Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №5

Безпека ПЗ

Тема: Засвоювання базових навичок роботи з валідацією токенів

Виконав
студент групи ІП-11:
Панченко С. В.

Перевірив:
Курченко О. А.

Київ 2024

TABLE OF CONTENTS

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Засвоїти базові навички роботи з валідацією токенів.

2 ЗАВДАННЯ

Розширити Лабораторну роботу 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, https://[API_DOMAIN]/pem. Надати код рішення.

3 ВИКОНАННЯ

Додаємо валідацію JWT-токена. Розглянемо, що треба додати до index.ts та .env:

index.ts

```
const checkJwt = auth({
  audience: process.env.AUDIENCE,
  issuerBaseURL: `https://${process.env.AUTH0_DOMAIN}/`,
})

app.get('/api/userinfo', checkJwt, async (req, res) => {
  const token = req.headers['authorization']
  try {
    const response = await axios({
      method: 'get',
      url: `https://${process.env.AUTH0_DOMAIN}/userinfo`,
      headers: {
        'content-type': 'application/json',
        Authorization: token,
      },
    })
    res.json({ success: true, user: response.data })
  } catch (e) {
    console.log(e)
  }
})
```

.env

```
AUDIENCE=https://dev-88qrd45kn4mz6ptn.us.auth0.com/api/v2/
```

Також додано нормальне візуальне відображення userinfo на рисунку 3.1:

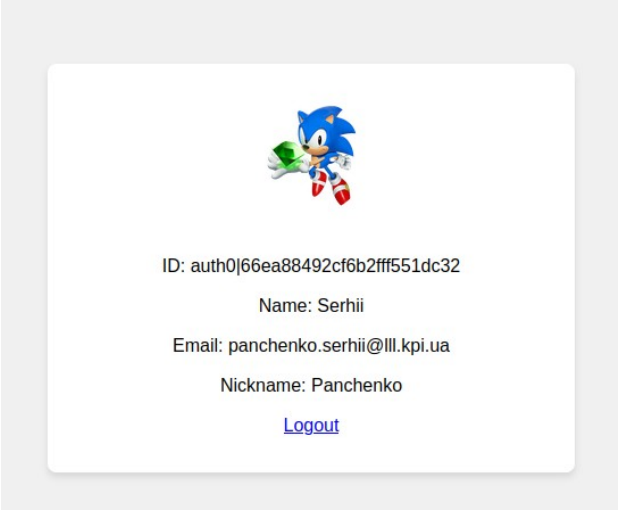

Рисунок 3.1 — Відображення userinfo

ВИСНОВОК

У результаті виконання цієї лабораторної роботи було досягнуто значного прогресу: створено новий ендпоінт для опанування базових навичок валідації токенів, суттєво вдосконалено відображення інформації, оновлено файл index.html для покращення інтерфейсу, а також оптимізовано функціонал шляхом видалення зайвих елементів. Ці комплексні зміни не лише підвищили ефективність та зручність користування проектом, але й сприяли глибшому розумінню процесів роботи з токенами та оптимізації веб-додатків.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.)

Студента групи ІП-11 4 курсу

Панченка С. В

.env.ts

```
SESSION_KEY=Authorization
AUDIENCE=https://dev-88qrd45kn4mz6ptn.us.auth0.com/api/v2/
AUTH0_DOMAIN=dev-88qrd45kn4mz6ptn.us.auth0.com
AUTH0_CLIENT_ID=zdzhN4dIDmIuptnArPhrZ4TEQAkQNGDH
AUTH0_CLIENT_SECRET=3e54llBevPi1cNu22Yr7ggExIntYXkJgs7_BZ5OTleTF1GFnT3eiOTrSlPkTLK4a
PORT=3000
```

index.ts.

```
import * as dotenv from 'dotenv';
import * as express from 'express';
import axios, { AxiosError } from 'axios';
import * as path from 'path';
import * as session from 'express-session';
import { auth } from 'express-oauth2-jwt-bearer';

dotenv.config();

const app = express();
const port = process.env.PORT ? parseInt(process.env.PORT, 10) : 3000;

declare module 'express-session' {
  interface SessionData {
    tokens?: {
      access_token: string;
      refresh_token: string;
      expires_in: number;
    };
  }
}

interface LoginBody {
  login: string;
  password: string;
}

interface RegisterBody {
  email: string;
  password: string;
  name: string;
  nickname: string;
}
```

