All Things Auth

How (not) to do authentication in modern applications

9th January 2020, Dennis Stötzel

whoami

- Born in Germany
- Lives in Đà Nẵng, Việt Nam
- Soon back in Germany

- Studied Mathematics in Munich, Germany

- 9 years security consulting and development
- Team lead of security team in Đà Nẵng for the last 2 years
- Teacher at Đà Nẵng University of Technology

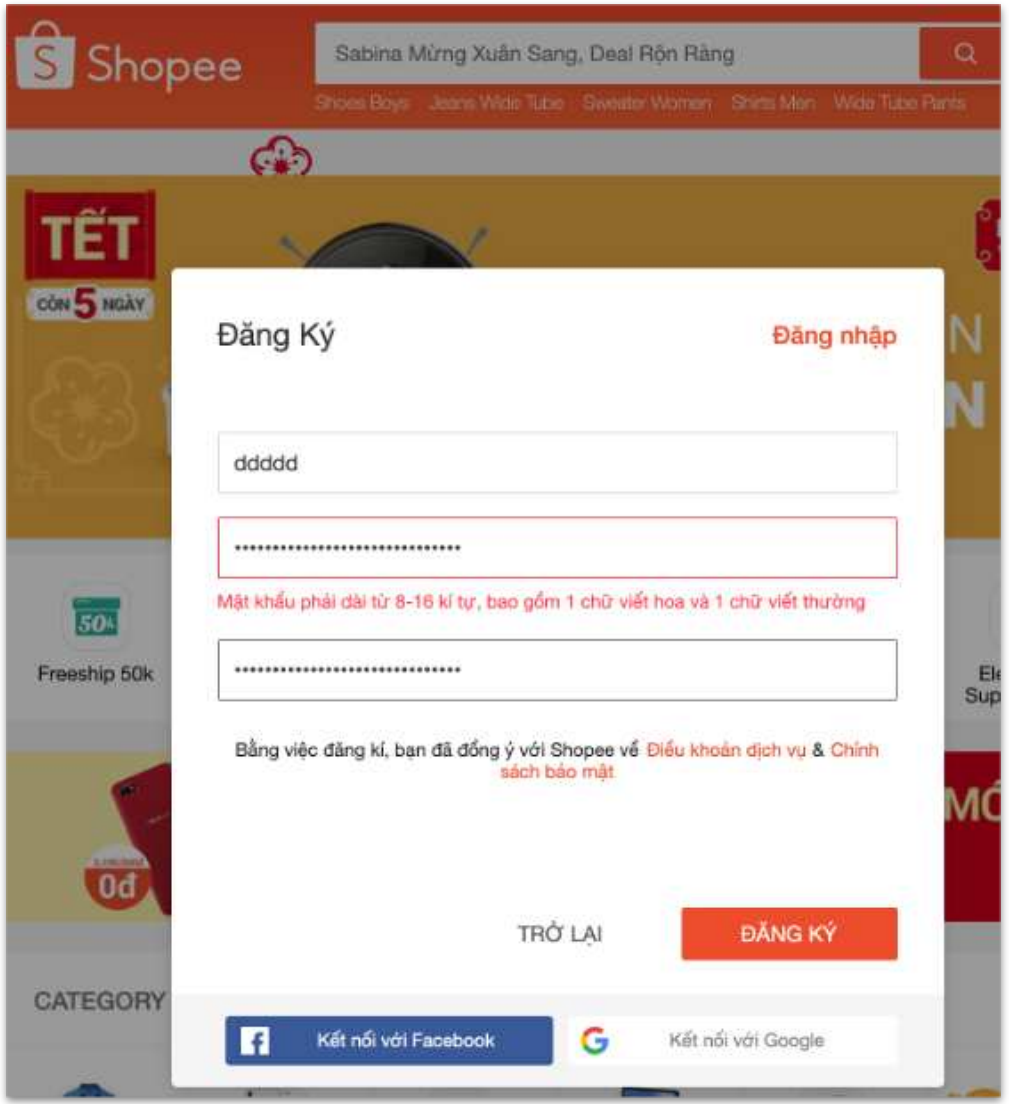
- Currently a free agent

Dennis Stötzel

How does Việt Nam authenticate?

How does Việt Nam authenticate?

Shopee


Shopee
Sabina Mừng Xuân Sang, Deal Rộn Ràng
Shoes Boys
Jeans Wide Tube
Sweater Women
Shirts Men
Wide Tube Pants
TẾT
CÒN 5 NGÀY
Freeship 50k
CATEGORY
Đăng Ký
Đăng nhập
ddddd
Mật khẩu phải dài từ 8-16 kí tự, bao gồm 1 chữ viết hoa và 1 chữ viết thường
Bằng việc đăng kí, bạn đã đồng ý với Shopee về Điều khoản dịch vụ & Chính sách bảo mật
TRỞ LẠI
ĐĂNG KÝ
Kết nối với Facebook
Kết nối với Google

Shopee

Maximum password length


```
HTTP/1.1 200 OK
Date: Sat, 04 Jan 2020 14:27:29 GMT
Content-Type: application/json; charset=utf-8
Connection: close
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://banhang.shopee.vn
Set-Cookie: SPC_SI=29k9d4lnckp2861waqib27lpbqocg8ha; Domain=.shopee.vn;
expires=Sun, 05-Jan-2020 14:27:29 GMT; httponly; Max-Age=86400; Path=/; secure
Content-Length: 30

{"data":...}
```

Uses cookies for session handling


```
HTTP/1.1 200 OK
Date: Sat, 04 Jan 2020 14:27:29 GMT
Content-Type: application/json; charset=utf-8
Connection: close
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://banhang.shopee.vn
Set-Cookie: SPC_SI=29k9d4lnckp2861waqib27lpbqocg8ha; Domain=.shopee.vn;
expires=Sun, 05-Jan-2020 14:27:29 GMT; httponly; Max-Age=86400; Path=/; secure
Content-Length: 30

{"data":...}
```

Domain cookies for fake SSO


```
GET /api/v2/location_filter/get_all HTTP/1.1
Host: shopee.vn
cept-Encoding: gzip, deflate
Connection: close
Cookie: SPC_SI=29k9d4lnckp2861waqib27lpbqocg8ha; SPC_U=216487310;
```

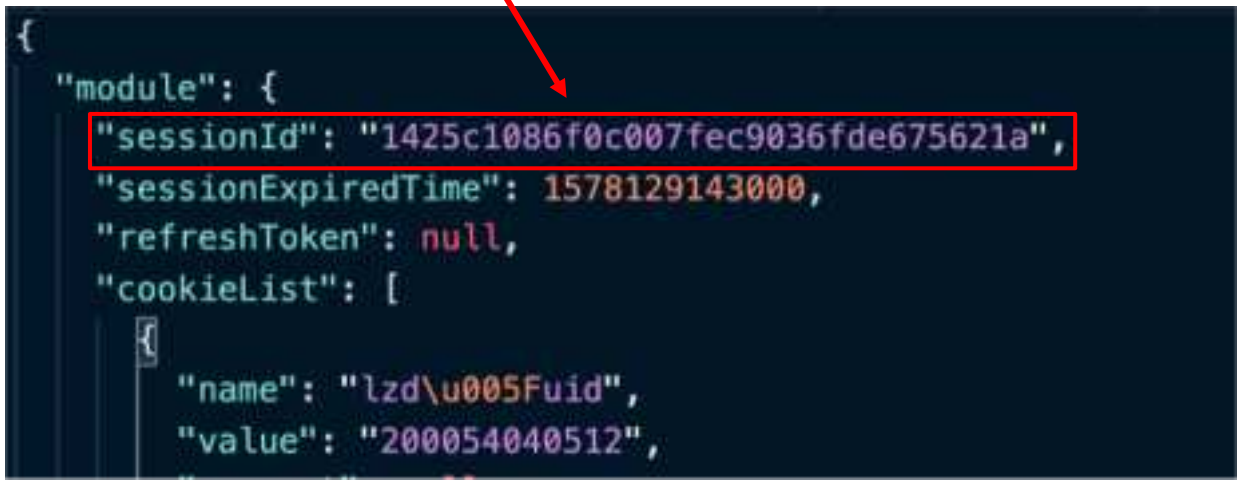
```
HTTP/1.1 200 OK
Date: Sat, 04 Jan 2020 14:27:29 GMT
Content-Type: application/json; charset=utf-8
Connection: close
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://banhang.shopee.vn
Set-Cookie: SPC_SI=29k9d4lnckp2861waqib27lpbqocg8ha; Domain=.shopee.vn;
expires=Sun, 05-Jan-2020 14:27:29 GMT; httponly; Max-Age=86400; Path=/; secure
Content-Length: 30

{"data":...}
```

Session refresh through heart beat function with session cookie expires time

Lazada

```
HTTP/1.1 200
Date: Sat, 04 Jan 2020 08:12:23 GMT
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
Set-Cookie: lzd_uid=200054040512; Domain=.lazada.vn; Path=/
Set-Cookie: lzd_sid=1425c1086f0c007fec9036fde675621a; Domain=.lazada.vn; Path=/; HttpOnly
```

Session ID in cookies and in JSON object



Lazada

```
HTTP/1.1 200
Date: Sat, 04 Jan 2020 08:12:23 GMT
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
Set-Cookie: lzd_uid=200054040512; Domain=.lazada.vn; Path=/
Set-Cookie: lzd_sid=1425c1086f0c007fec9036fde675621a; Domain=.lazada.vn; Path=/; HttpOnly
```

Domain cookies for fake SSO

Zalo


```
POST /account/authen?a=pw HTTP/1.1
Host: id.zalo.me
Origin: https://id.zalo.me
Content-Length: 684

phone_num=76123456&iso_country_code=VN&password=foofoo&g-recaptcha-response=123&captcha-
version=3&continue=https%3A%2F%2Fchat.zalo.me%2F
```

```
HTTP/1.1 200 OK
Date: Sat, 04 Jan 2020 08:30:57 GMT
Content-Type: application/json;charset=utf-8
Content-Length: 62
Set-Cookie: zpsid=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsidleg=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zsid=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsid=jvoQ.283222962.1.auD3Ti-MajnzDxxxmvQItBhbuExtfQ3k_Q-axhlqG9PYgdv6pjRVp96Majm; Path=/;
Max-Age=31536000; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsidleg=jvoQ.283222962.1.auD3Ti-MajnzDxxxmvQItBhbuExtfQ3k_Q-axhlqG9PYgdv6pjRVp96Majm;
Path=/; Max-Age=31536000; Domain=zalo.me; HttpOnly; Secure;
Strict-Transport-Security: max-age=86400; includeSubDomains

{"error_message":"ThÃ nh cÃ´ng","data":null,"error_code":1013}
```

Zalo

All the session cookies …

```
HTTP/1.1 200 OK
Date: Sat, 04 Jan 2020 08:30:57 GMT
Content-Type: application/json;charset=utf-8
Content-Length: 62
Set-Cookie: zpsid=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsidleg=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zsid=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsid=jvoQ.283222962.1.auD3Ti-MajnzDxxxmvQItBhbuExtfQ3k_Q-axhlqG9PYgdv6pjRVp96Majm; Path=/;
Max-Age=31536000; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsidleg=jvoQ.283222962.1.auD3Ti-MajnzDxxxmvQItBhbuExtfQ3k_Q-axhlqG9PYgdv6pjRVp96Majm;
Path=/; Max-Age=31536000; Domain=zalo.me; HttpOnly; Secure;
Strict-Transport-Security: max-age=86400; includeSubDomains

{"error_message":"ThÃ nh cÃ´ng","data":null,"error_code":1013}
```

