Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №1

з дисципліни

«Безпека програмного забезпечення»

тема: «Роздивитись основні методи авторизації»

Виконав:

студент групи ІТ-03

Філоненко Владислав

Київ 2023

**Завдання**

Викачати репозиторій з лекціями <https://github.com/Kreolwolf1/> auth\_examples Запустити кожен з 3 аплікейшенів та зробити скріншити запитів до серверу.

Для отримання додаткового балу: модифікувати token\_auth аплікейшен змінивши токен на JWT.

**Хід роботи**

**Код basic\_auth**

const express = require('express');

const app = express();

const port = 3001;

app.use((req, res, next) => {

    console.log('\n=======================================================\n');

    const authorizationHeader = req.get('Authorization');

    console.log('authorizationHeader', authorizationHeader);

    if (!authorizationHeader) {

        res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');

        res.status(401);

        res.send('Unauthorized');

        return;

    }

    const authorizationBase64Part = authorizationHeader.split(' ')[1];

    const decodedAuthorizationHeader = Buffer.from(authorizationBase64Part, 'base64').toString(

        'utf-8',

    );

    console.log('decodedAuthorizationHeader', decodedAuthorizationHeader);

    const login = decodedAuthorizationHeader.split(':')[0];

    const password = decodedAuthorizationHeader.split(':')[1];

    console.log('Login/Password', login, password);

    if (login == 'DateArt' && password == '2408') {

        req.login = login;

        return next();

    }

    res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');

    res.status(401);

    res.send('Unauthorized');

});

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

    res.send(`Hello ${req.login}`);

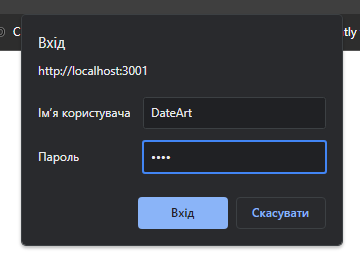
});

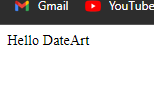
app.listen(port, () => {

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

});

**Результат basic\_auth**





**Код forms\_auth**

const uuid = require('uuid');

const express = require('express');

const cookieParser = require('cookie-parser');

const onFinished = require('on-finished');

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

const path = require('path');

const port = 3000;

const fs = require('fs');

const app = express();

app.use(bodyParser.json());

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

app.use(cookieParser());

const SESSION\_KEY = 'session';

class Session {

  #sessions = {};

  constructor() {

    try {

      this.#sessions = fs.readFileSync('./sessions.json', 'utf8');

      this.#sessions = JSON.parse(this.#sessions.trim());

      console.log(this.#sessions);

    } 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();

    res.set('Set-Cookie', `${SESSION\_KEY}=${sessionId}; HttpOnly`);

    this.set(sessionId);

    return sessionId;

  }

  destroy(req, res) {

    const sessionId = req.sessionId;

    delete this.#sessions[sessionId];

    this.#storeSessions();

    res.set('Set-Cookie', `${SESSION\_KEY}=; HttpOnly`);

  }

}

const sessions = new Session();

app.use((req, res, next) => {

  let currentSession = {};

  let sessionId;

  if (req.cookies[SESSION\_KEY]) {

    sessionId = req.cookies[SESSION\_KEY];

    currentSession = sessions.get(sessionId);

    if (!currentSession) {

      currentSession = {};

      sessionId = sessions.init(res);

    }

  } else {

    sessionId = sessions.init(res);

  }

  req.session = currentSession;

  req.sessionId = sessionId;

  onFinished(req, () => {

    const currentSession = req.session;

    const sessionId = req.sessionId;

    sessions.set(sessionId, currentSession);

  });

  next();

});

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

  console.log(req.session);

  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('/');

});

const users = [

  {

    login: 'Login',

    password: 'Password',

    username: 'Username',

  },

  {

    login: 'Login1',

    password: 'Password1',

    username: 'Username1',

  },

];

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

  const { login, password } = req.body;

  const user = users.find((user) => {

    if (user.login == login && user.password == password) {

      return true;

    }

    return false;

  });

  if (user) {

    req.session.username = user.username;

    req.session.login = user.login;

    res.json({ username: login });

  }

  res.status(401).send();

