项目：[VolantMQ/volantmq [891 stars]]

注：我们已向厂商通报此安全问题

0x01 攻击场景

- **攻击场景**

```
首先，攻击者通过猜测或是受害者泄露得到了受害者的clientID，并且攻击者是一个无权限的状态


1. 攻击者使用相同的clientID，并且以“Clean Start = False”连接broker。

2. broker会触发take over机制，将已存在的受害者session踢下线，并且将受害者session中保存的(1. 订阅关系；2. 未完成的消息)保存到新的session中。

3. broker随后触发受害者的will message。

4. 恶意的will message被投递到订阅者。

5. 攻击者继承受害者的订阅关系，无需任何权限便能继续接收消息。
```

- **漏洞危害**

1. 攻击者能继承受害者的订阅关系，能直接收取消息，而无需subscribe权限去订阅topic
2. DoS攻击，将相同clientID的受害者踢下线
3. 恶意的will message，虽然攻击者无法控制will message的内容，但是能选择触发该will message的时机，并且攻击者本身对于该will message没有权限，是一种越权行为。
4. 攻击者能够继承受害者的消息队列(QoS1/2)

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): **没有任何权限**

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. 观察者登录（admin）

> clientID: "inspector"
>
> 订阅topic: "test"

```
$ mosquitto_sub -u admin-user -P admin-password -t "test"
```

2. 受害者登录 (admin)

```
clientID: "cid"

will message: "mywill"

will topic: "test"
```

```
$ mosquitto_sub -i cid -t "test" -u admin-user -P admin-password --will-topic "test" --will-payload "mywill"
```

3. 攻击者登录 (attacker)

```
clientID: "cid"
```

```
$ mosquitto_pub -i cid -u user1 -P pass1 -t "test" -m "bad"
```

随后受害者will message "mywill"被inspector接收到

0x03 漏洞原理分析

1. 当broker向订阅者投递普通消息/retained message时，即没有检查发布者权限（仅在收到 `PUBLISH` 报文时检查，在收到 `PUBREL` 报文开始投递时没有检查），也没有检查订阅者是否拥有接收消息的权限

connection\session.go: 93

```go
// SignalPublish process PUBLISH packet from client
func (s *session) SignalPublish(pkt *mqttp.Publish) error {
    pkt.SetPublishID(s.subscriber.Hash())

    // [MQTT-3.3.1.3]
    if pkt.Retain() {
        if err := s.messenger.Retain(pkt); err != nil {
            s.log.Error("Error retaining message", zap.String("clientId", s.id), zap.Error(err))
        }
    }

    if err := s.messenger.Publish(pkt); err != nil {
        s.log.Error("Couldn't publish", zap.String("clientId", s.id), zap.Error(err))
    }
    return nil
}
```

2. 在发生session take over (MQTT spec定义的合法行为)时，没有进行检查新的session是否拥有exist session相关资源(例如订阅关系)的权限

connection\sessions.go: 351

```
    if ch, e := cn.Accept(); e == nil {
        for dl := range ch {
            var resp mqttp.IFace
            switch obj := dl.(type) {
            case *ConnectParams:
                connParams = obj
                resp, acl, e = m.processConnect(connParams, authMngr)
            case AuthParams:
                resp, e = m.processAuth(connParams, obj)
            case error:
                e = obj
            default:
                e = errors.New("unknown")
            }

            if e != nil || resp == nil {
                cn.Stop(e)
                cn = nil
                return nil
            }

            if resp.Type() == mqttp.AUTH {
                _ = cn.Send(resp)
            } else {
                ack = resp.(*mqttp.ConnAck)
                break
            }
        }
    }
```

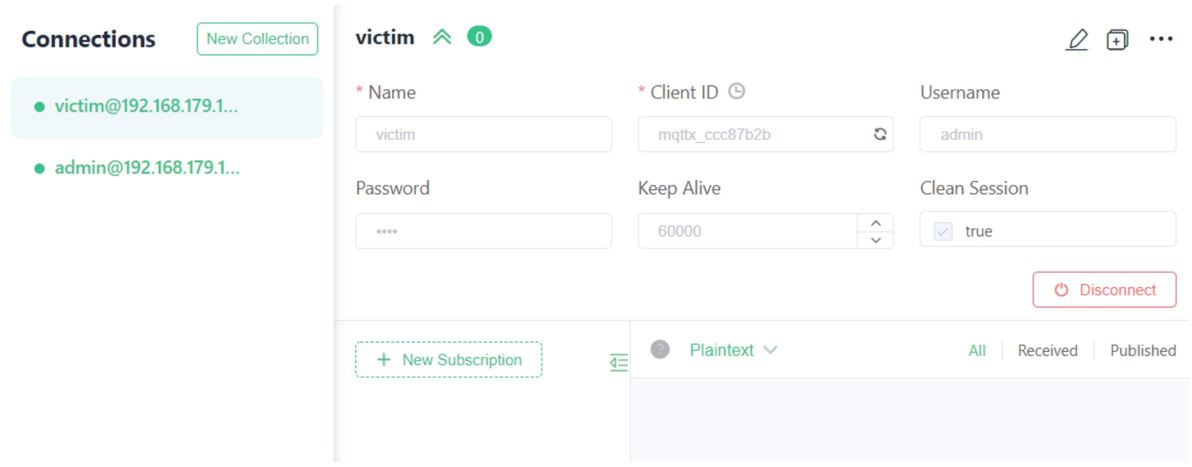
0x04 漏洞效果

测试前配置

测试用的账号：admin和user1

目前user1没有任何权限

```
#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 = "deny"
    return resp
```



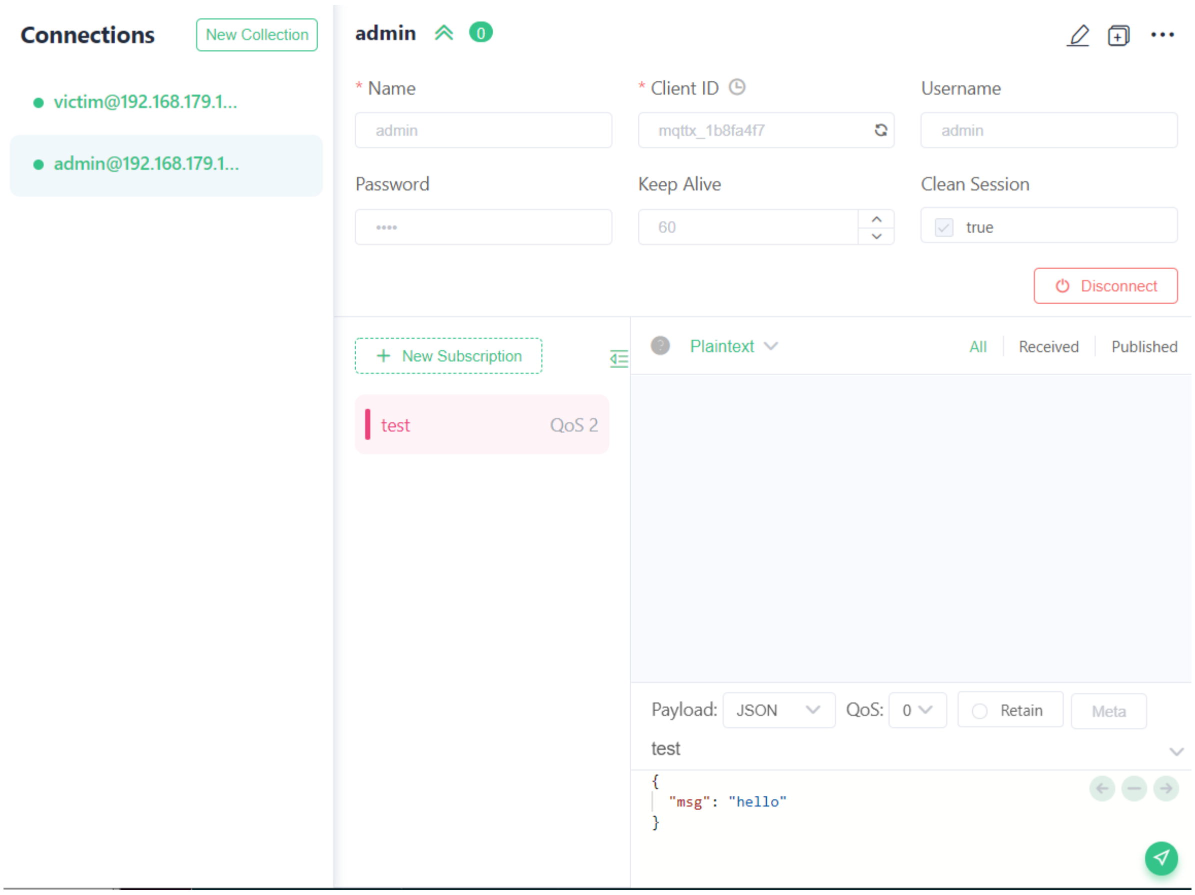
* Name	* Client ID	Username
victim	mqttx_ccc87b2b	admin
Password	Keep Alive	Clean Session
••••	60000	true

