STUDY OF FACEBOOK'S APPLICATION ARCHITECTURE

By

NATARAJ SUNDAR

A THESIS

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2011

Approved by:

Major Professor
Xinming Ou

Copyright

Abstract

Facebook is a social networking service launched in February of 2004, currently having 600 million active users. Users can create a personal profile, add other friends, and exchange messages and notifications when they change their profile. Facebook has the highest usage among all social networks worldwide. It's most valuable asset is access to the personal data of all its users, making the security of such data a primary concern. User's data can be accessed by Facebook and third parties using Applications. "On profile" advertisement in Facebook is a classic example of how Facebook tailors the advertisements a user can see, based on the information in his profile. Having prioritzed user friendlines and ease of use of the Applications over the security of the user's data, serious questions about privacy are raised.

We provide here an in-depth view of the Facebook's Application Authetication and Authorization architecture. We have included what, in our opinion, are the positives and negetives and suggested improvements. This document takes on the role of the User, the Application and Facebook server at appropriate points.

Table of Contents

List of Figures

Acknowledgements

Special thanks to Dr.Simon for the encouragement and advice to explore this field. I also thank all my collegues in our Network Security research group Argus for their help.

Dedication

Dedicated to my parents and family.

Preface

Online Social networking is a phenomenon that started in the 90's. It really picked up steam in the last decade. Almost all current Social Networks started after 2000, Facebook being a forerunner among them. During the Summer of 2007, Facebook made their Platform API openly available and through this enabled 3rd party application builders and websites to leverage Facebook's user base. Application creators could now obtain access to the user's data and the data of the user's friends who are a single hop away in the user's network graph. Farmville the most used application on Facebook has a user base of 83 million followed by Birthday cards and Cafe World with 47 and 30 million respectively. Loooking at such numbers one could argue that introduction of an application platform has a positive impact on the no of users using Facebook. There are Facebook platform developers from 190 countries, with users installing 20 millions applications every day. Every month, more than 250 million people engage with Facebook on external websites. An average of 10,000 new websites integrate with Facebook every day, since social plugins launched in April 2010. More than 2.5 million websites have integrated with Facebook

For Authentication of the end user and the application, and to authorize the application to access the user's data, Facebook uses Open Authention(OAuth)version 2.0.
OAuth ensures the identity of the user with a login and the identity of the application with a symetric key based secret. It helps the application to request the user for permission to access his profile information. Facebook applications are developed using their Software Development Kit which supports server side scripting(PHP) and client side scripting(JavaScript). API calls from the SDK enable the application to access various parts of the user's profile.

Facebook doumentation does not provide enough information about the authentication and authorization process, its intricacies and protocols. In this report we've tried to address this issue by building, monitoring and analyzing Facebook applications. The first chapter gives an overview of the modified version of Open Authetication as implemented by Facebook.
The second chapter goes into details and explain this process step by step as we learned while building and analyzing our own applications. The third chapter is about Access Tokens, a key part of the Open Authentication process. The concluding chapter provides suggestions for improvements.

Chapter 1 - Overview of Facebook's Open Authentication Model

Facebook has taken the standard Open Authentication 2.0 (OAuth) system and modified the same suit its requirements. OAuth protocols are used in practice in 2 different ways depending on where the application code resides, server-side or the client side. Although the generic steps remain the same in both cases, the implementation is different. Overall an Application strives to get an access token which it can use to request and control user's data.

This chapter goes into the details of the generic OAuth Authentication and Authorization procedure, the next chapter describes and compares the server-side and the client-side versions. In traditional OAuth there are two separate moderating servers and a resource server.

1. The Authentication server:

   This verifies the identity of the requesting application and the user.

2. The Authorization server:

   This protects the user's profile data. An application has to request this server in order to access any fields from the user's profile.

3. Resources Server:

This houses the user's information. It responds to authorized requests with JSON objects. Facebook has clubbed all the three servers into a single unit. So, in our discussion we will be addressing all of them together as A&R servers.

The generic OAuth protocol

1. Credential Check:

   The application requests the authentication and authorization server (A&R) for access to a user's information by providing the username to the server.
2. A&R Server response:

   The server responds with details required to establish contact with the authentication endpoint.
3. Request to the User:

   The application then sends an authorization request to the user requesting access to the user's resources held by the resource server.
4. User Permission Grant:

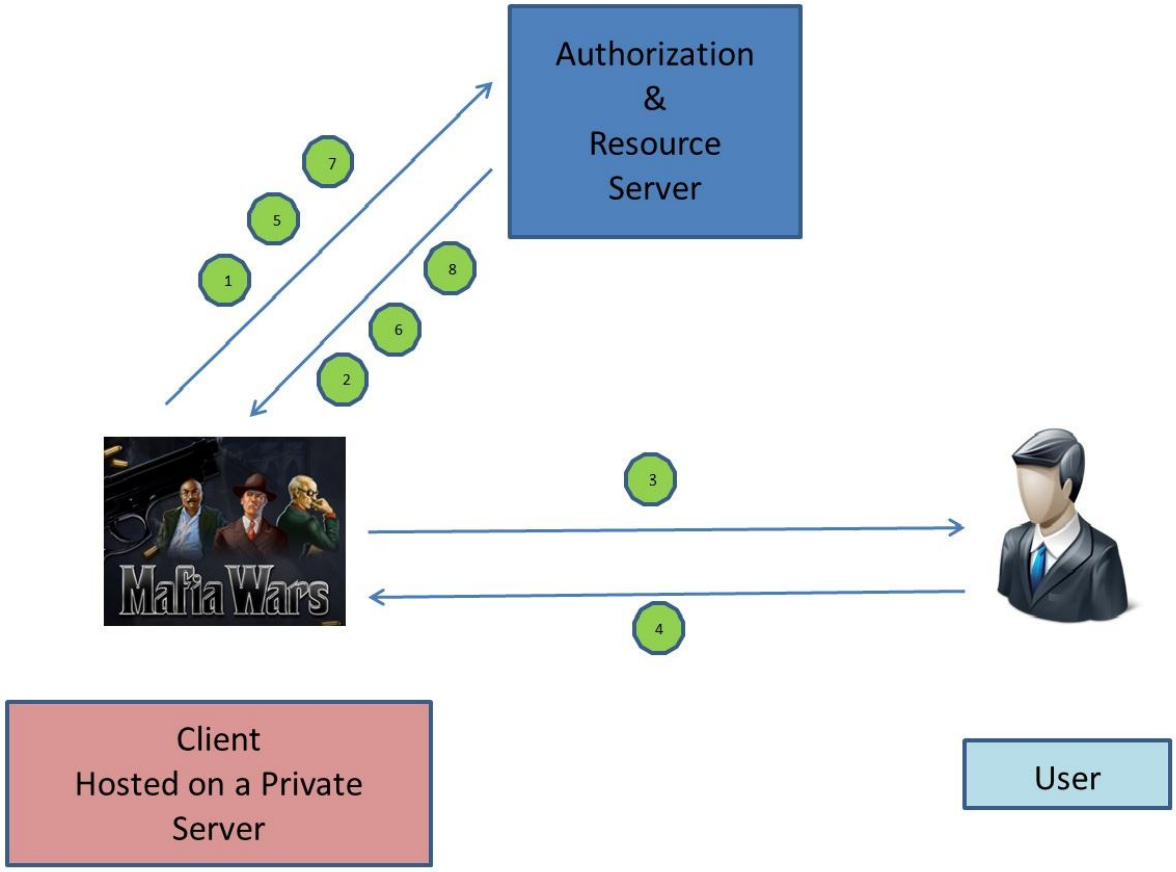
   The user agrees to let the application to access specific fields on the resource server.
5. Request to A&R Server:

   The application provides the user-access-approval and the application details to A&R servers.
6. Access Token Grant**:

   A&R server returns an access token to the application. This can be used in all future requests to obtain fields specified in the user approval, from the resource server for a period stipulated on the access token. (Chapter 3 describes the features of an access token in detail)
7. User Data Access Request :

   The application presents the access token to the A&R server (since the authorization and resource server are one and the same) asking for the authorized user data fields.
8. A&R Response with Data:

   The A&R server responds with the requested information.



