OAuth Hacking Marathon

Exploiting Common Security Pitfalls and Mitigating Them

Kaif Ahsan

Kumar Soorya

AISA

AISA

What looks like a **blessing**

can quickly become a **curse.**


228 FLINDERS STREET
TRUNKS

2010

OAUTH 1.0 RFC GETS PUBLISHED

Internet Engineering Task Force (IETF) E. Hammer-Lahav, Ed.
Request for Comments: 5849 April 2010
Category: Informational
ISSN: 2070-1721

The OAuth 1.0 Protocol

Abstract

OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair), using user-agent redirections.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

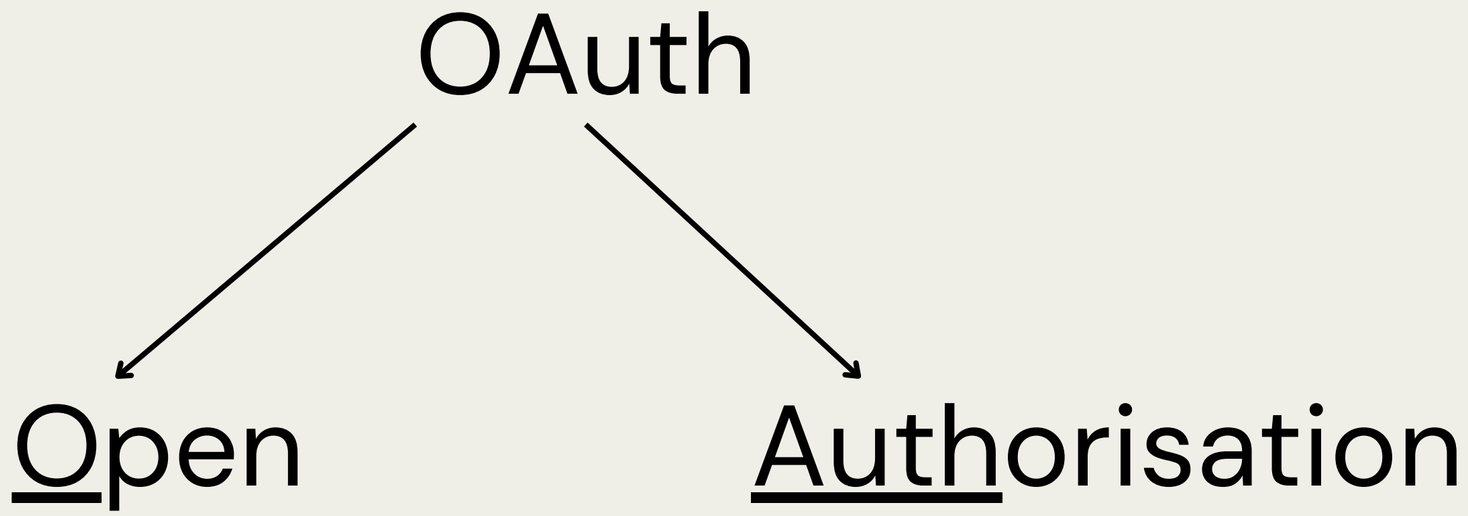
Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5849.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

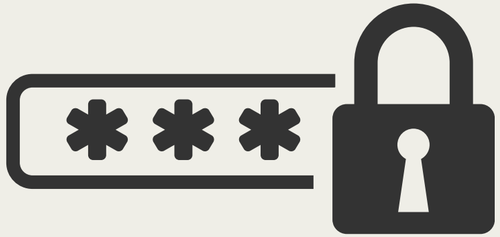
This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

AISA


OAuth
Open
Authorisation

THE CORE PROBLEM

How can **data** be **shared** between apps **without sharing** the **credentials**.




Photobucket
10 August 2010 ·
Now you can login to Photobucket with your Facebook credentials to seamlessly share your photos on Facebook, comment on photos and more! http://blog.photobucket.com/.../08/new-login-options.html
178
45
2
Like
Comment

Photobucket
10 August 2010 ·

Now you can login to Photobucket with your Facebook credentials to seamlessly share your photos on Facebook, comment on photos and more! http://blog.photobucket.com/.../08/new-login-options.html

178 45 2

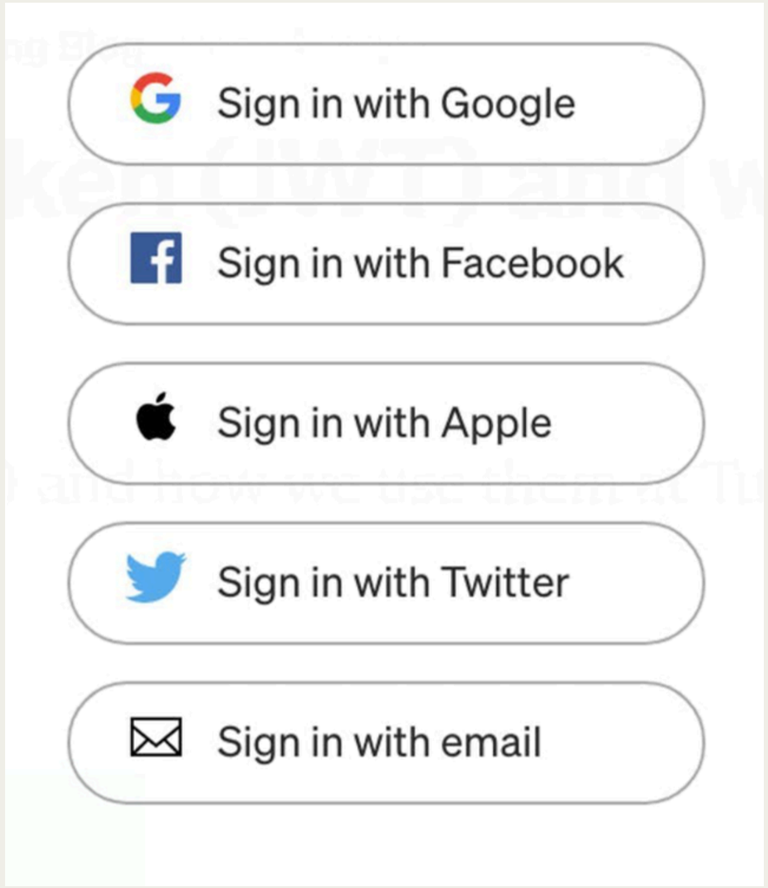
Like Comment

Most relevant

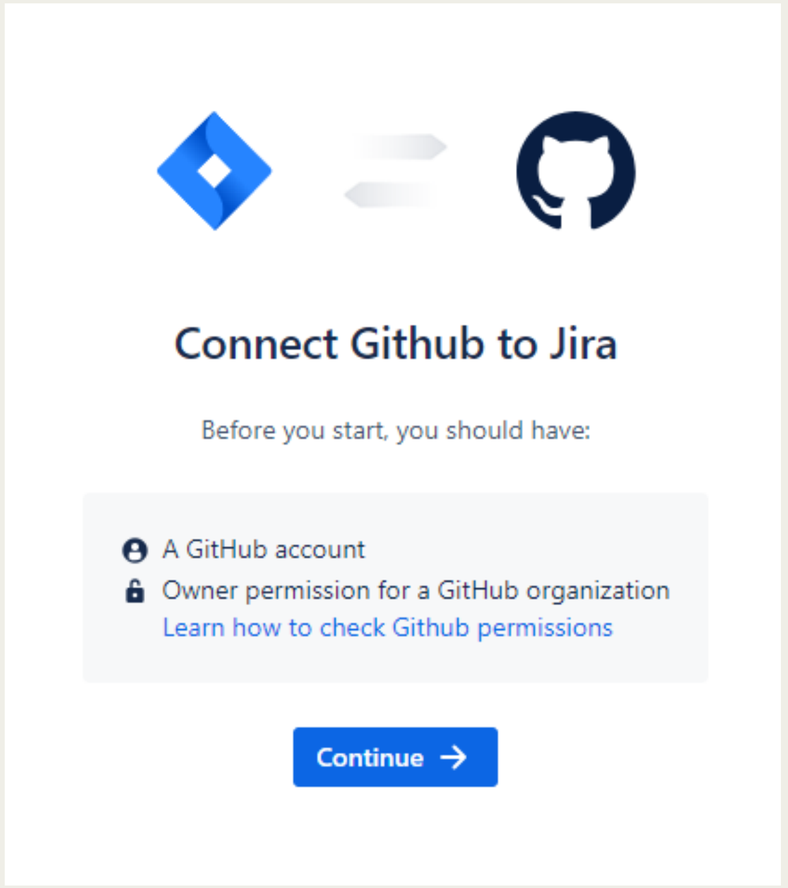
D[redacted]
why?

14y

OAUTH IS EVERYWHERE



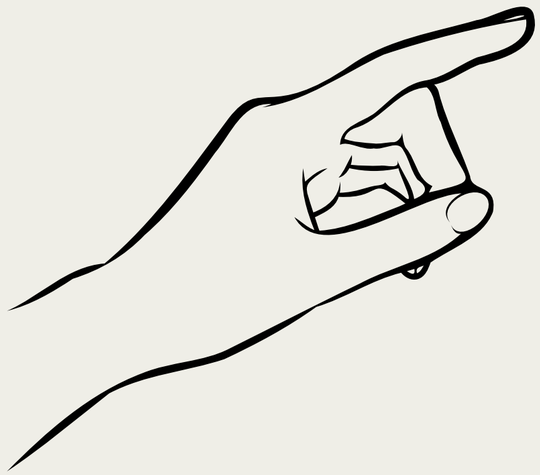
Social logins



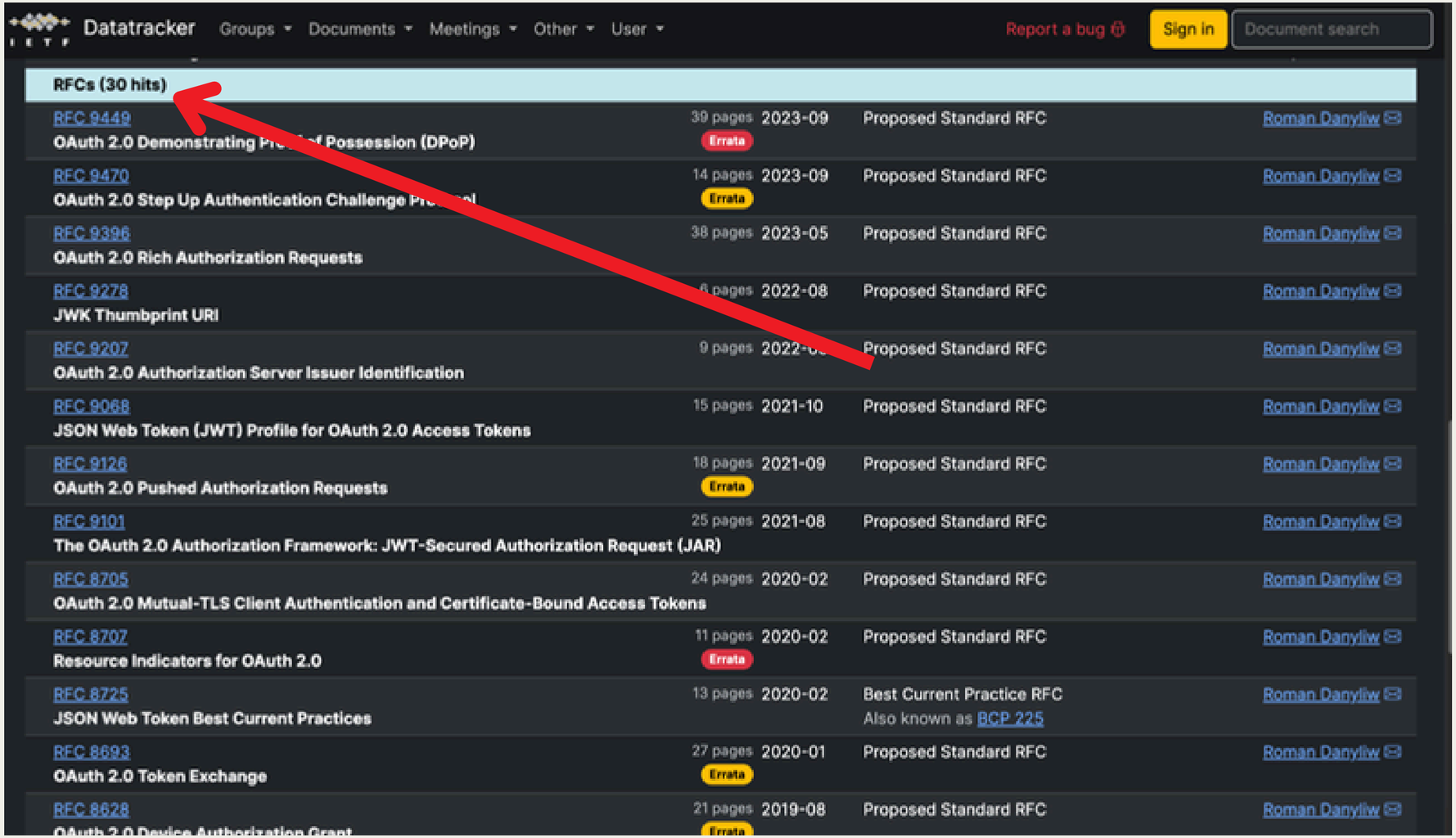
Collecting apps together

OAUTH IS EVERYWHERE

What the ‘Midas Touch’ in OAuth?



OVER 30 RFCS RELATED TO OAUTH

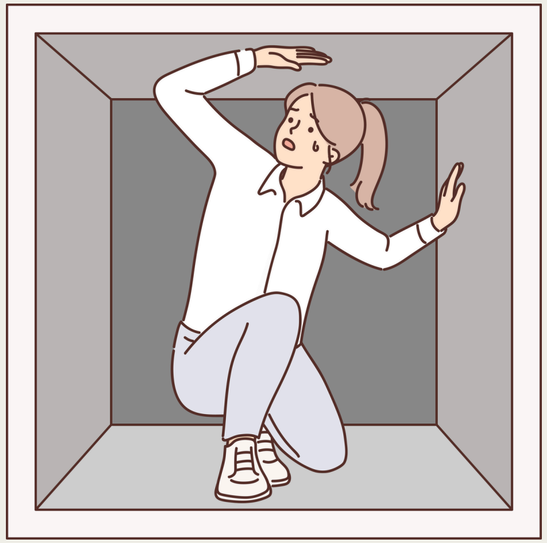


✨ Benefit ✨

OAuth Allows Customisation

Gives developers the **option** to implement various bells and whistles

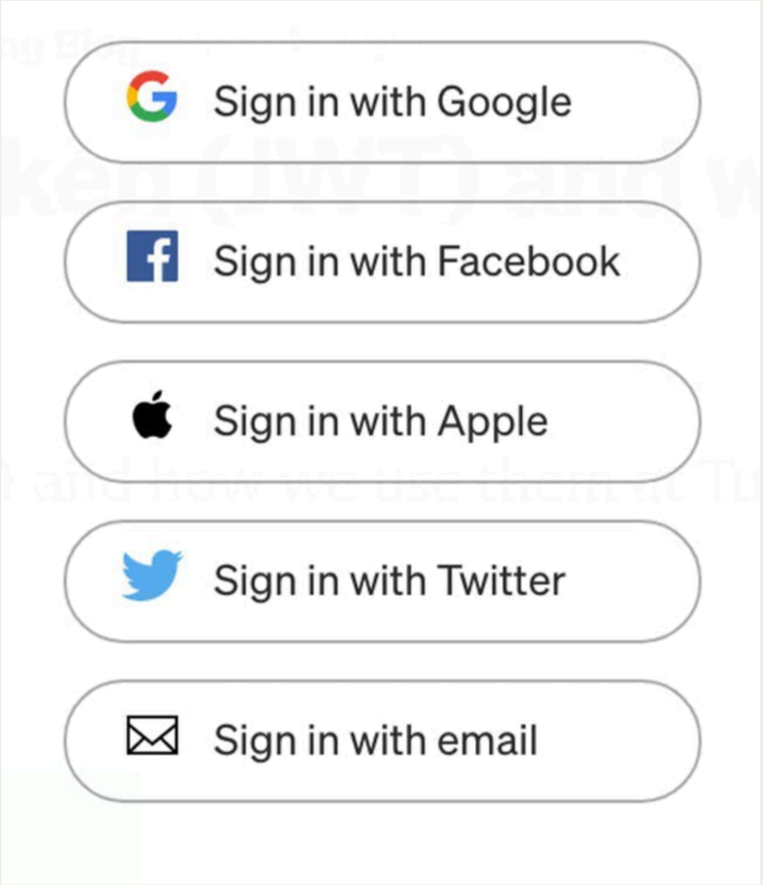
PITFALL



The chance of a **misconfiguration** is very **high**.

