

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 №5

курсу «Безпека програмного забезпечення»

Тема: «Засвоювання базових навичок роботи з валідацією токенів»

Перевірив: Виконав:

Курченко О. А. Група ІП-02

Трофимов Данило

Київ 2023

Завдання:

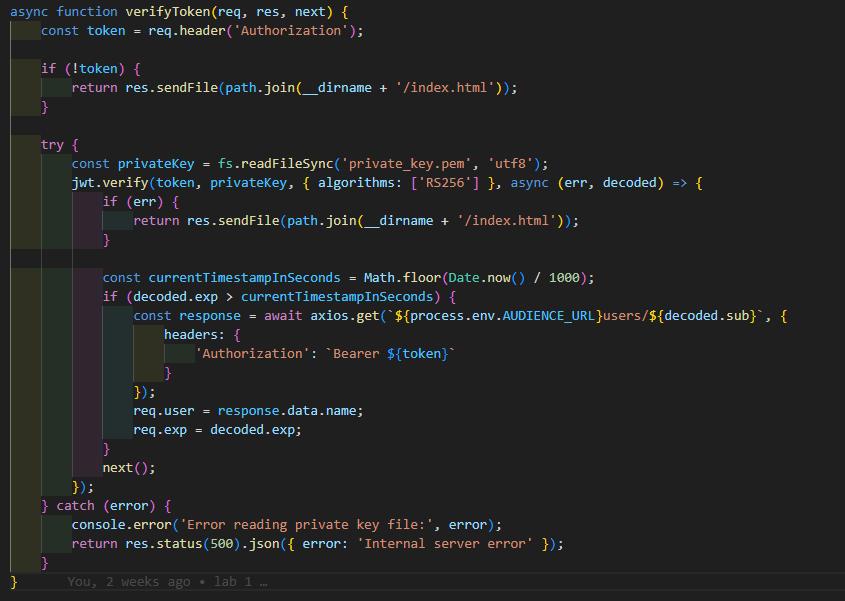
Розширити Лабораторну работу 4 перевіркой сігнатури JWT токена. Приклади

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>

Хід роботи:

Лістинг програмного коду:

Було модифіковано файл index.js так, що у функції verifyToken замість jwt.decode використовується функція jwt.verify з приватним ключем для валідації коректності jwt токену.



Index.js:

const uuid = require('uuid');

const express = require('express');

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

const path = require('path');

const dotenv = require('dotenv');

const port = 3000;

const jwt = require('jsonwebtoken');

const axios = require('axios');

const fs  = require('fs');

dotenv.config();

const app = express();

app.use(bodyParser.json());

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

const GrantTypes = {

    PASSWORD: 'password',

    AUTHORIZATION\_CODE: 'authorization\_code',

    CLIENT\_CREDENTIALS: 'client\_credentials',

    IMPLICIT: 'implicit',

    REFRESH\_TOKEN: 'refresh\_token',

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

};

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

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

});

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

    res.json({

        username: req.user,

        exp: req.exp,

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

    });

})

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

    res.redirect('/');

});

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

    const { username, password } = req.body;

    try {

        const response = await axios.post(`${process.env.API\_URL}oauth/token`, {

            grant\_type: GrantTypes.PASSWORD\_REALM,

            audience: process.env.AUDIENCE\_URL,

            client\_id: process.env.CLIENT\_ID,

            client\_secret: process.env.CLIENT\_SECRET,

            scope: 'offline\_access',

            realm: 'Username-Password-Authentication',

            username: username,

            password: password,

        }, {

            headers: {

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

                'Accept': 'application/json'

            }

        });

        res.status(201).json({ access\_token: response.data.access\_token, refresh\_token: response.data.refresh\_token, username: username });

    } catch (error) {

        console.log(error)

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

    }

});

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

    const { email, password } = req.body;

    try {

        const response = await axios.post(`${process.env.API\_URL}oauth/token`, {

            grant\_type: GrantTypes.CLIENT\_CREDENTIALS,

            audience: process.env.AUDIENCE\_URL,

            client\_id: process.env.CLIENT\_ID,

            client\_secret: process.env.CLIENT\_SECRET,

        }, {

            headers: {

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

                'Accept': 'application/json'

            }

        });

        const response2 = await axios.post(`${process.env.AUDIENCE\_URL}users`, {

            email: email,

            password: password,

            connection: 'Username-Password-Authentication'