Zalo

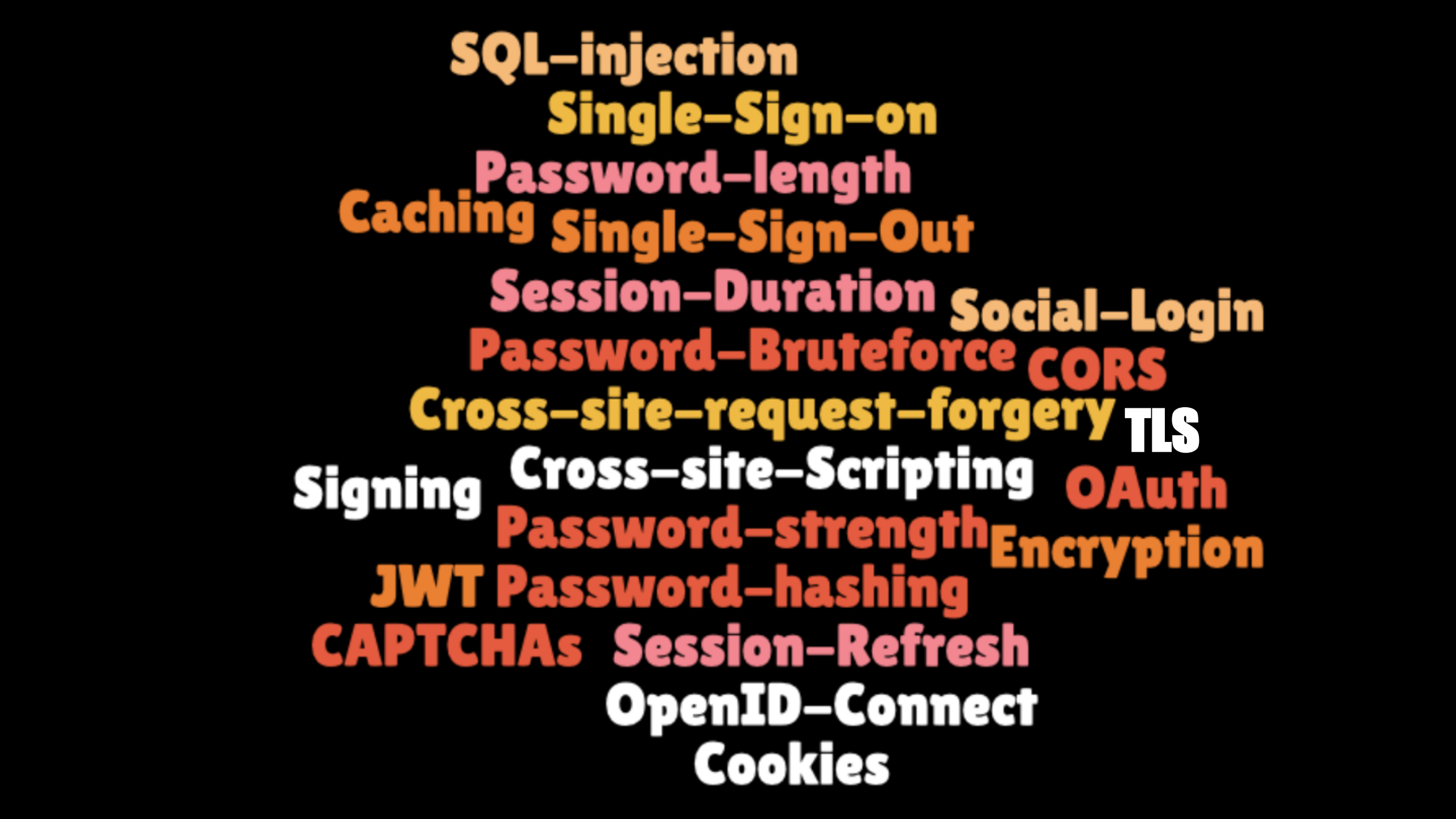
Domain cookies for fake SSO

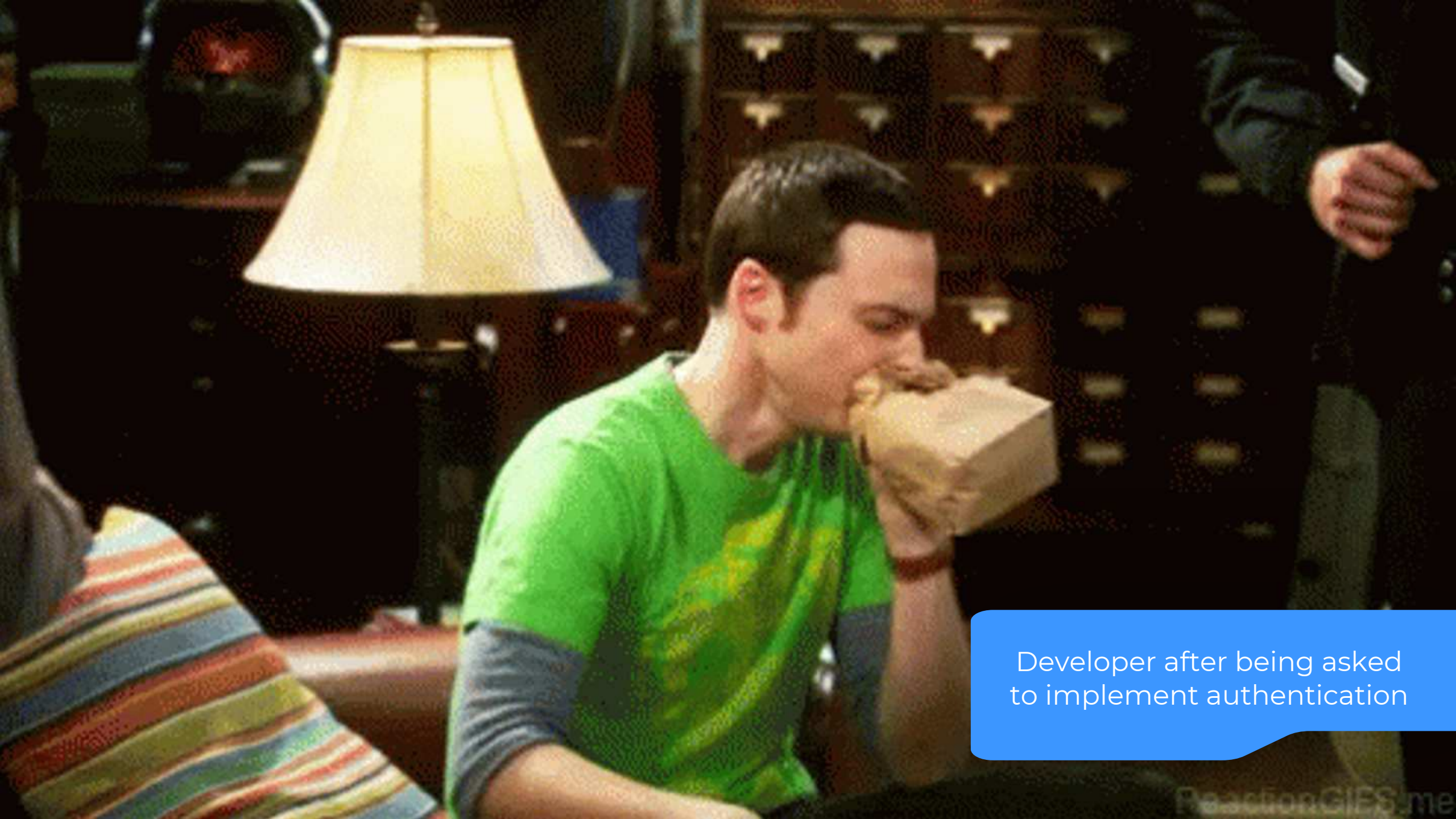
```
HTTP/1.1 200 OK
Date: Sat, 04 Jan 2020 08:30:57 GMT
Content-Type: application/json;charset=utf-8
Content-Length: 62
Set-Cookie: zpsid=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsidleg=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zsid=EXPIRED; Path=/; Max-Age=0; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsid=jvoQ.283222962.1.auD3Ti-MajnzDxxxmvQItBhbuExtfQ3k_Q-axhlqG9PYgdv6pjRVp96Majm; Path=/;
Max-Age=31536000; Domain=zalo.me; HttpOnly; Secure; SameSite=None
Set-Cookie: zpsidleg=jvoQ.283222962.1.auD3Ti-MajnzDxxxmvQItBhbuExtfQ3k_Q-axhlqG9PYgdv6pjRVp96Majm;
Path=/; Max-Age=31536000; Domain=zalo.me; HttpOnly; Secure;
Strict-Transport-Security: max-age=86400; includeSubDomains

{"error_message":"ThÃ nh cÃ´ng","data":null,"error_code":1013}
```

Zalo

444	https://stc-sp.zadn.vn	GET	/session-sdk/session-sdk.min.js
443	https://accounts.chat.zalo.me	GET	/account/pushsession?token=s8ElIPyi5HMHoA4TobzHKSQglHt0LLLHsgAbJfiWI6J6...
442	https://accounts.zalo.me	GET	/account/syncsession?token=s8ElIPyi5HMHoA4TobzHKSQglHt0LLLHsgAbJfiWI6J6...
441	https://accounts.zaloapp.com	GET	/account/syncsession?token=s8ElIPyi5HMHoA4TobzHKSQglHt0LLLHsgAbJfiWI6J6...
440	https://accounts.baomoi.com	GET	/account/syncsession?token=s8ElIPyi5HMHoA4TobzHKSQglHt0LLLHsgAbJfiWI6J6...
439	https://accounts.news.zing.vn	GET	/account/syncsession?token=s8ElIPyi5HMHoA4TobzHKSQglHt0LLLHsgAbJfiWI6J6...
438	https://id.zalo.me	POST	/account/verifygcaptchav3token
437	https://accounts.zingtv.vn	GET	/account/syncsession?token=s8ElIPyi5HMHoA4TobzHKSQglHt0LLLHsgAbJfiWI6J6...
436	https://accounts.zingmp3.vn	GET	/account/syncsession?token=s8ElIPyi5HMHoA4TobzHKSQglHt0LLLHsgAbJfiWI6J6...
435	https://www.google.com	POST	/recaptcha/api2/reload?k=6Lec6KEUAAAAAM6uytRwkKAzSNMvv7AhMWthwsmf
434	https://id.zalo.me	GET	/account/checksession?continue=https%3A%2F%2Fchat%2Ezalo%2Eme%2F&login_...
433	https://id.zalo.me	POST	/account/authen?a=pw

Automatic login in all other applications belonging to the company


SQL-injection
Single-Sign-on
Password-length
Caching
Single-Sign-Out
Session-Duration
Social-Login
Password-Bruteforce
CORS
Cross-site-request-forgery
TLS
Signing
Cross-site-Scripting
OAuth
Password-strength
Encryption
JWT
Password-hashing
CAPTCHAs
Session-Refresh
OpenID-Connect
Cookies


Developer after being asked
to implement authentication

Authentication is hard.

Leave it to professionals.

Authentication Professionals

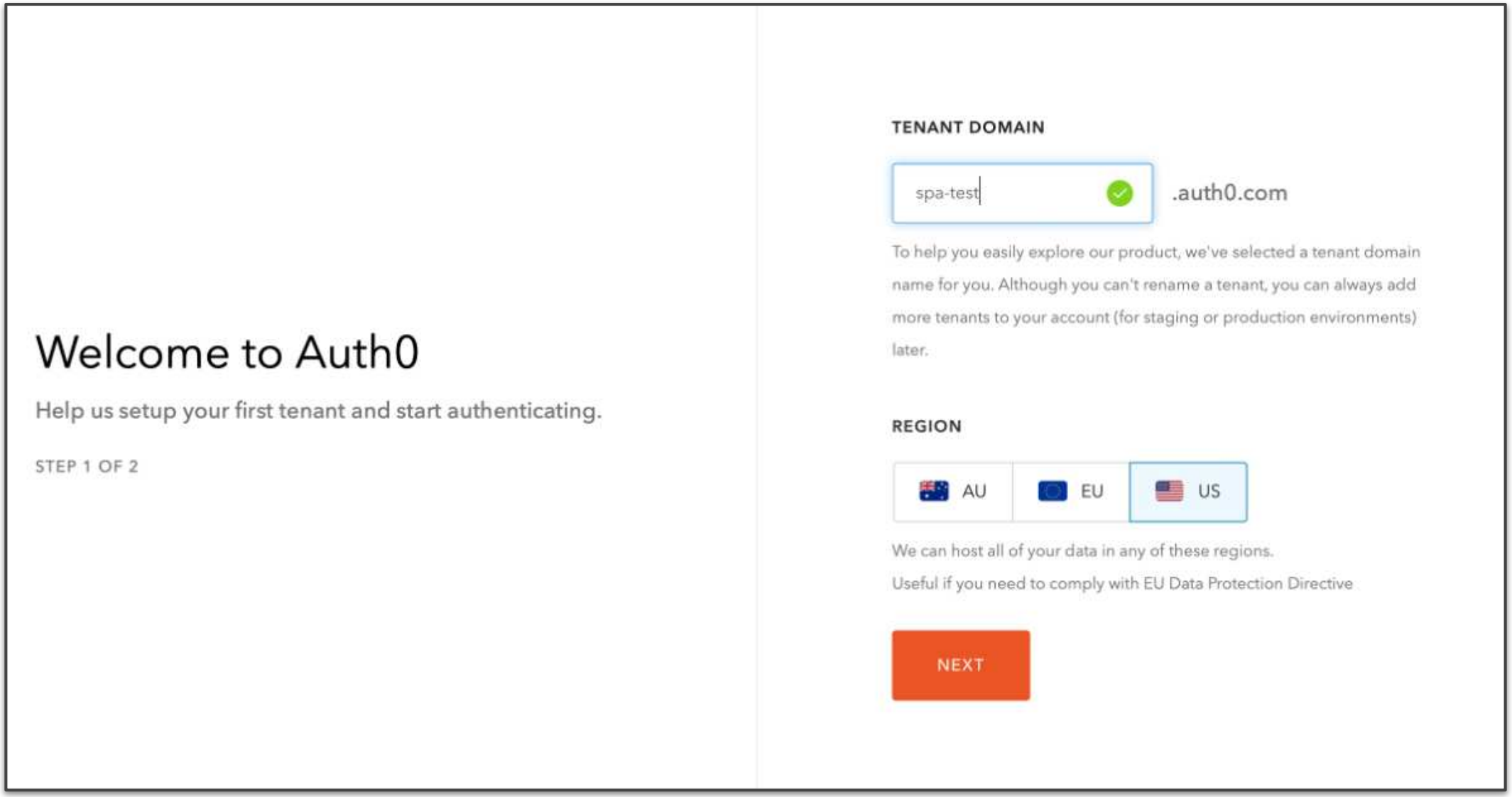
Auth0

{okta}

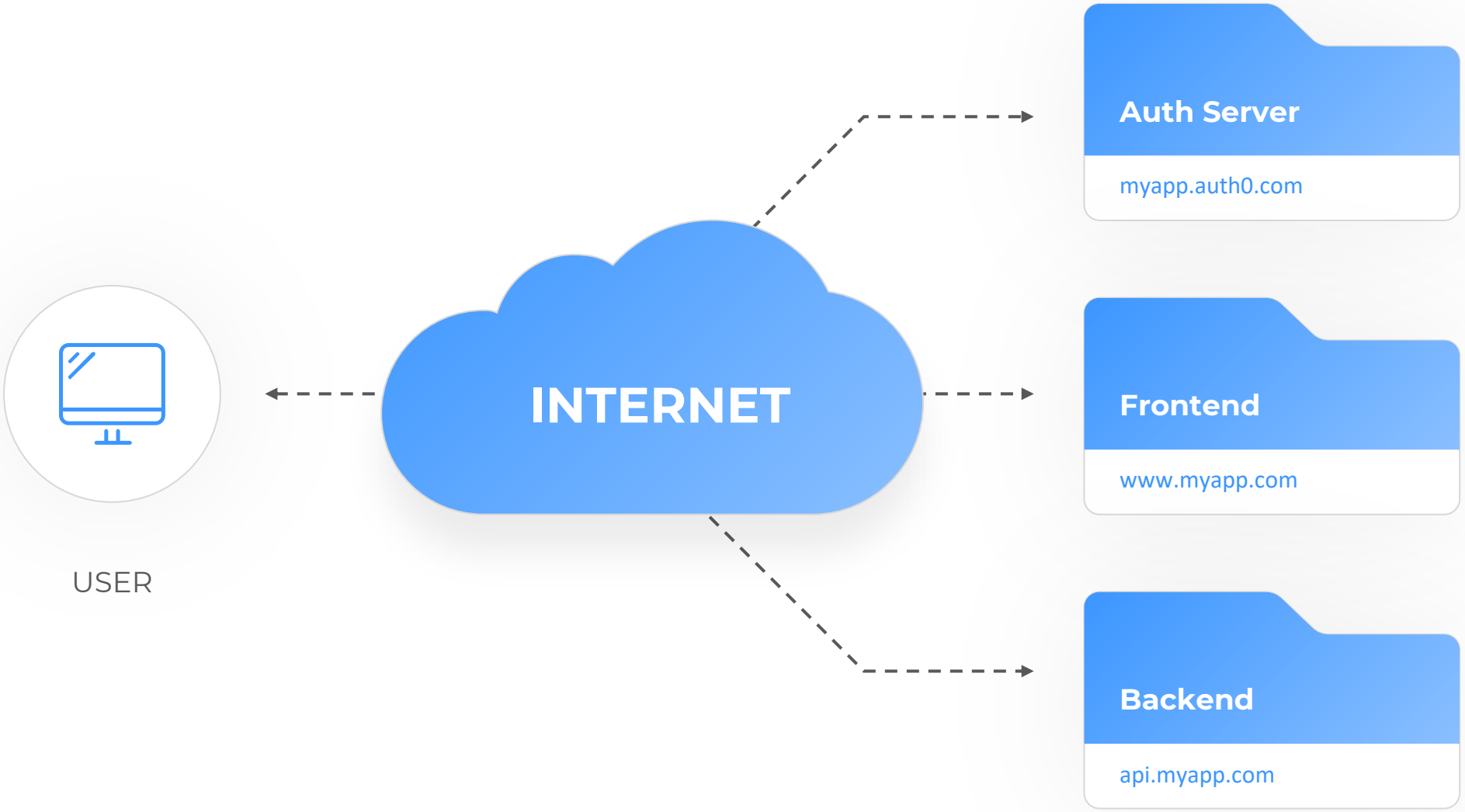
Amazon Cognito

Authentication Professionals

Authentication Professionals



Basic Setup Single Page Applications



OAuth 2.0 and OpenID Connect 1.0

- Published 2012
- Defines an **authorization** protocol
- Uses bearer tokens
- Defines most of the flows we still use today

- Published 2014
- Defines an **authentication** layer on top of OAuth
- Defines the use of ID tokens
- Has several additional specifications for Session Management, logout, etc.