OAUTH FUNCTIONALITY

We want the users to log in to our application using their **social login**.



Authorisation Code Flow

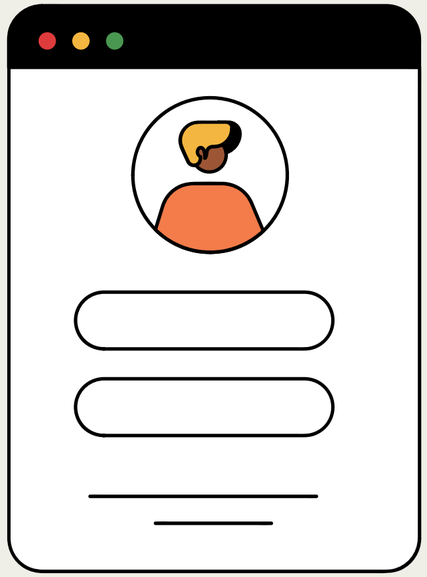
OAuth Flows allow a secure exchange of resources / data across applications.

AUTHORISATION CODE FLOW

User
(via Browser)

Photo Share App
(Client app)

Social Media Site
(Authorisation Server)



Username & email
(Resource Server)

AUTHORISATION CODE FLOW

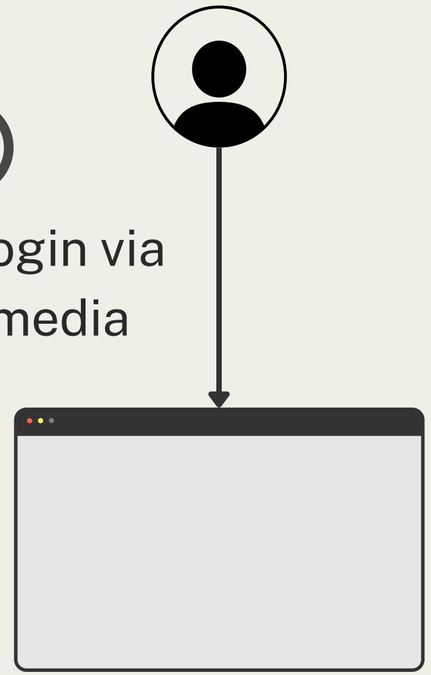
client = The app requesting data. *(Hotel guest)*

Auth server = Verifies the permission. *(Reception desk)*

Resource server = Holds the data. *(Hotel room)*

AUTHORISATION CODE FLOW

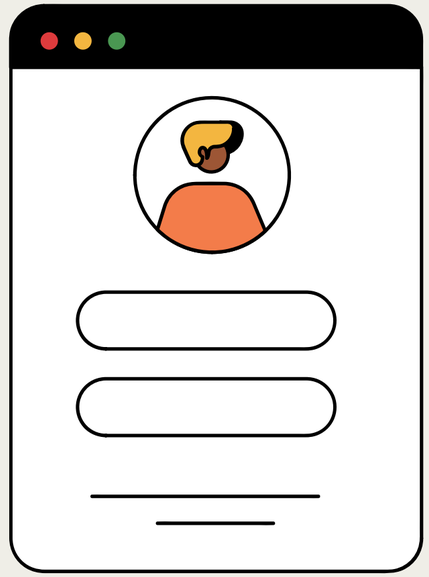
User
(via Browser)



① Please login via social media

Photo Share App
(Client app)

Social Media Site
(Authorisation Server)



Username & email
(Resource Server)

AUTHORISATION CODE FLOW

HOW THE REQUEST LOOKS LIKE

https://dummysocialmedia.com/oauth/authorize?**client_id**=4872984729472947&
redirect_uri=https://photoshare.com/oauth/callback&**scope**=email,public_profile,photos&
response_type=code&**access_type**=offline&**prompt**=consent&**flowName**=social_login

HOW THE REQUEST LOOKS LIKE

https://dummysocialmedia.com/oauth/authorize?client_id=4872984729472947&
redirect_uri=https://photoshare.com/oauth/callback&scope=email,public_profile,photos&
response_type=code&access_type=offline&prompt=consent&flowName=social_login

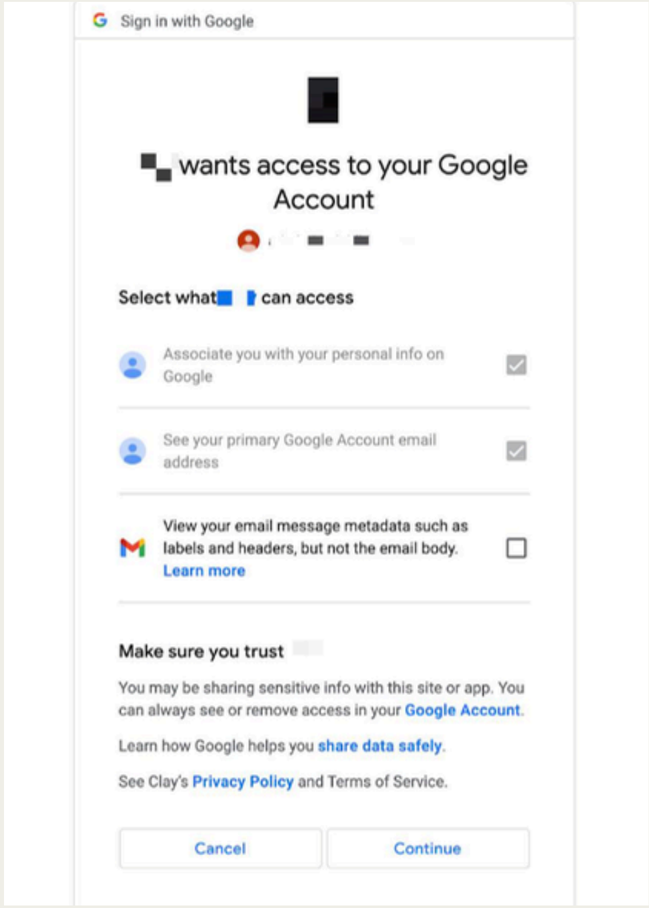
client_id = Unique identifier for our application.

redirect_uri = Where the user will be sent back after authorising

response_type = What type of flow (Authorisation code flow)

AUTHORISATION CODE FLOW

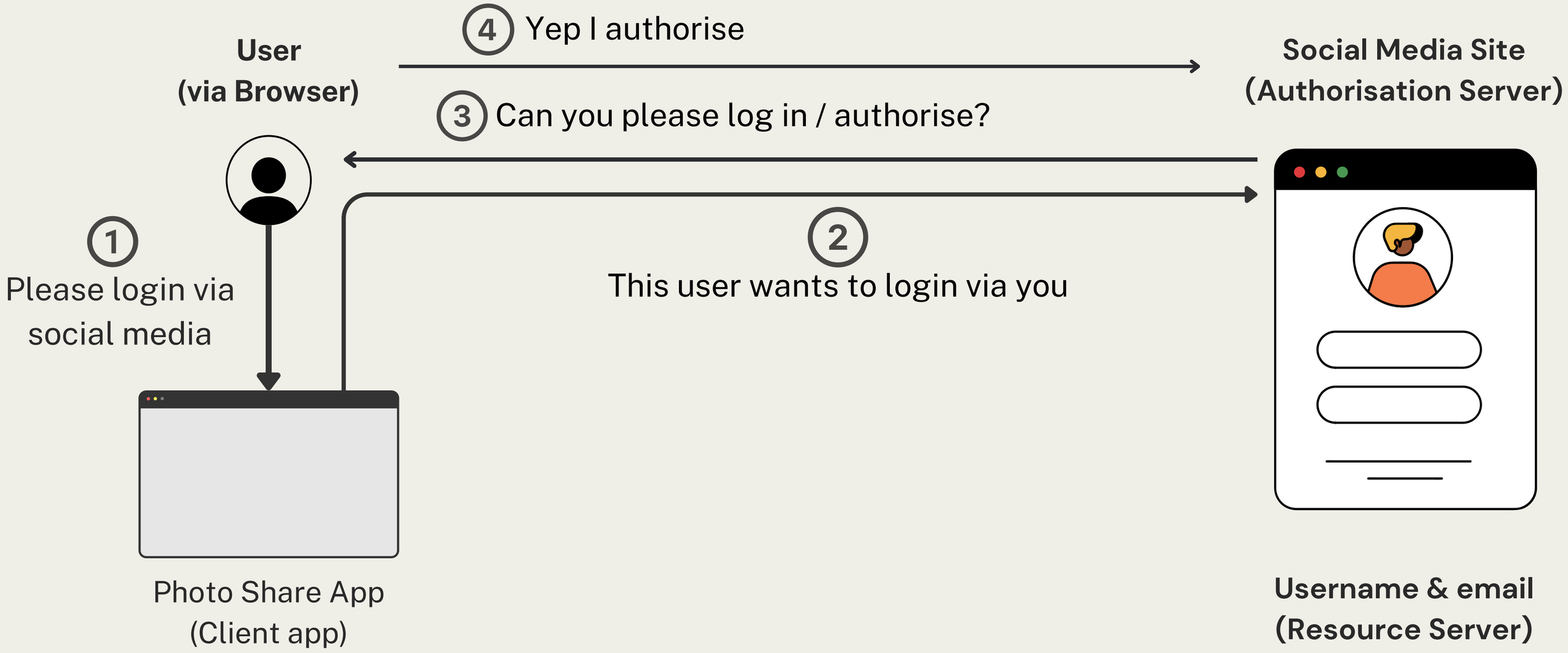
WHAT THE USER SEES



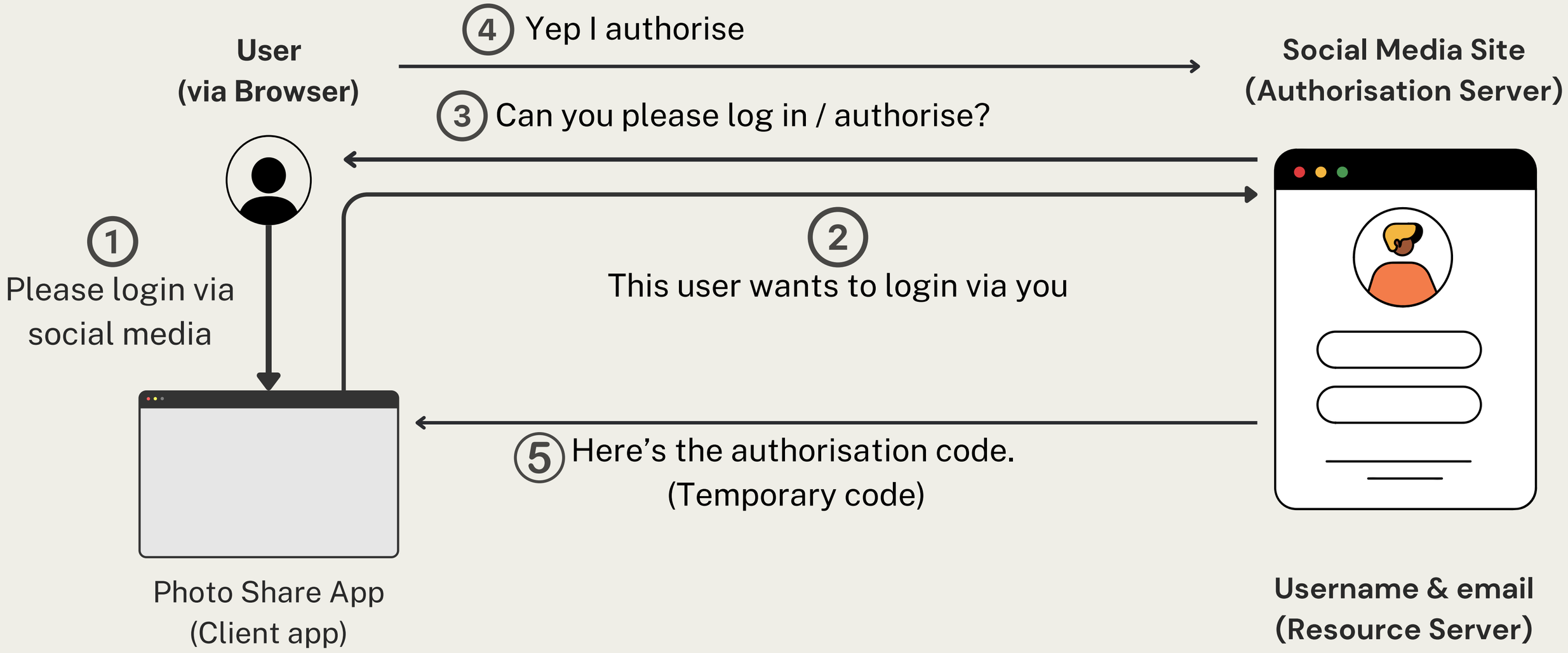
Example Authorisation Request

(Source: fusionoauth)

AUTHORISATION CODE FLOW



AUTHORISATION CODE FLOW

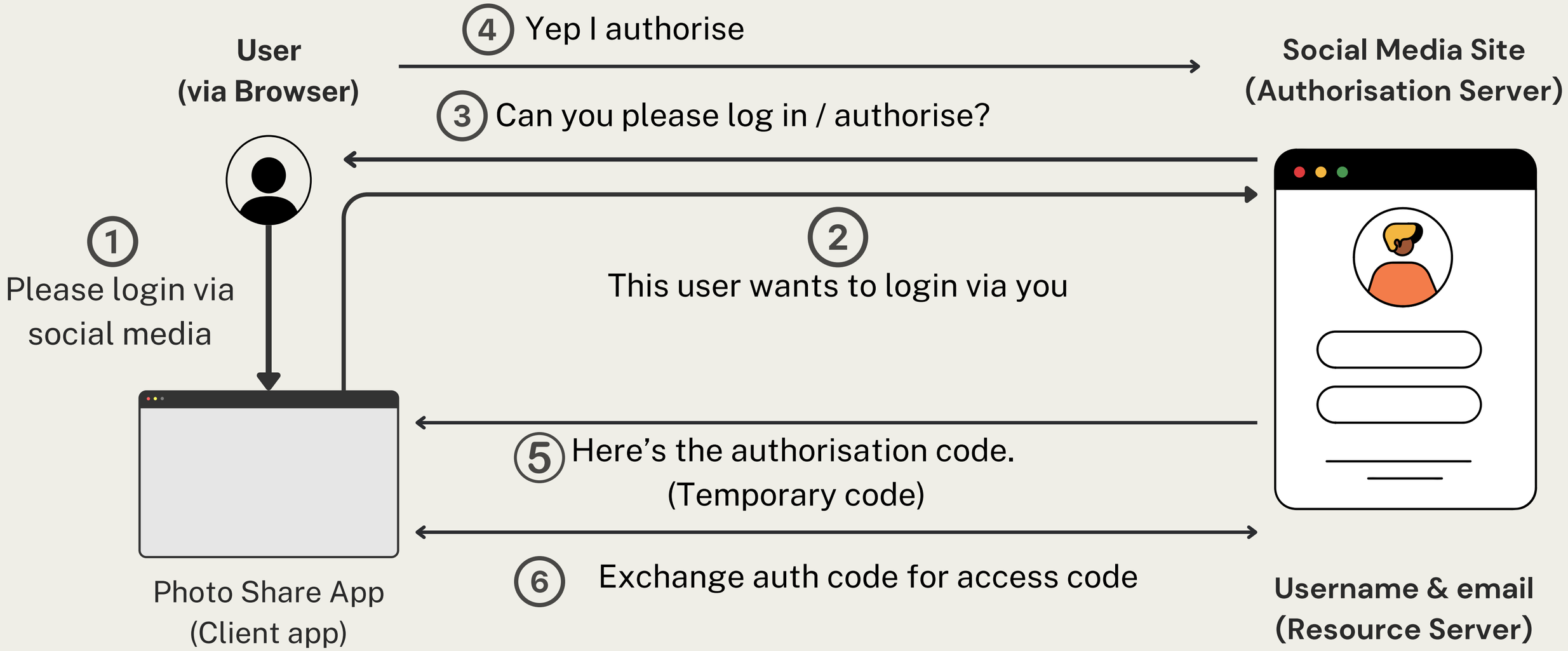


HOW THE REQUEST LOOKS LIKE

Authorisation code

A temporary token that is to be exchanged for the access code

AUTHORISATION CODE FLOW



Example request for access code

```
POST /oauth/token HTTP/1.1
Host: authorization-server.com

code=Yzk5ZDczMzRlNDEwY
&grant_type=code
&redirect_uri=https://example-app.com/cb
&client_id=mRkZGFjM
&client_secret=ZGVmMjMz
&code_verifier=Th7UHJdLswIYQxwSg29DbK1a_d9o41uNMTRmuH0PM8zyoMAQ
```

