universidade de aveiro

Report Project 1

from:

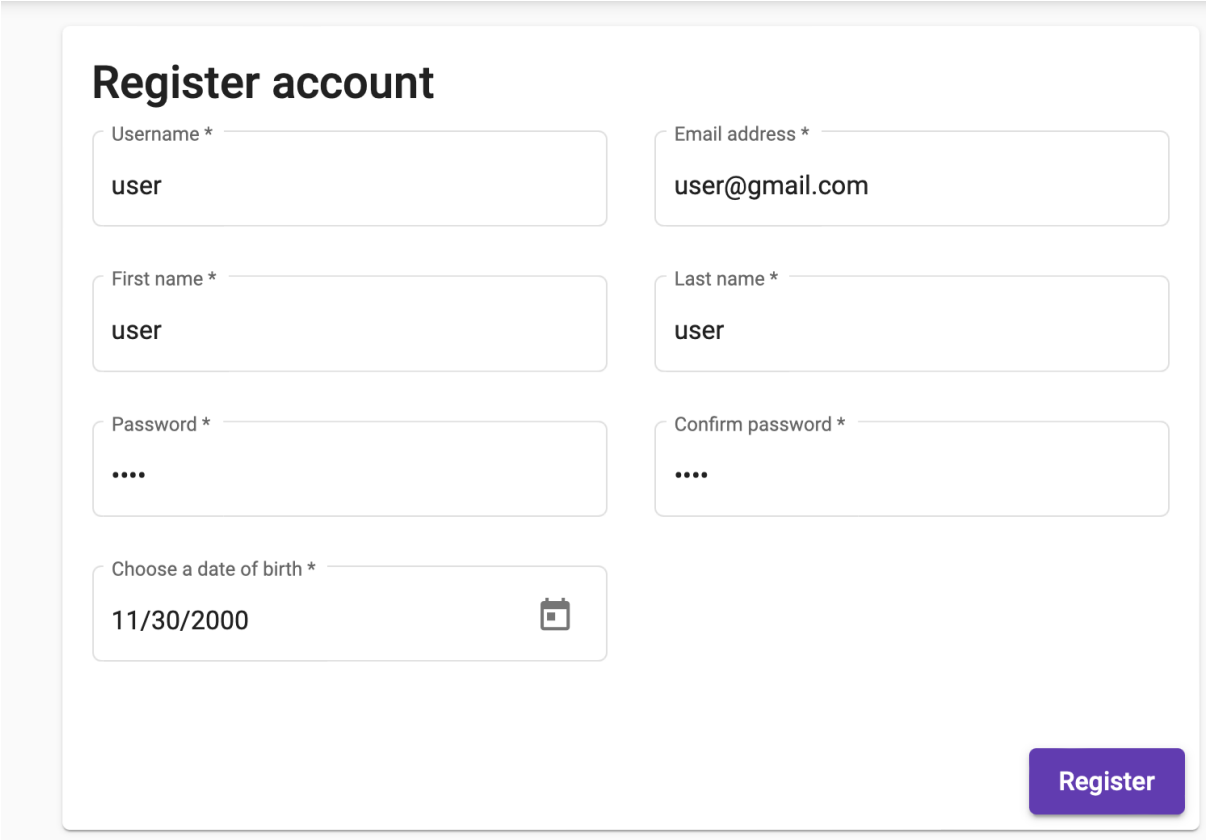
112169	MIRON ARTUR OSKROBA
112018	ZUZANNA SIKORSKA
112282	JANNIS JAKOB MALENDE
112059	STANISLAW FRANCZYK

Project Functionalities

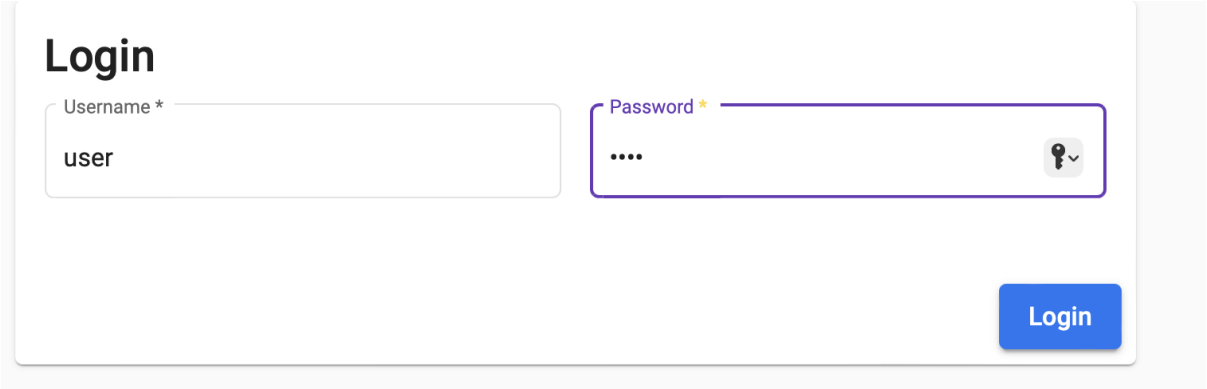
Our application offers the following services:

- registration of users
- login and logout of users
- make an appointment
- doctor simulator (automatic diagnosis)
- view diagnosis issued automatically by doctor simulator
- contact form to contact the clinic

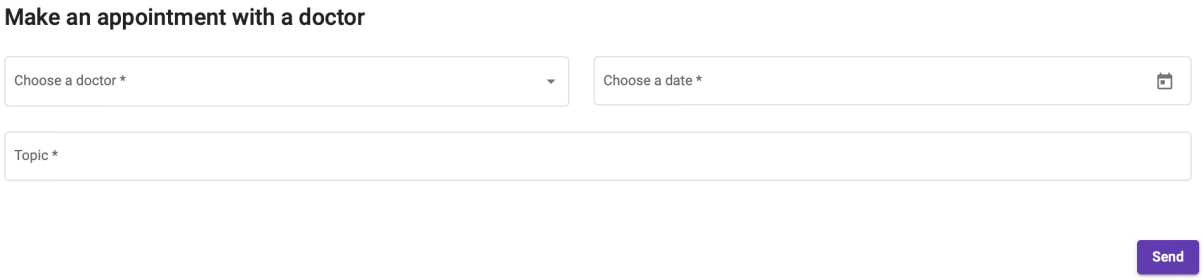
1. Registration of a new user



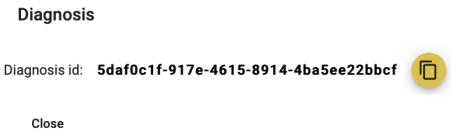
2. Login with the created user



3. Make an appointment

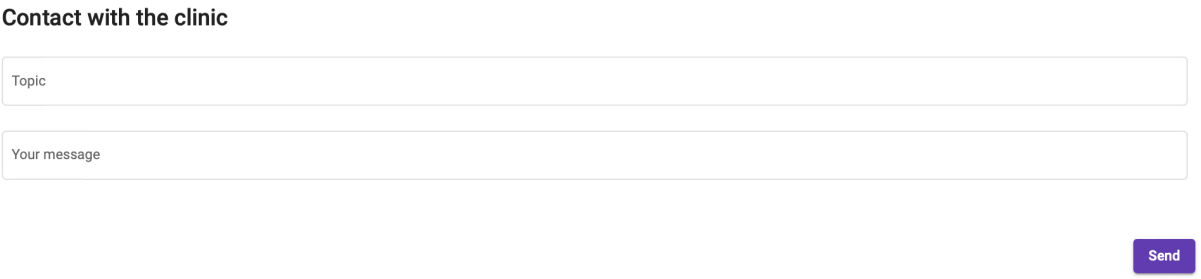


4. Copy the given diagnosis id



5. Paste the diagnosis id into Diagnosis of Examination form and see diagnosis

6. Contact with the clinic



Vulnerabilities

Assessment Strategy

To assess the vulnerabilities project team used a calculator [1], it's CVSS version is 3.1, it defines Base Score Metrics and possible options for Exploitability Metrics.

CWE-79

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Base Score Metrics	**Exploitability Metrics**
Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	Required
Scope	Changed
Confidentiality Impact	High
Integrity Impact	High
Availability Impact	High
CVSS Base Score	10.0

Vulnerability Description

The XSS attack can be performed in the contact form, which will get visible as soon as you are logged in. Scripts that are inserted in the "Your message" field, are getting executed, without any sanitizing.

Fix

In order to fix this vulnerability, we had to reactivate the built-in DomSanitizer provided by the Angular framework. This takes care of all inputs, which are provided by the frontend.

CWE-89

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Base Score Metrics	Exploitability Metrics
Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Changed
Confidentiality Impact	High
Integrity Impact	High
Availability Impact	High
CVSS Base Score	10.0

Vulnerability Description

The SQL injection can be performed in the login form or in the make appointment form. Some malicious inputs can be inserted, which will bypass the login or delete the database.

