A white building with a black background

Description automatically generated

Міністерство освіти і науки України

Національний технічний університет України „КПІ

імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**ЗВІТ**

лабораторної работи №4

курсу «Безпека програмного забезпечення»

Тема: «Засвоювання базових навичок роботи з валідацією токенів»

Перевірив:

Курченко О.А.

Виконав:

Група ІП-02, Сергієнко Яків

Київ 2023

**Завдання:**

Розширити **Лабораторну работу 4** перевіркою сігнатури JWT токена. Приклади SDK https://auth0.com/docs/quickstart/backend. У випадку асиметричного ключа, public є можливість отримати за посиланням https://kpi.eu.auth0.com/pem, aбо за формулою https://[API\_DOMAIN]/pem  
Надати код рішення.

**Хід роботи:**

В даній лабораторній роботі мною було модифіковано файл index.js таким чином, що токен перевіряється функцією jwt.verify з допомогою приватного ключа для валідації коректності.

A computer screen with many colorful text

Description automatically generated with medium confidence

**Лістинг коду:**

index.js

const uuid = require('uuid');

const express = require('express');

const onFinished = require('on-finished');

const bodyParser = require('body-parser');

const path = require('path');

const port = 3000;

const fs = require('fs');

const axios = require('axios');

const jwt = require('jsonwebtoken');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

const SESSION\_KEY = 'Authorization';

const URL = 'https://dev-d9q43cd8.us.auth0.com/';

const AUDIENCE\_URL = 'https://dev-d9q43cd8.us.auth0.com/api/v2/';

const CLIENT\_ID = 'SBAG5My1TL0hePd9MvWEdVkGJkM84For';

const CLIENT\_SECRET = 'SHcdsMA2WEkNmfEIwWEdw8NTy4IMHXxy2wD-8MlH6-eNHa0XD3LR1KuEbYPicX96';

app.get('/', async (req, res) => {

const access\_token = req.header('Authorization');

if (!access\_token) {

return res.sendFile(path.join(\_\_dirname + '/index.html'));

}

var privateKey = fs.readFileSync('dev-d9q43cd8.pem');

jwt.verify(access\_token, privateKey, { algorithms: ['RS256'] }, async (err, decoded) => {

if (err) {

return res.sendFile(path.join(\_\_dirname + '/index.html'));

} else {

const currentTime = Date.now() / 1000;

if (decoded.exp > currentTime) {

const response = await axios.get(`${AUDIENCE\_URL}users/${decoded.sub}`, {

headers: {

'Authorization': `Bearer ${access\_token}`,

'Content-Type': 'application/json',

'Accept': 'application/json'

}

})

return res.json({

username: response.data.nickname,

logout: 'http://localhost:3000/logout',

expires: decoded.exp

})

}

}

});

})

app.get('/logout', (req, res) => {

sessions.destroy(req, res);

res.redirect('/');

});

app.post('/api/login', async (req, res) => {

const { username, password } = req.body;

try {

const response = await axios.post(`${URL}oauth/token`, {

grant\_type: 'http://auth0.com/oauth/grant-type/password-realm',

client\_id: CLIENT\_ID,

client\_secret: CLIENT\_SECRET,

audience: AUDIENCE\_URL,

scope: 'offline\_access',

realm: 'Username-Password-Authentication',

username: username,

password: password

}, {

headers: {

'Content-Type': 'application/json',

'Accept': 'application/json'

}

});

const data = response.data;

res.status(201).send({

username: username,

access\_token: data.access\_token,

refresh\_token: data.refresh\_token

});

} catch (err) {

res.status(403).send(err.response.data.error\_description);

}

});

app.post('/api/refresh', async (req, res) => {

console.log(req.body);

try {

const response = await axios.post(`${URL}oauth/token/`, {

grant\_type: 'refresh\_token',

audience: AUDIENCE\_URL,

client\_id: CLIENT\_ID,

client\_secret: CLIENT\_SECRET,

refresh\_token: req.body.refresh\_token,

scope: 'offline\_access',

realm: 'Username-Password-Authentication',

})

return res.json({ access\_token: response.data.access\_token});

} catch (err) {

return res.status(401).send(err.response.data.error\_description);

}

});

app.get('/create-user', (req, res) => {

return res.sendFile(path.join(\_\_dirname + '/create-user.html'));

});

app.post('/api/create-user', async (req, res) => {

const { email, password } = req.body;

try {

const { data } = await axios.post(`${URL}oauth/token`, {

grant\_type: 'client\_credentials',

audience: AUDIENCE\_URL,

client\_id: CLIENT\_ID,

client\_secret: CLIENT\_SECRET,

}, {

headers: {

'Content-Type': 'application/json',

'Accept': 'application/json'

