International Journal of Engineering & Technology, 7 (1.1) (2018) 692-695

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A survey on OAUTH protocol for security

V. Srikanth [1] *, Jupalli. Sneha Latha [1], Dinne Ajay Kumar [1], Kakarla Uma Maheswari [1]

[1] *Dept. of Computer Science Engineering, KLEF, Vaddeswaram*
**Corresponding author E-mail:*

Abstract

Web is a dangerous place. For each administration, each API's, there are clients who might love simply to get through the different layers of security you've raised. It is one of the most powerful open standard authorization protocols available to all API developers today. Most of the popular social network API's like Google, Twitter and Facebook uses OAuth 2.0 protocol to intensify user experience while signing-on and social sharing. The code written for authorization may be leaked during transmission which then may lead to misuse. This paper uses an attacker model to study the security vulnerabilities of the OAuth protocol. The experimental results on Google API shows that some common attacks like Phishing, Replay and Impersonation may be possible on this protocol.

Keywords: *OAuth 2.0; Security Vulnerabilities; Authentication.*

1. Introduction

OAuth 2.0 implies open standard protocol which is a token based approval and validation predominantly used over web and different applications of the web. It empowers clients to concede access to the outsider applications like Facebook, Twitter in a constrained time to such an extent that the client's credentials isn't known to anybody. It gives the support of the end client inside the entrance token and the way toward getting the token is known as the flow. It is broadly utilized as a part of online networking applications. The answer for OAuth 2.0 is first proposed in 2007 in draft form by different developers from Twitter and Magnolia which was systematized in the OAuth Core 1.0 final draft in December 2007. OAuth was formally distributed as RFC 5849 out of 2010 after that numerous applications required the use of OAuth2.0.It is the new framework evolution that was first distributed as RFC 6749 alongside a Bearer Token Usage definition in RFC 6750.In OAuth 2.0,we send the tokens over SSL layer as instead of signing every request in OAuth 1.0 which become troublesome for the developers. It can address the local applications in the cell phones effectively. The protocol gives different roles for authorization server and resource server in which the task of authorization server is certify the customer.Resource Server handles API calls to get too confined assets. OAuth 2.0 provides a generic framework that resource owner authorizes the third-party in order to access the resources of the owner without revealing the owner's credentials to the third-party[1].
The working of the protocol is roughly shown as follows. The resource owner sends a request to the server to provide a valet key to the third-party. The server then authenticates the resource owner and sends a valet key to the third-party which in turn access the owner's resources. See figure 1

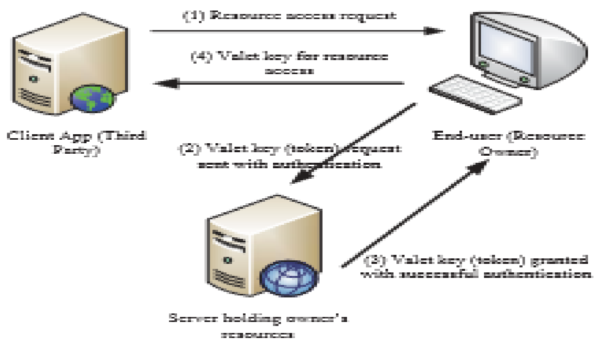

(1) Resource access request
(4) Valet key for resource access
Client App (Third Party)
End-user (Resource Owner)
(2) Valet key (token) request sent with authentication
(3) Valet key (token) granted with successful authentication
Server holding owner's resources

Fig. 1: The Rough Overview of the Protocol.

Sometimes, the valet key may be leaked which lead illegal access to owner's resources. This paper discusses about some of the mechanisms that may lead to this attacks [1] [2].
The rest of the paper is organized as follows. Section II introduces about the protocol. Section III shows related work. Section IV and V shows use of the protocol in internet of Things. Section VI summarizes the protocol.

2. Introduction to OAuth protocol

OAuth by itself does not explain any protocol for authentication. Instead, it is a simple framework for authentication decisions and mechanisms.
The OAuth 2.0 protocol roles are as follows:
Client:
This is the most important role of this protocol. An application uses an API to access the protected resources of the owner (User) with his/her authorization [7].
Resource owner:
An entity capable of providing access to a protected resource is known as resource owner. It is machine, not simply individuals [7].
Resource server:

An entity that hosts the protected resource of the owner is known as resource server. For example, Facebook or Twitter has a resource server. In other terms, It represents an application hosting cloud services to the end-user [7].

