# Oauth

n|u - The Open security community

Chennai Meet

Presenter : Vinoth Kumar

Date : 22/07/2017

# About Me

Application security engineer.

Blogger @ http://www.tutorgeeks.net

Email @ vinothpkumar333@gmail.com

Tweet @vinothpkumar

## Agenda for the session

- What is Oauth 2.0
- Why Oauth is required
- Oauth 2.0 Terminologies
- Oauth workflow
  - Sequence flow
  - Workflow request
- Exploiting Oauth for fun and profit
- Reference

## What is Oauth 2.0

- Oauth2.0 is an “authorization” framework for web applications. It permits selective access to a user’s resource without disclosing the password to the website which asks for the resource.
- OAuth works over HTTP and authorizes applications with access tokens rather than credentials
- Oauth is not SSO. SSO is an authentication / authorization flow through which a user can log into multiple services using the same credentials whereas OAuth is an authorization protocol that allows a user to selectively decide which services can do what with a user’s data.

## Why Oauth is required

- In classic authentication model, the user’s account credentials are generally shared with the third party website which results in several problems.
  - Third party website may insecurely handle the credentials.
  - Compromise of one 3rd party service eventually compromises all 3rd party services since same credentials are being used across all services.
  - Changing account password will eventually stop access to all 3rd party services.

## Oauth 2.0 Terminologies

- Roles
  - Resource owner
  - Resource server
  - Client
  - Authorization server
- Workflow parameters
  - Application registration
  - Redirect URI
  - Access token
  - Client ID
  - Client secret

