SECURITY REPORT

Simple Homes Project S3

Individual Project Semester 3

Contents

1. Introduction 2
2. OWASP Top 10 Risks In 2021 3
   - A01:2021 – Broken Access Control 3
   - A02:2021 - Cryptographic Failures 3
   - A03:2021 – Injections 3
   - A04:2021 – Insecure Design 3
   - A05:2021 – Security Misconfiguration 4
   - A06:2021 – Vulnerable and Outdated Components 4
   - A07:2021 - Identification and Authentication Failures 4
   - A08:2021 - Software and Data Integrity Failures 4
   - A09:2021 - Security Logging and Monitoring Failures 4
   - A10:2021 - Server-Side Request Forgery 4
3. Project Security Analysis 5
4. Conclusion 6

1. Introduction

This is a security report outlining concerns and remedies for the Simple Homes website based on the OWASP Top 10, a ranking for the top 10 most critical web application security risks ranked by experts from around the world.

The report is going to determine how secure the Simple Homes web application is by evaluating it for each of the top threats and will also explain what can and will be done to lower the risk factors, so that the end users will have a better experience and will not have to worry about the security of their private data.

The first chapter provides a brief description regarding all the security risks and outline some ways to prevent them and make sure they do not occur in web applications

The second chapter depicts a table which analyzes the Simple Homes web application and presents the security risks and their potential impacts on the website. It also provides some possible fixes for these issues.

The final chapter presents a conclusion based on the report analysis and provides some recommendations that are possible to strengthen the security of the web application for future releases.

2. OWASP Top 10 Risks In 2021

A01:2021 – Broken Access Control

Broken access control occurs when users can perform tasks or view information that is not meant to be seen by them. This can happen when proper account authorization is not present in a web application or if the website is vulnerable to force browsing and parameter tampering. All of this can be prevented by implementing a secure token handling system that properly invalidates session identifiers upon log out. Another prevention measure is to deny access to non-public resources by default and requiring another way of authentication and authorization before proceeding.

A02:2021 - Cryptographic Failures

Cryptographic failures can occur when sensitive data such as passwords or credit card numbers are improperly encrypted or stored. Failing to deal with this security risk leaves all the users on the website vulnerable to all sorts of issues. In order to prevent such things from happening the website should not transmit any potentially vulnerable data in plain text but should instead encrypt it in a secure way to make sure that it cannot be decrypted or accessed by anyone. Encrypting the passwords multiple times with a secure framework might also be a good idea just in case the passwords of the users are somehow leaked.

A03:2021 – Injections

Injections are a way of tampering with the database of a web application that does not have proper user input validation or sanitization. Injections can be used to pass unwanted data in search parameters in order to gain access to the database which can lead to data leaks or in some cases complete breaking of the website. Prevention of injections can be done by using secure APIs which validate the input of users at any time or to migrate to Object Relational Mapping.

A04:2021 – Insecure Design

Insecure design can occur all the way in the planning process and is important to be resolved as soon as possible, because the more the web application develops, the harder it will become to fix the design. Using insecure APIs or missing user input bounds can lead to an insecure design and once these issues have been found it is very hard to resolve them as they could already be playing a major part in the structure of the application itself. The prevention of such problems comes from the planning itself. A lot of research has to be done before deciding on the structure of the project. The web application has to be designed with a security-centric mindset and is also has to be scalable.

A05:2021 – Security Misconfiguration

The security of a web application is considered misconfigured when a proper security configuration process is not present. This can mean anything from out-of-date software to presence of unnecessary or non-functional features installed which can be exploited by the end user. Removing or uninstalling unused features or frameworks and using automated way of checking whether all the settings and configurations are properly configured.

A06:2021 – Vulnerable and Outdated Components

Components that are not supported anymore or are known for having insecure implementations and features should be avoided at all costs. It is better to opt for more recent and well-known frameworks and APIs which are frequently updated and maintained by reliable and competent software engineers. That way all the services used by the web application will always be as secure as possible.

A07:2021 - Identification and Authentication Failures

These failures can occur if the website permits weak or well-known passwords which are vulnerable to a number of different attacks such as brute forcing or dictionary searching. It can also happen if the passwords of the users are stored as plain text or poorly encrypted or if the session ids are improperly invalidated. Implementing a system that forces the users to use longer and more complex passwords can be a good solution to some of the risks mentioned above.

A08:2021 - Software and Data Integrity Failures

Software and data integrity failures can occur if the application uses insecure plugins, APIs, or libraries. It can also happen if all external sources are not tested and checked for any vulnerabilities than may occur when a new update is available.

A09:2021 - Security Logging and Monitoring Failures

Logging can be used to monitor the activity of the web application so that if anything happens, the problem can be easily identified and addressed. That is why it is crucial that there is a proper logging system which keeps track of important events like transactions and logins at all times. Another important aspect is generation of a proper log message in the event of an error.

A10:2021 - Server-Side Request Forgery

Server-Side Request issues can happen when the web application is getting information from the server without proper validation of the URL provided by the user. Proper validation and sanitization of all the data supplied to the client is one of the best ways to deal with this issues.

3. Project Security Analysis

	Likelihood	Impact	Risk	Actions possible	Planned
A1: Broken access control	Low	Moderate	**Low**	No crucial data is stored, tokens are properly invalidated upon log out.	N/A
A2: Cryptographic Failures	Low	Severe	**Moderate**	Passwords are properly encrypted	N/A
A3: Injections	Low	High	**Moderate**	Further filtering of user input	Risk accepted
A4: Insecure Design	Moderate	Low	**Low**	If the design is very insecure the application has to be restructured	Risk accepted
A5: Security Misconfiguration	Low	Low	**Low**	N/A	N/A
A6: Vulnerable and outdated components	Low	Low	**Low**	Web application uses up to date libraries and frameworks	N/A
A7: Identification and authorization failures	High	Low	**Moderate**	Password filtering can be implemented	Yes, will be implemented
A8: Software and data integrity failures	Low	Low	**Low**	External APIs and libraries are not present except for a trusted Email service	N/A
A9: Security logging and monitoring failures	High	Low	**Moderate**	Implement a logging service	Risk accepted
A10: Server-side request forgery	Low	Low	**Low**	Proper user validation is used for important pages	N/A

4. Conclusion and Recommendation

In conclusion, based on the risks displayed above, the Simple Homes web application appears to be relatively secure considering the scale and type of data that is used in it.

Most security issues show a low risk factor which is calculated based on the likelihood of that issue surfacing multiplied by the impact it could theoretically have on the application.

If the web application starts to scale and a payment system is implemented where the users can pay their rent through the website, then further security measures will be needed like card number encryption and more thorough account security. But for now the security if sufficient for the amount and type of data that is being used.

For future projects I would probably focus a bit more on a secure design since security was not the main focus of the planning process which, in retrospect is not the most ideal way of constructing the base of an application. I would also probably try to figure out a more secure way of encrypting and storing sensitive data like passwords and implement a logging feature so that it would be easier to keep track of important events. That way if something was to go wrong there would be an easy way of checking what caused it and thus the fix would be easier and quicker to deploy.