SECG4 : Scientific Report - Secure

Group

- 56459 - Antoine Ghigny
- 55019 - Ian Cotton

Contents

1 Introduction

The goal of this project is to implement a small but secure client/server instant messaging system per groups of two students. The client application hosts the sessions of users and allows them to talk to each other, and to the server. A registered user has the following characteristics like a credentials to allow authentification, a unique login to allowing to be identified by other users, « several "pieces of information" allowing to securely communicate with the server and other users. Their generation, storage and revocation are left to your discretion and a list of contacts who accepted to talk to him. Users should be able to register, login, logout, add and delete a contact. Contacts can communicate with each other and these must be whether they are logged in or not.

2 What security does our application provide?

One of the main sources of inspiration for paying attention to website security is to follow a number of principles advocated by OWASP. The Open Web Application Security Project, or OWASP, is an international non-profit organisation dedicated to web application security. The OWASP Top 10 is a regularly updated report that outlines web application security concerns, focusing on the 10 most critical risks

2.1 SQL Injection attacks

SQL injection attacks occur when untrusted data is sent to a code interpreter through form input or other data submission to a web application. For example, an attacker could enter SQL database code into a form that expects a username in plain text. If the input to this form is not properly secured, the SQL code will be executed. This is called an SQL injection attack.

We use the Eloquent model provided by Laravel, which protects us against SQL injections in most cases by transforming our queries into prepared queries. However, there are possibilities of injections despite the use of the template (using column names from a user, validation rules also using an element retrieved by the user or using the DB::row function). We have therefore taken care of this in our queries in order to guarantee security at this level.

We can prevent mass assignment by adding explicitly the fields that a model contains by using protected properties, "fillable" or "guarded":

2.2 Broken authentication

Vulnerabilities in login systems can give attackers access to user accounts and even the ability to compromise an entire system using an administrator account. For example, an attacker can take a list containing thousands of known username and password combinations obtained in a data breach and use a script to try all of these combinations on a login system to see if any work.

A person trying to log in multiple times without succeeding is considered potentially fraudulent, so we decided to implement a security feature that will prevent the person from logging in for 3 minutes before trying again. Laravel allows us to do this automatically by adding variables in the login controller that indicate the maximum number of attempts possible and the number of minutes to wait before the user can try again.

2.3 Broken access control

Access controls are designed to prevent users from acting outside their intended permissions. If these controls have vulnerabilities or are missing, users can perform actions beyond their permissions. This can allow attackers to steal information from other users, modify data and perform actions as other users.

We have only set up one role on the site, so it is not possible to use security holes to access other roles. We have also ensured that users can only access the conversations of people they are friends with. This is achieved by adding middleware that governs certain rules that prevent access to other conversations.

2.4 Cross-site scripting (XSS

Cross-site scripting (XSS) is a type of website security flaw that allows content to be injected into a page, thus causing actions to be taken by web browsers visiting the page. The possibilities of XSS are very wide since the attacker can use all the languages supported by the browser (JavaScript, Java...) and new possibilities are regularly discovered especially with the arrival of new technologies such as HTML5. For example, it is possible to redirect to another site for phishing or to steal the session by retrieving cookies.

Laravel automatically generates a CSRF token for each user. This token is used to verify that the authenticated person is the person who is actually making requests to the application. Since this token is stored in the user's session and changes with each new session regeneration, a malicious application will not be able to access it

Whenever a user provides a form the @CSRF tag is then added to it, which automatically checks that the token in the input matches the token stored in the session. If they match, then we know that the authenticated user is the one making the request. This has been implemented in the authentication phases.

In vue.js, when making POST and GET requests to axios to retrieve json, these lines have been added to the headers, which do the same thing as the @CSRF tag.

2.5 Security misconfiguration

Security misconfiguration is the most common vulnerability on the list, and is often the result of using default configurations or displaying overly verbose errors. For example, an application may present the user with overly descriptive errors that may reveal vulnerabilities in the application. This problem can be mitigated by removing any unused features in the code and ensuring that error messages are more general.

We have made it impossible to bypass https, so any user attempting to access a page in http will be automatically redirected to its https equivalent. We have done this by configuring the htaccess file. Only the public folder is accessible to users, so it is impossible to access confidential information such as credentials... We have also not published the environment files and the database on the repository.

2.6 Sensitive data exposure

If web applications do not protect sensitive data such as financial information and passwords, hackers can access this data and sell it for malicious purposes. A popular method of stealing sensitive information is to use a man-in-the-middle attack.

We encrypt the messages when they are sent and we decipher them only when they are displayed. As a result, someone who could (miraculously) intercept them would not be able to read them. In addition, our implementation of SSL and TLS allows us to check whether the message has been alternated or not.

Private channels have been set up with pusher to ensure end-to-end encryption. This means that no one except authorised subscribers can read the payload, not even pusher. These channels provide end-to-end authenticity

We used Laravel methods to encrypt/decrypt the messages:

- (Crypt::encryptString)
- (Crypt::decryptString)

For the implementation of SSL and TLS, we created and used a certificate via XAMPP

2.7 Insufficient monitoring and protection against attacks

If web applications do not protect sensitive data such as financial information and passwords, hackers can access this data and sell it for malicious purposes. A popular method of stealing sensitive information is to use a man-in-the-middle attack.

TLS SSL protects us from man-in-the-middle attacks. SSL/TLS encrypts communications between client and server to ensure that if traffic is intercepted, it cannot be decrypted. Information such as passwords are hashed in the database and encrypted during communication between the client and server (using the same protocol.

Private channels have been set up with pusher to ensure end-to-end encryption. This means that no one except authorised subscribers can read the payload data, not even pusher. These channels provide end-to-end authenticity

2.8 Cross site request forgery (CSRF)

In information systems security, the cross-site request forger or CSRF is a type of vulnerability in web authentication services. The purpose of this attack is to transmit to an authenticated user a falsified HTTP request that points to an internal action on the site, so that he executes it without being aware of it and using his own rights. The user thus becomes an accomplice to an attack without even realising it. As the attack is user-driven, a large number of authentication systems are bypassed.

See 2.4

2.9 Using components with known vulnerabilities

It is imperative to know the versions and vulnerabilities of the components you use. Including nested dependencies

We have checked the existing vulnerabilities for the different components we use. These do not currently have any security vulnerabilities.

3 Conclusion

Today, security is a major issue for all the actors involved. It is no longer confined solely to the role of the IT specialist. Its long-term objective is to maintain the confidence of users and customers. The medium-term objective is to ensure the coherence of the entire information system. In the short term, the aim is for everyone to have access to the information they need.

Security is therefore an extremely important and essential step when developing an application. It takes a predominant place in the development of the application, especially if it is an online application, and this is normal given the number of things to control. It is necessary to protect the personal information of the users, to check that this information has not been modified,... It took us more time to implement the necessary security than to develop the features of our chat.

4 References

Injections SQL : https://www.letecode.com/sql-injections-en-laravel

OWASP : https://actualiteinformatique.fr/cybersecurite/quest-ce-que-lowasp