[GXYCTF2019]BabysqliV3.0-phar反序列化

原创

pumpkin.zhu 于 2021-05-19 14:47:37 发布 590 收藏 2

分类专栏： CTF

本文链接：https://blog.csdn.net/soldi_er/article/details/116998447

文章目录

前期漏洞探测

探测SQL注入

登陆框。

面对登陆框时，可以使用admin测试盲注。或者使用pw测。

先fuzz一波，返回的都是"no this user"，没有防火墙提醒。

存在admin用户。在name中测7种闭合方式，都返回"no this user"。

`admin'%23`、`')`、`'")`、`'))`。`"`、`")`、`"))`。

这才贴合实战，要是实战基本就放弃了。一看就知道引号被转义了。

估计这道题目的利用场景是宽字节注入。

简单看一下宽字节的概念，固定输入name=admin，测试pw字段，

```
%E0' or sleep(5)%23，以这种形式测试7种闭合方式，都返回'Wrong pass'。
这是具有SQL注入漏洞的服务器该有的样子？？
```

果断查看WriteUp，md弱口令。小了，格局小了。

登录之后，看到URL地址栏是`home.php?file=upload`。

探测文件上传

上传正常图片，返回"413 Request Entity Too Large"。
上传PHP脚本，查看前端代码，发现直接给爷命名为.txt文件。
这是你文件上传漏洞该有的样子？？

探测文件包含漏洞

地址栏的参数`?file`，这不得想到文件包含漏洞，真灵活。
传递`?file=index.php`，返回"当前引用的是 index.php.fxxkyou!"
传递`?file=/etc/passwd`，返回"/etc/passwd.fxxkyou!"，欸你怎么骂人？
尝试使用%00截断，没有成功。

对home.php传递不同的file参数：

```
 file=upload，添加后缀php
file=index/index.php，添加后缀fxxkyou!
file=/etc/passwd，添加后缀fxxkyou!
多次测试推断，如果上传的参数是upload，则包含该文件。如果不是，就骂人。

 直接访问home.php，返回`no permission!`
file=home，返回"当前引用的是home.php"，说明很可能不是白名单。

 别测了，还记得文件包含漏洞常见的利用方式吗？
php:filter伪协议读源码。文件包含图片马。

 读取upload.php的源码，
?file=php://filter/read=convert.base64-encode/resource=upload
```

upload.php的源码：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<form action="" method="post" enctype="multipart/form-data">
 ä¸Šä¼ æ–‡ä»¶
 <input type="file" name="file" />
 <input type="submit" name="submit" value="ä¸Šä¼ " />
</form>

<?php
error_reporting(0);
class Uploader{
 public $Filename;
 public $cmd;
 public $token;


 function __construct(){
  $sandbox = getcwd()."/uploads/".md5($_SESSION['user'])."/";
  $ext = ".txt";
  @mkdir($sandbox, 0777, true);
  if(isset($_GET['name']) and !preg_match("/data:\/\/ | filter:\/\/ | php:\/\/ | \./i", $_GET['name'])){
   $this->Filename = $_GET['name'];
  }
  else{
   $this->Filename = $sandbox.$_SESSION['user'].$ext;
  }

  $this->cmd = "echo '<br><br>Master, I want to study rizhan!<br><br>';";
```

```
  $this->token = $_SESSION['user'];
 }

 function upload($file){
  global $sandbox;
  global $ext;

  if(preg_match("[^a-z0-9]", $this->Filename)){
   $this->cmd = "die('illegal filename!');";
  }
  else{
   if($file['size'] > 1024){
    $this->cmd = "die('you are too big (â€²â–½`ã€ƒ)');";
   }
   else{
    $this->cmd = "move_uploaded_file('".$file['tmp_name']."', '" . $this->Filename . "');";
   }
  }
 }

 function __toString(){
  global $sandbox;
  global $ext;
  // return $sandbox.$this->Filename.$ext;
  return $this->Filename;
 }

 function __destruct(){
  if($this->token != $_SESSION['user']){
   $this->cmd = "die('check token falied!');";
  }
  eval($this->cmd);
 }
}

