浅析MQTT安全

七月 30, 2018

近期做的项目中有涉及IoT的相关内容，其中协议使用了MQTT，因此也专门抽时间关注了其安全性问题，事实上在前两年的Defcon和BlackHat上都有人做过其安全性的演讲。目前来说，MQTT在使用中拥有相对的安全认证体系，但是其中仍然存在被恶意破解拿到权限的风险，这更大一部分不是MQTT的问题，而是使用者的问题。

MQTT概览

MQTT是一种机器对机器（M2M）的协议，它被广泛地用于 IoT 。其在1999年由IBM发明，当时是为了创建一个协议，用于通过卫星连接连接石油管道的最小电池损耗和最小带宽。因为它的耗能非常低，所以被IoT生态系统广泛采用。目前几乎所有的 IoT 云平台 都支持通过 MQTT 与几种不同实现的 IoT 智能设备发送接收数据。
官方定义：

> MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium.

其设计思想是开放、简单、轻量、易于实现。这些特点使它适用于受限环境。例如，但不仅限于此：

- 特别适合于网络代价昂贵，带宽低、不可靠的环境。
- 能在处理器和内存资源有限的嵌入式设备中运行。
- 使用发布/订阅消息模式，提供一对多的消息发布，从而解除应用程序耦合。
- 使用 TCP/IP 提供网络连接。
- 提供Last Will 和 Testament 特性通知有关各方客户端异常中断的机制。

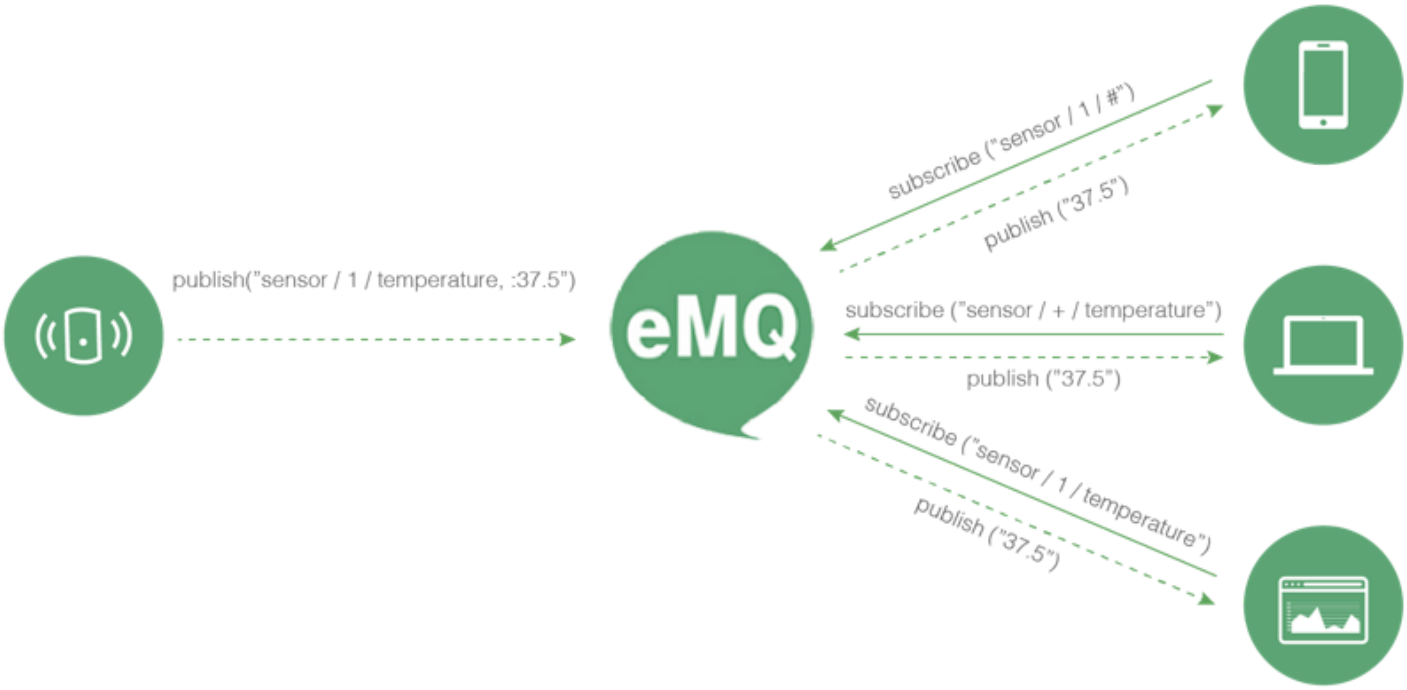
目前有许多MQTT消息中间件服务器，比如

- Mosquitto
- RabbitMQ
- Apache ActiveMQ
- HiveMQ
- EMQ
- ……

相关介绍

发布与订阅模式

MQTT是基于消息实现了发布者-订阅者模式的协议，其通过Borker其实也就是我们理解的Server来达到一个代理中转的作用，将消息接收并派发给订阅者。在转发消息的时候Borker使用`Topic`来过滤过客端。`Topic`就好比一个具体路径或者节点，客户端会有一个自己需要订阅或者发布的指定路径。也可以理解为`Topic`就像一个通道，它把发布者与它的订阅者连接起来。话题由MQTT代理管理。通过这个虚拟通道，发布者与订阅者解耦，客户端（发布者或订阅者）不必相互知道。



如上图MQTT架构，左边的传感器通过`publish`温度到Borker，右边的手机电脑等设备通过`subscribe`来订阅其实时数据，中间的EMQ Borker起到了消息的接收与转发作用。

协议报文

固定报头

Bit	7	6	5	4	3	2	1	0
Byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
Byte 2	Remaining Length							

报文格式

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

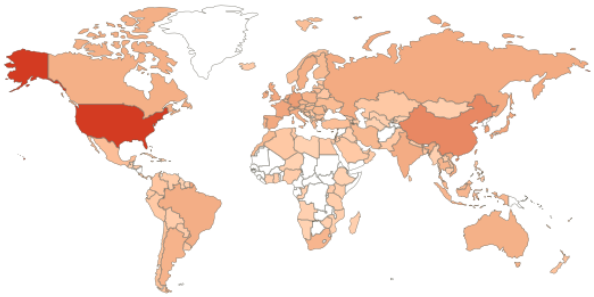
应用场景

目前物联网发展速度十分迅速，而MQTT在物联网设备的应用场景十分之广

- 智能家居
- 温度湿度传感器
- 健身器材
- 血压测量仪
- 位置服务
- 医疗设备
- ……

MQTT安全

Search for port:"1883" returned 432,082 results on 29-07-2018



Top Countries

1.	United States	364,204
2.	AP	19,054
3.	China	13,178
4.	Germany	3,586
5.	Hong Kong	2,355
6.	Korea, Republic of	2,349
7.	Taiwan	2,185
8.	France	1,646
9.	Japan	1,635
10.	Singapore	1,534

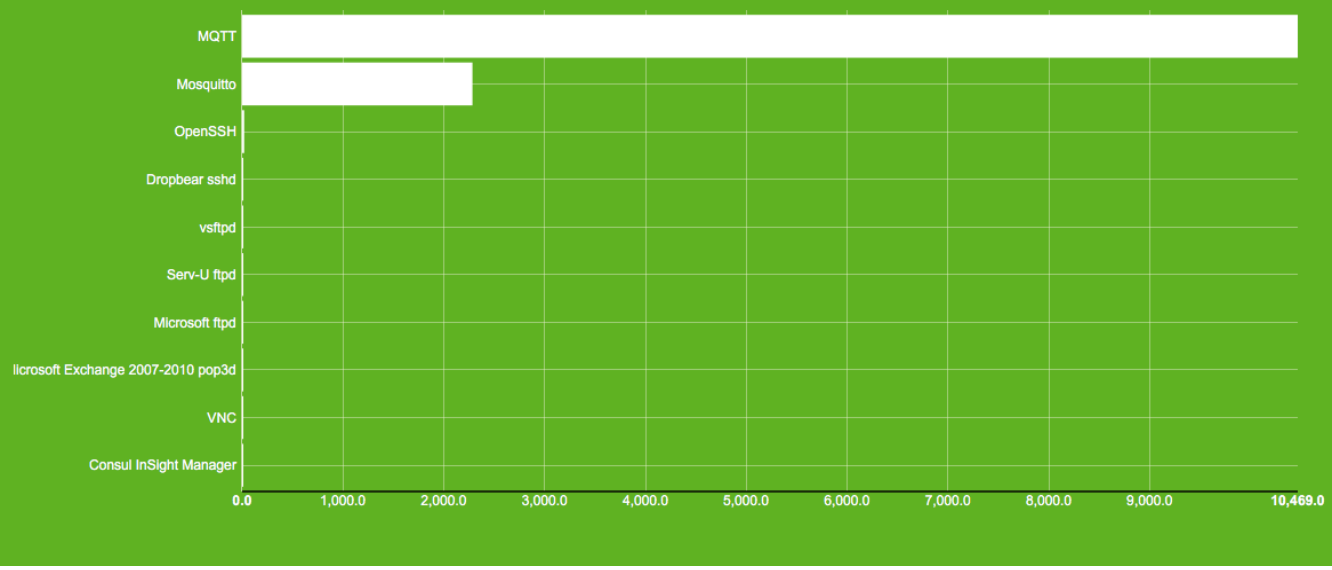
这是截止目前对国内1883端口的统计约有1万余台

Search for Country:"CN" port:"1883" returned 13,178 results on 29-07-2018

Top Cities

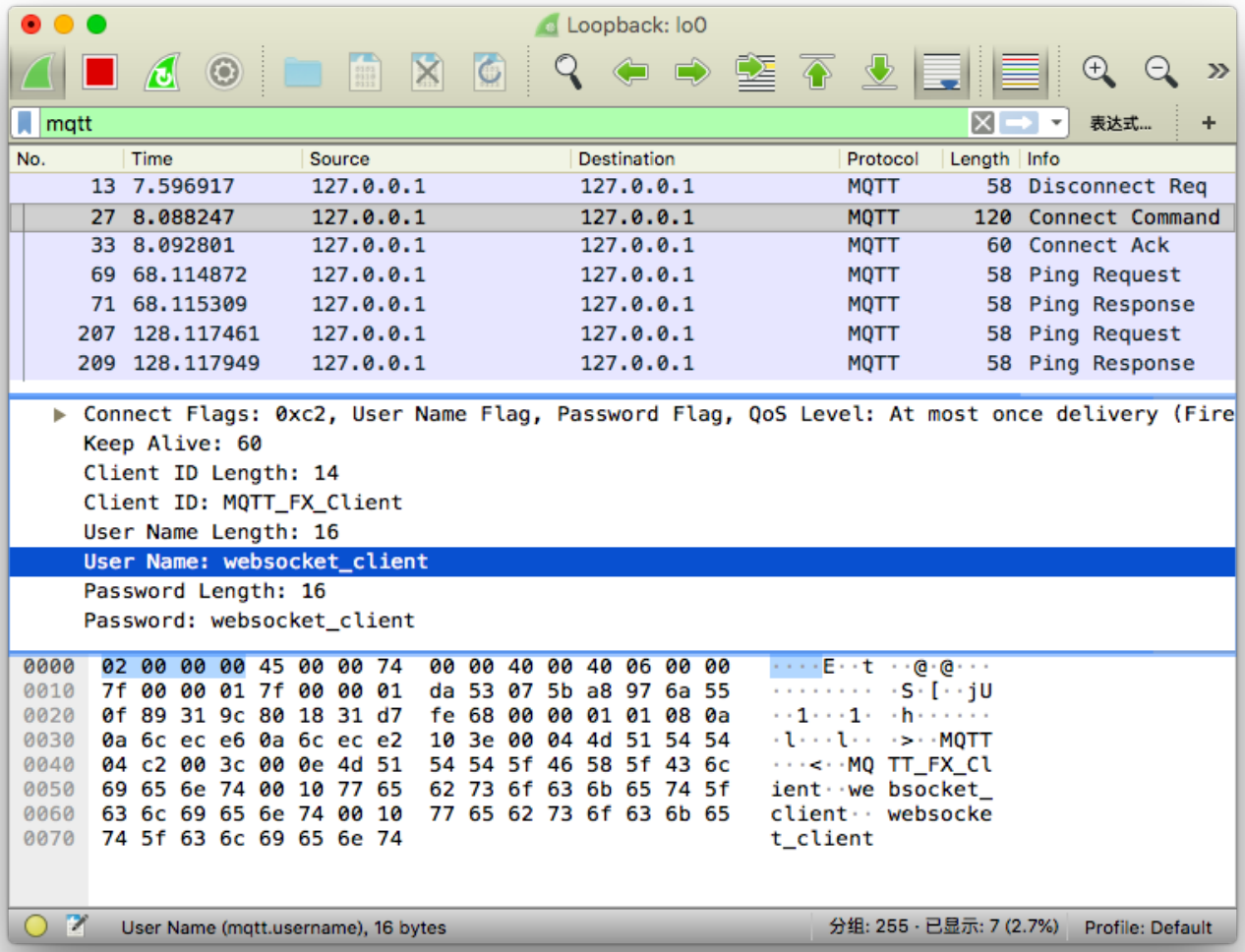
1.	Hangzhou	6,583
2.	Beijing	2,318
3.	Guangzhou	544
4.	Shanghai	430
5.	Nanjing	354
6.	Shenzhen	283
7.	Jinan	214
8.	Chengdu	204
9.	Zhengzhou	157
10.	Hefei	131

Top Products



1. MITM攻击

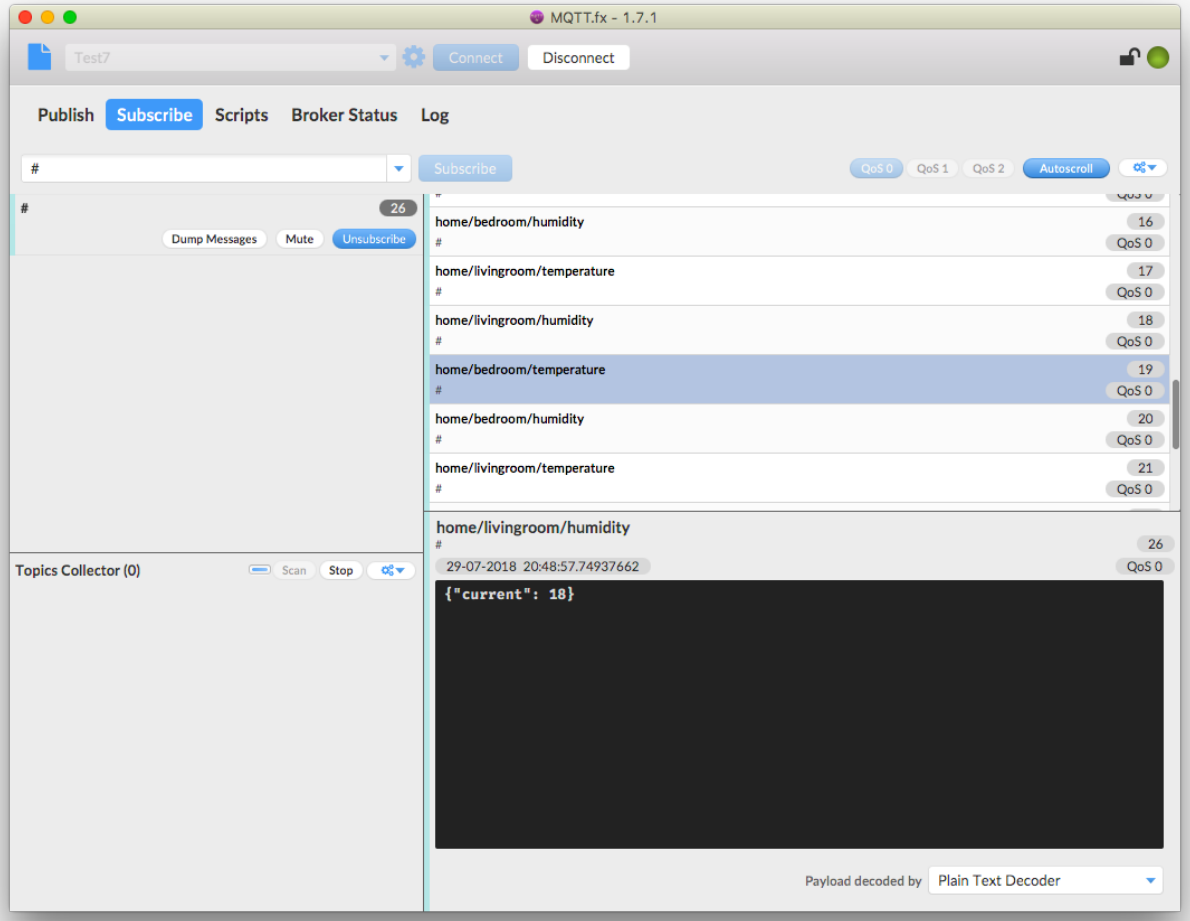
目前可以在MQTT协议V3.1中传递用户名密码来认证。MQTT基于TCP协议默认端口为1883，但容易受到MITM攻击。可以使用SSL来加密，其默认端口为8883。SSL虽然会给传输加密但是却增大了网络开销。

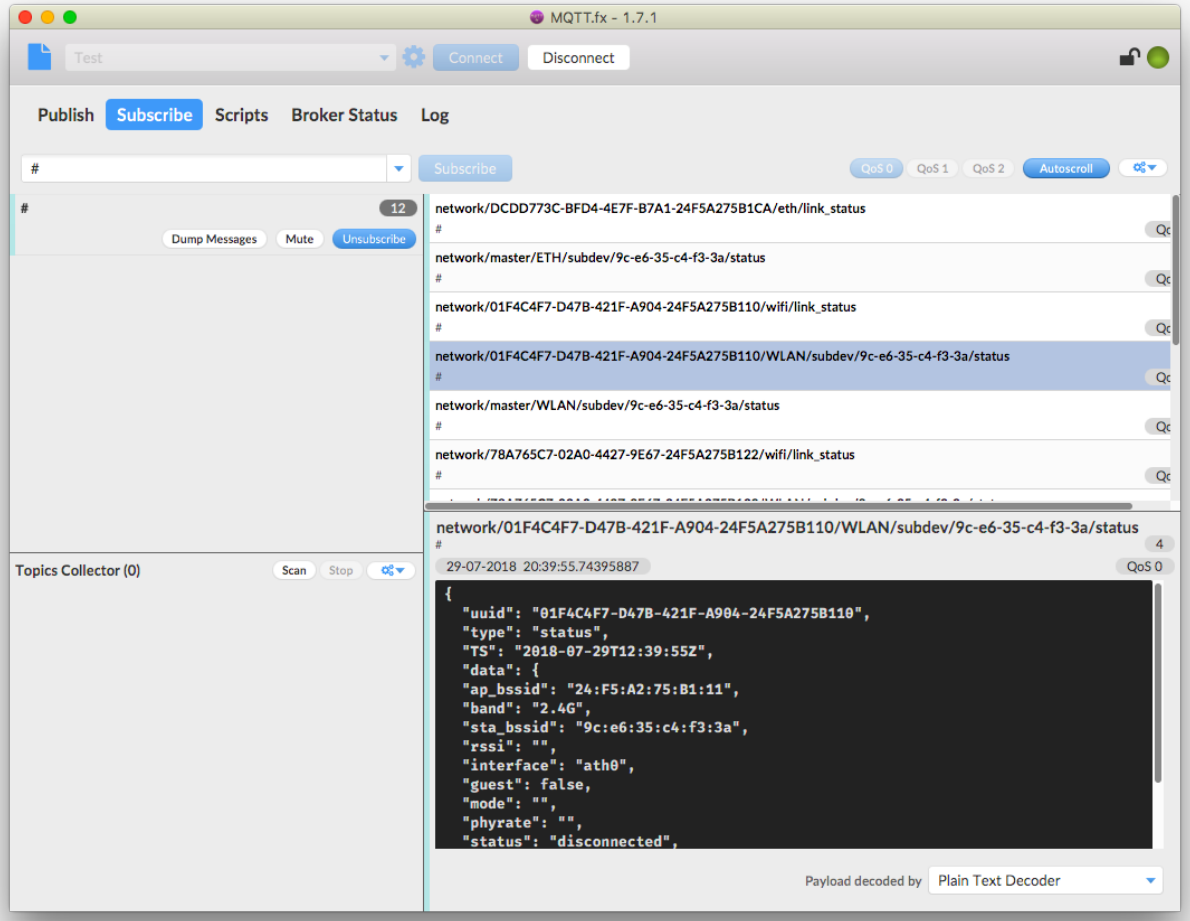


2. 未授权问题

虽然目前MQTT的消息服务器都会有相对完备的认证方式，可是经过Shodan拿到的数据发现有许多MQTT消息服务器存在配置错误，使用者没有配置认证造成未授权访问。
一旦我们进入，经过对Topic的分析，我们就可以监控全部设备，甚至发送命令控制或者SQL操作。

- **利用通配符获取订阅所有Topic**
  MQTT 主题(Topic) 支持'+', '#'的通配符，'+'通配一个层级，'#'通配多个层级(必须在末尾)。
  也就是说 如果我们的有两个个Topic分别为 `CMD/123/456` `CMD/789/666` 那么我们可以订阅 `CMD/#` 来获取其CMD下的全部消息。在攻击中我们首先就可以利用其来监听所有Topic。
  如下为两台未授权MQTT消息服务器。


MQTT.fx - 1.7.1
Test7
Connect
Disconnect
Publish
Subscribe
Scripts
Broker Status
Log
#
Subscribe
QoS 0
QoS 1
QoS 2
Autoscroll
#
26
Dump Messages
Mute
Unsubscribe
home/bedroom/humidity
16
#
QoS 0
home/livingroom/temperature
17
#
QoS 0
home/livingroom/humidity
18
#
QoS 0
home/bedroom/temperature
19
#
QoS 0
home/bedroom/humidity
20
#
QoS 0
home/livingroom/temperature
21
#
QoS 0
Topics Collector (0)
Scan
Stop
home/livingroom/humidity
#
26
29-07-2018 20:48:57.74937662
QoS 0
{"current": 18}
Payload decoded by
Plain Text Decoder



- **权限控制问题**

  前面是对无用户名密码的MQTT服务器的连接利用通配符订阅所有消息。难道有用户名和密码就可以制止了吗，类似WEB中的越权。试想一下，我们的一个设备如温度计需要通过用户名密码登录来publish温度，其通常会用设备ID等组合来形成一个key作为密码，或者就是我们的网站登陆密码吧。如果我们逆向了这个设备成功知道了用户名密码，而同时MQTT服务器配置没有对普通用户权限进行管理，允许其使用通配符，那么结果就会和上面的效果一样。

  假设如下为一个设备的配置，这里会完成登陆可是我们在MQTT服务器没有禁止其可以使用通配符

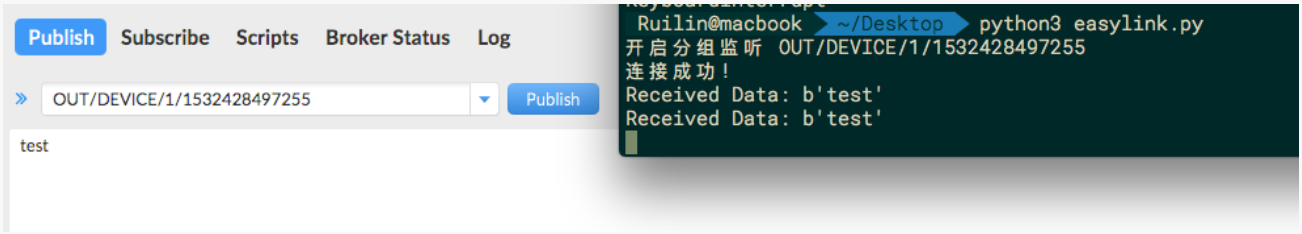
```
import paho.mqtt.client as mqtt
import time
import subprocess
import json

class SDK(threading.Thread):
    # SDK 1.2 直接用一个Key连接
    def __init__(self, host, port, key, on_message,add_group=False):
        super(SDK, self).__init__()
        self.host = host#服务器的IP
        self.port = port#服务器的MQTT端口
        self.add_group=add_group#是否监听分组
        self.key = key#连接KEY
        self.client = mqtt.Client(self.key.split("-")[2])
        self.client.username_pw_set(self.key.split("-")[2],self.key.split("-")[2])
        self.client.on_message = on_message
        self.client.on_connect = self.on_connect
        self.client.on_disconnect = self.on_disconnect

    def run(self):
        self.client.connect(self.host, self.port, 60)
        self.client.loop_forever()
```

```python
def on_connect(self, c, userdata, flags, rc):
    if rc == 0:
        self.client.subscribe("OUT/DEVICE/"+self.key.split("-")[0]+"/"+self.key.split("-")[1]+"/"+self.key.split("-")
        if(self.add_group):
            self.client.subscribe("OUT/DEVICE/"+self.key.split("-")[0]+"/"+self.key.split("-")[1])
            print("开启分组监听","OUT/DEVICE/"+self.key.split("-")[0]+"/"+self.key.split("-")[1])
        print("连接成功!")
    elif rc == 1:
        print("连接失败!MQTT协议错误!")
        self.client.disconnect()
        exit(1)
    elif rc == 2:
        print("连接失败!非法客户端标识!")
        self.client.disconnect()
        exit(1)
    elif rc == 3:
        print("连接失败!服务器访问失败!")
        self.client.disconnect()
    elif rc == 4:
        print("连接失败!账户或者密码错误!")
        self.client.disconnect()
        exit(1)
    elif rc == 5:
        print("连接失败!认证失败!")
        self.client.disconnect()
        exit(1)
    else:
        self.client.disconnect()
        exit(1)
```

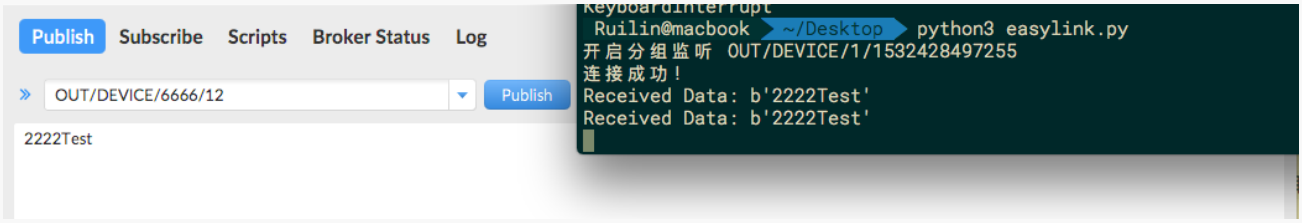
我们对其订阅的Topic发送消息，可以看到成功接收



现在我们把订阅内容改成通配符

```python
def on_connect(self, c, userdata, flags, rc):
    if rc == 0:
        # self.client.subscribe("OUT/DEVICE/"+self.key.split("-")[0]+"/"+self.key.split("-")[1]+"/"+sel
        self.client.subscribe("OUT/DEVICE/#")
        if(self.add_group):
            self.client.subscribe("OUT/DEVICE/"+self.key.split("-")[0]+"/"+self.key.split("-")[1])
            print("开启分组监听","OUT/DEVICE/"+self.key.split("-")[0]+"/"+self.key.split("-")[1])
        print("连接成功!")
    elif rc == 1:
        print("连接失败!MQTT协议错误!")
        self.client.disconnect()
        exit(1)
    elif rc == 2:
```

可以看到由于对用户没有权限验证导致其可以任意订阅内容，获取其他成员消息



- **匿名登陆问题**

  emqttd/etc/emq.conf

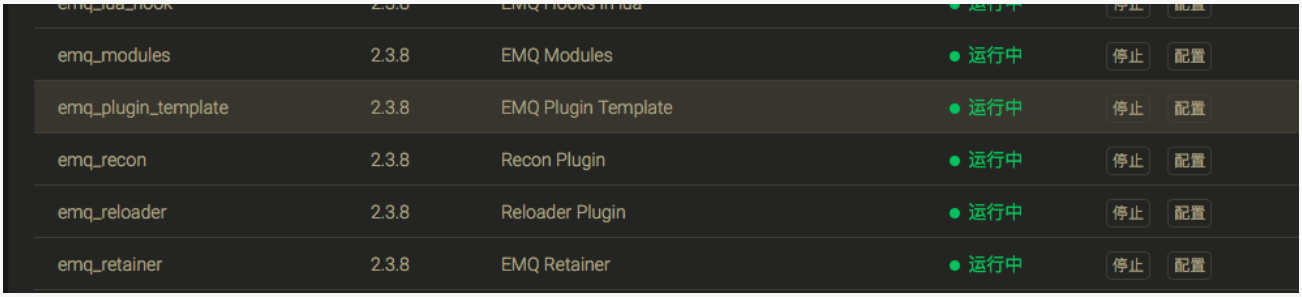
  默认是开启的任何人都能登陆 需改为false

  mqtt.allow_anonymous = true

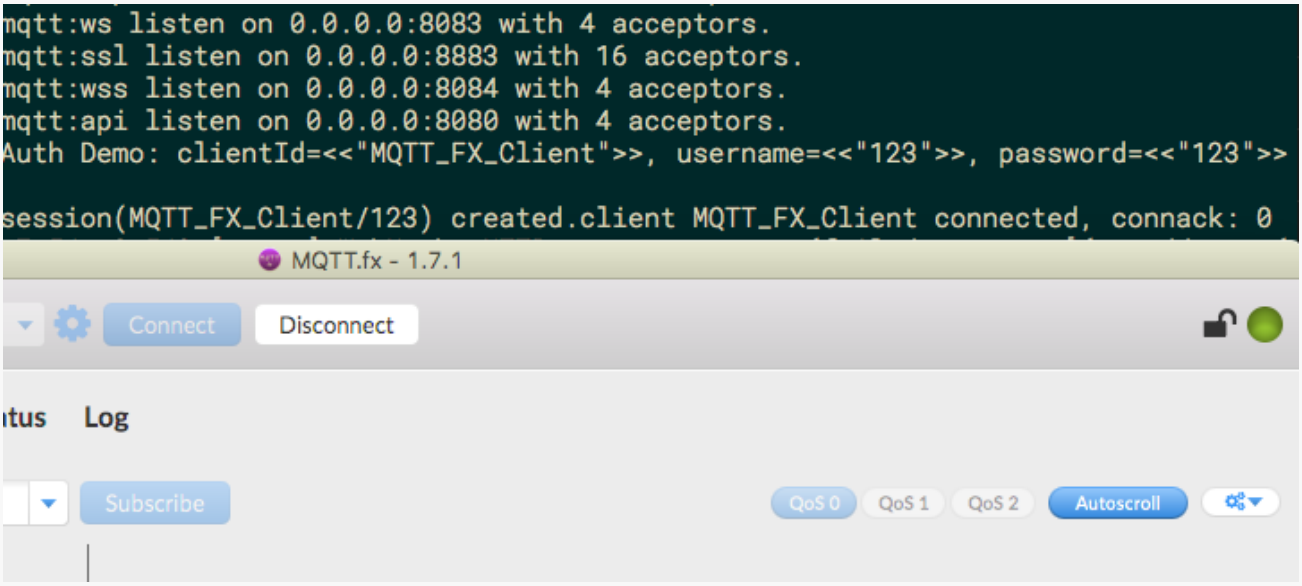
- **注意认证插件**

  目前MQTT中间件都会提供许多认证方式如MYSQL,LDAP等，但最近我使用EMQ中发现了一个坑，其中emq_plugin_template为一个开发插件的模版，它自动开启然后导致任意用户名密码都可以连接成功，如

