Neste documento pode ser consultado um resumo com as diretrizes e principais funções do Oauth2

1. Introdução

Para realizar o acesso de uma aplicação online é requerido uma etapa de autenticação do usuário, para que sua identidade possa ser validada e verificada. A maneira padrão do setor de lidar com autenticação para serviços de terceiros é o protocolo OAuth2. O OAuth 2.0 é um protocolo padrão para autorização. Permite que aplicativos como Web App, Mobile e Desktop obtenham acesso limitado às informações de usuários através do protocolo HTTP.

2. Diferença do OAuth para outros métodos

No modelo tradicional de autenticação cliente-servidor, o cliente solicita um recurso de acesso restrito (recurso protegido) no servidor e autentica com o servidor usando as credenciais do proprietário do recurso. Assim, o proprietário do recurso compartilha suas credenciais com o terceiro. Isso cria vários problemas e limitações:

- Os servidores são obrigados a suportar autenticação de senha, apesar as fragilidades de segurança inerentes às senhas;
- Os aplicativos de terceiros ganham acesso excessivamente amplo ao recursos protegidos do proprietário, deixando os proprietários dos recursos sem qualquer capacidade de restringir a duração ou acesso a um subconjunto limitado de recursos.

O Oauth soluciona esses problemas através de uma arquitetura com camadas de autorização. Quando o usuário solicita as informações protegidas e hospedadas no servidor são emitidos conjuntos de diferentes credenciais para validar o acesso de tal informação. Dessa forma, ao invés de utilizar as credenciais do proprietário, como no caso anterior. O Oauth cria tokens de acesso para o cliente por meio de um servidor de autorizaçao com a aprovação do proprietário do recurso. Desse modo, o cliente usa o token para acessar os recursos protegidos e hospedados pelo servidor de recursos.

3. Fluxo de Protocolo

O fluxo de autenticação é definido através de 4 entidades:

1. **Resource Owner**
   Concede o acesso a um recurso protegido.

2. **Resource Server**
   O servidor que hospeda os recursos protegidos, capaz de aceitar e responder às solicitações de recursos protegidos usando tokens de acesso.

3. **Client**
   Entidade que faz solicitações de recursos protegidos em nome do proprietário do recurso e com sua autorização. Geralmente também chamado de usuário.

4. **Authorization Server**
   O servidor que emite tokens de acesso ao cliente depois de autenticar com sucesso o proprietário do recurso e obter autorização.

Dessa forma, de acordo com tais entidades, o fluxo do protocolo ocorre conforme é ilustrado na Figura 1. **(A)** Inicialmente o cliente solicita autorização do proprietário do recurso, caso o cliente receba a concessão, **(B)** é emitida uma credencial. **(C)** Com essa credencial o cliente solicita ao servidor de autorização um token de acesso. **(D)** O servidor de autorização autentica o cliente e valida a concessão de autorização e, se válido, emite um token de acesso. **(E)** O cliente solicita o recurso protegido do servidor de recursos e autentica apresentando o token de acesso. **(F)** O servidor de recursos valida o token de acesso e, se válido, atende à solicitação.

