Security by design

Author: Denny Cox
Class: S6-RB10
Semester: 6

Table of contents

Security by design means thinking about the security of your system right from the start and not to implement it as an afterthought. For this I followed the security workshop where a few security risk examples where shown. After this I worked out the 2021 OWASP top 10 and explained what is implemented in Kwetter to prevent these security risks.

Workshop Security

During the security workshop from the company WhiteHats, we discussed the following examples of important security vulnerabilities and talked about how to prevent them:

SQL injection - use an ORM in the framework that is used. For my project I will use Entity Framework.

Cross site scripting - use output encoding (use protection from used framework).

Insecure file upload - only accept whitelisted file types, restrict file size, generate file names yourself, store files only in database, use library to verify zip files.

Insecure deserialization - don't deserialize user input if possible, otherwise apply integrity checks.

Padding oracle - encrypt using AES in GCM mode.

Closing tips from workshop:

- View Microsoft SDL: https://www.microsoft.com/en-us/securityengineering/sdl
- Implement security tests
- Follow OWASP top 10
- Follow security recommendation from the used framework
- Use tools like SonarQube in CI/CD
- Write security recommendations and implement them

OWASP top 10

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad agreement about the most critical security risks to web applications. The following subjects are the security risks from the OWASP top 10 list from 2021.

Broken Access Control

A scenario in which attackers can access, modify, delete, or perform actions outside an application or systems' intended permissions. This is prevented by using a system that has user roles.
In the case of Kwetter Microsoft Identity is used; a widely used, proven system that has this functionality built in.

Cryptographic Failures

A security risk where attackers are able to view sensitive data. This is prevented by implementing secure communication so sensitive information can only be viewed by authorized parties.
In the case of Kwetter https connections are used and passwords are hashed.

Injection

A scenario where untrusted data is sent to a code interpreter through a form input or some other mode of data submission. This is prevented by keeping data separate from commands and queries.
In the case of Kwetter Entity Framework is used as an ORM. This is a widely used, proven system that is a good SQL injection prevention mechanism.

Insecure Design

Security risk that exists because of design and architecture flaws. This is prevented by using threat modeling, secure design patterns and principles.
In the case of Kwetter unit and intergration test are made to validate critical flows of the system. Also Entity framework is implemented; a widely used, proven system.

Security Misconfiguration

A scenario where weaknesses found in web applications that are misconfigured can be exploited. This is prevented by using a different configuration for development and production environment and security scanners.
In the case of Kwetter no default cridentials are used in production environment, passwords are encrypted or stored at a safe place.

Vulnerable and Outdated Components

A security risk that occurs when software components in a system are unsupported, out of date, or vulnerable to a known exploit. This is prevented by removing unused dependencies and files, installing components only from trusted channels and maintaining the used components.
In the case of Kwetter the mentioned preventions are implemented.

Identification and Authentication Failures

A scenario where attackers may be able to exploit identification and authentication failures by compromising passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities. This is prevented by implementing multi factor authentication if possible, implementing weak password check and use a server-side session manager.
In the case of Kwetter a weak password check is implemented, and sessions are implemented with web tokens.

Software and Data Integrity Failures

A security risk that could occurs when critical data used by the application is not verified, attackers can tamper with it, which can lead to quite serious issues, such as introduction of malicious code into software. This is prevented by making sure that dependencies and third-party libraries use only trusted repositories, the source code of your application is reviewed to avoid unwanted configuration changes or introduction on malicious code into the software.
In the case of Kwetter the used dependencies use only trusted repositories.

Security Logging and Monitoring Failures

A security risk that could occur when the logging and monitoring of the system is not implemented correctly so that malfunctions or errors can go unnoticed for a long time. This is prevented by having logs that containing relevant data and having a monitoring and alerting system.
In the case of Kwetter The ILogger and the ILoggerFactory are implemented to log important system errors.

Server-Side Request Forgery

A security risk that allows an attacker to make requests to any domains through a vulnerable server, it occurs when an attacker has full or partial control of the request sent by the web application. This is prevented by whitelisting the hostnames (DNS name) or IP addresses that the application needs to access and check that the received response data is as expected.
In the case of Kwetter the response data is checked before sending it back to the client.