

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра Обчислювальної Техніки

Лабораторна робота №5
з дисципліни "Безпека програмного забезпечення"
Тема: "Засвоювання базових навичок роботи з валідацією токенів"

Виконав:

студент групи ІІІ-93

Домінський В.О.

Київ 2023

Зміст

Завдання:	3
Виконання:	3
Висновок:	8
Посилання:	8

Завдання:

1. Розширити Лабораторну роботу 4 перевіркою сигнатури 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](https://[API_DOMAIN]/pem)

Виконання:

У даній лабораторній роботі треба дописати функціонал верифікації JWT токена та, якщо використовується асиметричне шифрування, отримати public ключ:

Почнемо з перевірки сигнатури JWT токена

```
app.use((req, res, next) => {
  const authHeader = req.get(SESSION_KEY)
  const token = authHeader && authHeader.split(' ')[1]
  if (token == null) return next()

  try {
    const user = jwt.verify(token, config.publicKey, config.jwtVerifyOptions);
    // const user = jwt.decode(token);

    req.userID = user.sub;
  } catch (error) {
    console.log({jwtError: error.message});
    return res.status(401).send()
  }

  next()
});
```

Тепер замість простого декодування Ми використовуємо верифікацію, де першим аргументом йде сам token, другим – public key (про це дещо пізніше), а останнім – запит:

```
module.exports = {
  ...config,
  jwtVerifyOptions: {
    issuer: `https://${config.domain}/`,
    audience: config.audience,
    algorithms: ['RS256'],
  }
}
```

Перейдемо до public key:

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

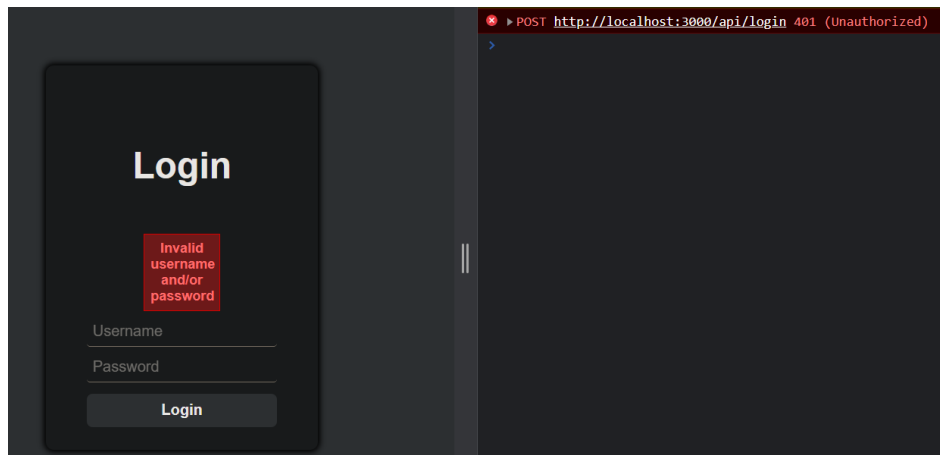
  const publicKey = await getPublicKey();
  const appTokenInfo = await getAppToken();
  config.appToken = appTokenInfo.access_token;
  config.publicKey = publicKey;
})
```

На самому старті Нашого застосунку Ми записуємо ключ до конфігу, а перед цим – отримуємо його за допомогою функції getPublicKey:

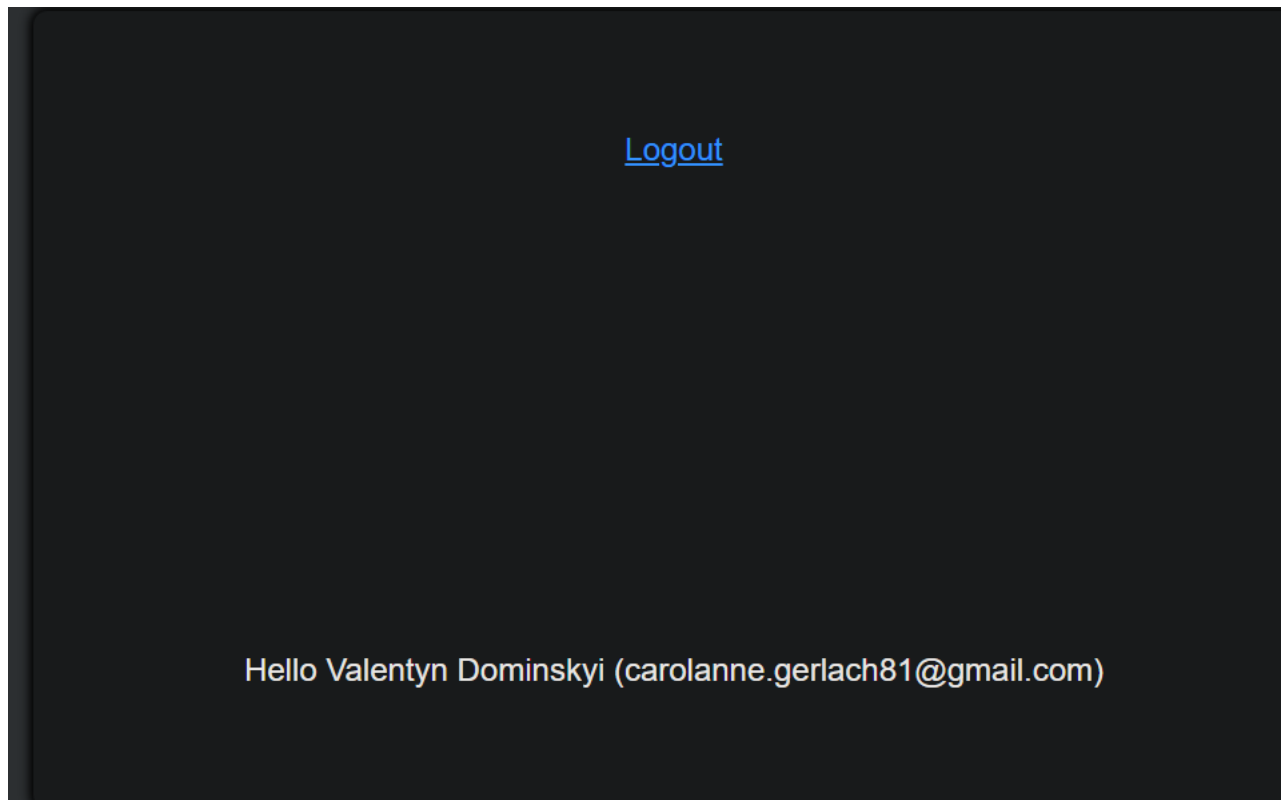
```
const pemOptions = {
  method: 'GET',
  url: `https://${config.domain}/pem`,
}

