SECG4 - SECURITY

Project: Security Chat

29 MAY 2022

YOUSSEF EL MAJDOUL 56172

YOUNES OUDAHYA 54314

Table of Contents

Introduction

The project consists of creating a secure chat in which users can discuss with each other in a private way and where the information is secure. In this project it is a question of registering or logging in as a user in order to enter the chat space to be able to discuss with other users while having a contact control.

Framework

For this project, we decided to start with a framework known as "Laravel".

This framework has a large community of development companies and developers and so, several optimizations are already included.
The features of Laravel allow to use everything safely, all data are cleaned if necessary.

Security

- Passwords are stored encrypted in the database and are hashed 12 times with a default salt by the "Hash::make()" method. This is a commonly used method in Laravel using the bcrypt hash algorithm. Bcyrpt allows to reduce the number of passwords per second that an attacker could hash during the building of a dictionary attack. The key used is in the ".env" file in APP_KEY, it is generated by means of php artisan key :generate.
- Messages are encrypted before insertion in the server database through Laravel which provides an encryption and decryption method using openssl to provide AES-256 encryption. The cookies created by Laravel are encrypted and signed, they will be invalid if the user tries to modify them.
- Since Laravel is a recent framework with recent updates, all the vulnerabilities discovered at the moment are corrected. It is necessary to keep up to date with the advances of Laravel to be always up to date from a security point of view.
- All debug outputs that may expose system details or access sensitive configuration values are disabled. Laravel simplifies this in the ".env" file by simply disabling APP_DEBUG=false
- There is no risk of SQL injection because all the queries here are prepared automatically when using the methods of the DB interface. There is also no risk of XSS vulnerability because users cannot enter any code since the < > tags are replaced by the html character and therefore it will be impossible to insert a script.
- After a few failed connection attempts, the user is blocked for a certain period of time, this prevents too fast requests and brute force.
- A captcha is implemented on all login, registration and forgotten password forms to prevent brute force attacks where a malicious actor could use automated means to try thousands of password combinations to access an account.
- All forms include @csrf to protect against CSRF flaws that allow a user to execute other requests from the application without being intended.

- Passwords must follow a specific format: at least 8 characters, one upper case, one lower case, one special character and one number. This avoids the dictionary attack.
- Invalid user actions are rejected without being analyzed, even though they are present in the "/storage/log" file, which can be configured in the server and allows for traceability of an intrusion attempt by a user.
- The use of POST when logging in or registering ensures that the user cannot retrieve the information in the URL and modify it in an attempt to retrieve data from other users.
- Laravel provides middleware, meaning a bridge between a request and a reaction, to prevent an unconnected person from doing non-permissible actions.
- To ensure non-repudiation, the use of the ECDSA algorithm with p-256 for the key signature for each message.
- The use of different external resources:
    - Spatie
    - RSA 4096: for message encryption
    - SHA256: for hash
    - SubtleCrypto: functions for security features -> authentication in the system
    - RSA-OAEP
- Messages sent by the user are signed and verified by the recipient so that the messages are not altered during transfer, this ensures that a user sending a message to another user is actually delivered and sent by the real user.
- The messages are also secured using end-to-end encryption, this ensures that only the communicating users can read the messages, this prevents reading or modification of data other than by the user concerned. Messages are encrypted by the sender and decrypted by the receiver.
- The addition of a ssl certificate will ensure the security of information between the client and server. The certificate allows an encrypted connection and a passage in https and so the cookies will also pass in httpsOnly, wich are more secure than simple cookie, it is used to prevent a Cross-Site Scripting exploit from gaining access to the session cookie and appropriate the victim's session.

Conclusion

The data are secured on the client side as well as on the server side, however, to secure the whole system it is necessary to secure the server with an SSL certificate issued by an authority.

The Laravel framework greatly facilitates this, but it is necessary to regularly update our application to avoid breaches.