A Privacy-aware OAuth Monitoring Platform

Abul Ehtesham

Advanced Information Security and Privacy , Department of Computer Science,
Kent State University, Kent, OH 44240, USA

aehtesha@kent.edu

**Abstract.** OAuth is an open standard authorization framework which provides applications to get limited access to the user accounts on HTTP service. It works by sending user authentication to the service provider (Facebook, Google, etc.) that hosts the user account and authorizes the third-party applications to access the user account. It provides a way to integrate social login on third-party applications on the web, mobile and desktop platform. The service provider of OAuth provides analytical tool for third-party application to monitor their user activity, types of user connected, devices on which their application downloaded and demographic of user which puts the privacy of the users at risk. But to the best of our knowledge, there is no platform/application available to the users to monitor how and when third-party applications use permissions which were given to them by the users when they logged in to the service providers’ (Facebook, Google, etc.) account. In this work, we have implemented a platform and developed an application called MyOAuth that can be used by the users to monitor the accesses by the third-parties. This is an interesting start attempting to give control to users regarding their data privacy.

**Keywords:** OAuth, Permission, Privacy.

1. Introduction

The OAuth [1] (Open standard for Authorization) protocol provides a generic framework to let a resource *owner*/*user* authorize *third-party* (a.k.a. the *client*) to access the owner’s information (a.k.a. resource) such as email, gender, friend list, etc. held at a server without revealing owner’s credentials (such as username and password) to the third-party. The protocol normally works as follows. When the client requests the user to give permission to access their resource, if the user authorizes the request, the application receives an authorization grant. The application requests an access token from authorization server by sending the *client id*, the *secret key* and the *authorization grant*, if the application is authenticated and authorization grant is valid, the authorization server sends success response by sending access token to the application which results in the completion of authorization. Finally, the application requests by sending valid access token to the resource server which sends resource of user to the application.

The provider of OAuth, such as Facebook, Google, etc., provides analytical tools[[1]](#footnote-1),[[2]](#footnote-2) (such as the one shown in Appendix A Fig. 3) for third-parties to monitor active users (Monthly, Weekly, Daily users), the number of downloads, demographics of application users and revenue they are earning. But there is no platform designed for the information/resource owner to monitor the accesses and the activity with the data.

The resource owners who are giving access to their data to the client application are not able to monitor the client activity that is how and when the client is using their data. This becomes a serious privacy issue specially when the client is posting on behalf of the user on the social media with the given write permission to the client application.

In this paper, we introduce a platform and application which will allow the user to monitor the client activity, how and when the client is using their data. An implementation of OAuth protocol is used as the basis for understanding the protocol with an additional function to monitor the client activity on user data by maintaining log of the client activity. The paper also discusses analytical tools available for the client to track the applications used by the users.

The rest of this paper is organized as follows. Section 2 introduces the OAuth protocol. Section 3 reviews related work. Section 4 presents the platform to monitor the client activity. Section 5 gives information of tools and technology used. The final section concludes the work and discusses future work.

1. OAuth

OAuth provides the third-party application a framework to access user’s account. Before the emergence of OAuth, the simplest way to access user’s account was by sharing the user’s password. This practice had many security and privacy issues related to the user’s account. The OAuth framework provides user’s privacy based on their preference. The OAuth framework comprises of OAuth Roles and Grant types.

The different OAuth Roles are *Resource Owner*, *Resource/Authorization Server* and *client*.

2.1. Resource Owner

The resource owners are the users who authorize the third-party to access their account. The access to the account can be controlled by the user based upon the *scope*.

2.2. Resource / Authorization Server- API

The Resource Server and Authorization Server are controlled by OAuth Server. Both Servers are responsible for maintaining the privacy of the user. Resource server provides the user’s account details to the client only if the authorization server confirms the client is authorized by issuing an access token to the client. From development point of view, the servers are referred as service or API.

2.3. Client

The Client is the third-party application that accesses the user’s account. A client needs to authorize itself with the OAuth Server initially.

Authorization grant varies depending upon the requirement. In OAuth, the different cases for grant types are as follows:

### 2.3.1. Authorization Code. The Authorization code is used when we have server-side applications. The benefits of authorization code is that the code is not available for users which result in the client secret confidentiality. It follows redirection-based flow which forces the application to interact with user agent and receives the API authorization code via the user agent.

### 2.3.2. Implicit. The Implicit grant type targets the Mobile and Web application platform. With the implicit grant type, the client’s confidentiality is not guaranteed. As it is redirection-based flow, the access token is given to application via user agent. Thus, the access token is available to the users and other applications on user’s device. The refresh token is not supported in this flow.

