

Google 2fa tutorial

In this tutorial I will be showing how you successfully add Google's 2 factor authentication to a Laravel project

For this tutorial I assume you already know how to create a Laravel project, or maybe you already have one, in homestead. Also I assume you know how to launch a project to Heroku. If you don't I suggest you search those before doing this as it makes deployment to production a lot easier. I hope that makes sense. Now let's implement authentication and Google's 2fa authentication.

Step one, add Laravel/ui to your project.

First run `composer require laravel/ui`

```
vagrant@homestead:~/Google2fa$ composer require laravel/ui
1/2:  http://repo.packagist.org/p/provider-latest$5900ea15e6a5d5e090bdfc44d74c8be70c01e454fa8aabbf3d11b0b1afd2c6.json
2/2:  http://repo.packagist.org/p/provider-2020-04$65d8202d48b2c0913ca7b8cd8c95077825443a1956463aff50a388ae4f62fd45.json
Finished: success: 2, skipped: 0, failure: 0, total: 2
```

This will setup your basic routing for your authentication routes, controllers and models.

Now run `php artisan ui vue --auth`

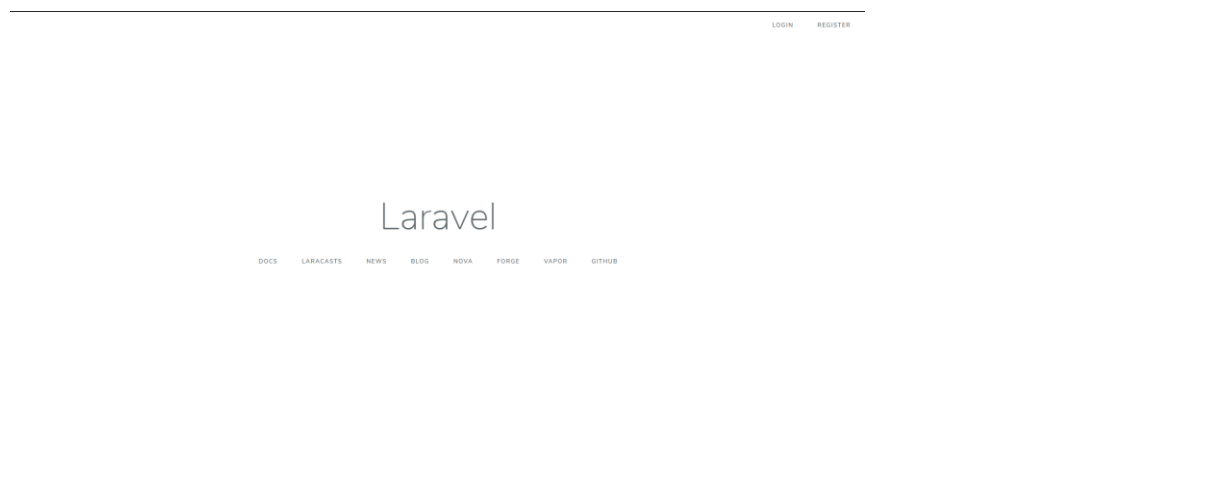
```
vagrant@homestead:~/Google2fa$ php artisan ui vue --auth
```

This will install a layout, registration and login view as well as routes.

Your application will now ask to run `npm install` and `npm run dev` to compile your recently added scaffolding

```
Vue scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
Authentication scaffolding generated successfully.
vagrant@homestead:~/Google2fa$ npm install && npm run dev
```

Do as it says load your application and you should be greeted with this.



Notice the login and register buttons at the top, congratulations you have successfully enabled authentication on your Laravel project.

Step two, let's start adding Google2fa to our project.

Now let's start building our Google 2fa authentication.

We will start with a couple of composer commands:

```
composer require pragmarx/google2fa-laravel
```

After composer has done it's thing run

```
composer require bacon/bacon-qr-code
```

Now let's publish our config file

```
php artisan vendor:publish --provider=PragmaRX\Google2FALaravel\ServiceProvider
```

If you did this correctly this should show up:

```
Copied File [/vendor/pragmarx/google2fa-laravel/src/config/config.php] To [/config/google2fa.php]
Publishing complete.
```

Now we are going to go move away from our command line for a bit and change some things in our actual application.

