IT and Network Security
WiSe 2020/21

Practical Assignment No. 2

Nils Aschenbruck

Leonhard Brüggemann, **Alexander Tessmer**, Stefanie Thieme

Till Zimmermann

Release	10.11.2020
Submission	23.11.2020

General information for the practical assignment:

- Please read the assignment sheet **thoroughly from top to bottom**, before you start working!
- For the admission to the exam, you have to acquire at least 8 points.
- You have to work on all assignments **on your own**.
- **(Code) plagiarism will not be tolerated!**
  This includes the use of code from other sources without reference or unusual amounts of code with references and applies to all parties involved.
- Submission is **no later than 9:00 a.m.** on the respective submission date via **StudIP**. Related files must be packed as a ZIP archive and uploaded in the assignment area of the lecture. The archive name should conform to the following pattern:

  ITS_WiSe_202021_PA2_*MatrNo*

  where ***MatrNo*** has to be replaced with your matriculation number. On unpacking the archive, a **subfolder with the same naming pattern** must be created automatically. Please note that potential changes to this submission method can occur for individual assignments.
- **Comply with all stated specifications** for file names, folder structure, program parameters, etc. Otherwise the time for assignment evaluation is unnecessarily extended and the result announcement will be delayed.
- Please use the **mailing list** of the lecture for questions of general interest. Alternatively you can attend the **Q&A tutoring session**. An application by mail to the teaching assistant (**tessmer@uos.de**) is optional, but recommended for extensive questions.

Practical Assignment 2: IoT Sniffing

The Message Queue Telemetry Transport (MQTT) protocol [1] is a lightweight open-source message protocol, which was created for resource-limited devices in low-bandwidth networks. Therefore MQTT was standardized in 2013 by OASIS as the protocol for Internet of Things (IoT) [3]. MQTT is based on a publish-subscribe architecture and enables Machine-to-Machine (M2M) communication over TCP/IP in many areas of the IoT, like Industry 4.0 or Smart Homes. Devices (Clients) can send messages as a MQTT publisher to a broker (Server) and receive these messages as a subscriber. The publisher assigns a specific topic to every message, which can be subscribed to by multiple subscribers.

Security features are included since protocol version 3.1. On the one hand SSL/TLS [4] is supported and on the other hand there is user management for authorization. This enables individual users to authenticate themselves with passwords and to define read and write permissions for individual topics with Access Control Lists (ACLs) [5]. If TLS is not used, the username and the corresponding password is transmitted in plain text during the MQTT connection establishment (see [6]). Since TLS is disabled by default and requires the generation and distribution of certificates, this security option is often neglected in practice.

Scenario

Smart home devices provide an IoT solution for many features of home automation. This includes door locks, lighting, wireless speaker systems, home security monitoring, smoke detectors, thermostats and many more. A big market share of smart home devices holds the Tuya Smart framework of the Alibaba Group, China. These devices use MQTT and are easily hacked [9]. The MQTT communication is encrypted with TLS, but the key is transmitted as plain text. Therefore, we consider the MQTT communication for simplification as unencrypted plain text in this assignment. We examine the application of a smart thermostat that doubles as a fire alarm system. A typical fire sprinkler system triggers at 68 °C. Our thermostat regularly publishes the current temperature on the topic `/smart/supervisor/emergency/thermostat/current_temperature`. A fire alarm subscriber checks the measured temperature and reports a warning message, if the temperature is above 68 °C. The program was compiled into a single file (supervisor.pyc) to simplify the operation. To provide reliability, the program developers included a feature that restarts the broker within 10 seconds on the next highest port, if a port is blocked. But they neglected to secure the broker against multiple simultaneous connections from MQTT clients and it is therefore susceptible to Denial-of-Service (DoS) attacks.