Trust the Code!

https://github.com/dennisMeeQ/auth0-react-express

https://spa-auth0.herokuapp.com

Login

Token Storage

API Protection

Session Refresh

Logout

Login

Token Storage

API Protection

Session Refresh

Logout

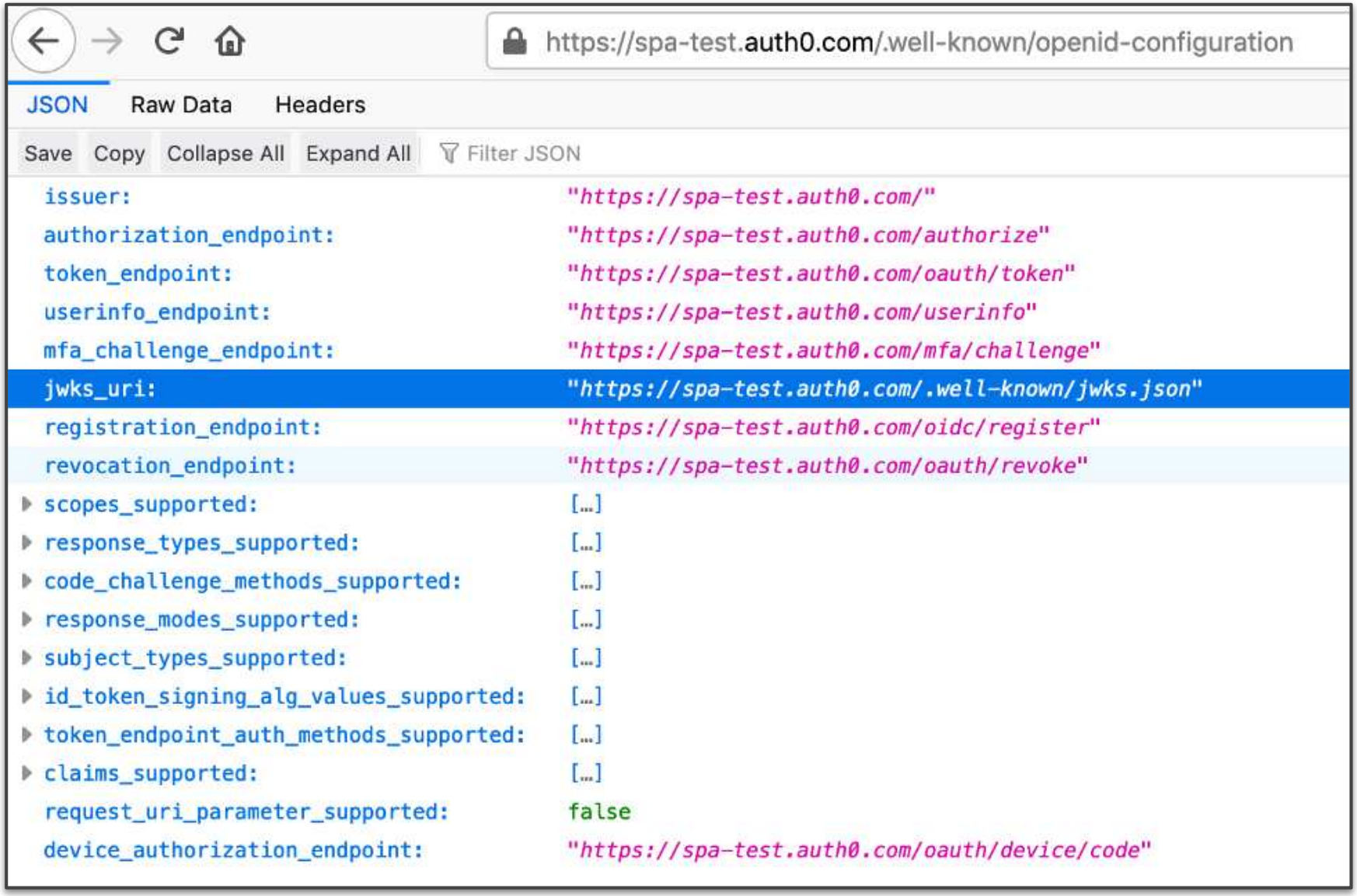
Protected API endpoints

```
const express = require('express');
const { authenticated } = require('./security');
const { species, people } = require('./api-data');

const app = express();

app.get('/people', authenticated, (req, res) => {
 res.send({ people });
});

app.get('/species', (req, res) => {
 res.send({ species });
});

app.use((err, req, res, next) => {
 if (err.name === 'UnauthorizedError') {
   res.status(401).send({ msg: 'Invalid token' });
 } else next();
});
```

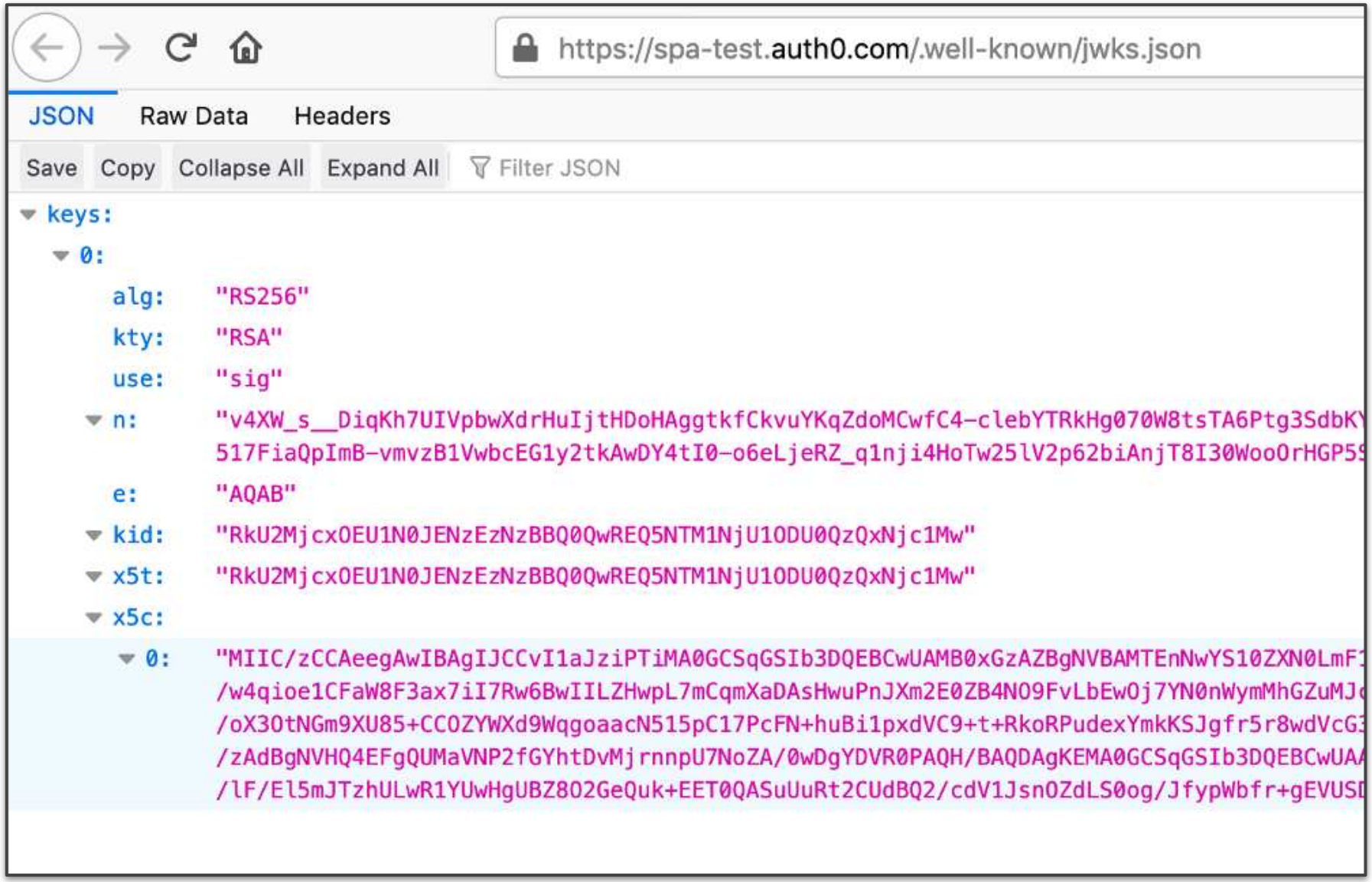
Authenticated middleware

```
const jwt = require('express-jwt');
const jwksRsa = require('jwks-rsa');

// Auth0 configuration
const authConfig = {
 issuer: 'https://spa-test.auth0.com/',
 audience: 'https://api.allthingsauth.dev',
 algorithm: ['RS256'],
};

const secret = jwksRsa.expressJwtSecret({
 cache: true,
 rateLimit: true,
 jwksRequestsPerMinute: 5,
 jwksUri: `${authConfig.issuer}.well-known/jwks.json`,
});

export const authenticated = jwt({ secret, ...authConfig });
```

OpenID Connect Configuration



OpenID Connect JWKS Endpoint



Login

Token Storage

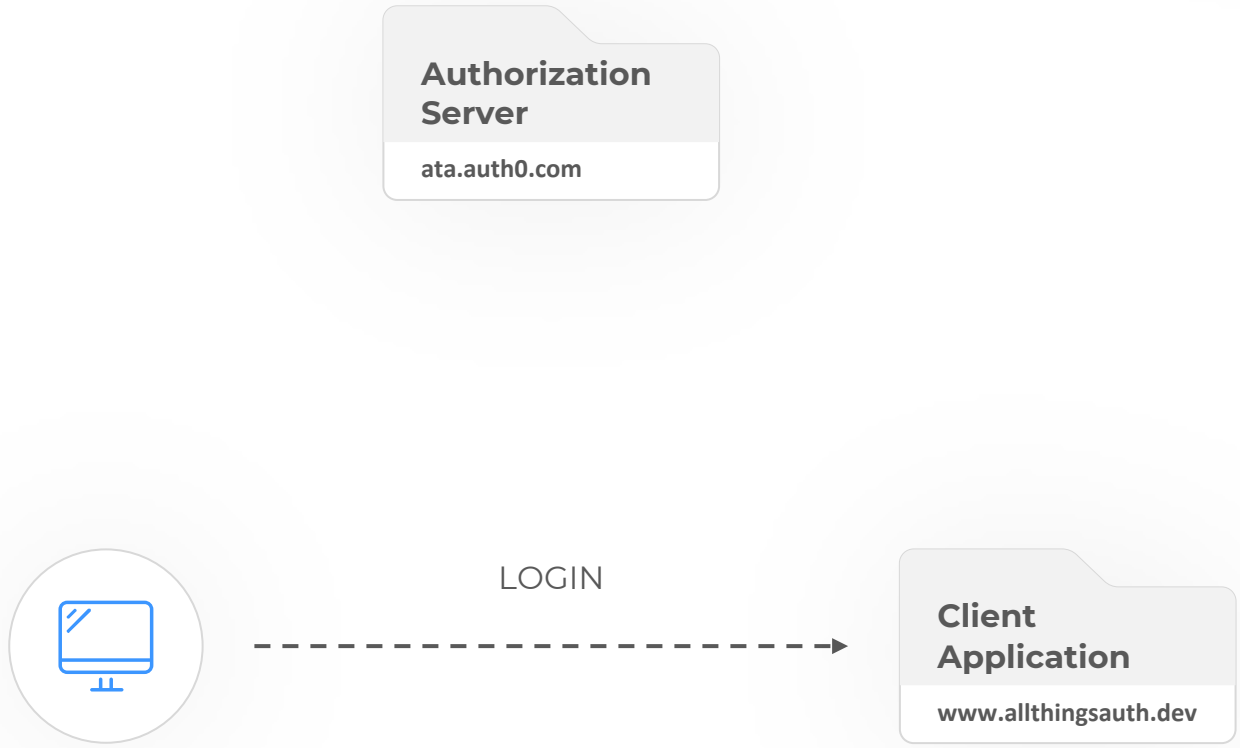
API Protection

Session Refresh

Logout

Use the Authorization Code flow with PKCE in single page applications.

01



The user clicks the login button.

02

RANDOM

CODE VERIFIER

CODE VERIFIER

HASH

CODE CHALLENGE

The client application randomly generates a code verifier.
It also generates a code challenge by hashing the code verifier and then stores both values.

Authorization Server

ata.auth0.com

USER

Client Application

www.allthingsauth.dev

CODE VERIFIER

CODE CHALL.

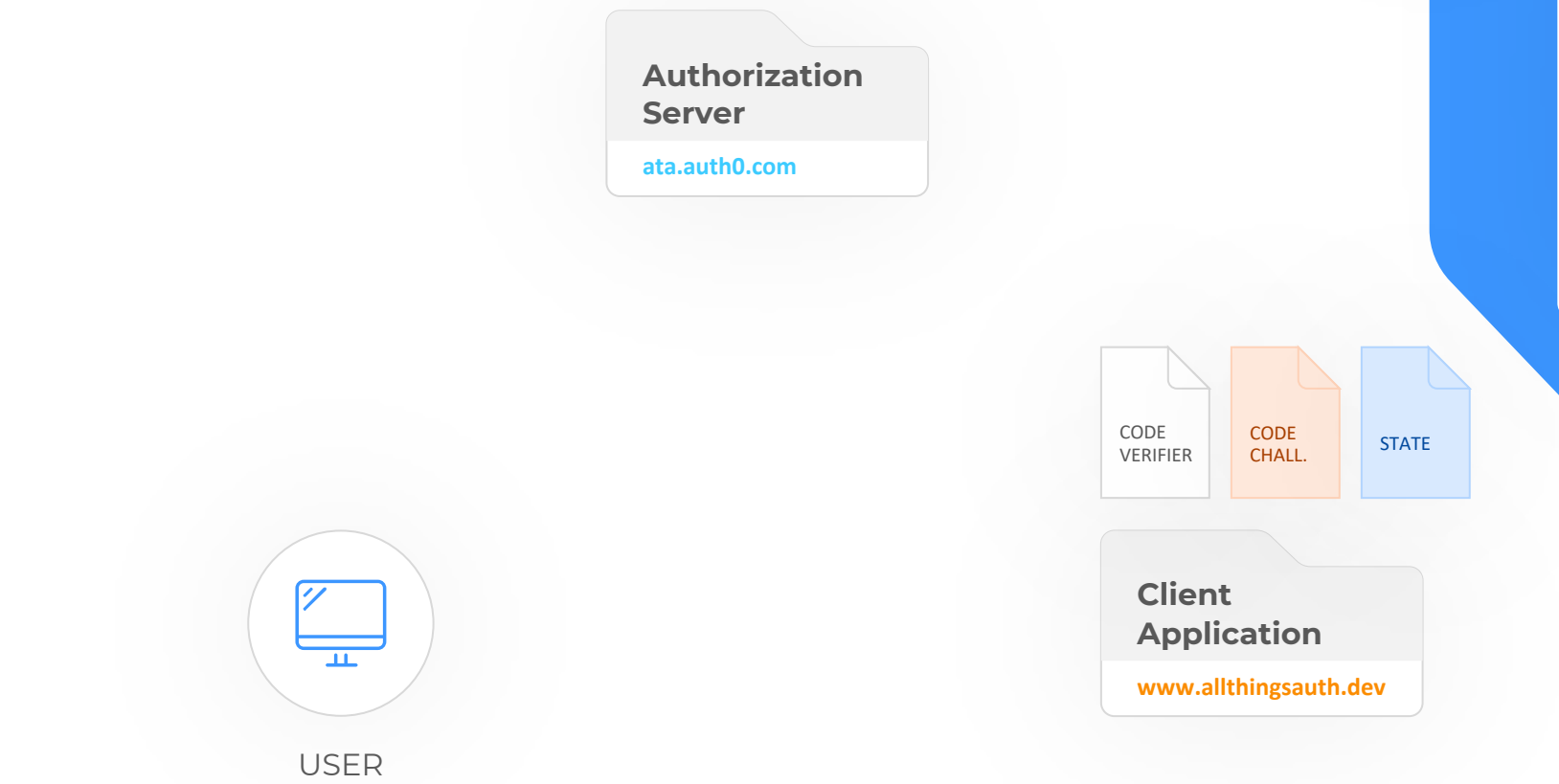
03

RANDOM

STATE

The client application randomly generates a state parameter.

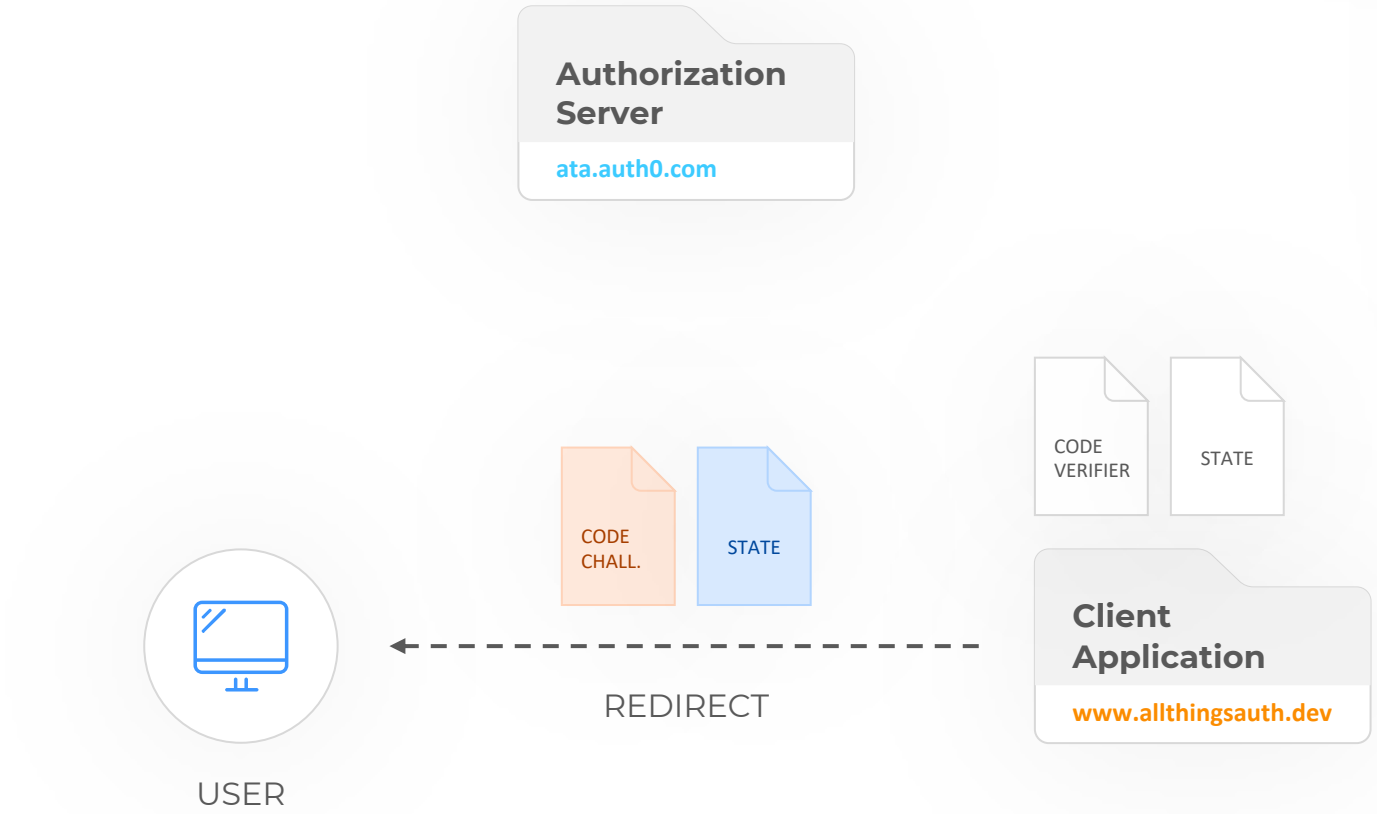
04




```
GET
/authorize?client_id=123abc&redirect_uri=https%3A
%2F%2Fwww.allthingsauth.dev%2Fcallback&scope=o
penid&response_type=code&response_mode=query
&state=987zyx&code_challenge=666rrr&code_challe
nge_method=S256 HTTP/1.1
Host: ata.auth0.com
referer: https://www.allthingsauth.dev
```

The client application redirects the browser to the authorization server with the code challenge and the state parameter.