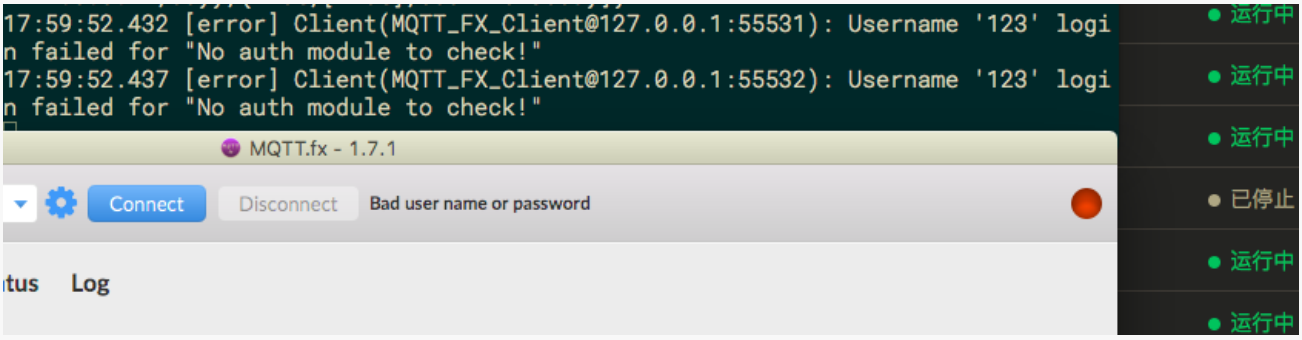
下



任意用户名密码连接



关闭后，可以正常判断



所以各位开发者在开启认证插件时一定要注意和检查，以免出现不必要的问题。

3. 暴力破解

https://github.com/zombiesam/joffrey

```
Usage: python joffrey-BH-2017.py [ARGS]

Options:
  -h, --help            show this help message and exit
  -t TARGET             Target domain or ip to invade
  -p PORT               Target port (optional)
  --threads=NRTHREADS   Amount of threads for the King to do as he please with
  -u USERNAME           Specify username
  -w WORDLIST           Path to wordlist
```

```
[*] Thread argv not supplied, setting threads to 1
[*] TARGET => 127.0.0.1
[*] PORT => 1883
[*] THREADS => 1
[*] USERNAME => admin
[*] WORDLIST => ./admin
[*] Parsed 3 passwords from ./admin
[*] Hearteater will try to strike true!
[+] Username: admin
[+] Password: admin
[*] Took a good 3 stabs to find the heart!
[*] Long live the king!
```

4.XSS

前端显示或者后端存消息切记需要注意特殊字符过滤。

安全配置

这里以EMQ为例，EMQ 消息服务器认证由一系列认证插件(Plugin)提供，系统支持按用户名密码、ClientID 或匿名认证。
系统默认开启匿名认证(anonymous)，通过加载认证插件可开启的多个认证模块组成认证链:

```
           ----------------           ----------------           ------------
Client --> | Username认证 | -ignore-> | ClientID认证 | -ignore-> | 匿名认证 |
           ----------------           ----------------           ------------
                  |                          |                        |
                 \|/                        \|/                      \|/
            allow | deny               allow | deny               allow | deny
```

> 注解
> EMQ 2.0 消息服务器还提供了 MySQL、PostgreSQL、Redis、MongoDB、HTTP、LDAP 认证插件。

etc/acl.conf 默认访问规则设置:

```
%% 允许'dashboard'用户订阅 '$SYS/#'
{allow, {user, "dashboard"}, subscribe, ["$SYS/#"]}.

%% 允许本机用户发布订阅全部主题
{allow, {ipaddr, "127.0.0.1"}, pubsub, ["$SYS/#", "#"]}.

%% 拒绝用户订阅'$SYS#'与'#'主题
{deny, all, subscribe, ["$SYS/#", {eq, "#"}]}.

%% 上述规则无匹配，允许
{allow, all}.
```

详情可见文档 http://www.emqtt.com/docs/v2/guide.html

总结

物联网是一个目前发展迅速的行业，安全影响力越来越大。MQTT无疑推动了物联网的发展，希望我们在不断方便自身的情况下，也一定要安全合理的使用相关技术。

参考资料

http://emqtt.com/docs/v2/index.html
https://morphuslabs.com/hacking-the-iot-with-mqtt-8edaf0d07b9b
https://dzone.com/articles/mqtt-security
https://www.blackhat.com/docs/us-17/thursday/us-17-Lundgren-Taking-Over-The-World-Through-Mqtt-Aftermath.pdf