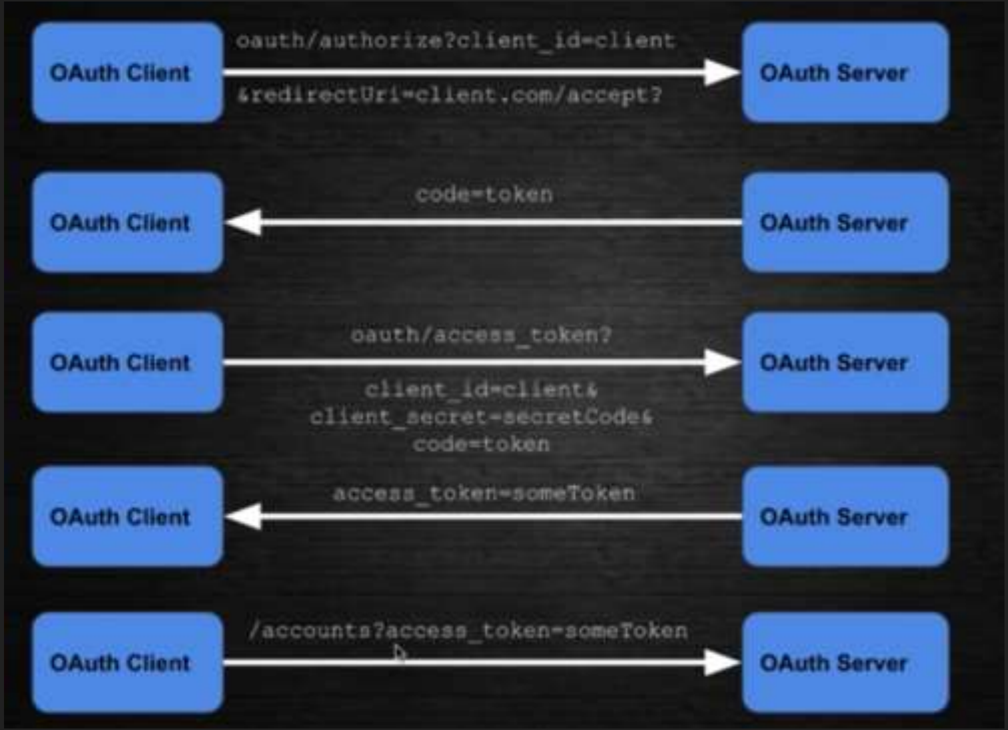
## Oauth workflow

- The user requests the client to start the authorization process through the user-agent by issuing a GET request. This happens when the user clicks on ‘Connect’/’Sign in with’ button on the client’s website.
- The client redirects the user-agent to the authorization server using the following query parameters:
    - **response_type**: *code*
    - **client_id**: The *id* issued to the client.
    - **redirect_uri**(optional): The URI where the authorization server will redirect the response to.
    - **scope**(optional): The scope to be requested.
    - **state**(recommended): An opaque value to maintain state between the request and callback.

## Oauth workflow ( Cont )

- After the user authenticates and grants authorization for requested resources, the authorization server redirects the user-agent to the *redirect_uri* with the following query parameters:
  - **code**: The authorization code.
  - **state**: The value passed in the above request.
- The client further uses the *authorization code* to request for an access token(with appropriate client authentication) using the following parameters in the request body:
  - **grant_type**: *authorization_code*
  - **code**: The authorization code received earlier.
  - **redirect_uri**: The *redirect_uri* passed in the first request.

# Oauth sequence flow



## Workflow request

- https://www.facebook.com/oauth/authorize?response_type=code@client_id=******&redirect_uri=https://example.com/callback/redirect&scope=read
- https://example.com/redirect?code=90jo42hasdoasdh
- https://facebook.com/oauth/token?client_id=*******&client_secret=*******&grant_type=authorization_code&code=90jo42hasdoasdh&redirect_uri=https://example.com/token

Access token = Auth code + Client ID + Client secret + Redirect URI

{ “access_token” : aknasda809asdak }

https://facebook.com/comments?access_token=aknasda809asdak

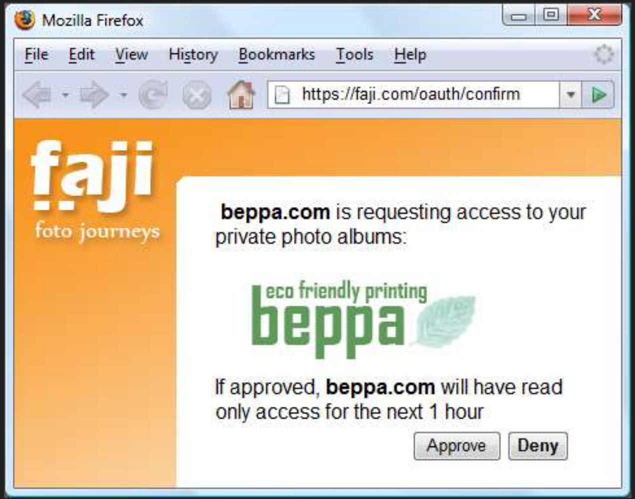
Refresh token: A Refresh Token is a special kind of token that can be used to obtain a renewed access token that allows accessing a protected resource at any time

## Exploiting Oauth for fun and profit

- Clickjacking
- Covert redirect
- Directory traversal tricks
- Domain tricks
- Missing “State” parameter
- Fiddling with the “scope” parameter
- Authorization code not invalidated during access revocation

## Clickjacking

Have secure implementations for preventing Clickjacking in Oauth dialog prompt.


