Dismiss

DARK MODE

You've been asking for dark mode for *years*.
The dark mode beta is finally here.

Change your preferences any time.

Laravel 5 - redirect to HTTPS

Asked 5 years, 2 months ago Active 1 month ago Viewed 172k times

116

64

Working on my first Laravel 5 project and not sure where or how to place logic to force HTTPS on my app. The clincher here is that there are many domains pointing to the app and only two out of three use SSL (the third is a fallback domain, long story). So I'd like to handle this in my app's logic rather than .htaccess.

In Laravel 4.2 I accomplished the redirect with this code, located in `filters.php` :

```
App::before(function($request)
{
    if( ! Request::secure())
    {
        return Redirect::secure(Request::path());
    }
});
```

I'm thinking Middleware is where something like this should be implemented but I cannot quite figure this out using it.

Thanks!

UPDATE

If you are using Cloudflare like I am, this is accomplished by adding a new Page Rule in your control panel.

php laravel laravel-routing laravel-5

edited Feb 9 '15 at 18:01

asked Feb 9 '15 at 4:14

NightMICU
7,540 22 79 113

So what happens with the 3rd domain? If you force https on all routes - will the 3rd domain keep working? – Laurence Feb 9 '15 at 4:28

Detecting that with `$_SERVER['HTTP_HOST']` – NightMICU Feb 9 '15 at 4:30

How long did it take for cloudflare page rule to take effect – CodeGuru Jun 8 '17 at 8:35

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

19 Answers

Active | Oldest | Votes

240

You can make it works with a Middleware class. Let me give you an idea.

```php
namespace MyApp\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\App;

class HttpsProtocol {

    public function handle($request, Closure $next)
    {
            if (!$request->secure() && App::environment() === 'production') {
                return redirect()->secure($request->getRequestUri());
            }

            return $next($request);
    }
}
```

Then, apply this middleware to every request adding setting the rule at `Kernel.php` file, like so:

```php
protected $middleware = [
    'Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode',
    'Illuminate\Cookie\Middleware\EncryptCookies',
    'Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse',
    'Illuminate\Session\Middleware\StartSession',
    'Illuminate\View\Middleware\ShareErrorsFromSession',

    // appending custom middleware
    'MyApp\Http\Middleware\HttpsProtocol'

];
```

At sample above, the middleware will redirect every request to https if:

1. The current request comes with no secure protocol (http)
2. If your environment is equals to `production`. So, just adjust the settings according to your preferences.

Cloudflare

I am using this code in production environment with a WildCard SSL and the code works correctly. If I remove `&& App::environment() === 'production'` and test it in localhost, the redirection also works. So, having or not a installed SSL is not the problem. Looks like you need to keep a very hard attention to your Cloudflare layer in order to get redirected to Https protocol.

Edit 23/03/2015

Thanks to `@Adam Link` 's suggestion: it is likely caused by the headers that Cloudflare is passing. CloudFlare likely hits your server via HTTP and passes a X-Forwarded-Proto header that declares it is forwarding a HTTPS request. You need add another line in your Middleware

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
$request->setTrustedProxies( [ $request->getClientIp() ] );
```

...to trust the headers CloudFlare is sending. This will stop the redirect loop

Edit 27/09/2016 - Laravel v5.3

Just need to add the middleware class into `web` group in `kernel.php file` :

```
protected $middlewareGroups = [
    'web' => [
        \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
        \Illuminate\Session\Middleware\StartSession::class,
        \Illuminate\View\Middleware\ShareErrorsFromSession::class,

