OWASP report

Budget Simple

Contents

	Likelihood	Impact	Risk	Actions possible	Planned
A1: Broken access control	Moderate	Severe	High	Use multi-factor authentication; Use short expiration of JWT token and a refresh token; Delete any inactive or unnecessary accounts;	No, risk accepted
A2: Cryptographic failure	Unlikely	Severe	Moderate	Encrypt JWT token; Use DTO objects for communication with front-end; Use the TLS protocol with forward secrecy to encrypt all data in transit;	No, risk accepted
A3: Injection	Very unlikely	Severe	Low	Run a SQL Injection Test or XSS Test; Positive server-side input validation	No, risk accepted
A4: Insecure Design	Likely	Low	Moderate	Apply threat modeling methods to critical flows;	No, risk accepted
A5: Security Misconfiguration	Moderate	Low	Moderate	Enable CRSF;	No, risk accepted
A6: Vulnerable and Outdated components	Unlikely	Low	Low	Follow security news and keep product up to date;	Yes
A7: Identification and Authentication failures	Moderate	High	High	Implement multi-factor authentication; Perform weak password checks; Limit or progressively delay repeated login attempts after failure;	No, risk accepted

A8: Software and Data Integrity Failures	Unlikely	Moderate	Moderate	No unsigned or unencrypted serialized data should be sent to untrusted clients	No, risk accepted
A9: Security Logging and Monitoring Failures	Very unlikely	High	Low	No logging in the application	N / A
A10: Server side request forgery	High	Moderate	Moderate	Improve framework implementation	No, risk accepted

Broken Access Control

The broken access control vulnerability is giving access to pages that should not be reachable by all visitors because this makes the application open to attacks. In order to secure your application the se of multi-factor authentication is a must. Also the accounts that are not used for a long period of time should be removed. The access page to the admin or other roles accounts should not be visible to all other users of the application. In BudgetSimple UUID's are used which secure the direct access of objects, but multi-factor authentication is not used, accounts are not removed and admins of the application login from the same page as other users which makes this vulnerability with a high risk.

Cryptographic failure

Cryptographic failure is connected with sensitive data exposure or system compromise. In BudgetSimple application passwords of users are encrypted, no informaation is stored in local storage in the front-end and the JWT token is stored in HTTP-only cookie which is considered to be one of the most secured ways of storing a JWT token. Also DTO objects are used on some places that reveal only some data. However, the JWT token should be encrypted before saving it in the cookie so that the information inside it cannot be read and it cannot be used. Also DTO objects with selective data should be used all over the application where there is data transfer to or from the front-end.

Injection

Injection attacks happen when an attacker sends invalid data to the web application with the intention to make it do something that the application was not programmed to do. One of the things that is commonly happening is when dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter which is not happening in BudgetSimple.

Insecure Design

As this is of the first designs that I have build, I consider that is likely to be attacked. However, in the future ways to secure the software design that are suggested by OWASP website are:

- Establish and use a secure development lifecycle with AppSec professionals to help evaluate and design security and privacy-related controls
- Establish and use a library of secure design patterns or paved road ready to use components
- Use threat modeling for critical authentication, access control, business logic, and key flows
- Integrate security language and controls into user stories

Security Misconfiguration

One of the most common ways to have this vulnerability is by using unpatched flaws, default configurations, unused pages, unprotected files and directories, unnecessary services. In BudgetSimple application a way to prevent it is by enabling CSRF which is preventing unauthorized commands that are submitted from a user that the web application trusts.

Vulnerable and Outdated Components

This vulnerability can be prevented by using softwares that check the secuity of the application and always keeping updated all the components, frameworks, etc. with the latest security standards. This is planned to be done with the BudgetSimple application in the future.

Identification and Authentication failures

Again a way to prevent this vulnerability is by using multi-factor authentication. Also passwords should be checked when created and the ones like “password123”(easy to guess) should not be allowed to be used. Also if a user fails to login a couple of times there should be some waiting time between othe authentication attempts. Using these features will make the user feel safer in the application too. However, BudgetSimple doesn’t have them and this makes this vulnerability with high risk.

Software and Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. The use of digital signatures or similar mechanisms to verify the software or data is from the expected source and has not been altered is a way of preventing it.

Security Logging and Monitoring Failures

There is no logging used in BudgetSimple application which makes it with low risk.

Server-Side Request Forgery (SSRF)

With these attacks, hackers get unauthorized access to sensitive configurations and make malicious attacks that appear to originate from the organization hosting the vulnerable application, causing potential legal liabilities and reputational damage. By not sending raw responses and whitelist any domain or address that your application accesses.

Conclusion

BudgetSimple application has some security implemented in it but still I have a lot more to develop in order to make it fully secured accoring to the OWASP report so that users of the application can use it and bes ure that their data and information is safe.