        }, {

            headers: {

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

                'Accept': 'application/json',

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

            }

        });

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

    } catch (error) {

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

    }

});

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

    const { refresh\_token } = req.body;

    try {

        const response = await axios.post(`${process.env.API\_URL}oauth/token`, {

            grant\_type: GrantTypes.REFRESH\_TOKEN,

            audience: process.env.AUDIENCE\_URL,

            client\_id: process.env.CLIENT\_ID,

            client\_secret: process.env.CLIENT\_SECRET,

            refresh\_token: refresh\_token,

            scope: 'offline\_access',

            realm: 'Username-Password-Authentication',

        }, {

            headers: {

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

                'Accept': 'application/json'

            }

        });

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

    } catch (error) {

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

    }

});

app.listen(port, () => {

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

})

async function verifyToken(req, res, next) {

    const token = req.header('Authorization');

    if (!token) {

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

    }

    try {

        const privateKey = fs.readFileSync('private\_key.pem', 'utf8');

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

            if (err) {

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

            }

            const currentTimestampInSeconds = Math.floor(Date.now() / 1000);

            if (decoded.exp > currentTimestampInSeconds) {

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

                    headers: {

                        'Authorization': `Bearer ${token}`

                    }

                });

                req.user = response.data.name;

                req.exp = decoded.exp;

            }

            next();

        });

    } catch (error) {

        console.error('Error reading private key file:', error);

        return res.status(500).json({ error: 'Internal server error' });

    }

}

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>

        <button id="refresh"> Manual refresh </button>

        <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;

    }

    #refresh {

        opacity: 0;

    }

    #create-user {

        opacity: 1;

    }

    #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');

    let exp;

    if (access\_token) {

        axios.get('/', {

            headers: {

                Authorization: access\_token

            }

        }).then((response) => {

            const { username } = response.data;

            exp = response.data.exp;

            if (username) {

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

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

                loginForm.remove();

                loginErrorMsg.remove();

                loginHeader.remove();

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

                logoutLink.style.opacity = 1;

                refreshButton.style.opacity = 1;

                createUserLink.style.opacity = 0;

            }

        });

        if (exp < Math.floor(Date.now() / 1000) - 300) {

            refresh();

        }

    }

    function refresh() {

        axios({

            method: 'post',

            url: '/api/refresh',

            data: {

                refresh\_token

            }

        }).then((response) => {

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

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

            location.reload();

        }).catch((response) => {

            sessionStorage.removeItem('access\_token');

            sessionStorage.removeItem('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 refreshButton = document.getElementById("refresh");

    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) => {

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

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

            location.reload();

        }).catch((response) => {

            loginErrorMsg.style.opacity = 1;

            sessionStorage.removeItem('access\_token');

            sessionStorage.removeItem('refresh\_token');

        });

    })

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

        e.preventDefault();

        refresh();

    })

</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-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>

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

            <p id="create-successfull-msg">User created successfully</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: 23%;

        text-align: center;

        margin: 0;

        padding: 5px;

        font-size: 12px;

        font-weight: bold;

        color: rgb(0, 0, 0);

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

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

    if (access\_token) {

        axios.get('/', {

            headers: {

                Authorization: access\_token

            }

        }).then((response) => {

            const { username } = response.data;

            if (username) {

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

                const createHeader = document.getElementById("create-header");

                createForm.remove();

                createErrorMsg.remove();

                createHeader.remove();

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

                logoutLink.style.opacity = 1;

            }

        });

    }

    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',

            data: {

                email,

                password

            }

        }).then((response) => {

            createSuccessfullMsg.style.opacity = 1;

        }).catch((response) => {

            createErrorMsg.style.opacity = 0;

            createErrorMsg2.style.opacity = 0;

            console.log(response.response);

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

                createErrorMsg.style.opacity = 1;

            else

                createErrorMsg2.style.opacity = 1;

        });

    })

</script>

</html>

Код програми представлено в інтернет-ресурсі: <https://github.com/DanyloTrofymov/Software-security/tree/lab5>

Висновки: У цьому завданні було модифіковано програму token\_auth так, що після авторизації перевіряється валідність jwt токену за допомогою стандартних методів бібліотеки jsonwebtoken та приватного ключа у форматі .pem