Lab 2: Capture and Analyze IoT Device Traffic

Lab Scenario

As a professional ethical hacker or pen tester, you must have sound knowledge to capture and analyze the traffic between IoT devices. Using various tools and techniques, you can capture the valuable data flowing between the IoT devices, analyze it to obtain information on the communication protocol used by the IoT devices, and acquire sensitive information such as credentials, device identification numbers, etc.

Lab Objectives

- Capture and analyze IoT traffic using Wireshark

Overview of IoT and OT Traffic

Many IoT devices such as security cameras host websites for controlling or configuring cameras from remote locations. These websites mostly implement the insecure HTTP protocol instead of the secure HTTPS protocol and are, hence, vulnerable to various attacks. If the cameras use the default factory credentials, an attacker can easily intercept all the traffic flowing between the camera and web applications and further gain access to the camera itself. Attackers can use tools such as Wireshark to intercept such traffic and decrypt the Wi-Fi keys of the target network.

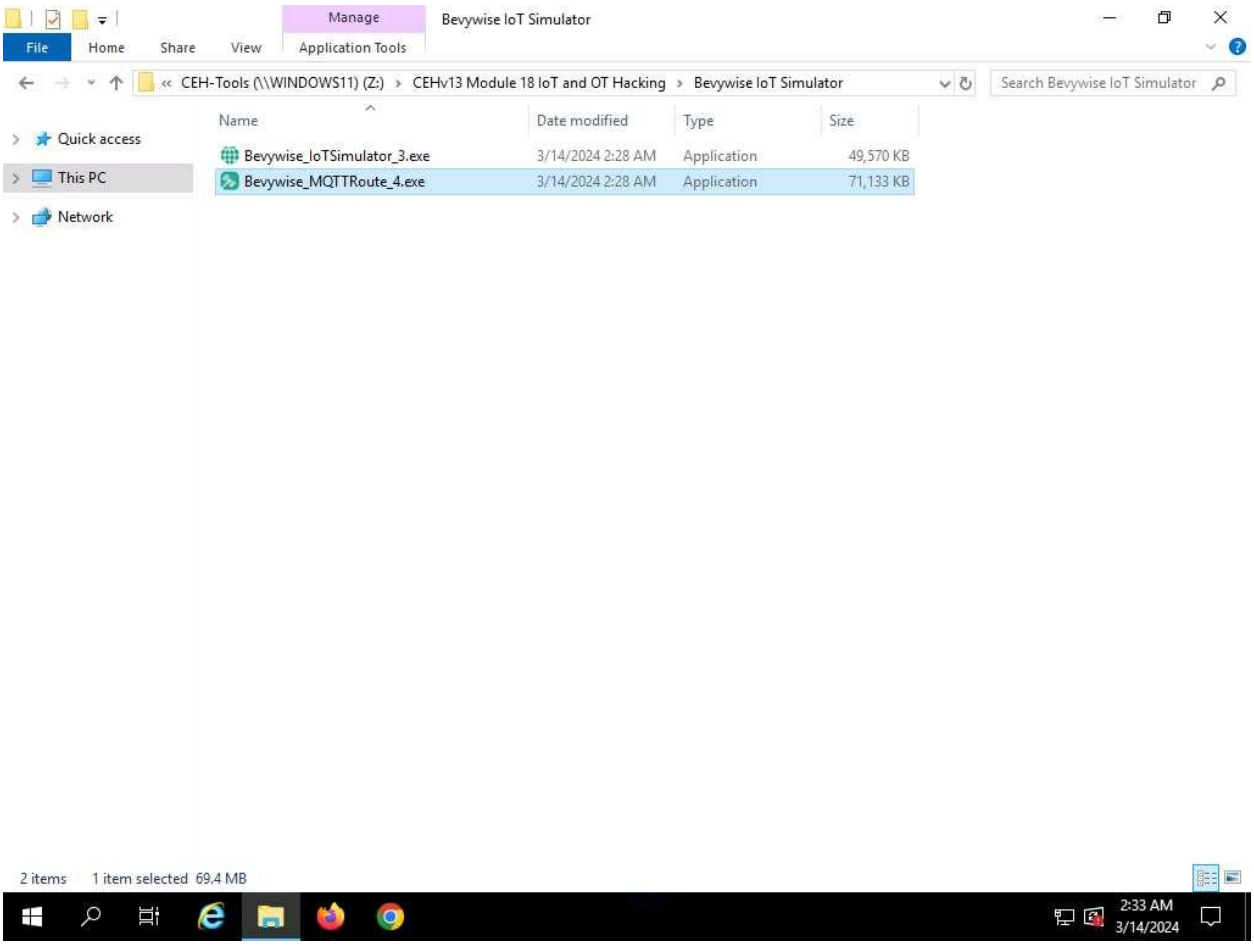
Task 1: Capture and Analyze IoT Traffic using Wireshark

Wireshark is a free and open-source packet analyzer. It facilitates network troubleshooting, analysis, software and communications protocol development, and education. It is used to identify the target OS and sniff/capture the response generated from the target machine to the machine from which a request originates.

