Autenticazione con OAuth

L'implementazione dell'autenticazione OAuth è essenziale per ottenere un Access Token e poter utilizzare le API. Segui i passaggi di questa guida per comprendere il processo di OAuth e integrarlo correttamente nella tua applicazione.

1. **Comprendi il concetto di OAuth**: OAuth è un protocollo standard che consente di delegare l'autenticazione di un utente a un servizio terzo in modo sicuro. Il processo coinvolge tre attori principali: il client (la tua applicazione), il server di autorizzazione (Riseact Account) e il server di risorse (Riseact Core, che fornisce accesso alle APIs). OAuth garantisce che il client ottenga un Access Token valido per accedere alle risorse protette dal server a nome dell'organizzazione che installa l'applicazione.
2. **Registra la tua applicazione**: Prima di implementare l'autenticazione OAuth, è necessario registrare la tua applicazione su Riseact Parnters per ottenere le credenziali necessarie. Queste credenziali includono un Client ID e un Client Secret, che verranno utilizzati per identificare e autenticare la tua applicazione durante il processo di autorizzazione.
3. **Configura l'autenticazione nel tuo backend**: Nel tuo backend, dovrai implementare la logica per gestire il flusso di autorizzazione OAuth. Ciò comporta la creazione di un endpoint per l'autorizzazione che reindirizzerà l'utente al server di autorizzazione per l'autenticazione. Durante questa fase, dovrai includere il tuo Client ID e generare le chiavi PKCE che verranno utilizzate per scambiare il codice di autorizzazione nella callback e che quindi dovrai temporaneamente salvare. Quando un organizzazione installerà la tua applicazione su Riseact, dal pannello di amministrazione verrà visualizzato un iframe che punta all'url dell'app che hai indicato in fase di registrazione. Insieme all'url che hai fornito, verrà passato un parametro `__organization` che potrai utilizzare per identificare l'organizzazione che sta utilizzando la tua applicazione e saltare il roundtrip di selezione dell'organizzazione su Riseact Admin. Per farlo, dovrai reindirizzare l'utente al server di autorizzazione con un parametro `__organization` che contiene lo slug dell'organizzazione che hai ricevuto dal parametro `__organization` dell'url di reindirizzamento.

Ecco un esempio in node.js:

```
app.get('/oauth/authorize', (req, res) => {
  const { codeChallenge, codeVerifier } = generatePkceKeys();

  // Salva le chiavi PKCE come preferisci. In questo caso utilizziamo un database
  db.savePkceKey(codeChallenge, codeVerifier);

  const params = {
    client_id: "CLIENT_ID",
```

```
    redirect_uri: 'https://your-app.com/oauth/callback',
    response_type: 'code',
    code_challenge_method: 'S256',
    code_challenge: 'YOUR_CODE_CHALLENGE',
    __organization: req.query.__organization,
  };

  res.redirect(`https://accounts.riseact.org/oauth/authorize/?${qs.stringify(params)}`);
});
```

Esempio della chiamata con curl:

```
curl -X GET \
  "https://accounts.riseact.org/oauth/authorize/\
?client_id=CLIENT_ID\
&redirect_uri=https://your-app.com/oauth/callback\
&response_type=code\
&code_challenge_method=S256\
&code_challenge=YOUR_CODE_CHALLENGE\
&__organization=YOUR_ORGANIZATION"
```

4. **Gestisci il reindirizzamento di callback**: Dopo che l'utente si è autenticato con successo presso il server di autorizzazione, verrà reindirizzato alla tua applicazione tramite un URL di callback specificato nel precedente passaggio. Nel caso l'URL non corrispondesse a uno di quelli autorizzati in fase di registrazione la richiesta fallirà. Il tuo backend dovrà gestire questo reindirizzamento e recuperare il codice di autorizzazione restituito dal Riseact Accounts. Verifica l'autenticità della richiesta controllando che il codice di autorizzazione corrisponda a quello generato in precedenza. Utilizzando il codice di autorizzazione ricevuto, effettua una richiesta al server di autorizzazione per ottenere un Access Token. Questo Access Token sarà utilizzato per autenticare le successive richieste alle API protette.