### 2.3.3. Resource Owner Password Credentials. In this grant type, the username and password are given to the application by the user which obviously is the least secure method of grant type.

### 2.**3.4. Client Credentials.** This type of grant is used when someone wants to use their own resource.

In this paper, we are referring to Authorization Code (2.3.1) grant type. The steps for the grant type - Authorization Code are as follows.

### Step 1 - Client Registration. The Client registers itself to OAuth Server by providing the application name, *callbackurl* and scope client wants to access. On successful registration, the service gives credential to the client which is the combination of the client id and the client secret key and scope for the client. Callbackurl is the address at which service redirects the user after he authorizes or denies. This is the end URL at which application receives the authorization code. The Scope can be defined as the list of permissions requested by the client. Fig. 1a illustrates the client registration on OAuth Server where the client requests for read (email, name and age) and write permissions from the OAuth Server. The OAuth Server in return provides a client ID and Secret key to the client. The client ID is the unique public identifier for the client, and the secret key is private key with the combination of both, the client identification is performed by the API or service.



1. Client Registration on OAuth Server

 

b) User Authorization followed by application receiving authorization code

Fig. 1 – OAuth client registration and authorization model

### Step 2 User Authorization.

Whenever a user tries to login, the first step done by the OAuth Server is to validates the client application with the client identifier and then it asks the user to give consent to access the scope that client has requested. Fig. 1b illustrates the User Authorization process.

### Step 3 Application Receives Authorization Code

When the user authorizes the application, the service redirects the user to the client’s callbackurl where the client receives the authorization code.

### Step 4 Accessing User Information

The client requests the access token from the API by sending its authentication detail along with its secret key to the API token end URL. On successful authorization, the API will send access token, token type, expiration for the token and refresh token. With the access token, the client can access the resource of the user from the server. The validity of token decides whether the refresh token can be used.

1. Related work

The providers of OAuth such as Facebook, Google, etc. prepare analytical application for client to monitor their application to find active users (Monthly, Weekly, Daily users), number of downloads, demographics of application user, the type of operating system (iOS, Android) used by user and revenue they are earning. The Client can get the information about the popularity of its application among the type of users as he is getting demographic information such as gender and age of the users, using this analytical app. For instance, Facebook research shows the fewer the number of scope permissions an application asks for, the more will be the number of users using it. So Facebook is providing guidelines to the client regarding how to design an application that can have the significant rise in the number of users.

Sun and Beznosov [3], Shehab and Mohsen [4], and Wang *et al.* [5] have revealed different attacks against various OAuth implementations. Yang and Manoharan [2] have also presented a comprehensive analysis of the security with OAuth protocol. Although there have been many research performed in this area, no one has actually focused on designing a platform for the users to monitor and take control over their data privacy.

From user’s perspective, normally OAuth provider gives address URL to check setting related to the connected client application and permission given to scope (read or write) using their social login. Facebook has the feature in its setting URL that helps the user to check the number of application connected using login with Facebook. It also gives information about different permission given to the application by the user. Facebook provides easy step to revoke permission to the access scope to the client. Removing a client application from Facebook notify the user that its information is still there in the client database. So, the user needs to contact the client application to revoke his data completely. Thus, it is a lengthy way of contacting the developer of the application to request to remove the information collected by them after user consent. Google[[3]](#footnote-3) is also providing the information about the data accessible by the client such as the read and write permissions. Also, it is providing the time of authorization granted by the user to the client application using Google credentials. Unlike Facebook, it does not provide information about the eradication of user data from the client’s database but provides a note[[4]](#footnote-4). Presently, we cannot find any guideline available for the user, based on which he can evaluate which permissions should be given to a client. Regarding user perspective, presently, there is no platform or application available that can track client activity at every instance of time. Such as, what information about the user, a client application is accessing from within the defined scope. It basically means tracking the client activity with user’s account.

The User must also be aware of the status of the client applications attached to his account. Such as, in case a client the application is no more useful for the user, then he must be notified by anyways so that he can revoke the permission from the client. Also, there is no facility available for the user, using which he can observe, how many times a client application is hitting his account, to retrieve his account information or to perform any activity defined within the scope. Thus, we found that client is getting more benefits than the user. In addition to the currently available benefits, we have developed a model which is demonstrating all the above defined benefits for the user.

1. OAuth Monitoring Platform and Application

The focus of Monitoring Platform and Application (MyOAuth) is to maintain the log for third-party application (client) activity by enhancing present OAuth framework by adding the additional table with column attribute like application name, the scope of data accessed and accessed time of scope. Our way of implementation as shown in Fig. 2 for OAuth will help in monitoring activity of client such as

• Number of active and inactive applications connected to the account.

