SI5/M2II - IoT Security Lab
October 10th, 2022
Yves Roudier - UCA / Polytech Nice Sophia

This lab aims at giving you a hands-on experience with security issues with the Message Queuing Telemetry Transport Protocol (MQTT), a common protocol for connecting IoT devices together, using the Mosquitto broker and see how it might be attacked and protected.

1) MQTT broker setup and fingerprinting

From your Kali box (or a linux distrib), you can perform the install as follows:
sudo apt install mosquitto
sudo apt-get install mosquitto-clients

The broker can be started or restarted using:
systemctl restart mosquitto

and you can check its status using the command:
systemctl status mosquitto

Finally, the configuration file can be visualized and edited from /etc/mosquitto/mosquitto.conf

The broker can also be installed natively on other OSes (check the instructions on the website (https://mosquitto.org/download/)

Now that the broker is up and running, you can try to check whether the service is indeed available from the standard port using nmap (this is the kind of fingerprinting that hackers commonly perform):
nmap -p- localhost

You can also know more about the exact version of the MQTT broker with (the slightly lengthier command):
nmap -p- -sV localhost

Or the even more precise:
nmap -p1883 -sV -sC localhost

2) Observing MQTT messages

MQTT uses a publish/subscribe mechanism for distributing messages based on topic-related notification. We will first experiment with these to understand the messages exchanged.

Fire up wireshark and configure it to listen to your localhost.
After starting the broker, open two other terminal windows.
In each of these windows, start two subscribers on the same topic using the following command:
mosquitto_sub -h localhost -t topic/hello

Now do a publish operation as follows:
mosquitto_pub -h localhost -t topic/hello -m “Hello World !”

Now stop the second subscriber and make it subscribe to “topic/alert”.
Redo the publication and make sure that the second subscriber did not receive the message this time.
You can now also test how to listen to all topics using wildcard “#” for the topic description.

What do you observe in the Wireshark captures? You can compare your captures with the packet structure outlined here: http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/
Is MQTT as scalable as touted?
What do you think of using MQTT without additional security mechanisms?

3) MQTT with a password

Access to the MQTT publish/subscribe can be controlled using a password. Update the configuration file to add user “alice” with password “tiffany”, and “bob” with password “banana”, as explained in the following page:
http://www.steves-internet-guide.com/mqtt-username-password-example/.

Restart the Mosquitto server.
Now can you listen to published messages or publish on your own without a login/password?

Try instead with with:
mosquitto_sub -h localhost -t topic/hello -u bob -P banana
mosquitto_pub -h localhost -t topic/hello -u alice -P tiffany

Create an ACL file and register it in the mosquitto.conf file by following the instructions on the following page, so that Alice an publish and subscribe messages about topic/hello, but Bob can only read the data from that topic:
http://www.steves-internet-guide.com/topic-restriction-mosquitto-configuration/
Try to send a message on Bob’s part now.
Try to listen or post to a different topic too.

Based on Wireshark captures, is the situation any different from what you observed previously?

Now open Metasploit using:
msfconsole

Look for MQTT related attacks:
search mqtt

Now we can use the exploit available here:
use auxiliary/scanner/mqtt/connect

You can display the options to configure the attack as follows:
show options

Now let’s use the following options and run the exploit:

set RHOSTS 127.0.0.1
set USERNAME alice
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/passwords.lst
set USER_FILE “”
set VERBOSE false
exploit

Now try to listen to incoming messages using the same publish subscribe as above:
mosquitto_sub -t “#” -h localhost -u alice -P found_password

4) MQTT with encryption

MQTT can make use of TLS encryption to protect the content of messages. Change its configuration to enable this mechanism as described in https://openest.io/linux-embarque/communication-linux/chiffrement-communication-mqtt-tls-ssl-mosquitto-et-paho/.

Don’t forget to configure the clients (publishers/subscribers) in addition to the broker.

Run the same experiments as before. What do you observe with Wireshark?

5) Fuzzing MQTT

It is also possible to fuzz test the MQTT protocol implementation (in our case with the Mosquitto broker) and to try and find vulnerabilities. It is possible to find denial of service attacks for instance. Running the fuzzer produces unpredictable results due to the randomness of the testing and it may even take a few hours before a glitch can be found for instance.

We will be experimenting the mqtt_fuzz python tool written by F_Secure on top of the Radamsa fuzzing framework.

In order to install Radamsa first, you will need to run the following commands on top of your Kali distrib:
sudo apt install gcc git make wget
git clone https://gitlab.com/akihe/radamsa.git && cd radamsa && make && sudo make install

If you want to check the correct installation of the fuzzing framework and at the same time see how it behaves, you can type:
echo “Hello World” | radamsa

Now download mqtt_fuzz (https://github.com/F-Secure/mqtt_fuzz.git).

Finally run it:
Python mqtt_fuzz.py localhost 1883

You can check examples of vulnerabilities that might have been exploited in the past on top of Mosquitto here:
https://mosquitto.org/security/
https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=mosquitto