autenticação e autorização: utilizar mecanismos como (OAuth e JWT) como consumidores de mensageria

OAuth

O que é?

OAuth 2.0, que significa "Autorização Aberta", é um padrão projetado para permitir que um site ou aplicativo acesse recursos hospedados por outros aplicativos da web em nome de um usuário.

Como funciona?

No nível mais básico, antes que o OAuth 2.0 possa ser usado, o Cliente deve adquirir suas próprias credenciais, uma *id do cliente* e um *client secret*, do servidor de autorização para se identificar e autenticar ao solicitar um token de acesso.

Usando o OAuth 2.0, as solicitações de acesso são iniciadas pelo Cliente, por exemplo, um aplicativo móvel, site, aplicativo de TV inteligente, aplicativo de desktop, etc. A solicitação, a troca e a resposta do token seguem este fluxo geral:

1. O Cliente solicita autorização (solicitação de autorização) do servidor de Autorização, fornecendo a ID e o segredo do cliente como identificação; ele também fornece os escopos e um URI de endpoint (URI de redirecionamento) para enviar o token de acesso ou o código de autorização.
2. O servidor de Autorização autentica o Cliente e verifica se os escopos solicitados são permitidos.
3. O proprietário do recurso interage com o servidor de autorização para conceder acesso.
4. O servidor de autorização é redirecionado de volta para o cliente com um código de autorização ou token de acesso, dependendo do tipo de concessão, como será explicado na próxima seção. Um token de atualização também pode ser retornado.
5. Com o token de acesso, o cliente solicita acesso ao recurso a partir do servidor de recursos.

Qual produto "programa" que utiliza esse recurso?

Alguns exemplos incluem:

1. **Google** (para autenticar usuários em serviços como Gmail, YouTube, Google Drive)
2. **Facebook** (para login em aplicativos e sites de terceiros)
3. **GitHub** (para autenticar desenvolvedores em ferramentas de integração contínua ou plataformas de colaboração)
4. **Spotify** (para permitir acesso a dados de usuário para aplicativos de terceiros)
5. **Twitter** (para compartilhar permissões de conta em aplicações de terceiros)
6. **Salesforce** (em integrações empresariais)

OAuth é amplamente utilizado para fornecer autenticação de usuário e acesso limitado a dados em APIs de terceiros.

autenticação e autorização: utilizar mecanismos como (JWT) como consumidores de mensageria

O que é?

O JWT (JSON Web Token) é uma forma de autenticação que permite que um servidor **verifique a identidade de um usuário** sem precisar armazenar informações sobre ele.

Ele é um padrão aberto para representar dados de forma compacta e segura entre as partes.

O JWT token tem várias vantagens, mas uma das principais é aumentar a segurança da troca de informações ou acesso que algum usuário possa ter.

Quer saber mais sobre essa forma de autenticação? Neste artigo, separamos as principais informações sobre o assunto para tirar todas as suas dúvidas! Acompanhe o nosso conteúdo e entenda sobre o tema!

Como funciona?

O JWT (JSON Web Token) é um padrão de token baseado em JSON usado para transmitir informações entre duas partes de forma segura. Ele é frequentemente usado para autenticação e autorização em sistemas distribuídos.

Um JWT é composto por três partes:

1. **Header (Cabeçalho)**: Contém o tipo de token (JWT) e o algoritmo de assinatura usado, como HMAC ou RSA.

Exemplo:

```
json
Copiar código
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2. **Payload (Corpo)**: Armazena as informações (claims) que você deseja transmitir, como dados do usuário, permissões e tempo de expiração. Esses dados não são criptografados, mas são codificados em Base64.

Exemplo:

```
json
Copiar código
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

3. **Signature (Assinatura)**: A assinatura garante a integridade e autenticidade do token. Ela é criada combinando o cabeçalho, o corpo e uma chave secreta (ou chave privada) usando o algoritmo especificado no cabeçalho.

Exemplo de assinatura:

```
scss
Copiar código
HMACSHA256(
  base64UrlEncode(header) + "." + base64UrlEncode(payload),
  secret
```

)

Como funciona o fluxo de um JWT?

1. **Autenticação**: O cliente faz login com suas credenciais (usuário e senha, por exemplo).
2. **Geração do JWT**: O servidor valida as credenciais e gera um JWT com informações relevantes no payload, como ID do usuário e permissões. O token é então assinado.
3. **Envio do JWT**: O JWT é enviado ao cliente (geralmente no cabeçalho de resposta HTTP ou como um cookie).
4. **Uso do JWT**: O cliente armazena o JWT e o envia em cada requisição subsequente, geralmente no cabeçalho `Authorization` como um `Bearer token`.

Exemplo de requisição:

```
vbnet
Copiar código
GET /api/secure-endpoint HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

5. **Validação do JWT**: O servidor verifica a assinatura do token e, se for válida, concede o acesso conforme as informações contidas no payload.

Benefícios:

- **Descentralização**: O servidor não precisa armazenar tokens, pois todas as informações necessárias estão no próprio JWT.
- **Escalabilidade**: Ideal para arquiteturas de microserviços.
- **Autossuficiência**: O token contém todas as informações, permitindo uma validação rápida.

Qual produto "programa" que utiliza esse recurso?

Diversos programas e serviços utilizam JWT para autenticação e autorização, principalmente em sistemas web e APIs. Alguns exemplos incluem:

1. **Auth0**: Plataforma de autenticação que utiliza JWT para login e autorização em aplicações web e mobile.

2. **Firebase**: Usa JWT para autenticação segura em seus serviços, como Firebase Authentication e Firebase Cloud Messaging.

3. **Amazon Cognito**: Serviço da AWS para gerenciamento de identidades e autenticação, que emite tokens JWT para controlar o acesso aos recursos da AWS.

4. **APIs RESTful**: Muitos sistemas que expõem APIs REST, como serviços de back-end em Node.js, Django ou Spring Boot, utilizam JWT para garantir a segurança das interações entre o cliente e o servidor.

5. **GitLab**: Usa JWT para autenticar usuários em suas APIs e serviços de CI/CD.

6. **Kubernetes**: Utiliza JWT para autenticação em clusters, especialmente em integrações com provedores de identidade.

7. **Spotify**: Utiliza JWT em sua API para permitir que aplicações de terceiros acessem dados de usuários com permissões apropriadas.

Esses tokens são amplamente utilizados para proteger APIs e autenticar usuários em diversos tipos de plataformas e serviços.