МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

Про виконання лабораторної роботи №6
З дисципліни “Безпека програмного забезпечення”
На тему “ Засвоювання базових навичок роботи з валідацією токенів”

Виконали:
Студенти групи ІП-15
Мєшков А. І.

Перевірила:
пос. Соколовський В. В.

Київ 2024

ЛАБОРАТОРНА РОБОТА №6

Завдання:

Розширити **Лабораторну роботу 4**, змінивши логін сторінку на стандартну від SSO провайдера, для цього, треба зробити редірект на API_DOMAIN

https://kpi.eu.auth0.com/authorize

та додатково додати параметри Вашого аплікейшена

client_id, redirect_uri, response_type=code, response_mode=query

https://kpi.eu.auth0.com/authorize?client_id=JIvCO5c2IBHlAe2patn6l6q5H35qxti0&redirect_uri=http%3A%2F%2Flocalhost%3A3000&response_type=code&response_mode=query

Надати код рішення.

ХІД РОБОТИ

1. Код було розширено.

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" style="opacity: 0;">Logout</a>

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

        <button id="login-btn">Login with Auth0</button>
    </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 session = sessionStorage.getItem('session');
        let token;

        try {
            token = JSON.parse(session).access_token;
        } catch(e) {}

        if (token) {
            axios.get('/', {
                headers: {
                    Authorization: token
                }
            }).then((response) => {
                const { username } = response.data;


                if (username) {
                    const mainHolder = document.getElementById("main-holder");
                    const loginHeader = document.getElementById("login-header");

                    loginForm.remove();
                    loginHeader.remove();
                    mainHolder.append(`Hello ${username}`);
                    logoutLink.style.opacity = 1;
                }
            });
        }

        const loginBtn = document.getElementById("login-btn");
        loginBtn.addEventListener("click", () => {
            window.location.href = '/login';
        });

        const logoutLink = document.getElementById("logout");
        logoutLink.addEventListener("click", (e) => {
            e.preventDefault();
            sessionStorage.removeItem('session');
            location.reload();
```

```
        });
        window.onload = () => {
            const urlParams = new URLSearchParams(window.location.search);
            const username = urlParams.get('username');
            const access_token = urlParams.get('access_token');
            console.log(username)

            if (username && access_token) {
                const sessionData = {
                    username: username,
                    access_token: access_token
                };

                sessionStorage.setItem('session', JSON.stringify(sessionData));

                // window.location.href = '/';
            }
        };

    </script>
</html>
```

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 jwksClient = require('jwks-rsa');

const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

const SESSION_KEY = 'Authorization';

const AUTH0_DOMAIN = 'dev-6sww0yh4s3mew71l.us.auth0.com';
const CLIENT_ID = 'bpgWdV1Dlbin2T2VYq3J0nmsRe7zrZ5G';
const CLIENT_SECRET = '1eJqaL3x_EKq-
_682T6vY5KEKWKLWMK77l5R1WAc02CYvbAI7oUia49C02gRWiFf';
const AUDIENCE = 'https://dev-6sww0yh4s3mew71l.us.auth0.com/api/v2/';
const TOKEN_URL = `https://${AUTH0_DOMAIN}/oauth/token`;
const REDIRECT_URI = 'http://localhost:3000/callback';

const client = jwksClient({
    jwksUri: `https://${AUTH0_DOMAIN}/.well-known/jwks.json`
});
```