Go into your RegisterController and add this class to the top of it.

```
use Illuminate\Http\Request;
```

Then we go into our controller and define the register method. This should go directly under these lines

```
class RegisterController extends Controller
{
    /**
     |-----
     | Register Controller
     |-----
     |
     | This controller handles the registration of new users as well as their
     | validation and creation. By default this controller uses a trait to
     | provide this functionality without requiring any additional code.
     |
     */
}
```

Let's define it using this code:

```
public function register(Request $request)
{
    //Validate the incoming request using the already included validator
    method
    $this->validator($request->all())->validate();

    // Initialise the 2FA class
    $google2fa = app('pragmarx.google2fa');

    // Save the registration data in an array
    $registration_data = $request->all();

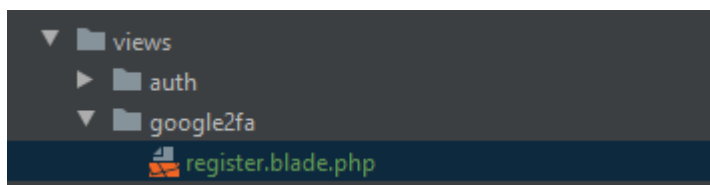
    // Add the secret key to the registration data
    $registration_data["google2fa_secret"] = $google2fa-
>generateSecretKey();

    // Save the registration data to the user session for just the next
    request
    $request->session()->flash('registration_data', $registration_data);

    // Generate the QR image. This is the image the user will scan with
    their app
    // to set up two factor authentication
    $QR_Image = $google2fa->getQRCodeInline(
        config('app.name'),
        $registration_data['email'],
        $registration_data['google2fa_secret']
    );

    // Pass the QR barcode image to our view
    return view('google2fa.register', ['QR_Image' => $QR_Image, 'secret' =>
    $registration_data['google2fa_secret']]);
}
```

Next up we need to define the view for displaying Google's qr code. Add a new directory within views called google2fa. In that folder add a file called register.blade.php.



The contents of that file should look like this:

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2">
                <div class="panel panel-default">
                    <div class="panel-heading">Set up Google
Authenticator</div>

                    <div class="panel-body" style="text-align: center;">
                        <p>Set up your two factor authentication by
scanning the barcode below. Alternatively, you can use the code {{ $secret
}}</p>

                        <div>
                            
                        </div>
                        <p>You must set up your Google Authenticator app
before continuing. You will be unable to login otherwise</p>
                        <div>
                            <a href="/complete-registration"><button
class="btn-primary">Complete Registration</button></a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
@endsection
```

Now let's check if everything works.

Refresh your application and click register.

You should be greeted with this:

Register

Name

E-Mail Address

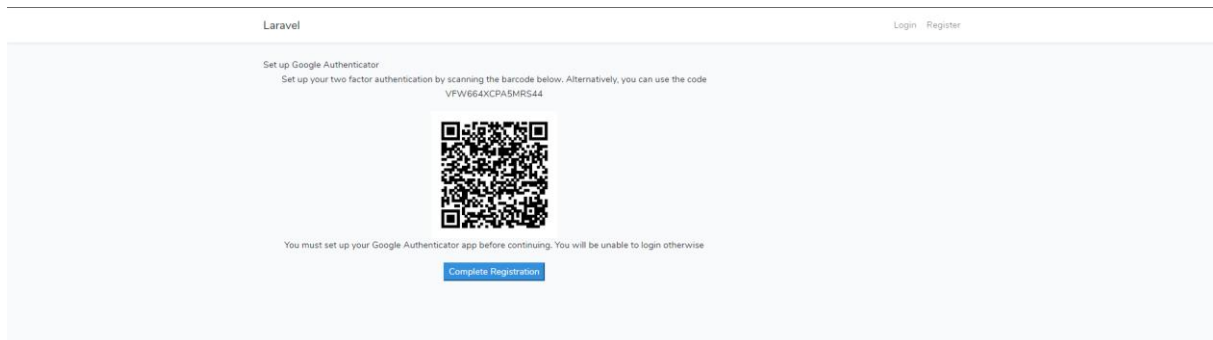
Password

Confirm Password

Register

Fill that in using your data of choice and click register.

You should now see a Google qr code.



Congratulations, you've made a authenticate token show up. We do need to change something though because if you click further it throws an error. This is because we need to setup the route and controller action.

Before we do that we need to make a migration in our users table.

Run