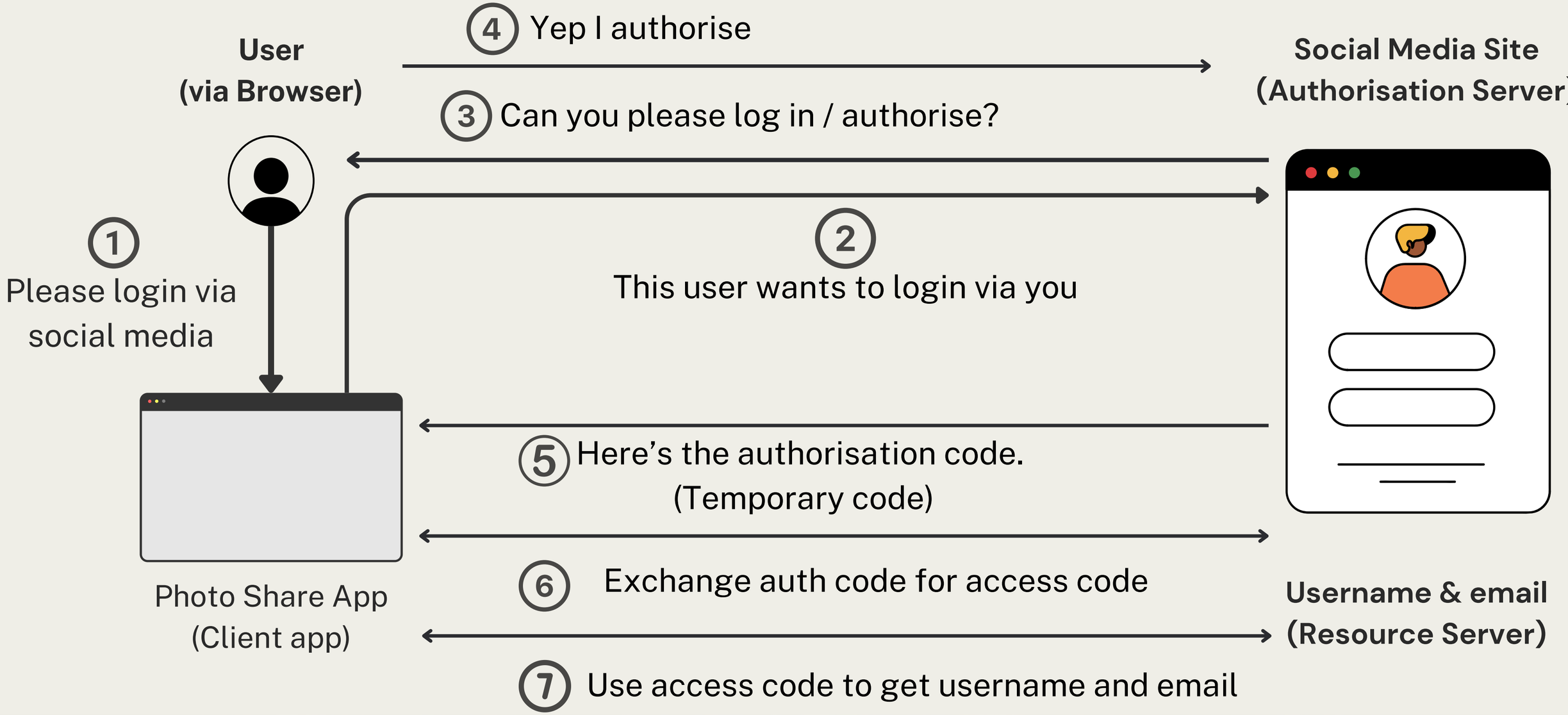
Example Auth code

Example response

```
{
  "access_token": "AYjcyMzY3ZDhiNmJkNTY",
  "refresh_token": "RjY2NjM5NzA2OWJjuE7c",
  "token_type": "Bearer",
  "expires": 3600
}
```

Example Access code

AUTHORISATION CODE FLOW



A BRIEF SUMMARY OF OUR IMPLEMENTATION

1. User **requests** for social login. 📲

2. Our app **initiates** a request and gets an **auth code**. 🎟️

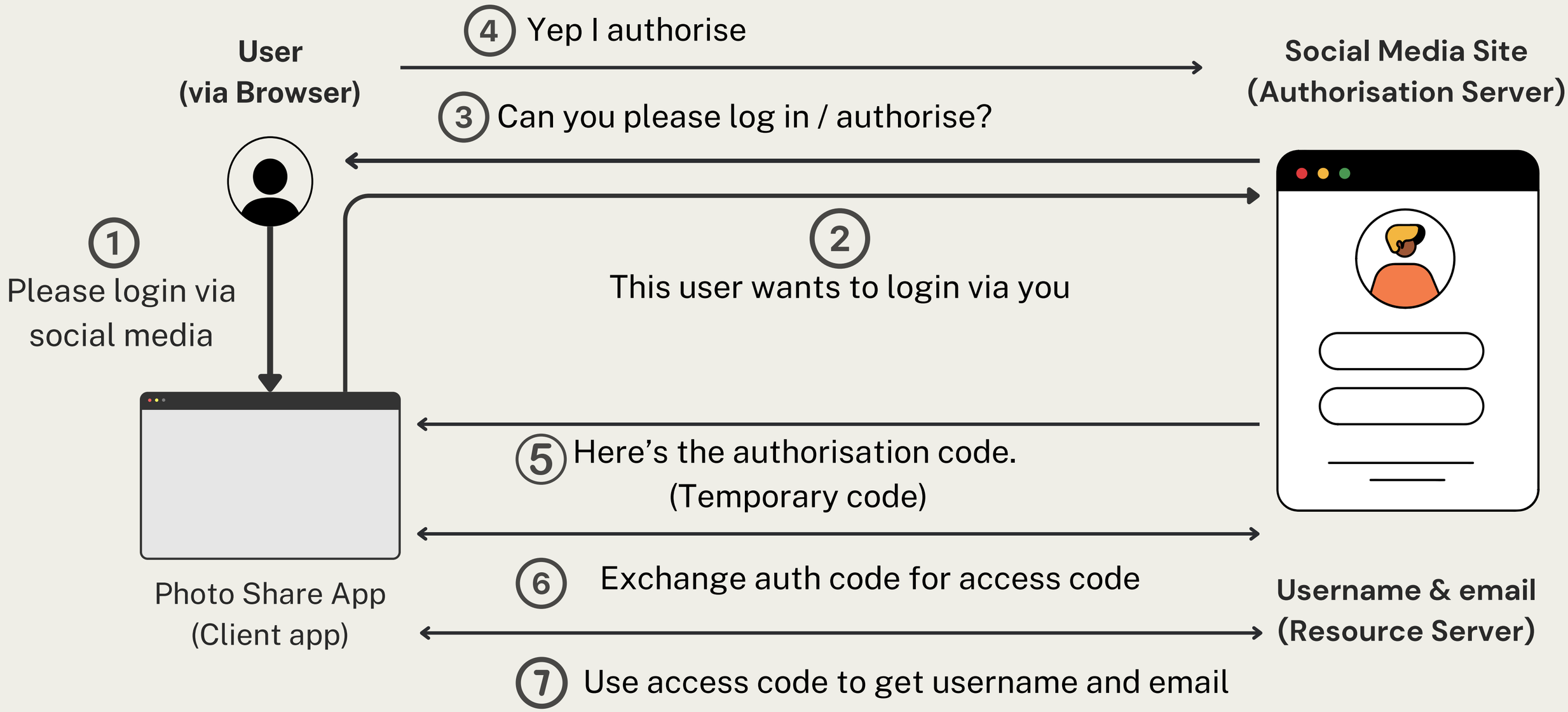
3. Our app **exchanges** the **auth code** for an associated **access token**. 🔑

4. Our app receives **data related to the access token**. 💰

FOOD FOR THOUGHT

Q. How does our app know who requested the auth code?

AUTHORISATION CODE FLOW



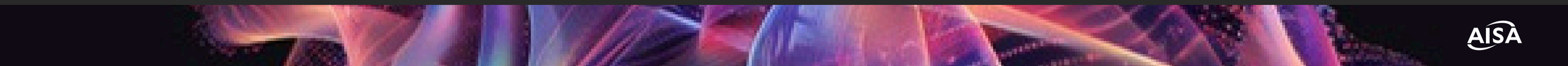
FOOD FOR THOUGHT

🤔

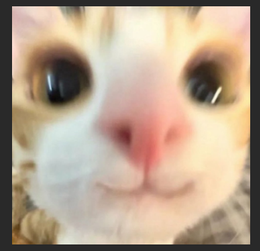
Q. How does our app know who requested the auth code?

A. It currently doesn’t. (It’s stateless)

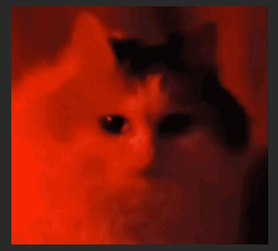
Let the hacking begin



ATTACKER SCENARIO



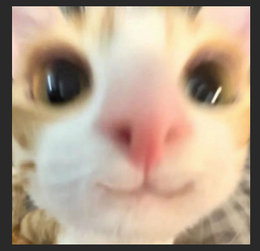
Party Cat



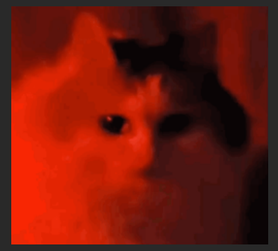
Angy Cat

ATTACKER SCENARIO

ATTACKER SCENARIO

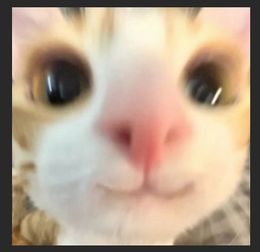


Party Cat
(Victim)

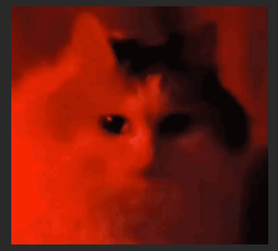


Angy Cat
(Attacker)

ATTACKER SCENARIO



Party Cat
(Victim)



Angy Cat
(Attacker)

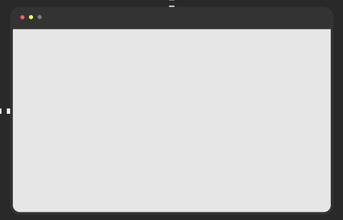
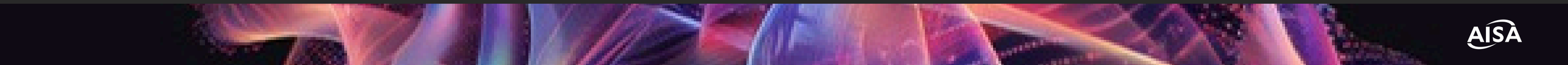
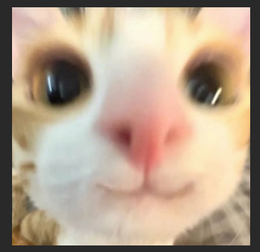


Photo Share App
(vulnerable app)

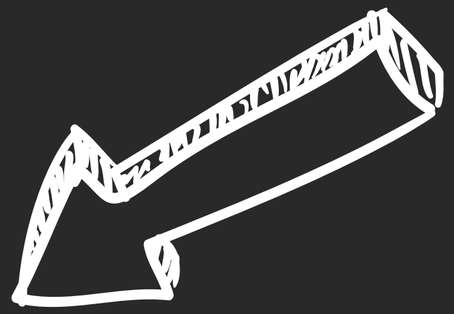


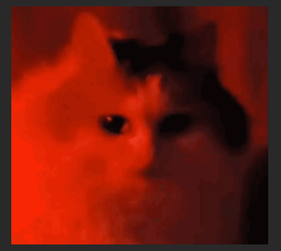
ATTACKER SCENARIO



Party Cat
(Victim)

Steal Photos





Angy Cat
(Attacker)

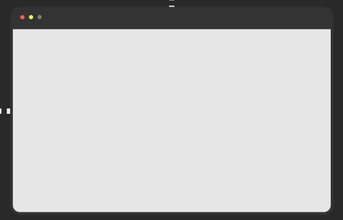
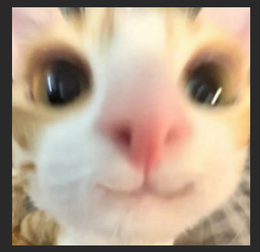


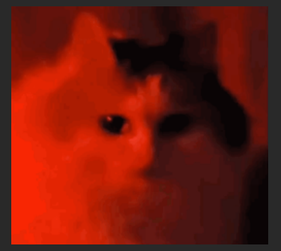
Photo Share App
(vulnerable app)

ATTACKER SCENARIO



Party Cat
(Victim)

Phishing Link
(Triggers OAuth
Request)



Angy Cat
(Attacker)

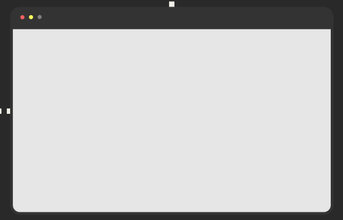
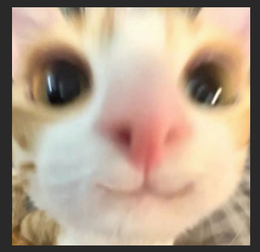


Photo Share App
(vulnerable app)

ATTACKER SCENARIO



Party Cat
(Victim)

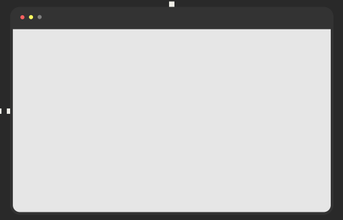
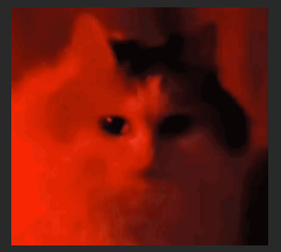


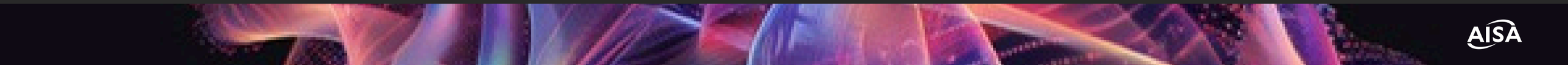
Photo Share App
(vulnerable app)



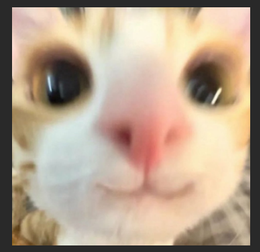
Angy Cat
(Attacker)

GitHub

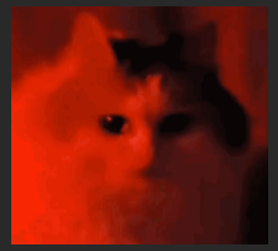
Attacker’s GitHub



ATTACKER SCENARIO



Party Cat
(Victim)



Angy Cat
(Attacker)

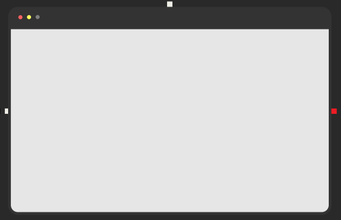


Photo Share App
(vulnerable app)

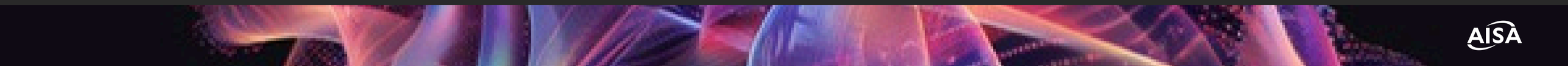
Angy cat
can login via their
Github

Attacker’s GitHub

ATTACKERS GOAL

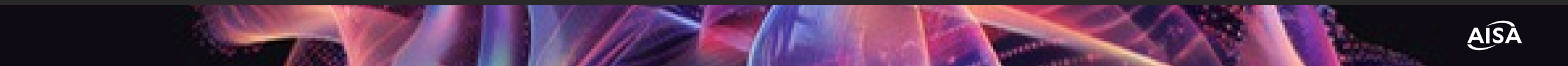
What - Login to Victim’s PhotoShare

How - Connect Attacker’s Social Media with Victim’s Photo Share App



ACCOUNT TAKEOVER ATTACK

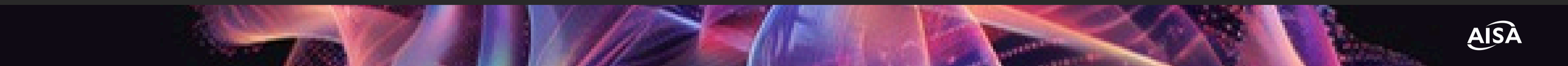
Demo



PRE-REQUISITE

The victim has a normal logged-in session with our app in the browser.

