Leong Paegh-Hing 56133

Arturk Yohan 56514

Report on the security of the application

Table of contents

1) Introduction

This report will explain what is our application and the security implementation that has been chosen.

The security of an application is a really point because it protect the user private data in the internet or just to protect the application itself from attackers that try to attack or get data from the data base that contains sensitive information.

2) Web Application

Our web application is a simple application where user can communicate with his friends. It ‘s a little private chat application. The user need to register to enter in the application. To communicate with someone, the user need to send a friend request to the user that he want communicate with it. The other user can accept or refuse the friend request. If the other user accept the friend request, then both user can communicate with each other.

If one of the two user deletes the other from his friend list, then both user can’t communicate with each other anymore.

3) Security implementation

Now the main point of this report, the security of our web application.

To explain our security implementation, we will answer to 15 questions. For each question, the problem and our solution to prevent from this problem will be explain.

3.1) Confidentiality

Problem

The data that a user sends on the internet must be confidential because those data can be sensitive information. If the data are not protected, then someone can easily stole those information. The data must also be protected in data base because someone from the inside (like an administrator) could steals those data.
In addition to data, sensitive requests must also be protected to prevent a malicious person from doing things against the user's will. The possible attack can be a CSRF-attack or replay attack.

Solution

To transmit data, we will use https. When the client want to communicate with the server, the client and the server will do TLS Handshake. The TLS handshake will create a symmetric key that only the client and the server know. The symmetric key will be used to encrypt and decrypt the data. So if someone has not the key, then he can’t read those data.

On the server side, the sensitive data (in our case, the message) will also be encrypted by a symmetric that only the server know so nobody, even the system administrator can read those data.

When there is data transmitted with the websocket, the data are encrypted with an encryption key. So even if someone intercept the data, he will not be able to decrypt the data send to the client.

To protect against CSRF-attack, we used a csrf token that will be check each time the client send a request to the server. For the case of Laravel, Laravel automatically generates a CSRF token for each active user session managed by the application. This token will be used to verify that the authenticated user is the person actually making the request to the application. Since this token is stored in the user's session and changes each time the session is regenerated, a malicious application is unable to access it.

For the replay attack, each time a user do a request to the server, the request will add the current time. When the application receives the request, it will compare the receive time and the current time. If the difference between those two values is smaller than 30 or 60 then the request is considered valid. But if the value is greater than 30 or 60, then an error 403 page will be return.

3.2) Integrity of stored data

Problem

The integrity of stored data protect against data loss or a data leak because otherwise the data in the database will not be more consistent.

Solution

In term of integrity of stored data, it difficult to protect the physical integrity but we can at least protect the logical integrity. In our application we have protected the logical integrity through the 3 following point :

- For the first one, we ensure that every table has a primary key and there is no field that can be null.
- The second one is about foreign key. Every foreign key of our data base refer to a primary key. The main table of our data base is the users table and every other table refer to the primary key of the users table. If a user is deleted, then every foreign key that refer to the row that has been deleted, will also be deleted to avoid reference to some data that doesn't exist anymore.

- For the third one, we ensure that each field of each table is set to only one domain.

3.3) Non-repudiation

Problem

When we try to communicate with a server, there is no guarantee that the response we receive is from the server, it is possible that there is a man in the middle attack. If that happen, then the user could compromise his sensitive data because he believes that he communicate with the server.

Solution

To protect against Man In The Middle attack, a certificate can be used to proof the identity of the entity that respond. With https, when the client try to communicate with the server, the server will respond him by sending his certificate to proof that the server is the server.

3.4) Authentication scheme

Problem

A strong and proper authentication scheme is necessary because it protects users from having their accounts stolen. If the authentication scheme is not strong enough then the user's account can be comprise.

Solution

In our application, a user need to provide his email and his password to authenticate. If the user fails more than 3 times then a decay will force the user to wait 2 minutes before trying again.

To authenticate the password, the application will hash the given with the salt that is stored in the data base. If the hash correspond with the one in the data base then the user is authenticated. The salt is used to avoid that 2 same password has the same hash.

The hash used for the password is Bcrypt because it's good for password hashing because it reduces the number of password by second an attacker could hash when crafting dictionary attack

3.5) Secrecy

Problem

We need that someone has something that ensure he identity without telling the secret. This is what is call zero knowledge proof.

Solution

Our application don't rely on secrecy because to ensure the identity of the user, he only need to give his email and password and nothing more.

3.6) Injection vulnerability

Problem

Injection can be dangerous because someone could easily get the data from a data base or even execute script on the client for malicious intention.

Solution

For the data, our application used query builder to access the data base. Query builder uses PDO parameter binding to protect the application against SQL injection.

When the server receives a request, the content is sanitize with the help of a custom middleware that sanitize the content.

On the client side, double bracket are used to escape the content.

3.7) Data remanence attack

Problem

Data Remanence represents residual information of the data remained after deletion. Various file handling operations such as the reformatting of storage, deletion operation may result in data remanence. Such operations can cause disclosure of user's sensitive information.

Solution

There is no data remanence because all the variable are always deleted.

3.8) Replays attacks

Problem

Replay attack can cause problem because an attacker don't even need to decrypt the content to perform this kind of attack. The attacker will simply catch the encrypted and then resend it to the server. With this attack, the attacker can easily be authenticated without even knowing the credentials of the victim.

Solution

Each time the client communicate with the server, the server will check the timestamp of the request and compare it to the current timestamp. If the difference between the 2 timestamps is too big then the server will reject the request. And the same technique is used on the client side when the server respond to the client.

3.9) Request forgery

Problem

Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user. A CSRF attack exploits a vulnerability in a Web application if it cannot differentiate between a request generated by an individual user and a request generated by a user without their consent.

Solution

In our application, Laravel will automatically generates a CSRF token for each active user session. This token is used to verify that the authenticated user is the person actually making the request to the application. Since this token is stored in user's session and the token changes each time session is regenerated, a malicious application is unable to access it. In each form or each request, the client will include the CSRF token in the request.

3.10) Monitoring

Problem

If an application don't monitor enough the user activity then it will be hard to prevent a potential attack because an attacker is always an authenticated user.

Solution

Each time a user do a request that cause an error, the user id, the suspicious action and the time is registered in data base for further analyze.

3.11) Components with know vulnerabilities

Problem

Because most of the dependencies used are open source, it allows malicious users to analyze the packages and find some ways to exploit some vulnerability that has not been patch yet.

Solution

The Laravel version used is the 9.11 and the PHP version is 8.1.6, the latest version.

3.12) System

Yes the system is updated.

3.13) Broken access control

Problem

If there is no access control, than an attacker can exploit this to access unauthorized functionality and data such as other user's account, view sensitive data change access right.

Solution

The only resource that a user can access are the message that the user has with someone by giving the id of the friend. In our application, even if guess the id of the friend, the id he guess is a fake one while real one is keep secret.

3.14) Broken authentication

Problem

If the authentication scheme is not strong enough or if the session management is bad, then an attacker can easily take the rights of another user.

Solution

The password stored in the data base are hash with Bcrypt with a cost of 12. A salt is also used to avoid that 2 same password has the same hash.

The user is also limited in the number login attempts, if he fails more than 3 times then he will be lock for 2 minutes before he can try again.

If user close or do nothing during 15 minutes, the server will disconnect automatically the user.

3.15) General security features misconfigured

Problem

An attacker can exploit a vulnerability because the application keep the default configuration.

Solution

Our application has deactivated the app debug option so the stack trace will not be visible anymore.