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

Звіт

з лабораторної роботи №5 з дисципліни
«Системи безпеки програм і даних»

«Засвоювання базових навичок роботи з валідацією токенів»

Виконав(ла) *ІП-11 Сідак Кирил Ігорович*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Іваніщев Б. В.*
(прізвище, ім'я, по батькові)

Київ 2024

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – засвоювання базових навичок роботи з валідацією токенів.

2 ЗАВДАННЯ

2.1 Основне 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, або за формулою https://[API_DOMAIN]/pem

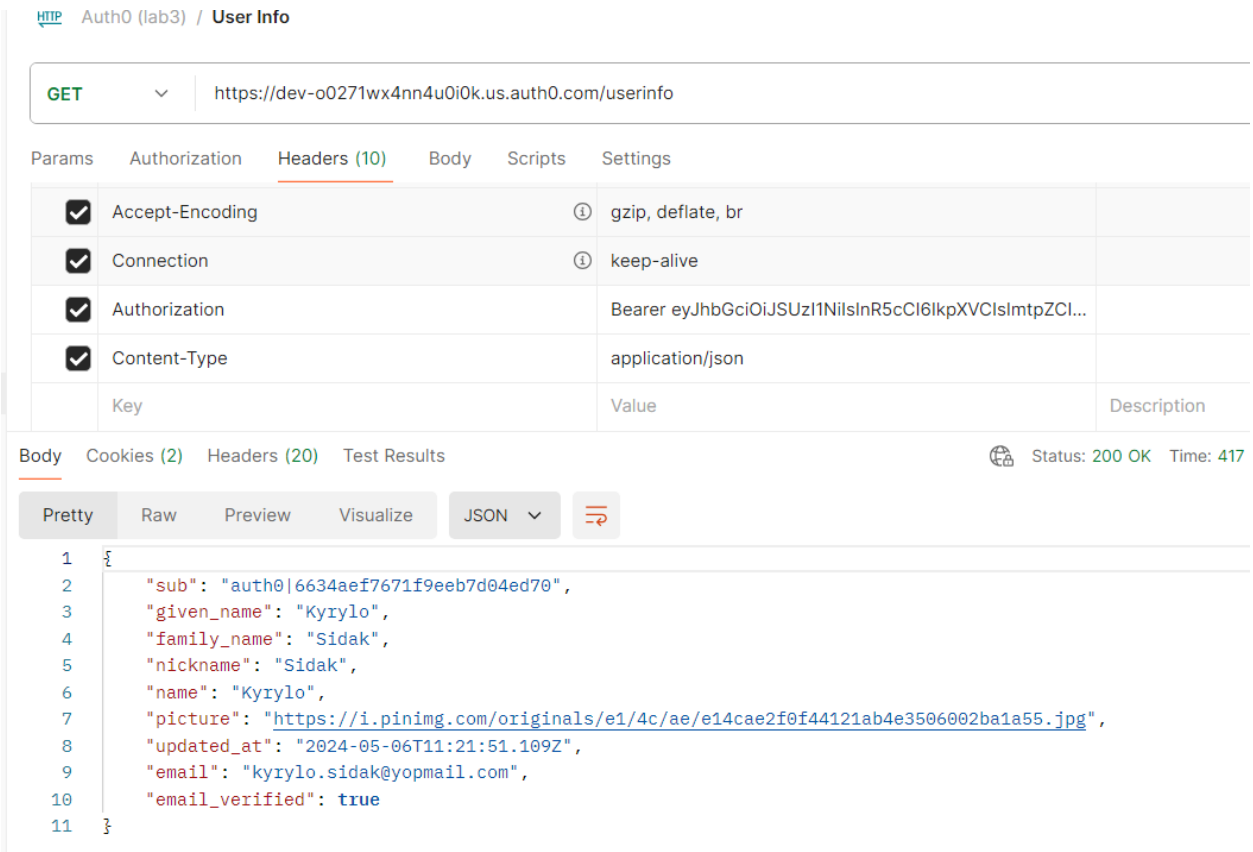
Надати код рішення.

3 ВИКОНАННЯ ОСНОВНОГО ЗАВДАННЯ

В попередній лабораторній роботі було мною допущено помилку, адже використання сесій вже не потрібно, тому бібілотеку express-session було видалено. Також для для відображення інформації і перевірки jwt було створено новий запит:

```
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)
  }
})
```

Роботу даного запиту було продемонстровано та протестована в лабораторній роботі №3:



Оновлений код index.js тепер виглядає так:

```
require('dotenv').config()
const express = require('express')
const axios = require('axios')
const path = require('path')
const { auth } = require('express-oauth2-jwt-bearer')

const app = express()
const port = process.env.PORT || 3000

app.use(express.json())
app.use(express.urlencoded({ extended: true }))

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

app.get('/', async (req, res) => {
  res.sendFile(path.join(__dirname, 'index.html'))
})

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)
  }
})

app.post('/api/login', async (req, res) => {
  try {
    const { login, password } = req.body
    const response = await axios({
      method: 'post',
      url: `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
      headers: { 'content-type': 'application/x-www-form-urlencoded' },
      data: 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',
      }),
    })

    res.json({ success: true, token: response.data.access_token })
  } catch (error) {
    console.error('Login failed:', error.response?.data || error.message)
    res.status(401).send('Login failed')
  }
})

app.post('/api/register', async (req, res) => {
  const { email, password, name, nickname } = req.body

  try {
    // Получение токена
    const authData = await axios.post(
      `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' },
      }
    )

    // Запрос регистрации
    const userResponse = await axios.post(
      `https://${process.env.AUTH0_DOMAIN}/api/v2/users`,
      {
        email: email,
        password: password,
        connection: 'Username-Password-Authentication',
        verify_email: true,
        name: name,
        nickname: nickname,
        picture:
          'https://i.pinimg.com/originals/e1/4c/ae/e14cae2f0f44121ab4e3506002ba1a
55.jpg',
      },
      {
        headers: {
          Authorization: `Bearer ${authData.data.access_token}`,
          'content-type': 'application/json',
        },
```

```
      }
    )
    res.status(201).json({
      success: true,
      userId: userResponse.data,
      login: '/',
    })
  } catch (error) {
    console.error('Registration failed:', error.response?.data || error.message)
    res.status(400).json({
      success: false,
      error: error.response?.data,
    })
  }
})

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

Вивід інформації більше не працює в вигляді json, тепер все відображається в 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>
    <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,
  #register-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;
```

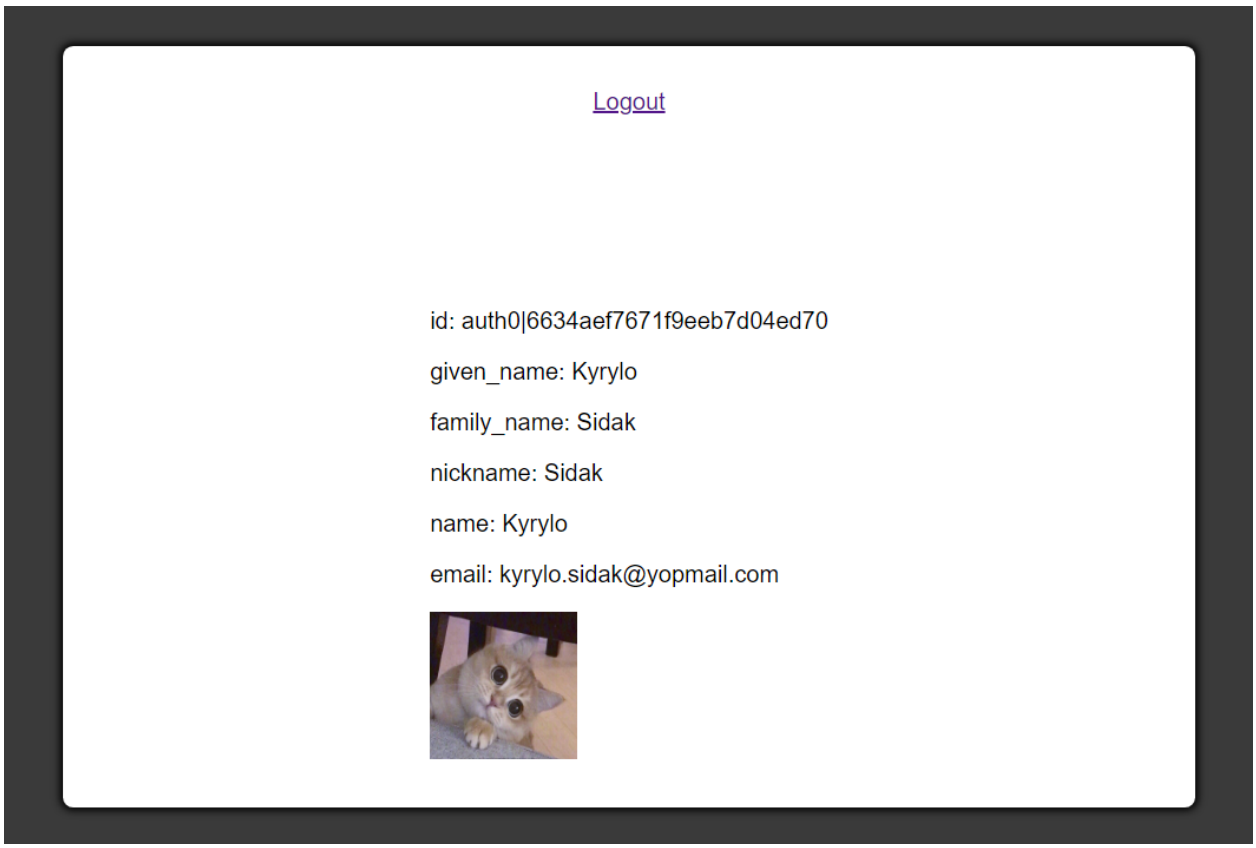
```
      border-radius: 3px;
      outline: none;
      padding: 0px 0px 5px 5px;
    }

    #login-form-submit,
    #register-form-submit {
      width: 100%;
      padding: 7px;
      border: none;
      border-radius: 5px;
      color: white;
      font-weight: bold;
      background-color: #3a3a3a;
      cursor: pointer;
      outline: none;
    }

    #login-form,
    #register-form,
    #login-header,
    #register-header {
      display: none;
    }
  </style>
</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"
        required />
      <input
        type="password"
        name="password"
```