Fix

To resolve this vulnerability the project team had to stop using SQL vulnerable commands that were designed specifically to allow SQL injection. New solution assumes a spring-boot JPARepository instead. It is using prepared statements sanitizing the input. It is a common spring-boot practice to use repositories. Solutions like this are already secured to malicious input vulnerabilities.

CWE-1104

Use of Unmaintained Third Party Components

Base Score Metrics	Exploitability Metrics
Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	Required
Scope	Changed
Confidentiality Impact	High
Integrity Impact	High
Availability Impact	High
CVSS Base Score	10.0

This vulnerability can be assigned to an existing CVE-ID: **CVE-2021-44228**. In this case the logger Lo4gj had a severe vulnerability that allowed taking control over the whole system, by especially crafted inputs, which are getting logged and if correct later executed. Log4j was badly implemented by its developers, so it was exploited globally.

Vulnerability Description

It is a common practice to gather application logs. Because every service is logged, therefore every Input field is vulnerable to this attack. In order to make the app sensitive to this vulnerability, the project team had to explicitly use the vulnerable version of Log4j as well as exclude default spring-boot loggers.

Fix

In order to fix this issue, we had to upgrade to a newer version of Log4j, where the issue has finally been resolved by its developers. This was be done in the pom.xml file.

vulnerable dependency	fixed dependency
<dependency> <groupId>org.apache.logging.log4j</groupId> <artifactId>log4j-core</artifactId> <version>2.14.1</version> </dependency>	<dependency> <groupId>org.apache.logging.log4j</groupId> <artifactId>log4j-core</artifactId> <version>2.19.0</version> </dependency>

CWE-522

Insufficiently Protected Credentials

Base Score Metrics	Exploitability Metrics
Attack Vector	Local
Attack Complexity	High
Privileges Required	High
User Interaction	None
Scope	Changed
Confidentiality Impact	High
Integrity Impact	High
Availability Impact	None
CVSS Base Score	7.4

Vulnerability Description

In the vulnerable application the database credentials are stored in plain text. If an attacker gets access to this system, he would be able to get any data from the database.

Fix

To fix this problem we encrypt the patient's password using the Blowfish & Crypt algorithm. The password is encrypted by the backend service, precisely by the BCryptPasswordEncoder checks if the encrypted password matches the one that tries to authenticate. It is a must-have practice to hash database passwords like this.

CWE-259

Use of Hard-coded Password

Base Score Metrics	Exploitability Metrics
Attack Vector	Local
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Changed
Confidentiality Impact	High
Integrity Impact	High
Availability Impact	None
CVSS Base Score	6.5

Vulnerability Description

If the source code database credentials leak in the future, hard-coded credentials make it easy for an attacker to connect to the database and compromise data, integrity and availability. The vulnerable application source code contains a file “application.properties“, which contains sensitive data for the connection to the database.

Fix

.env file holds some secret environment variables. The solution assumes that the application code is hardcoded credentials free. The strategy of securing the credentials is to decouple them to a separate logic. Pass them during the application run-time, which the deployment management system would handle. They mostly supply the secrets manager solutions that are safe, so the code with the repository itself would be credentials-free. Normally, the .env file should not be pushed into this repository - it is just for explaining purposes and easier assessment.

CWE-532

Insertion of Sensitive Information into Log File

Base Score Metrics	Exploitability Metrics
Attack Vector	Local
Attack Complexity	High
Privileges Required	High
User Interaction	None
Scope	Changed
Confidentiality Impact	High
Availability Impact	Low
CVSS Base Score	5.6

Vulnerability Description

The vulnerable application will log the typed in username and password, which are considered as sensitive data. It is important to not log user information or system information, in order to not expose them accidentally to potential attackers. Through the help of the vulnerable version of the logger Log4j, it would be possible to get access to the system and by the log file enable to expose the sensitive information.

Fix

Remove the lines of code that are responsible for logging sensitive data.

Vulnerabilities Summary

CWE	CVSS 3.1
CWE-79	10
CWE-89	10
CWE-1104	10
CWE-522	7.4
CWE-259	6.5
CWE-532	5.6
Total	49.5

Conclusion

The frameworks chosen for the project had built-in SQL-injection and XSS protection enabled by default. We needed to explicitly bypass that protection in our application in order to make it vulnerable. Modern frameworks take security of the applications very seriously into consideration. This is why it was hard to omit the built-in protection mechanisms.

Bibliography

[1] https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator