Modeling and Assessing OAuth 2.0 under PoP (Proof of Possession) for Secrecy

[1]Bharatkumar K. Talaviya, [2]Namrate Shroff
[1]Student, [2]Assistant Professor
[1,2]Computer Science and Engineering,
[1,2]Government Engineering College, Sector-28, Gandhinagar

***Abstract* - OAuth 2.0 decides to combine the implementation and experience of delegated authentication into a single communication protocol. The OAuth protocol allows applications to access protected resources from resource server via application programming interface, without acquiring users to break their service provider credentials to consumers. OAuth is based on generic methodology for API authentication. In this work, we assess the different OAuth security approach and formalize the protocol using proof of possession architecture. The proof of possession gives some hope that the days of relying primarily on passwords and access tokens may be behind us within a few years.**

***Index Terms* - Authorization, Information Security, OAuth, Proof of Possession (PoP).**

I. Introduction

Open APIs are one of the driving forces behind the modern web. Entire eco-systems of third party applications have grown around social media, content services and shopping websites resulting in more engaging and connected web experiences. In return, services with open APIs have benefitted by increases in the number of users and creating new revenue streams.

OAuth is an open protocol to allow secure API authorization in a simple and standard method from desktop, web and mobile applications. OAuth is a security protocol that enables users to grant third-party access to their web resources (e.g. profile information) without sharing their authentication credentials with the third-party. Applications (called “clients” in the OAuth specification) have their own credentials and communicate with an authorization server to acquire an “access token” which is then used to authenticate requests to an API server.

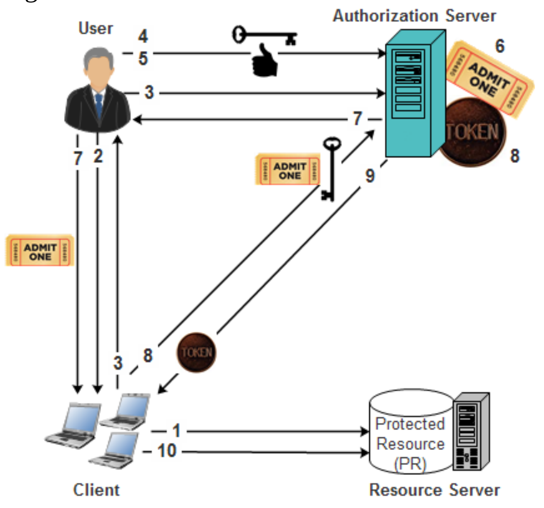
II. Players of OAuth 2.0

OAuth 2.0 gives client application a delegated access to resource server on interest of a resource owner. OAuth introduces procedure for resource owners to authorize third party access to server protected resources without sharing their credentials. Followings are four players of OAuth.

1) Client: Client is application that attempts to get access to user account. Before doing, client gets the permission form user.
2) Resource Server: Resource server accesses the user information
3) Resource Owner: Resource owner gives the access to some portion of their user account.
4) Authorization Server: After authenticating resource owner, Authorization Server issues access tokens to the client and obtaining authorization.

III. OAuth 2.0 Protocol Flow

Figure 1 outlines OAuth Protocol Flow.



OAuth 2.0 Protocol Consists of following steps:

1) Connect the Client to the Protected Resource.
2) End User initiates the Client action.
3) Client redirects User to Authorization Server.
4) User authenticates to Authorization Server.
5) User authorizes Client
6) Authorization Server issues authorization code.
7) Authorization Server redirects User to Client.
8) Client sends code to Authorization Server.
9) Authorization Server issues token(s).
10) Client accesses Protected Resource.

IV. Existing OAuth Security Assessment Approach

Several assessment approaches is used to examine the security of OAuth 2.0

1. Alloy Framework: Alloy Framework uses alloy specification language and alloy analyzer. OAuth model in Alloy Framework works as follow: First both the principals (participants) and values (unit of knowledge) are modeled as first class citizens i.e. both these sets of entities are represented using the sig (similar to class in OOL). Second, two abstract sig's principal and value are declared. Third, abstract sig principal is extended by client, authorization server, resource server and resource owner. Here, the intruder is one or more of the four principals themselves, and intruder is not modeled as a separate individual [2].

2. ProVerif framework: ProVerif framework uses proverif specification language and follows pi-calculus. OAuth in ProVerif Framework consists of an unbounded number of users and servers. Each user browse any trusted or malicious website but only sends secret data to trusted sites. Following applications are hosted by server.

Login: An application models form based login and consists of two processes, LoginApp for server process and LoginUserAgent for user agent process.

DataServer: An application models resource servers and consists of two process, DataServerApp with two functions getData and storeData, and DataServerUserAgent for modeling the behavior of users.

OAuthAuthorization (User-Agent Flow): A third party application models user agent flow of OAuth protocol and consists of two processes, OAuthImplcitServerApp for authorization servers, and OauthUserAgent for resource owners.

OAuthAuthorization (Web Server Flow): A third party application models web server flow of OAuth protocol and consists of two processes, OAuthExplicitServerApp for authorization servers, and OauthExplicitClientApp for clients [3].

3. Proverif Framework with WebSpi Library: Users and Servers are the principals or agents of our model. Users contain credentials to authenticate with respect to a specific web application identified by a host name and by a path in Credentials table. Servers contain private and public keys to implement TLS secure connections in serverIdentities table.

Table: Credentials (host, path, principal, id, secret)

Table: serverIdentities (host, principal, public, private)

Credentials and serverIdentities tables are private to the model and represent a preexisting distribution of secrets (passwords and keys). They are populated by the process CredentialFactory that provides an API for the attacker (explained later) to create an arbitrary population of principals and compromise some of them. The process WebSurfer models a generic user principal that wants to browse a public URL [3].

4. CryptoVerif Framework: CryptoVerif framework uses Blanchet Calculus for formalizing OAuth 2.0. In the computational model, Blanchet calculus is used to assess OAuth 2.0 protocol with CryptoVerif mechanized tool. In this framework, non-injective and injective correspondences are used to model the authentication from authorization server to end user and from authorization server to client. [4]

non-injective correspondence:

event authorization(x) → user(x) ;to authenticate end-user by authorization server.

injective correspondence:

event authorization(x) → client(x) ;to authenticate client by authorization server.

5. Universal Composability (UC) Security Framework: In this framework, an ideal third party functionality pass inputs by all protocol participants, and then computes the requisite function, and finally each of the protocol participants get portions of computed function. Security is based on simulation argument which shows adversary can obtain the same information in the apotheosis third based protocol, it can get any information in the real world implementation. In the ideal functionality definition of OAuth, there are three parties named, User (resource owner), Provider (HTTP Service), and Consumer (third-party). Here assume that the Provider and the Consumer have globally known names, e.g. public keys with certified domain names in PKI. Only the User will not be assumed to have a global name [5].

6. POAuth (Privacy Enhancing Open Authorization) Architecture: Trusted Computing is used to protect sensitive data and to secure cryptographic keys from eavesdropping. Trusted Computing Group (TCG) defines a hardware chip called Trusted Platform Module (TPM) with cryptographic mechanisms. TPM stored stores digests in shielded memory locations called Platform Configuration Registers (PCRs) using Secure Hash Algorithm (SHA-1). Upon first activation, the TPM generates a Storage Root Key (SRK). This non-migratable key is always present to secure underlying structure of TPM. In TPM, endorsement, storage, sealing, binding and attestation identity keys provide different security functionalities. The private part of the SRK itself never leaves the TPM and is thus protected by hardware. To enable device specific authorization, we use the binding construct with trusted computing. The protocol is extended to device specific authorization to enhance privacy, called POAuth [6].

V. Proposed Flow

Our analytic methodology is Proof of Possession (PoP) architecture to assess the security of OAuth 2.0 protocol. Proof-of-possession is a means of proving that a party sending a message is in possession of a particular cryptographic key. This is used as a proof that the correct party is sending the message, under the assumption that only that sender has possession the key [11].

Figure 2 shows the steps for Proposed Flow.

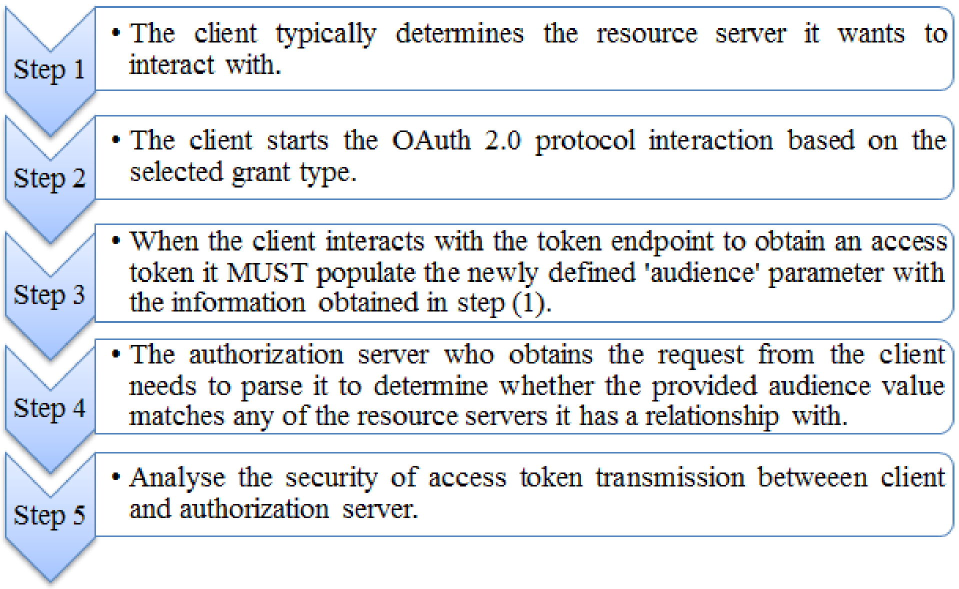

Step 1
• The client typically determines the resource server it wants to interact with.
Step 2
• The client starts the OAuth 2.0 protocol interaction based on the selected grant type.
Step 3
• When the client interacts with the token endpoint to obtain an access token it MUST populate the newly defined 'audience' parameter with the information obtained in step (1).
Step 4
• The authorization server who obtains the request from the client needs to parse it to determine whether the provided audience value matches any of the resource servers it has a relationship with.
Step 5
• Analyse the security of access token transmission betweeen client and authorization server.

Figure 2: Proposed Flow

VI. Implementation Flow

Steps involved in implementation are as under:

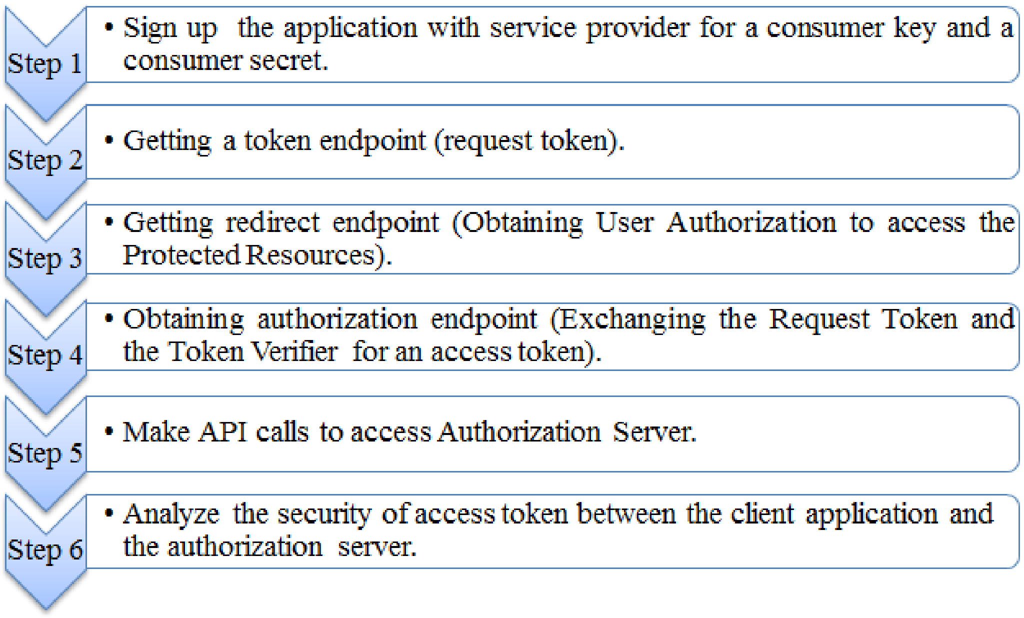

Step 1
• Sign up the application with service provider for a consumer key and a consumer secret.
Step 2
• Getting a token endpoint (request token).
Step 3
• Getting redirect endpoint (Obtaining User Authorization to access the Protected Resources).
Step 4
• Obtaining authorization endpoint (Exchanging the Request Token and the Token Verifier for an access token).
Step 5
• Make API calls to access Authorization Server.
Step 6
• Analyze the security of access token between the client application and the authorization server.

Figure 3: Implementation Flow

VII. Conclusion

OAuth is secure and efficient method for authorizing third party applications without releasing a user's access credentials. There are different security approaches available for verifying the OAuth protocol like Alloy framework, ProVerif framework, ProVerif Framework with WebSpi Library, CryptoVerif Framework, Universal Composability (UC) Security Framework and POAuth (Privacy Enhancing Open Authorization) Architecture. In my work, OAuth 2.0 is implemented through PoP model for providing the benefits such as access token reuse and TLS channel binding support.

References

Book:

[1] Ryan Boyd (O'Reilly) , "Getting started with OAuth 2.0"

Papers:

[2] Suhas Pai, Yash Sharma, Sunil Kumar, Radhika M Pai and Sanjay Singh. "Formal Analysis of OAuth 2.0 using Alloy Framework" 2011 International Conference on Communication Systems and Network Technologies. 978-0-7695-4437-3/11, 2011 IEEE DOI 10.1109/CSNT.2011.141

[3] Chetan Bansal, Karthikeyan Bhargavan and Sergio Maffeis. "Discovering Concrete Attacks on Website Authorization by Formal Analysis" 2012 IEEE 25th Computer Security Foundations Symposium, 2012 IEEE 10.1109/CSF.2012.27

[4] Xingdong Xu, Leyuan Niu and Bo Meng. "Automatic Verification of Security Properties of OAuth 2.0 Protocol with CryproVerif in Computational Model" 2013 Asian Network for Scientific Information. Information Technology Journal 12 ISSN 1812-5638 /DOI:10.3923/itj.2013.2273.2285

[5] Suresh Chari, Charanjit Jutla and Arnab Roy. "Universally Composable Security Analysis of OAuth v2.0" IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

[6] Mohammad Nauman, Sohail Khan, Abu Talib Othman, Shahr ulniza Musa and Najeeb Ur Rehman. "POAuth: Privacy-aware Open Authorization for Native Apps on Smartphone Platforms" 2012 ACM 978-1-4503-1172-4

[7] Michael B. Jones, "The Increasing Importance of Proof-of-Possession to the Web" W3C Workshop on Authentication, Hardware Tokens and Beyond

[8] Feng Yang and Sathiamoorthy Manoharan, "A security analysis of the OAuth protocol" 978-1-4799-1501-9/13/$31.00 2013 IEEE

Web References:

[9] E. E. Hammer-Lahav. The oauth 2.0 authorization protocol draft-ietf-oauth-v2-13. [Online]. Available: http://tools.ietf.org/html/draftietf-oauth-v2-13

[10] By Jakob Jenkov, "Tutorial on OAuth 2.0". Available: http://tutorials.jenkov.com/oauth2/index.html

[11] T. Lodderstedt, M. McGloin, and P. Hunt. Oauth 2.0 threat model and security considerations. [Online]. Available: http://tools.ietf.org/html/draft-ietf-oauth-v2-threatmodel-01

[12] Alloy: a language & tool for relational model [Online]. Available: http://alloy.mit.edu/alloy/index.html