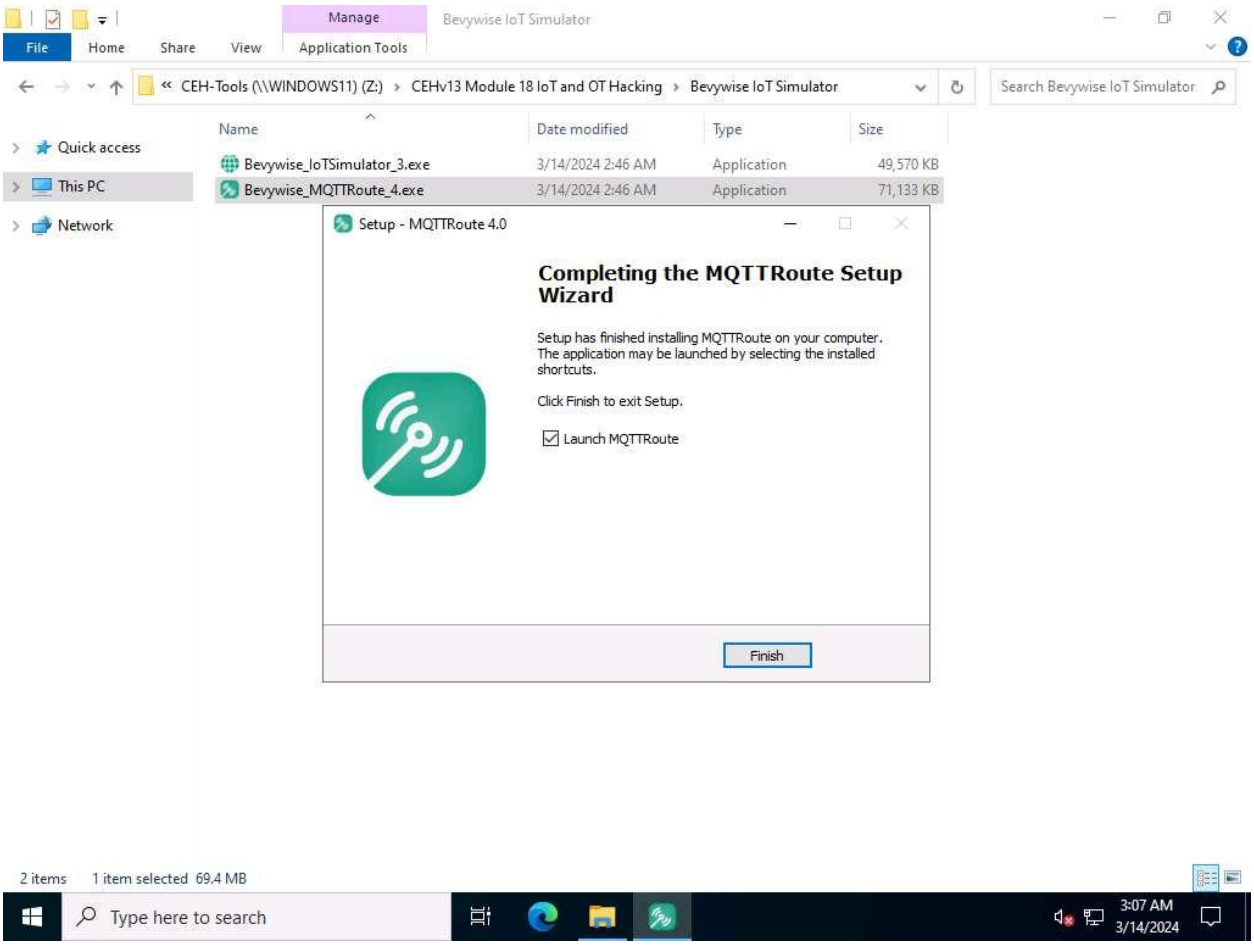
MQTT is a lightweight messaging protocol that uses a publish/subscribe communication pattern. Since the protocol is meant for devices with a low-bandwidth, it is considered ideal for machine-to-machine (M2M) communication or IoT applications. We can create virtual IoT devices over the virtual network using the Bevywise IoT simulator on the client side and communicate these devices to the server using the MQTT Broker web interface. This interface collects data and displays the status and messages of connected devices over the network.