        // here
        \MyApp\Http\Middleware\HttpsProtocol::class

    ],
];
```

Remember that `web` group is applied to every route by default, so you do not need to set `web` explicitly in routes nor controllers.

Edit 23/08/2018 - Laravel v5.7

- To redirect a request depending the environment you can use `App::environment() === 'production'` . For previous version was `env('APP_ENV') === 'production'` .
- Using `\URL::forceScheme('https');` actually does not redirect. It just builds links with `https://` once the website is rendered.

edited Aug 23 '18 at 22:53

answered Feb 9 '15 at 6:22

manix

12.7k 8 55 92

4 This seems to be giving me a redirect loop... looks like it should work, though. I do not know if it makes any difference but we're using a Cloudflare SSL. But I do not think that would change the simple redirect. – NightMICU Feb 9 '15 at 15:29

3 @NightMICU I'm not sure if you've solved the issue with the redirect, but it is likely caused by the headers that Cloudflare is passing. CloudFlare likely hits your server via HTTP and passes a X-Forwarded-Proto header that declares it is forwarding a HTTPS request. You need add another line in your Middleware that says `$request->setTrustedProxies( [ $request->getClientIp() ] );` to trust the headers CloudFlare is sending. This will stop the redirect loop. – Adam Link Mar 23 '15 at 18:59

2 @manix Awesome. Just went through this HTTPS issue this weekend with my own project - that little stuff will frustrate you for hours! – Adam Link Mar 23 '15 at 19:18

8 Excellent answer! Just one detail: It's better to use a 301 redirect to indicate to Google that's a permanent movement. Like: `return redirect()->secure($request->getRequestUri(), 301);` – adriaroca May 4 '17 at 14:14

4 for those whos under load balancer or proxy can change to secure() to `$request->server('HTTP_X_FORWARDED_PROTO') != 'https'` this works for me – Shiro May 30 '17 at 4:00

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

59

An other option that worked for me, in AppServiceProvider place this code in the boot method:

```
\URL::forceScheme('https');
```

The function written before forceScheme('https') was wrong, its forceScheme

edited May 27 '19 at 4:37

Rohan Kumar
37.7k 11 66 96

answered Oct 2 '16 at 14:10

Constantin Stan
897 8 11

14 Hey, just found this through googling around - note that in 5.4 it's `\URL::forceScheme('https');` – dev Mar 18 '17 at 6:47

2 In the same file, you can also do `if($this->app->environment() === 'production'){ $this->app['request']->server->set('HTTPS', true); }` – Rory May 15 '17 at 18:25

2 did u mean `\URL::forceScheme('https')` – Ernest Okot Jul 11 '17 at 3:35

16 I'm pretty sure that this is only for building links. This wont force a user to https, it will only serve links prepended with https:// – Weston Watson Oct 18 '17 at 19:54

yeah it happened @WestonWatson. kindly share the solution if found – Harat Oct 17 '19 at 21:25

32

Alternatively, If you are using Apache then you can use `.htaccess` file to enforce your URLs to use `https` prefix. On Laravel 5.4, I added the following lines to my `.htaccess` file and it worked for me.

```
RewriteEngine On

RewriteCond %{HTTPS} !on
RewriteRule ^.*$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

edited Apr 26 '17 at 12:07

answered Apr 26 '17 at 12:00

Assad Ullah Ch
586 5 11

2 This is not good if you have multiple environments (dev, stage, production) than you need to set SSL on all of them. – Mladen Janjetovic Nov 15 '17 at 10:09

@MladenJanjetovic you can use `RewriteCond %{HTTP_HOST} !=localhost` on dev to get around this. – Dan Jul 17 '18 at 4:21

3 @Dan - Yes, but you will still have to set it up for stage, local (and this is more complicated if developers are using different URLs in local development, i.e. .dev, .local, subdomains,...etc). I would prefer to have that kind of logic in the application. – Mladen Janjetovic Jul 17 '18 at 7:05

16

for laravel 5.4 use this format to get https redirect instead of .htaccess

```
namespace App\Providers;

use Illuminate\Support\Facades\URL;
use Illuminate\Support\ServiceProvider;
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
    public function boot()
    {
        URL::forceScheme('https');
    }
}
```

answered Apr 28 '17 at 9:50

Arun Yokesh
1,142 17 17

13 Just to clarify: 1) these changes should be done in app/Providers/AppServiceProvider.php; 2) this is only to set the links generated inside the app to use SSL, it does not force you to use SSL – phoenix Jun 8 '17 at 20:03

