# SECG4

Secure software development and web Security Project

53204 – ZEDZIAN PAWEL

53203 – BEN ALLAL ANAS

2021-202

# Table of Contents

# Introduction

In this report we will be explaining all the choices regarding the security of this project.

## Server

For this project we've chosen to code in the Laravel framework using PHP, where we have a Controller/View/Route organization, in there we define all the different actions that the user can choose.

With this web applications, authentication is managed by sessions which take the input parameters such as emails/usernames and passwords, for user identification. If these parameters match, the user is said to be authenticated.

After identification, the user can upload, download, or delete the desired files in a secure way, and he is the only one who can see them because they are crypted.

For storage of the password, we crypt the password with Bcrypt method with a Brainstud\FileVault\Facades\FileVault library, in addition this method is using salt to protect the user from rainbow table attacks, Bcrypt is an adaptive function, we can increase the number of its iterations to make it slower, yet it continues to be resistant to brute force attacks despite the increase in computing power.

Blowfish is a block cipher algorithm notable for its relatively expensive key establishment phase. Bcrypt uses this property and goes further, it uses key establishment algorithm named Eksblowfish (for Expensive Key Schedule Blowfish). In this algorithm, a first phase consists in creating the subkeys using the key and the salt. Then a number of rounds of the standard blowfish algorithm are applied alternately with the salt and the key. Each round begins with the state of the previous round's subkeys. This does not make the algorithm any more powerful than the standard version of blowfish, but one can choose the number of iterations which makes it arbitrarily slow and helps deter rainbow table and brute force attacks.

For the validation of the password we use the option "require", we use the Password Validation Rule Object .The ability to set default password rules means that you can centralize the expected validation behavior for a password by setting them in a service provider (e.g., AppServiceProvider) The default values are stored and you can retrieve them

later in a validator with the Password::defaults() method, basically allows to have conditions to validate the password min of uppercase character etc...

Once the user is validated, we create the user, then we recover the email, name and the password which will be hashed via the Make method of the Hash Laravel class.

To be able to access our server the user must be able to have an authorization (to be "logged"): When he "registers" he receives his identifier (name, email, password), his password is directly hashed thanks to a hash function and stored in the database.

# Functionality

The user can choose to upload/download/delete a file, for a new user we call the Storage class that create a folder location in the public folder with his ID.

All files are encrypted for better security. For encrypting, we use the Advanced Encryption Standard (AES) with a key of 128, 192 or 256 bits. When the user will choose to download, the file will be decrypted for the user to consult it.

We encrypt with key 32 byte long for the AES-256-CBC.

# Client

Each user receives his own folder and therefore is unable to wander through the folders of other users for a better security/privacy: For example, user 1 will have the folder public/1 and user 2 will have the folder public /2.

The names of the files are encrypted to avoid revealing personal information. All web routes are blocked for any unauthenticated user, the only route for "guests" are "/", "/login" or "/register ».