测试流程

1. 观察者登录（admin）

```
clientID: "mqttx_1b8fa4f7"

订阅topic: "test"
```


Connections New Collection

victim@192.168.179.1...

admin@192.168.179.1...

admin 0

* Name: admin

* Client ID: mqttx_1b8fa4f7

Username: admin

Password: ••••

Keep Alive: 60

Clean Session: true

Disconnect

New Subscription

test QoS 2

Plaintext All Received Published

Payload: JSON QoS: 0 Retain Meta

test

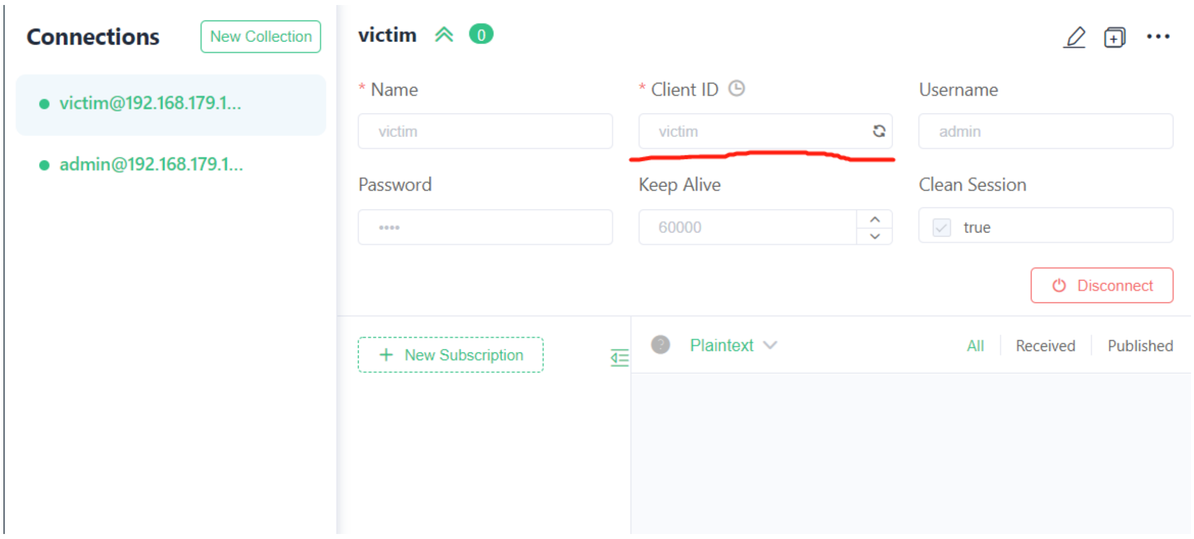
```
{
  "msg": "hello"
}
```

2. 受害者登录 (admin)

```
clientID: "victim"

will message: "will"

will topic: "test"
```


Connections New Collection

victim@192.168.179.1...

admin@192.168.179.1...