Authorization server:

An entity that authorizes the client application to get the protected resources of the owner is known as authorization server. Internally, the authorization server and the resource server are not separate and the developers of both servers should know how they communicate if they are separate.

1) The client sends an approval request to the resource owner directly or indirectly via authorization server to access the protected resources.
2) The client gets an authorization grant that could be a credentials showing the resource owner's authorization, communicated utilizing one in every of four grant types outlined during this protocol specification or using an extension grant type.

- Authorization code grants:

First the resource owner is authorized by the authorization server and then the resource owner grants the associate authorization code to the client. This type of grant is known as authorization code. grant. The client does not need the login details of the owner. This type of grant is very secure.

- Implicit grants

This grant type is implemented mainly on the client so as to get the access token immediately. The advantage of this implicit grant is that reduces the protocol overhead, but there is a drawback that it causes security threats. It is a simplified authorization code flow optimized for clients enforced by every browser employing a scripting language JavaScript. The responsiveness and potency of some client's is increased by these grants since it reduces the number of round trips required to obtain access token.

- Resource owner password credential grants:

In this type of grant, using the credentials owned by resources, the access token is issued to the client. Mainly this grant type is implemented only when the user trusts the client completely.

- Client credential grants:

This type of grant can be used to limit access to the owner's resources.

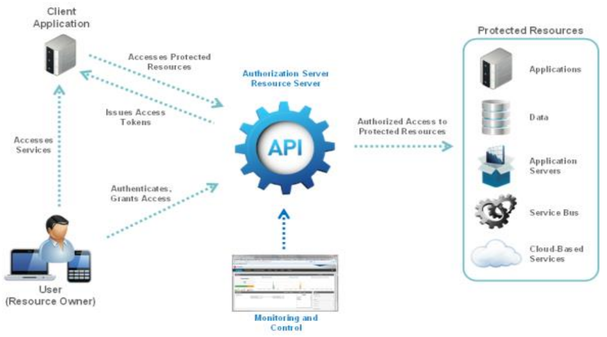


Fig. 2: Shows the Working of the OAuth 2.0 Protocol.

3) The client requests the access token after it is authorized by the authorization server by giving authorization grant.
4) By verifying the client the authorization server approves the authorization grant, and issues an access token if it is valid.
5) With the help of resource server the client sends the requests for the protected resource and authenticates by giving the access token.
6) The resource server verifies the access token and serves the request.

3. Related work

Francisco and Karen differentiated OAuth and Open ID as a double-redirection protocol. In the first Redirection, the browser gets access permission to a third-party authorization end point. The second Redirection verifies whether user is vulnerable to the basic attack like Phishing or not and enables the web browser to call-back endpoint of the application. If the call-back point is not protected by Transport Layer Security Layer and provide security with the help of TLS mechanism [1].

Urena et al has undergone a study on the risks involving in the Open ID Single sign-on mechanism .They found that Open ID authentication protocol completely depends on URL parameters where there is no privacy for the Application Users. We can get access to these URLs due to Hypertext Transfer Protocol. The CSRF attacks and account traffic analysis can be forbided in the HTTP request header by adding Referer field. Generally, hackers make use of this referer field in order to figure out user information that is passes in the delegated-based authentication and authorization protocol. Urena and Busquiel proposed a solution to this vulnerability i.e. in HTTP request header we disable the Referer field, by redesigning the Open ID protocol and stopping usage of external resources extremely [2].

Pai et al by using knowledge flow analysis, he clarified the OAuth protocol with help of some formulas using predicate logic. These formulas show that protocol has security vulnerabilities in the user's login credentials. We can retrieve the password which is stored by the client which is used to access the application that is stored in the software package Using reverse engineering in which there is a huge vulnerability [3].

Bansal et al. using applied pi-calculus explained 2.0 protocol profiles of the OAuth and by using ProVerif introduced by Webspi which handles all the attacks related to web applications, he verified them. The web application has three parties such as Client, Client Agent, and Server. Webspi consists of some processes such as 1. Server site process 2.Client site process 3.User process an attacker model to create different attacks in which the attacker uses commands. Attackers can be found by Managing principles, Network attackers, and website attacker's categories. Webspi model identifies various attacks like Automatic CSRF, Sharing CSRF, and Social Login CSRF. OAuth has token redirection attacks.Using the Universal Compatibility Security Framework Chari et al. presented a detailed report on the OAuth protocol 2.0 authorization code by implementing server-side authorization. User App and the authentication server are encrypted with the help of SSL functionality which removes the session fixation and swapping attacks [4].

4. OAuth mechanism in internet of things

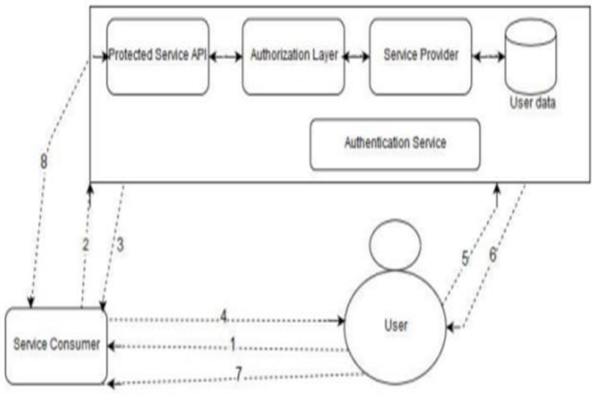
The main drawback for this mechanism is that all the approved users use the IOT networks. In order to solve these problem only legitimate users should have access to the particular IOT network. This is an authentication mechanism which can be done with the help of Security Access Manager who compares the user ID obtained from the service provider using access token and then permits the user to access the IoT network. By this approach unauthenticated users cannot use the IOT network. The main approach is to make use of Authentication and Authorization process.

First create the information in a Security Access Manager which is made using the access token from the service provider to induce friend list of constrained network manager account. When a user tries to access the IoT network it is redirected to security access manager that performs the authentication method. In authentication method user ID is obtained by OAuth protocol.

Here we are created a database for the User ID's with the help of the friends list of IOT network Manager. Security Access manager requests for the refresh token before the expiration of access token in order to maintain the synchronization between friend list and database moreover IoT network manager has an choice to login to security access manager application for instant synchronization between them. Data is updated with the help of the Refresh token.

5. Authentication scheme for fire alarm system

Fire alarm system send associate alert message when there is any danger encountered. The temperature sensor is connected to it which will enable the buzzer to detect the high temperature the notification is going to be sent to the Social Media networking Sites like Twitter, Facebook and Gmail.



The important components present in the Fire Alarm System are:

1) Service Consumer: He receives the information of the user from the service provider in place of resource owner by requesting token to the user. Data is first sent form this only to the authentication server.
2) Authentication Service: Both Client and service provider authenticates and exchanges the information with each other.
3) Protected Services: The information will move through the authorization process in order to check whether the user credentials are true or not.
4) Service provider: Service Provider sends the access token to the consumer who in turn sends to the Service Consumer and service provider directly communicates with the service consumer.
5) Dotted Lines: These lines are generally imaginary lines which are used for the authentication services from the service provider.

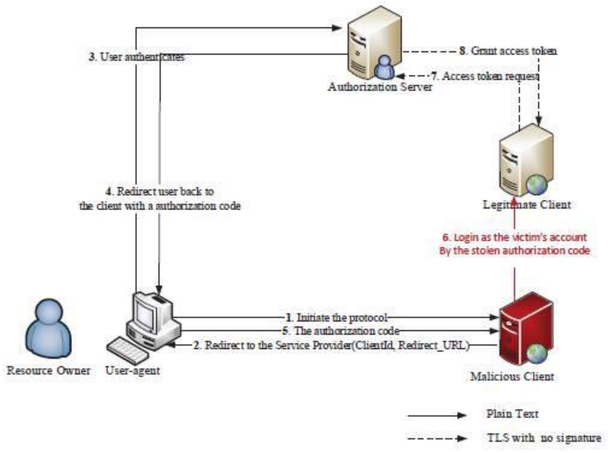
6. Security analysis of OAuth protocol

Here, we have analysed four common attack models that lead to failure of OAuth 2.0 protocol. The attack models are made based on the following assumptions

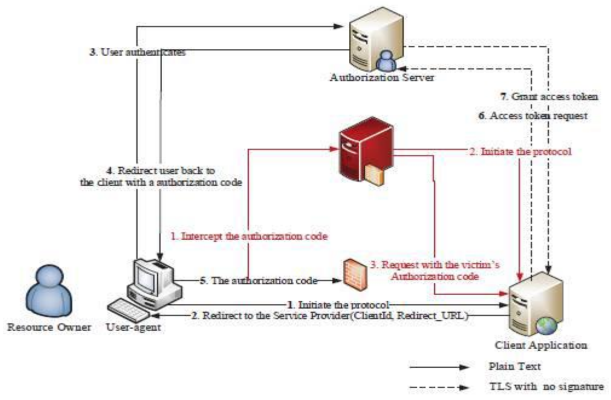
In the OAuth 2.0, the authorization code flow has an authorization code which provides access grant to the client by acting as resource owner. Once the authorization code will obtain client application, it has to be avoided. Any authorization code is used only once. If this is not the case, the replay attack can possible. In this scenario, an unknown attacker is able to catch the authorization code redirection request during the transmission between the client application and the resource owner's user-agent. Afterward, the assailant will resend previous request to the client application to gain access to the accounts linked to the authorization code. Here these are the some of the attacking modules. They are:

1) Phishing.
2) Replay.
3) Impersonation.

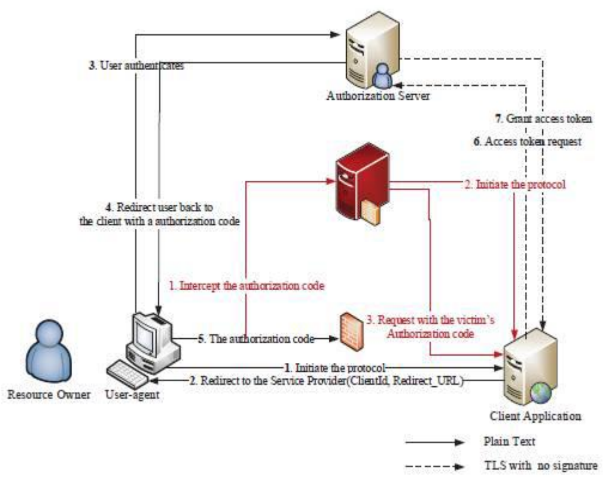
1) Phishing attack module: It is a method while the attacker masquerades as a trust entity to achieve username password access code and alternative security info. In this attack the attacker disturbs DNS cache to hack user's desktop or computer. So, whenever the victim tries to travel to some licensed websites, user is taken to a illegal client application rather than the website that isn't meant to visit. As this attack mainly seen on social networking techniques which are used for sending emails or other communication methods like SMS text messages and other different messaging modes. The following are some types of phishing attacks. Deceptive Phishing, Malware-Based, Key loggers and Screen loggers, Session Hijacking, Web Trojans, Whaling, Clone Phishing, Covert Redirect, Social Engineering, Phone Phishing.


3. User authenticates
8. Grant access token
7. Access token request
Authorization Server
4. Redirect user back to the client with a authorization code
Legitimate Client
6. Login as the victim's account By the stolen authorization code
1. Initiate the protocol
5. The authorization code
2. Redirect to the Service Provider(ClientId, Redirect_URL)
Resource Owner
User-agent
Malicious Client
Plain Text
TLS with no signature

2) Replay attack module: Replay attack module is otherwise called as playback attack. It is a kind of network attack used to avoid the attack is digital signatures with timestamps and generates session keys randomly, Time bound and process bound, One-time password for every request and including order of messages and discarding messages which are duplicate. The authorization code will be used only once. An attacker who is not known to the user is able to take the authorization code redirection request during the transmission between the client application and the resource owner's user-agent. Afterward it will resend the previous request to customer application to gain access to the accounts linked with the authorization code.