Haven’t connected socials login yet.



PLAYERS INVOLVED

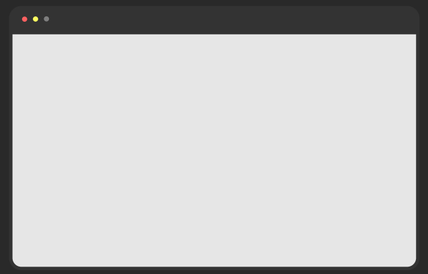
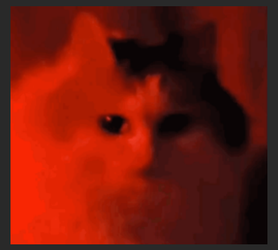
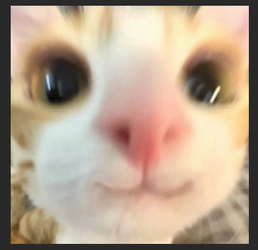


Photo Share App
(vulnerable app)



Attacker



Victim
(logged in user)

PLAYERS INVOLVED

① Initiates oauth request

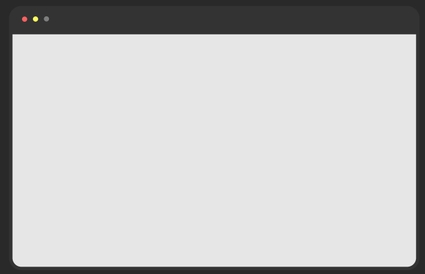
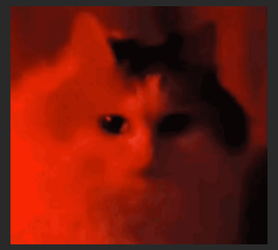
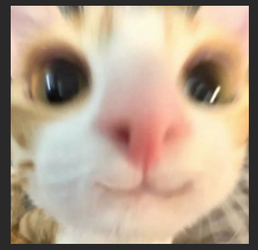


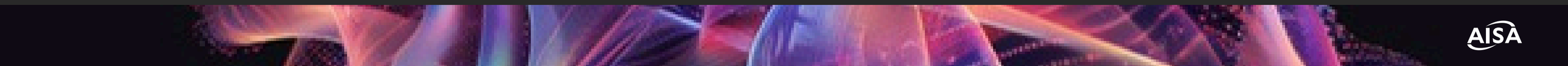
Photo Share App
(vulnerable app)



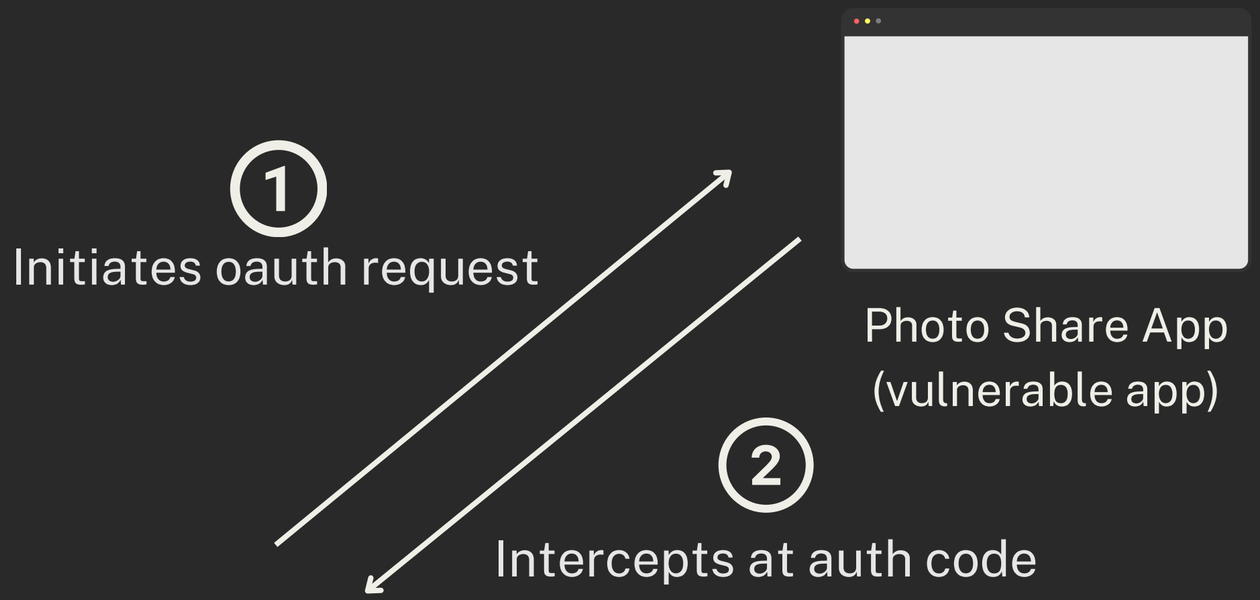
Attacker

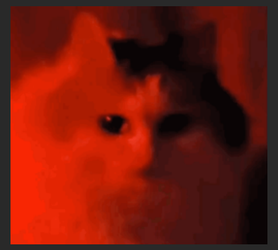


Victim
(logged in user)

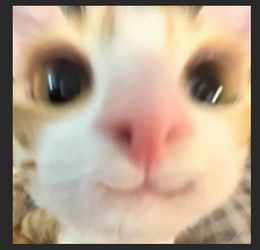


ATTACKER STOPS FLOW MIDWAY

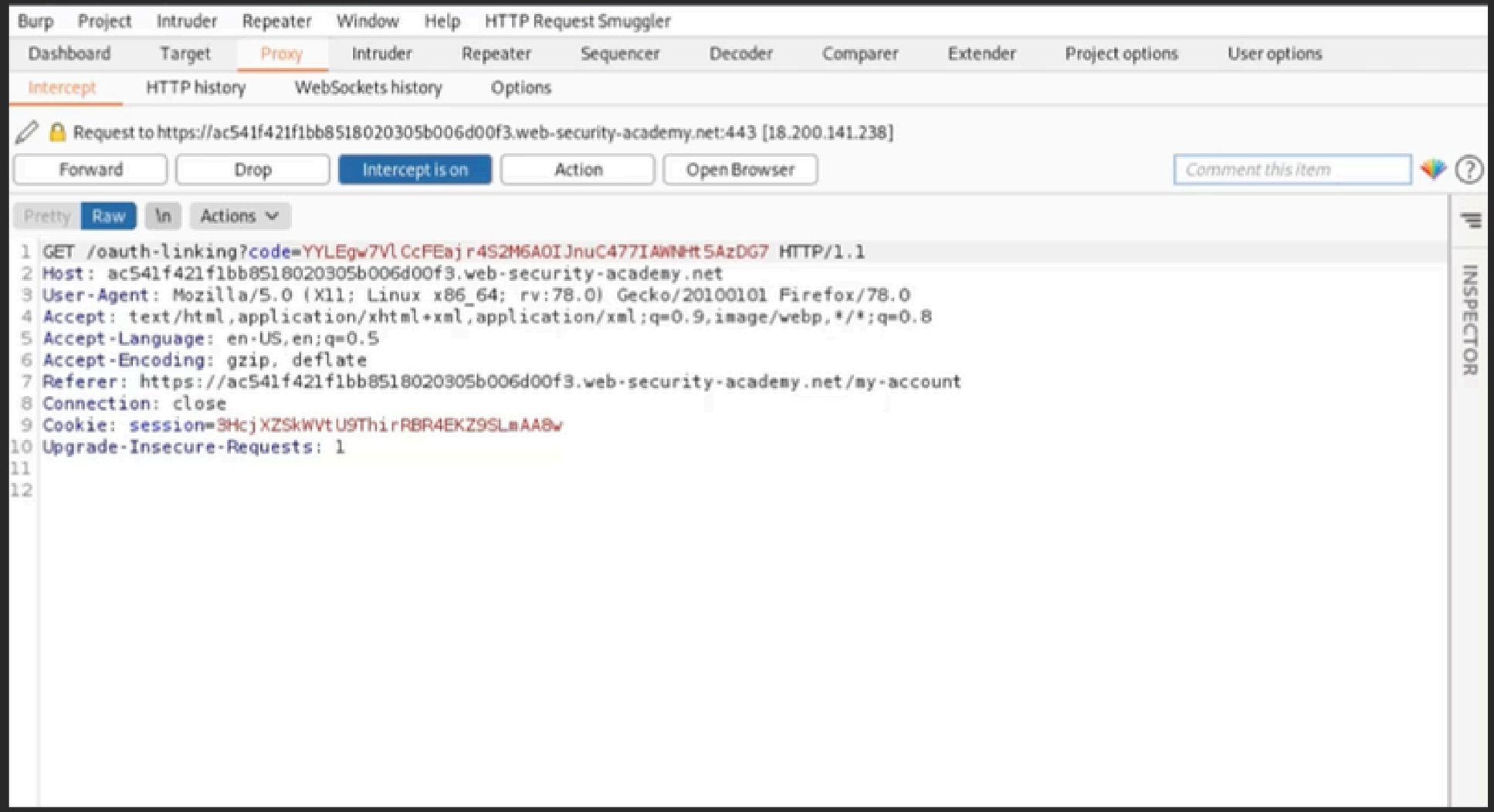




Attacker



Victim
(logged in user)

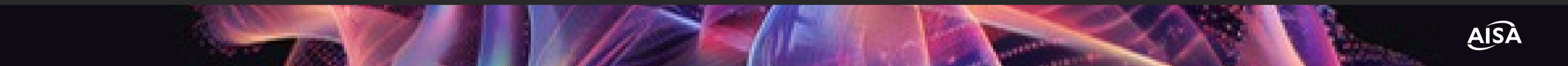


Example interception of an OAuth flow using Burp

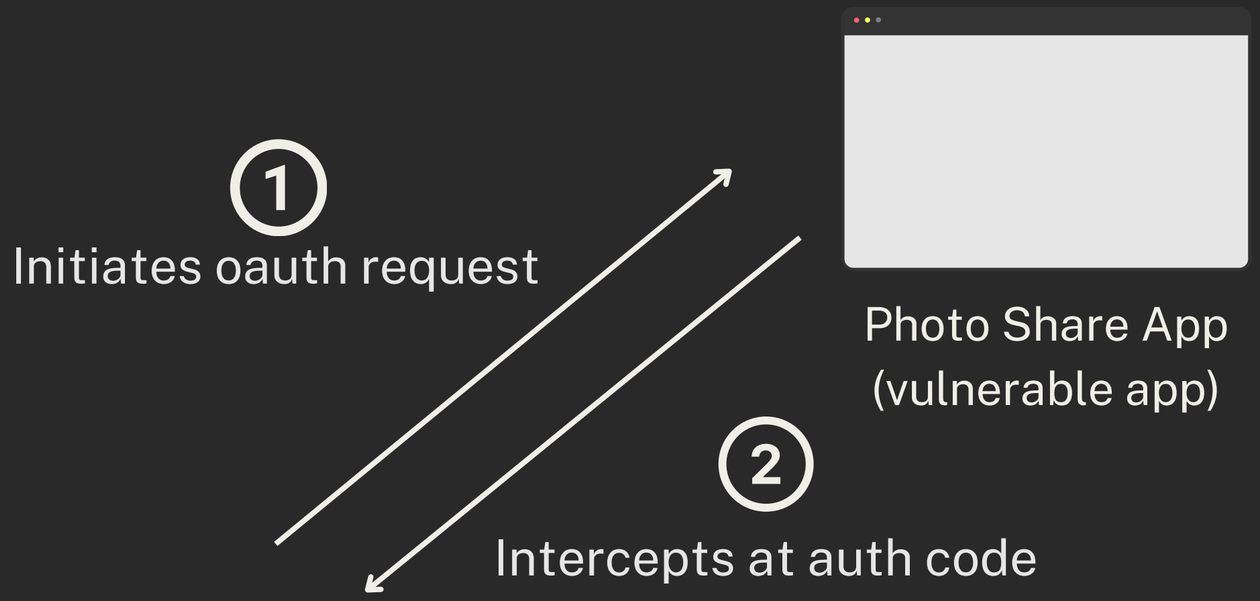
THE INTERCEPTED AUTH CODE

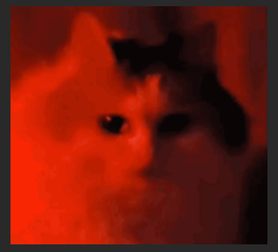
The code & token is linked to a user.

The server processes the flow/token for the linked user.



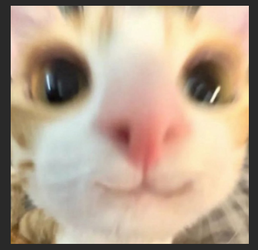
ATTACKER SENDS LINK TO VICTIM



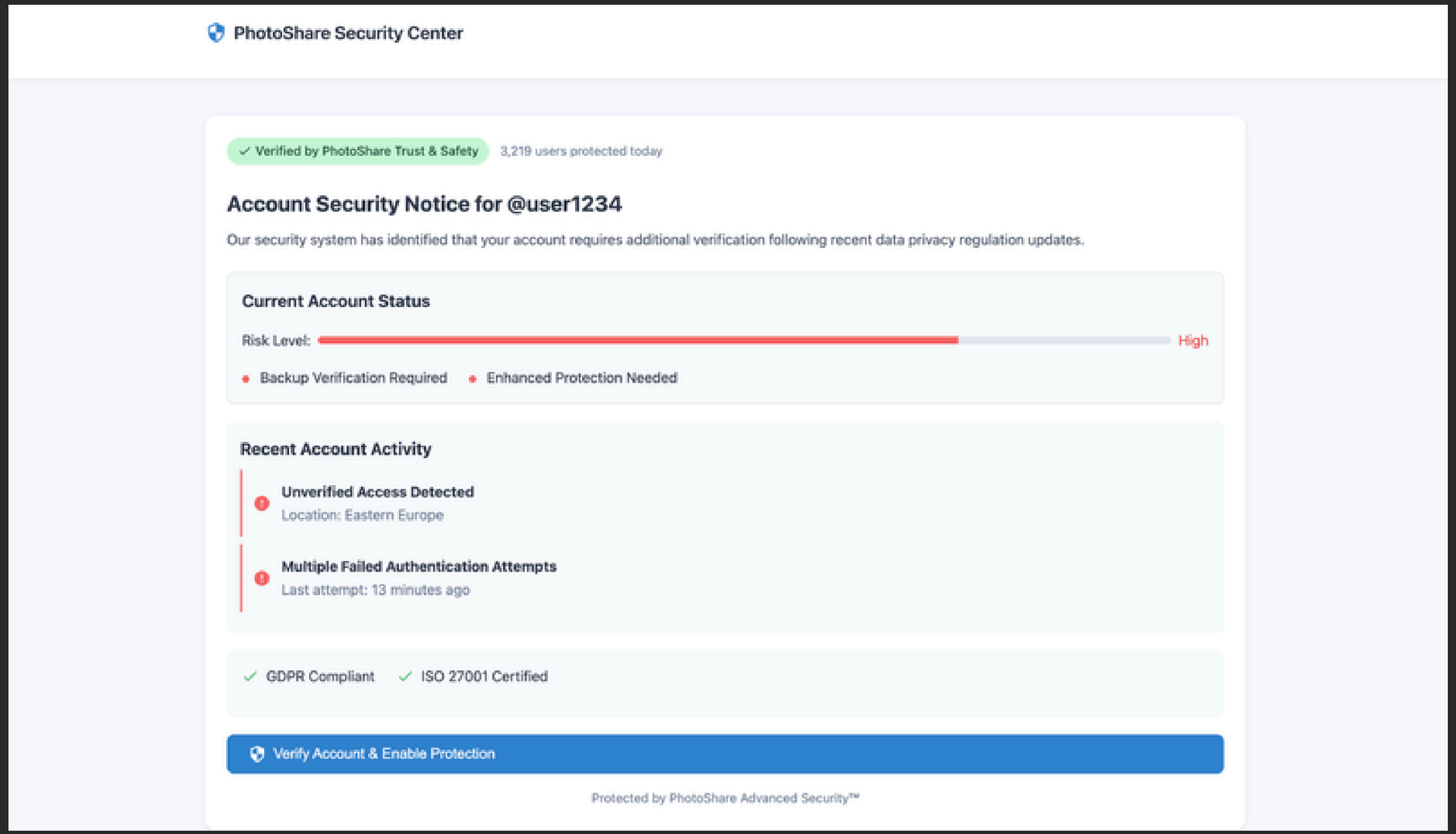


Attacker

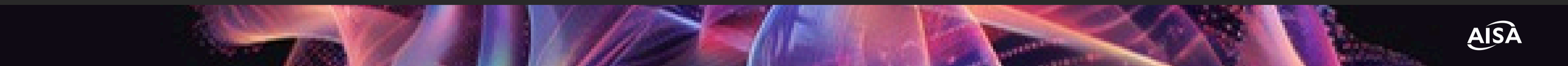
3 Phish user to click on link



Victim
(logged in user)