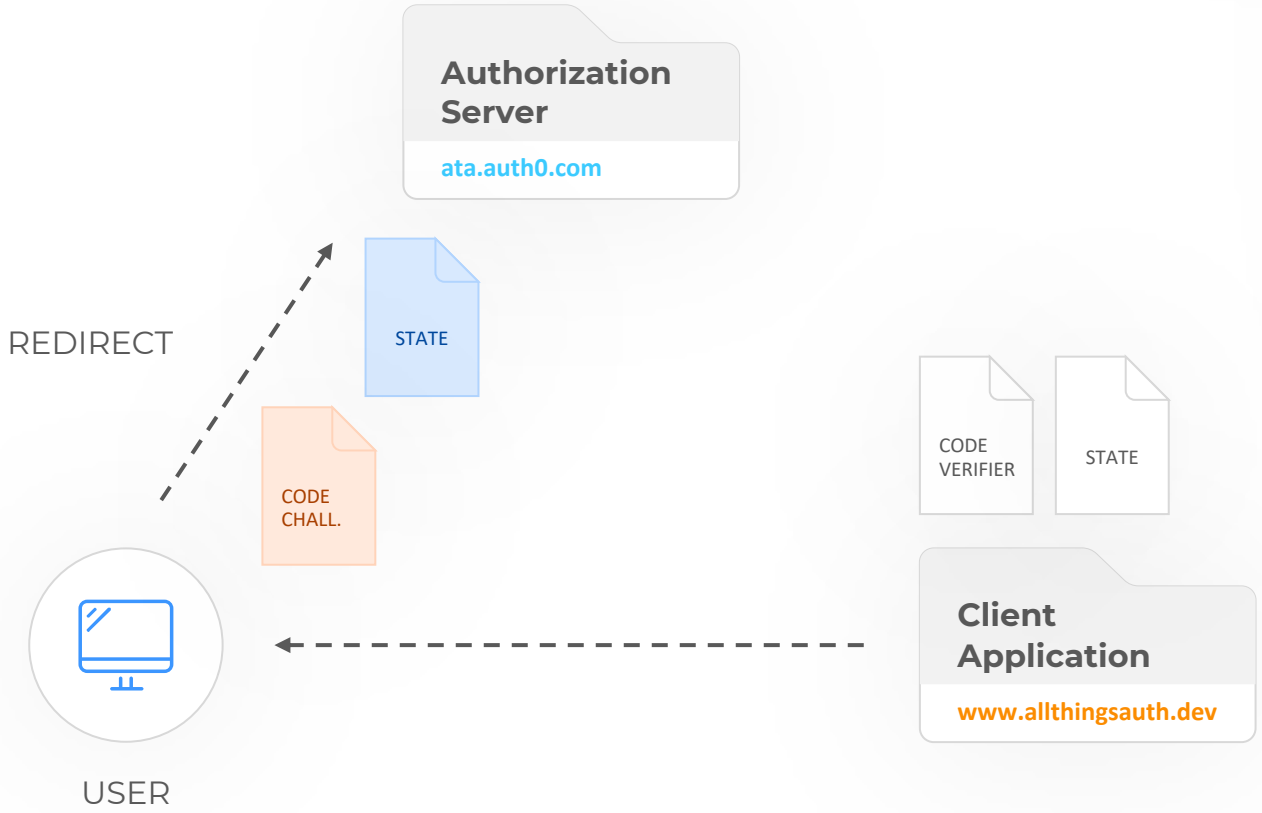
04




```
GET
/authorize?client_id=123abc&redirect_uri=https%3A
%2F%2Fwww.allthingsauth.dev%2Fcallback&scope=o
penid&response_type=code&response_mode=query
&state=987zyx&code_challenge=666rrr&code_challe
nge_method=S256 HTTP/1.1
Host: ata.auth0.com
referer: https://www.allthingsauth.dev
```

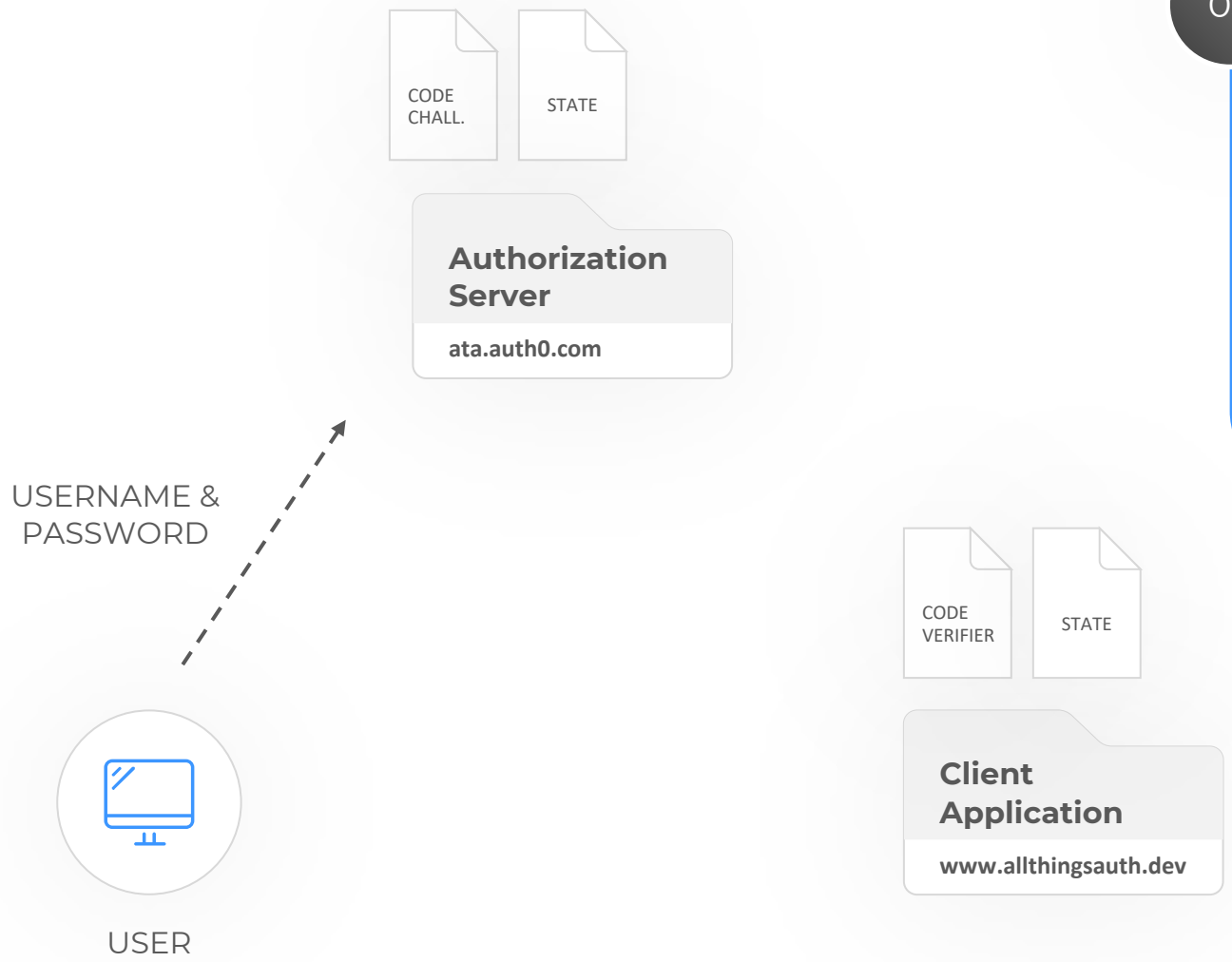
The client application redirects the browser to the authorization server with the code challenge and the state parameter.

04




```
GET
/authorize?client_id=123abc&redirect_uri=https%3A
%2F%2Fwww.allthingsauth.dev%2Fcallback&scope=o
penid&response_type=code&response_mode=query
&state=987zyx&code_challenge=666rrr&code_challe
nge_method=S256 HTTP/1.1
Host: ata.auth0.com
referer: https://www.allthingsauth.dev
```

The client application redirects the browser to the authorization server with the code challenge and the state parameter.

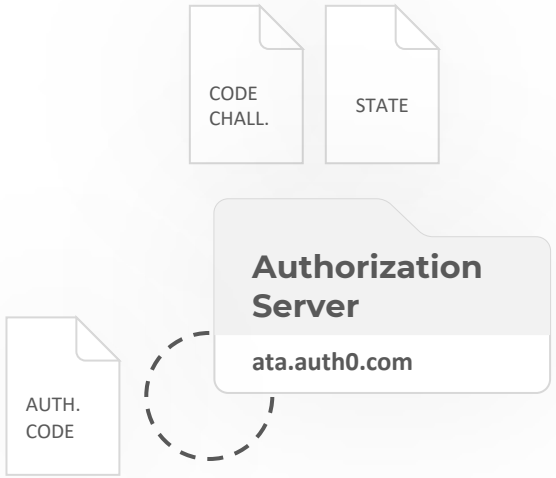


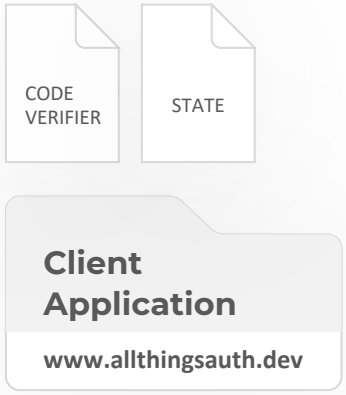
Username: Paco

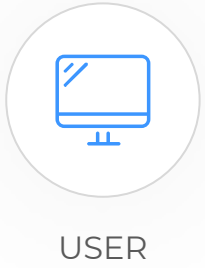
Password: ●●●●●●●●●●●●

LOGIN

The user enters their username and password in the login form.

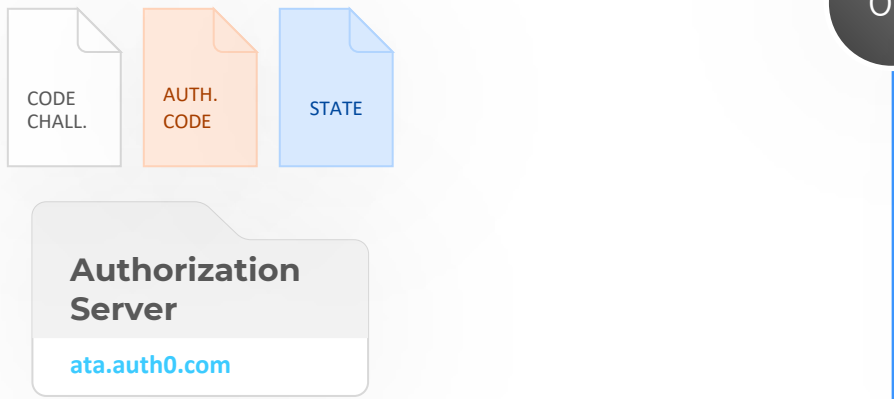






The authorization server verifies username and password.

On success, it generates an authorization code and stores it together with the code challenge and the state.

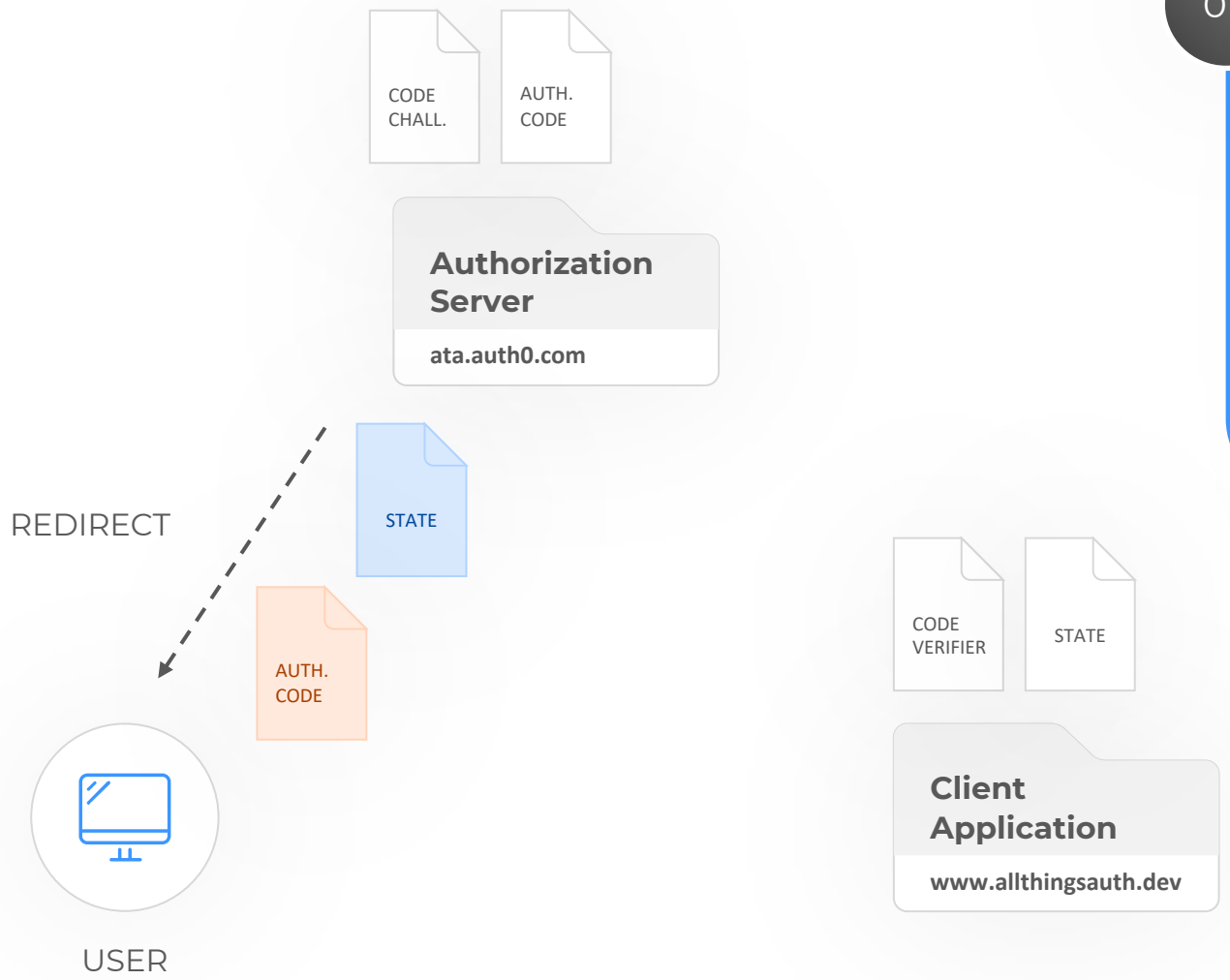


07

```
GET /callback?code=oiu357&state=987zyx
HTTP/1.1
Host: www.allthingsauth.dev
Referer: https://ata.auth0.com
```

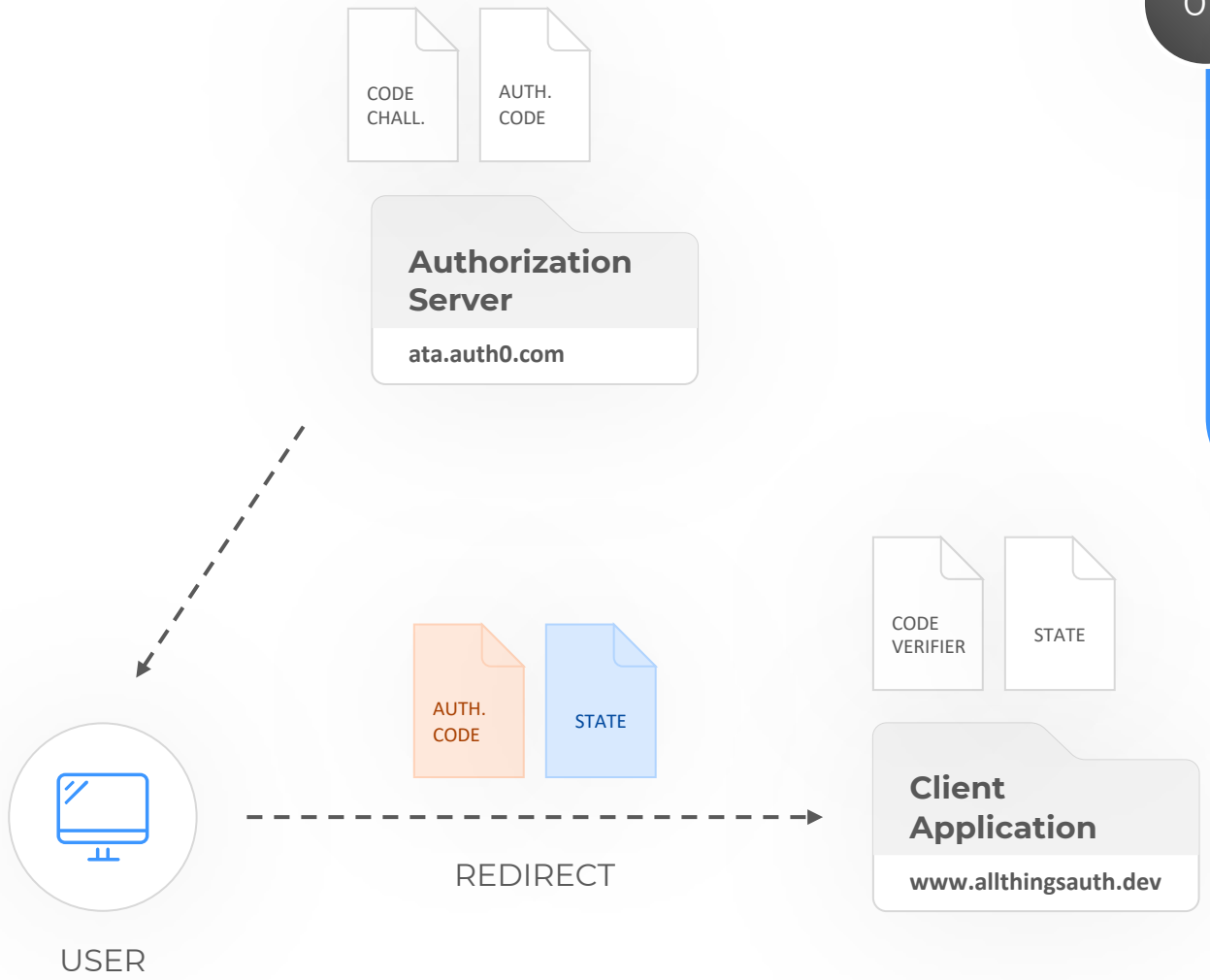
USER

The authorization server redirects the browser to the client application with the authorization code and the state parameter.



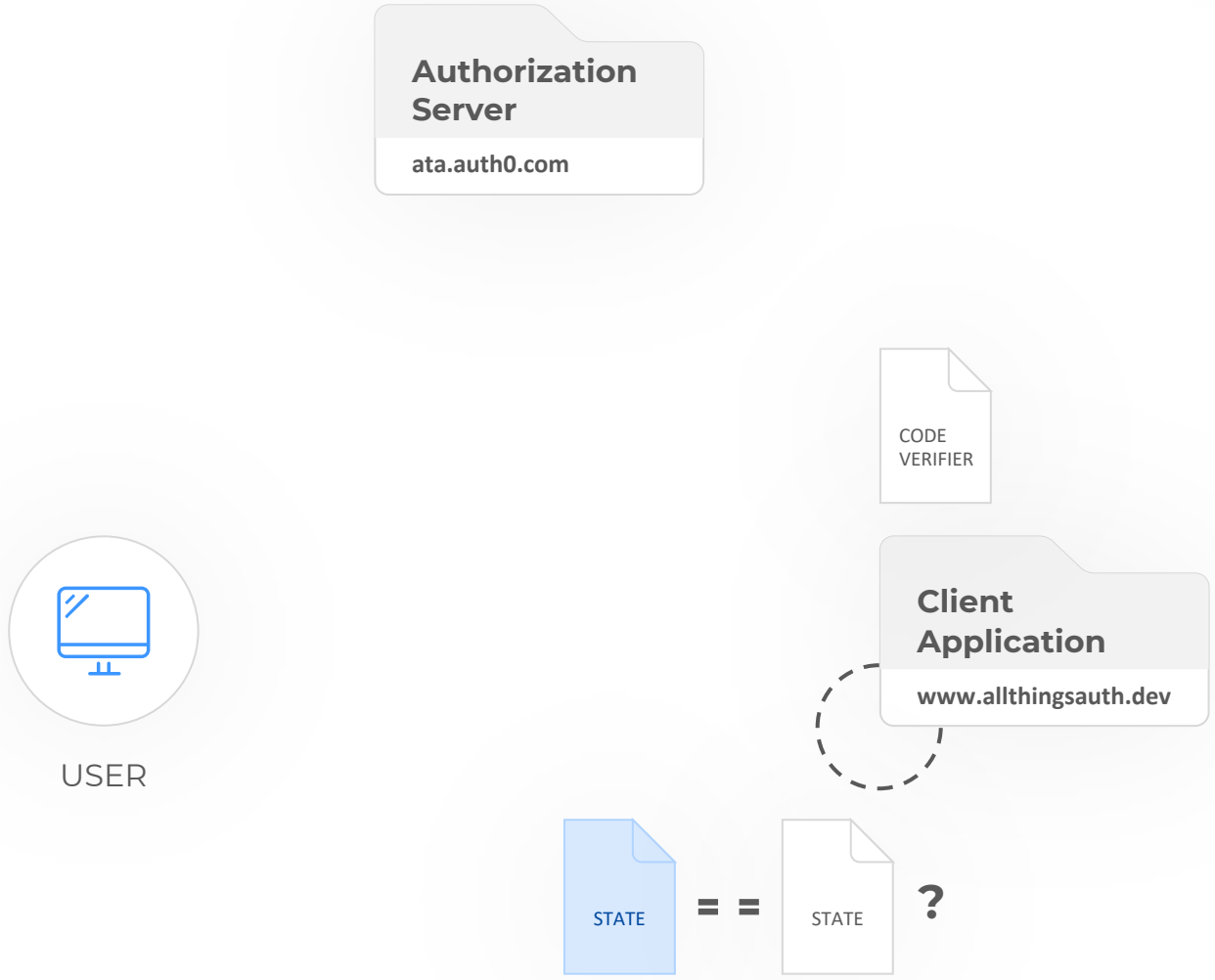

```
GET /callback?code=oiu357&state=987zyx
HTTP/1.1
Host: www.allthingsauth.dev
Referer: https://ata.auth0.com
```

The authorization server redirects the browser to the client application with the authorization code and the state parameter.

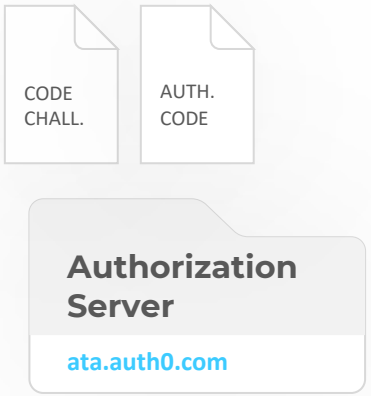


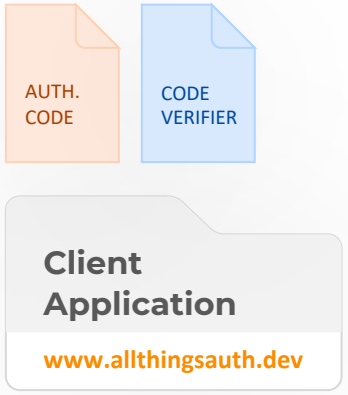

```
GET /callback?code=oiu357&state=987zyx
HTTP/1.1
Host: www.allthingsauth.dev
Referer: https://ata.auth0.com
```

The authorization server redirects the browser to the client application with the authorization code and the state parameter.



The client application compares the stored state value with the one returned by the authorization server.

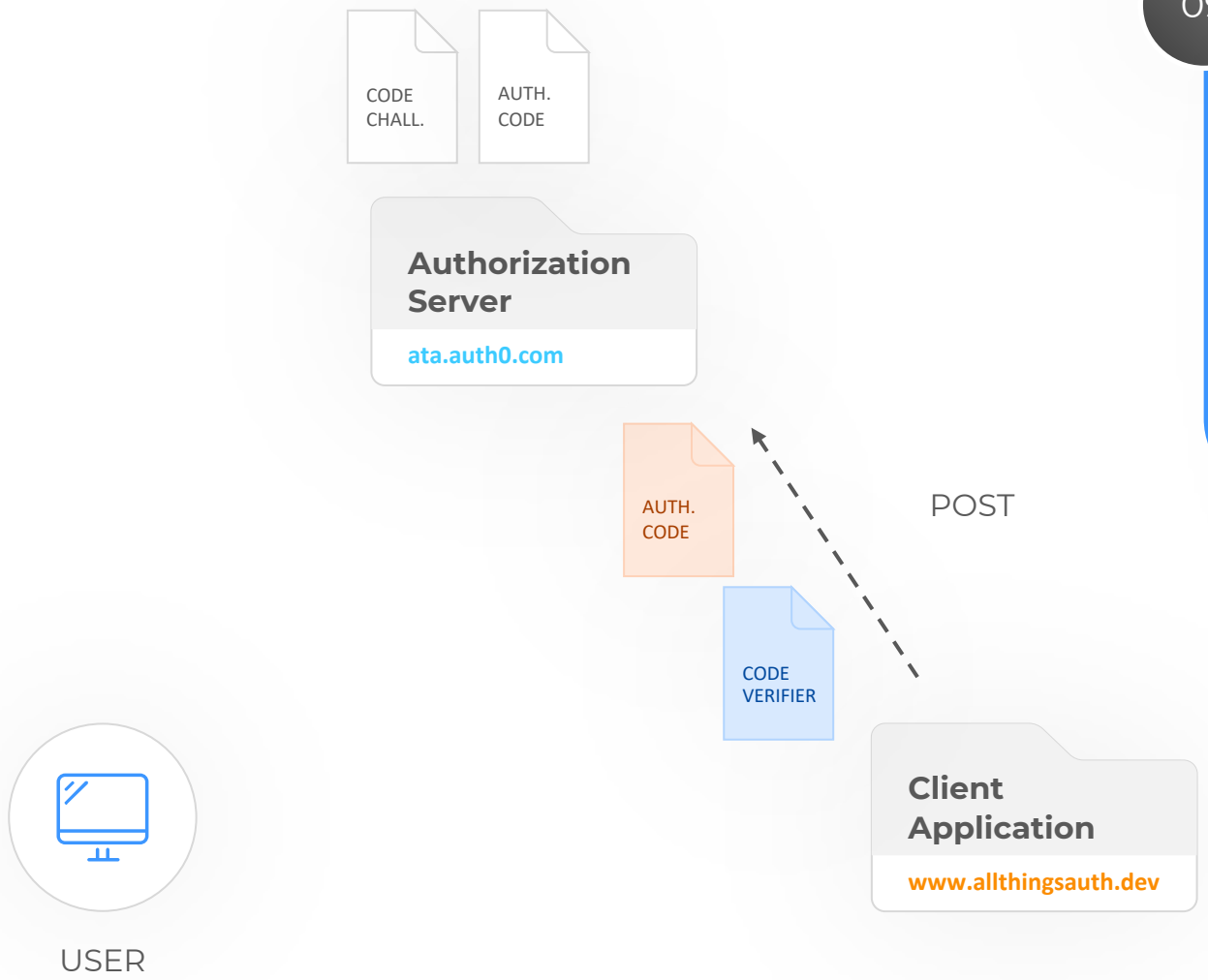




USER

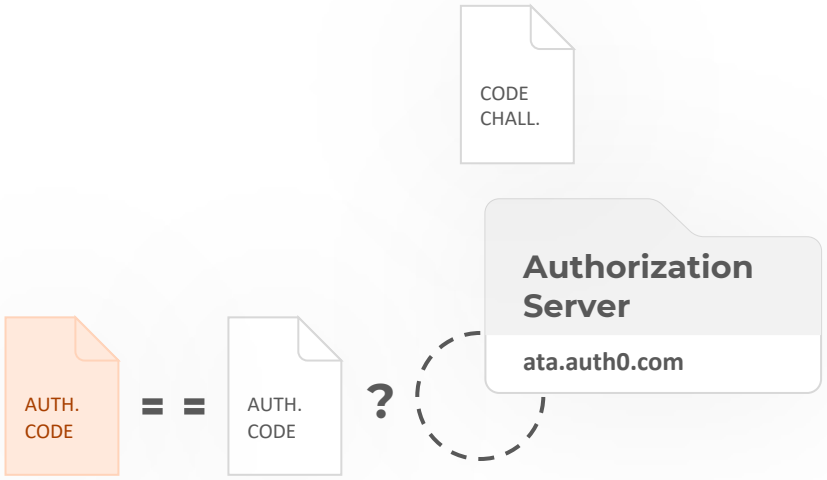
```
POST /oauth/token HTTP/1.1
Host: ata.auth0.com
Origin: https://www.allthingsauth.dev
Content-type: application/json
Content-Length: 153

{"grant_type":"authorization_code","redirect_uri":"h
ttps://www.allthingsauth.dev/callback","client_id":"1
23abc","code_verifier":"hij777","code":"oiu357"}
```

Through an XHR POST request, the client application sends the authorization code and the code verifier to the authentication server.




```
POST /oauth/token HTTP/1.1
Host: ata.auth0.com
Origin: https://www.allthingsauth.dev
Content-type: application/json
Content-Length: 153

{"grant_type":"authorization_code","redirect_uri":"h
ttps://www.allthingsauth.dev/callback","client_id":"1
23abc","code_verifier":"hij777","code":"oiu357"}
```

Through an XHR POST request, the client application sends the authorization code and the code verifier to the authentication server.



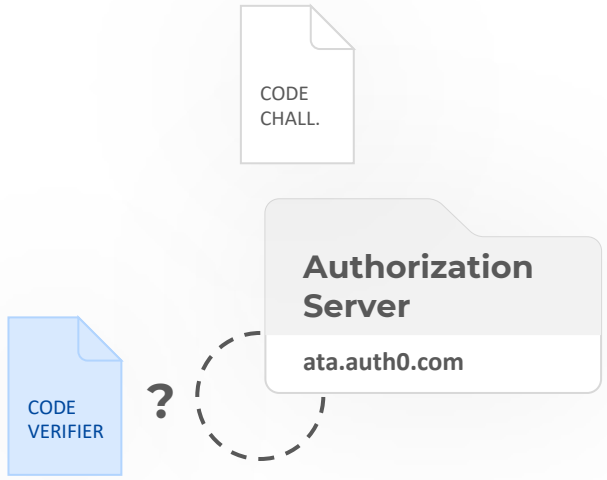
USER

Client Application

www.allthingsauth.dev

The authentication server verifies both the authorization code and the code verifier.

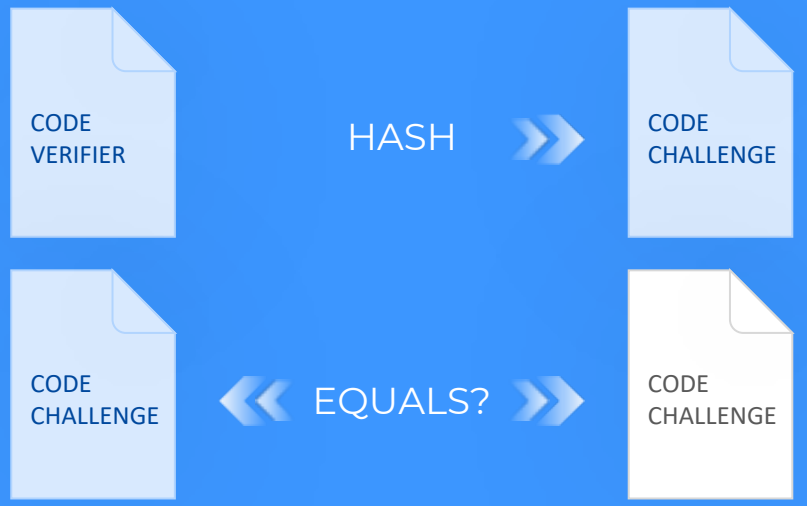
The latter is verified by hashing it and then comparing it with the stored code challenge.



USER

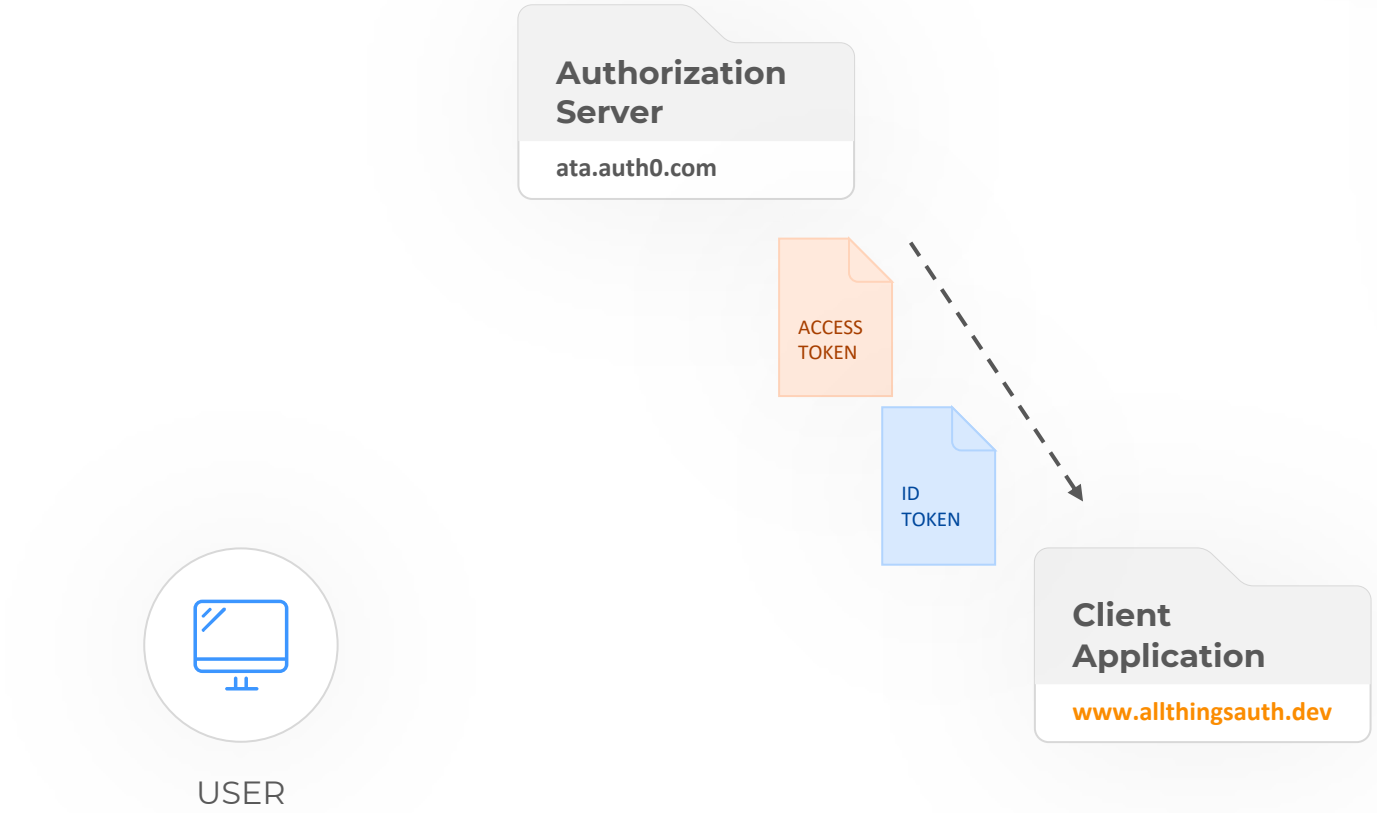
Client Application

www.allthingsauth.dev



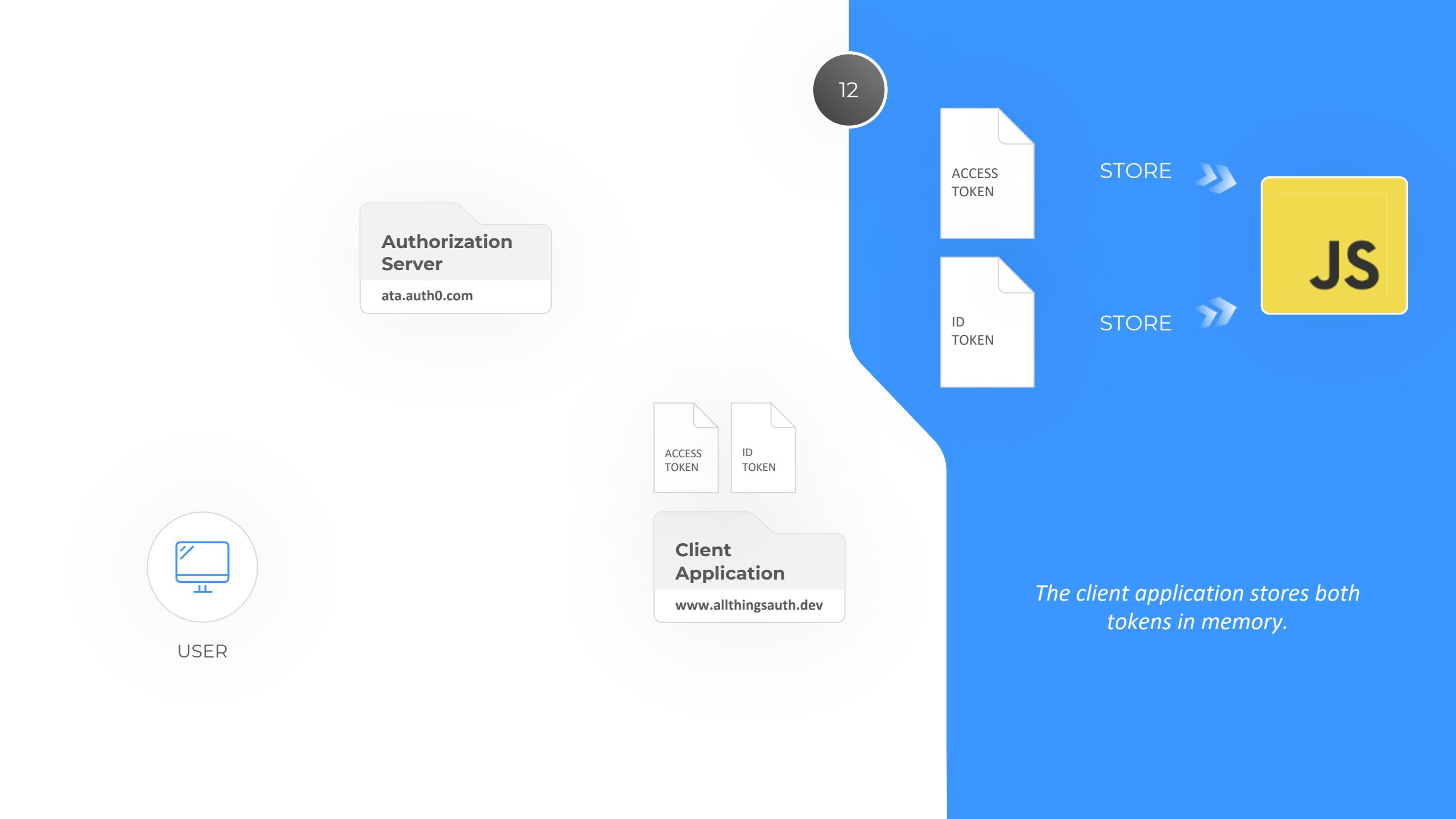
The authentication server verifies both the authorization code and the code verifier.

The latter is verified by hashing it and then comparing it with the stored code challenge.




```
HTTP/1.1 200 OK
Access-control-allow-origin:
   https://www.allthingsauth.dev
Content-type: application/json
Cache-control: no-store
Pragma: no-cache
Content-length: 106

{"access_token":"nmo951","scope":"openid","id_token
":"eyJ...6LA","expires_in":86400,"token_type":"Bearer"}
```

On success, the authentication server responds with the access token and the ID token.



The client application stores both tokens in memory.

Auth0, React Context and Hooks

```
import React, { useState, useEffect, useContext } from 'react';
import createAuth0Client from '@auth0/auth0-spa-js';

export const Auth0Context = React.createContext();
export const useAuth0 = () => useContext(Auth0Context);
export const Auth0Provider = ({ children, onRedirectCallback,
       redirect_uri, ...initOptions }) => {
// ...
 return (
   <Auth0Context.Provider
     value={{ isAuthenticated, user, loading, // ...
       loginWithRedirect: (...p) => auth0Client.loginWithRedirect(...p),
       getTokenSilently: (...p) => auth0Client.getTokenSilently(...p),
       logout: (...p) => auth0Client.logout(...p),
     }}>
     {children}
   </Auth0Context.Provider>
 );
};
```

Wrap it around the App

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Auth0Provider } from './auth/react-auth0-spa';
import App from './App';

ReactDOM.render(
 <Auth0Provider
    domain={"spa-test.auth0.com"}
    client_id={"melxSZVFgL9ONwTaggwozHOsjeTVvJOF"}
    audience={"https://api.allthingsauth.dev"}
    redirect_uri={window.location.origin}
 >
    <App />
 </Auth0Provider>,
 document.getElementById('root')
);
```

Basic Auth0 Hook Usage

```
import React from 'react';
import { useAuth0 } from '../auth/react-auth0-spa';

export const NavBar = () => {
 const { isAuthenticated, loginWithRedirect, logoutWithRedirect } = useAuth0();

 return (
   <nav>
       {!isAuthenticated && (
           <button onClick={() => loginWithRedirect({})} type="button">
             Log in
           </button> )}
       {isAuthenticated && (
           <button onClick={() => logoutWithRedirect({})} type="button">
             Log out
           </button> )}
   </nav>
 );
};
```

Call a Protected API

```
import React, { useEffect } from 'react';
import { useAuth0 } from '../auth/react-auth0-spa';

export const People = () => {
 const { getTokenSilently } = useAuth0();

 const fetchData = async () => {
   const baseUrl = process.env.REACT_APP_API_URL || 'http://localhost:8080';
   const token = await getTokenSilently();

   const res = await fetch(`${baseUrl}/people`, {
     headers: { Authorization: `Bearer ${token}` },
   });
   // handle result
 };
 useEffect( () => { fetchData(); }, [] );

 return ({/* render data */});
};
```

Login

Token Storage

API Protection

Session Refresh

Logout

Token Storage Locations

	Secure against XSS attacks	Secure against CSRF attacks	Deleted on page session end
Cookies	✗	✗	✗ (2)
localStorage	✗	✓	✗
sessionStorage	✗	✓	✗ (2)
JS object in memory	✓ (1)	✓	✓

1. Depends on the frontend framework

2. Depends on the session resume feature of the browser

Always store your tokens in JavaScript memory.

Login

Token Storage

API Protection

Session Refresh

Logout

Don't use refresh tokens

in single page applications.

Session Refresh: iFrame & Web Messaging

Login

Token Storage

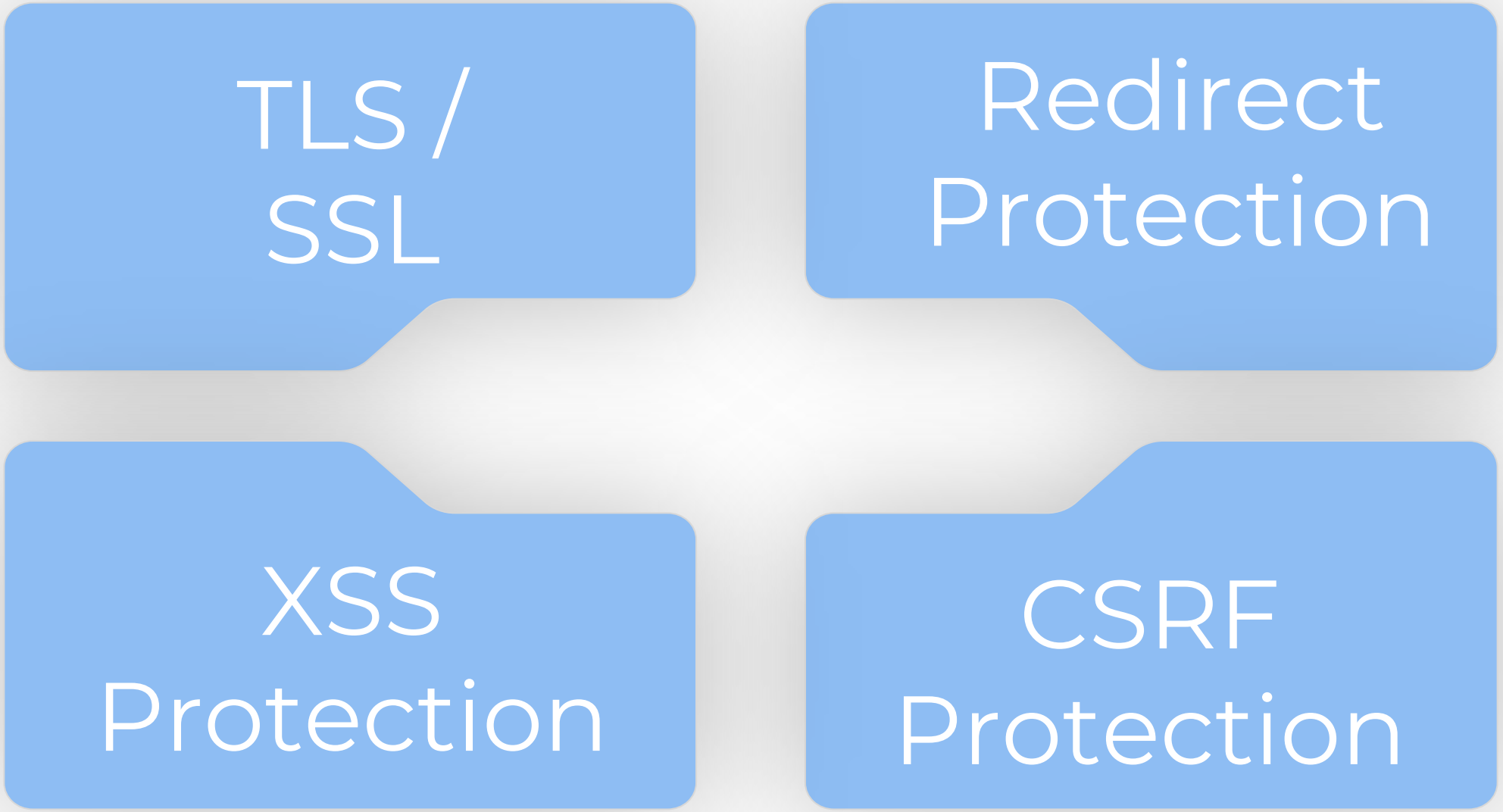
API Protection

Session Refresh

Logout

Easy Logout

```
import React from 'react';
import { useAuth0 } from '../auth/react-auth0-spa';

export const NavBar = () => {
 const { isAuthenticated, loginWithRedirect, logoutWithRedirect } = useAuth0();

 return (
   <nav>
       {!isAuthenticated && (
           <button onClick={() => loginWithRedirect({})} type="button">
             Log in
           </button> )}
       {isAuthenticated && (
           <button onClick={() => logoutWithRedirect({})} type="button">
             Log out
           </button> )}
   </nav>
 );
};
```


TLS /
SSL
Redirect
Protection
XSS
Protection
CSRF
Protection

TLS / SSL

Redirect Protection

XSS Protection

CSRF Protection

TLS / SSL

- 100% required
- Key security feature of all OAuth / OpenID Connect flows
- Get one for free!
  - Let's encrypt
  - Heroku
  - AWS
  - etc.

Never do authentication without protection.

Always use TLS/SSL.

TLS / SSL

Redirect Protection

XSS Protection

CSRF Protection

Redirect Protection

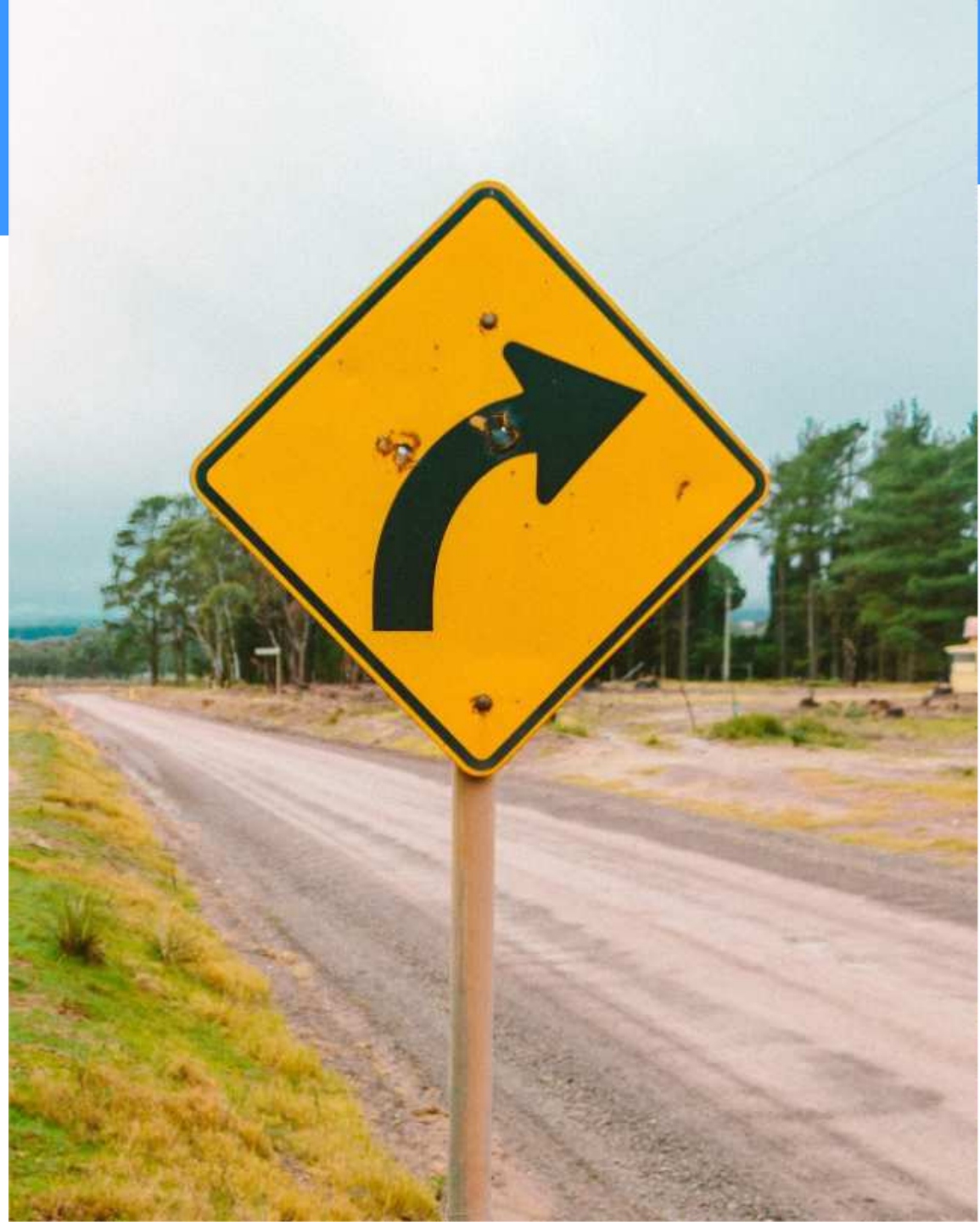
- Use exact redirect targets

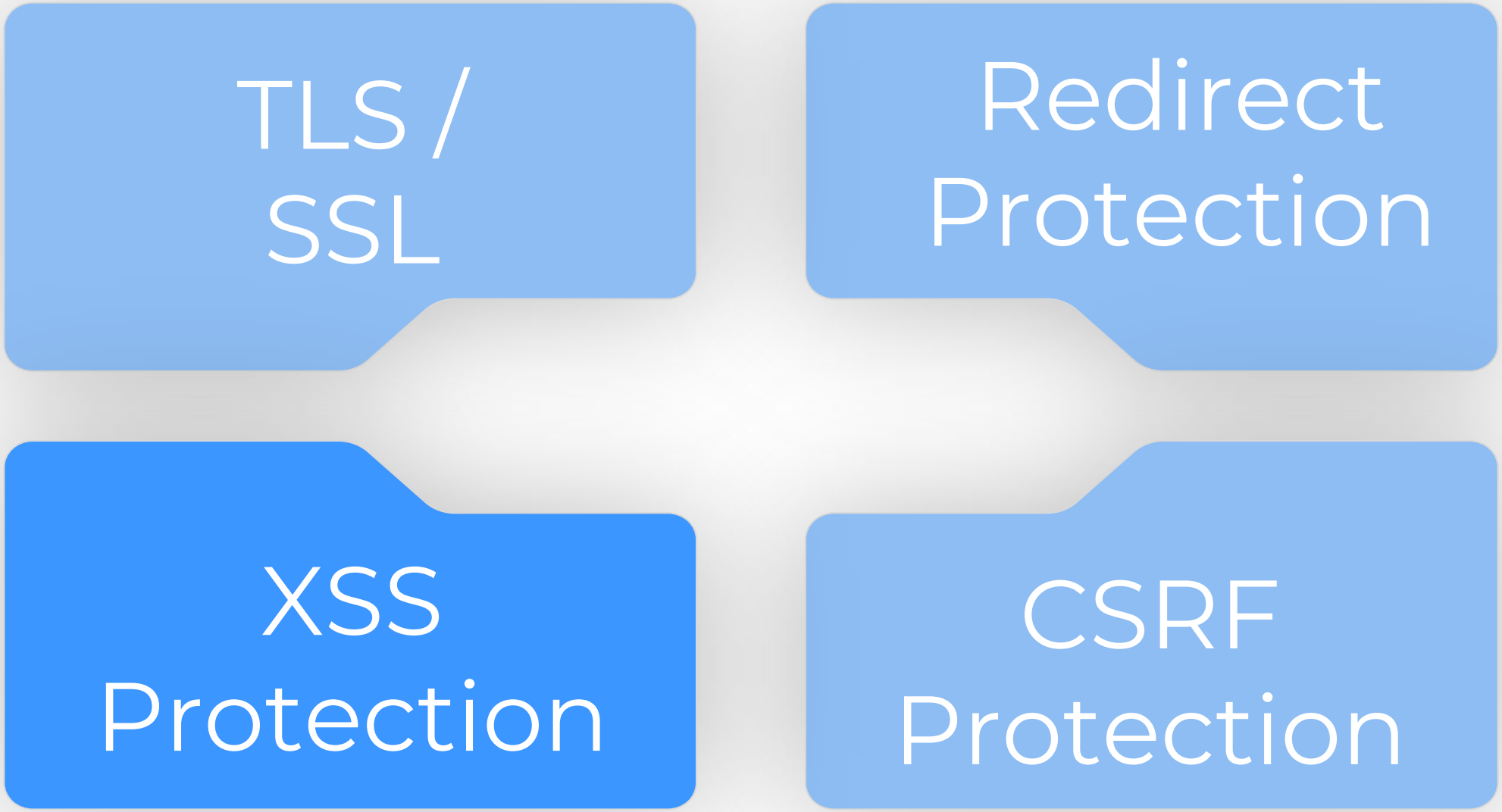
```
https://www.allthingsauth.dev/callback
```

- Do not use wildcards when specifying redirect targets

```
*.allthingsauth.dev/*
```

- Use PKCE




TLS /
SSL
Redirect
Protection
XSS
Protection
CSRF
Protection

Cross-site scripting in React

```
const Profile = ({ name }) => {

  const setHTMLWithXSS = () => {
    return { __html: name };
  };

  return (
  <>
    <p>Your name is: </p>
    <p dangerouslySetInnerHTML={setHTMLWithXSS()}></p>
  </>
  );
};
```

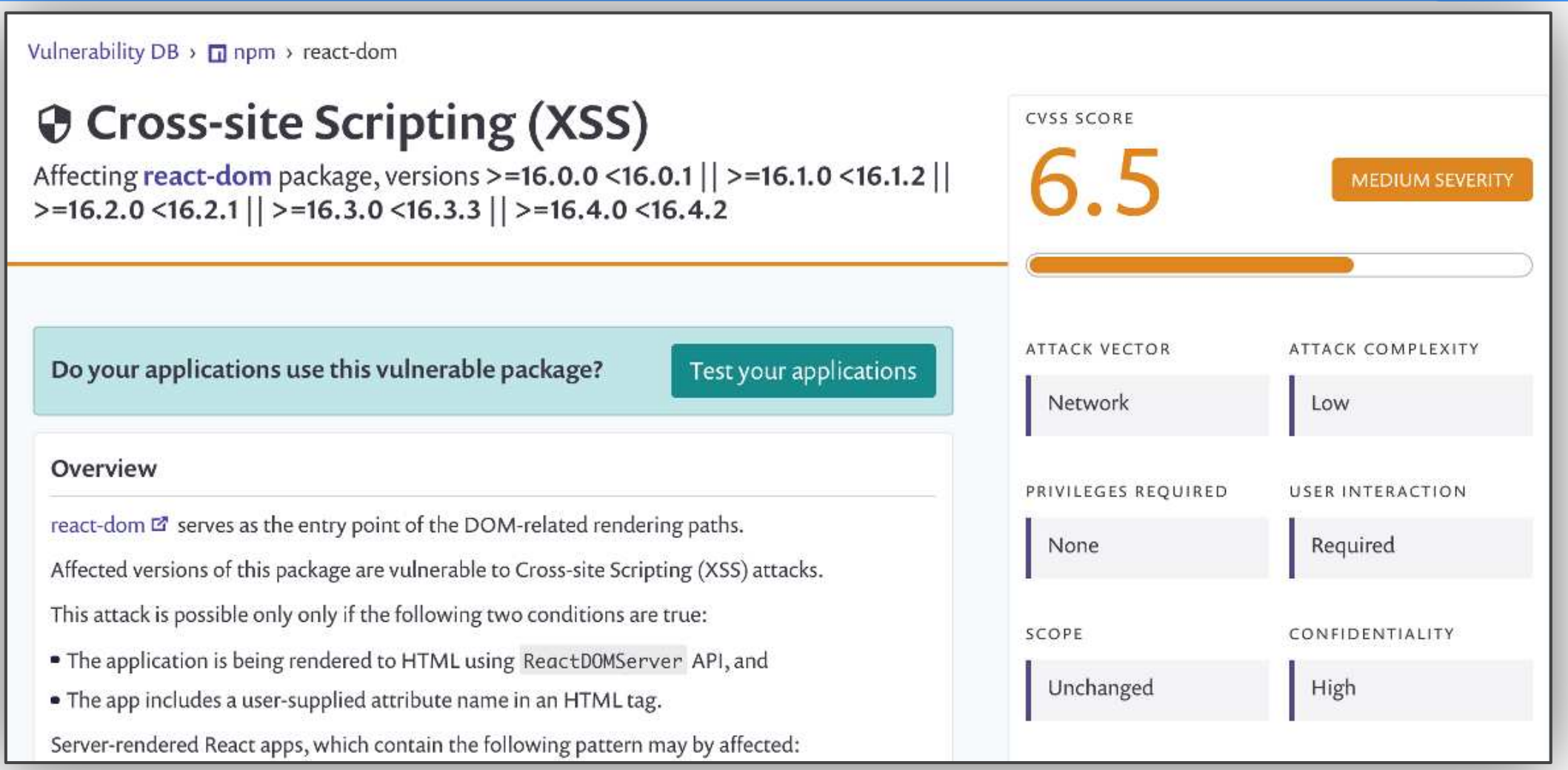
XSS Prevention with ESLint ...

```
},
 plugins: ['react', 'prettier'],
 extends: ['airbnb', 'prettier'],
 rules: {
   'prettier/prettier': 'warn',
   'no-console': 'off',
   'no-use-before-define': 'off',
   'no-plusplus': 'off',
   'react/jsx-filename-extension': 0,
   'react/jsx-boolean-value': 'off',
   'react/prop-types': [0],
   'react/no-danger': 'error',
 },
 overrides: [
```

… and GitHub Actions

