[VolantMQ/volantmq [891 stars]](https://github.com/VolantMQ/volantmq)

注：我们已向厂商通报此安全问题及修复建议

#### 0x01 攻击场景与测试

考虑IoT应用的共享场景，即智能家居系统使用 MQTT 协议进行物联网设备和用户管理，其中有两个用户角色。管理员，也就是房主  
可以授权其他普通用户（例如，Airbnb 客人）访问他的智能家居设备的权利。普通用户的访问权限可能会被撤销和到期。我们  
认为管理员和设备是良性的，而客人可能是恶意的，会尽可能地去试图未授权访问设备（越权或是维持被撤销的权限）。

* **攻击场景**

首先，攻击者登记入住，因此目前攻击者拥有主题“test”的“写权限”。受害者订阅该主题  
  
  
1. 攻击者连接broker，并且设置will字段 (will message: "mywill", will topic: "test")  
  
2. 攻击者的权限被管理员或设备所有者撤销。  
  
3. 攻击者保持连接不断开，并选择一个时刻，通过主动断开网络连接或其他异常掉线的方式来触发这条will message  
  
4. will message被投递到受害者

* **漏洞危害**

攻击者可以在退房前留下一条恶意的will message，并保持连接，在其退房后，若智能门锁订阅主题“test”，攻击者可在未来某个时刻触发will message来打开门锁。

#### 0x02 漏洞测试步骤

* **测试环境**

**VolantMQ**: 0.4.0

**mqtt client**: 任意客户端即可 (paho.mqtt)

**访问控制插件**: 官方插件[http auth]([VolantMQ / vlplugin / Auth / http · GitLab](https://gitlab.com/VolantMQ/vlplugin/auth/http))（由于golang更新已不再支持plugin模块，因此这个插件目前无法使用），也可修改VolantMQ内置的auth测试插件 (见附录auth.go, 替换cmd/volantmq/auth.go)，由于漏洞的原理为broker的permission check位置不当 (或没有进行足够的检查)，而无关于permission check本身的正确与否，因此无论权限检查插件使用何种机制 (使用http请求授权服务器、使用database存储ACL等)，漏洞本身都是存在的。

配置测试用户：

admin: 拥有所有权限

user1(attacker): 拥有publish权限

配置文件如下：

version: v0.0.1  
system:  
 log:  
 console:  
 level: info # available levels: debug, info, warn, error, dpanic, panic, fatal  
 http:  
 defaultPort: 8080  
plugins:  
 enabled:  
 - auth\_http  
 config:  
 auth: # plugin type  
 - name: internal  
 backend: simpleAuth  
 config:  
 users:  
 admin: "d74ff0ee8da3b9806b18c877dbf29bbde50b5bd8e4dad7a3a725000feb82e8f1" # pass  
 user1: "e6c3da5b206634d7f3f3586d747ffdb36b5c675757b380c6a5fe5c570c714349" # pass1  
auth:  
 anonymous: false  
 order:  
 - internal  
mqtt:  
 version:  
 - v3.1.1  
 - v5.0  
 keepAlive:  
 period: 60 # KeepAlive The number of seconds to keep the connection live if there's no data.  
 # Default is 60 seconds  
 force: false # Force connection to use server keep alive interval (MQTT 5.0 only)  
 # Default is false  
 options:  
 connectTimeout: 5 # The number of seconds to wait for the CONNECT message before disconnecting.  
 # If not set then default to 2 seconds.  
 offlineQoS0: true # OfflineQoS0 tell server to either persist (true) or ignore (false) QoS 0 messages for non-clean sessions  
 # If not set than default is false  
 sessionPreempt: true # Either allow or deny replacing of existing session if there new client with same clientID  
 # If not set than default is false  
 retainAvailable: true # don't set to use default  
 subsOverlap: true # tells server how to handle overlapping subscriptions from within one client  
 # if true server will send only one publish with max subscribed QoS even there are n subscriptions  
 # if false server will send as many publishes as amount of subscriptions matching publish topic exists  
 # Default is false  
 subsId: true # don't set to use default  
 subsShared: false # don't set to use default  
 subsWildcard: true # don't set to use default  
 receiveMax: 65530 # don't set to use default  
 maxPacketSize: 268435455 # don't set to use default  
 maxTopicAlias: 65535 # don't set to use default  
 maxQoS: 2  
listeners:  
 defaultAddr: "0.0.0.0" # default 127.0.0.1  
 mqtt:  
 tcp:  
 1883:  
 auth:  
 tls:  
 ws:  
 8883:

若使用[http auth]([VolantMQ / vlplugin / Auth / http · GitLab](https://gitlab.com/VolantMQ/vlplugin/auth/http))或是附录中的auth.go，则仅需简单写一个http服务 (见附录app.py)，在broker请求/acl页面获取用户是否拥有进行敏感操作的权限时，回复"allow" (代表拥有权限)/"xxxxx"即可。

from flask import Flask, request, render\_template, session, jsonify  
from flask\_cors import CORS, cross\_origin  
import json  
import time as mytime  
from datetime import \*  
  
app = Flask(\_\_name\_\_)  
cors = CORS(app)  
  
  
@app.route('/acl', methods=['GET'])  
def Start():  
 user = request.args.get('user')  
 resp = "deny"  
 if(user == "admin"):  
 resp = "allow"  
 elif(user == "user1"):  
 resp = "allow"  
 return resp  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(host='0.0.0.0', debug=True, port=80)

* **测试步骤**

1. 攻击者连接broker，并且设置will字段 (will message: "mywill", will topic: "test")

$ mosquitto\_sub -i cid -t "test" -u user1 -P pass1 --will-topic "test" --will-payload "mywill"

若受害者订阅topic "test"

$ mosquitto\_sub -t "test" -u admin P pass

1. 攻击者的权限被管理员或设备所有者撤销。

若使用auth.go进行访问控制，则可手动控制auth server的访问控制配置来进行测试，例如当撤销attacker全新啊时，修改web服务代码app.py中的回复为deny：

from flask import Flask, request, render\_template, session, jsonify  
from flask\_cors import CORS, cross\_origin  
import json  
import time as mytime  
from datetime import \*  
  
app = Flask(\_\_name\_\_)  
cors = CORS(app)  
  
  
@app.route('/acl', methods=['GET'])  
def Start():  
 user = request.args.get('user')  
 resp = "deny"  
 if(user == "admin"):  
 resp = "allow"  
 elif(user == "user1"):  
 resp = "deny"  
 return resp