Generic OAuth Protocol : Figure 1

(Numbers in the figure correspond to the steps described previously)

Brief Overview of Client-Side and Server-Side OAuth

Depending on where the application code is running, Facebook uses two different variants of the OAuth protocol previously discussed. Below is a brief summary of how the two different systems are related to the generic protocol. Client side applications make use of IFrame and Server-Side applications make use of FBML Canvas.

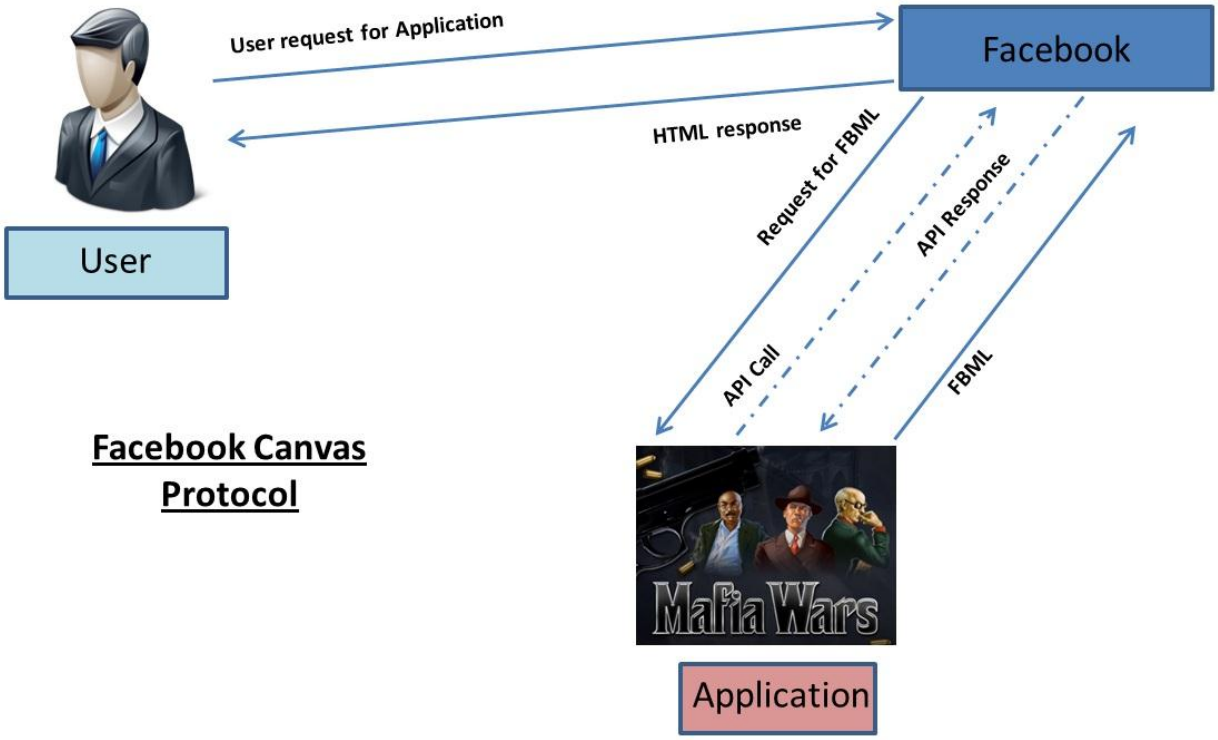
Server Side

Canvas based applications use the contents on remote application servers, they are rendered by Facebook servers.

The following steps summarize a server side authentication and authorization process.

1. When a user visits an FBML Canvas based application Facebook will send a request to the application for the content.
2. While the application is processing it may make multiple API calls to fetch the social informatics of the user.
3. Once the task is complete the content is delivered back to Facebook first and then to the user. FBML only fetches on the server-side since it does not support JavaScript.

The following figure depicts the steps,



Server Side OAuth : Figure 2

Client Side

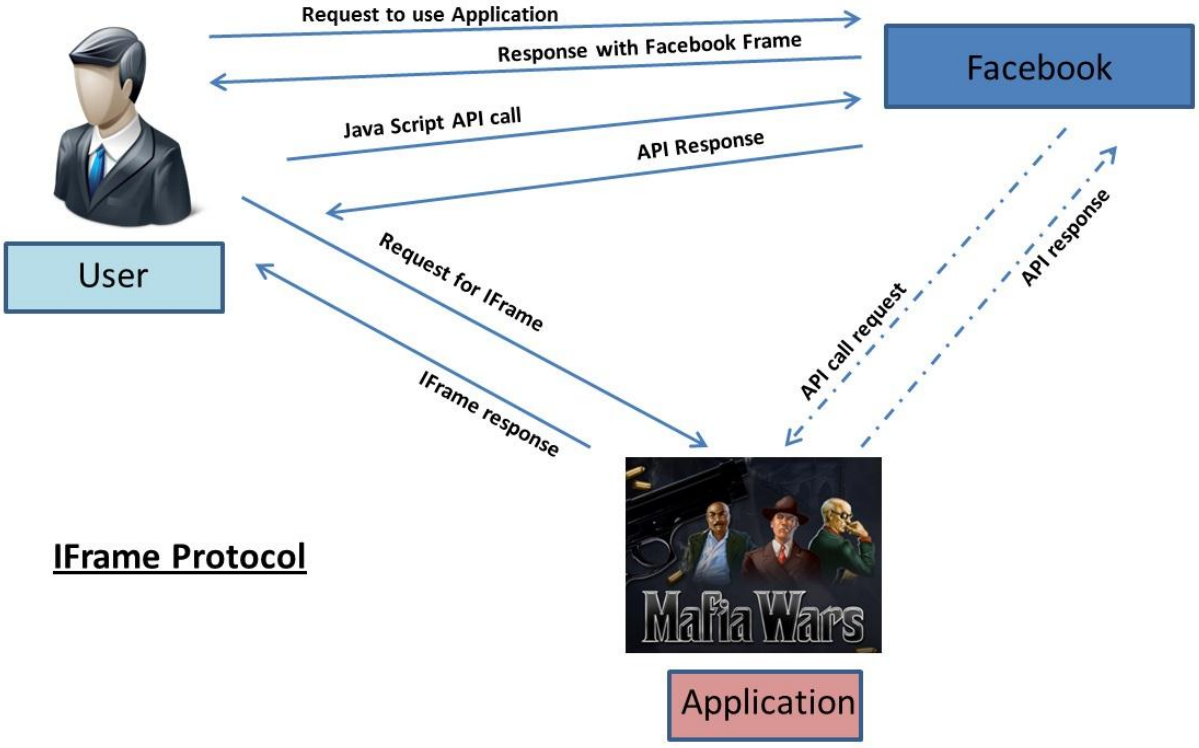
IFrame Canvas applications and Facebook Connect are directly rendered by application servers. Facebook Connect and IFrame may also fetch the social informatics on the client side with JavaScript.

