Security of Web Applications  
OWASP Top 10

Semester 3 individual project

S3-S-CB-03

Onea Bianca

December 17, 2021

Contents

[1. A1-Broken Access Control 2](#_Toc90657325)

[2. A2- Cryptographic Failures 2](#_Toc90657326)

[3. A3- Injection 3](#_Toc90657327)

[4. A4- Insecure Design 3](#_Toc90657328)

[5. A5- Security Misconfigurationn 3](#_Toc90657329)

[6. A6- Vulnerable and Outdated Components 4](#_Toc90657330)

[7. A7- Identification and Authentication Failures 4](#_Toc90657331)

[8. A8- Software and Data Integrity Failures 4](#_Toc90657332)

[9. A9- Software and Data Integrity Failures 4](#_Toc90657333)

[10. A10- Server Side Request Forgery (SSRF) 5](#_Toc90657334)

[11. Analysis Table 5](#_Toc90657335)

# 1. A1-Broken Access Control

Definition: Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

What I did for my individual project:

* Send a 403 error message when the user tries to perform an action and does not have authorization for it.
* Restrict some of the features if the user is not logged in.
* CORS Configuration which allow only trusted origins.
* When the user tries to login for five time without success, the 6th unsuccessfull try will result in blocking the user’s account (for bruce attacks).
* Separate content for admin, which requires the admin to be logged in.
* Encrypting the password of the user in the database.
* Actions that can only be done if the user’s logged in require the token the user received on log in.

# 2. A2- [**Cryptographic Failures**](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

Definition: The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws.

What I did for my individual project:

* Instead of storing the actual users’ passwords in the database, I encrypt them.
* When I store the received token in the local storage after log in, I first encrypt it.

# 3. A3- INJECTION

Definition: Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections.

What I did for my individual project:

* I am not using SQL querries for manipulating the data from the database.
* I am making use of the JPA Repository and I use its methods for CRUD operations.

# 4. A4- INSECURE DESIGN

Definition: Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design.” There is a difference between insecure design and insecure implementation. We differentiate between design flaws and implementation defects for a reason, they have different root causes and remediation. A secure design can still have implementation defects leading to vulnerabilities that may be exploited.

What I did for my individual project:

Backend:

* I structured my project into multiple parts: model, DAO, DAL, service, controllers
* Good security configuration

Frontend:

* Structured my project in: component and services (which contain classes that communicate with the backend and make API calls)

# 5. A5- SECURITY MISCONFIGURATION

90% of applications were tested for some form of misconfiguration, with an average incidence rate of 4.%, and over 208k occurences of a Common Weakness Enumeration (CWE) in this risk category.

What I did for my individual project

* Good security configuration
* I don’t make use of unecessary pages, services, ports, etc.

# 6. A6- Vulnerable and Outdated Components

Definition: Vulnerable Components are a known issue that we struggle to test and assess risk and is the only category to not have any Common Weakness Enumerations mapped to the included CWEs.

What I did for my individual project:

* All the components and their version are in the README file of the repository
* Almost all the dependencies (backend) are upgraded to their last version
* I am using only the dependencies I need (both frontend and backend)

# 7. A7- Identification and Authentication Failures

Definition: Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks.

What I did for my individual project:

* I restrict brute force attacks: when the user tries to login for five time without success, the 6th unsuccessfull try will result in blocking the user’s account (for bruce attacks).
* The ”Reset Password” feature will give the user a new, random password, which will be recceived on the user’s e-mail.

# 8. A8- Software and Data Integrity Failures

Definition: Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise

What I did for my individual project:

* All my used dependencies (backend) and libraries (frontend) are from trusted sources
* Good CI/CD pipeline configuration, the access to my repository is restricted

# 9. A9- Software and Data Integrity Failures

Definiton: This category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time.

What I did for my individual project:

* Show error messages on actions (ex: Username or password is incorrect)
* Show error messages on incorrect format (ex: if a required input is empty)
* Show error messages if the user does not have authorization when trying to do certain actions/trying to access certain pages
* Don’t show too much or too little information to the user

# 10. A10- Server Side Request Forgery (SSRF)

Description: SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list.

What I did for my individual project:

* CORS Configuration which allow only trusted origins.

# 11. ANALASYS TABLE

|  |  |  |  |
| --- | --- | --- | --- |
|  | Likelihood | Impact | Planned |
| A1 | High | HIGH | Yes |
| A2 | Unlikely | LOW | Yes |
| A3 | Unlikely | LOW | Yes |
| A4 | Likely | MODERATE | No, risk accepted |
| A5 | Likely | HIGH | Yes |
| A6 | Unlikely | LOW | Yes |
| A7 | Unlikely | HIGH | Yes |
| A8 | Likely | MODERATE | No, risk accepted |
| A9 | Unlikely | MODERATE | Yes |
| A10 | High | MODERATE | No, risk accepted |

# 12. CONCLUSION

To conclude with, I would say that my web application is kind of secure, thank to some important risks that I kept in mind while developing my project. Furthermore, I think it would need some improving in the future to ensure a more secure and a safer application.