```yaml
name: Basic Checks
on: pull_request

jobs:
 basic:
   name: Linting and dependency checks for client
   runs-on: ubuntu-18.04
   steps:
     - name: checkout
       uses: actions/checkout@v1
     - uses: actions/setup-node@v1
       with:
         node-version: "10.x"
     - run: npm install
     - name: linter
       run: npm run lint
     - name: dependency-check
       run: npm audit
```

Cross-site scripting in Third-Party Packages



Vulnerability DB › npm › react-dom

Cross-site Scripting (XSS)

Affecting **react-dom** package, versions >=16.0.0 <16.0.1 || >=16.1.0 <16.1.2 || >=16.2.0 <16.2.1 || >=16.3.0 <16.3.3 || >=16.4.0 <16.4.2

CVSS SCORE

6.5

MEDIUM SEVERITY

ATTACK VECTOR	ATTACK COMPLEXITY
Network	Low
PRIVILEGES REQUIRED	**USER INTERACTION**
None	Required
SCOPE	**CONFIDENTIALITY**
Unchanged	High

Do your applications use this vulnerable package? Test your applications

Overview

react-dom serves as the entry point of the DOM-related rendering paths.

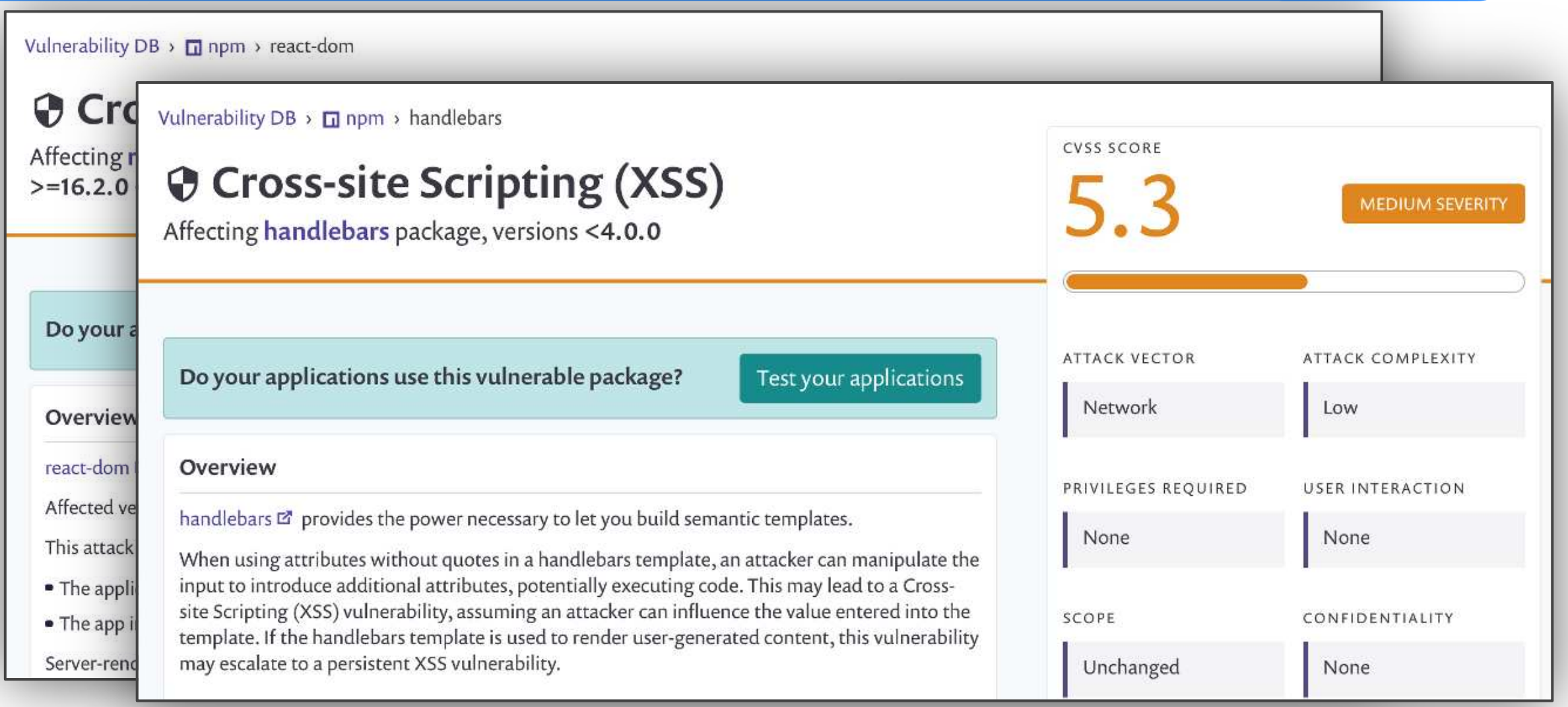
Affected versions of this package are vulnerable to Cross-site Scripting (XSS) attacks.

This attack is possible only only if the following two conditions are true:

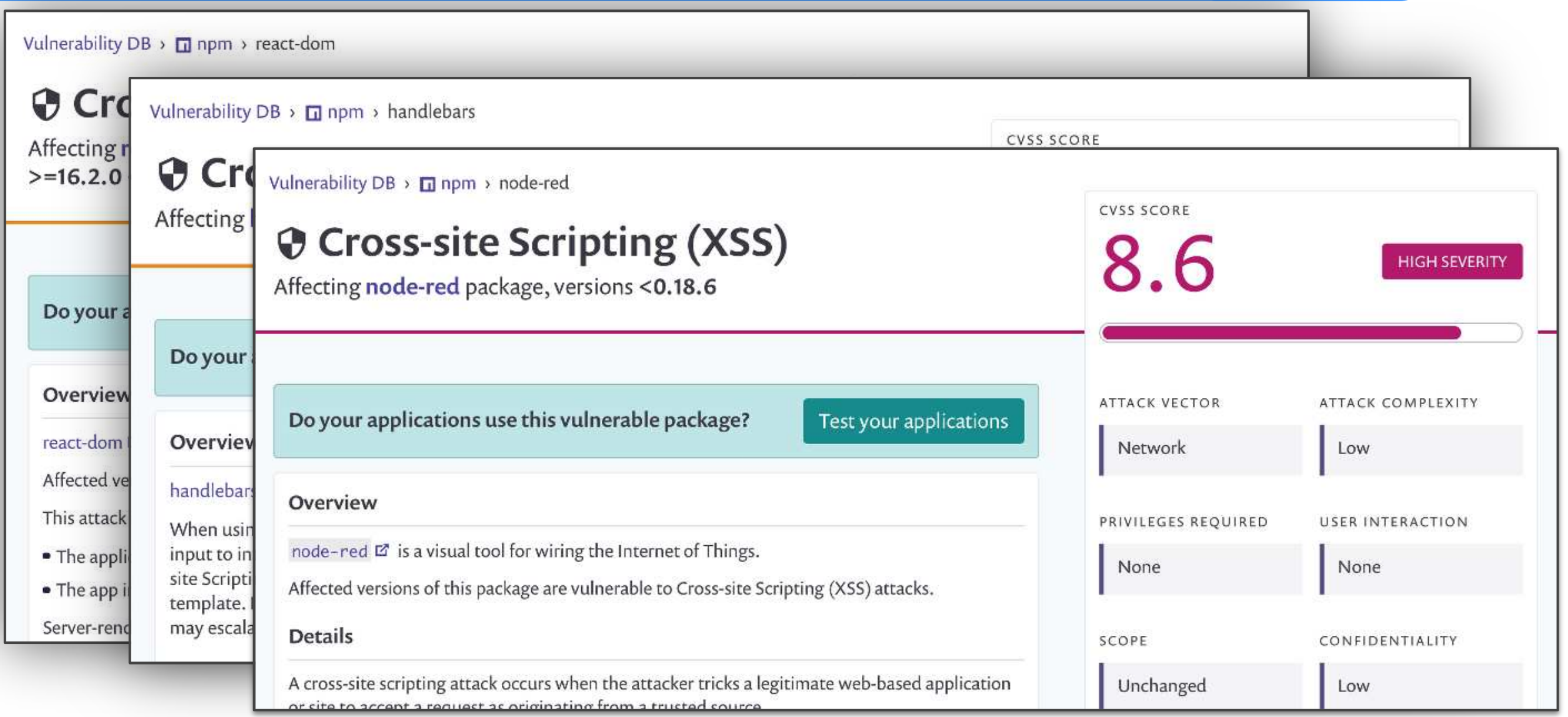
- The application is being rendered to HTML using `ReactDOMServer` API, and
- The app includes a user-supplied attribute name in an HTML tag.

Server-rendered React apps, which contain the following pattern may by affected:

Cross-site scripting in Third-Party Packages



Cross-site scripting in Third-Party Packages

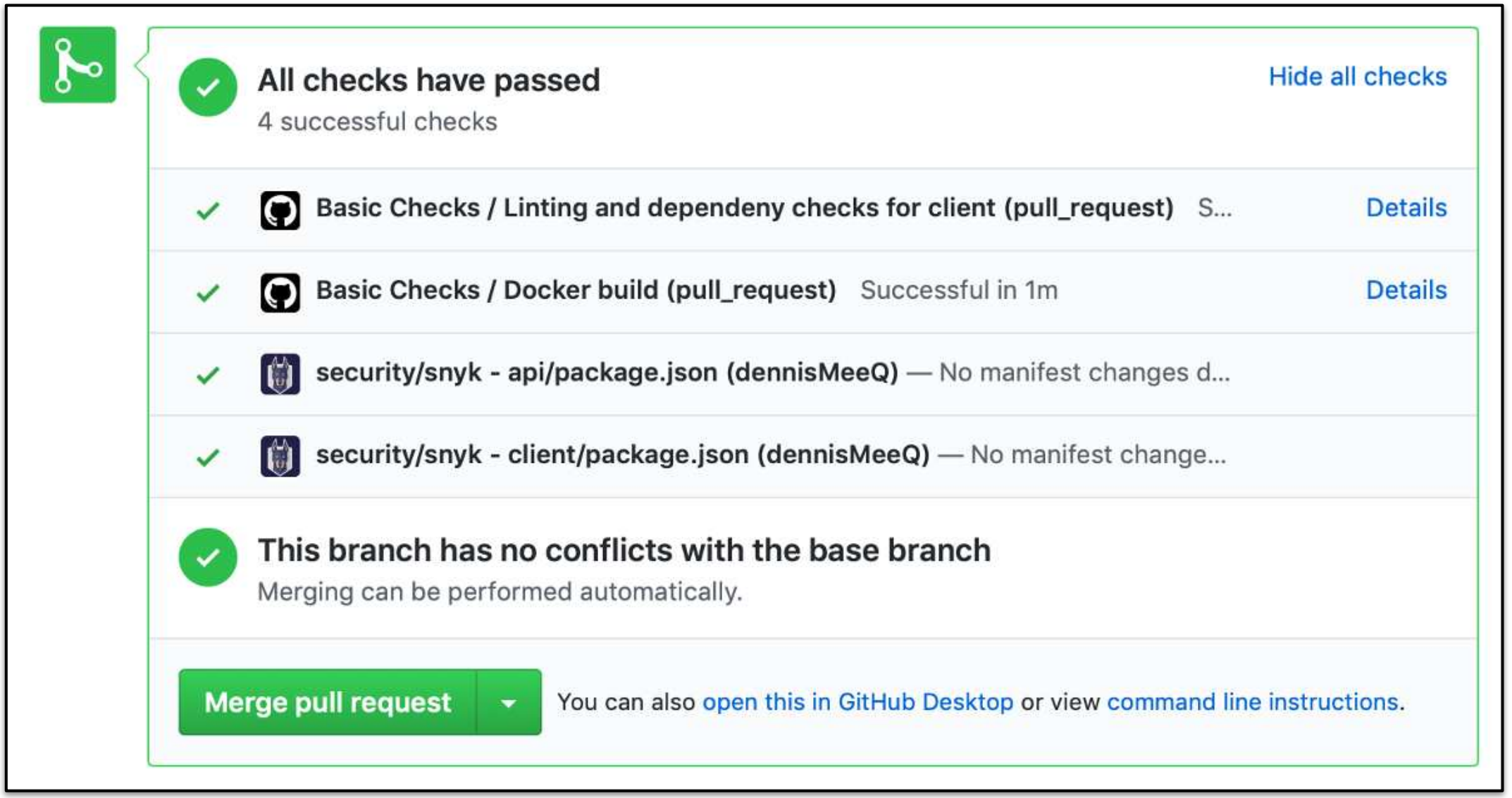


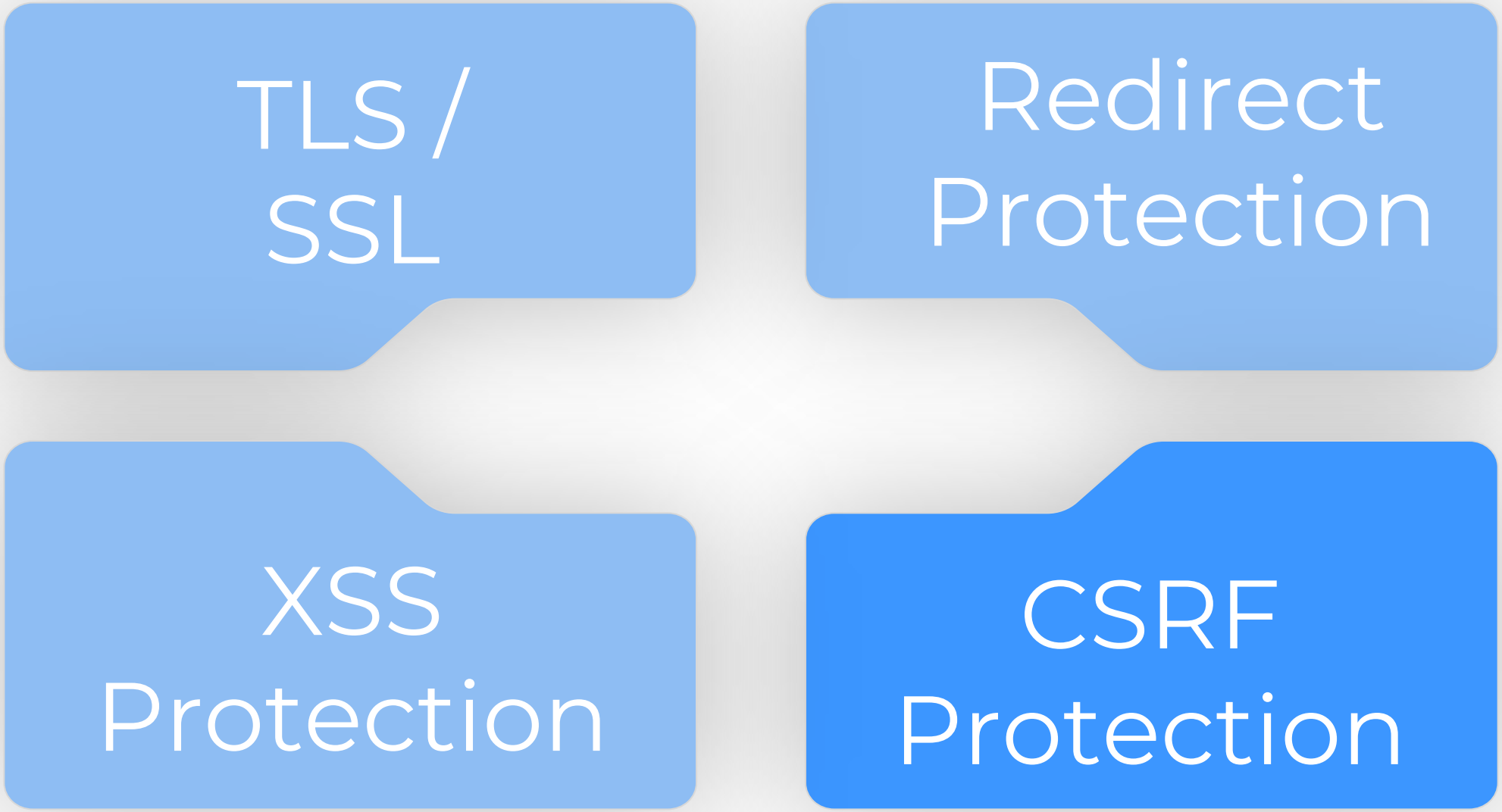
Dependency Checks against 3rd Party XSS

```yaml
name: Basic Checks
on: pull_request

jobs:
 basic:
   name: Linting and dependency checks for client
   runs-on: ubuntu-18.04
   steps:
     - name: checkout
       uses: actions/checkout@v1
     - uses: actions/setup-node@v1
       with:
         node-version: "10.x"
     - run: npm install
     - name: linter
       run: npm run lint
     - name: dependency-check
       run: npm audit
```

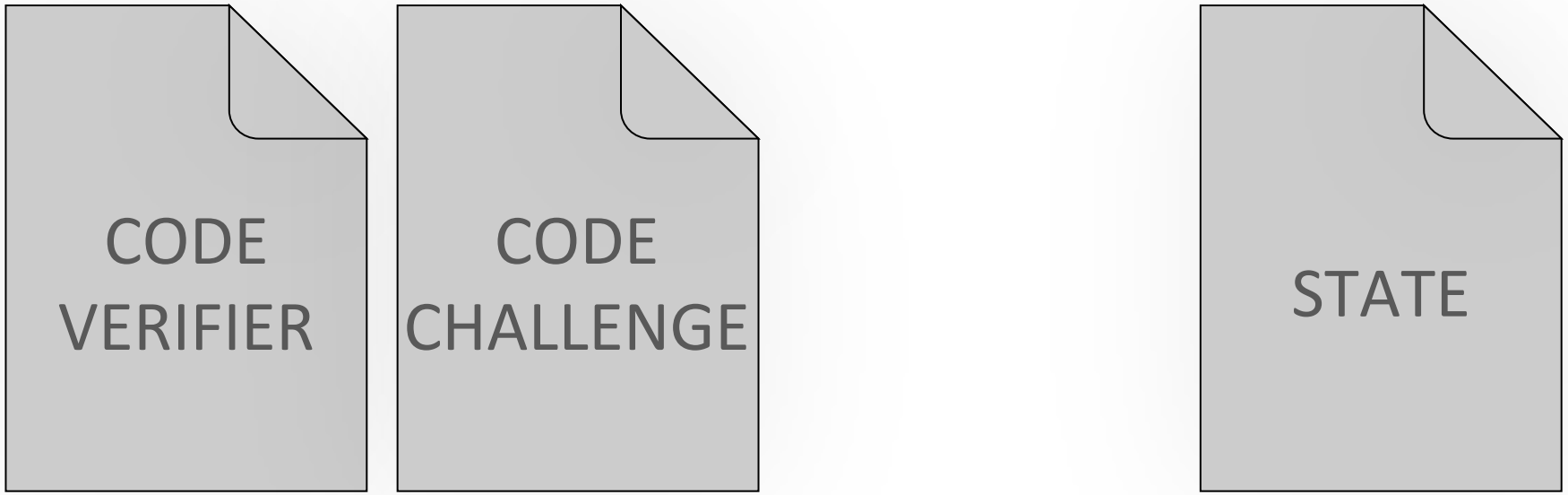
snyk against 3rd Party XSS

snyk against 3rd Party XSS




TLS /
SSL
Redirect
Protection
XSS
Protection
CSRF
Protection

CSRF Protection



PKCE

Further Reading?

Article 1: Concepts, flows and Security considerations

Article 2: Implementation details