Test Environment

Please first install the needed packages for this assignment with the following commands in your VM as root user:

```
$ sudo su
$ apt install mosquitto
$ pip3 install paho-mqtt
```

Secondly, deactivate the broker, which was automatically configured by the setup assistant:

```
$ systemctl disable mosquitto
$ systemctl stop mosquitto
```

Then unpack the archive *PA_02_files.zip* from the website and execute the included python script:

```
$ python3 supervisor.pyc
```

This starts the broker and a corresponding publisher and subscriber. The shell shows a continuous output of randomized temperature values. If a threshold of 68 °C is exceeded, a warning will be displayed. Additionally the current number of clients is shown.

Assignment

Assume that you are part of a team for an IT security audit of the previously described software. Your goal is to show how security gaps can be exploited in the described scenario by using a Man-In-The-Middle (MITM) attack to transmit false temperature data to the control unit.

The preparatory work was already done for you. The vulnerabilities were detected and a corresponding framework (framework.pyc) has been created in Python 3.8. Two core parts must be implemented by you:

1. Because TLS was not enabled by the developers, you should start with sniffing the unprotected credentials from the connection establishment of the MQTT protocol. The format of the `CONNECT` message that is used for connection establishment can be looked up at [6]. You can assume for simplicity that the client has not set the *Will Flag* and the *Clean Session* options. But you should take into account that parts of the message can have different lengths (e.g. username, password and client id).

   The preparatory work for this task has also already been done for you. Therefore you can start the work directly on the received TCP packets. Implement the `parse_tcp_packet` function of the `MQTT_Dissector` class (`mqtt_dissector.py`)

using the Python inherent `Struct` class ([7]). You can test your implementation with the following command (while the `supervisor` is running):

```
$ python3 sniffer_raw_socket.pyc
```

On success, this command displays – if correctly implemented – the intercepted access information.

2. In the next step you will use the read credentials to execute a DoS attack. Implement the `MQTT_DoS` class (`dos.py`) and use the `mqtt-paho` library [8] to continually establish new client connections until the broker crashes. Ensure that the existing `run` function of this class only returns if this is the case.

If both parts of the assignment are implemented correctly, you can execute the existing framework with the following command:

```
$ python3 framework.pyc
```

This framework uses your implementations to execute a MITM attack: After the start, you should be able to notice the broker crashing and the initialization of the MITM broker. After the `supervisor` has restarted the broker within 10 seconds, you should be able to see that the messages with a value above the threshold will be changed and no warnings will be issued.

Functional Testing

The `supervisor` starts by default with fixed parameters for username, password and client id of the publisher and a fixed number of clients needed to cause a program crash. Validate that your program also works with varying values for these parameters. You can start the supervisor from the shell with defined command parameters to use randomly generated parameters. The following command shows an overview of the existing command parameters:

```
$ python3 supervisor.pyc --help
```

The other commands you have to use also include this help to simplify testing, if there are defined parameters.

Happy Coding!

Submission:

- ZIP-archive named *ITS_WiSe_202021_PA2_MatrNo.zip* that automatically creates the corresponding folder *ITS_WiSe_202021_PA2_MatrNo* on unpacking (cf. general information on page 1).
- This folder must contain all relevant source files (.py); at least:
  - the parser for the credentials: *mqtt_dissector.py*
  - the program that executes the DoS attack: *dos.py*
- upload the zip-archive to **StudIP**:
  IT- und Netzwerksicherheit → *Tasks* → *PA2*

Sources

[1] https://mqtt.org

[2] https://mosquitto.org/

[3] https://en.wikipedia.org/wiki/Internet_of_things

[4] https://en.wikipedia.org/wiki/Transport_Layer_Security

[5] https://en.wikipedia.org/wiki/Access_control_list

[6] https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718028

[7] https://docs.python.org/3/library/struct.html

[8] https://www.eclipse.org/paho/clients/python/docs/

[9] https://www.heise.de/ct/artikel/Tuya-Convert-Escaping-the-IoT-Cloud-no-solder-needed-4284830.html