OWASP Report

Veronika Valeva

St num: 4090349

Versioning table

VERSION	DATE	AUTHOR	CHANGES	STATE
0.1	16.12.2021	Veronika Valeva	Initial state of the design document.	Sprint 5 deliverable
0.2	08.01.2022	Veronika Valeva	Added risk context.	Sprint 6 deliverable

	Likelihood	Impact	Risk	Actions possible	Planned
A01: Broken access control	High	Default severity	Low	For all the routes/controllers of an application, the authorized HTTP methods should be explicitly defined and safe HTTP methods should only be used to perform read-only operations.	No.
A02: Cryptographic Failures	Very unlikely	Default severity	Moderate	Use a standard algorithm instead of creating a custom one.	N/A. No passwords or user data used. No risk, accepted
A03: Injection	Unlikely	Default severity	Moderate	Sanitize all values used as JavaBean properties. Don't set any sensitive properties	No
A04: Insecure Design	Likely	Severe	Moderate	Establish and use a library of secure design patterns. Use threat modeling for critical authentication, access control, business logic, and key flows	No, risk accepted
A05: Security Misconfiguration	Likely	Severe	High	Protection against CSRF	Solution is not

				attacks. Activated by default for all unsafe HTTP methods.	possible for this state of the project.
A06: Sensitive Data Exposure	Very unlikely	Default severity	High	Don't use a token without verifying its signature before.	No. Risk accepted.
A07: Identification and Authentication Failures	Likely	Severe	High	The same message should be used regardless of whether it is the wrong user or password (Bad credentials)	Yes.
A08: Software and Data Integrity Failures	Very unlikely	Default severity	Low	Don't hard-code the IP address in the source code, instead make it configurable with environment variables, configuration files, or a similar approach.	Yes.
A09: Security Logging and Monitoring Failures	Likely	Severe	Moderate	End the logging practice .	Yes.
A10: Severe-Side Request Forgery	High	Default severity	Moderate	Improve framework implementation	No, risk accepted

Risk context

A01: Broken access control. Low. An HTTP method is safe when used to perform a read-only operation, such as retrieving information. In contrast, an unsafe HTTP method is used to change the state of an application, for instance to update a user's profile on a web application. Allowing both safe and unsafe HTTP methods to perform a specific operation on a web application could impact its security. The risk is low as the defined and used HTTP methods for a route/controller cannot change the state of an application.

A02: Cryptographic Failures. Moderate. The use of a non-standard algorithm is dangerous because a determined attacker may be able to break the algorithm and compromise whatever data has been protected.

A03: Injection. Moderate. Setting JavaBean properties is security sensitive. JavaBeans can have their properties or nested properties set by population functions. An attacker can leverage this feature to push into the JavaBean malicious data that can compromise the software integrity.

A04: Insecure Design. Moderate. The system holds the billing information of its users. This billing information is used in the ordering process. Attackers could threat model this flow steal or fake billing info, causing a massive loss of income.

A05: Security Misconfiguration. High. A cross-site request forgery (CSRF) attack occurs when a trusted user of a web application can be forced, by an attacker, to perform sensitive actions that he didn't intend, such as updating his profile, more generally anything that can change the state of the application. The attacker can trick the user/victim to click on a link, corresponding to the privileged action, or to visit a malicious web site that embeds a hidden web request and as web browsers automatically include cookies, the actions can be authenticated and sensitive.

A06: Sensitive Data Exposure. High. If a JSON Web Token (JWT) is not signed with a strong cipher algorithm (or not signed at all) an attacker can forge it and impersonate user identities.

A07: Identification and Authentication Failures. High. In a Spring-security web application the username leaks when the string used as argument of 'loadUserByUsername' method is used in an exception message.

A08: Software and Data Integrity Failures. Hardcoding IP addresses is security sensitive. Attackers might be able to decompile the code and thereby discover a potentially sensitive address.

A09: Security Logging and Monitoring Failures. Having production deployment's loggers in "debug" mode can lead to writing sensitive information in logs. Production logs should not be stored in a location which is easily accessible to threat.

A10: Severe-Side Request Forgery. Moderate. In a Server-Side Request Forgery (SSRF) attack, the attacker can abuse functionality on the server to read or update internal resources. The attacker can supply or modify a URL which the code running on the server will read or submit data to, and by carefully selecting the URLs, the attacker may be able to read server configuration.

Conclusion

In this report I've analyzed the weaknesses in the security and the structure of the project. Actions will be taken to prevent those weaknesses. The planned refactoring of the code will circle around the topics of Broken Access Control, Sensitive Data exposure, Identification and Authentication Failures, Software and Data Integrity Failures and Security Logging and Monitoring Failures. The controllers will be refactored so that the authorized HTTP methods are explicitly defined and safe HTTP methods should only be used to perform read-only operations. The debug feature will be disabled. The exceptions will be refactored so that they do not present threat to the security of the application. The cors policy headers and authenticators will be refactored, using separate variables and not the permitted ip address in the actual configuration. The console logging will be disabled.