VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE

Network Security Project

CVE-2021-43617

Done by:
Edvinas Šarauskas

Supervisor:
lect. Virgilijus Krinickij

Vilnius
2022

Contents

1 About **3**

2 Goal **3**

3 Exploitation **3**

4 Fix **8**

1 About

Laravel Framework through 8.70.2 does not sufficiently block the upload of executable PHP content because Illuminate/Validation/Concerns/ValidatesAttributes.php lacks a check for .phar files, which are handled as application/x-httpd-php on systems based on Debian.

2 Goal

Project goal – recreate CVE-2021-43617 vulnerability in a Linux-based machine, with Laravel 8.70.1 version. A script will be uploaded from the attacker machine to the Laravel based website. As a result, the website server will contain a malicious file inside, which the attacker will use to reverse shell the machine.

3 Exploitation

```
1286    protected function shouldBlockPhpUpload($value, $parameters)
1287    {
1288        if (in_array('php', $parameters)) {
1289            return false;
1290        }
1291
1292        $phpExtensions = [
1293            'php', 'php3', 'php4', 'php5', 'phtml',
1294        ];
1295
1296        return ($value instanceof UploadedFile)
1297            ? in_array(trim(strtolower($value->getClientOriginalExtension())), $phpExtensions)
1298            : in_array(trim(strtolower($value->getExtension())), $phpExtensions);
1299    }
1300
1301    /**
1302     * Validate the size of an attribute is greater than a minimum value.
1303     *
1304     * @param  string  $attribute
1305     * @param  mixed  $value
1306     * @param  array  $parameters
1307     * @return bool
1308     */
```

Figure 1. Vulnerable code inside Illuminate/Validation/Concerns/ValidatesAttributes.php

To start off we can see that inside the ValidatesAttributes.php file, the function shouldBlockPhpUpload is vulnerable to uploading files with .phar extension, because it is not validated. Files with .phar extension are used on systems based on Debian.

Next, we will create a Laravel project with validated file upload functionality:

```php
<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\ImageUploadController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/


Route::get('/', [ ImageUploadController::class, 'imageUpload' ])->name('
    image.upload');

Route::post('/', [ ImageUploadController::class, 'imageUploadPost' ])->name('
    image.upload.post');

```

Figure 2. Laravel routes used in web interface

```php
public function imageUploadPost(Request $request)

{

$request->validate([

'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',

]);

$imageName = time().'.'.$request->image->getClientOriginalExtension();

$request->image->move(public_path('images'), $imageName);

/* Store $imageName name in DATABASE from HERE */

return back()

->with('success','You have successfully upload image.')

->with('image',$imageName);

}
```

Figure 3. imageUploadPost method

Figure 2 shows Laravel routes used in web interface. These routes are needed to route all application requests to its appropriate controller and method. Figure 3 shows imageUploadPost method in which the main image upload logic is implemented. Firstly, validation rules are defined. In this case we require that the image field in the form is not empty, also it should an be image with extensions of jpeg, png, jpg, gif, svg and size not more than 2048 kB. Then the image name is composed and image is moved to dedicated directory with a return of success message.

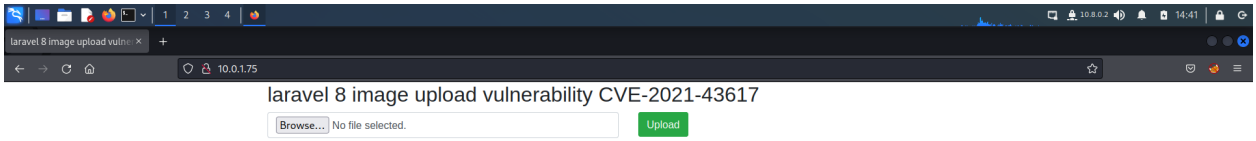

Figure 4. User interface

Now that we have the file upload functionality, we can start exploiting the system.