```
php artisan make:migration add_google2fa_column_to_users --table=users
```

And you should be greeted with this:

```
Created Migration: 2020_06_11_120511_add_google2fa_column_to_users
```

Your migration file should look like this:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddGoogle2faColumnToUsers extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            // add a text column in the users table for the
            google2fa_secret
            $table->text('google2fa_secret');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('users', function (Blueprint $table) {
            //
            $table->dropColumn('google2fa_secret');
        });
    }
}
```

Let's run php artisan migrate to migrate this into our database

And behold....

```
php artisan migrate
Migrating: 2020_06_11_120511_add_google2fa_column_to_users
Migrated: 2020_06_11_120511_add_google2fa_column_to_users (0.16 seconds)
```

It's migrated.

Now we will edit our register method in our register controller.

```
59 use RegistersUsers;
```

It looks like this and we are going to change it to this:

```
use RegistersUsers {  
    // change the name of the name of the trait's method in this class  
    // so it does not clash with our own register method  
    register as registration;  
}
```

Next up we are going into our web.php and we add this route:

```
Route::get( uri: '/complete-registration', action: 'Auth\RegisterController@completeRegistration');
```

This is our route to complete our registration.

Go into your RegisterController and define the method

```
public function completeRegistration(Request $request)  
{  
    // add the session data back to the request input  
    $request->merge(session( key: 'registration_data'));  
  
    // Call the default laravel authentication  
    return $this->registration($request);  
}
```

But Laravel doesn't save our unique Authenticator key yet so we need to define that in our create method.

```
protected function create(array $data)  
{  
    return User::create([  
        'name' => $data['name'],  
        'email' => $data['email'],  
        'password' => Hash::make($data['password']),  
        'google2fa_secret' => $data['google2fa_secret'],  
    ]);  
}
```

Add the bottom line to that method.

We also need to change our \$fillable in our User model.

In our User.php update the \$fillable property to look like this:

```
protected $fillable = [  
    'name', 'email', 'password', 'google2fa_secret'  
];
```

And our \$hidden property to this:

```
protected $hidden = [  
    'password', 'remember_token', 'google2fa_secret',  
];
```

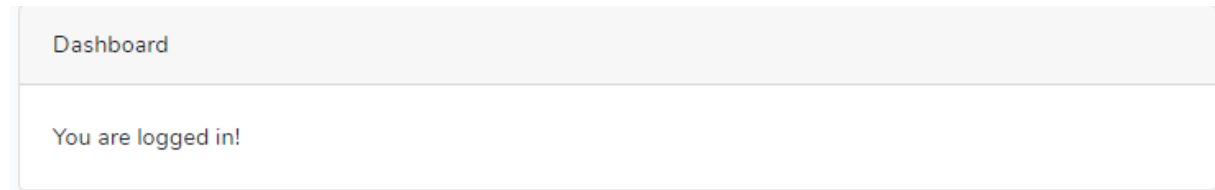
We also need to add some new methods to our User model:

```
/**  
 * Encrypt the user's google_2fa secret.  
 *  
 * @param string $value  
 * @return string  
 */  
public function setGoogle2faSecretAttribute($value)  
{  
    $this->attributes['google2fa_secret'] =  
encrypt($value);  
}  
  
/**  
 * Decrypt the user's google_2fa secret.  
 *  
 * @param string $value  
 * @return string  
 */  
public function getGoogle2faSecretAttribute($value)  
{  
    return decrypt($value);  
}
```


Now let's check if everything works.

Reminder you need to have the Google Authenticator app installed on your phone.

Go to your site and register. This time with the app Scan the qr code. You will need it to login in future steps. For now, you should be greeted with this message:



Congratulations, your registration system is in place.

Step three, let's setup our session login with our unique key.

In this step we will ensure that logging in happens with a valid key instead of just scanning the code and logging in immediately.

In our app/Http/Kernel.php we need to add something to the \$routeMiddleware method

```
'2fa' => \PragmaRX\Google2FALaravel\Middleware::class,
```

Add that line somewhere in that method.

Now we will define that the user enters their unique key to login, for that we will have to create a new blade.php file. in the same directory we added our register.blade.php earlier add index.blade.php.

