MQTTSA Report

Details of the assessment

Test configuration		Vulnerabilities	
Broker host and port	127.0.0.1:1883	Outdated Broker	Yes
Listening time	60 seconds	Use of TLS	False*
Message to send	testtesttest	Information Disclosure	True
Sniffing interface	None	Accessible service	True
Data/Msg tampering	True	or weak Access Control	
Brute-forcing	False	Unlimited** payload	Skipped
Flooding DoS conn.	None	Unlimited** connections	Skipped
- Payload size	None	Unlimited** msg queues	Skipped
Slow DoS conn.	None	Overall risk	HIGH

*: False if not providing X.509 certificates or according to the broker implementation (e.g., Mosquitto). Verify with TLS Assistant (https://github.com/stfbk/tlsassistant).
**: With respect to provided parameters.

Authentication

[!] MQTTSA did not detect any authentication mechanism
The tool was able to connect to the broker without specifying any kind of credential information. This may cause remote attackers to successfully connect to the broker. It is strongly advised to support authentication via X.509 client certificates.

Suggested mitigations

Please modify Mosquitto's configuration ("/etc/mosquitto/mosquitto.conf" by default) according to the official documentation. For further authentication or authorization mechanisms refer to mosquitto-go-auth. An excerpt of "mosquitto.conf" is provided below:

```
listener 1883 127.0.0.1 #Binds Mosquitto to a specific IP/Port
cafile <path to the certificate authority certificate > #Tipically /etc/mosquitto/certs/ca.crt
certfile <path to Mosquitto X.509 certificate > #Tipically /etc/mosquitto/certs/hostname.crt
keyfile <path to Mosquitto private key > #Tipically /etc/mosquitto/certs/hostname.key
crlfile <path to Mosquitto certificate revocation list > #Tipically /etc/mosquitto/certs/ca.crl
require_certificate true #The client must present a certificate
use_identity_as_username true #- And the certificate Common Name (CN) is used as username
use_username_as_clientid true #- And the username is used as the unique client ID
```

By using "require_certificate", an attacker need to have access to a valid certificate (and the corresponding private key), rather than the less secure username and password (that can be possibly bruteforced). In addition, by indicating both "use_identity_as_username" and "use_username_as_clientid", an attacker that steals and use the client certificate (and the key) cannot be connected together with the client (as the client ID is unique); hence, the client will be disconnected and possibly detect the attack. Upon detection (or if the client is not authorised

anymore), the certificate can be revoked via the certificate revocation list.

Information disclosure

MQTTSA waited for 60 seconds after having subscribed to the '#' and '$SYS/#' topics. By default, clients who subscribe to the '#' topic can read to all the messages exchanged between devices and the ones subscribed to '$SYS/#' can read all the messages which includes statistics of the broker. Remote attackers could obtain specific information about the version of the broker to carry on more specific attacks or read messages exchanged by clients.

[!] MQTTSA successfully intercepted all the messages belonging to 43 topics, 0 of them non $SYS. Intercepted data was stored in the 'messages' folder.
The SYS topics are: ['$SYS/broker/store/messages/count', '$SYS/broker/load/messages/received/1min', '$SYS/broker/load/sockets/5min', '$SYS/broker/load/connections/5min', '$SYS/broker/publish/bytes/sent', '$SYS/broker/load/bytes/sent/15min', '$SYS/broker/clients/total', '$SYS/broker/version', '$SYS/broker/retained messages/count', '$SYS/broker/load/publish/received/15min', '$SYS/broker/messages/received', '$SYS/broker/load/messages/received/5min', '$SYS/broker/load/publish/sent/5min', '$SYS/broker/messages/sent', '$SYS/broker/load/sockets/1min', '$SYS/broker/messages/stored', '$SYS/broker/clients/connected', '$SYS/broker/bytes/sent', '$SYS/broker/store/messages/bytes', '$SYS/broker/load/bytes/received/1min', '$SYS/broker/subscriptions/count', '$SYS/broker/load/bytes/received/5min', '$SYS/broker/load/sockets/15min', '$SYS/broker/publish/messages/sent', '$SYS/broker/load/bytes/sent/1min', '$SYS/broker/bytes/received', '$SYS/broker/load/connections/1min', '$SYS/broker/load/publish/received/1min', '$SYS/broker/load/messages/sent/15min', '$SYS/broker/uptime', '$SYS/broker/heap/current', '$SYS/broker/publish/bytes/received', '$SYS/broker/load/publish/sent/1min', '$SYS/broker/load/messages/sent/5min', '$SYS/broker/load/publish/received/5min', '$SYS/broker/load/connections/15min', '$SYS/broker/load/messages/received/15min', '$SYS/broker/clients/active', '$SYS/broker/load/bytes/sent/5min', '$SYS/broker/load/publish/sent/15min', '$SYS/broker/publish/messages/received', '$SYS/broker/load/bytes/received/15min', '$SYS/broker/load/messages/sent/1min']