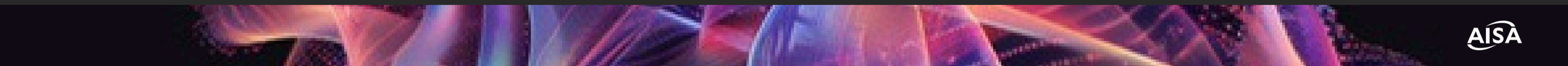
Example phishing site created using Claude



```html
<!DOCTYPE html>
<html>
<body>
  <h1>Win Free Photo Storage!</h1>
  <!-- Hidden iframe that silently triggers the OAuth flow -->
  <iframe style="display:none"
      src="http://photosharing.example.com/connect/thirdparty">
  </iframe>

  <!-- Or using JavaScript -->
  <script>
    // Automatically trigger the OAuth flow when page loads
    window.onload = function() {
      window.location.href =
        'http://photosharing.example.com/connect/thirdparty';
    }
  </script>
</body>
</html>
```

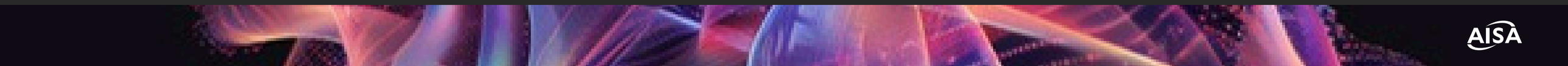
OAuth flow will triggered in the background when victim visits phishing link



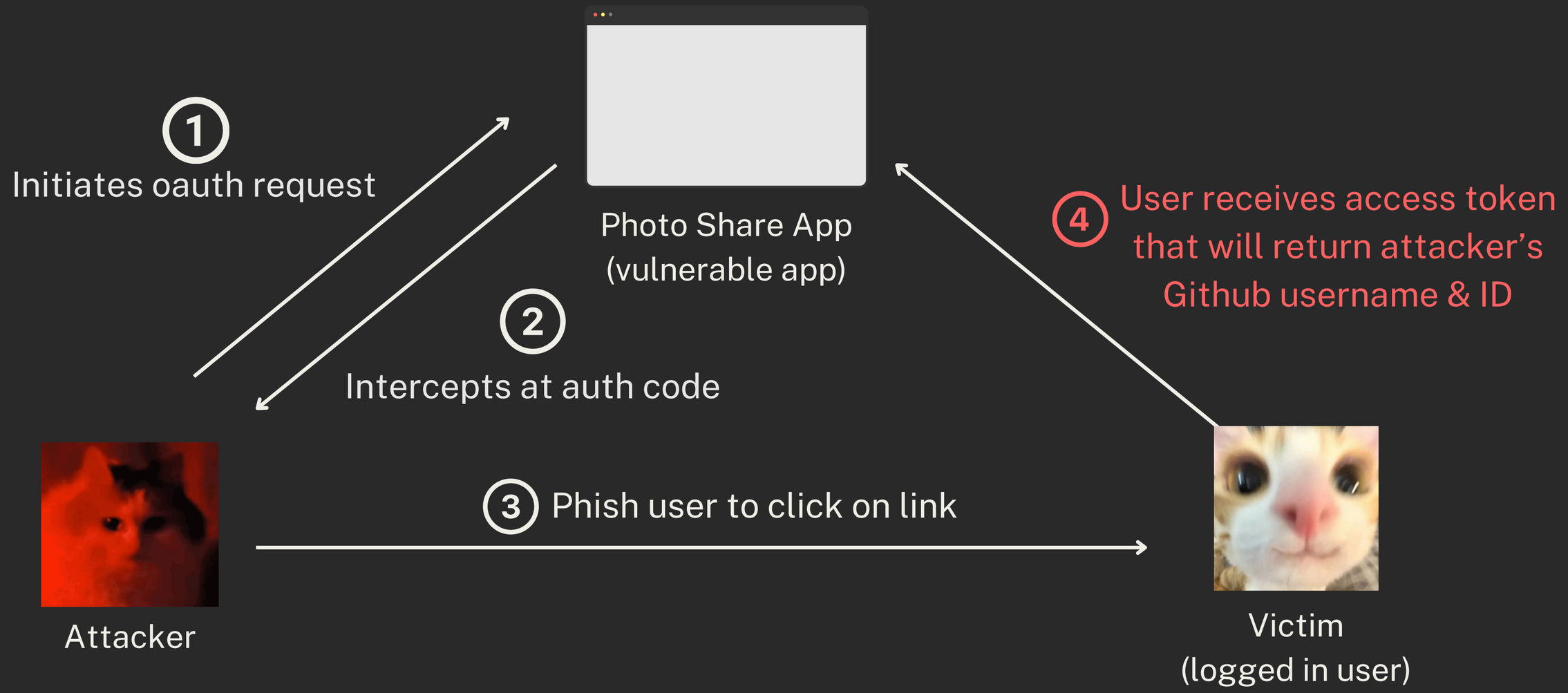
PRE-REQUISITE

The victim has a normal logged-in session with our app in the browser.

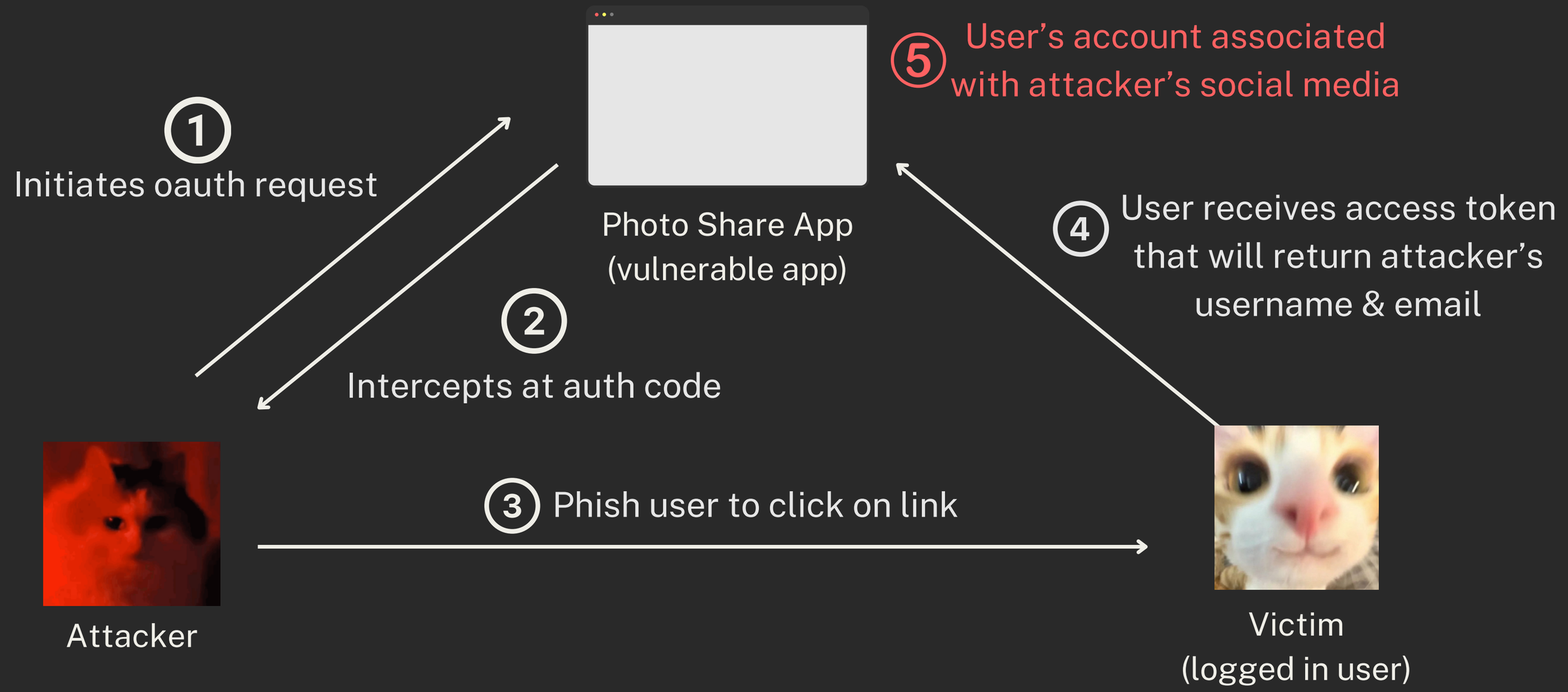
Haven’t connected socials login yet.