The following steps summarize a client side authentication and authorization process.

1) To access an IFrame Canvas application, a user would first open a browser and navigate to the URL of the application.
2) The initial request to Facebook causes an IFrame to open in the browser for the application to display the content.
3) The browser sends another request to the application server. Similarly, the application server may make multiple API calls to fetch the social informatics from Facebook while producing the content.

4) Once completed, the content is delivered back to the IFrame in the browser directly instead of through Facebook.

Facebook connect a feature using which users can login to supporting websites with Facebook login instead of the website login works in a similar fashion. The only difference is that the display is not limited to an IFrame.



Cleint Side OAuth : Figure 3

Plus and Minus Points of Facebook's existing system

Negatives with suggested remedies

Following are few aspects of Facebook's existing system that raise concern:

1. Automatic renewal of approval for clients:

   Although each authentication token used to obtain the clients information has a expiry time (to avoid replay attacks) the token renewal after this expiry time is done without consent of the user. If the user authorizes an application once, the application has access to the user's information infinitely and can generate tokens valid for durations at its will. The application can also pass on such a token as it wishes. A better approach would be to let the user select the duration of validity for the token and after the expiry time ask the application to re-request the user for permissions. The Android application model which works in this manner can be a good example to follow.

   Every token renewal should explicitly involve the user whose data is being accessed. This way the user knows how and when his profile data is being used.

2. Scope of OAuth token:

   Access tokens should be provided with just enough scope that they need in order to implement the respective application function. The application code can be examined to determine the fields the application needs to access, in order to be fully functional.

   The scope of the token should be limited to only those essential fields. This way an application can be stopped from accessing and using data irrelevant to the application.

3. Confidentiality of tokens and sensitive data:

   Transmission of data such as the access tokens, renewal tokens, resource owner passwords, access permissions over secure transport should be enforced.

   In the current system to avoid the slight time lag induced, secure transport is disabled by default and normal non secure modes are used. Using transport layer security at least for the above mentioned types of data is essential and should be made default.

4. HTTP vs HTTPS sessions for the authentication steps:

   All the steps are executed using a HTTP session as opposed to a HTTPS session. A maliciously listener who has compromised the router for example can wait till the steps are completed and then hijack the session. Using HTTPS will prevent this.

Positives

Following are existing security features implemented by Facebook which effectively thwart different types of attacks:

1. Thwarting Token Guessing Attacks:

   A high level of randomness in token creation ensures that token handles and other secrets not intended for human use can't be guessed.

2. Effective Authentication:

   In case of web based applications, the authorization servers authenticate the client and validate that the authorization code has been issued to the correct client. If authorization server observes multiple attempts to redeem an authorization code it revokes all tokens granted based on the authorization code.

3. Prevention of Session Fixation:

   During the authorization, servers ensure that the redirect_uri used in the protocol sequence is same as the redirect_uri that is used to convert the respective authorization code into tokens. Clients must register and validate the actual redirect t_uri against the pre-registered value. In this way attacks that use the authorization code flow to get another user to log-in and authorize access on behalf of the attacker can be stopped.

(There are more such features but these stand out.)

Chapter 2 - Facebook Authentication Overview

A protocol can be broadly defined as a set of procedures to be followed when communicating. It is a system of message formats and rules for exchanging messages in or between computing systems. Protocols may include signaling, authentication and error detection and correction capabilities. A protocol defines the syntax, semantics, and synchronization of communication. Messages are sent and received to establish communications. Protocol specifies rules governing the transmission of such messages.

OAuth protocol is used for Authentication and Authorization of websites, mobile and desktop applications by the Facebook system. There are 2 ways of implementing the OAuth mechanism. Server-Side flow (using PHP for server side code) and Client-Side flow (using JavaScript for client side code).

User Sign-in Process

Server side mechanism is used when we need to call the Facebook Graph API from the web server hosting the application. The client side mechanism is used when calls are made to the Graph API from a client like JavaScript code running inside a web browser or mobile/desktop applications. Reading and writing any data from Facebook involves the Graph API. It provides a hierarchical depth first based view of the social graph. The social graph is made up of individual units like people, photos, pages etc. The graph indicates connection between the units in terms of friendships, likes and photo tags.

Graph API Object Details:

Since Graph API objects are the basic units of data that we'll be dealing with here is a brief about them.

- Every unit in a social graph has a primary key. Using this key in the request url allows access to the properties of the unit. Example: My unique id on Facebook is 665186330.

Using https://graph.facebook.com/665186330 to access my properties returns the following JSON object:


```
{"id": "665186330",    "name": "Nataraj Sundar",    "first_name": "Nataraj",
"last_name": "Sundar",    "link": "http://www.facebook.com/nataraj.sundar",
"username": "nataraj.sundar",    "gender": "male",    "locale": "en_US" }
```

This is the publicly available information about me. Recently Facebook has allowed access to an object's onfirmation using the username.

- Relationships can be obatined by following the id with a connection type as follows `https://graph.facebook.com/ID/CONNECTION_TYPE`.

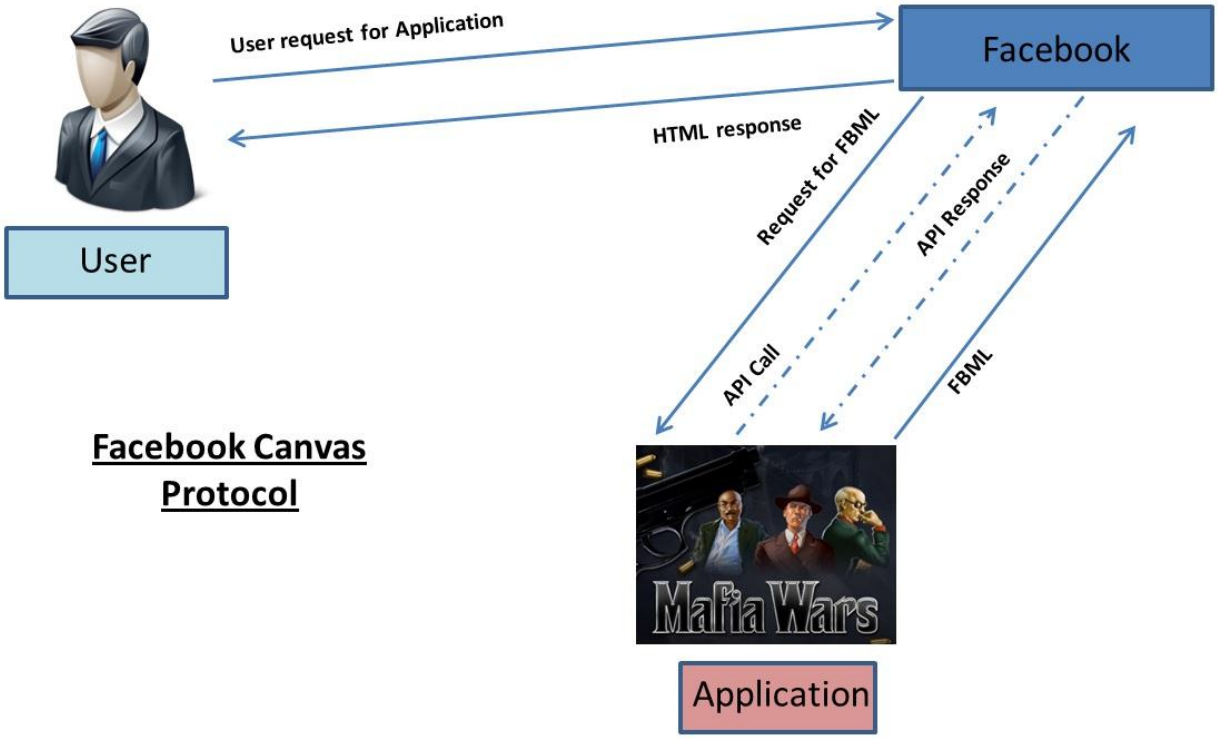
  Example: https://graph.facebook.com/665186330/friends?access_token=… returns the list of my friends in the form of a JSON array.

