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 ІП-93

Домінський В.О.

Зміст

Завдання:	3
Виконання:	3
Висновок:	
Посилання:	

Завдання:

1. Розширити Лабораторну работу 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

Виконання:

У даній лабораторній роботі треба дописати функціонал верифікації ЈWТ 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}`);

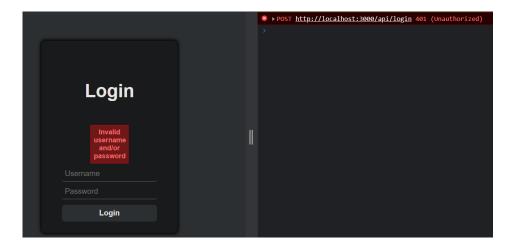
const publicKey = await getPublicKey();
    const appTokenInfo = await getAppToken();
    config.appToken = appTokenInfo.access_token;
    config.publicKey = publicKey;
})
```

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

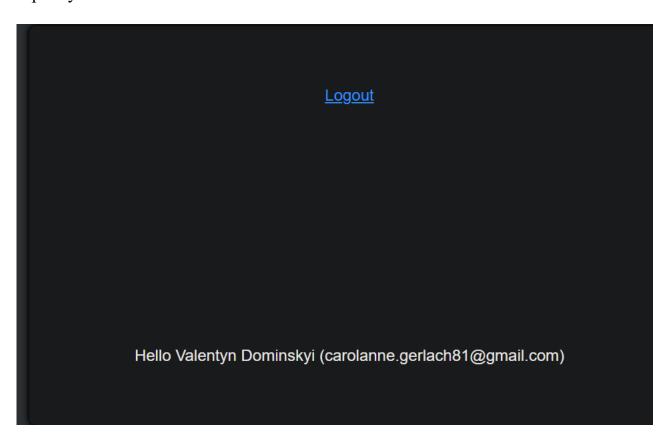
```
const pemOptions = {
    method: 'GET',
    url: `https://${config.domain}/pem`,
}

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:

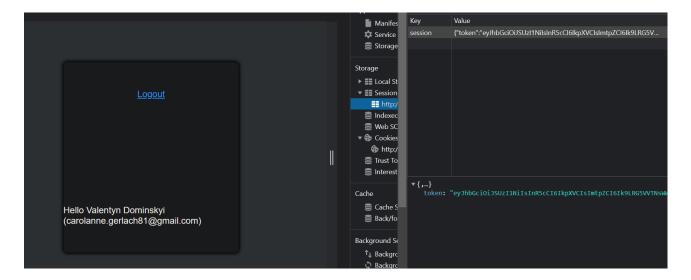


Як видно з консолі – Нам видало помилку 401. Тепер увійдемо в Нашого користувача:



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:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImt
pZCI6Ik9LRG5VVTNsWWRjTzZJME5heU9MNSJ9.e
                                                              PAYLOAD: DATA
jQ3NW12MXduLnVzLmF1dGgwLmNvbS9hcGkvdjIv
IiwiaWF0IjoxNjczMTAzNjIwLCJleHAiOjE2NzM
                                                                 "iss": "https://dev-wyvlhkwn475iv1wn.us.auth0.com/",
"sub": "auth0|63b88ba59ea19ba4b8e85778",
xOTAwMjAsImF6cCI6IjVIZExxb3hrR2FJd1Jvaj
                                                                "azp": "5HdLqoxkGaIwRoj3RQ9aGyGoc09oDV4V",
"scope": "read:current_user
RfdXN1c19tZXRhZGF0YSBjcmVhdGU6Y3VycmVud
F91c2VyX2RldmljZV9jcmVkZW50aWFscyBkZWxl
                                                               delete:current_user_metadata
                                                               update:current_user_identities offline_access",
50aXRpZXMgb2ZmbGluZV9hY2Nlc3MiLCJndHkiO
iJwYXNzd29yZCJ9.Ni2b1c2UQoxdCiV-
h4YANrY61TBhQXjbMw0NYhT8P17n_u7aJx-
                                                              VERIFY SIGNATURE
BI5WeY9TNd-
\verb"cuvX37QL_uGa7osCyW6rM_LWFchcmcc2_Yoc7-"
                                                              RSASHA256(
135nMfZzeM1jppMrjzXiQtfRL6Jga3mzEREKVoj
7vu4yUJ6u6588JFJuIpp-bqsHKqV0A-
FsHvI20c9Rmwjwb1eQEYJtbIUm0Y7NLMbV3XLSB
{\tt OwMgkyoEWsslppOKnBHNxAIi\_SRXLhpLv88m0GQ}
                                                                  "e": "AOAB".
u@dNrBz9qyh9fx\_2U2hPX21xeWSizKopdQrcmF2\\
                                                                  "n": "3wluQrwvayqYfC9LRh
56azwEJkEMEONqHfTYjJ8MTwEgI4-2up-
FN3G9Eo2LkGR2AHEWj1Z4A
```

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

yJpc4MiOiJodHRwczovL2Rldi13eXZsaGt3bjQ3 NWl2MXduLnVzLmF1dGgwLmNvbS8iLCJzdWIiOiJ hdXRoMHw2M2I4OGJhNT1lYTE5YmE0Yjh1ODU3Nz giLCJhdWQiOiJodHRwczovL2Rldi13eXZsaGt3b jQ3NWl2MXduLnVzLmF1dGgwLmNvbS9hcGkvdjIv IiwiaWF0IjoxNjczMTAzNjIwLCJleHAiOjE2NzM xOTAwMjAsImF6cCI6IjVIZExxb3hrR2FJd1Jvaj NSUT1hR31Hb2MwOW9EVjRWIiwic2NvcGUiOiJyZ

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

PAYLOAD: DATA

{
    "is** ": "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!т

Висновок:

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

Посилання:

• Проект на GitHub