https://laravel-china.org/articles/1989/laravel-52-auth-switch-to-saltpasswrod-encryption-authentication

Laravel 5.2 Auth 改用 salt+passwrod 加密验证的实现

 1年前 ⋅  4264 ⋅  19 ⋅  9

本文原链接来自我的博客，地址:[Laravel 5.2 Auth 认证解析以及改用 salt+passwrod 加密验证](http://www.yuansir-web.com/2016/04/12/laravel-5-2-auth-ren-zheng-gai-yong-salt-passwrod/)

Larval 5.2的默认Auth登陆传入邮件和用户密码到attempt 方法来认证，通过email的值获取，如果用户被找到，经哈希运算后存储在数据中的password将会和传递过来的经哈希运算处理的passwrod值进行比较。如果两个经哈希运算的密码相匹配那么将会为这个用户开启一个认证Session。

但是往往我们一些系统中的密码是通过salt+password的方式来做密码认证的，或者一些老的系统是通过salt+passwrod来认证的，现在重构迁移到Laravel框架中，那么密码认证如何不用默认的passwrod的方式而用salt+password的方式认证？

**要解决问题，我们最好还是先要弄明白根源，顺藤摸瓜**

首先看一下Laravel默认如何做密码验证的，看看Auth::guard($this->getGuard())->attempt($credentials)方法做了什么：

**Illuminate/Contracts/Auth/StatefulGuard.php**

namespace Illuminate\Contracts\Auth;

interface StatefulGuard extends Guard

{

/\*\*

\* Attempt to authenticate a user using the given credentials.

\*

\* @param array $credentials

\* @param bool $remember

\* @param bool $login

\* @return bool

\*/

public function attempt(array $credentials = [], $remember = false, $login = true);

......

上面代码看到attempt 是StatefulGuard 接口中的方法,第一个参数为需要认证的字段，第二个参数为是否记住登陆，第三个参数是否登陆，继续往下看attempt 在SessionGuard中是如何实现的

**illuminate/auth/SessionGuard.php**

class SessionGuard implements StatefulGuard, SupportsBasicAuth

{

use GuardHelpers;

......

/\*\*

\* Attempt to authenticate a user using the given credentials.

\*

\* @param array $credentials

\* @param bool $remember

\* @param bool $login

\* @return bool

\*/

public function attempt(array $credentials = [], $remember = false, $login = true)

{

$this->fireAttemptEvent($credentials, $remember, $login);

$this->lastAttempted = $user = $this->provider->retrieveByCredentials($credentials);

if ($this->hasValidCredentials($user, $credentials)) {

if ($login) {

$this->login($user, $remember);

}

return true;

}

return false;

}

/\*\*

\* Determine if the user matches the credentials.

\*

\* @param mixed $user

\* @param array $credentials

\* @return bool

\*/

protected function hasValidCredentials($user, $credentials)

{

return ! is\_null($user) && $this->provider->validateCredentials($user, $credentials);

}

.......

}

看到通过 $this->provider->retrieveByCredentials($credentials);和$this->provider->validateCredentials($user, $credentials);来实现验证，retrieveByCredentials是用来验证传递的字段查找用户记录是否存在，validateCredentials才是通过用户记录中密码和传入的密码做验证的实际过程。

这里需要注意的是$this->provider,这个provider其实是实现了一个Illuminate\Contracts\Auth\UserProvider 的Provider,我们看到Illuminate/Contracts/Auth下面有两个UserProvider的实现，分别为DatabaseUserProvider.php和EloquentUserProvider.php。但是我们验证密码的时候是通过哪个来验证的是在怎么决定的？

**config/auth.php**

'providers' => [

'users' => [

'driver' => 'eloquent',

'model' => App\Models\User::class, //这是User Model

],

],

这里我配置了'driver' => 'eloquent',那么就是通过EloquentUserProvider.php中的retrieveByCredentials来验证的了，我们继续看看它都干了啥

**illuminate/auth/EloquentUserProvider.php**

class EloquentUserProvider implements UserProvider

{

......

/\*\*

\* Retrieve a user by the given credentials.

\*

\* @param array $credentials

\* @return \Illuminate\Contracts\Auth\Authenticatable|null

\*/

public function retrieveByCredentials(array $credentials)

{

// First we will add each credential element to the query as a where clause.

// Then we can execute the query and, if we found a user, return it in a

// Eloquent User "model" that will be utilized by the Guard instances.

$query = $this->createModel()->newQuery();

foreach ($credentials as $key => $value) {

if (! Str::contains($key, 'password')) {

$query->where($key, $value);

}

}

return $query->first();

}

/\*\*

\* Validate a user against the given credentials.

\*

\* @param \Illuminate\Contracts\Auth\Authenticatable $user

\* @param array $credentials

\* @return bool

\*/

public function validateCredentials(UserContract $user, array $credentials)

{

$plain = $credentials['password'];

return $this->hasher->check($plain, $user->getAuthPassword());

}

......

}

上面两个方法 retrieveByCredentials用除了密码以外的验证字段查看记录是否存在，比如用email来查找用户记录是否存在, 然后 validateCredentials 方法就是通过 $this->hasher->check来将输入的密码和哈希的密码比较来验证密码是否正确，$plain 是提交过来的为加密密码字符串，$user->getAuthPassword()是数据库记录存放的加密密码字符串。

好了，看到这里就很明显了，我们需要改成我们自己的密码验证不就是自己实现一下validateCredentials方法就可以了吗，改变 $this->hasher->check为我们自己的密码验证就可以了，开始搞吧！

* 首先我们来实现$user->getAuthPassword();把数据库中用户表的salt和password传递到validateCredentials中来：

**修改 App\Models\User.php 添加如下代码**

public function getAuthPassword()

{

return ['password' => $this->attributes['password'], 'salt' => $this->attributes['salt']];

}

* 然后我们建立一个自己的UserProvider.php 的实现，你可以放到任何地方，我放到自定义目录中：

**新建 app/Foundation/Auth/RyanEloquentUserProvider.php**

<?php namespace App\Foundation\Auth;

use Illuminate\Auth\EloquentUserProvider;

use Illuminate\Contracts\Auth\Authenticatable;

use Illuminate\Support\Str;

class RyanEloquentUserProvider extends EloquentUserProvider

{

/\*\*

\* Validate a user against the given credentials.

\*

\* @param \Illuminate\Contracts\Auth\Authenticatable $user

\* @param array $credentials

\* @return bool

\*/

public function validateCredentials(Authenticatable $user, array $credentials)

{

$plain = $credentials['password'];

$authPassword = $user->getAuthPassword();

return sha1($authPassword['salt'] . sha1($authPassword['salt'] . sha1($plain))) == $authPassword['password'];

}

我这里通过$user->getAuthPassword();传递过来了用户记录的salt和password,然后将认证提交的密码$plain和salt进行加密，如果加密结果和用户数据库中记录的密码字符串匹配那么认证就通过了， 当然加密的算法完全是自定义的。

* 最后我们将User Providers换成我们自己的RyanEloquentUserProvider

**修改 app/Providers/AuthServiceProvider.php**

public function boot(GateContract $gate)

{

$this->registerPolicies($gate);

\Auth::provider('ryan-eloquent', function ($app, $config) {

return new RyanEloquentUserProvider($this->app['hash'], $config['model']);

});

}

**修改 config/auth.php**

'providers' => [

'users' => [

'driver' => 'ryan-eloquent',

'model' => App\Models\User::class,

],

],

好了，再试试可以用过salt+passwrod的方式密码认证了！