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(
  session({
    secret: process.env.SESSION_SECRET || 'default_secret',
    resave: false,
    saveUninitialized: true,
    cookie: { secure: process.env.NODE_ENV === 'production' },
  })
);

const checkJwt = auth({
  audience: process.env.AUDIENCE,
  issuerBaseURL: `https://${process.env.AUTH0_DOMAIN}/`,
})

app.get('/api/userinfo', checkJwt, async (req, res) => {
  const token = req.headers['authorization']
  try {
    const response = await axios({
      method: 'get',
      url: `https://${process.env.AUTH0_DOMAIN}/userinfo`,
      headers: {
        'content-type': 'application/json',
        Authorization: token,
      },
    })
    res.json({ success: true, user: response.data })
  } catch (e) {
    console.log(e)
  }
})

const expiration_period = 5 * 60 * 1000;

const refreshToken: express.RequestHandler = async (req, res, next) => {
  const tokens = req.session.tokens;

  if (!tokens) return next();

  const isTokenExpired = (expiresIn: number, bufferPeriod: number) =>
    Date.now() > expiresIn - bufferPeriod;

  const logExpirationDate = (expiresIn: number) => {
```

```
          const expirationDate = new Date(expiresIn);
          console.log(`Expiration time point: $
{expirationDate.toLocaleString()}`);
        };

        const refreshAuthToken = async (refreshToken: string) => {
          try {
            const response = await axios.post<{ access_token: string;
expires_in: number }>(
              `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
              new URLSearchParams({
                grant_type: 'refresh_token',
                client_id: process.env.AUTH0_CLIENT_ID!,
                client_secret: process.env.AUTH0_CLIENT_SECRET!,
                refresh_token: refreshToken,
              }),
              { headers: { 'content-type': 'application/x-www-form-urlencoded' }
}
            );

            return {
              access_token: response.data.access_token,
              expires_in: Date.now() + response.data.expires_in * 1000,
            };
          } catch (error) {
            throw new Error(
              (error as AxiosError).response?.data
                ? JSON.stringify((error as AxiosError).response?.data)
                : (error as Error).message
            );
          }
        };

        if (tokens.expires_in) {
          logExpirationDate(tokens.expires_in);
        }

        if (isTokenExpired(tokens.expires_in, expiration_period)) {
          try {
            const newTokens = await refreshAuthToken(tokens.refresh_token);
            req.session.tokens = {
              ...newTokens,
              refresh_token: tokens.refresh_token,
            };
```

```
          } catch (error) {
            console.error('Error refreshing token:', error.message);
            return res.status(401).json({ error: 'Failed to refresh token' });
          }
        }

        next();
      };

      app.use(refreshToken);

      async function handleHomeRoute(req: express.Request, res:
express.Response) {
        const session = req.session;

        if (session.tokens) {
          try {
            const userInfo = await getUserInfo(session.tokens.access_token);
            res.json({
              user: userInfo,
              logout: '/logout',
            });
          } catch (error) {
            await handleUserInfoError(req, res, error);
          }
        } else {
          res.sendFile(path.join(__dirname, 'index.html'));
        }
      }

      async function handleUserInfoError(req: express.Request, res:
express.Response, error: unknown) {
        console.error('Error:', (error as AxiosError).response?.data || (error
as Error).message);
        await new Promise<void>((resolve) => req.session.destroy(() =>
resolve()));
        res.status(500).json({ error: 'An error occurred while fetching user
info' });
      }

      function handleLogout(req: express.Request, res: express.Response) {
        req.session.destroy(() => {
          res.redirect('/');
        });
```