3. 攻击者保持连接不断开，并选择一个时刻，通过主动断开网络连接或其他异常掉线的方式来触发这条will message

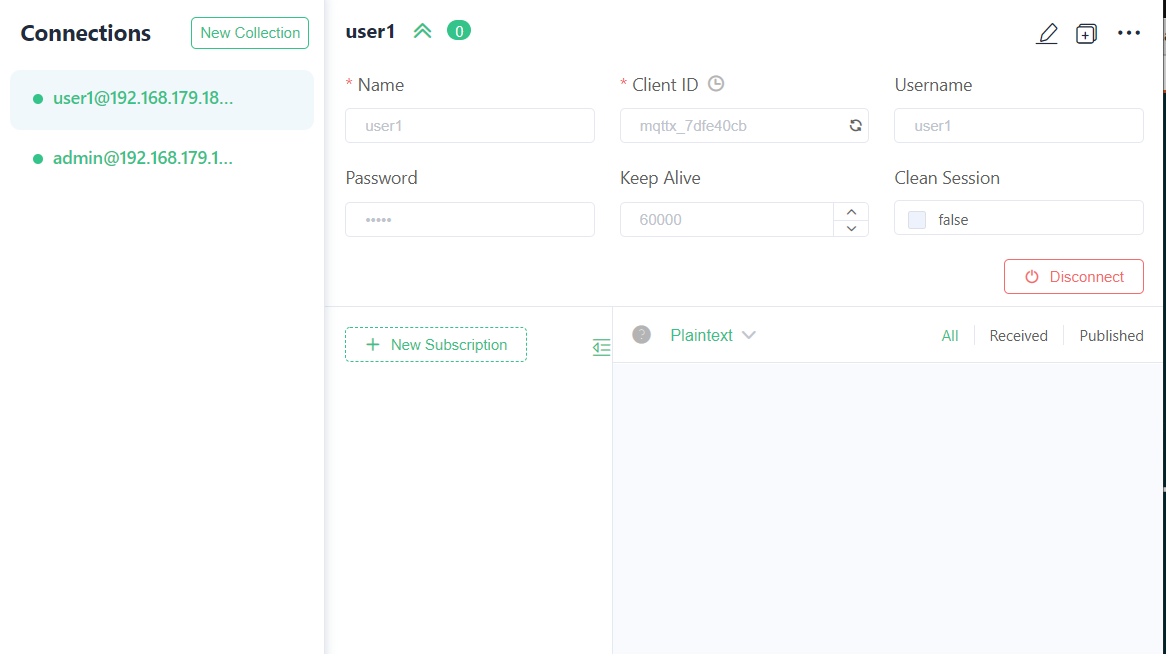
#### 0x03 漏洞效果

##### 测试前配置

测试用的账号：admin和user1

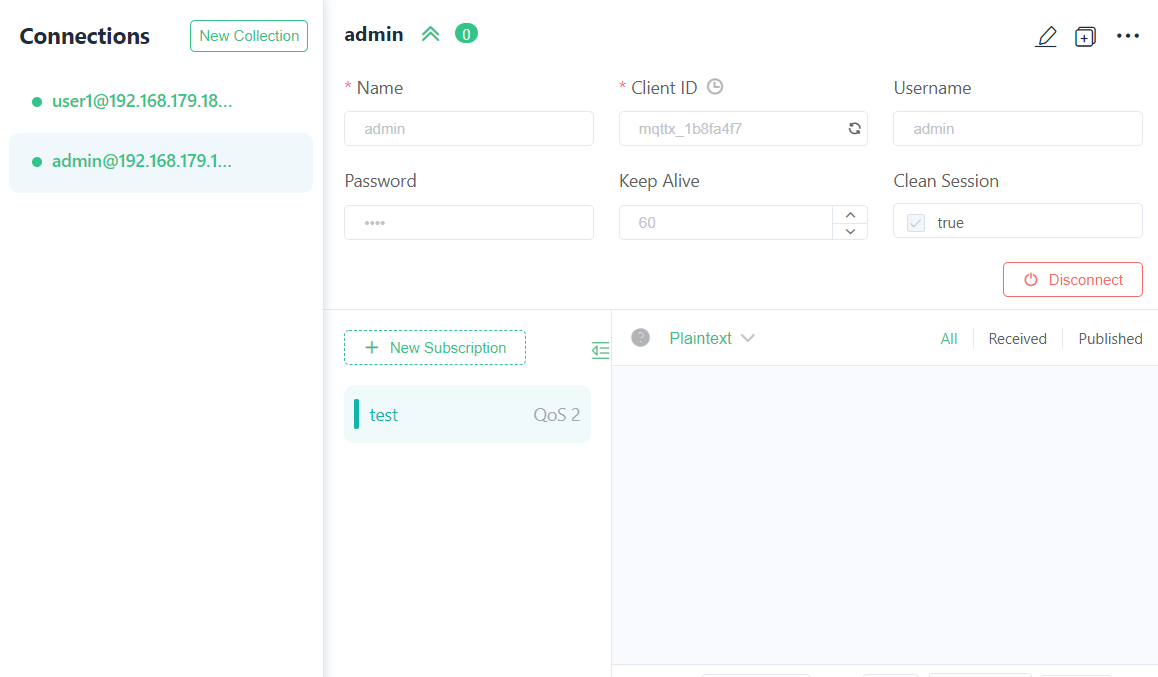
目前user1拥有test topic的发布权限

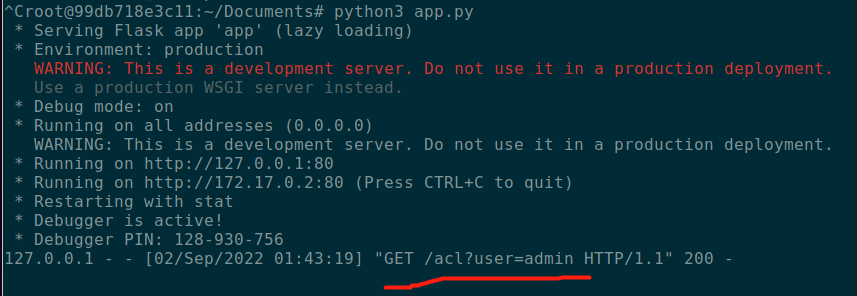
#app.py  
@app.route('/acl', methods=['GET'])  
def Start():  
 user = request.args.get('user')  
 resp = "deny"  
 if(user == "admin"):  
 resp = "allow"  
 elif(user == "user1"):  
 resp = "allow"  
 return resp