victim 0

* Name: victim

* Client ID: victim

Username: admin

Password: ••••

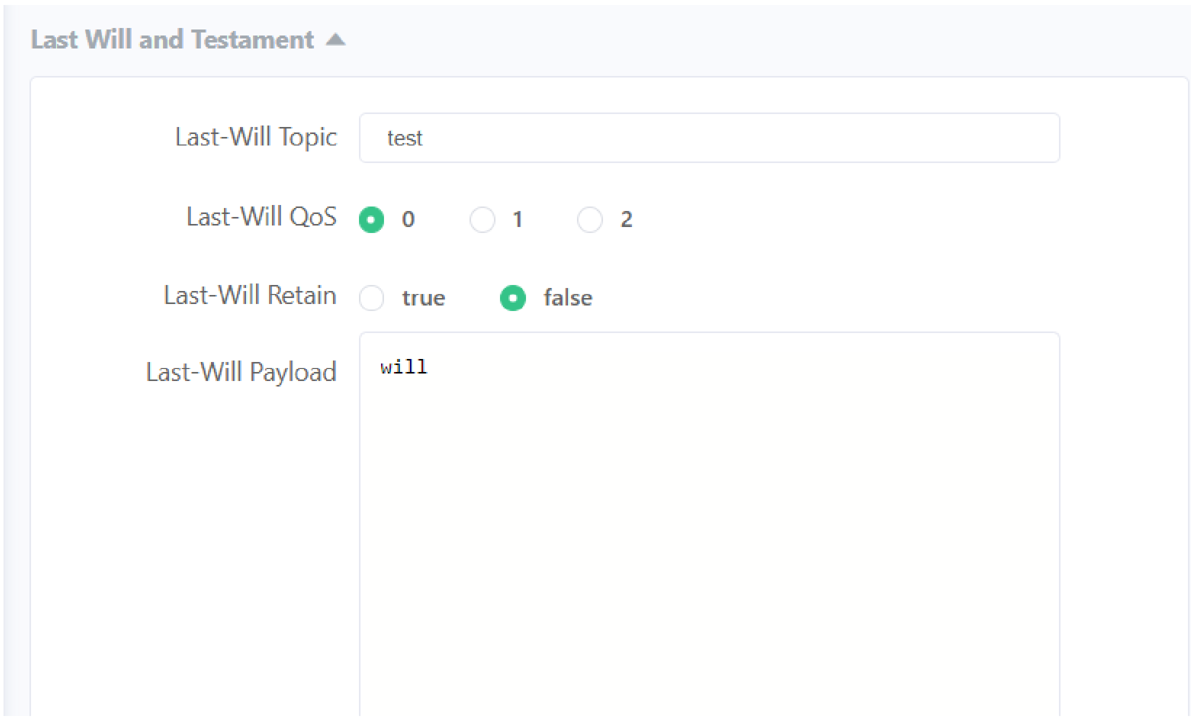
Keep Alive: 60000

Clean Session: true

Disconnect

New Subscription

Plaintext All Received Published


Last Will and Testament

Last-Will Topic: test

Last-Will QoS: [✓ 0] 1 2

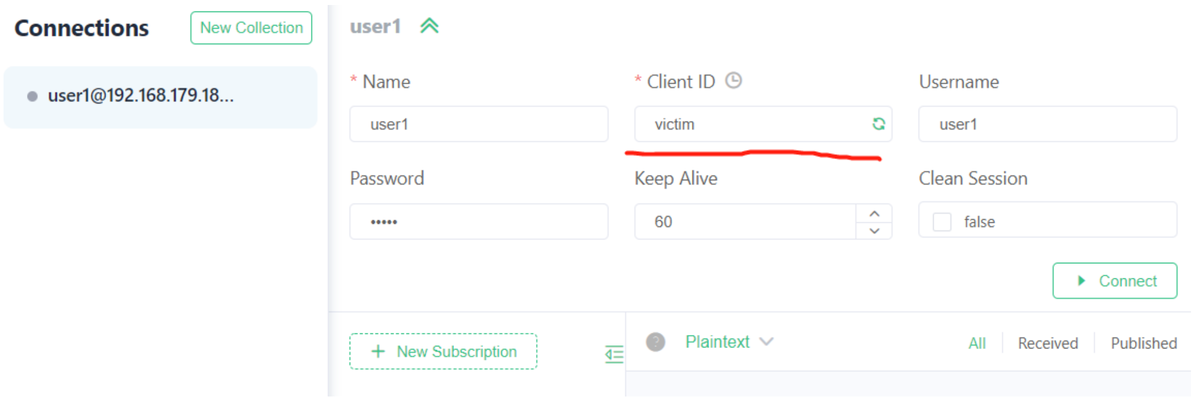
Last-Will Retain: true [✓ false]