Hi, Route is not generating by this method if i click on a button which send me to the next route its not giving me the error 404 – Saket Sinha Sep 11 '17 at 1:19

This is not an https redirect. But it allows the serving of https:// site. If you change it to http:// it will serve as well. – franc Oct 16 '19 at 7:49

10

Similar to manix's answer but in one place. Middleware to force HTTPS

```
namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

class ForceHttps
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request $request
     * @param  \Closure $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if (!app()->environment('local')) {
            // for Proxies
            Request::setTrustedProxies([$request->getClientIp()]);

            if (!$request->isSecure()) {
                return redirect()->secure($request->getRequestUri());
            }
        }

        return $next($request);
    }
}
```

answered Jun 14 '16 at 17:00

Mladen Janjetovic
9,350 7 54 64

does Request need to be static? – GFxJamal Jan 4 '18 at 11:01

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

7

This is for Larave 5.2.x and greater. If you want to have an option to serve some content over HTTPS and others over HTTP here is a solution that worked for me. You may wonder, why would someone want to serve only some content over HTTPS? Why not serve everything over HTTPS?

Although, it's totally fine to serve the whole site over HTTPS, severing everything over HTTPS has an additional overhead on your server. Remember encryption doesn't come cheap. The slight overhead also has an impact on your app response time. You could argue that commodity hardware is cheap and the impact is negligible but I digress :) I don't like the idea of serving marketing content big pages with images etc over https. So here it goes. It's similar to what others have suggest above using middleware but it's a full solution that allows you to toggle back and forth between HTTP/HTTPS.

First create a middleware.

```
php artisan make:middleware ForceSSL
```

This is what your middleware should look like.

```
<?php

namespace App\Http\Middleware;

use Closure;

class ForceSSL
{

    public function handle($request, Closure $next)
    {

        if (!$request->secure()) {
            return redirect()->secure($request->getRequestUri());
        }

        return $next($request);
    }
}
```

Note that I'm not filtering based on environment because I have HTTPS setup for both local dev and production so there is not need to.

Add the following to your routeMiddleware \App\Http\Kernel.php so that you can pick and choose which route group should force SSL.

```
    protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'can' => \Illuminate\Foundation\Http\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'forceSSL' => \App\Http\Middleware\ForceSSL::class,
];
```

Next, I'd like to secure two basic groups login/signup etc and everything else behind Auth

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
Route::group(array('middleware' => 'forceSSL'), function() {
/*user auth*/
Route::get('login', 'AuthController@showLogin');
Route::post('login', 'AuthController@doLogin');

// Password reset routes...
Route::get('password/reset/{token}', 'Auth\PasswordController@getReset');
Route::post('password/reset', 'Auth\PasswordController@postReset');

//other routes like signup etc

});


Route::group(['middleware' => ['auth','forceSSL']], function()
 {
Route::get('dashboard', function(){
    return view('app.dashboard');
});
Route::get('logout', 'AuthController@doLogout');

//other routes for your application
});
```

Confirm that your middlewares are applied to your routes properly from console.

```
php artisan route:list
```

Now you have secured all the forms or sensitive areas of your application, the key now is to use your view template to define your secure and public (non https) links.

Based on the example above you would render your secure links as follows -

```
<a href="{{secure_url('/login')}}">Login</a>
<a href="{{secure_url('/signup')}}">SignUp</a>
```

Non secure links can be rendered as

```
<a href="{{url('/aboutus',[],false)}}">About US</a></li>
<a href="{{url('/promotion',[],false)}}">Get the deal now!</a></li>
```

What this does is renders a fully qualified URL such as https://yourhost/login and http://yourhost/aboutus

If you were not render fully qualified URL with http and use a relative link url('/aboutus') then https would persists after a user visits a secure site.

Hope this helps!

answered Aug 27 '16 at 14:45

na-98

841 8 13

What about just using **.htaccess** file to achieve https redirect? This should be placed in

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
<IfModule mod_rewrite.c>
   RewriteEngine On
   # Force SSL
   RewriteCond %{HTTPS} !=on
   RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
   # Remove public folder form URL
   RewriteRule ^(.*)$ public/$1 [L]
</IfModule>
```