VICTIM'S LOGGED IN SESSION GETS CONNECTED

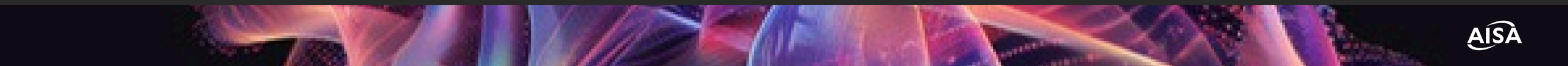


UNAUTHROISED ACCESS

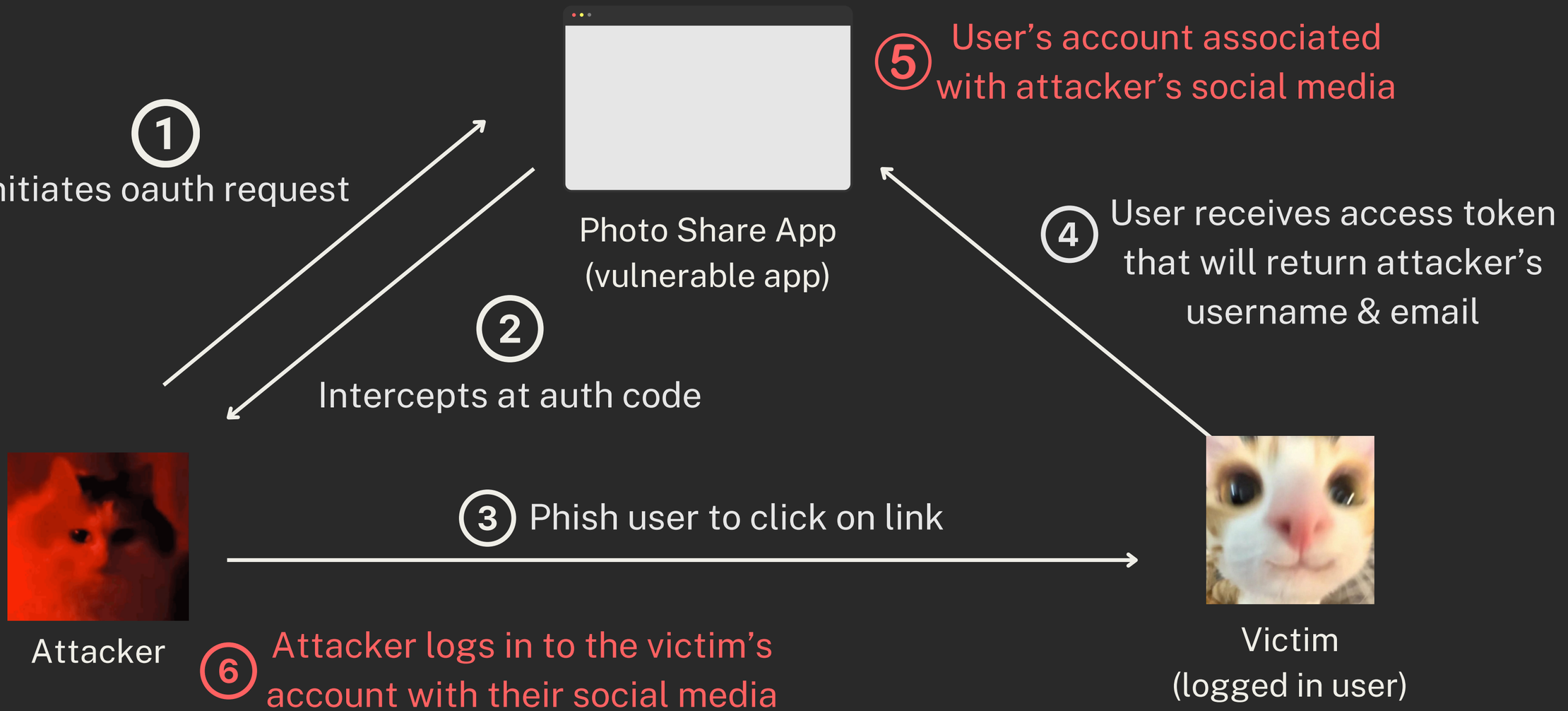


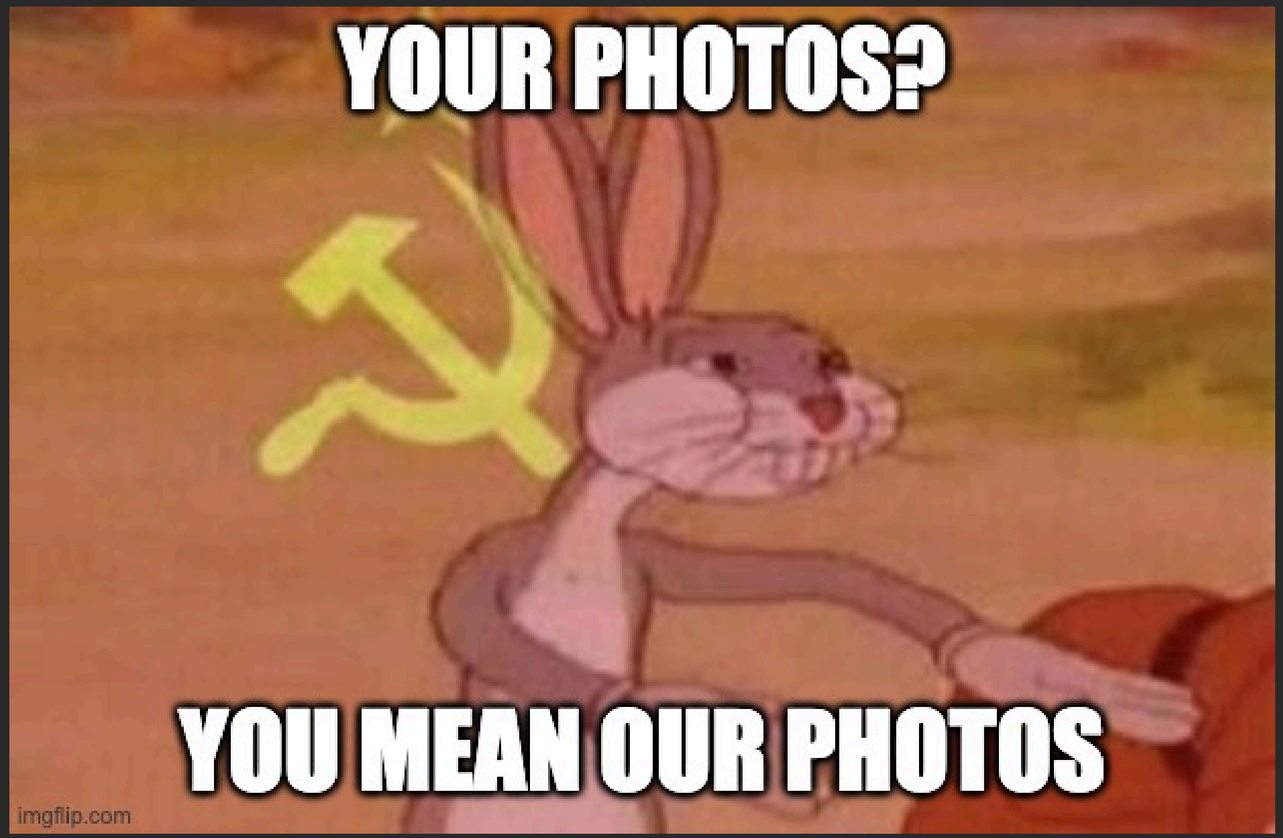
Our app will **assign**

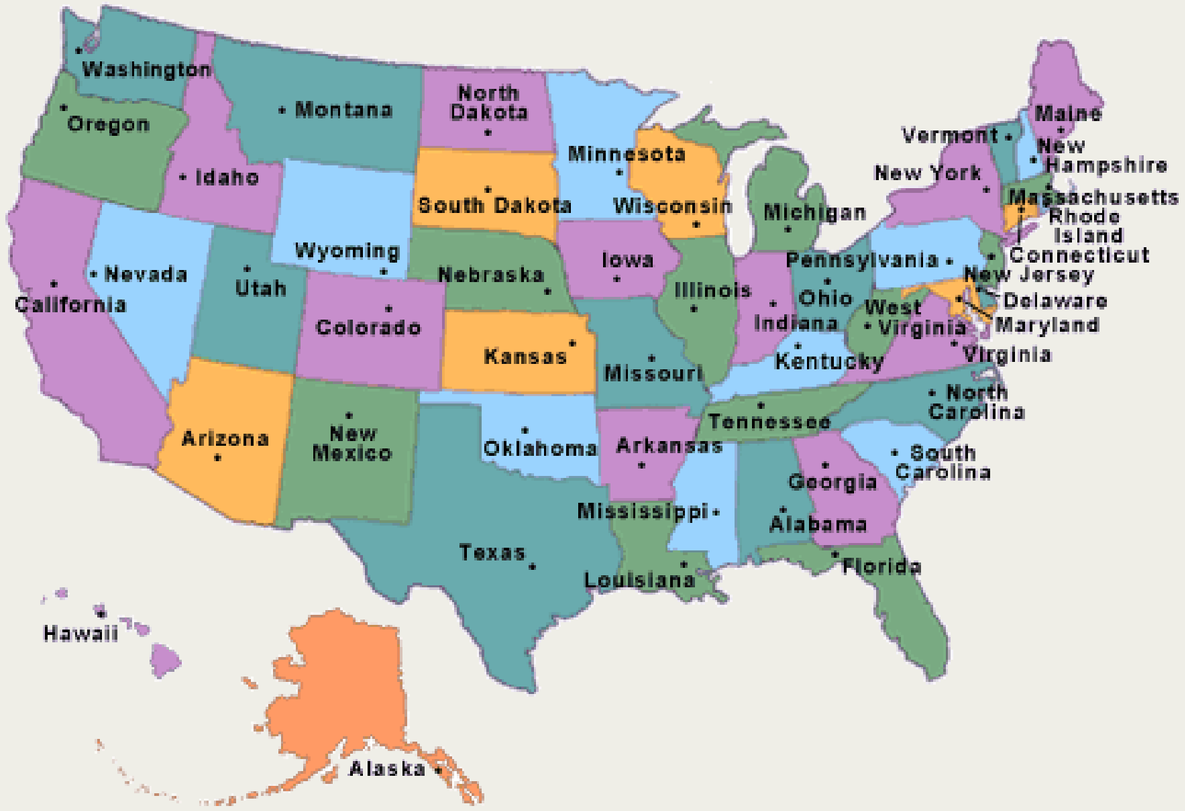
Attacker's **GitHub id/username** == **current logged-in session** in the browser (The victim).



UNAUTHROISED ACCESS



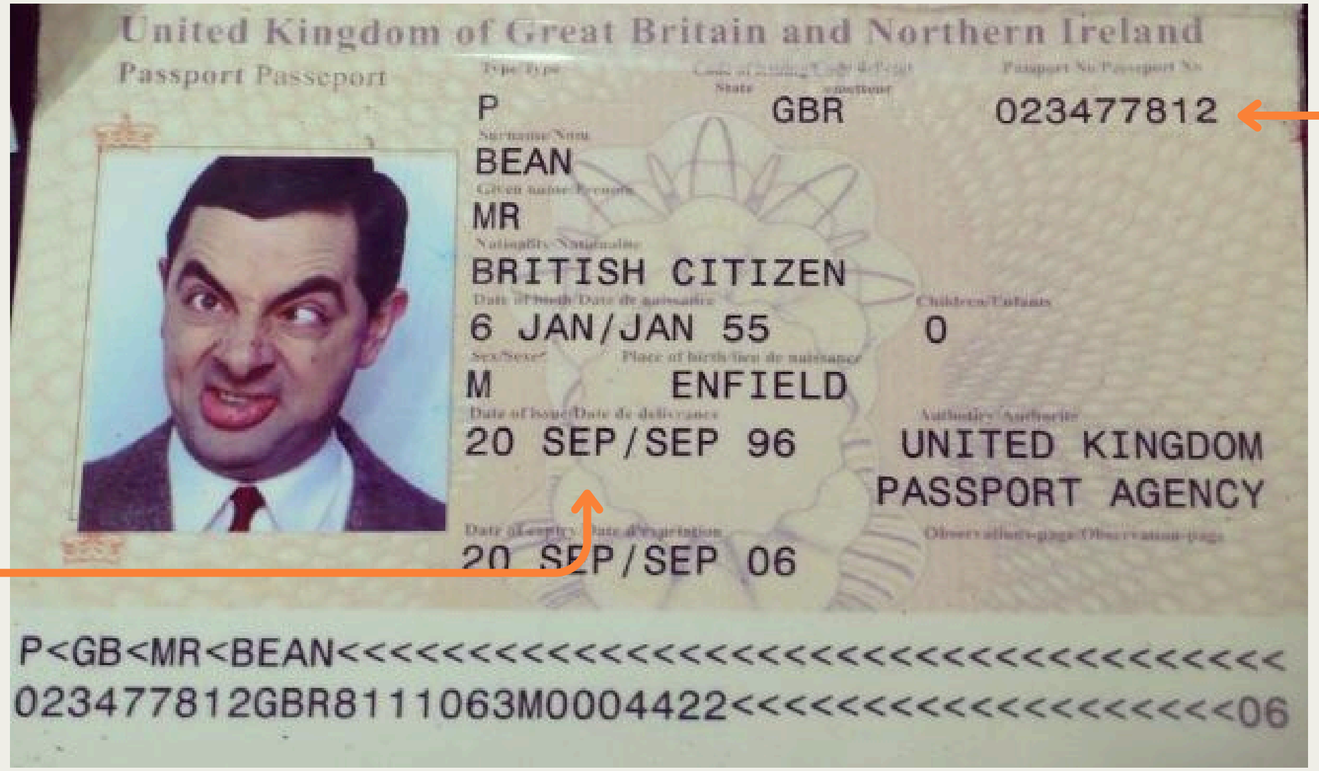

YOUR PHOTOS?
YOU MEAN OUR PHOTOS
imgflip.com


Washington
Oregon
Montana
North Dakota
Minnesota
Vermont
Maine
New Hampshire
Idaho
South Dakota
Wisconsin
Michigan
New York
Massachusetts
Rhode Island
Wyoming
Nebraska
Iowa
Pennsylvania
Connecticut
New Jersey
Nevada
Utah
Illinois
Ohio
Delaware
California
Colorado
Indiana
West Virginia
Maryland
Kansas
Missouri
Kentucky
Virginia
North Carolina
Arizona
New Mexico
Oklahoma
Arkansas
Tennessee
South Carolina
Georgia
Mississippi
Alabama
Texas
Louisiana
Florida
Hawaii
Alaska

The state parameter

Using a randomly generated unique parameter to tie a request to a specific user

THE STATE PARAMETER



State parameter

(Passport number)

Associated

values

(Name, expiry etc.)

THE STATE PARAMETER

Booking.com

You made a booking online.

THE STATE PARAMETER

Booking.com

You made a booking online.

You arrive at the Hotel to check-in

THE STATE PARAMETER

Booking.com

You made a booking online.

You arrive at the Hotel to check-in

How does the reception know who booked?

THE STATE PARAMETER

Booking.com

You made a booking online.

You arrive at the Hotel to check-in

How does the reception know who booked?

They check the passport to match the name

HOW THE STATE PARAMETER WORKS

1. At the start - a **random value** and **tie it** to the **user.** 🔐

2. When an **auth request** comes back - **check** the **state** parameter with the requesting **user.** 🔎

OUR ORIGINAL REQUEST

https://dummysocialmedia.com/oauth/authorize?client_id=4872984729472947&
redirect_uri=https://photoshare.com/oauth/callback&scope=email,public_profile,photos&
response_type=code&access_type=offline&prompt=consent&flowName=social_login

Does not have any state parameter ❌

NEW SECURE REQUEST

https://dummysocialmedia.com/oauth/authorize?**client_id**=487298472947 2947&
redirect_uri=https://photoshare.com/oauth/callback&**scope**=email,public_profile,photos&
response_type=code&**access_type**=offline&**prompt**=consent&**flowName**=social_login&
state=5ca75bd30

A randomly generated unique value

GENERATING THE STATE

```python
def generate_state(self, user_id: str) -> str:
    """Generate and store state parameter"""
    state = secrets.token_urlsafe(32)

    # Store state with user ID and timestamp
    self.redis_client.setex(
        f'oauth_state:{state}',
        300,  # 5 minute expiry
        json.dumps({
            'user_id': user_id,
            'created_at': datetime.utcnow().isoformat(),
            'client_ip': request.remote_addr
        })
    )
```

Storing the **state** with **the corresponding user ID** and **timestamp**

STATE PARAMETER EVERYWHERE

https://ourexampleapp.com/oauth/callback? **code**=4/0AfJohXkBE-YXVPrRfRk4j9xlIryTNKtJMHRmUsr5UQF3OzYtCHDUrztCc4G1q3nrdDz3MQ&**scope**=email profile https://www.googleapis.com/auth/photoslibrary.readonly&**state**=5ca75bd30

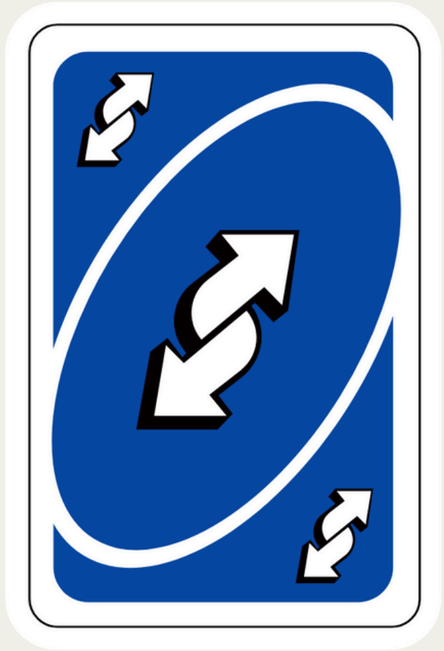
State parameter **present** in **follow-up requests**

to Auth server


SentrySafe

CHANGE OF ROLES

Instead of **our app asking** for data



Let’s consider **other apps asking us** for users' data.

DATA SHARING SCENARIOS

Better UX

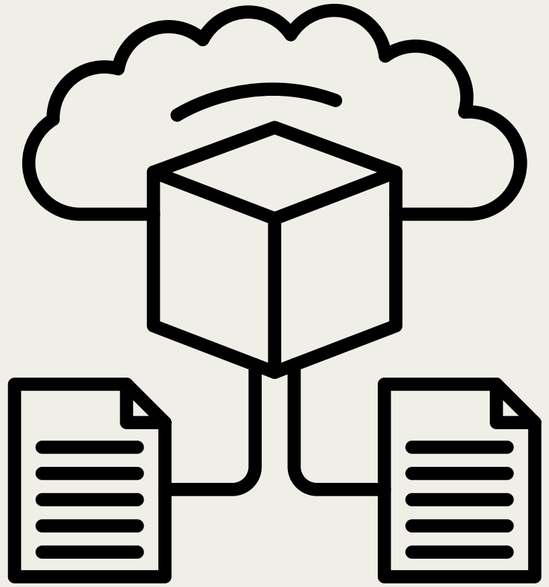
- Jira issue previews are available on GitHub.

Actions on behalf of the user

- Send Slack updates when the Jira issue is closed.

Plugins

- 3rd party plugins for our app.



✨ Benefit ✨

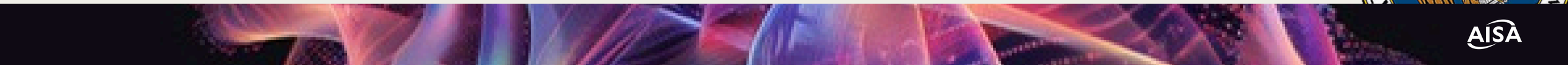
Not sharing passwords with random applications

SCOPES – FINE GRAINED ACCESS

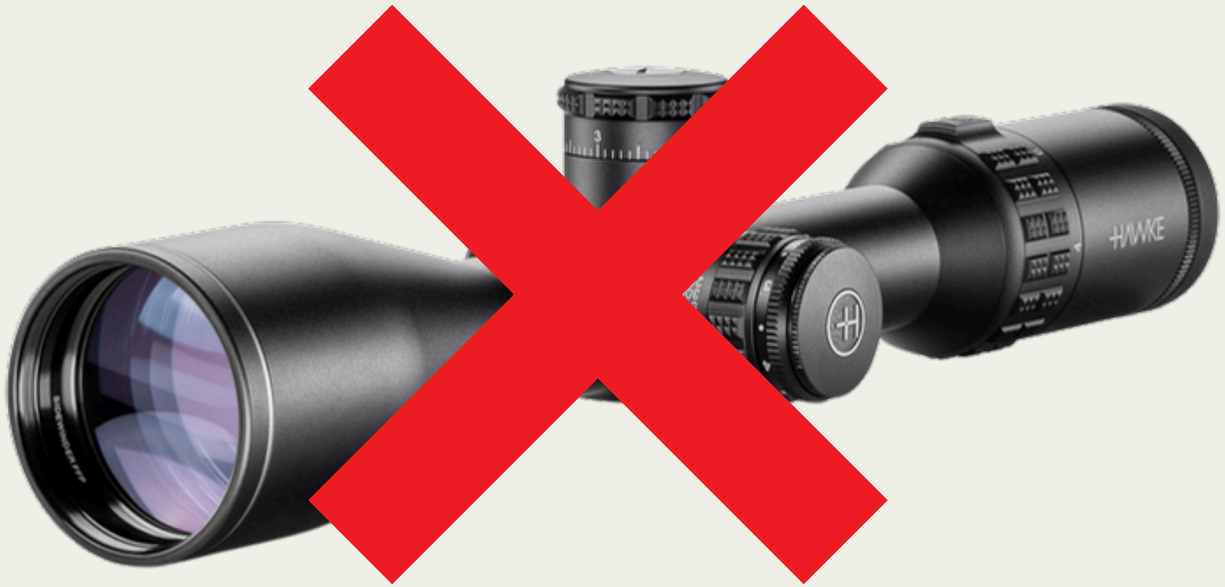
- [x] Read your files
- [x] Help with tasks