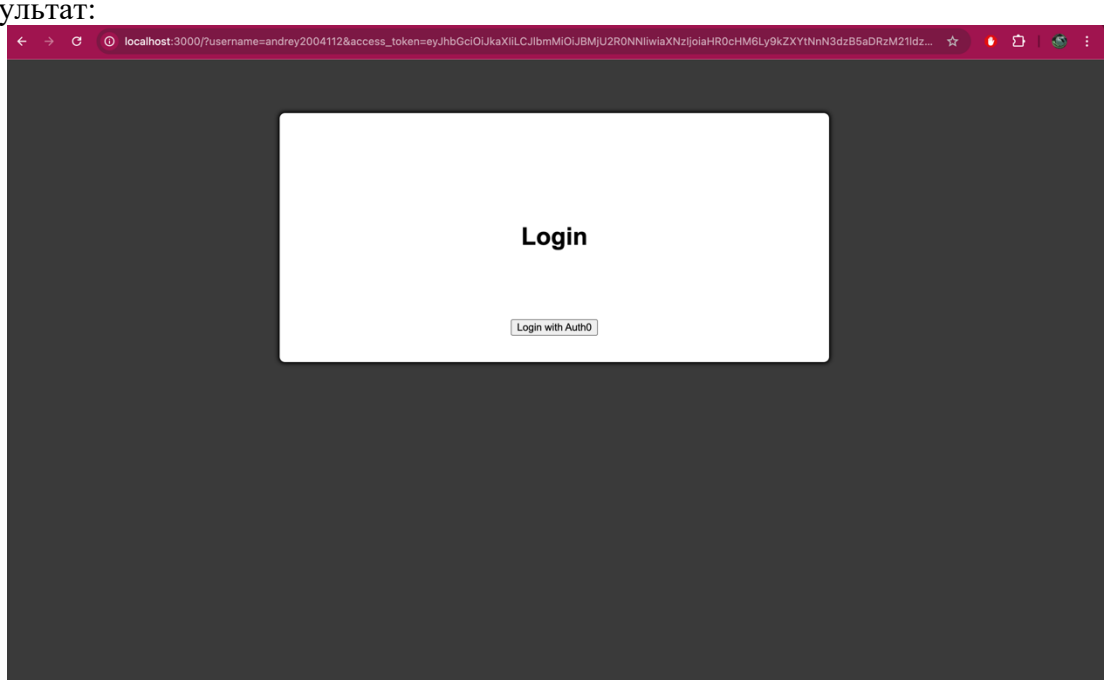
```
const getKey = (header, callback) => {
    client.getSigningKey(header.kid, (err, key) => {
        if (err) return callback(err);
        const signingKey = key.publicKey || key.rsaPublicKey;
        callback(null, signingKey);
    });
};

class Session {
    #sessions = {}

    constructor() {
        try {
            this.#sessions = fs.readFileSync('./sessions.json', 'utf8');
            this.#sessions = JSON.parse(this.#sessions.trim());
        } catch (e) {
            this.#sessions = {};
        }
    }
    #storeSessions() {
        fs.writeFileSync('./sessions.json', JSON.stringify(this.#sessions), 'utf-
8');
    }

    set(key, value) {
        if (!value) {
            value = {};
        }
        this.#sessions[key] = value;
        this.#storeSessions();
    }

    get(key) {
        return this.#sessions[key];
    }

    init(res) {
        const sessionId = uuid.v4();
        this.set(sessionId);
        return sessionId;
    }

    destroy(req, res) {
        let sessionId = this.findSessionByAccessToken(req.session.access_token);
        while(sessionId){
            delete this.#sessions[sessionId];
            sessionId = this.findSessionByAccessToken(req.session.access_token);
        }

        this.#storeSessions();
```

```
    }

    findSessionByAccessToken(accessToken) {
        for (const sessionId in this.#sessions) {
            if (this.#sessions[sessionId].access_token === accessToken) {
                return this.#sessions[sessionId];
            }
        }
        return null;
    }

    getSessionFromAccessTokenOrCreate(accessToken, res) {
        let currentSession = this.findSessionByAccessToken(accessToken);
        let sessionId;

        if (currentSession) {
            sessionId = currentSession.sessionId;
        }

        return { currentSession, sessionId };
    }
}

const sessions = new Session();

app.use((req, res, next) => {
    let currentSession = {};
    const accessToken = req.get(SESSION_KEY);
    let sessionId = req.sessionId;

    if (accessToken) {
        console.log("Trying to find session by access_token");

        jwt.verify(accessToken, getKey, { algorithms: ['RS256'] }, (err, decoded)
=> {
            if (err) {
                return res.status(401).json({ error: 'Unauthorized' });
            }

            const { currentSession, sessionId } =
sessions.getSessionFromAccessTokenOrCreate(accessToken, res);
            req.session = currentSession;
            req.sessionId = sessionId;
            next();
        });
    } else {
        if (sessionId) {
            currentSession = sessions.get(sessionId);
        } else {
            sessionId = sessions.init(res);
        }
```

```
        req.session = currentSession;
        req.sessionId = sessionId;
        next();
    }

    onFinished(req, () => {
        const currentSession = req.session;
        const sessionId = req.sessionId;
        sessions.set(sessionId, currentSession);
    });
});

app.get('/', (req, res) => {
    if (req.session.username) {
        return res.json({
            username: req.session.username,
            logout: 'http://localhost:3000/logout'
        });
    }
    res.sendFile(path.join(__dirname + '/index.html'));
});

app.get('/logout', (req, res) => {
    sessions.destroy(req, res);
    res.redirect('/');
});

app.get('/login', (req, res) => {
    const authUrl = `https://${AUTH0_DOMAIN}/authorize?` +
                    `client_id=${CLIENT_ID}&` +
                    `redirect_uri=${encodeURIComponent(REDIRECT_URI)}&` +
                    `response_type=code&` +
                    `response_mode=query&` +
                    `scope=openid profile email nickname name`;
    res.redirect(authUrl);
});

app.listen(port, () => {
    console.log(`Example app listening on port ${port}`);
});

app.get('/callback', async (req, res) => {
    const { code } = req.query;

    if (!code) {
        return res.status(400).send('Authorization code not found');
    }

    try {
        const response = await axios.post(TOKEN_URL, {
```

```
            grant_type: 'authorization_code',
            code: code,
            redirect_uri: REDIRECT_URI,
            client_id: CLIENT_ID,
            client_secret: CLIENT_SECRET,
        });

        const { access_token } = response.data;
        req.session.access_token = access_token;

        const userProfileResponse = await
axios.get(`https://${AUTH0_DOMAIN}/userinfo`, {
            headers: {
                Authorization: `Bearer ${access_token}`,
            },
        });

        const username = userProfileResponse.data.nickname ||
userProfileResponse.data.name;

        req.session.username = username;

        res.redirect(`/?username=${username}&access_token=${access_token}`);
    } catch (error) {
        console.error('Error during token exchange:', error);
        res.status(500).send('Authentication failed');
    }
});
```