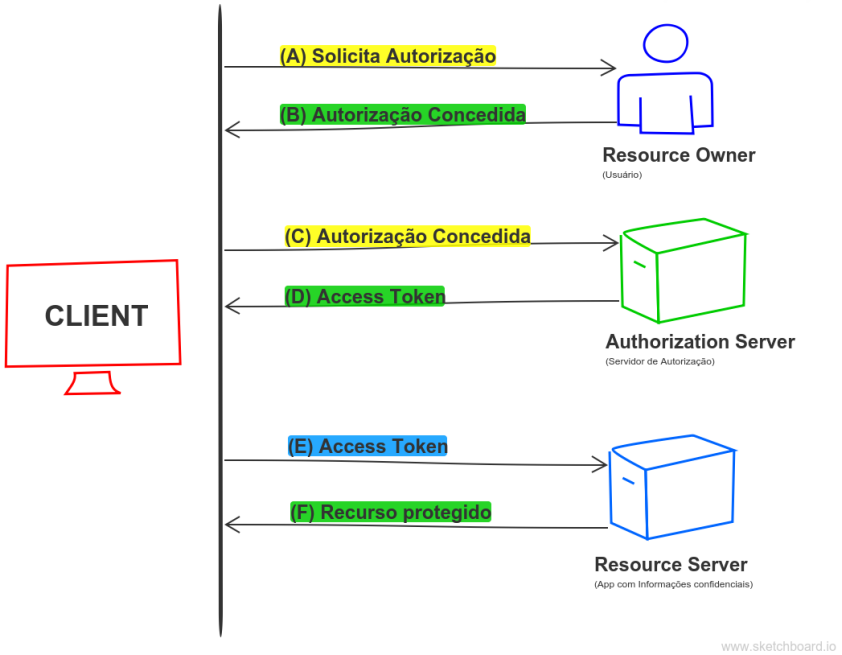

Figura 1. Fluxo de Protocolo

4. Concessão de Autorização

As concessões de autorização podem ser classificadas em 4 tipos:

1. **Código de autorização** (Authorization Code);
2. **Implícito** (Implicit);
3. **Credenciais de senhas do proprietário do recurs**o (Resource Owner Password Credentials);
4. **Credenciais do Cliente** (Client Credentials).

4.1 Código de Autorização

é obtido usando o authorization server como intermediário entre o client e o usuário. O cliente redireciona o usuário para um servidor de autorização, que por sua vez direciona o proprietário do recurso de volta ao cliente com o código de autorização. Esse tipo de client é utilizado em aplicações de terceiros, ou seja, não confiáveis. Como o proprietário do recurso só se autentica com o servidor de autorização, o proprietário do servidor de recursos, as credenciais do nunca são compartilhadas com o cliente.

Vantagens: *a capacidade de autenticar o cliente, bem como a transmissão do token de acesso diretamente para o cliente, sem a passagem por terceiros.*

4.2 Implícitos

é um fluxo de autorização simplificado, otimizado para clients web. Ao emitir um token de acesso, o authorization server não autentica o cliente. Em alguns casos, a identidade do cliente pode ser verificada por meio do URI de redirecionamento usado para entregar o token de acesso ao cliente. É muito utilizado em SPAs e aplicações MVC.

Vantagens: *Melhoram a capacidade de resposta e a eficiência de alguns clientes, pois reduz o número de viagens de ida e volta necessárias para obter um token de acesso.*

4.3 Credenciais de senhas do proprietário do recurso

Geralmente, é utilizado quando o client solicita o usuário e senha diretamente, esse já utilizado em aplicações chamadas de confiáveis, como aplicações da própria empresa. Ou seja, as credenciais do proprietário do recurso são usadas para uma única solicitação e trocadas por um token de acesso.

Aplicações*: As credenciais só devem ser usadas quando há um alto grau de confiança entre o proprietário do recurso e o cliente.*

4.4 Credenciais do cliente

Pode ser usado quando a aplicação client é protegida. Um serviço que consulta uma api. As credenciais do cliente são usadas como uma concessão de autorização, normalmente quando o cliente está agindo em seu próprio nome ou está solicitando acesso a recursos protegidos com base em uma autorização previamente combinada com o servidor de autorização.

Para visualizar o artigo sobre a utilização do Client Credentials clique aqui.

5. Referências

- *IETF, Documentação do OAuth.*
- *TreinaWeb, OAuth 2.*
- *Bruno Brito, OAuth 2.0: Guia para iniciante.*
- *Developers, Autenticação para os serviços OAuth 2.*

Revisão #4
Criado 2024-02-29 13:24:35 UTC por sfsantos
Atualizado: 2024-02-29 13:29:40 UTC por sfsantos