SCOPES – FINE GRAINED ACCESS

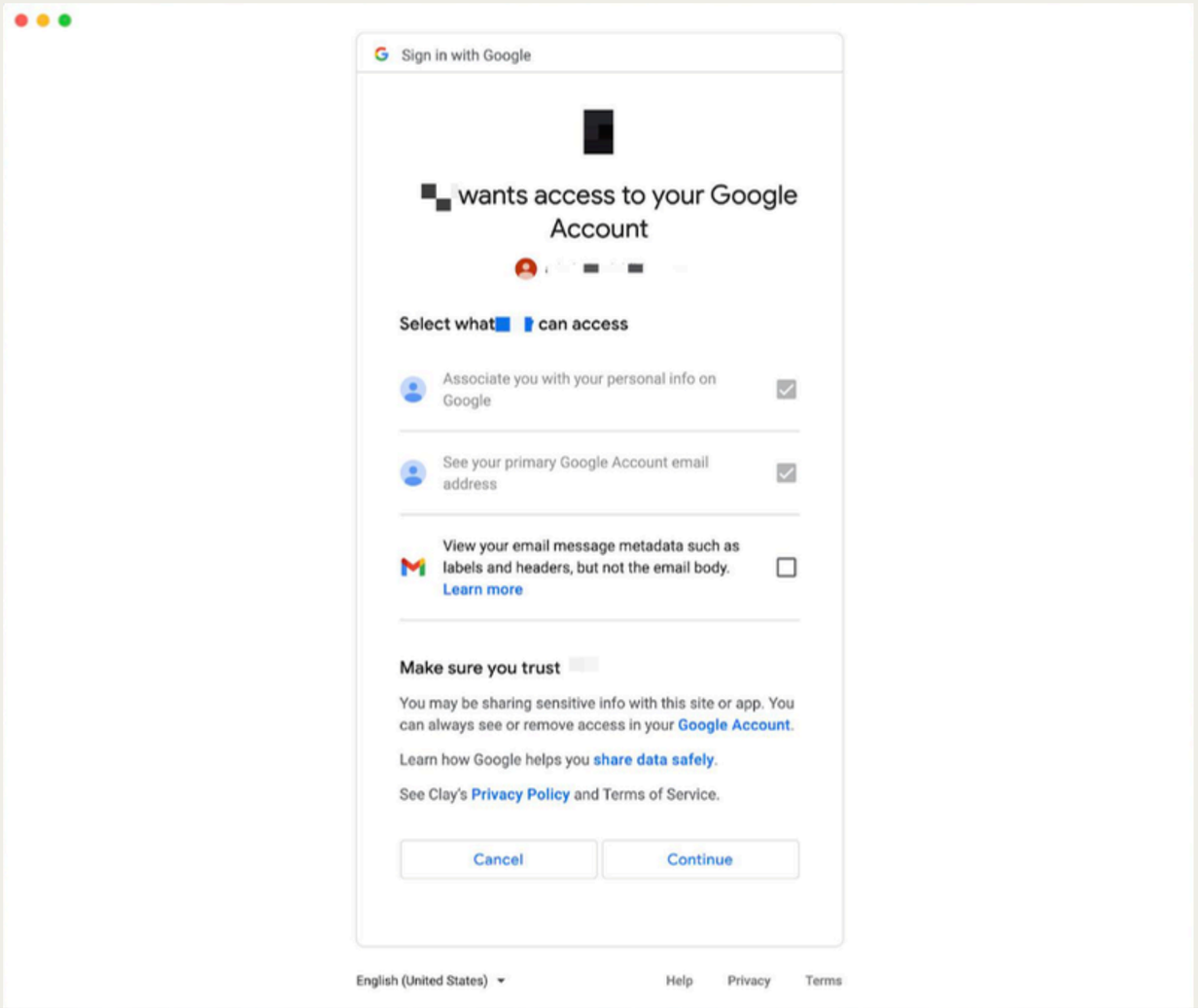
- ☑ Read your files
- ☑ Help with tasks
- ⊗ Backdoor your computer




HAWKE


HAWKE

EXAMPLE SCOPE CHECK



SCOPES – FINE GRAINED ACCESS

With great power comes great responsibility

- Uncle Ben (circa 2002)

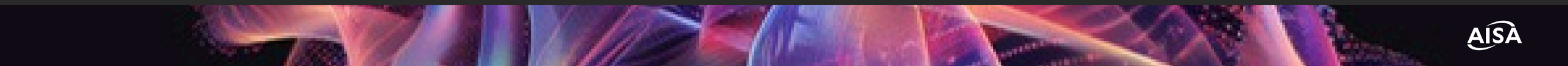
PITFALL

Broken OAuth Scopes

Mishandling the power that lies within

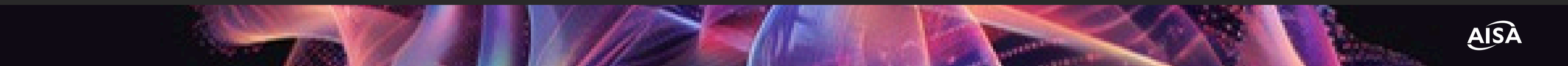
SCOPE PITFALL #1

Not validating scope on each request



SCOPE UPGRADE ATTACK

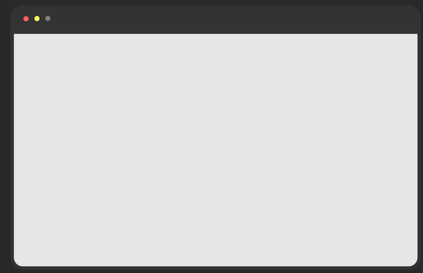
A malicious plugin/app can **request elevated scopes** than what the user approved



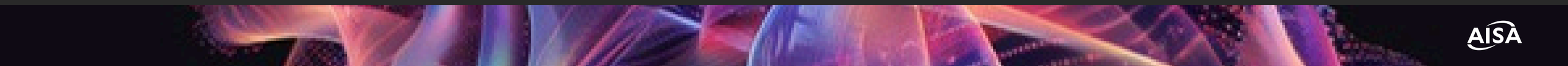
SCOPE UPGRADE ATTACK

Request Auth Code
Scope : { read_profile}

Malicious App



Our App
(Resource server)



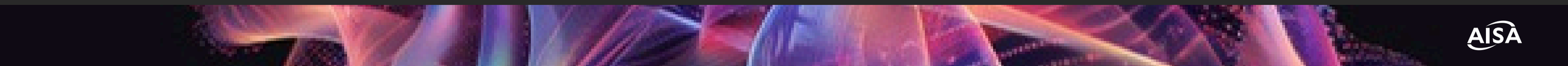
SCOPE UPGRADE ATTACK

Request Auth Code
Scope : { read_profile}

Send Auth Code

Malicious App

Our App
(Resource server)



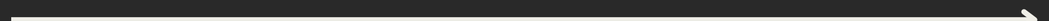
SCOPE UPGRADE ATTACK

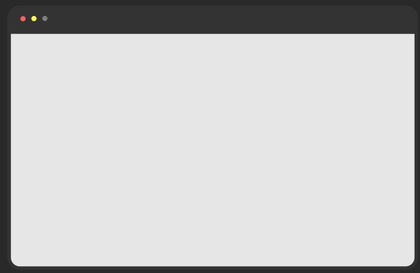
Request Auth Code
Scope : { read_profile}

Send Auth Code

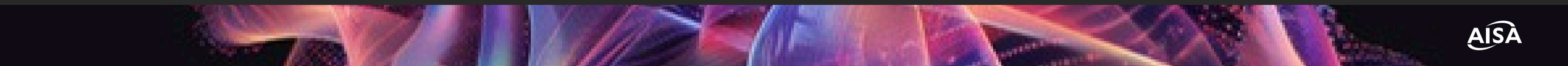
Malicious App

Request Access Code
Scope : { read_profile, write_profile, admin}





Our App
(Resource server)



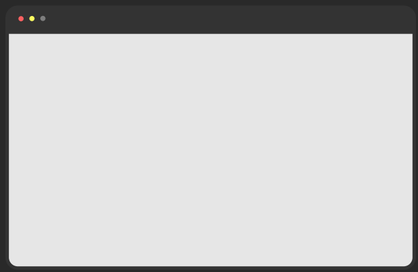
SCOPE UPGRADE ATTACK

Request Auth Code
Scope : { read_profile}

Send Auth Code

Malicious App

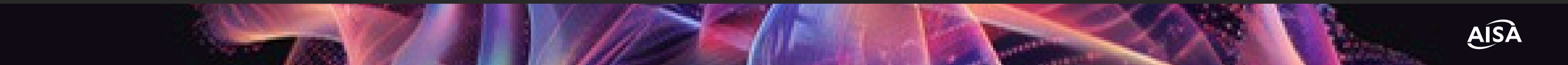
Request Access Code
Scope : { read_profile, write_profile, admin}



Our App
(Resource server)

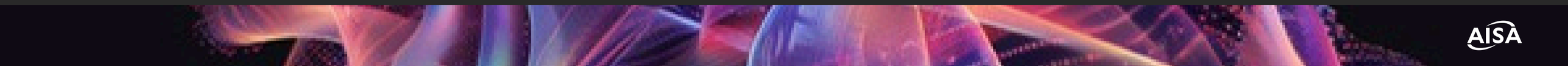
❌ No Scope revalidaiton

Send Elevated Access Code



SCOPE UPGRADE ATTACK

Demo



SOLUTION

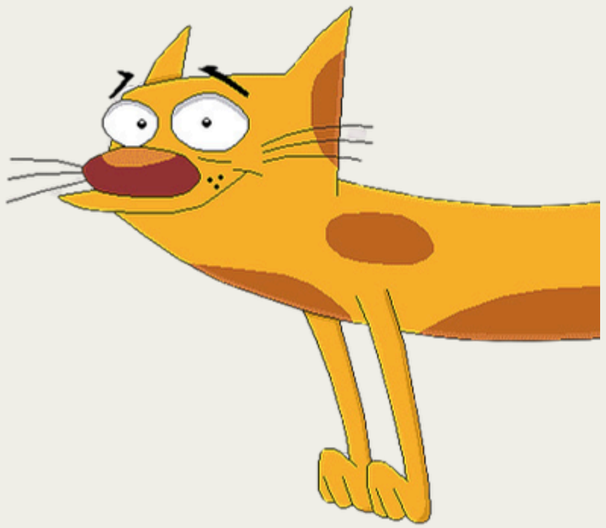
Tracking scopes in every major step.

Scopes can be tied with the **state** parameter.

SCOPE VALIDAITON

When to validate

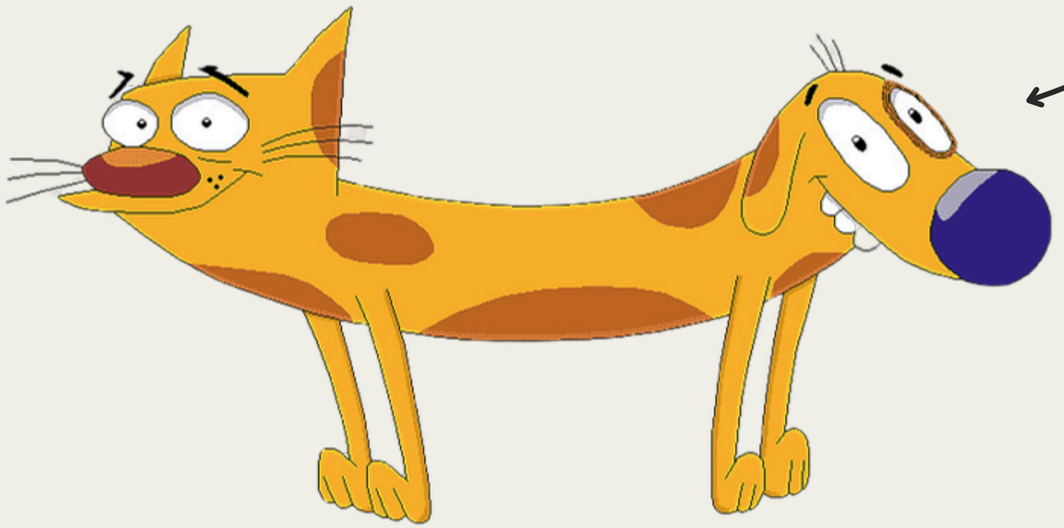
the scope



SCOPE VALIDAITON

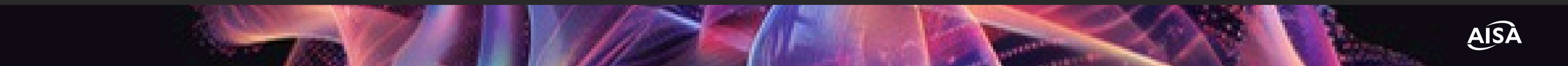
When to validate the scope

What scope to validate



SCOPE PITFALL #2

Scope Explosion



SCOPE EXPLOSION

When our app accumulates an **unmanageable** number of granular scopes.

SCOPE EXPLOSION

```
// App Scope
const expandedScopes = [
  'users:read',
  'users:write',
  'users:delete',
  'users.profile:read',
  'users.profile:write',
  'users.settings:read',
  'users.settings:write'
];
```

Initial list of scopes

SCOPE EXPLOSION

```
// App Scope
const expandedScopes = [
  'users:read',
  'users:write',
  'users:delete',
  'users.profile:read',
  'users.profile:write',
  'users.settings:read',
  'users.settings:write'
];
```

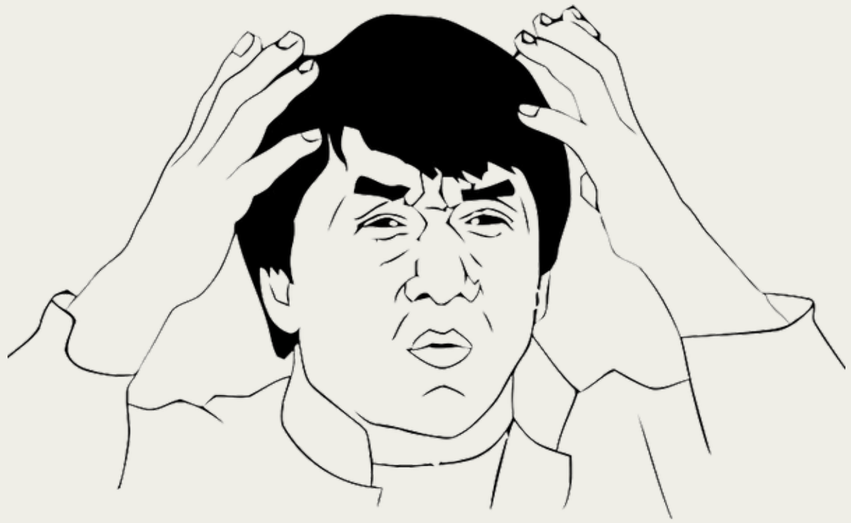
Initial list of scopes

```
// After adding organization features
const explodedScopes = [
  'users:read',
  'users:write',
  'users:delete',
  'users.profile:read',
  'users.profile:write',
  'users.settings:read',
  'users.settings:write',
  'org:read',
  'org:write',
  'org:delete',
  'org.members:read',
  'org.members:write',
  'org.teams:read',
  'org.teams:write',
  'org.teams.members:read',
  'org.teams.members:write',
  // ... and it keeps growing
];
```

Scope sprawl

SCOPE EXPLOSION

```
// Client trying to request multiple specific permissions
const authorizationUrl = 'https://auth-server.com/oauth2/authorize?' +
  'response_type=code&' +
  'client_id=client123&' +
  'scope=' + encodeURIComponent(
    'users:read ' +
    'users.profile:read ' +
    'users.settings:read ' +
    'org:read ' +
    'org.teams:read ' +
    'org.teams.members:read'
  ) +
  'redirect_uri=https://app.example.com/callback';
```



Easy to misconfigure

SCOPE EXPLOSION

Quite useful

SCOPE EXPLOSION

Quite useful

Difficult to navigate

SOLUTION – IMPLEMENT SCOPE HIERARCHY

```
class HierarchicalScopeSystem:
    def __init__(self):
        self.scope_tree = {
            'admin': {
                'read': True,   # Terminal scope
                'write': True,  # Terminal scope
                'users': {
                    'read': True,
                    'write': True,
                    'profile': {
                        'read': True,
                        'write': True
                    }
                }
            }
        }
```

Permissions are organized in a tree structure

Benefits

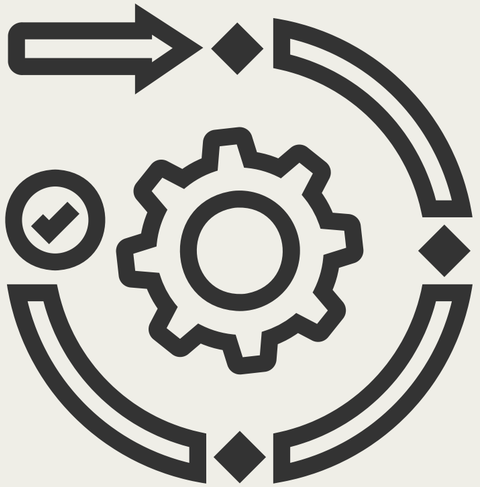
- Allows you to group scopes
- Easy to add / maintain / analyse

OTHER SOLUTIONS TO CONSIDER

Other cool things that we don’t time to talk about

- Implementing scope templates.

- Monitoring scope usage & audits.