I use this for laravel 5.4 (latest version as of writing this answer) but it should continue to work for feature versions even if laravel change or removes some functionality.

edited Jun 9 '17 at 1:32

answered Mar 18 '17 at 4:06

Maulik Gangani
1,821 1 16 22

Chrome gives me an error: Too many redirects.. Seems like this makes a loop – phoenix Jun 8 '17 at 20:05

Hi, I've updated answer. Make sure you put this .htaccess in project root directory and point your server (apache config) to project root. – Maulik Gangani Jun 9 '17 at 1:33

1 @MladenJanjetovic you can have different htaccess files for these environments – Burgi Feb 13 '18 at 10:00

1 @MladenJanjetovic having it in the application definitely has its advantages, but from an efficiency and speed perspective I'd say it's advantageous to have this in the server config so that you don't have to load up Laravel just for a redirect. Environment-specific configs in a single versioned .htaccess can be achieved by using a rewrite condition to check the domain, something like `RewriteCond %{HTTP_HOST} productiondomain\.com$ [NC]` – Chris Mar 11 '18 at 14:54

1 I too prefer to place this in .htaccess in the root directory, and as Crhis said, environment-specific settings can be accomplished here, albeit a bit less elegantly than in Mladen's solution. – jovan Dec 17 '18 at 15:04

6

You can use RewriteRule to force ssl in .htaccess same folder with your index.php
Please add as picture attach, add it **before all rule others**

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
<IfModule mod_rewrite.c>
    <IfModule mod_negotiation.c>
        Options -MultiViews
    </IfModule>

    RewriteEngine On

    # Force SSL
    RewriteCond %{HTTPS} !=on
    RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

    # Redirect Trailing Slashes If Not A Folder...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^(.*)/$ /$1 [L,R=301]

    # Handle Front Controller...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]

    # Handle Authorization Header
    RewriteCond %{HTTP:Authorization} .
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]

</IfModule>

```

answered Jun 1 '18 at 4:47

Quy Le
1,926 17 18

3

in IndexController.php put

```
public function getIndex(Request $request)
{
    if ($request->server('HTTP_X_FORWARDED_PROTO') == 'http') {

        return redirect('/');
    }

    return view('index');
}
```

in AppServiceProvider.php put

```
public function boot()
{
    \URL::forceSchema('https');

}
```

In AppServiceProvider.php every redirect will be go to url https and for http request we need once redirect so in IndexController.php Just we need do once redirect

edited Feb 5 '17 at 19:22 answered Feb 2 '17 at 18:29

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Can you explain how your answer solves the question? – soundslikeodd Feb 2 '17 at 18:56

Please add this explanation to your answer. – soundslikeodd Feb 4 '17 at 18:44

3

The answers above didn't work for me, but it appears that Deniz Turan rewrote the .htaccess in a way that works with Heroku's load balancer here: https://www.jcore.com/2017/01/29/force-https-on-heroku-using-htaccess/

```
RewriteEngine On
RewriteCond %{HTTPS} !=on
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

answered Dec 13 '17 at 14:50

Phil

31 3

2

Here's how to do it on Heroku

To force SSL on your dynos but not locally, add to end of your .htaccess in public/:

```
# Force https on heroku...
# Important fact: X-forwarded-Proto will exist at your heroku dyno but wont locally.
# Hence we want: "if x-forwarded exists && if its not https, then rewrite it":
RewriteCond %{HTTP:X-Forwarded-Proto} .
RewriteCond %{HTTP:X-Forwarded-Proto} !https
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

You can test this out on your local machine with:

```
curl -H"X-Forwarded-Proto: http" http://your-local-sitename-here
```

That sets the header X-forwarded to the form it will take on heroku.

i.e. it simulates how a heroku dyno will see a request.

You'll get this response on your local machine:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://tm3.localhost:8080/">here</a>.</p>
</body></html>
```