Mozilla Firefox

File Edit View History Bookmarks Tools Help

https://faji.com/oauth/confirm

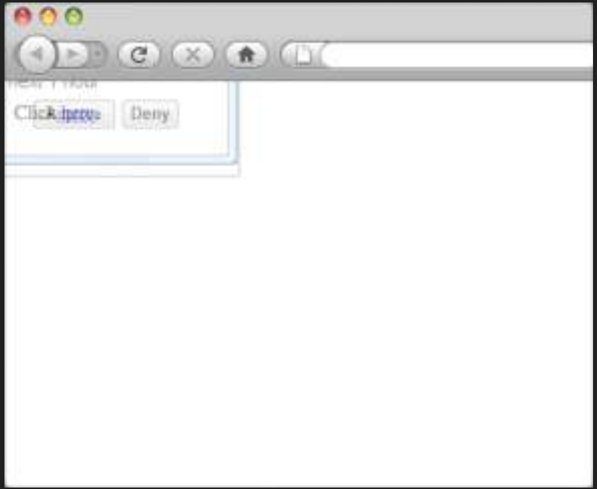
faji foto journeys

**beppa.com** is requesting access to your private photo albums:

eco friendly printing beppa

If approved, **beppa.com** will have read only access for the next 1 hour

Approve Deny



- [https://stephensclafani.com/2009/05/04/clickjacking-oauth/](https://stephensclafani.com/2009/05/04/clickjacking-oauth/)
- Clickjacking in Coinbase Oauth is worth 5000USD : [https://hackerone.com/reports/65825](https://hackerone.com/reports/65825)

## Covert redirect

If the domain is not verified, we could tamper the “redirect_uri” parameter and steal the “code”. Try to find an Open URL Redirection vulnerability in the target.

https://www.facebook.com/oauth/authorize?response_type=code@client_id=******&redirect_uri=https://example.com/hello&scope=read

https://www.facebook.com/oauth/authorize?response_type=code@client_id=******&redirect_uri=https://example.com/hello/anything.php&u=attacker.com&scope=read

Open redirection vulnerability : https://example.com/hello/anything.php&u=attacker.com

## Directory traversal tricks

It assumes that we can save certain files of our choice under the allowed domain. Possible in web applications which allow uploading of files

https://example.com/token/callback/../../../our/path

/our/path/../../http://example.com/token/callback

Try different encoding techniques to fool the authorization server.

## Domain tricks

If the allowed redirect_uri is https://example.com/token/callback, then we can use the following tricks to bypass the restriction

- Naked domain

  https://subdomain.example.com/token/callback

  You are lucky, if you find a subdomain takeover vulnerability :)

- TLD Suffix confusion

  Replace the suffix with .com.mx, .com.br.

  Slack OAuth2 "redirect_uri" Bypass - https://hackerone.com/reports/2575

## Missing “state” parameter

During authentication, the application sends this parameter in the authorization request, and the Authorization Server will return this parameter unchanged in the response.

State parameter is used to prevent CSRF attacks.

State parameter missing in Slack :https://hackerone.com/reports/111218

https://pinterest-commerce.shopifyapps.com/auth/pinterest/callback?code=fe373552c348b50000b4951184e86224ddde63c4

Reference: https://auth0.com/docs/protocols/oauth2/oauth-state

## Fiddling with the scope parameter

Try providing different values for “scope” and observe the response.

Phabricator hack : https://hackerone.com/reports/3930

In the case of Phabricator, providing a different scope in phabricator OAuth Dialog, automatically redirected to the attacker site.

https://secure.phabricator.com/oauthserver/auth/?redirect_uri=http://files.nirgoldshlager.com&response_type=code&client_id=PHID-OASC-oyfqtnanxsukiw5lsnce&scope=ggg)

## Authorization code not invalidated

When some user denies access for the application, all “Access tokens” should be revoked and become invalid. But not only Access tokens should be revoked, authorization “Codes” must be revoked too.

Vimeo Oauth hack : https://hackerone.com/reports/57603

- Connect the app “attacker” to vimeo.com by approving the Oauth flow. You would’ve obtained “Code” which you exchanged with Vimeo to obtain “Access token”
- Now go to Vimeo account settings page and revoke the access.
- You can still re-generate the “Access token” using the “Code” obtained in step 1 and have continued access to Vimeo.
- User will have no control to revoke the access now :(

## Reference

- Mastering modern web penetration testing - Prakhar Prasad
- https://auth0.com/
- Hackerone public disclosures
- Wikipedia

Thank You