Last-Will Payload:

```
will
```

3. 攻击者登录 (attacker)

```
clientID: "victim"
```



Name: user1

Client ID: victim

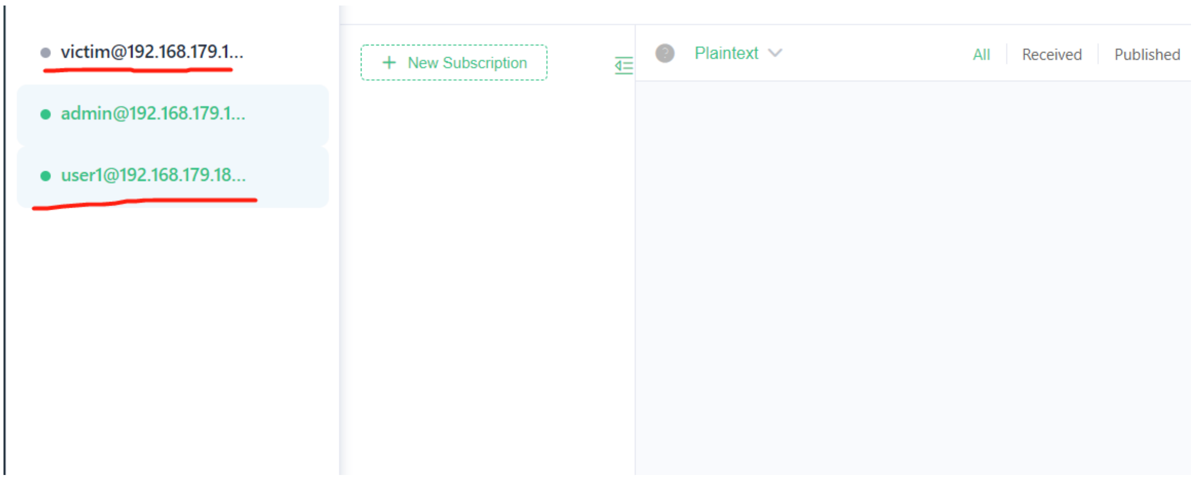
Username: user1

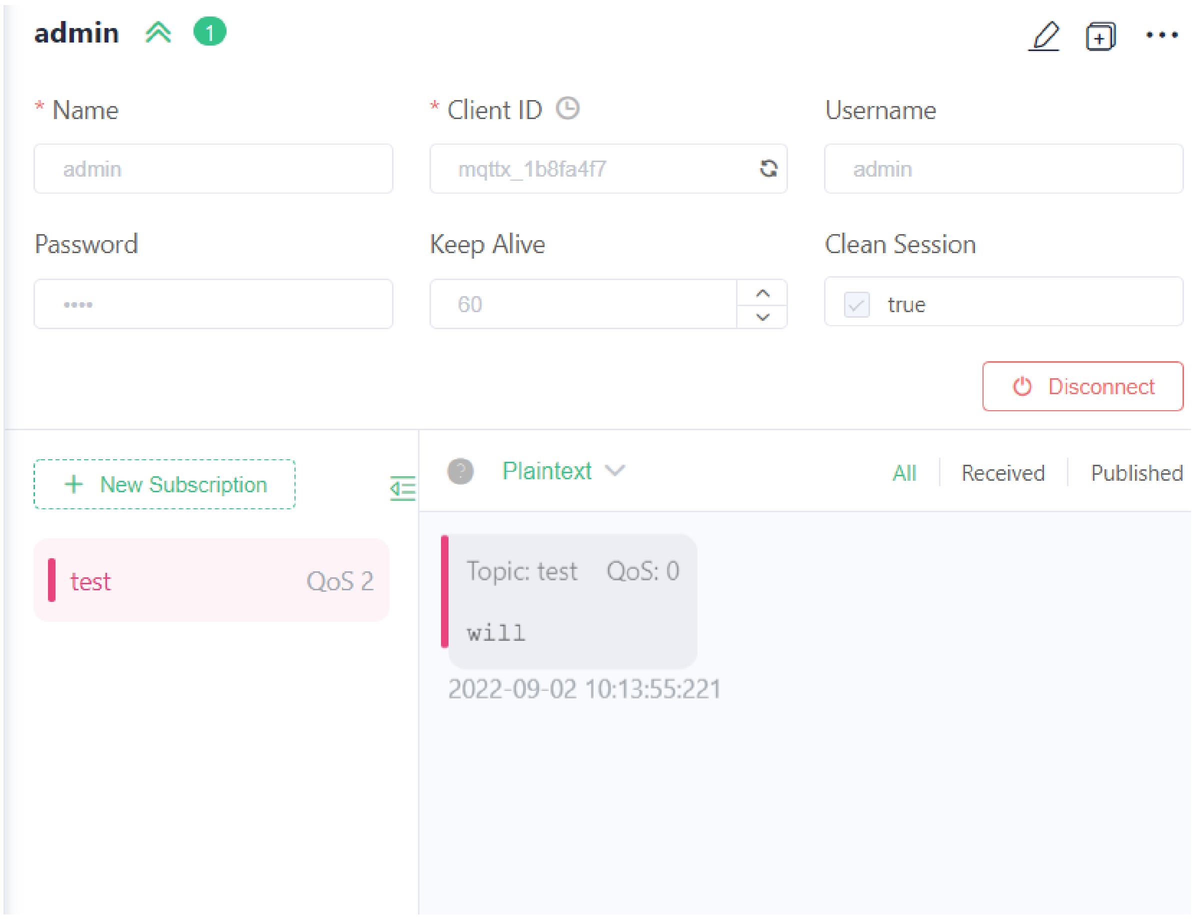
Password: •••••

Keep Alive: 60

Clean Session: false

随后受害者被抢占下线, 并且它的will message "will"被inspector接收到





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

test QoS 2

Topic: test QoS: 0

```
will
```

2022-09-02 10:13:55:221