That is a redirect. That is what heroku is going to give back to a client if you set the .htaccess as above. But it doesn't happen on your local machine because X-forwarded won't be set (we faked it with curl above to see what was happening).

answered Feb 20 '18 at 17:32

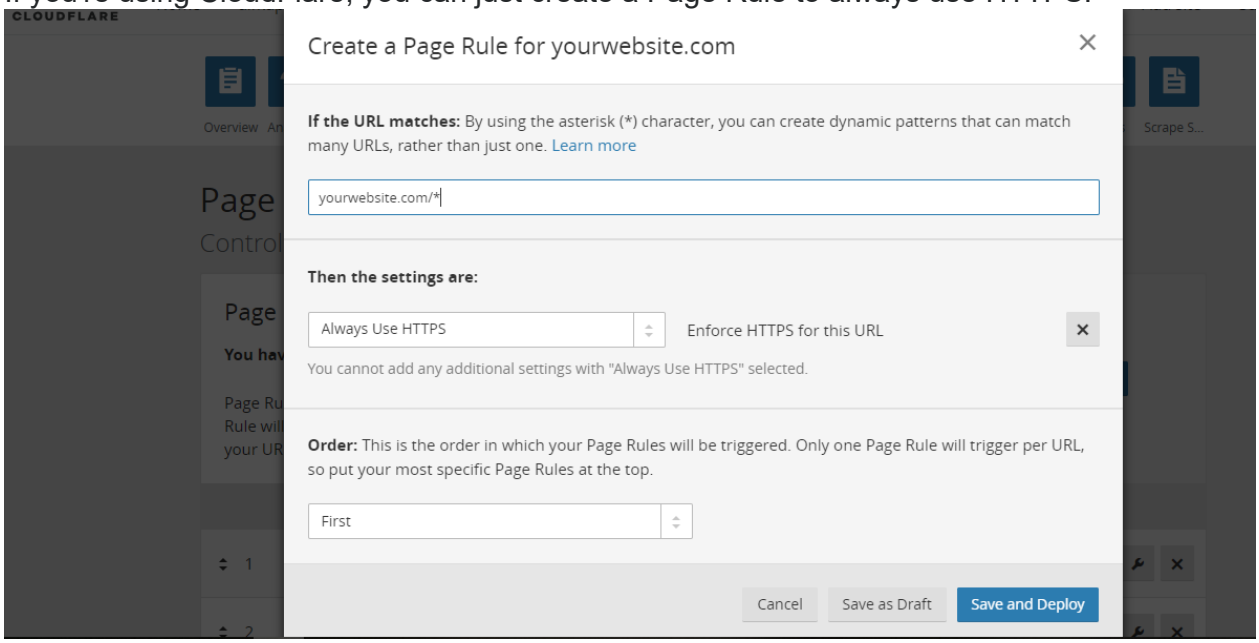
mwal

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

worked for me! thanks – Ricardinho Apr 9 at 2:38

2

If you're using CloudFlare, you can just create a Page Rule to always use HTTPS:



This will redirect every http:// request to https://

In addition to that, you would also have to add something like this to your \app\Providers\AppServiceProvider.php boot() function:

```
if (env('APP_ENV') === 'production' || env('APP_ENV') === 'dev') {
     \URL::forceScheme('https');
}
```

This would ensure that every link / path in your app is using https:// instead of http://.

answered Jul 5 '18 at 21:24

butaminas
337 3 13

2

I'm adding this alternative as I suffered a lot with this issue. I tried all different ways and nothing worked. So, I came up with a workaround for it. It might not be the best solution but it does work -

