УДК [004.56+003.26] (075.8)

Stud. A.P. Górecki
Supervisor prof. P. Urbanovich
(the John Paul II Catholic University of Lublin, Poland)

WEB-APPLICATION OF AN ONLINE SHOP WITH A DATABASE ON THE POSTGRESQL PLATFORM WITH AN INCREASED LEVEL OF SECURITY

Nowadays, more and more web applications are created that may be exposed to attacks by crackers who want to earn money on sensitive data [1, 2]. The e-commerce industry is developing just as well – people are more and more willing to shop without leaving home. This translates into the fact that stores collect more and more sensitive customer data. They are therefore more vulnerable to attacks. The consequence of the attack on the store is large losses, both financial and image. Customer data should be safely stored in the system, and the application itself should be well secured.

The purpose of this bachelor's thesis is to create a proprietary online store application with increased security. The application will be based on JAVA language, Spring Framework technology, the PostgreSQL database, TypeScript, JavaScript and Angular framework, HTML markup language and CSS style sheet using the SCSS preprocessor. Because of the division of the application into the frontend and backend, the frontend part must authenticate each request sent to the server with a token. This is a safer solution than if the browser were to have the user's password saved. The token is managed by the OAuth 2.0 library, and its creation process is based on the JWT standard, i.e. JSON Web Tokens.

The security problem of web-applications and stores is little noticed. Only after a successful attack on our client's application solutions are sought. That is why it is recommended to perform penetration tests that will show the weaknesses of the system. It is worth noting that it is impossible to secure the system fully, but you can minimize the risk of attacking it. In bachelor's thesis was decided to analyse the XSS attack, SQL-Injection, data security in the database and present in the proprietary application of the online store how to protect it from these attacks.

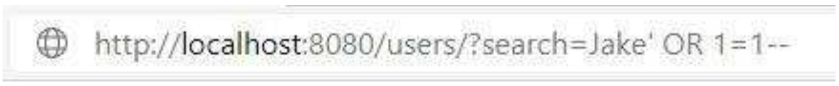
This knowledge about these attacks and protection can be used to build both online stores and other web-applications. Attacks and security tests should not be carried out without the consent of the owner of the IT system.

The most harmful attack is SQL-Injection, which involves injecting SQL code into the application, which enables:

- reading, modifying, deleting data even without authentication,

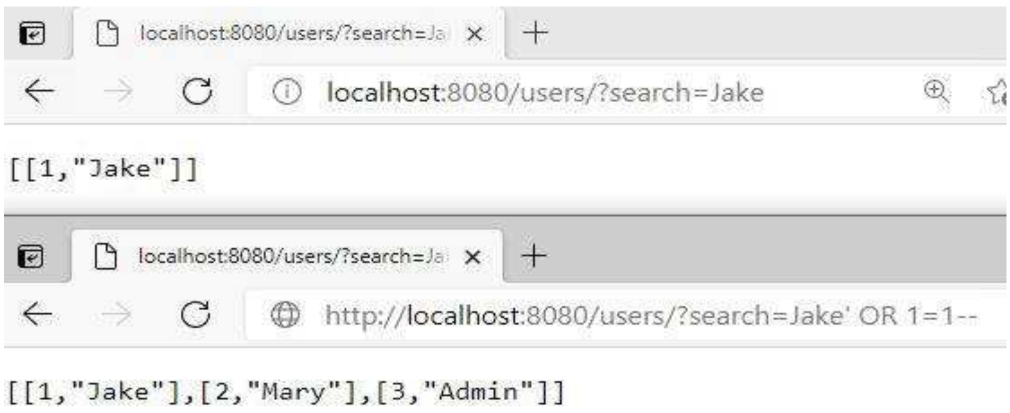
- access to the file system and saving files in the system on which the database is set up,
- running the code in the file system.

An example SQL query for the SQL-Injection attack is shown in Figure 1 – SQL code in the URL bar. Next, in Figure 2 **– Example without SQL-Injection (top) and after the attack (bottom)** in the first window, you can see that in the normal name search, one record appears, and in the second window, after the SQL injection, all records appear. A way to protect your application from SQL-Injection is to use parameterized queries in your systems [3].



```
http://localhost:8080/users/?search=Jake' OR 1=1--
```

Figure 1 – SQL code in the URL bar



```
localhost:8080/users/?search=Jake
```

```
[[1,"Jake"]]
```

```
http://localhost:8080/users/?search=Jake' OR 1=1--
```

```
[[1,"Jake"],[2,"Mary"],[3,"Admin"]]
```

Figure 2 – Example without SQL-Injection (top) and after the attack (bottom)

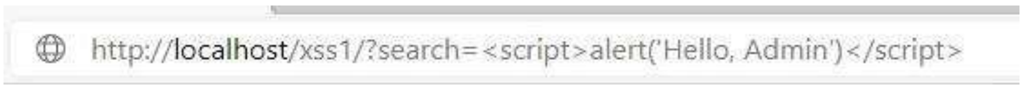
Emphasis was also placed on protection against XSS attacks. The Angular framework and the verification of the libraries attached to the project will be used for this. The XSS attack may lead to:

- intercepting the user's token or capturing passwords,
- dynamic code modification or action execution as logged-in user,

Typesof XSS attacks:

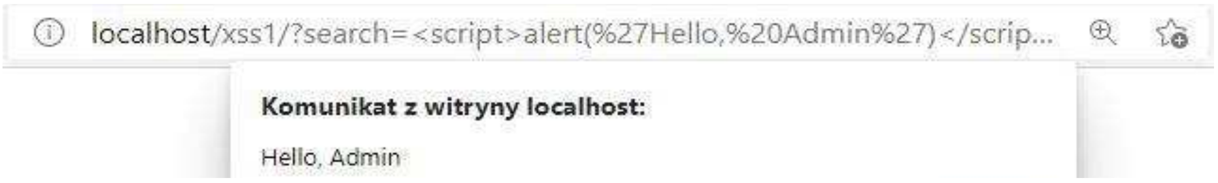
- reflected XSS – it consists of injecting JavaScript code into a parameter, e.g. GET, while passing the parameter value to the HTML view,
- stored XSS – stored malicious code in the database,
- DOM-based XSS – code injection in the address bar, which is passed to the eval or location function in the application.

An example of the JavaScript code for an XSS attack is shown in Figure 3 **– JavaScript code in the URL bar**. Figure 4 **– Successful reflected XSS attack** shows only a harmless alert, but it could become, for example, an attempt to steal a user's session. A way to protect against this attack is to use libraries to clean up untrusted HTML code [3].



```
http://localhost/xss1/?search=<script>alert('Hello, Admin')</script>
```

Figure 3 – JavaScript code in the URL bar



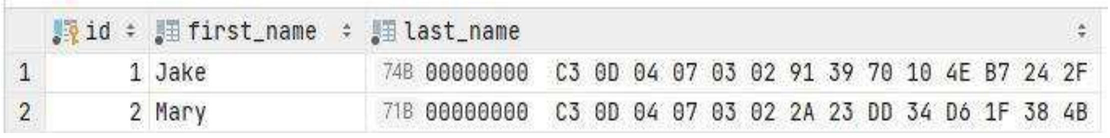
```
localhost/xss1/?search=<script>alert(%27Hello,%20Admin%27)</scrip...
```

Komunikat z witryny localhost:

Hello, Admin

Figure 4 – Successful reflected XSS attack

The last thing is the problem of storing data securely in the database. If the data is not encrypted, anyone who has access to the database can view confidential data. To make such action difficult, the PostgreSQL database will be used and its *pgp_sym_decrypt()* and *pgp_sym_encrypt()* functions to encrypt sensitive customer data [4]. It is worth noting that the database and the store's application must be run on two different servers. An example of encryption of the last name attribute can be seen in Figure 5 – The encrypted last_name attribute in PostgreSQL.



	id	first_name	last_name
1	1	Jake	74B 00000000 C3 0D 04 07 03 02 91 39 70 10 4E B7 24 2F
2	2	Mary	71B 00000000 C3 0D 04 07 03 02 2A 23 DD 34 D6 1F 38 4B

Figure 5 – The encrypted last_name attribute in PostgreSQL

REFERENCES

1. Ochrona informacji w sieciach komputerowych / pod red. prof. P. Urbanowicza. – Lublin: Wydawnictwo KUL, 2004. – 150 s.
2. Urbanowicz, P. Bezpieczenstwo w cyberprzestrzeni a prawo karne / P. Urbanowicz, M. Smarzewski // Ksiega pomiatkowa pomiatkowa ku czci Księdza Profesora Andrzeja Szostka MIC, Lublin: KUL. – 2016. – P. 479-488.
3. Bezpieczeństwo aplikacji webowych/ M. Bentkowski [at al.]. – Kraków: Securitum Szkolenia, 2019.
4. How to encrypt and decrypt data with Hibernate. [Electronic resource]. – Access mode: https://vladmihalcea.com/how-to-encrypt-and-decrypt-data-with-hibernate. – Access date: 23.01.2021.

УДК [004.56+003.26] (075.8)

Stud. Justyna Winiarczyk
Supervisor prof. P. Urbanovich
(the John Paul II Catholic University of Lublin, Poland)