In order to exploit the vulnerability we have generated a reverse shell with a .phar extension using Weevely.

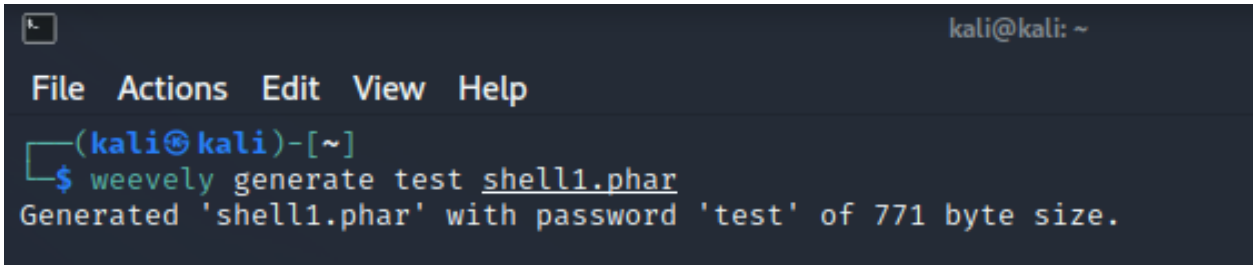

Figure 5. User interface

After trying to upload it we get an error:

Figure 6. Validation fail

Figure 4 shows user interface with the image upload form. Figure 6 shows validation in progress - when the uploaded file does not meet defined criteria, an error message is displayed, with rules of validation.

We need to modify the file:

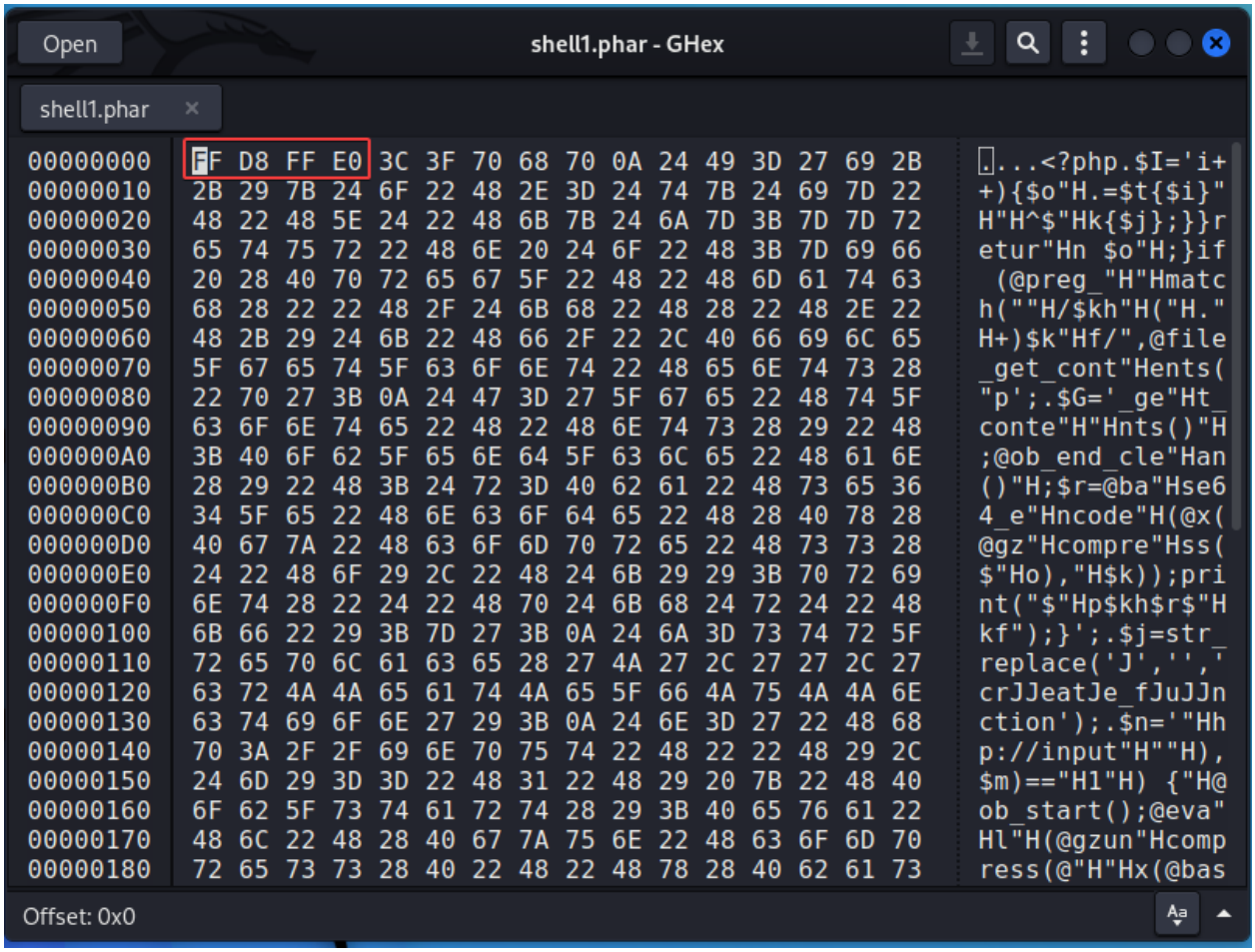

Figure 7. Reverse shell with changes at the front

Using a hex editor, changes at the front of the file where made so that file is read as having JPG format. This is achieved by adding **FF D8 FF E0**. Figure 6 shows these changes circled in red.