Back to the authorization and authentication, irrespective of which method is chosen the basic steps are the same.

a) User Authentication:
   Make sure the user is who he claims to be.
b) Application Authorization:
   Give the user control over what data and actions over the data he is allowing the application to have.
c) Application Authentication:
   Make sure the user is passing his information to the right application.

Once these three steps are completed the application is granted an access token. The access token is like having a power of attorney on behalf of the user. The application can use the token to access the user's information and perform actions on the user's behalf.

Server-side A&A Mechanism



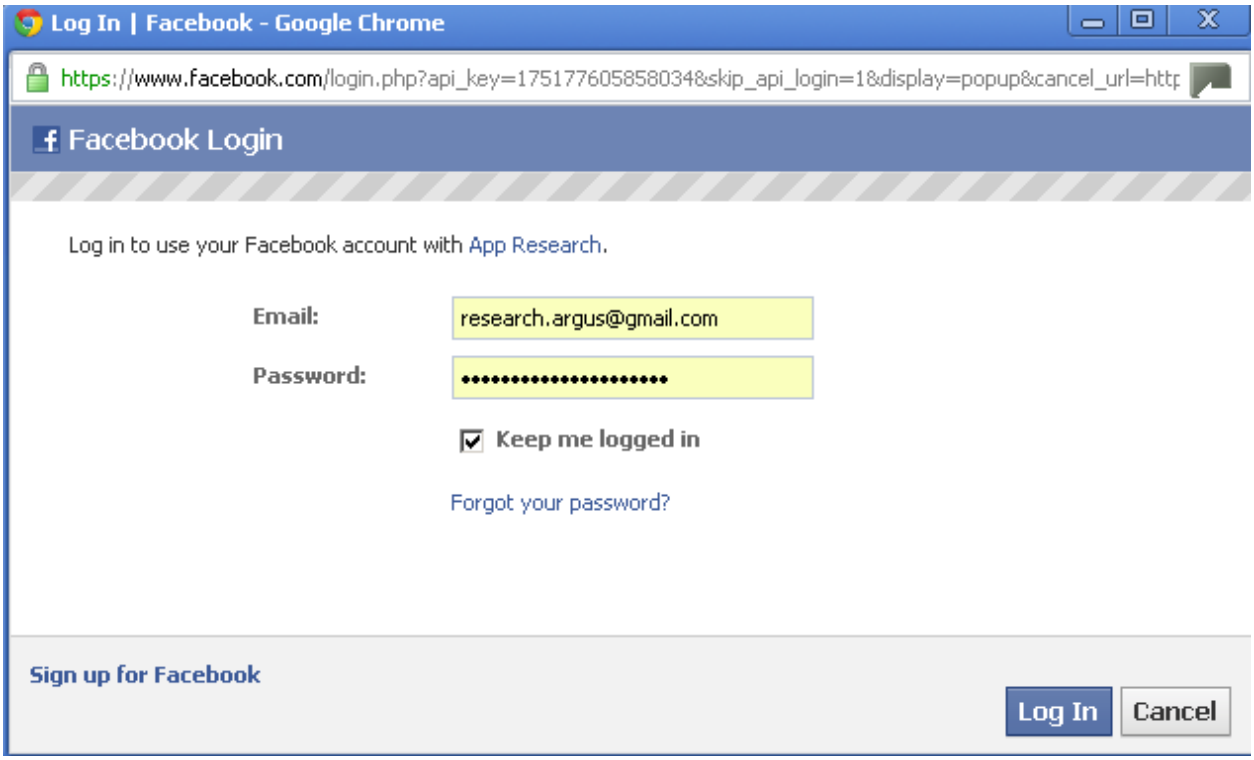
Server Side A&A : Figure 4

Authentication and Authorization are combined into one single step. The user on selecting the application is redirected to OAuth dialog. The application passes the client_id parameter that uniquely identifies the application (obtained when creating the application) and the URL to redirect the browser to once the authentication is complete (redirect_uri parameter), the redirect uri must be in the same domain as the site URL in the website specified by the application creator.

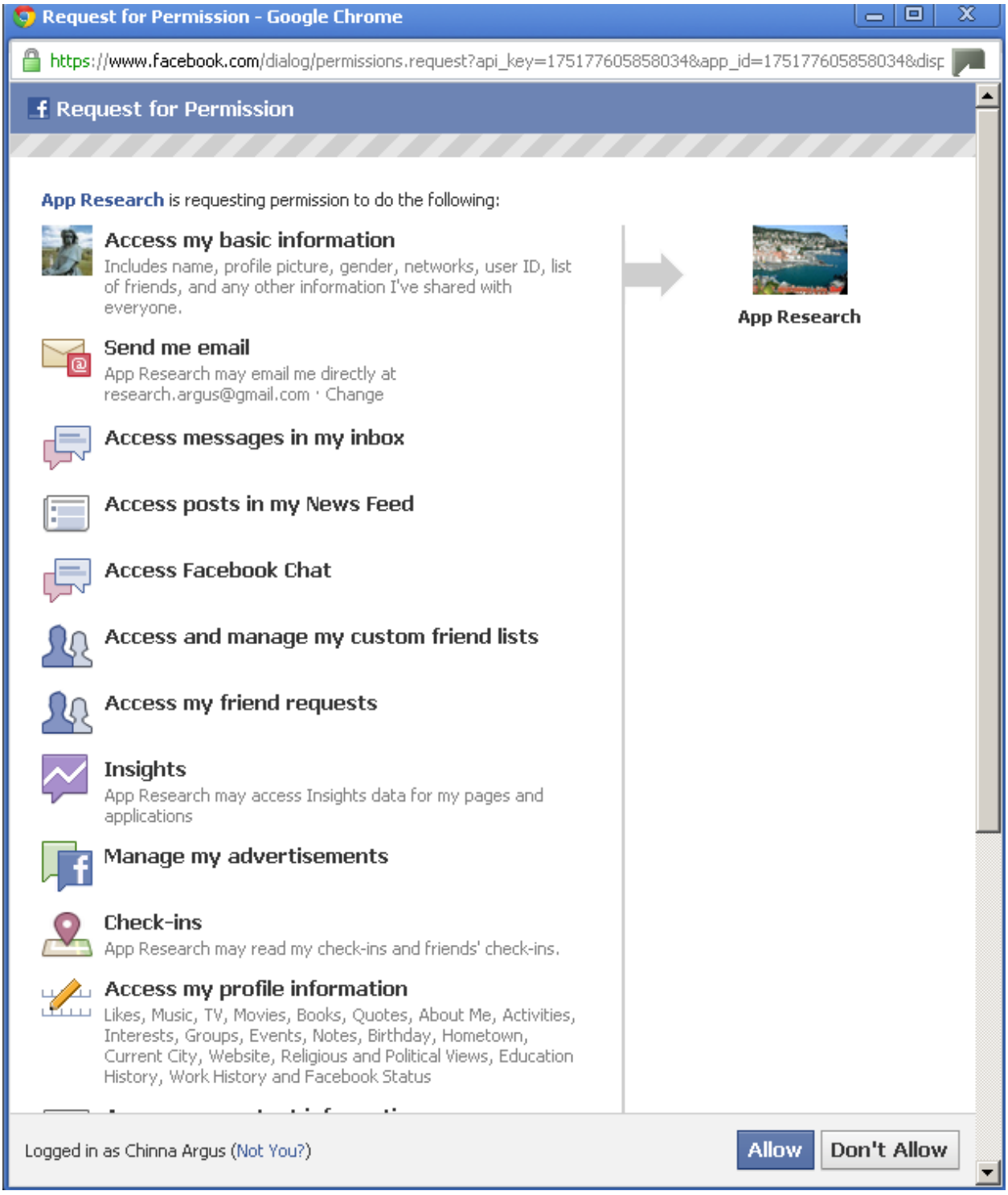
Sample format:

https://www.facebook.com/dialog/oauth/client_id=YOUR_APP_ID&redirect_uri=YOUR_URL

First the user is prompted to enter his facebook credentials. If already logged in the login cookie on the user's browser is validated.



In the next step the Oauth dialog prompts the user to authorize the application to access various parts of the user's profile information. The default is basic information that is publicly on Facebook. If the application needs more than the basic information it should request each group of permissions individually. The permissions required are provided following the URI.



Example: to request offline access and access to the user's mailbox the application would provide the following URL:

```
https://www.facebook.com/dialog/oauth?client_id=YOUR_APP_ID&redirect_uri=YOUR_URL&scope=email,read_stream
```

If the user presses **Don't Allow**, application is not authorized. The OAuth Dialog will redirect to the application provided **redirect_uri** parameter with the following error information:

```
http://YOUR_URL?error_reason=user_denied&
error=access_denied&error_description=The+user+denied+your+request
```

.

If the user presses **Allow**, your app is authorized. The OAuth Dialog will to the **redirect_uri** parameter with an **authorization code**:

```
http://YOUR_URL?code=A_CODE_GENERATED_BY_SERVER
```

With this code in hand, you can proceed to the next step, app authentication, to gain the access token you need to make API calls.

To authenticate your app you must pass the authorization code and your app secret obtained during app creation to Graph API token endpoint

```
https://graph.facebook.com/oauth/access_token?
client_id=YOUR_APP_ID&redirect_uri=YOUR_URL&
client_secret=YOUR_APP_SECRET&code=THE_CODE_FROM_ABOVE
```

If your app is successfully authenticated and the authorization code from the user is valid, the authorization server will return the access token:

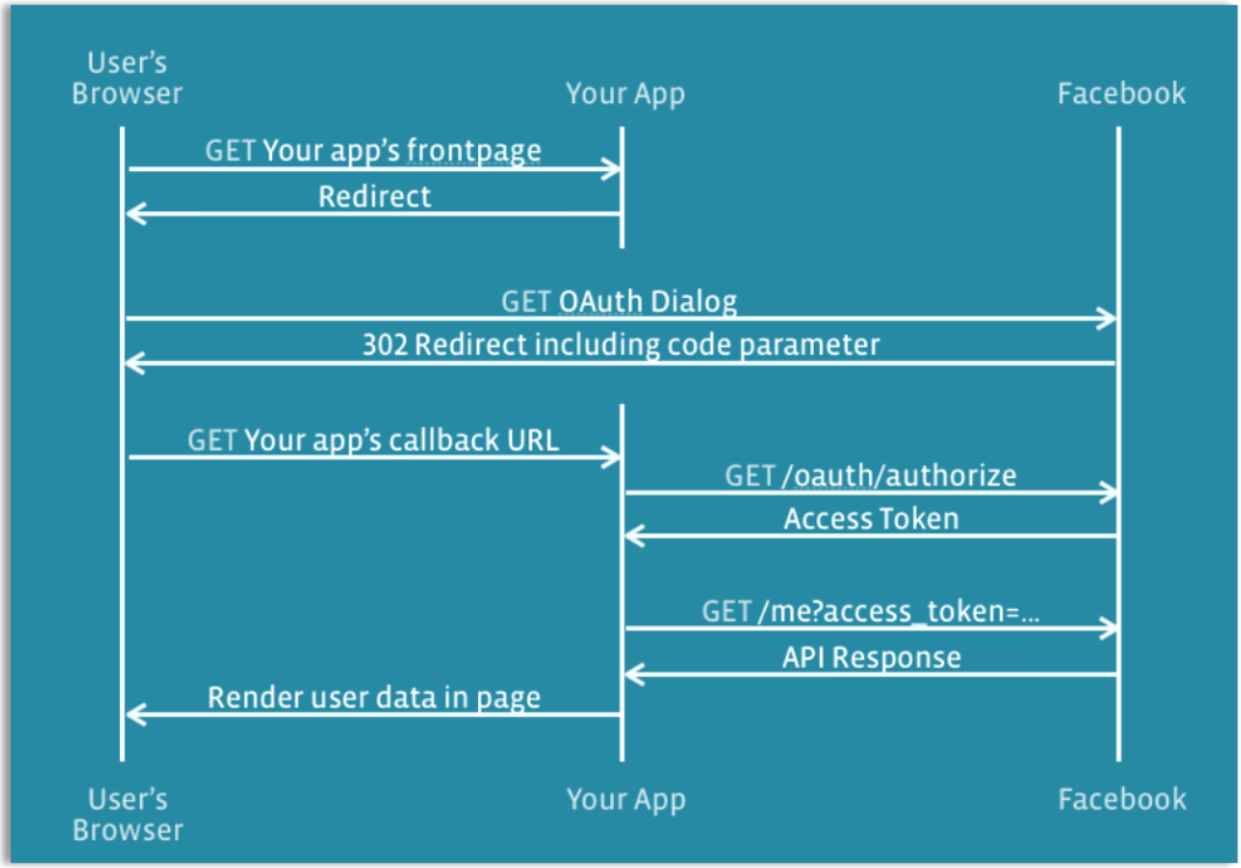
```
[access_token] => 175177605858034|2.AQAQdflPm135WCtv.3600.1311570000.1-
100002165571928|FJLJl8Pl_8RqMX1PgEtwpenK60I
```

The details about the token are given in the next chapter.