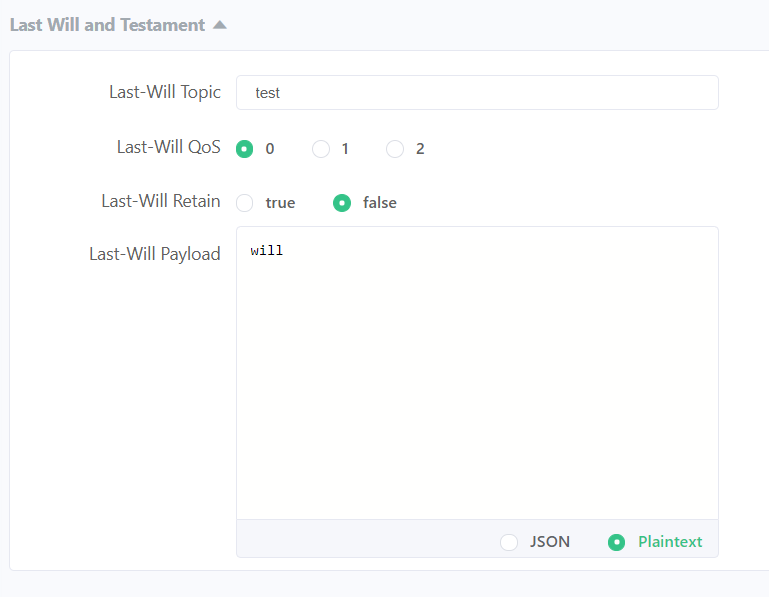
##### 测试流程

1. 使用MQTTX客户端，先使用admin用户连接broker，并订阅test主题：

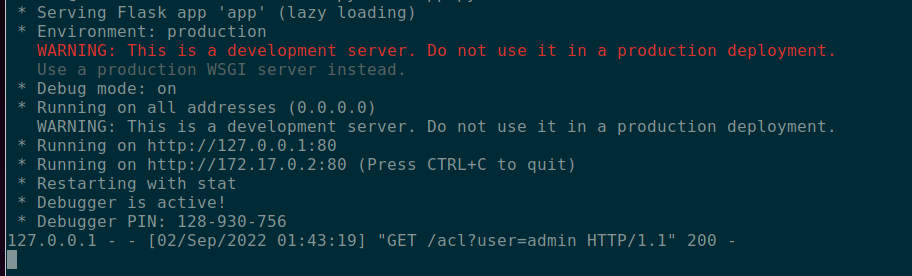




1. 攻击者连接broker，并且设置will字段 (will message: "will", will topic: "test")



注意，此时acl http服务器没有收到权限检查的请求，因此broker在客户端创建will message时没有检查其权限

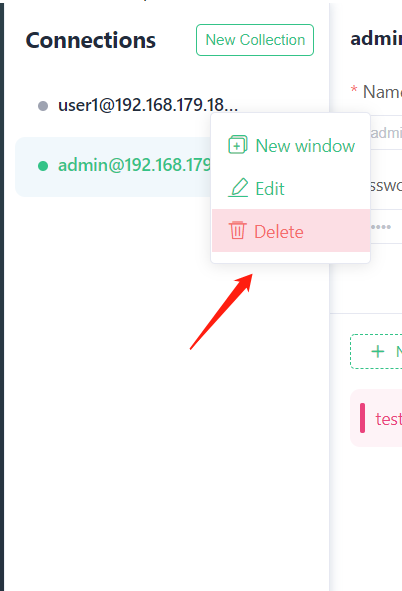


1. 攻击者的权限被管理员或设备所有者撤销。

#app.py  
@app.route('/acl', methods=['GET'])  
def Start():  
 user = request.args.get('user')  
 resp = "deny"  
 if(user == "admin"):  
 resp = "allow"  
 elif(user == "user1"):  
 resp = "deny"  
 return resp

1. 攻击者保持连接不断开，并选择一个时刻，通过主动断开网络连接或其他异常掉线的方式来触发这条will message

* 在MQTTX中，额可以通过关闭客户端或删除客户端来实现（即不发送MQTT协议中的DISCONNECT报文进行异常断开连接）



1. will message被投递到受害者

