

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

ЗВІТ

Про виконання лабораторної роботи №4
З дисципліни “Безпека програмного забезпечення”
На тему “Засвоювання базових навичок OAuth2 авторизаційного
протокола”

Виконали:

Студенти групи ІП-15

Мешков А. І.

Перевірила:

пос. Соколовський В. В.

Київ 2024

ЛАБОРАТОРНА РОБОТА №4

Завдання:

Використовуючи наведені налаштування з лабораторної роботи 2 - 3 та
приведених запитів модифікувати аплікейшен

https://github.com/Kreolwolf1/auth_examples/tree/main/token_auth

Використовуючи перевірку юзера та отримання токена з auth0 (**password grant type**)

Надати код модифікованного аплікейшена.

ХІД РОБОТИ

1. Через проблему авторизацію було створено нове API. Буде використано це налаштування.

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>

      <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="login" 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 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();
        loginErrorMsg.remove();
      }
    });
  }

```

```

        loginHeader.remove();

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

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");

logoutLink.addEventListener("click", (e) => {
    e.preventDefault();
    sessionStorage.removeItem('session');
    location.reload();
});

loginButton.addEventListener("click", (e) => {
    e.preventDefault();
    const login = loginForm.login.value;
    const password = loginForm.password.value;

    axios({
        method: 'post',
        url: '/api/login',
        data: {
            login,
            password
        }
    }).then((response) => {
        const { username } = response.data;
        sessionStorage.setItem('session', JSON.stringify(response.data));
        location.reload();
    }).catch((response) => {
        loginErrorMsg.style.opacity = 1;
    });
})
</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 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`;

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){
    console.log("hi", 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");
    const { currentSession, sessionId } =
sessions.getSessionFromAccessTokenOrCreate(accessToken, res);
    req.session = currentSession;
    req.sessionId = sessionId;
  } else {

    if (sessionId) {
      currentSession = sessions.get(sessionId);
    } else {
      sessionId = sessions.init(res);
    }
  }
}

```



```

    }

    req.session = currentSession;
    req.sessionId = sessionId;
  }

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

  next();
});

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.post('/api/login', async (req, res) => {
  const { login, password } = req.body;

  try {
    const response = await axios.post(TOKEN_URL, {
      grant_type: 'password',
      username: login,
      password: password,
      audience: AUDIENCE,
      client_id: CLIENT_ID,
      client_secret: CLIENT_SECRET,
    });

    const { access_token } = response.data;

    req.session.username = login;
    req.session.access_token = access_token;

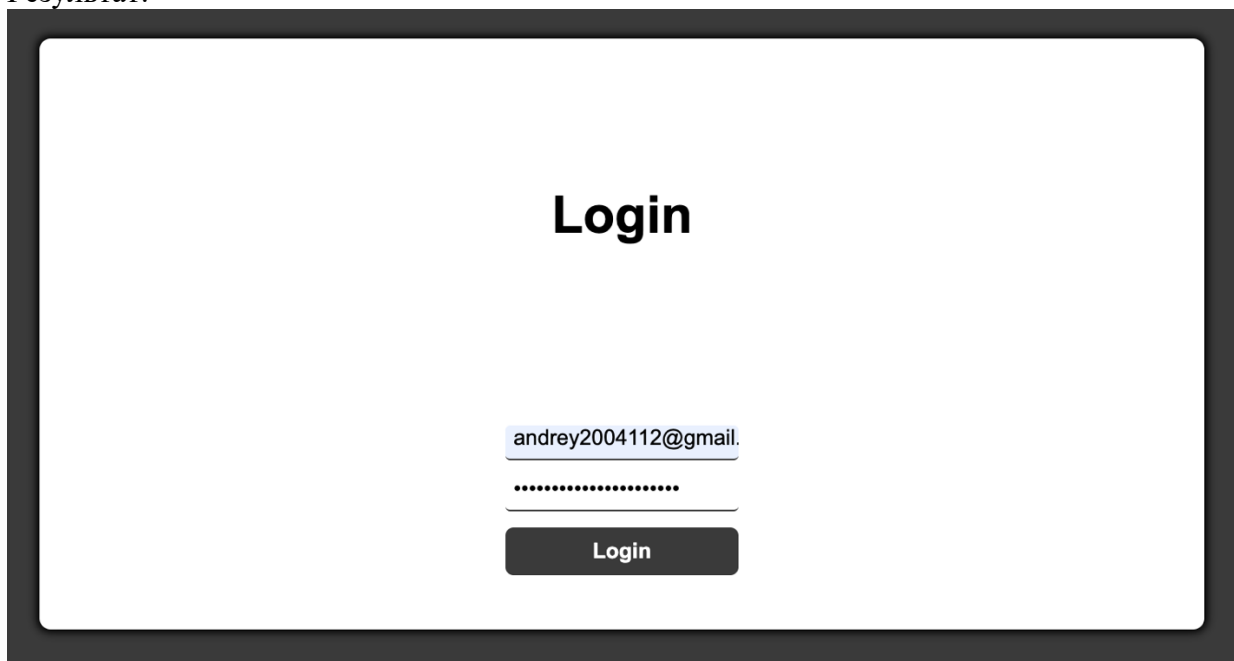
    return res.json({

```

```
        message: 'Login successful',
        access_token: access_token,
      });
    } catch (error) {
      console.error('Error during Auth0 login:', error.response?.data ||
error.message);
      res.status(401).json({
        error: 'Invalid login credentials or unauthorized.'
      });
    }
  }
});

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

Результат:



Login

andrey2004112@gmail.

.....

Login

[Logout](#)

Hello andrey2004112@gmail.com

Application

Manifest

Service workers

Storage

Local storage

Session storage

http://localhost:3000

IndexedDB

Cookies

Private state tokens

Interest groups

Shared storage

Cache storage

Storage buckets

Console

Coverage

What's new

AI assistance

Issues

Autofill

Elements

Console

Network

Sources

Performance

Memory

Application

Lighthouse

Recorder

Performance insights

AdBlock

Redux

Filter

http://localhost:3000

Origin http://localhost:3000

Key	Value
session	("message":"Login successful","access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IkpFTY...

(message: "Login successful",...)

access_token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IkpFTY..."

message: "Login successful"

ВИСНОВКИ

В результаті виконання лабораторної роботи було реалізовано авторизацію користувачів за допомогою протоколу OAuth2 через Auth0. Зокрема, була здійснена інтеграція з використанням "password grant type" для отримання токена доступу. Використовуючи цей токен, система авторизації дозволяє доступ до захищених ресурсів лише після успішної аутентифікації користувача.

Ключовими етапами були:

Реалізація API для аутентифікації через Auth0 за допомогою password grant type, що дозволяє користувачам входити в систему, вводячи свої логін та пароль.

Модифікація існуючого додатку для підтримки аутентифікації через токен доступу, а також обробки помилок при введенні неправильних даних.

Розробка механізму зберігання сесій, що включає зберігання токена доступу в сесії на стороні сервера, а також можливість виходу з системи з очищенням сесії.

В результаті виконання лабораторної роботи вдалося отримати практичний досвід у налаштуванні аутентифікації за допомогою OAuth2, використовуючи популярний сервіс Auth0 для забезпечення безпеки програмного забезпечення. Цей досвід є важливим кроком у розумінні принципів безпеки в веб-додатках та їх інтеграції з сучасними авторизаційними протоколами.