OWASP Top10 Security

Bsky Airline

Bsky

Version table

Author:	Florin Deleanu 3784711	Revision Date
	Version 1	May 2021

Contents

Bsky

Introduction

For any application which contains user data, security is an important concern that the developers need to constantly keep in mind. This document will list the ten risks and then assess them based on the following criteria:

- Is the risk applicable for the application, and why (or why not)?
- How did the risk impact your implementation? How does this effect the risk?

A1:2017-Injection:

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

This risk is not applicable as the data queries are handles by a spring boot API which automatically manages the queries

A2:2017-Broken Authentication:

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

This is a risk in every application that has an authentication system. In order to minimize it I use a trusted spring boot security system (JWT security) which uses a bearer token with certain permissions(member/admin)

A3:2017-Sensitive Data Exposure:

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

In order to address this change I use a password encoder to hash the password in the database. Other fields do not contain sensitive data. (I have a field in the database which holds the users budget but that is purely for demonstration purposes and in a real application I would use a 3rd party payment service to handle everything securely)

A4:2017-XML External Entities (XXE):

Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

Not applicable as I do not use any XML files

A5:2017-Broken Access Control:

Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

JWT Security is a very good security system and even if the user tried to access hidden API calls they would be blocked by the back end because they are lacking the permission.

A6:2017-Security Misconfiguration:

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

This is a constant issue in all existing applications, but at the moment it is not a problem as my application is newly made with the latest technology using proper programming principles.

A7:2017-Cross-Site Scripting XSS:

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

The React framework that I'm using automatically escapes XSS by design, so this is not applicable

A8:2017-Insecure Deserialization:

Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

For my application for all created objects there is a data transfer object(DTO) created to nullify this vulnerability.

A9:2017-Using Components with Known Vulnerabilities:

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

This vulnerability is more prevalent in applications which use outdated frameworks and dependencies. I am currently using the latest technology which is updated to minimize this risk

A10:2017-Insufficient Logging & Monitoring:

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

At the moment my application does not implement any logging of the information, only basic printing the errors to the console to let the user know what happened. This could be changed in the future to instead log the errors into a text file.

Conclusion

To conclude, I have listed all the risks, gave a definition on what they are (OWASP) and how I address them in my application (if they apply).

Bibliography

OWASP. (n.d.). Retrieved from https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%2526Monitoring