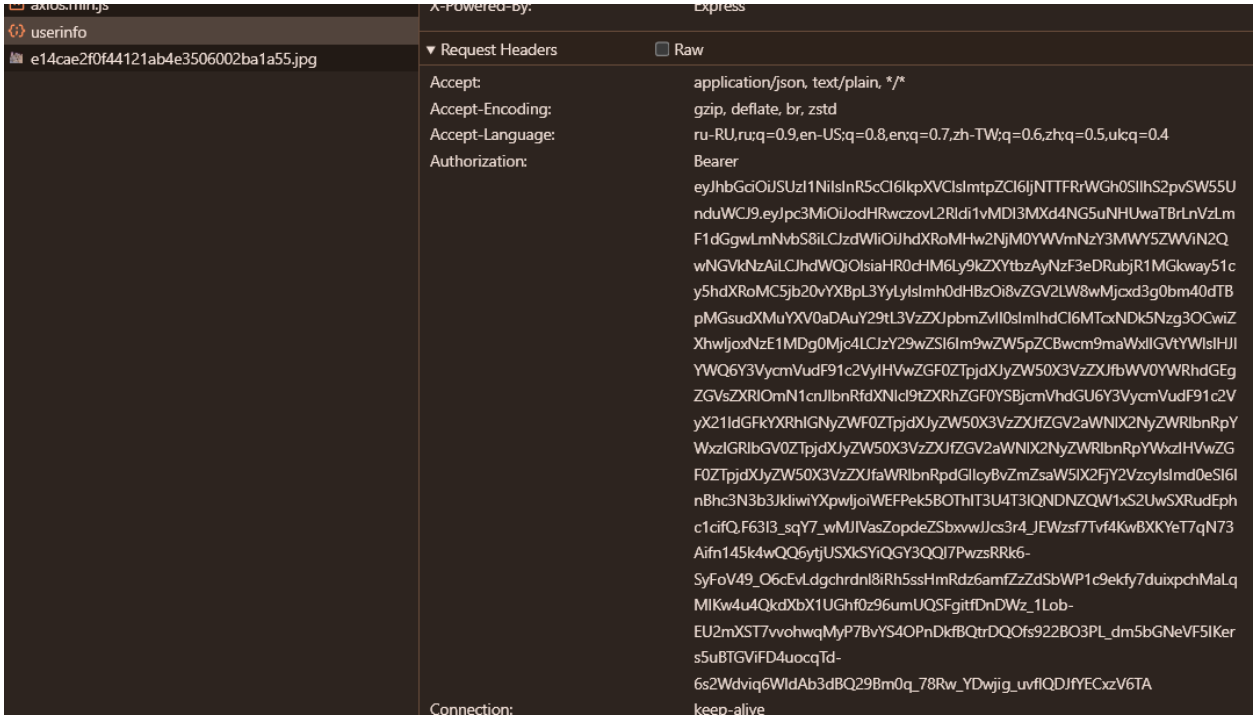
```
        id="password-field"
        class="login-form-field"
        placeholder="Password"
        required />
      <input type="submit" value="Login" id="login-form-submit" />
    </form>

    <h1 id="register-header">Register</h1>

    <form id="register-form" action="/api/register" method="post">
      <input
        type="email"
        name="email"
        id="email"
        class="login-form-field"
        placeholder="Email"
        required />
      <input
        type="text"
        name="name"
        id="name"
        class="login-form-field"
        placeholder="Name"
        required />
      <input
        type="text"
        name="nickname"
        id="nickname"
        class="login-form-field"
        placeholder="Nickname"
        required />

      <input
        type="password"
        name="password"
        id="password-field"
        class="login-form-field"
        placeholder="Password"
        required />
      <input type="submit" value="Regiser" id="register-form-submit" />
    </form>
  </main>
</body>

<script>
  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 RegisterForm = document.getElementById('register-form')
```

```
    const RegisterButton = document.getElementById('register-form-submit')
    const RegisterErrorMsg = document.getElementById('register-error-msg')

    const mainHolder = document.getElementById('main-holder')
    const loginHeader = document.getElementById('login-header')
    const registerHeader = document.getElementById('register-header')

    const session = sessionStorage.getItem('session')

    let token

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

    if (!token) {
      loginForm.style.display = 'grid'
      RegisterForm.style.display = 'grid'
      loginHeader.style.display = 'block'
      registerHeader.style.display = 'block'
    }
    if (token) {
      axios
        .get('/api/userinfo', {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        })
        .then((response) => {
          const { user } = response.data

          if (user) {
            loginForm.remove()
            loginErrorMsg.remove()
            loginHeader.remove()
            registerHeader.remove()
            RegisterForm.remove()

            const div = document.createElement('div')
            div.innerHTML = `
                <p>id: ${user.sub}</p>
                <p>given_name: ${user.given_name}</p>
                <p>family_name: ${user.family_name}</p>
                <p>nickname: ${user.nickname}</p>
                <p>name: ${user.name}</p>
                <p>email: ${user.email}</p>
                <img src='${user.picture}' alt='Profile pic' width="100"
height="100"/>
            `
            mainHolder.appendChild(div)
```

```
            logoutLink.style.opacity = 1
          }
        })
    }

    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>
```

Якщо токен існує, то на сервер надсилається запит для отримання інформації про користувача. Протестуємо авторизацію:





Авторизація працює. Перевірка jwt здійснена за офіційною документацією та додана для запиту userinfo.

4 ВИСНОВОК

В результаті виконання цієї лабораторної роботи, було створено новий ендпоінт для засвоєння базових навичок роботи з валідацією токенів.

Також при виконанні було виявлено та вирішено проблеми попередньої роботи. А саме: поліпшено відображення інформації, змінено index.html, та прибрано зайвий функціонал.