“Ari on tour” concert ticketing website

– OWASP Report –

Irina-Maria David – 3371581

Student at Fontys University of Applied Sciences – S3CB04

Document version 1.0

Table of Contents

OWASP Top 10 security risks table

Name	Likelihood	Impact	Risk	Actions possible	Planned
A1: Broken Access Control	High	Severe	High	Fixed	Yes
A2: Cryptographic Failures	Unlikely	Severe	High	Fixed	Yes
A3: Injection	Moderate	Severe	Moderate	Enforce stronger security for multiple types of attacks.	Not yet
A4: Insecure Design	Low	Moderate	Moderate	Take care of handling exceptions and add error messages.	Yes
A5: Security Misconfiguration	Unlikely	High	High	Remove any unnecessary feature or functionality from the website to reduce the attack surface.	Yes
A6: Vulnerable and Outdated Components	Unlikely	High	Low	Fixed	Yes
A7: Identification and Authentication Failures	Unlikely	Severe	High	Fixed	Yes
A8: Software and Data Integrity Failures	Unlikely	High	High	Fixed	Yes

A9: Security Logging and Monitoring Failures	Moderate	Moderate	High	Not yet taken care of	Accepted the risks
A10: Server-Side Request Forgery	High	Moderate	Moderate	To properly validate all user input and reject any request that contains invalid or malicious data.	Yes

Security risks and conclusions for my website

A1: Broken Access Control

Examples

Some examples are lack of proper authentication, lack of proper authorization, insecure direct object references and failure to restrict URL access.

- If the website does not require users to authenticate before accessing certain data or functionality, it could allow unauthorized users to access sensitive information.

- If the website does not properly enforce authorization (check if the user has necessary permissions to access a certain resource), it could allow unauthorized users to access sensitive information or functionality.

- An attacker is able to access sensitive data by manipulating URLs or form input to reference resources they should not have access to. An attacker could try to access a user's account information by changing the URL to reference a different user's account.

- A website does not properly restrict access to certain URLs or functions, it could allow unauthorized users to access sensitive data or functionality. An attacker could try to access a restricted URL by guessing the URL or manipulating the request.

Solutions

I will ensure that the users are required to provide valid credentials (such as a username and password) before they can access sensitive resources or perform actions. I am using role-based access to grant different levels of access to different users based on their roles. I will properly validate the user input to ensure that it is in the expected format and meets any other set requirements. This can help to prevent attackers from manipulating the application to gain unauthorized access to resources or perform actions that they should not be able to perform.

A2: Cryptographic Failures

Examples

Some examples are using weak or reused passwords and using weak or outdated cryptographic algorithms.

- If a website allows users to choose weak passwords or does not enforce password complexity requirements, it could be vulnerable to password cracking attacks. If a website does not store passwords securely (e.g., using a hashing function) or allows users to reuse passwords, it could be vulnerable to attacks.

- If a website uses weak or outdated cryptographic algorithms, it could be vulnerable to attacks that can break the encryption and allow attackers to access sensitive data.

Solutions

I am using Bcrypt hashing algorithm to sore hashed versions of their passwords in the database. This helps to prevent attackers from accessing the plaintext passwords, even if they gain access to the database.

A3: Injection

Examples

Some examples are SQL injection and Cross-Site Scripting (XSS).

- A type of injection attack where an attacker can inject malicious code into a website's database, potentially allowing them to access sensitive data or compromise the website.

- A type of injection attack where an attacker can inject malicious code into a website, which is then executed by the victim's browser.

Solutions

I implemented my solution in such way that it ensures input validation and always checks user input, making sure that it is safe and expected. This helps prevent attackers from injecting malicious code into the website. The passwordEncoder compares the provided password with the users' encoded password retrieved from the database. This ensures that the user-provided input is properly escaped and quoted, and it prevents malicious SQL from being injected into the queries.

A4: Insecure Design

Examples

Some examples are lack of input validation, lack of proper access controls and lack of proper error handling.

- If a website does not properly validate user input, it could be vulnerable to injection attacks or other types of attacks that rely on manipulating input.