If there is an issue authenticating your app, the authorization server will issue an HTTP 400 and return the error in the body of the response:

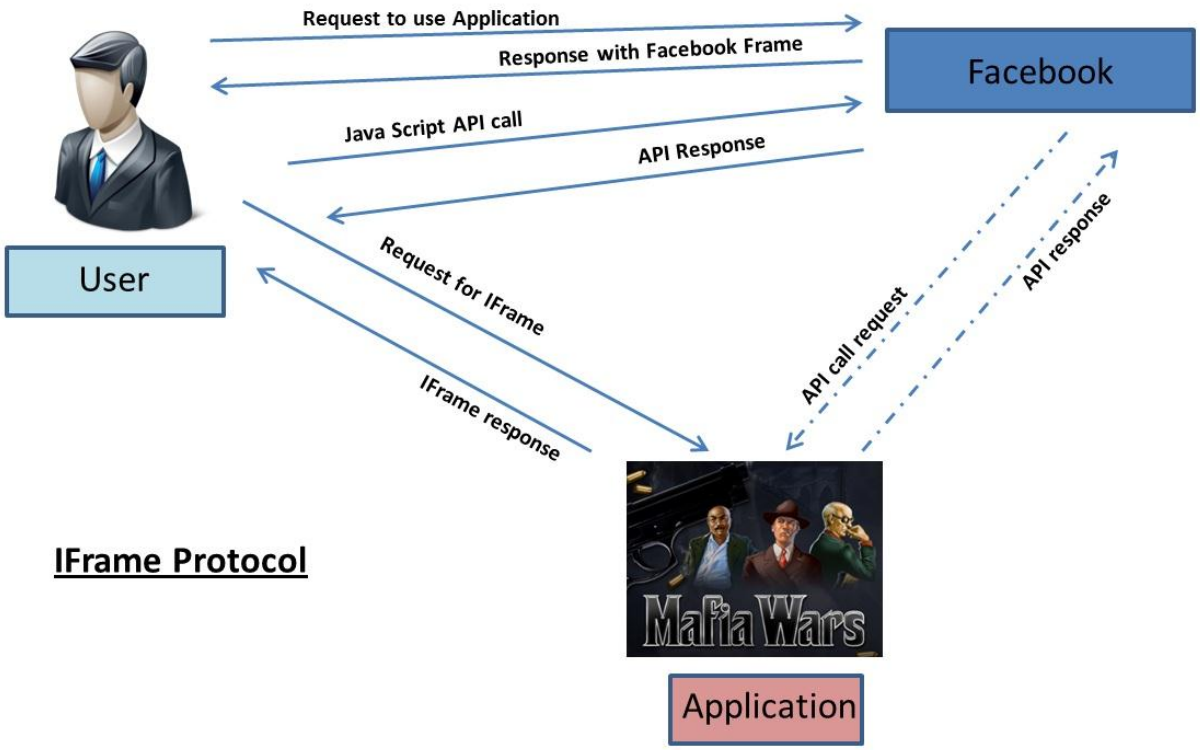
```
{    "error": {       "type": "OAuthException",       "message": "Error
validating verification code."    } }
```

```
These are the HTTP calls made during Server side flow:
```



Server Side Protocol : Figure 5

Client-side A&A Mechanism



Client Side A&A : Figure 6

The client side flow allows if selected to eliminate the role of the Facebook Server once authentication and authorization is complete. The main difference is that the application has to specify the response-type parameter with an OAuth token directly.

Example:

```
https://www.facebook.com/dialog/oauth?client_id=Application_ID&redirect_uri=URL_On_SERVER&response_type=OAuth-token
```

In the same way as server side flow extra control over data can be requested using the scope parameter before the response_type.

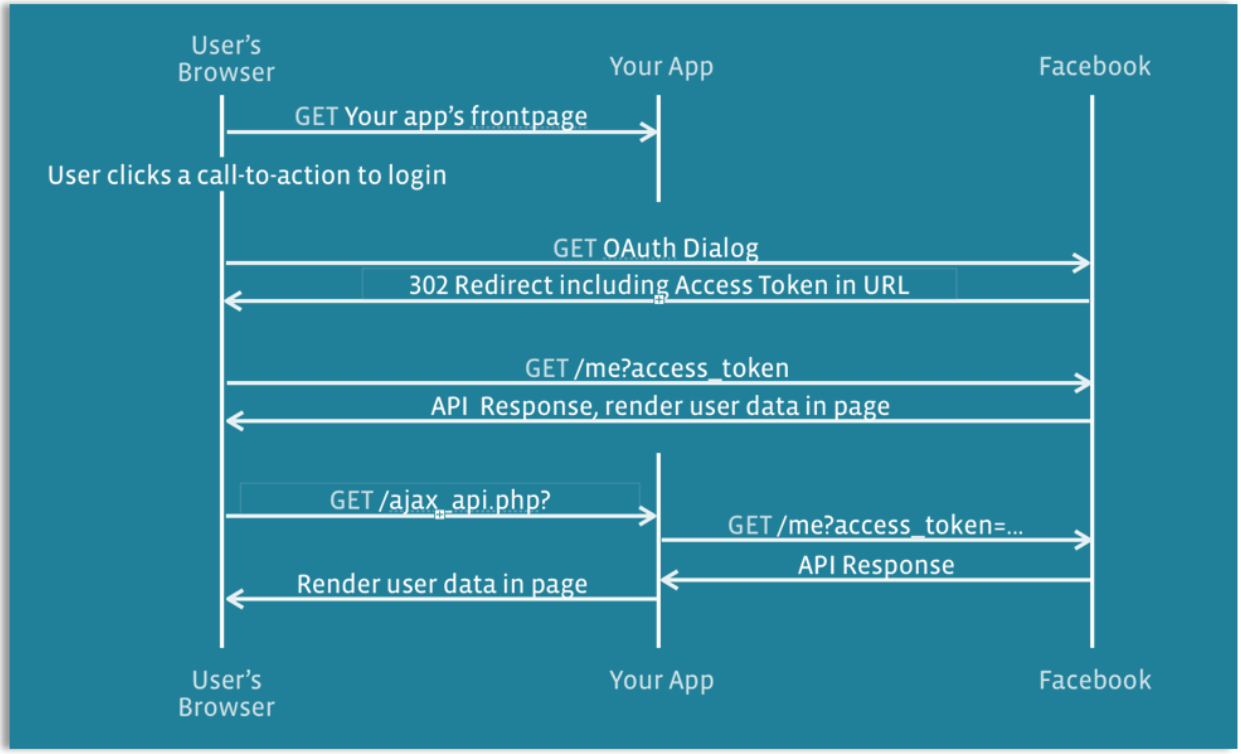
Once the process of identifying the user, applications and granting of privileges over data is complete the browser fetches the value of redirect_uri. As a URI fragment the authorization code is passed to the redirect_uri.

Example:

http://YOUR_URL#access_token=166942940015970%7C2.sa0&expires_in=64090

This feature of return of Oauth token as a URI parameter enables the Javascrupt running in the browser to retrieve the token.

These are the HTTP calls made during Client-Side flow:



Client Side Protocol : Figure 7

Using the Access Token

A valid access token enables an application invoke Graph API on a user. If the user changes his password or if the time limit on the access token expires the above mentioned procedure is to be repeated. Even in the case where the user explicitly de-authorizes the application the whole procedure is to be repeated. De-authorization causes a request with the user id sent to the Facebook server which expires all of the user's tokens.

In such a case re-invoking the API call on the user throws an OAuth exception.

Application Login

In order to take various control actions on our applications such as retrieving insights data (no of users using the application, locations of users etc) the application itself assumes he role of the user in the above described OAuth process. Such tasks use application access token. Application access token can be obtained by providing app secret and app id in the client_credentials and grant_type parameters in the oauth request

Example:

```
https://graph.facebook.com/oauth/access_token?client_id=YOUR_APP_ID&client_secret=YOUR_APP_SECRET&grant_type=client_credentials
```

The properties of the acces token obtained here is the same as the end-user generated access token. It is discussed in full detail in the next chapter.

Page Login

A Facebook page is an advertising front end for people and business. They follow a similar OAuth process for authentication and obtaining user data.