Ecco un esempio in node.js:

```
app.get('/oauth/callback', async (req, res) => {
  const { code, state } = req.query;

  if (state !== 'YOUR_CODE_CHALLENGE') {
    return res.status(400).send('Invalid state');
  }
```

```
  // Recupera le chiavi PKCE dal database
  const { codeChallenge, codeVerifier } = await db.getPkceKey(state);

  const formData = {
    client_id: CLIENT_ID,
    client_secret: CLIENT_SECRET,
    grant_type: 'authorization_code',
    code,
    redirect_uri: 'https://your-app.com/oauth/callback',
    code_verifier: codeVerifier,
  };

  const { data } = await axios.post('https://accounts.riseact.org/oauth/token/', qs.stringify(
formData), {
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
    },
  });

  // Salva le credenziali ottenute come preferisci. In questo caso utilizziamo un database
  db.saveCredentials(data.access_token, data.refresh_token, data.expires_in);
});
```

Esempio della chiamata con curl:

```
curl -X POST \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "client_id=YOUR_CLIENT_ID" \
  -d "client_secret=YOUR_CLIENT_SECRET" \
  -d "grant_type=authorization_code" \
  -d "code=YOUR_CODE" \
  -d "redirect_uri=https://your-app.com/oauth/callback" \
  -d "code_verifier=YOUR_CODE_VERIFIER" \
  https://accounts.riseact.org/oauth/token/
```

5. **Utilizza l'Access Token per accedere alle risorse protette dell'organizzazione**: Ogni volta che desideri accedere alle risorse protette dalle API, dovrai includere l'Access Token nella tua richiesta nell'header `Authorization`. Le API utilizzeranno l'Access Token

per verificare l'autenticità della richiesta e fornire le risorse richieste solo se l'Access Token è valido.

Ecco un esempio in node.js:

```
const { data } = await axios.get('https://core.riseact.org/admin/graphql/', {
  headers: {
    Authorization: `Bearer ${access_token}`,
  },
});
```

Esempio della chiamata con curl:

```
curl -X GET \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  https://core.riseact.org/admin/graphql/
```

6. **Gestisci il rinnovo dell'Access Token**: Gli Access Token hanno una durata limitata. Per garantire un'esperienza utente senza interruzioni, dovrai implementare la logica per rinnovare automaticamente l'Access Token prima che scada. Ciò può essere fatto utilizzando il processo di aggiornamento dell'Access Token fornito dal server di autorizzazione.

Ecco un esempio in node.js:

```
app.get('/oauth/refresh', async (req, res) => {
  const { refresh_token } = req.query;

  const formData = {
    client_id: CLIENT_ID,
    client_secret: CLIENT_SECRET,
    grant_type: 'refresh_token',
    refresh_token,
  };

  const { data } = await axios.post('https://accounts.riseact.org/oauth/token/', qs.stringify(
formData), {
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
    },
```

```
  });

  // Salva le credenziali ottenute come preferisci. In questo caso utilizziamo un database
  db.saveCredentials(data.access_token, data.refresh_token, data.expires_in);
});
```

Esempio della chiamata con curl:

```
curl -X POST \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "client_id=YOUR_CLIENT_ID" \
  -d "client_secret=YOUR_CLIENT_SECRET" \
  -d "grant_type=refresh_token" \
  -d "refresh_token=YOUR_REFRESH_TOKEN" \
  https://accounts.riseact.org/oauth/token/
```

Implementando correttamente l'autenticazione OAuth nella tua applicazione, sarai in grado di ottenere un Access Token valido e accedere alle risorse protette tramite le API. Assicurati di seguire le specifiche e le documentazioni fornite da Riseact per un'implementazione corretta e sicura. Buona implementazione!

Revision #1
Created 2026-04-23 08:47:09 UTC by Daniele Cirio
Updated 2026-04-23 08:47:09 UTC by Daniele Cirio