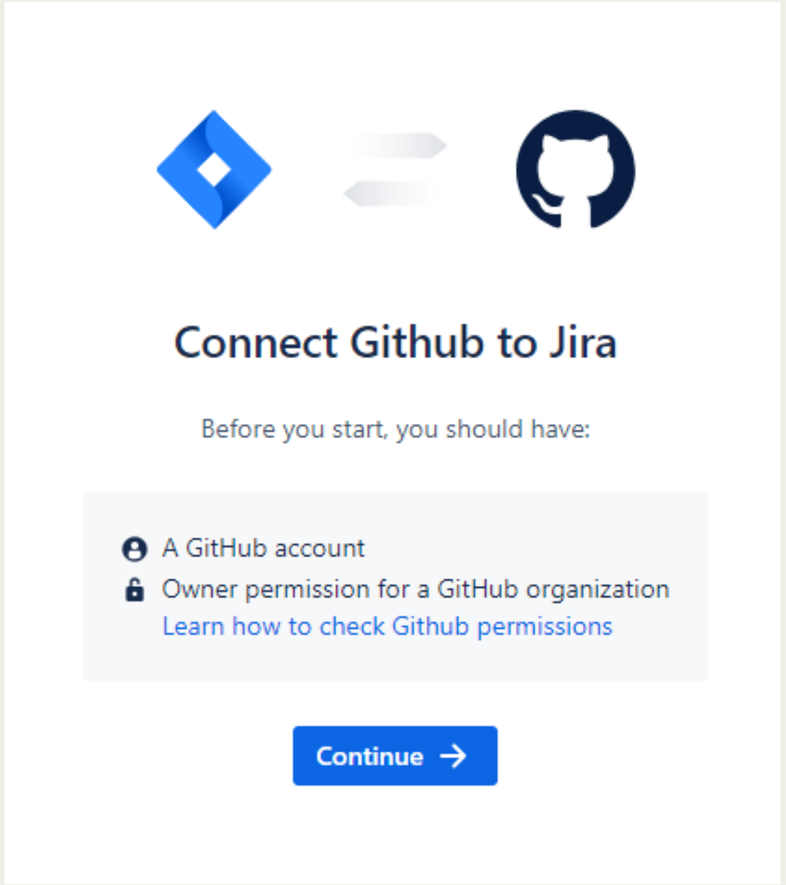
TAKEAWAYS

Systematic way of **managing** scopes.

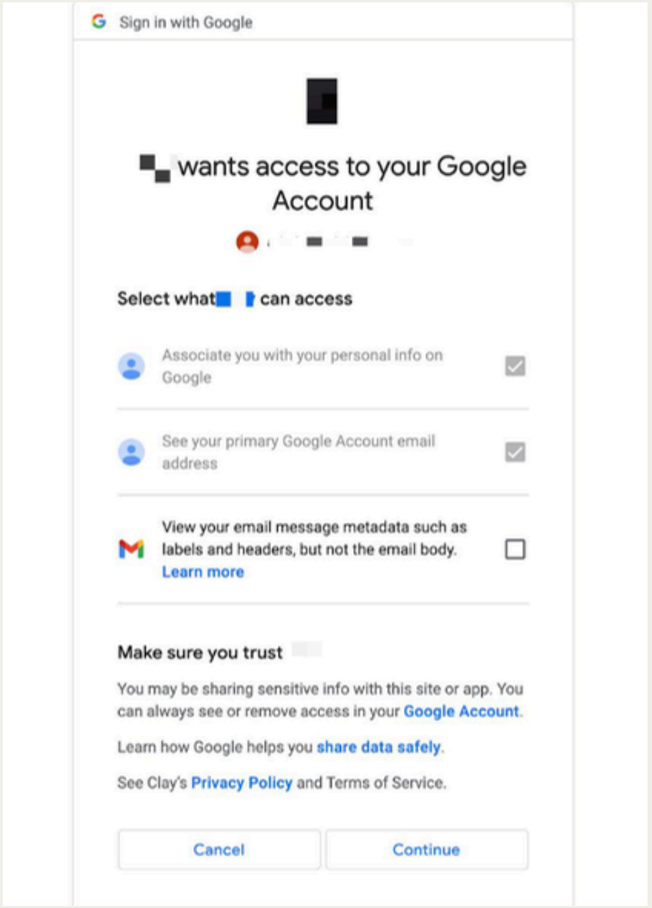
Ensuring we are **enforcing** scopes at **every step** of the flow.

Let’s talk about another OAuth super power

WHAT THE USER SEES

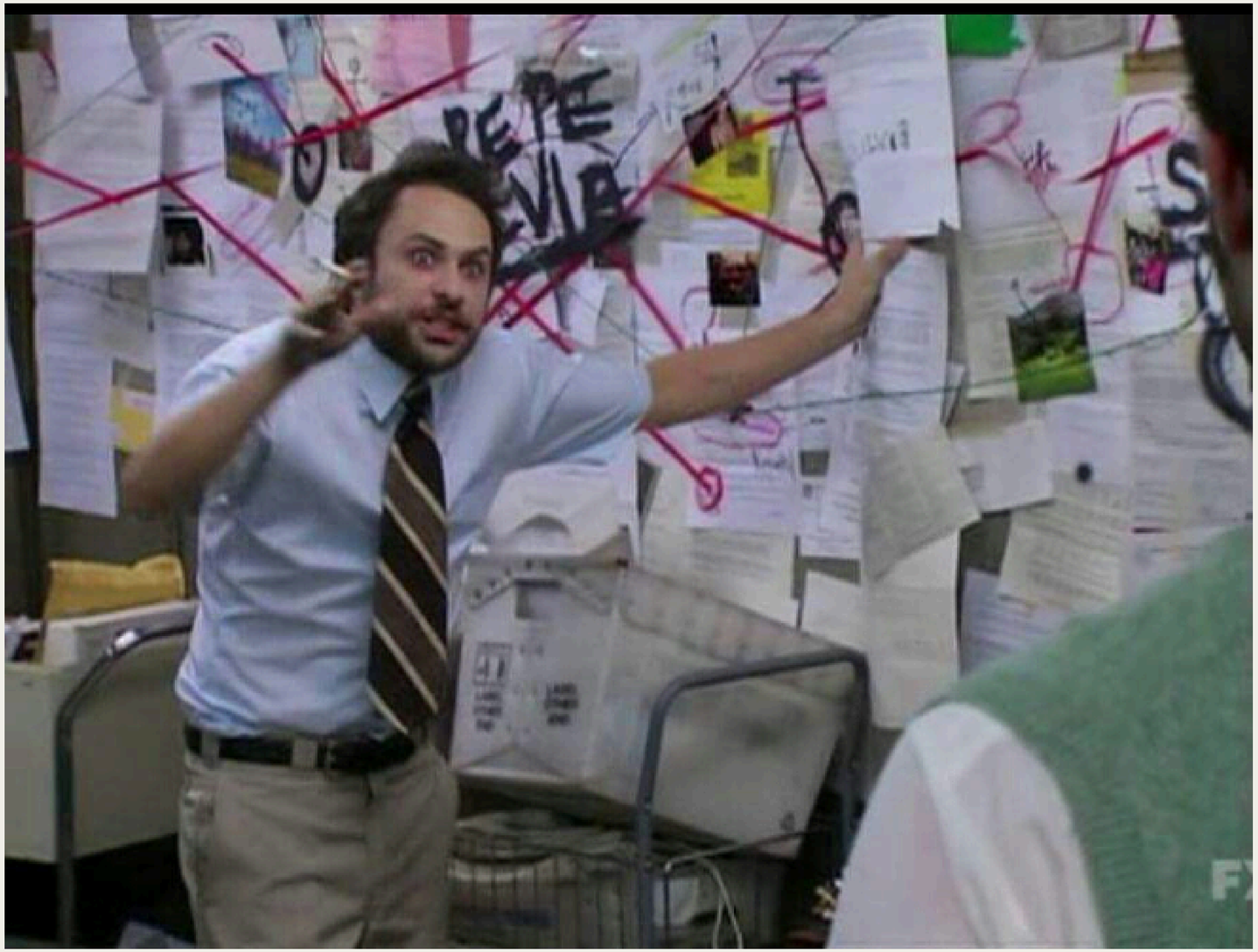


#1 Initiation



#2 Consent

OAUTH UNDER THE HOOD


FX

✨Benefit✨

OAuth provides a seamless flow utilising the **Redirect / Callback** URI

REDIRECT URI

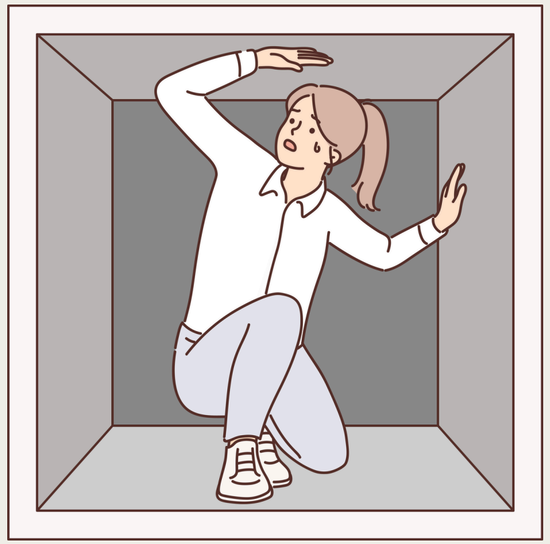
https://dummysocialmedia.com/oauth/authorize?**client_id**=487298472947 2947&
redirect_uri=https://photoshare.com/oauth/callback&**scope**=email,public_profile,photos&
response_type=code&**access_type**=offline&**prompt**=consent&**flowName**=social_login

The URL we configure to send the user back to original screen.

PREREQUISITE

The OAuth client & providers need to **securely configure** the **redirect_uri**


USER COMING
BACK VIA REDIRECT URL
PHISHING SITE
TOKEN THEFT
SSRF



Improper redirect URI config

Where did it come from, where did it go

If redirect_uri misconfigs were people

THE "REGEX MASTER"

```
# Feeble attempt to match subdomains
ALLOWED_REDIRECT = "https://*.myapp.com/*"

def validate_uri(uri):
    return uri.matches(ALLOWED_REDIRECT)  # What could go wrong?
    # Narrator: Everything went wrong
```

Attempt to cater for subdomains

leading to too open whitelist

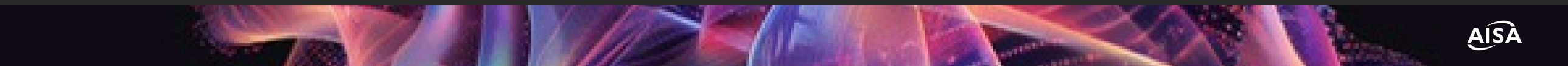
SAMPLE ATTACK URLS

https://example.com/?client_id=CLIENT&redirect_uri=client-app.com/callback&redirect_uri=attacker.com.myapp.com/callback

Crafting a regex bypass

https://example.com/?client_id=CLIENT&redirect_uri=client-app.com/callback&redirect_uri=evil-user.net

Double redirect attacks



THE "LOCALHOST" LOVER

```
Accidentally leaving test servers
ALLOWED_REDIRECTS = [
    "https://myapp.com/callback",
    "http://localhost:*"  # For "testing" 🤡
]
```

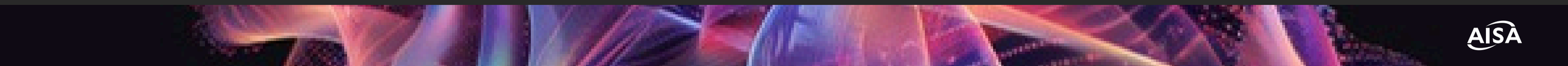
Accidentally leaving localhost or test servers in the whitelist

SAMPLE ATTACK URLS

https://example.com/oauth/connect/facebook? client_id=CLIENT_ID&
redirect_uri=http://localhost.attacker.com/steal& response_type=code

Attacker can perform DNS Rebinding

attacks



TAKEAWAYS

Exact matches are your friends.

Whitelist of allowed redirect_uri (when possible)

If you think **regex** will solve your security problems, you will have two problems now.

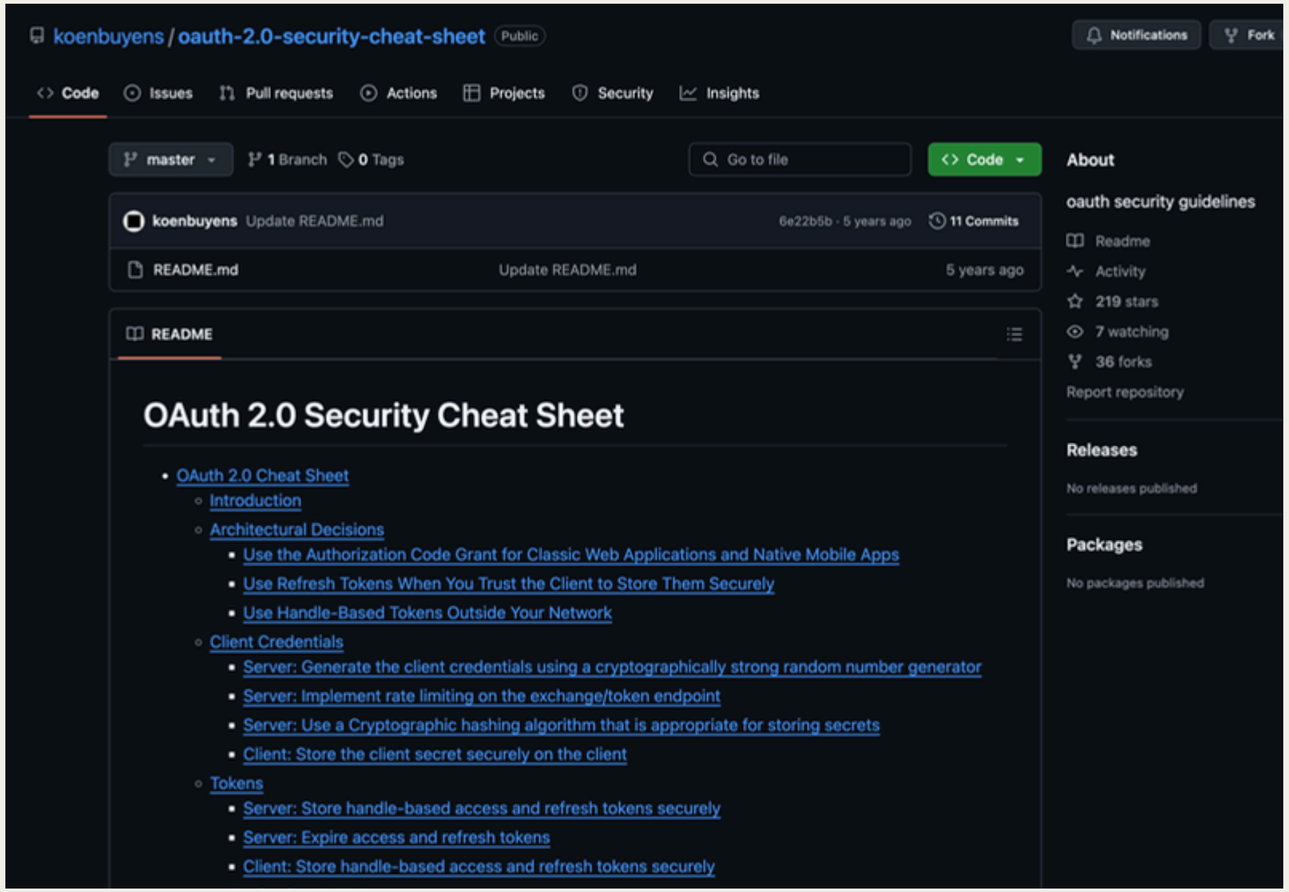
IN SUMMARY

With great power comes great responsibility

- Uncle Ben (circa 2002)

DEVELOPER CHECKLIST

koenbuyens/oauth-2.0-security-cheat-sheet



WHAT I DIDN’T TOUCH ON

OIDC

Bunch of other OAuth issues

IN-DEPTH SECURITY

RFC 6819

Internet Engineering Task Force (IETF)
Request for Comments: 6819
Category: Informational
ISSN: 2070-1721

T. Lodderstedt, Ed.
Deutsche Telekom AG
M. McGloin
IBM
P. Hunt
Oracle Corporation
January 2013

OAuth 2.0 Threat Model and Security Considerations

Abstract

This document gives additional security considerations for OAuth, beyond those in the OAuth 2.0 specification, based on a comprehensive threat model for the OAuth 2.0 protocol.

Status of This Memo

Thank You!

Any Questions?