Fontys Hogescholen

Software Security Analysis

S-CB-ITS3-S3-CB02 - Individual Project

Ömer Faruk GÖKBAK

3782174

Table of Contents

Document Change Record

Date	Version	Author	Comments
15/1/2021	0.0.1	Ömer Faruk GÖKBAK	First version was created.
19/1/2021	0.0.2	Ömer Faruk GÖKBAK	Designing was updated.

Introduction

Overview

The goal of the document is to become aware of and validate on top of the most critical and potential security risks and vulnerabilities.

This document contains the evaluation for Web Application Security Assessment of Online Inventory Management project, and it is based on OWASP Top 10[1] criteria , the audit of the 10 most common vulnerabilities for applications. These criteria also help to check the possible risks, impacts and countermeasures as well. Using the criteria is also an effective step to produce more secure code.

Risks research

1. Injection:

Attackers try to inject with SQL, LDAP or CRLF injection. Untrusted data, input is sent to server and try to access data with invalid authorization.

Needs Action: It doesn't need any action, however if SQL queries were used in the application, steps below should be taken.

- LIMIT and further SQL controls should be used inside the queries to prevent mass disclosure of records in case of SQL injection
- For any dynamic queries, needs to be escaped special characters using the specific escape syntax for that interpreter.

Action(s) Taken: Yes

- Project was created on React environment and React DOM escapes any values embedded in JSX before rendering them, because JSX renders text as text and HTML in user input is not seen as an HTML, just as a plain text.Therefore it is not possible to inject extra thing that's specifically written for the React App. JSX
- I have used ORM and JPA for data access to avoid using any SQL query, therefore the application is safe for any kind of injection.

2. Broken Authentication:

Invalid credentials, failure of the authentication process and incorrect implementation give attackers a chance to manipulation of credential data in a web app.

Needs Action: To prevent extra vulnerabilities, steps below also could be followed.

- Failed login attempts should be limited or increasingly delayed. All failures should be logged, and administration should be alerted when credential stuffing, brute force, or other attacks are detected.

- Where possible, should be implemented multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks.
- Do not deploy with any default credentials, particularly for admin users. In my project each employee is assigned default password, but they can update in the first login.

Action(s) Taken: Yes

- Communication between REST-API and React app is provided by JWT with strong secret keys.
- Passwords are not stored as a plaintext in the database. They are all hashed by bcrypt function.

3. Sensitive Data Exposure

Attackers try to exploit different type of scenarios to get data when it is in transit or temporarily stored in a text, by cracking a weaker encryption scheme or trying to decryption of pre-hashing popular passwords.

Needs Action: Yes

- Session cookies and other sensitive browser data should be only sent on HTTPS and that they're not available in client-side javascript. I send the data with Axios on HTTP by localhost.

Action(s) Taken: Yes

- User password is the single sensitive data within the application. It is encrypted with bcrypt salted hashing function.
- Passwords are not stored in the JWT which is used to authentication for each time.

4. XML External Entities(XXE)

This type of attacks is also considered as an injection attack. Attackers try to steal essential data from the server by uploading external XML files. The easiest way to prevent these attacks is avoid to use XML. Pick a stricter data format like JSON or YAML.

Needs Action: No, because XML entities are not used in the project.

Action(s) Taken: No, because XML entities are not used in the project.

5. Broken Access Control

Similar with the broken authentication issue. However, in this case it occurs when a logged-on person tries to access data which is not related to him. Additionally, it also happens due to the lacking limitation of authority. The situations that rise are events such as accessing other user accounts, accessing sensitive files, changing other users' data and changing access rights

Needs Action:

- Log access control, failures alert admins when repeated failures are detected.

Action(s) Taken:

- When an attacker intends to access sensitive data without being logged in as the correct user, system automatically logs out that user and redirects to the login page.
- CRUD calls to API are user-specific and managed with explicit role model permissions.

6. Security Misconfiguration

These are the vulnerabilities that arise when the applications running on the system are left in the default settings. In this case, the attacker provides access to the system by guessing the default settings. In this type of vulnerability, misconfiguration also gives the attacker an opportunity.That is not a result of programming mistake but a configuration failure.

Needs Action: Yes

- Security headers could be used but I did not use them due to the lack of knowledge.

Action(s) Taken: Yes

- The most up-to-date version of current framework & libraries are used in the project.

7. Cross-Site Scripting (XSS)

It is the issue that occurs when an attacker tries to make a transaction without the knowledge of user authority.

Needs Action: No

Action(s) Taken: Yes

- React is an up-to-date framework that protect against XSS includes provision for XSS. JSX renders everything just as a plain string. Application was created as a React project.
- In the project, user inputs are validated and follows the JSX rules. Additionally, dangerouslysetinnerhtml method is considered as a danger in React, which is not used in my project.

8. Insecure Deserialization

Basically, it can occur by deserializing a serialized data into an object. Attacker attempts to deserialize the data insecurely.

Needs Action: No

Action(s) Taken: Using Spring boot REST API

- Objects are not serialized directly from Spring boot app to React app. Spring boot app sends JSON schema of objects as a DTO, which does not contain any crucial data.

9. Using Components with Known Vulnerabilities

These are the vulnerabilities caused by #rd party software used in the application. Even if your app is safe but the plugin which is used may have a security flaw.

Needs Action: No

Action(s) Taken: Yes

- I have removed all unused dependencies, unnecessary features, components, files, and documentation.
- I checked/monitored the versions of the frameworks, libraries.
- I used reliable sources, signed packages to reduce the chance of including a modified, malicious component in React.

10. Insufficient Logging & Monitoring

The fundamental reason, why this vulnerability occurs, is absence of attack detection systems in applications or APIs.

Needs Action: Yes

- Ensure all login, access control failures, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts and held for sufficient time to allow delayed forensic analysis.
- Alert thresholds can be used for suspicious activity.
- Preferably, third party monitoring service could be used to consume and visualize logs and raising alarms in real time

Action(s) Taken: No, I did not take any action because I do not have enough knowledge to take the actions above.

Conclusion

As a result, I have evaluated my project to the OWASP TOP 10 criteria to see potential vulnerabilities and listed as what to do for each step. These findings raise my awareness about injections and guide me to boost the security level of the current application and the upcoming ones as well. Security issues will be constant problems for developers forever, therefore further approaches and solutions must be found to resolve application security risks.

References

1- Owasp.org. 2021. *OWASP Top Ten Web Application Security Risks | OWASP*. [online] Available at: <https://owasp.org/www-project-top-ten/> [Accessed 15 January 2021].

2- Medium. 2021. *Developers: Don'T Make These Top 10 Security Mistakes In Your Applications*. [online] Available at: <https://medium.com/swlh/top-10-web-app-security-risks-how-to-stop-them-according-to-the-experts-at-owasp-f568b881f406> [Accessed 15 January 2021].

3- Ictresearchmethods.nl. 2021. *The DOT Framework - ICT Research Methods*. [online] Available at: <http://ictresearchmethods.nl/The_DOT_Framework> [Accessed 15 January 2021].

4- Simform.com. 2021. *React Security Vulnerabilities That You Should Never Ignore! | Simform*. [online] Available at: <https://www.simform.com/react-security-vulnerabilities-solutions/> [Accessed 15 January 2021].