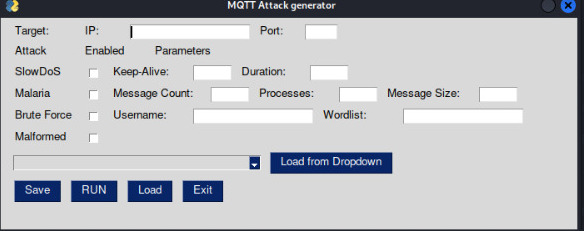
**Tools and their essence:**

**Overall View:**

This is a GUI application written in Python which implements 4 kinds of MQTT attacks (the individual attacks are basically individual tools that were taken from GitHub). The attacks are namely SlowDoS, Flooding-DoS, Brute Force and Malformed Attack. The GUI tool has multiple input parameters, the most important of them being the IP address and the port number of the MQTT server. Each attack has a checkbox beside it which is used to enable or disable the attack, alongwith other parameters specifically designed for a particular attack. The GUI tool can just be run from the command line and there is no need to install anything in particular. But, each attack tool needs some set of softwares to be installed for them to function properly. The steps to installing these dependencies are provided in the individual GitHub links. There are also buttons to save and load the configurations in the form of a .yaml file. A drop down button is also provided which lets the user see and load all the previously loaded files (configuration history).  
  
Important note – Please run the command **ulimit –n 4096** before doing the SlowDoS attack.



**Time for each attack:**  
SlowITe – 10 seconds

Flooding DoS – Depends on the number of processes

Brute Force – Depends whether the password has been cracked

Malformed – 60 seconds.

There is a delay of 10 seconds in between each attack.

**MQTT-SlowITe:**

[GitHub - gianluca2414/MQTT\_SlowITe](https://github.com/gianluca2414/MQTT_SlowITe)

**python3 MQTT\_SlowDoS.py -a broker\_address [-p broker\_port] [-k keep\_alive] [-h | --help]**

MQTT\_SlowDoS.py: This script is designed to carry out the SlowITe attack against an MQTT broker. The script works by establishing a large number of connections with the MQTT broker and keeping them open for as long as possible, thereby exhausting the broker's ability to handle additional connections. This is a form of denial-of-service (DoS) attack, aimed at making the MQTT broker unavailable to other legitimate users.

**The keep alive (-k) parameter:**

The -k or keep\_alive parameter in MQTT (MQ Telemetry Transport) is a time interval, expressed in seconds, that the client commits to for when it communicates with the broker. If no other messages are being exchanged, the client ensures that a message is sent within each keep\_alive interval.

The purpose of the keep\_alive parameter is to let both the client and the broker know that the other is still online and available. It's essentially a heartbeat mechanism. If a client doesn't send a message in the keep\_alive interval, the broker assumes the client is offline, disconnects it, and can initiate any clean-up processes associated with that. Similarly, if a client doesn't receive a message from the broker in the keep\_alive interval, it can assume the broker is unavailable and attempt to reconnect or take other appropriate action.

In a SlowITe attack, the goal is to exhaust all available connections on the server by opening a large number of connections and keeping them open for as long as possible. The keep\_alive parameter is used to keep these connections open. This is achieved by setting the keep\_alive parameter to a large value when connecting to the server.

The MQTT protocol specifies that the server must keep the connection open for a period of time (1.5 times the keep\_alive value) after receiving a CONNECT packet. By setting a large keep\_alive value, the client can force the server to keep the connection open for a longer period, even if no further messages are exchanged.

This can be problematic for the server, as each open connection consumes resources. If a large number of connections are opened and kept alive, the server may become unable to handle additional connections, effectively denying service to other clients. This is why it is a form of DoS attack.

In short, the keep\_alive parameter is used in the attack script to keep the connections open for a longer period, thereby exhausting the server's resources.

**python3 MQTT\_keepAlive.py -a broker\_address [-p broker\_port] [-k keep\_alive] [-h | --help] //under investigation**

The purpose of this script seems to be more diagnostic in nature. It is used to test the effects of different keep\_alive values on the MQTT protocol's functioning. The keep\_alive parameter in MQTT is a time interval during which, if no other messages are exchanged, the client ensures a message is sent to maintain the connection with the broker. By adjusting this value, you can observe how the MQTT broker responds to different keep\_alive intervals and understand its impact on the broker's performance. I haven’t used this script in the conduction of my attack. I didn’t need to.

**python3 client\_pub.py -a broker\_address [-p broker\_port] [-k keep\_alive] [-h | --help]**

Creates a publisher and establishes a connection to the broker.

**python3 client\_sub.py -a broker\_address [-p broker\_port] [-k keep\_alive] [-h | --help]**

Creates a subscriber and establishes connection to the broker.

**MQTTSA:**

[GitHub - stfbk/mqttsa: A tool to assist IoT developers in securing MQTT-based IoT deployments](https://github.com/stfbk/mqttsa)

This is used for brute force and publishing malformed data.

The difference between flooding-based Denial of Service (DoS) and slow DoS attacks lies in the methodology used to overwhelm or disrupt the target system.

**Flooding-based DoS:** In a flooding-based DoS attack, the goal is to overwhelm the target system (in this case, an MQTT broker) by flooding it with a large volume of malicious or excessive traffic. The attack floods the system with an excessive number of connection requests or data packets, exhausting its resources and making it unresponsive or unable to handle legitimate requests. Flooding-based DoS attacks often involve generating a high number of connections or sending large amounts of data to consume system resources, causing service disruption or degradation.

**Slow DoS:** In a slow DoS attack, the objective is to degrade the target system's performance gradually over time, rather than overwhelming it with a sudden flood of traffic. The attack involves exploiting vulnerabilities or inefficiencies in the system's handling of requests, such as utilizing resource-intensive operations or consuming processing power over an extended period. Slow DoS attacks aim to exhaust system resources gradually, making it difficult for legitimate users to access the service due to increased response times or system unavailability.

**MQTT Malaria:**

[GitHub - etactica/mqtt-malaria: Attacking MQTT systems with Mosquittos (scalability and load testing utilities for MQTT environments)](https://github.com/etactica/mqtt-malaria)

This tool is used for flooding DoS attack.

**My Personal Tool (script.py):**

This tool is a precursor to the GUI tool, essentially being the blueprint or the CLI version. It should ideally be able to generate all the attacks that I wish to implement. You need to mention the mode with ‘-m’ parameter to let the tool know which attack you want to perform. For SlowDoS, I have included one more parameter which is the ‘-t’ parameter by which the user can provide for how long the attack is to be generated in seconds.

The GUI application is based on this script and implements the logic similarly.