Suggested mitigations

It is strongly recommended to enforce an authorization mechanism in order to grant the access to confidential resources only to the specified users or devices. There are two possible approaches: Access Control List (ACL) and Role-based Access Control (RBAC).
If restricting access via ACLs, please follow those guidelinesand modify Mosquitto's configuration according to the official documentation. For instance, integrate the *acl_file* parameter (*acl_file /mosquitto/config/acls*) and restict a client to interact only on topics with his clientname as prefix (ACL *pattern readwrite topic/%c/#*).
In addition, consider the adoption of TLS 1.3 by setting *tls_version tlsv1.3* in "mosquitto.conf" - the strongest cipher is used by default, but the client may require a weak one: use *ciphers_tls1.3* in "mosquitto.conf" to indicate a secure cipher list(if not working, use TLS 1.2 and the *ciphers* parameter).

Tampering data

After having successfully intercepted some messages, MQTTSA automatically created a new message (having as a payload the string 'testtesttest') and attempted sending it to every topic it was able to intercept. Remote attackers could exploit it to write in specific topics pretending to be a client (by his ID); e.g., send tampered measures to a sensor.

MQTTSA was not able to write in any topic.

Broker Fingerprinting

MQTTSA detected the following MQTT broker: mosquitto version 2.0.11.
[!]Mosquitto version is not updated: please refer to the last Change logfor bugs and security issues.

Malformed data

MQTTSA tried to stress the broker by sending malformed packets in the Topic1 topic.
An attacker could send malformed packets aiming at triggering errors to cause DoS or obtain information about the broker. We suggest to perform a full fuzzing test to stress the implementation with random well-crafted values. A fuzzer designed for MQTT is developed by F-Secure and can be found on the following link:
[Fuzzer F-Secure]

Parameter of the CONNECT packet tested: client_id

Values that did not generate an error:
test, $, $topic, ,
testtesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttes
ttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttestte
sttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttestt
esttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttest
testtesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttest

Values that generated an error and the related error:

Parameter of the CONNECT packet tested: clean_session

Values that did not generate an error:
True, 1, 2, -1

Values that generated an error and the related error:
Value: False, Error: A client id must be provided if clean session is False.
Value: 0, Error: A client id must be provided if clean session is False.

Parameter of the CONNECT packet tested: userdata

Values that did not generate an error:
test, $, $topic, ,
testtesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttes
ttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttestte
sttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttestt
esttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttest
testtesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttest

Values that generated an error and the related error:

Parameter of the CONNECT packet tested: keepalive

Values that did not generate an error:
0, 1, 2, 3, 234

Values that generated an error and the related error:
Value: -1, Error: Keepalive must be >=0.
Value: -100, Error: Keepalive must be >=0.
Value: 0.12, Error: required argument is not an integer
Value: -0.12, Error: Keepalive must be >=0.
Value:
893427908127340981723498712309487120937492813749721394719023740971230948710293847091273409871230
49710293749128374097239017409237409123749071209347091237490321, Error: argument out of range
Value:

-19283491820374981273498712390470923874097231049712309479237490127304972109348712930749231749213790470123470927 34, Error: Keepalive must be >=0.

Parameter of the PUBLISH packet tested: topic

Values that did not generate an error:
$, $topic, ///////, /../../../../

Values that generated an error and the related error:
Value: #, Error: Publish topic cannot contain wildcards.
Value: /#/#/#, Error: Publish topic cannot contain wildcards.

Parameter of the PUBLISH packet tested: payload

Values that did not generate an error:
test, $, $topic, ,
testtesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttes
ttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttestte
sttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttestt
esttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttest
testtesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttesttest

Values that generated an error and the related error:

Parameter of the PUBLISH packet tested: qos

Values that did not generate an error:
0, 1, 2

Values that generated an error and the related error:
Value: 3, Error: Invalid QoS level.
Value: -1, Error: Invalid QoS level.
Value: -100, Error: Invalid QoS level.
Value: 234, Error: Invalid QoS level.
Value: 0.12, Error: unsupported operand type(s) for <<: 'float' and 'int'
Value: -0.12, Error: Invalid QoS level.
Value:
8934279081273409817234987123094871209374928137497213947190237409712309487102938470912734098712304971029374912837409723901740923740912374907120934709123749032 1, Error: Invalid QoS level.
Value:
-19283491820374981273498712390470923874097231049712309479237490127304972109348712930749231749213790470123470927 34, Error: Invalid QoS level.

In case the report refer to values like '$' or '$topic', it might be possible to be exploit a bug included in an old version of Mosquitto. We strongly suggest to always keep the broker updated to avoid similar issues.