Figure 8. Successful file upload

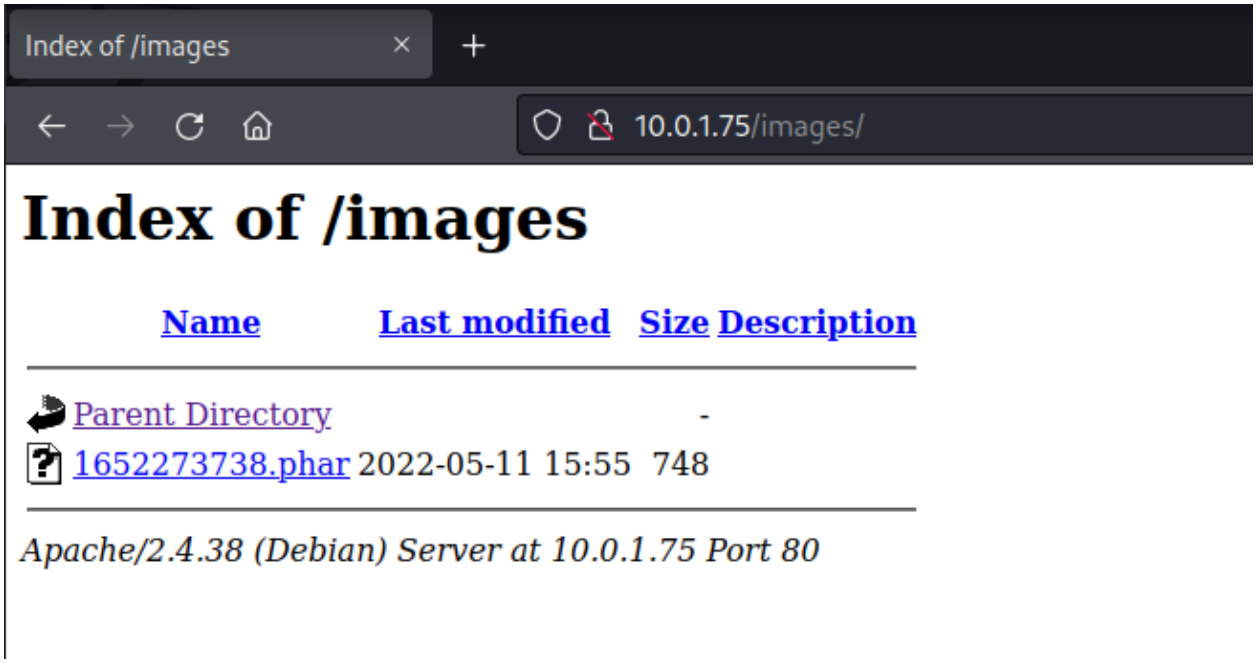

Figure 9. File with .phar extension available in images directory

Figure 8 and 9 show that after changes mentioned above, validation is bypassed, file with .phar extension is uploaded successfully and is available in /images directory.

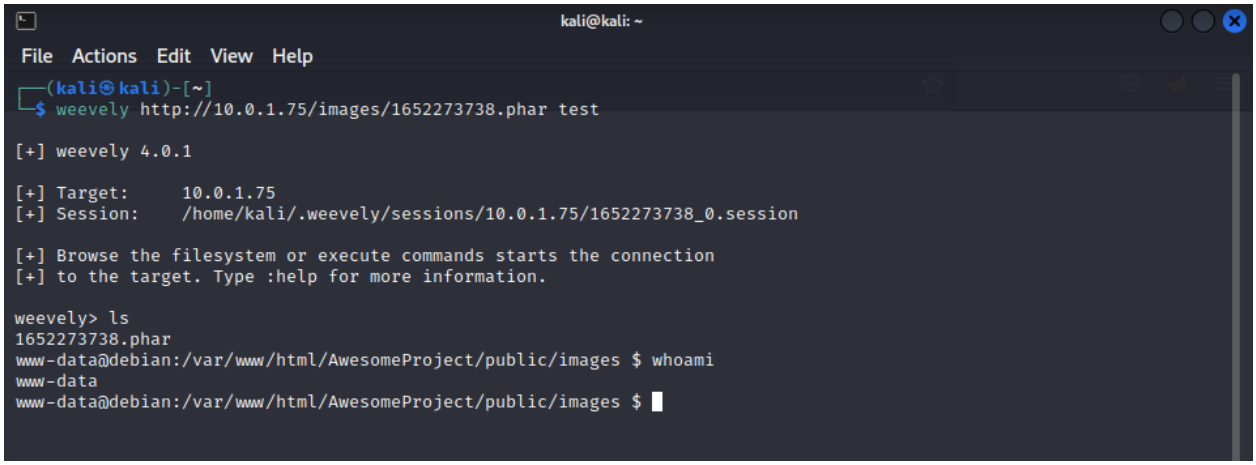

Figure 10. Successful connection to victim

Using Weevely it is possible to connect to the server and navigate inside of it. Now having access to the server it is possible to exploit it further by escalating privileges and more.

4 Fix

```php
/**
 * Check if PHP uploads are explicitly allowed.
 *
 * @param  mixed  $value
 * @param  array  $parameters
 * @return bool
 */
protected function shouldBlockPhpUpload($value, $parameters)
{
    if (in_array('php', $parameters)) {
        return false;
    }

    $phpExtensions = [
        'php', 'php3', 'php4', 'php5', 'phtml', 'phar',
    ];

    return ($value instanceof UploadedFile)
        ? in_array(trim(strtolower($value->getClientOriginalExtension())), $phpExtensions)
        : in_array(trim(strtolower($value->getExtension())), $phpExtensions);
}

/**
 * Validate the size of an attribute is greater than a minimum value.
 *
 * @param  string  $attribute
 * @param  mixed  $value
 * @param  array  $parameters
 * @return bool
 */
```

Figure 11. A patch for the vulnerability

Figure 11 shows the fix for the vulnerable function inside the Illuminate/Validation/Concerns/ValidatesAttributes.php file, .phar extension needs to be added to the phpExtensions array so that it would be blocked.