```
}

async function getUserInfo(accessToken: string) {
  const response = await axios.get(
    `https://${process.env.AUTH0_DOMAIN}/userinfo`,
    {
      headers: {
        Authorization: `Bearer ${accessToken}`,
      },
    }
  );
  return response.data;
}

app.get('/', handleHomeRoute);
app.get('/logout', handleLogout);

app.post('/api/login', async (req: express.Request<{}, {}, LoginBody>,
res) => {
  try {
    const { login, password } = req.body;
    const response = await axios.post<{ access_token: string;
refresh_token: string; expires_in: number }>(
      `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
      new URLSearchParams({
        grant_type: 'password',
        username: login,
        password: password,
        client_id: process.env.AUTH0_CLIENT_ID!,
        client_secret: process.env.AUTH0_CLIENT_SECRET!,
        audience: `https://${process.env.AUTH0_DOMAIN}/api/v2/`,
        scope: 'offline_access openid profile email',
      }),
      {
        headers: { 'content-type': 'application/x-www-form-urlencoded' },
      }
    );

    req.session.tokens = {
      access_token: response.data.access_token,
      refresh_token: response.data.refresh_token,
      expires_in: Date.now() + response.data.expires_in * 1000,
    };
    res.json({ success: true, token: response.data.access_token });
```

```
    } catch (error) {
      console.error('Login failed:', (error as AxiosError).response?.data ||
(error as Error).message);
      res.status(401).send('Login failed');
    }
  });

  async function getAuth0Token() {
    const response = await axios.post<{ access_token: string }>(
      `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
      new URLSearchParams({
        grant_type: 'client_credentials',
        client_id: process.env.AUTH0_CLIENT_ID!,
        client_secret: process.env.AUTH0_CLIENT_SECRET!,
        audience: `https://${process.env.AUTH0_DOMAIN}/api/v2/`,
      }),
      {
        headers: { 'content-type': 'application/x-www-form-urlencoded' },
      }
    );
    return response.data.access_token;
  }

  async function registerUser(req: express.Request<{}, {}, RegisterBody>,
res: express.Response) {
    const { email, password, name, nickname } = req.body;

    try {
      const authToken = await getAuth0Token();
      const user = await createAuth0User(authToken, { email, password, name,
nickname });

      res.status(201).json({
        success: true,
        userId: user,
        login: '/',
      });
    } catch (error) {
      handleError(res, error as Error | AxiosError);
    }
  }

  async function createAuth0User(token: string, userData: Omit<RegisterBody,
'password'> & { password: string }) {
```

```
    const response = await axios.post(
      `https://${process.env.AUTH0_DOMAIN}/api/v2/users`,
      {
        ...userData,
        connection: 'Username-Password-Authentication',
        verify_email: true,
        picture:
'https://static.wikia.nocookie.net/sonic-x9874/images/6/6f/Sonic_135.png/
revision/latest/thumbnail/width/360/height/360?cb=20160226182328',
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
          'content-type': 'application/json',
        },
      }
    );
    return response.data;
  }

  function handleError(res: express.Response, error: Error | AxiosError) {
    console.error('Registration failed:', (error as
AxiosError).response?.data || error.message);
    res.status(400).json({
      success: false,
      error: (error as AxiosError).response?.data || error.message,
    });
  }

  app.post('/api/register', registerUser);