}

});

const response = await axios.post(`${AUDIENCE\_URL}users`, {

email: email,

password: password,

connection: 'Username-Password-Authentication'

}, {

headers: {

'Content-Type': 'application/json',

'Accept': 'application/json',

'Authorization': `Bearer ${data.access\_token}`

}

});

if (response.data) {

res.status(201).json({ message: 'User created' });

} else {

res.status(401);

}

} catch (error) {

res.status(401).json({ error: error.response.data });

}

});

app.listen(port, () => {

console.log(`Example app listening on port ${port}`)

})

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

</head>

<body>

<main id="main-holder">

<a href="/logout" id="logout">Logout</a>

<a href="/create-user" id="create-user">Create user</a>

<h1 id="login-header">Login</h1>

<div id="login-error-msg-holder">

<p id="login-error-msg">Invalid username <span id="error-msg-second-line">and/or password</span></p>

</div>

<form id="login-form" action="/api/login" method="post">

<input type="text" name="username" id="username-field" class="login-form-field" placeholder="Username">

<input type="password" name="password" id="password-field" class="login-form-field" placeholder="Password">

<input type="submit" value="Login" id="login-form-submit">

</form>

</main>

</body>

<style>

html {

height: 100%;

}

body {

height: 100%;

margin: 0;

font-family: Arial, Helvetica, sans-serif;

display: grid;

justify-items: center;

align-items: center;

background-color: #3a3a3a;

}

#logout {

opacity: 0;

}

#main-holder {

width: 50%;

height: 70%;

display: grid;

justify-items: center;

align-items: center;

background-color: white;

border-radius: 7px;

box-shadow: 0px 0px 5px 2px black;

}

#login-error-msg-holder {

width: 100%;

height: 100%;

display: grid;

justify-items: center;

align-items: center;

}

#login-error-msg {

width: 23%;

text-align: center;

margin: 0;

padding: 5px;

font-size: 12px;

font-weight: bold;

color: #8a0000;

border: 1px solid #8a0000;

background-color: #e58f8f;

opacity: 0;

}

#error-msg-second-line {

display: block;

}

#login-form {

align-self: flex-start;

display: grid;

justify-items: center;

align-items: center;

}

.login-form-field::placeholder {

color: #3a3a3a;

}

.login-form-field {

border: none;

border-bottom: 1px solid #3a3a3a;

margin-bottom: 10px;

border-radius: 3px;

outline: none;

padding: 0px 0px 5px 5px;

}

#login-form-submit {

width: 100%;

padding: 7px;

border: none;

border-radius: 5px;

color: white;

font-weight: bold;

background-color: #3a3a3a;

cursor: pointer;

outline: none;

}

</style>

<script>

const access\_token = sessionStorage.getItem('access\_token');

const refresh\_token = sessionStorage.getItem('refresh\_token');

const loginForm = document.getElementById("login-form");

const loginButton = document.getElementById("login-form-submit");

const loginErrorMsg = document.getElementById("login-error-msg");

const logoutLink = document.getElementById("logout");

const createUserLink = document.getElementById("create-user");

const refreshToken = () => {

axios.post('/api/refresh', {

refresh\_token: sessionStorage.getItem('refresh\_token')

}).then((response) => {

sessionStorage.setItem('access\_token', response.data.access\_token);

location.reload();

}).catch((response) => {

loginErrorMsg.style.opacity = 1;

loginErrorMsg.innerHTML = response

});

};

if (access\_token) {

axios.get('/', {

headers: {

Authorization: access\_token

}

}).then((response) => {

const { username } = response.data;

if (username) {

console.log(response.data.expires \* 1000 - Date.now());

const mainHolder = document.getElementById("main-holder");

const loginHeader = document.getElementById("login-header");

loginForm.remove();

loginErrorMsg.remove();

loginHeader.remove();

createUserLink.remove();

mainHolder.append(`Hello, ${username}`);

logoutLink.style.opacity = 1;

setTimeout(() => {

refreshToken();

}, (response.data.expires - 60) \* 1000 - Date.now());

// 5000);

}

});

}

logoutLink.addEventListener("click", (e) => {

e.preventDefault();

sessionStorage.removeItem('access\_token');

sessionStorage.removeItem('refresh\_token');

location.reload();

});

loginButton.addEventListener("click", (e) => {

e.preventDefault();

const username = loginForm.username.value;

const password = loginForm.password.value;

axios({

method: 'post',

url: '/api/login',

data: {

username,

password

}

}).then((response) => {

const { username } = response.data.username;

sessionStorage.setItem('access\_token', response.data.access\_token);

sessionStorage.setItem('refresh\_token', response.data.refresh\_token);

location.reload();