In that file add this code:

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2">
                <div class="panel panel-default">
                    <div class="panel-heading">Register</div>

                    <div class="panel-body">
                        <form class="form-horizontal" method="POST"
action="{{ route('2fa') }}">
                            {{ csrf_field() }}

                            <div class="form-group">
                                <label for="one_time_password" class="col-
md-4 control-label">One Time Password</label>

                                <div class="col-md-6">
                                    <input id="one_time_password"
type="number" class="form-control" name="one_time_password" required
autofocus>
                                </div>
                            </div>

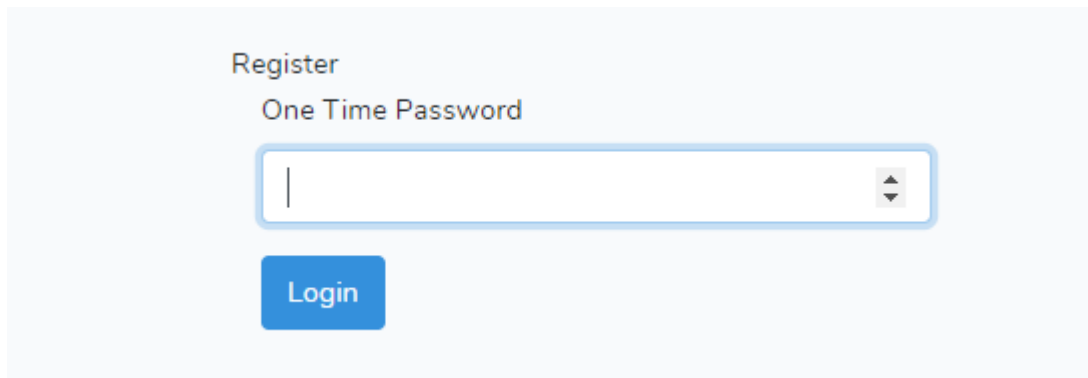
                            <div class="form-group">
                                <div class="col-md-6 col-md-offset-4">
                                    <button type="submit" class="btn btn-
primary">
                                        Login
                                    </button>
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
@endsection
```

Now we will make sure the user routes past this file so add this to our web.php file:

Now let's change our middleware to use it so change your __construct method in your HomeController to this:

```
public function __construct()
{
    $this->middleware(['auth', '2fa']);
}
```

Now let's check if everything works. You are now registered so logout and log back in. your application now requires a unique key.



Register

One Time Password

Login

Enter your code and tadaa:



Dashboard

You are logged in!

Congratulations you have now setup Google 2 factor Authentication to your Laravel application

I'm deploying my application to Heroku: the link to the Heroku and the code are included

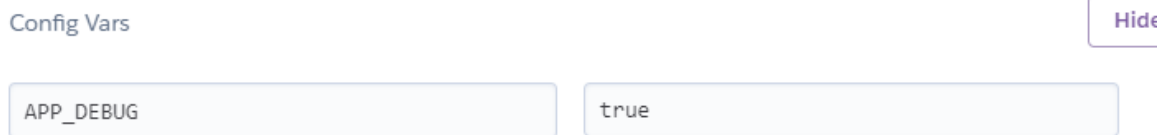
Heroku site: <https://arcane-hamlet-11116.herokuapp.com/>

Github: <https://github.com/LeslieScherbeijn/Google2FA>

Extra: Heroku errors.

Like I said at the beginning of this tutorial. It is useful to use your Laravel Homestead VM for this. Here's why.

When trying to run this project on Heroku after clicking register you may encounter a strange error. To make this less strange add this to your config vars:



The screenshot shows the Heroku 'Config Vars' section. At the top left is the label 'Config Vars' and at the top right is a 'Hide' button. Below these are two input fields. The first field is labeled 'APP_DEBUG' and the second field contains the value 'true'.

Now if you use this you will encounter the Error that you need to install the imagick extension. This is weird because in your composer you can't seem to find the imagick extension. This is where your vm comes in.

Heroku is hosted through Linux, guess what so does your VM. Imagick is a composer Linux package.

To acquire it run composer `require ext-imagick` after this `run composer update` After you have done this push your composer.json and composer.lock files to github. Your registration should now work.

References:

I used sources from Google, Laravel documentation, stackoverflow and scotch.io

The main ones are these:

Google 2f auth: <https://scotch.io/tutorials/how-to-add-googles-two-factor-authentication-to-laravel>

Laravel Auth: <https://laravel.com/docs/7.x/authentication>

Heroku Dev Center: <https://devcenter.heroku.com/>