3. User authenticates
Authorization Server
7. Grant access token
6. Access token request
2. Initiate the protocol
4. Redirect user back to the client with a authorization code
1. Intercept the authorization code
3. Request with the victim's Authorization code
5. The authorization code
1. Initiate the protocol
2. Redirect to the Service Provider(ClientId, Redirect_URL)
Resource Owner
User-agent
Client Application
Plain Text
TLS with no signature

3) Impersonation attack module: In this attack an unknown user may impersonate a legitimate user to call other legitimate users on the network. The protection penetrability that the recall termination of a consumer application is an advantage of impersonation attack. In this the attacker first eavesdroppes the authorization code than it assailant the block native request to keep the authorization where should use only once. The consumer application starts the authorization flow and sends the fake request with the purloined authorization code.


3. User authenticates
Authorization Server
7. Grant access token
6. Access token request
2. Initiate the protocol
4. Redirect user back to the client with a authorization code
1. Intercept the authorization code
3. Request with the victim's Authorization code
5. The authorization code
1. Initiate the protocol
2. Redirect to the Service Provider(ClientId, Redirect_URL)
Resource Owner
User-agent
Client Application
Plain Text
TLS with no signature

7. Conclusion

The objective of OAuth 1.0 and OAuth 2.0 are different from each other. Generally, the primary version of the protocol allows users to permit third-party application access to protected resources without divulging their login details to the third party whereas the next version of the protocol removed several security features to make the implementation of the protocol simple and easy. The removal of signature necessities may be a serious security concern that makes the execution of attacks in network very easier [6].

Here, we described about main causes of security attacks in each phase of execution of the OAuth protocol. The model presented in this paper discusses about many general network attacks that would be probably dispensed by hackers to imitate users and access or authenticate their protected resources and attacks like replay attacks, network eavesdropping, impersonation attacks and forced-login CSRF attacks and we further investigated some details about the root causes of this attacks and founded that are occurred mainly because of (1) No demand of using TLS protection on decision back endpoints; (2) Disables the authorization server to validate the authenticity of the client application by removal of the signature; (3) Multiple uses and no authenticity of authorization codes (4) Validation of redirection URIs mechanism is not custom made to the protocol [5].

In this paper, our analysis of OAuth protocol was mainly based on the communication which is between the authorization server and client-agent and conjointly between the customer application and client agent.

References

[1] Feng Yang, Sathiamoorthy Manoharan, "A security analysis of the OAuth protocol", IEEE, available online: https://www.scribd.com/document/267173600/Yang-2013.

[2] Manuel Urueña, Alfonso Muñoz and David Larrabeiti, "Analysis of privacy vulnerabilities in single sign-on mechanisms for multimedia websites", Springer, Multimedia Tools and Applications (2014) pp. 159–176, available online: https://earchivo.uc3m.es/bitstream/handle/10016/20008/analysis_MTA_2014_ps.pdf?sequence=1.

[3] Suhas Pai, Yash Sharma, Sunil Kumar, Radhika M. Pai, Sanjay Singh, "Formal Verification of OAuth 2.0 Using Alloy Framework", IEEE, available online: http://ieeexplore.ieee.org/document/5966531/.

[4] Chetan Bansal, Karthikeyan Bhargavan and Sergio Maffeis, "Discovering Concrete Attacks on Website Authorization by Formal Analysis", IEEE, available online: http://ieeexplore.ieee.org/document/6266164/.

[5] E. Hammer-Lahav, "The OAuth 1.0 protocol", available online: http://www.rfc-editor.org/info/rfc5849.

[6] J. Richer, W. Mills and H. Tschofenig, "OAuth 2.0 message authentication code (MAC) Tokens," November (2012), available online: https://tools.ietf.org/pdf/draft-ietf-oauth-v2-http-mac-02.pdf.

[7] Renzo E. Navas, Manuel Lagos and Laurent Toutain, "Nonce-based Authenticated Key Establishment over OAuth 2.0 IoT Proof-of-Possession Architecture", IEEE, available online: http://ieeexplore.ieee.org/document/7845424/.