const getPublicKey = async () => {
  try {
    const body = await axios.request(pemOptions);
    return body.data;
  } catch (error) {
    console.log(error);
    return null;
  }
}
```

Давайте запустимо Наш застосунок та спробуємо ввести неправильні дані:



Як видно з консолі – Нам видало помилку 401. Тепер увійдемо в Нашого користувача:



Усе пройшло успішно!

Проте давайте розглянемо, чи дійсно працює Наша перевірка JWT токену.
При вдалому вході Ми можемо побачити значення Нашого токену:

Якщо перейдемо на сайт jwt.io та спробуємо його розшифрувати, то отримаємо ось такі значення:

Пропоную зараз замінити один символ у частині з Data і отримуємо результат:

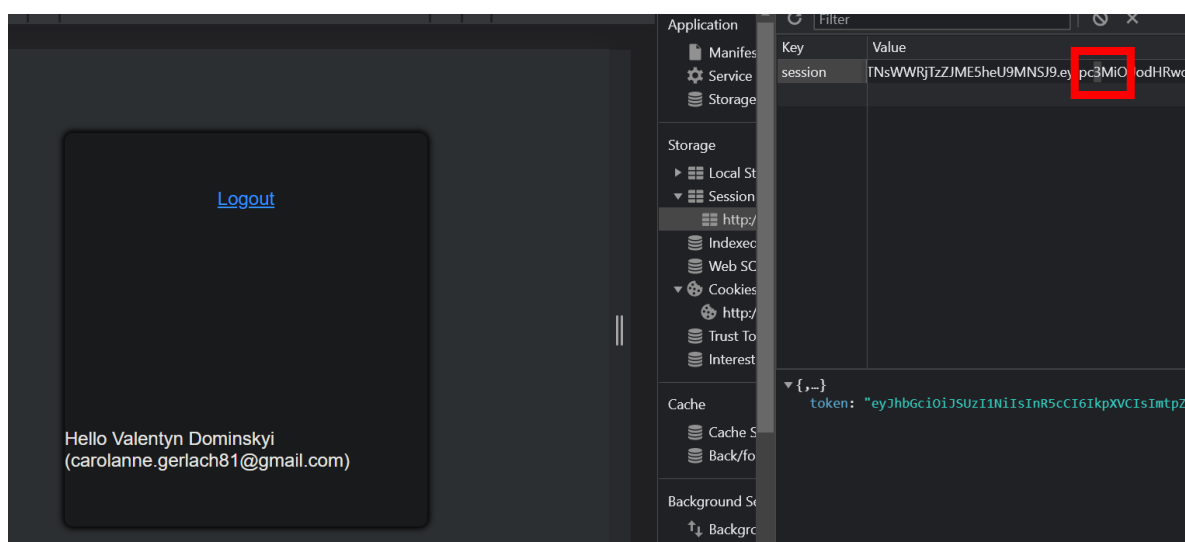
yJpc4MiOiJodHRwczovL2Rldi13eXZsaGt3bjQ3NW12MXduLnVzLmF1dGgwLmNvbS8iLCJzdWIiOiJhdXRoMHw2M2I4OGJhNT11YTE5YmE0Yjh1ODU3NzgiLCJhdWQiOiJodHRwczovL2Rldi13eXZsaGt3bjQ3NW12MXduLnVzLmF1dGgwLmNvbS9hcGkvZjIvIiwiaWF0IjoxNjczMTAzNjIwLCJleHAiOjE2NzMxOTAwMjAsImF6cCI6IjVIZExxb3hrR2Fjd1JvZjNSUT1hR31hb2MwOW9EVjRWIiwic2NvcGUiOiJyZ