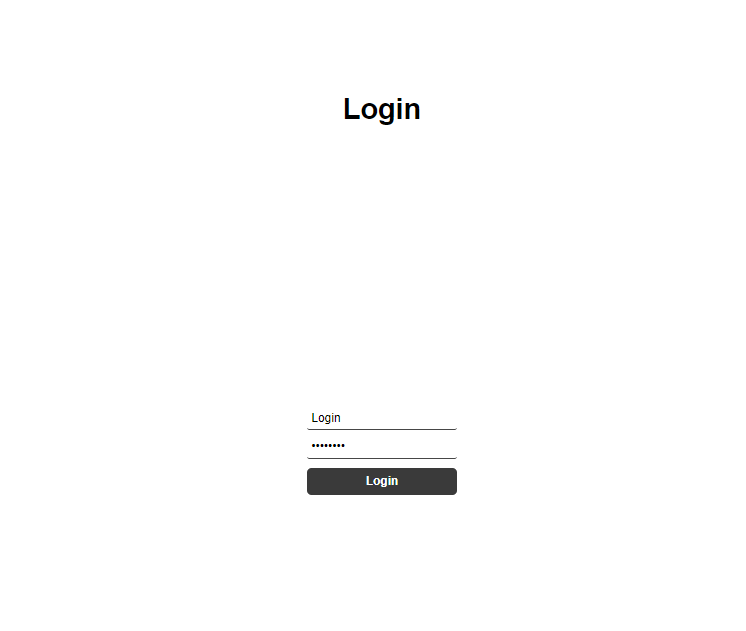
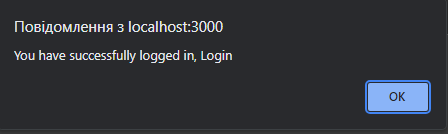
});

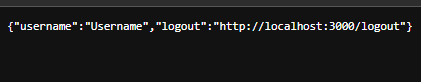
app.listen(port, () => {

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

});

**Результат forms\_auth**



**Код token\_auth**

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 app = express();

app.use(bodyParser.json());

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

const SESSION\_KEY = 'Authorization';

class Session {

    #sessions = {};

    constructor() {

        try {

            this.#sessions = fs.readFileSync('./sessions.json', 'utf8');

            this.#sessions = JSON.parse(this.#sessions.trim());

            console.log(this.#sessions);

        } 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) {

        const sessionId = req.sessionId;

        delete this.#sessions[sessionId];

        this.#storeSessions();

    }

}

const sessions = new Session();

app.use((req, res, next) => {

    let currentSession = {};

    let sessionId = req.get(SESSION\_KEY);

    if (sessionId) {

        currentSession = sessions.get(sessionId);

        if (!currentSession) {

            currentSession = {};

            sessionId = sessions.init(res);

        }

    } else {

        sessionId = sessions.init(res);

    }

    req.session = currentSession;

    req.sessionId = sessionId;

    onFinished(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('/');

});

const users = [

    {

        login: 'Login',

        password: 'Password',

        username: 'Username',

    },

    {

        login: 'Login1',

        password: 'Password1',

        username: 'Username1',

    },

];

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

    const { login, password } = req.body;

    const user = users.find((user) => {

        if (user.login == login && user.password == password) {

            return true;

        }

        return false;

    });

    if (user) {

        req.session.username = user.username;

        req.session.login = user.login;

        res.json({ token: req.sessionId });

    }

    res.status(401).send();

});

app.listen(port, () => {

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

});

**Результат token\_auth**



**Додаткове завдання**

**Код jwt\_auth**

const uuid = require('uuid');

const express = require('express');

const jwt = require('jsonwebtoken');

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

const path = require('path');

const port = 3000;

const fs = require('fs');

const app = express();

app.use(bodyParser.json());

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

const SECRET\_KEY = 'SECRET\_KEY';

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

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

    if (token) {

        const decoded = jwt.verify(token, SECRET\_KEY);

        return res.json({

            username: decoded.username,

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

        });

    }

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

});

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

    sessions.destroy(req, res);

    res.redirect('/');

});

const users = [

    {

        login: 'Login',

        password: 'Password',

        username: 'Username',

    },

    {

        login: 'Login1',

        password: 'Password1',

        username: 'Username1',

    },

];

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

    const { login, password } = req.body;

    const user = users.find((user) => {

        if (user.login == login && user.password == password) {

            return true;

        }

        return false;

    });

    if (user) {

        const token = jwt.sign({ username: user.username, login: user.login }, SECRET\_KEY);

        res.json({ token });

    }

    res.status(401).send();

});

app.listen(port, () => {

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

});

**Результат jwt\_auth**



**Висновок**

В ході лаболаторної роботи, були розглянуто 3 різні способи авторизіції і в додатковому завдянні було використано JWT токен.