}).catch((response) => {

loginErrorMsg.style.opacity = 1;

loginErrorMsg.innerHTML = response.response.data

});

})

</script>

</html>

create-user.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Create user</title>

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

</head>

<body>

<main id="main-holder">

<a href="/" id="login">Login</a>

<h1 id="create-header">Create-user</h1>

<div id="create-successfull-msg-holder">

<p id="create-successfull-msg">User created successfully</p>

</div>

<div id="create-error-msg-holder">

<p id="create-error-msg">User already exists</p>

</div>

<div id="create-error-msg-holder">

<p id="create-error-msg-2">Email must be correct and

password must contain at least 8 characters including

a lower-case letter, an upper-case letter, and a number.

</p>

</div>

<form id="create-form" action="/api/create" method="post">

<input type="text" name="email" id="email-field" class="create-form-field" placeholder="Email">

<input type="password" name="password" id="password-field" class="create-form-field" placeholder="Password">

<input type="submit" value="Create" id="create-form-submit">

</form>

</main>

</body>

<style>

html {

height: 100%;

}

body {

height: 100%;

margin: 0;

font-family: Arial, Helvetica, sans-serif;

display: grid;

justify-items: center;

align-items: center;

background-color: #3a3a3a;

}

#logout {

opacity: 0;

}

#main-holder {

width: 50%;

height: 70%;

display: grid;

justify-items: center;

align-items: center;

background-color: white;

border-radius: 7px;

box-shadow: 0px 0px 5px 2px black;

}

#create-error-msg-holder {

width: 100%;

height: 100%;

display: grid;

justify-items: center;

align-items: center;

}

#create-error-msg {

width: 23%;

text-align: center;

margin: 0;

padding: 5px;

font-size: 12px;

font-weight: bold;

color: #8a0000;

border: 1px solid #8a0000;

background-color: #e58f8f;

opacity: 0;

}

#create-error-msg-2 {

width: 23%;

text-align: center;

margin: 0;

padding: 5px;

font-size: 12px;

font-weight: bold;

color: #8a0000;

border: 1px solid #8a0000;

background-color: #e58f8f;

opacity: 0;

}

#create-successfull-msg-holder {

width: 100%;

height: 100%;

display: grid;

justify-items: center;

align-items: center;

}

#create-successfull-msg {

width: 20%;

text-align: center;

margin: 0;

padding: 5px;

font-size: 12px;

font-weight: bold;

color: rgb(0, 0, 0);

box-shadow: 0px 0px 2px 2px green;

opacity: 0;

}

#error-msg-second-line {

display: block;

}

#create-form {

align-self: flex-start;

display: grid;

justify-items: center;

align-items: center;

}

.create-form-field::placeholder {

color: #3a3a3a;

}

.create-form-field {

border: none;

border-bottom: 1px solid #3a3a3a;

margin-bottom: 10px;

border-radius: 3px;

outline: none;

padding: 0px 0px 5px 5px;

}

#create-form-submit {

width: 100%;

padding: 7px;

border: none;

border-radius: 5px;

color: white;

font-weight: bold;

background-color: #3a3a3a;

cursor: pointer;

outline: none;

}

</style>

<script>

const createForm = document.getElementById("create-form");

const createButton = document.getElementById("create-form-submit");

const createErrorMsg = document.getElementById("create-error-msg");

const createErrorMsg2 = document.getElementById("create-error-msg-2");

const createSuccessfullMsg = document.getElementById("create-successfull-msg");

createButton.addEventListener("click", (e) => {

e.preventDefault();

const email = createForm.email.value;

const password = createForm.password.value;

axios({

method: 'post',

url: '/api/create-user',

data: {

email,

password

}

}).then((response) => {

createSuccessfullMsg.style.opacity = 1;

createErrorMsg.style.opacity = 0;

createErrorMsg2.style.opacity = 0;

}).catch((response) => {

createSuccessfullMsg.style.opacity = 0;

createErrorMsg.style.opacity = 0;

createErrorMsg2.style.opacity = 0;

if (response.response.data.error.message == 'The user already exists.') {

createErrorMsg.style.opacity = 1;

} else {

createErrorMsg2.style.opacity = 1;

createErrorMsg2.innerHTML = response.response.data.error.message;

}

});

})

</script>

</html>

**Висновки:**

Під час виконання даної лабораторної роботи мною було модифіковано програму token\_auth таким чином, щоб після авторизації та отримання токену, перевіряється валідність jwt токену, використовуючи стандартні методи бібліотеки jwt та приватний ключ pem.