여 인증코드를 검증함으로써 안전하게 사용자 권한을 인증할 수 있는 방법을 제안하였다.

제안 방법은 OAuth 프로토콜의 보안 취약점을 극복하기 때문에 기존의 OAuth 프로토콜과 비교하여 능동적인 서비스를 제공할 수 있게 한다. 향후 제안 방법을 이용한 OAuth 프로토콜의 사용으로 보안 사고를 예방할 수 있으며 또한 현재 활발하게 연구되고 있는 OpenID 분야에 OAuth 프로토콜을 적용함으로써 OpenID를 활성화 시킬 수 있을 것이다.

REFERENCES

[1] Seon-Joo Kim, An Efficient Access Control Mechanism for Application Software using the OAuth in the SaaS Cloud System, Graduate School of PaiChai University, 2013.
[2] Jeong-Kyung Moon, A Delegator for Authentication

Management System using OAuth in Cloud Computing Environment, Graduate School of Kongju National University, 2013.

[3] Myung Hyun Han, Research on the extended OAuth protocol for real-name authentication, Graduate of School of Information Technology Chung-Ang University, 2013.

[4] E. Hammer-Lahav, The Oauth 1.0 Protocol. Internet Engineering Task Force(IETF) RFC 5849, 2010.

[5] Mohamed Shehab, Said Marouf, Recommendation Models for Open Authorization. IEEE transactions on dependable and secure computing, Vol. 9, No. 4, 583-595, 2012.

[6] D. Hardt, The OAuth 2.0 Authorization Framework. Internet Engineering Task Force(IETF) RFC 6749, 2012.

[7] M. Jones, The OAuth 2.0 Authorization Framework: Bear Token Usage. Internet Engineering Task Force(IETF) RFC 6750, 2012.

[8] M. Noureddine, R. Bashroush, A Provisioning Model towards OAuth 2.0 Performance Optimization. Proceedings of the 2011 10th IEEE International Conference On Cybernetic Intelligent Systems, pp. 76-80, 2011.

[9] Sooyoung Lee, Jonguk Kim, Sukin Kang, Manpyo Hong, Improving the Security of OAuth Client using Obfuscation Techniques, Proceedings of the 2013 KSII Conference, Vol. 14, No. 1, pp. 159-60, 2013.

[10] Young Gon Jung, Sanf Rea Lee, Gi Hun JANG, Heung Youl YOUM, Security Problems for Secure OAuth authentication protocol, Proceesings of the 2011 KICS Conference, pp. 952-953, 2011.

[11] Feng Yang, Sathiamoorthy Manoharan, A security analysis of the OAuth protocol. In Proc. Of Communications, Computers and Signal Processing, pp. 271-276, 2013

채 철 주(Chae, Cheol-Joo)

- 2006년 2월 : 한남대학교 컴퓨터공학과(공학석사)
- 2009년 8월 : 한남대학교 컴퓨터공학과(공학박사)
- 2009년 9월 ~ 2013년 4월 : 한국전자통신연구원 선임연구원
- 2013년 4월 ~ 현재 : 한국과학기술정보연구원 선임연구원
- 관심분야 : 정보보호, 바이오 보안, 네트워크 보안
- E-Mail : cjchae@kisti.re.kr

이 준 환(Lee, June-Hwan)

- 1996년 2월 : 단국대학교 전자공학과(공학석사)
- 2001년 2월 : 단국대학교 전자공학과(공학박사)
- 2001년 3월 ~ 현 재 : 극동대학교 스마트모바일학과 교수
- 관심분야 : 음성처리시스템, 멀티미디어응용, 스마트미디어, 모바일앱
- E-Mail : rainbow@kdu.ac.kr

조 한 진(Cho, Han-Jin)

- 1999년 2월 : 한남대학교 컴퓨터공학과(공학석사)
- 2002년 8월 : 한남대학교 컴퓨터공학과(공학박사)
- 2002년 8월 ~ 현재 : 극동대학교 스마트모바일학과 교수
- 관심분야 : 정보보호, 스마트폰 보안, 모바일 콘텐츠
- E-Mail : hanjincho@hotmail.com