```
"alg": "RS256",  
"typ": "JWT",  
"kid": "OKDnUU3:  
}
```

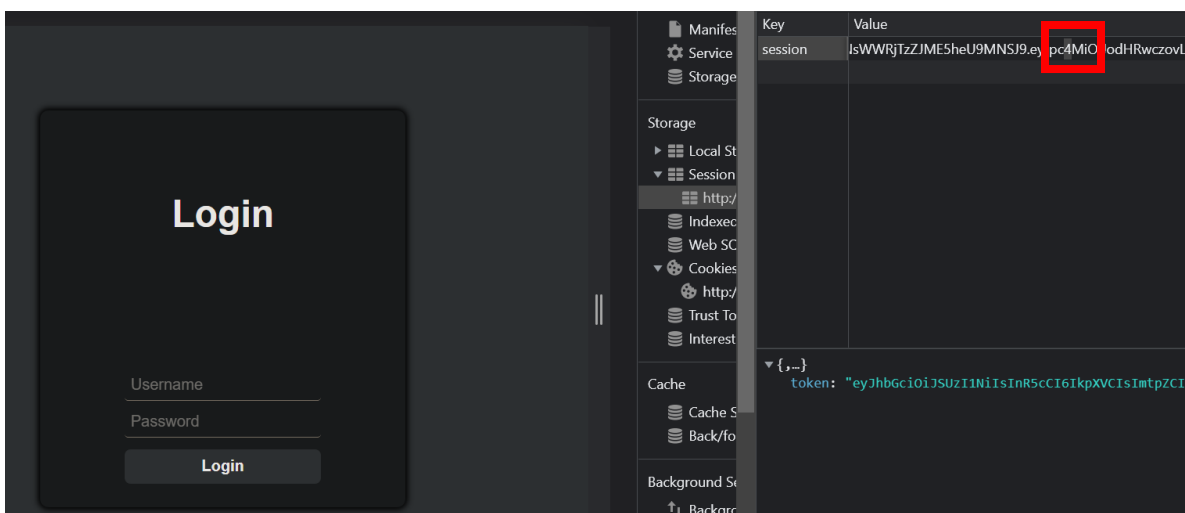
PAYLOAD: DATA

```
{  
  "iss": "https://  
  "sub": "auth0|6:  
  "aud": "https://
```

Але треба подивитися, що станеться у Нашому застосунку:



Змінимо цю цифру та перезавантажимо сторінку:



Ми не змогли верифікувати токен, тому Нас викинуло на початкову сторінку!Г

Висновок:

Під час виконання роботи Я отримав кращі практичні знання у верифікації JWT токенів та що відбувається, коли вони не проходять перевірку (наприклад при зміні токenu власноруч). Попрацював з асинхронним шифруванням, яке має публічний ключ

Посилання:

- [Проект на GitHub](#)