• How many times the application is making use of user data.

• Notification for the inactive app to revoke permission.

Fig. 2: Model of MyOAuth platform

* 1. Active and Inactive application

The Application gives information of active and inactive client applications connected with user’s account. The user can also filter active and inactive application. Appendix A, Fig. 4 shows some screenshots of the application. In Fig. 4a, we can easily visualize the active and inactive application. To determine the status of application (whether active or inactive), we are calculating the number of times, a client application is accessing user’s data in a specific period (e.g. 60 days). If a client application is accessing user’s information within the defined time, then we consider it as an active application otherwise inactive, which in future helps the user to decide whether to revoke the permission from the application or not.

* 1. Count of client accessing the user data

The app provides a count that is how many times the client has accessed the user data along with detail information such as time of access of scope such as email, gender, likes, write on the wall. From Fig. 4b, we can see scope such as read (email and gender) and write client has accessed along with message written by the client with date and time. The user can also see the count of data accessed by client week wise with the help of line graph as shown in Fig. 4c. It is a graph of hit (number of time data accessed by clients per week).

* 1. Scope

The application visualizes different scope given to different applications using an easy to read pie chart, as shown in Fig. 4d.

* 1. Notification to revoke permission

Once the status of the application (active or inactive) is determined, we target on revoking permission from the client application based upon user’s requirement. The revoke permission will be helpful for revoking the inactive applications. Similarly, revoke permission will be effective if user suspects intention of the client application.

**5. Tools and Technology Used**

 In this application, we have implemented an OAuth provider and named it as MyOAuth. The implementation for MyOAuth (server/provider) is done using PHP OAuth 2.0 library[[5]](#footnote-5). The web service is written in PHP and database is MySQL. The user interface is made in SAPUI5[[6]](#footnote-6) JavaScript framework and will be available for the public to download at [link will be provided in camera-ready].

**6. Conclusion and Future Work**

Currently there is no privacy-aware monitoring application for the user to give control over the use of their data in OAuth. Our platform and application is a solid first attempt to give power back to the users. The application targets on client’s activity based upon monitoring HOW and WHEN, a client uses user’s data. The information about retention policy of client application should be added at the time of authorization of application by the resource owner. The future work is to decide active and inactive application based on parameter such as average hit with respect to weeks, week with respect to number of times data got accessed. Easy step to request the client to remove user’s data completely from its database. The main challenge is to maintain a universal monitoring application which manages setting of all OAuth service providers so that the users can have a centralized control over their accounts given by different providers.

**REFERENCES**

1 Barry Leiba, Huawei Technologies, OAuth Web Authorization Protocol, January/February (2012 vol. 16).

2 Feng Yang and Sathiamoorthy Manoharan, A security analysis of the OAuth protocol, Department of Computer Science, Uni. of Auckland, New Zealand, 2013

3 - S.-T. Sun and K. Beznosov. The devil is in the (implementation) details: An empirical analysis of oauth sso systems. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS ’12, pp 378–390, NY, USA, 2012.

4- M. Shehab and F. Mohsen. Towards enhancing the security of oauth implementations in smart phones. In Proceedings of the IEEE 3rd International Conference on Mobile Services, 2014.

5- R. Wang, Y. Zhou, S. Chen, S. Qadeer, D. Evans, and Y. Gurevich. Explicating sdks: Uncovering assumptions underlying secure authentication and authorization. In Proceedings of the 22Nd USENIX Conference on Security, SEC’13, pages 399–414, Berkeley, CA, USA, 2013. USENIX Association.

Appendix A – Supplementary figures of the analytical tools and MyOAuth interface



Fig 3. – Sample snippet of Facebook analytical tool

 a) Client Status b) Client Scope Accessed Details

 c) Graph Hit vs week d) Scope for different client

Fig. 4: MyOAuth Application Interface

1. https://developers.facebook.com/products/analytics [↑](#footnote-ref-1)
2. https://console.developers.google.com/apis/dashboard [↑](#footnote-ref-2)
3. https://security.google.com/settings/security/permissions [↑](#footnote-ref-3)
4. “*Google is not affiliated with third-party services that ask for authentication. Google simply provides sign-in technology to these sites*” available at:

 https://support.google.com/accounts/answer/112802?hl=en&ref\_topic=7188760 [↑](#footnote-ref-4)
5. PHP OAuth 2.0 library at: https://github.com/bshaffer/oauth2-server-php-docs [↑](#footnote-ref-5)
6. Available at: https://sapui5.hana.ondemand.com [↑](#footnote-ref-6)