if(isset($_FILES['file'])) {
 $uploader = new Uploader();
 $uploader->upload($_FILES["file"]);
 if(@file_get_contents($uploader)){
  echo "以下是你上传的文件<br>".$uploader."<br>";
  echo file_get_contents($uploader);
 }
}
?>
```

home.php的源码：

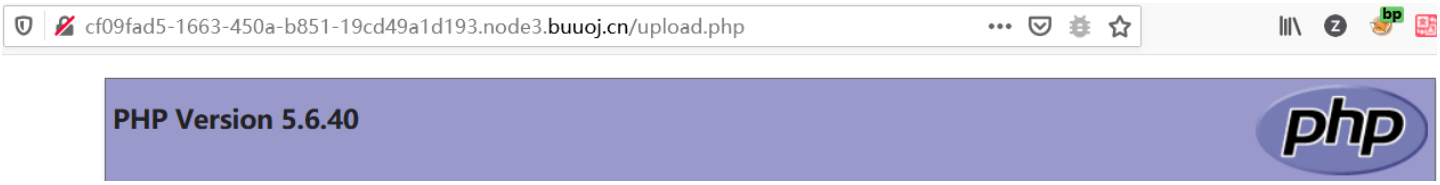
```php
<?php
session_start();
echo "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\" /> <title>Home</title>";
error_reporting(0);
if(isset($_SESSION['user'])){
 if(isset($_GET['file'])){
  if(preg_match("/.?f.?l.?a.?g.?/i", $_GET['file'])){
   die("hacker!");
  }
  else{
   if(preg_match("/home$/i", $_GET['file']) or preg_match("/upload$/i", $_GET['file'])){
    $file = $_GET['file'].".php";
   }
   else{
    $file = $_GET['file'].".fxxkyou!";
   }
   echo "å½“å‰ å¼•ç”¨çš„æ˜¯ ".$file;
   require $file;
  }
 }
 else{
  die("no permission!");
 }
}
?>
```

审计源码

home.php文件：看到`require $file;`和preg_match()时，想到上传文件名包含`upload`的php脚本、或图片马。但upload.php对文件名使用了MD5加密，这种文件包含木马的利用受到了限制。

upload.php文件：熟悉的__construct()构造函数，反序列化?
老规矩，先看明白代码的功能，再寻找漏洞点，最后编写exp。
拿出纸笔先溜了，一会儿继续快乐审计！

解（1）文件上传覆盖已有文件：开发的失误呀！
通过Get传入name参数时，上传的文件名可控，且程序没有检查
上传的文件是否已存在。所以抓包构造`POST upload.php?name=upload`，写入一句话木马，蚁剑连接成功，flag就在根目录下，文件名是flag.php。
`核心代码：$this->Filename = $_GET['name'];`



这特么不是预期解吧，我的`eval()`函数还没用呢！
查看cmd参数，发现这家伙不是die()就是bool.....

```php
$this->cmd = "move_uploaded_file('".$file['tmp_name']."', '" . $this->Filename . "');";
eval($this->cmd);
```

使用%00截断测试home.php，白加黑拿不到flag。

解（2）：`file_get_contents($uploader);`任意文件读取。
上传一个文件，把文件内容清空防止覆盖。

```
Get参数添加?name=flag.php，得到flag

传递?name=/etc/passwd，成功读取系统文件
```

（3）预期解：Phar反序列化
这篇文章太长了，另开一篇Phar反序列化漏洞文章。

宽字节概念

单字节字符集：所有的字符都使用一个字节来表示，比如 ASCII 编码(0-127)。
多字节字符集：在多字节字符集中，一部分字节用多个字节来表示，另一部分（可能没有）用单个字节来表示。

宽字节注入时利用mysql的一个特性，使用GBK编码的时候，会认为两个字符是一个汉字。

函数addslashes()

addslashes() 函数返回在预定义字符之前添加反斜杠的字符串。
预定义字符：单引号（'），双引号（"），反斜杠（\），NULL

替换反斜杠，反斜杠的GBK编码为%5C，根据GBK编码在前面加上%DE，%DF，%E0。。。都可以组成一个汉字，从而把反斜杠给'吃'掉。

Payload：`?id=%E0' or sleep(3)%23`。

Phar反序列化

详情见专栏文章《[GXYCTF2019]BabysqliV3.0-phar反序列化正解》。