- If a website does not properly enforce access controls, it could allow unauthorized users to access sensitive data or functionality.

- If a website does not properly handle errors, it could reveal sensitive information, such as stack traces or system details, to attackers.

Solutions

The same explanation as in the A3 section applies, including displaying user-friendly error messages to prevent the attacker from gathering information about the website. I will also handle exceptions in a friendly manner.

A5: Security Misconfiguration

Examples

Some examples are default or weak passwords and unnecessary services or components.

- If a website or its underlying infrastructure uses default or weak passwords, it could be vulnerable to attacks that exploit these vulnerabilities.

- If a website or server has unnecessary services or components installed, it could increase the attack surface and make the website or server more vulnerable to attacks.

Solutions

I am making sure to remove any unnecessary feature or functionality from the website to reduce the attack surface.

A6: Vulnerable and Outdated Components

Examples

Some examples of components with known vulnerabilities could include outdated versions of ReactJS or Spring Boot, third-party libraries or frameworks that have known vulnerabilities, such as Bootstrap or custom code or scripts that have been identified as having vulnerabilities.

Solutions

I will make sure to have the dependencies up to date to use the most recent versions of the components that may contain security fixes and other improvements.

A7: Identification and Authentication Failures

Examples

Some examples of identification and authentication failures could include using weak or easily guessable passwords, failing to properly validate user input, such as allowing users to enter arbitrary values for username or password, failing to properly hash or encrypt passwords or failing to properly authenticate users before allowing them to access protected resources or perform sensitive actions.

Solutions

I am using password hashing to store the passwords in the database securely and I will encourage the users to provide unique and strong passwords for their accounts.

A8: Software and Data Integrity Failures

Examples

Some examples of software and data integrity failures could include unauthorized modification or deletion of data by a user with improper permissions, injection of malicious code or data into the application, accidental data loss or corruption due to system errors or failures.

Solutions

I am using version control, to track changes and roll back to a previous version in case if something goes wrong. I am testing the code that I am implementing into the application for catching bugs and issues.

A9: Security Logging and Monitoring Failures

Examples

Some examples are logs of API not being monitored for suspicious activity, warnings and errors generate no, inadequate, or unclear log messages and auditable events, such as logins, failed logins, and high-value transactions, are not logged.

Solutions

Not considered yet.

A10: Server-Side Request Forgery

Examples

An attacker can send a request to a server from a vulnerable application on behalf of the server, tricking the server into performing actions that the attacker wants it to perform. This can be done by manipulating the application to send a request to a server-side resource that is not intended to be accessible to the attacker.

An example that could potentially allow the attacker to gain unauthorized access to sensitive data is when the attacker manipulates the application to send a request to an internal network resource, such as a database or a file system that is not intended to be accessible to the attacker.

Another example is that an attacker could manipulate the application to send a request to an external resource, such as a third-party API, that is not intended to be accessible to the attacker. This could potentially allow the attacker to carry out actions on behalf of the server, such as making unauthorized requests or sending spam emails.

Solutions

I made sure to properly validate all user input and reject any request that contains invalid or malicious data. This prevents attackers from sending malicious requests.

Conclusions

For my website I still have to take care of handling all the exceptions and provide user friendly messages, enforce security for multiple types of attacks and remove all the unnecessary functionalities or features to minimize the attack surface..

References

OWASP Top 10:2021. (n.d.-c). https://owasp.org/Top10/

Veracode. (n.d.). *OWASP Top 10 Vulnerabilities*. https://www.veracode.com/security/owasp-top-10

Sucuri WebSite Firewall - Access Denied. (n.d.). https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2020/

Chachak, E. (2021, November 2). *Real-World Examples for OWASP Top 10 Vulnerabilities*. CyberDB. https://www.cyberdb.co/real-world-examples-for-owasp-top-10-vulnerabilities/

Tal, L. (2022, October 6). *OWASP Top 10 | OWASP Top 10 Vulnerabilities 2021*. Snyk. https://snyk.io/learn/owasp-top-10-vulnerabilities/