The flow of control is a little different and one needs to be an administrator to request any access. Such administrator privileges can be granted as follows by the page creator:

```
https://www.facebook.com/dialog/oauth?client_id=YOUR_APP_ID&redirect_uri=YOUR_URL&scope=manage_pages&response_type=token
```

Chapter 3 - Access Token Deciphered

Access token enables an authenticated application to access the user's information and take actions on their behalf. This chapter describes what an access token is made up of and how to build an access token given all the required parameters.

Examples of access tokens:

App Research

```
1751776058580 34|2.AQDsp8G5t863_jHK.3600.1312524000.0-665186330|CDR-ny89DO_aQx-4oSaVk6qtNvo
```

Argus Research App

```
2058939194388 74|2.AQDxgSULyWHyWBnv.3600.1312524000.0-665186330|OM2Vg-VW8QqaHQBW7I3dqXUx71U
```

An access token is made up of six different parts. Each part is collected at a different step of Application-Facebook interaction before the issue of the access token. The parts are:

1. Application ID:
   This number uniquely identifies an application, provided during the application authentication step.
2. User ID:
   Uniquely identifies a user and binds it to the application id. Date of Issue: the date and the time accurate up to minutes including AM/PM when the token was issued to the application.
3. Date of Expiry:
   The date and time at which the access token will expire. Accuracy is maintained up to minutes.
4. Validity:
   Is the token currently valid. Token can be active(valid) or expired(invalid)
5. Origin:
   The URL where the token generation request was initiated. This field contains the URI of the server in case of server-side access or a null value if generated on the client side.
6. Scopes:

The data fields on the user's profile the application can access and or control.

Directly Obtainable Fields:

- User ID : 665186330
- Application ID : 175177605858034
- Date of Expiry: 1312747200 : 1:00 pm Aug 7 2011

Directly Obtainable Fields:

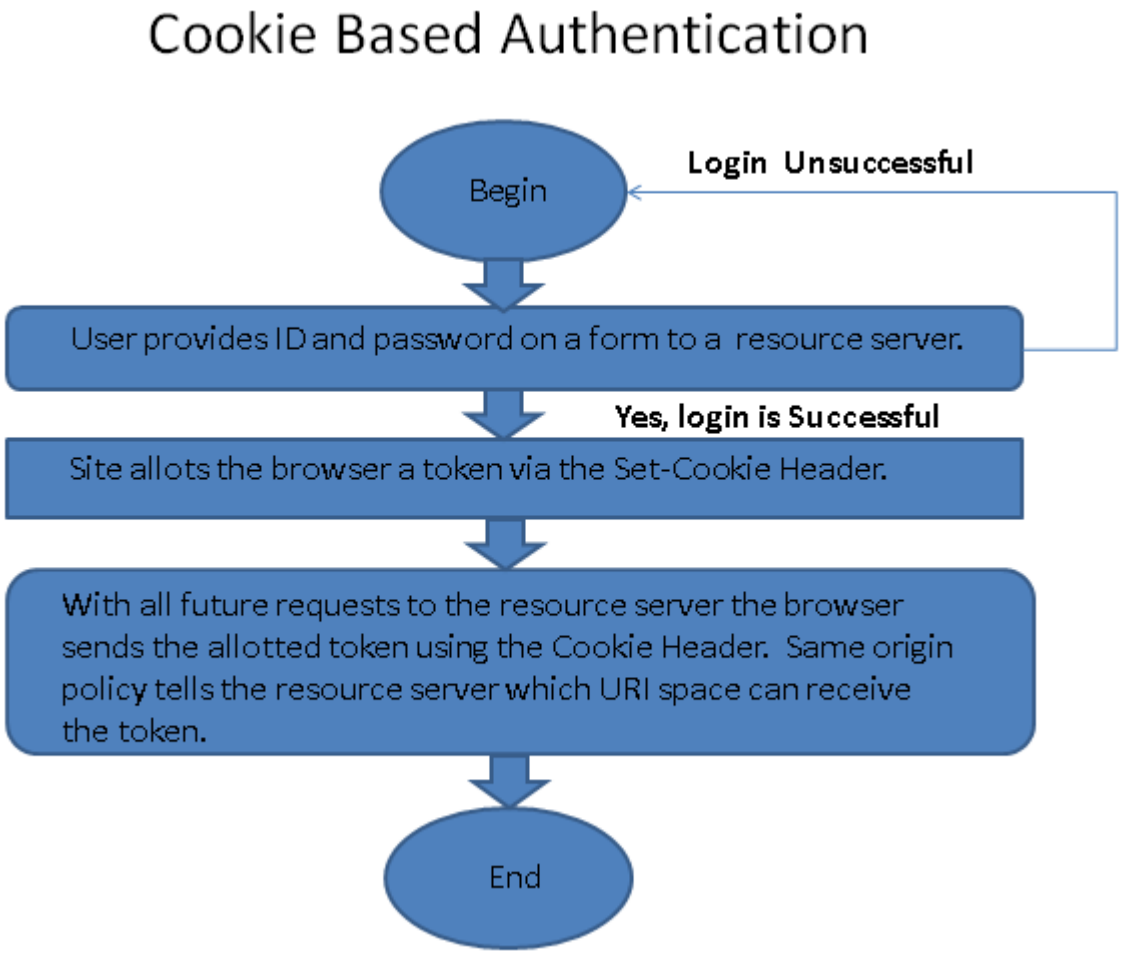
- User ID : 665186330
- Application ID : 205893919438874
- Date of Expiry: 1312747200 : 1:00 pm Aug 7 2011

Security Concerns with OAuth Token

*(*or Similarities between an OAuth Token and a Cookie*)*

The access tokens that the applications use to access protected resources is exchanged like a cookie.

The following figures depict how the process of obtaining and using an access token is similar to storing and using cookies. This makes it vulnerable to same kind of attacks that cookies are subjected
to.