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

```
/* CSS styles */
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    background-color: #f0f0f0;
}

.container {
    background-color: white;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    max-width: 400px;
    width: 100%;
}

h1 {
    text-align: center;
    margin-bottom: 1.5rem;
}

.form {
    display: flex;
    flex-direction: column;
}

.form-field {
    margin-bottom: 1rem;
}

.form-field input {
    width: 100%;
    padding: 0.5rem;
    border: 1px solid #ccc;
    border-radius: 4px;
}

.form-submit {
    background-color: #3a3a3a;
    color: white;
```

```
                    border: none;
                    padding: 0.75rem;
                    border-radius: 4px;
                    cursor: pointer;
                    font-weight: bold;
                }

                .error-message {
                    color: #d32f2f;
                    text-align: center;
                    margin-bottom: 1rem;
                    display: none;
                }

                .user-info {
                    text-align: center;
                }

                .user-info img {
                    border-radius: 50%;
                    margin-bottom: 1rem;
                }

                #logout {
                    display: none;
                    text-align: center;
                    margin-top: 1rem;
                }
            </style>
        </head>
        <body>
            <div class="container">
                <div id="auth-forms">
                    <h1 id="login-header">Login</h1>
                    <p id="login-error-msg" class="error-message">Invalid username
and/or password</p>
                    <form id="login-form" class="form">
                        <div class="form-field">
                            <input type="text" name="login" placeholder="Username"
required>
                        </div>
                        <div class="form-field">
                            <input type="password" name="password"
placeholder="Password" required>
```

```
                    </div>
                    <button type="submit" class="form-submit">Login</button>
                </form>

                <h1 id="register-header">Register</h1>
                <p id="register-error-msg" class="error-message">Registration
failed</p>
                <form id="register-form" class="form">
                    <div class="form-field">
                        <input type="email" name="email" placeholder="Email"
required>
                    </div>
                    <div class="form-field">
                        <input type="text" name="name" placeholder="Name"
required>
                    </div>
                    <div class="form-field">
                        <input type="text" name="nickname"
placeholder="Nickname" required>
                    </div>
                    <div class="form-field">
                        <input type="password" name="password"
placeholder="Password" required>
                    </div>
                    <button type="submit"
class="form-submit">Register</button>
                </form>
            </div>

            <div id="user-info" class="user-info"></div>
            <a href="#" id="logout">Logout</a>
        </div>

        <script>
            // JavaScript code
            const loginForm = document.getElementById('login-form');
            const registerForm = document.getElementById('register-form');
            const loginErrorMsg = document.getElementById('login-error-msg');
            const registerErrorMsg = document.getElementById('register-error-
msg');
            const logoutLink = document.getElementById('logout');
            const authForms = document.getElementById('auth-forms');
            const userInfoDiv = document.getElementById('user-info');
```

```
            function showAuthForms() {
                authForms.style.display = 'block';
                userInfoDiv.style.display = 'none';
                logoutLink.style.display = 'none';
            }

            function showUserInfo(user) {
                authForms.style.display = 'none';
                userInfoDiv.style.display = 'block';
                logoutLink.style.display = 'block';

                userInfoDiv.innerHTML = `
                    <img src="${user.picture}" alt="Profile pic" width="100"
height="100">
                    <p>ID: ${user.sub}</p>
                    <p>Name: ${user.name}</p>
                    <p>Email: ${user.email}</p>
                    <p>Nickname: ${user.nickname}</p>
                `;
            }

            function handleLogin(event) {
                event.preventDefault();
                const login = loginForm.login.value;
                const password = loginForm.password.value;

                axios.post('/api/login', { login, password })
                    .then(response => {
                        sessionStorage.setItem('session',
JSON.stringify(response.data));
                        location.reload();
                    })
                    .catch(() => {
                        loginErrorMsg.style.display = 'block';
                    });
            }

            function handleRegister(event) {
                event.preventDefault();
                const formData = new FormData(registerForm);
                const data = Object.fromEntries(formData);

                axios.post('/api/register', data)
                    .then(response => {
```

```
                        sessionStorage.setItem('session',
JSON.stringify(response.data));
                        location.reload();
                    })
                    .catch(() => {
                        registerErrorMsg.style.display = 'block';
                    });
            }

            function handleLogout(event) {
                event.preventDefault();
                sessionStorage.removeItem('session');
                location.reload();
            }

            function init() {
                const session = sessionStorage.getItem('session');
                const token = session ? JSON.parse(session).token : null;

                if (token) {
                    axios.get('/api/userinfo', {
                        headers: { Authorization: `Bearer ${token}` }
                    })
                    .then(response => {
                        const { user } = response.data;
                        if (user) {
                            showUserInfo(user);
                        } else {
                            showAuthForms();
                        }
                    })
                    .catch(() => {
                        showAuthForms();
                    });
                } else {
                    showAuthForms();
                }
            }

            loginForm.addEventListener('submit', handleLogin);
            registerForm.addEventListener('submit', handleRegister);
            logoutLink.addEventListener('click', handleLogout);

            init();
```

```
    </script>
</body>
</html>
```