FYI, I am using Laravel 5.6

```
if (App::environment('production')) {
    URL::forceScheme('https');
}
```

production <- It should be replaced with the APP_ENV value in your .env file

answered Jul 21 '18 at 11:10

thebrownkid

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

A little different approach, tested in Laravel 5.7

2

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Str;

class ForceHttps
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if ( !$request->secure() && Str::startsWith(config('app.url'), 'https://') ) {
            return redirect()->secure($request->getRequestUri());
        }
        return $next($request);
    }
}
```

PS. Code updated based on @matthias-lill's comments.

edited Feb 17 at 19:37

answered Nov 14 '18 at 9:13

Zoli
886 1 7 20

1 Works on Laravel 6 too. – Rubens Oct 29 '19 at 14:15

1 using the env() functions will not work with cached config files. This should point to `config('app.url')` instead. Also, Laravel comes with a very handy string function `Str::startsWith(config('app.url'), 'https://')` . – Matthias Lill Feb 17 at 10:46

1

For Laravel 5.6, I had to change condition a little to make it work.

from:

```php
if (!$request->secure() && env('APP_ENV') === 'prod') {
return redirect()->secure($request->getRequestUri());
}
```

To:

```php
if (empty($_SERVER['HTTPS']) && env('APP_ENV') === 'prod') {
return redirect()->secure($request->getRequestUri());
}
```

answered Apr 29 '18 at 11:08

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

1

This worked out for me. I made a custom php code to force redirect it to https. Just include this code on the header.php

```
<?php
if (isset($_SERVER['HTTPS']) &&
    ($_SERVER['HTTPS'] == 'on' || $_SERVER['HTTPS'] == 1) ||
    isset($_SERVER['HTTP_X_FORWARDED_PROTO']) &&
    $_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https') {
  $protocol = 'https://';
}
else {
  $protocol = 'http://';
}
$notssl = 'http://';
if($protocol==$notssl){
    $url = "https://$_SERVER[HTTP_HOST]$_SERVER[REQUEST_URI]";?>
    <script>
    window.location.href ='<?php echo $url?>';
    </script>
 <?php } ?>
```

answered Jul 3 '18 at 19:04

Geeky Ashim
11 1

1

I am using in Laravel 5.6.28 next middleware:

```
namespace App\Http\Middleware;

use App\Models\Unit;
use Closure;
use Illuminate\Http\Request;

class HttpsProtocol
{
    public function handle($request, Closure $next)
    {
        $request->setTrustedProxies([$request->getClientIp()],
Request::HEADER_X_FORWARDED_ALL);

        if (!$request->secure() && env('APP_ENV') === 'prod') {
            return redirect()->secure($request->getRequestUri());
        }

        return $next($request);
    }
}
```

answered Jul 30 '18 at 15:35

fomvasss
11 1

The easiest way would be at the application level. In the file

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
use Illuminate\Support\Facades\URL;
```

and in the boot() method add the following:

```
$this->app['request']->server->set('HTTPS', true);
URL::forceScheme('https');
```

This should redirect all request to https at the application level.

(Note: this has been tested with laravel 5.5 LTS)

answered Oct 18 '18 at 13:45

Pinak Saha
36 3

0

This work for me in **Laravel 7.x** in **3 simple steps** using a middleware:

1) Generate the middleware with command `php artisan make:middleware ForceSSL`

Middleware

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\App;

class ForceSSL
{
    public function handle($request, Closure $next)
    {
        if (!$request->secure() && App::environment() === 'production') {
            return redirect()->secure($request->getRequestUri());
        }

        return $next($request);
    }
}
```

2) Register the middleware in `routeMiddleware` inside Kernel file

Kernel

```
protected $routeMiddleware = [
    //...
    'ssl' => \App\Http\Middleware\ForceSSL::class,
];
```

3) Use it in your routes

Routes

```
Route::middleware('ssl')->group(function() {
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

here the full documentation about middlewares

==========

.HTACCESS Method

If you prefer to use an `.htaccess` file, you can use the following code:

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{SERVER_PORT} 80
    RewriteRule ^(.*)$ https://yourdomain.com/$1 [R,L]
</IfModule>
```

Regards!

answered Mar 11 at 17:24

Radames E. Hernandez
2,283 13 26

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.