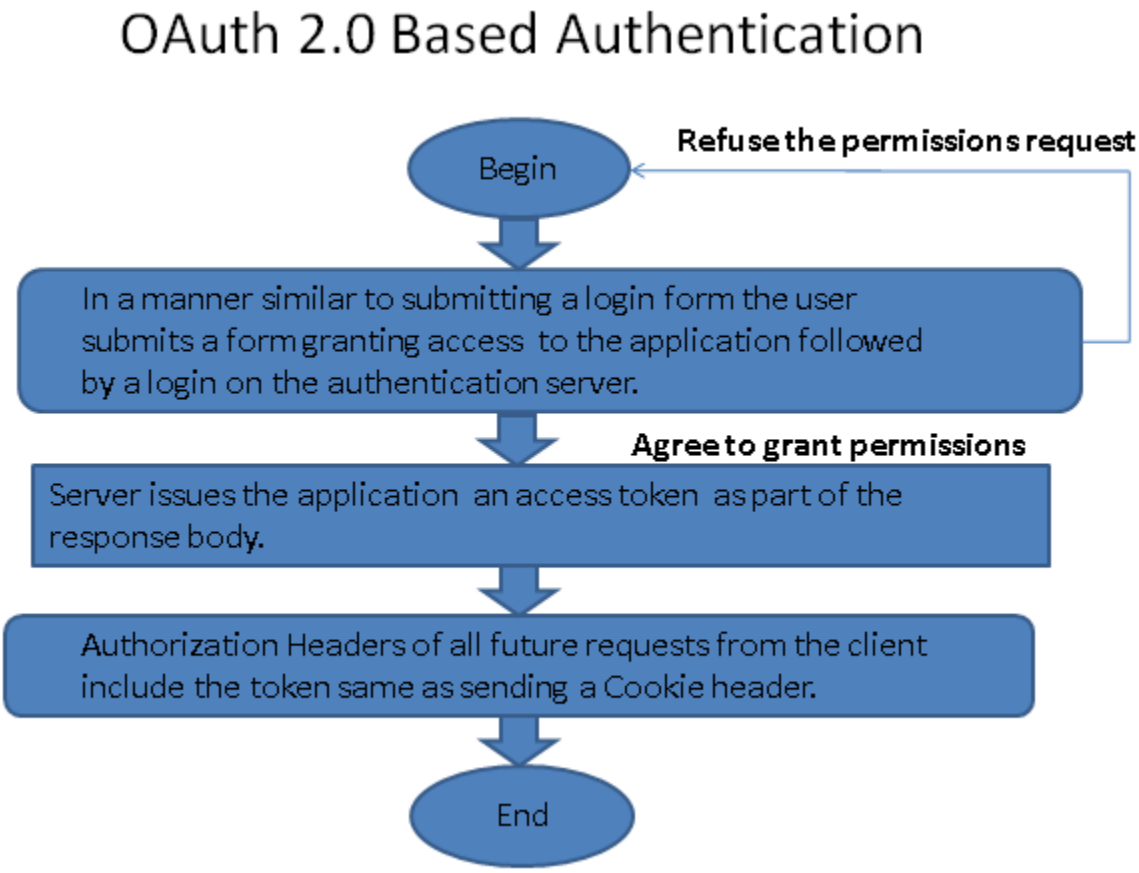


Few of the security issues being:

1. Network Eavesdropping.

2. Publishing a false domain – DNS Poisoning.

3. Cross Site Scripting.



Chapter 4 - Suggestions for Improvement

Following are some the features and improvements we suggest for allaying security concerns:

Monitoring and analyzing applications

1. Transparency:

   Activities of the application should be made more transparent to the end user. The API calls made by the application along with the response of those API calls and resulting data accessed from the user's profile should be shown. This will give the user a better idea of what the application is trying to accomplish. An end-user who authorizes an application does not have a clear idea of what he is actually sharing. If made transparent the user will be better informed of data being shared. This will on the application side add more legitimacy to the functioning of the application as a result of which more users will feel secure using it.

2. Ungrouping Permissions:

   Permission is given to the application is given in groups. i.e. basic info has name, location etc, interests info has musical preferences, personal info has date of birth, address and so on. Although this method is very user-friendly as the user only has to choose a category instead of looking at the field names it is not very secure. The sensitive nature of the user's profile information warrants a finer control over the fields if a group policy is to be included then the user should be asked as to decide as to what field belongs to what category. for example address may be considered private information by some and public by others.

Social Graph Issues

Social graph is a mutli level tree. It has the account owner at the root level.

Owner's friend at level 1 and the friends of friend at level 2+. Applications are also at the first level. This structure enables Facebook to restrict access on a level by level basis. Applications are like friends of the owner and warrant a separate privacy setting.

a) Inflation prevention:

Because of the lack of the flexibility to add users and restrict their visibility to a fixed application, often users end up adding friends who are not really friends but partners in an application. For example in the application Mafia Wars, a user needs to have atleast 15 other friends who are also on Mafia Wars to advance to the next level. And so for the sake of an application friends are added who not only get to be on the same team but also get to access all the fields on the user's profile. An alternate way of implementing this would be to have application specific friends. i.e. friends who can see only the data that you have exposed to the application. Their existence from the user's point of view should end with application and so they are not included on the generic friends but friends on the application only. This will prevent adding friends for an application only but giving them full friend privileges over you profile.

b) Information Leakage through friends:

The way the social graph is implemented by Facebook, even if you dont add an application your data is exposed to the application through your friends who add the application. By default all of your data that is visible to the friend is also visible to the application added by your friend. The user is out in the dark with no visibility into the application that have been added by his friends. With the average length of friends list being 56 even with such a visiblity it would be very hard to keep track. Making a system where the user is notified everytime his data is implicitly accessed can circumvent such data leak and protect the user from a feature he has no control over.

3. The pressing issue and the most debated is the meaning of the privacy settings.

Improving Documentation & UI Design

More documentation pressing issue should be provided to the end-user so he understands what a setting a certain field to a specific level of access means. A recently a survey was conducted at Columbia designed to measure the user's privacy priorities, confidence in existing settings, Facebook usage, history of privacy violations, and exposure to privacy-related media coverage.

The survey was conducted in four stages:

i. Provide a survey to understand the general privacy attitude:
ii. Collect participants sharing intentions for each profile group.
iii. Examining participant's Facebook data to identify potential violations based on the intentions stated.
iv. Present participants with the potential violations as found in the previous step. Ask them to confirm the actual violations, and survey their intent to act on the violation.

The results showed that 93.8% of participants revealed some information that they did not want disclose and 84.6% of participants are hiding information that they wish to share. This shows that user interface in efficient in reflecting people’s intentions. Between these two figures every single participant had at least one sharing violation. This shows that more needs to be done to educate the end user.

Chapter 5 - Summary

We conducted an evaluation of the existing security architecture and behavior of Facebook applications. By building and testing our own server-side application we were able to deduce the steps involved from creation to deployment of an application. We monitored and analyzed various aspects such as OAuth access tokens. We felt that there are shortcomings in the authentication/authorization process and the privacy settings due to Facebook prioritising ease of use over security. We provide improvements to the current mechanism based on our findings and suggest directions. We as external researchers have limited abilities to perform any changes but with the advent of the client-side API from Facebook that may change. We may be able to build external modules to monitor client side code like JavaScript running on the browser.

Incentives from Facebook like “Security Bounty” rewarding developers who point out flaws or suggesting improvements shows that Facebook does take security seriously. There is a lot that can be done.

References

Facebook Developers Documentation, http://developers.facebook.com/docs

OAuth 2.0 Authorization Protocol, http://tools.ietf.org/pdf/draft-ietf-oauth-v2-20.txt

Report on Facebook's version of 2.0, http://news.techworld.com/security/3240746/oauth-20-api-security-tool-used-by-facebook-too-easy-to-crack/

Michelle Madejski, Maritza Johnson, Steven M. Bellovin .The Failure of Online Social Network Privacy Settings(2011)

Facebook flaw allowed websites to steal users' personal data without consent, http://nakedsecurity.sophos.com/2011/02/02/facebook-flaw-websites-steal-personal-data/

Informatics students discover, alert Facebook to threat allowing access to private data, http://www.physorg.com/news/2011-02-informatics-students-facebook-threat-access.html

Inside Facebook Top 25 games, http://www.insidefacebook.com/2011/08/01/the-top-25-facebook-games-for-august-2011/

Facebook Statistics, http://www.facebook.com/press/info.php?statistics

Besmer, A., and Richter Lipford, H. Moving beyond untagging: photo privacy in a tagged world. In CHI '10: Proceedings of the SIGCHI conference on Human factors in computing systems

Nazir, A., Raza, S., and Chuah, C. 2008. Unveiling facebook: a measurement study of social network based applications. In Proceedings of the 8th ACM SIGCOMM Conference on internet measurement

Gross, R., and Acquisti, A. Information revelation and privacy in online social networks. In Proceedings of the 2005 ACM workshop on Privacy in the electronic society

Hart, M., Johnson, R., and Stent, A. More content-less control: Access control in the web 2.0. In WOSP '08: Proceedings of the _rst workshop on Online social networks