Результат:



ВИСНОВКИ

В результаті виконання лабораторної роботи було реалізовано базову систему авторизації через **Auth0** для отримання токенів доступу та автентифікації користувача. Кроки, виконані під час роботи, включають:

Розширення попередньої лабораторної роботи: було додано редірект на стандартну сторінку авторизації від SSO провайдера (Auth0). Тепер, при натисканні кнопки "Login with Auth0", користувач перенаправляється на сторінку авторизації Auth0 з необхідними параметрами:

client_id: Ідентифікатор клієнта вашого застосунку.

redirect_uri: URL, на який користувач буде перенаправлений після авторизації.

response_type=code: тип відповіді, який вказує на використання коду авторизації.

response_mode=query: спосіб передачі коду (через параметри запиту).

Обробка коду авторизації: на серверній стороні реалізовано отримання токена доступу після отримання коду авторизації. Токен використовувався для отримання профілю користувача за допомогою запиту до API Auth0.

Збереження даних сесії: було реалізовано збереження токену доступу та імені користувача в сесії браузера через sessionStorage. Це дозволяє користувачеві залишатися автентифікованим навіть після перезавантаження сторінки.

Ця лабораторна робота допомогла здобути навички роботи з **Auth0**, використовуючи стандартні протоколи для авторизації та аутентифікації користувачів, а також показала, як інтегрувати сторонні SSO рішення у веб-застосунки.

Завдання виконано успішно.