Here, we use Wireshark to capture and analyze traffic between IoT devices.

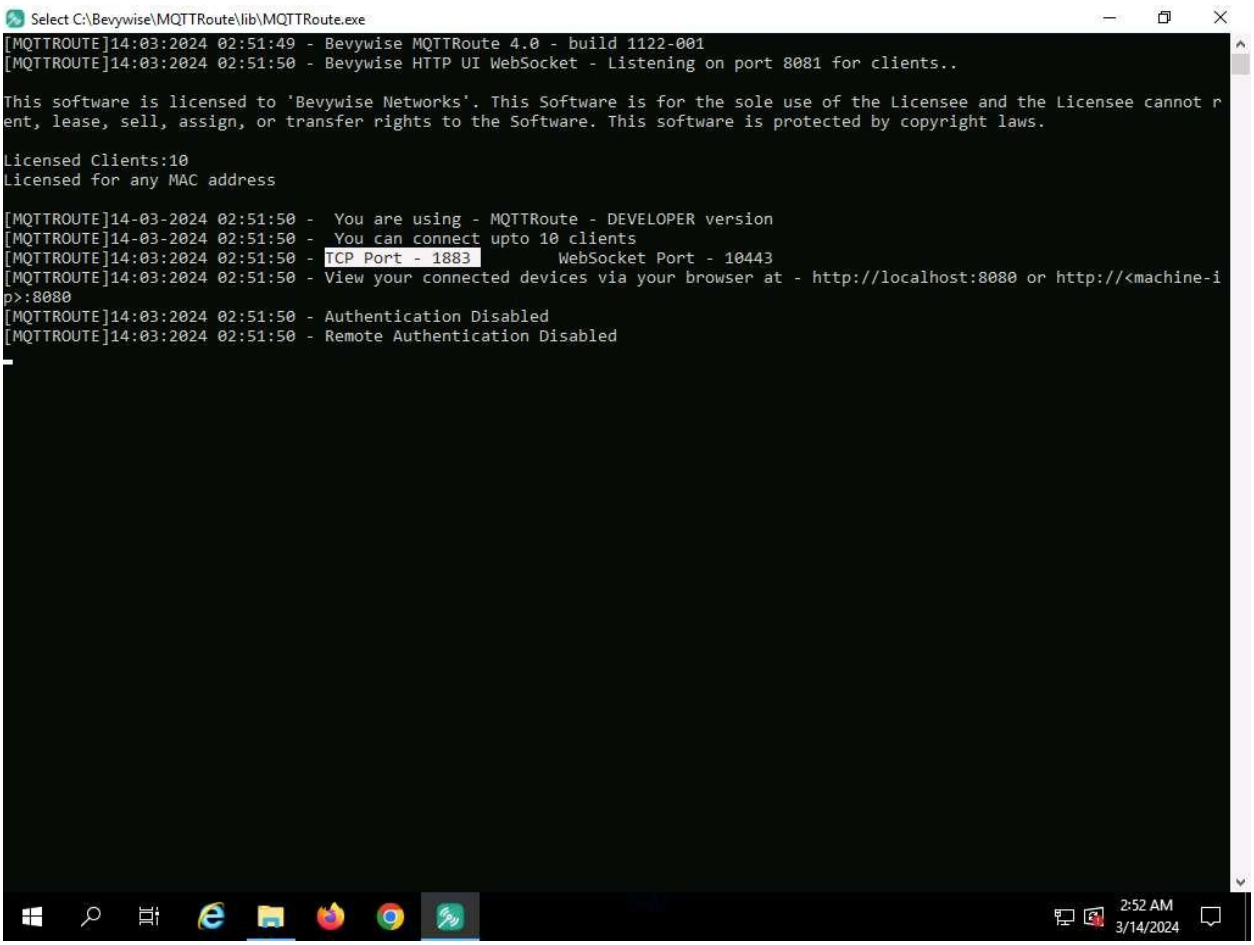
1. To install the **MQTT Broker** on the **Windows Server 2019**, click Windows Server 2019 to launch **Windows Server 2019** machine. Click Ctrl+Alt+Delete and login with **Administrator/Pa$$w0rd**.
2. Navigate to **Z:\CEHv13 Module 18 IoT and OT Hacking\Bevywise IoT Simulator** folder and double-click on the **Bevywise_MQTTRoute_4.exe** file.



3. If **Open File - Security Warning** popup appears, click **Run**.
4. The **Setup - MQTTRoute 4.0** window opens. Select **I accept the agreement** and click on **Next**. Follow the wizard driven steps to install the tool.
5. After the installation completes, click on **Finish**. Ensure that **Launch MQTTRoute** is checked.



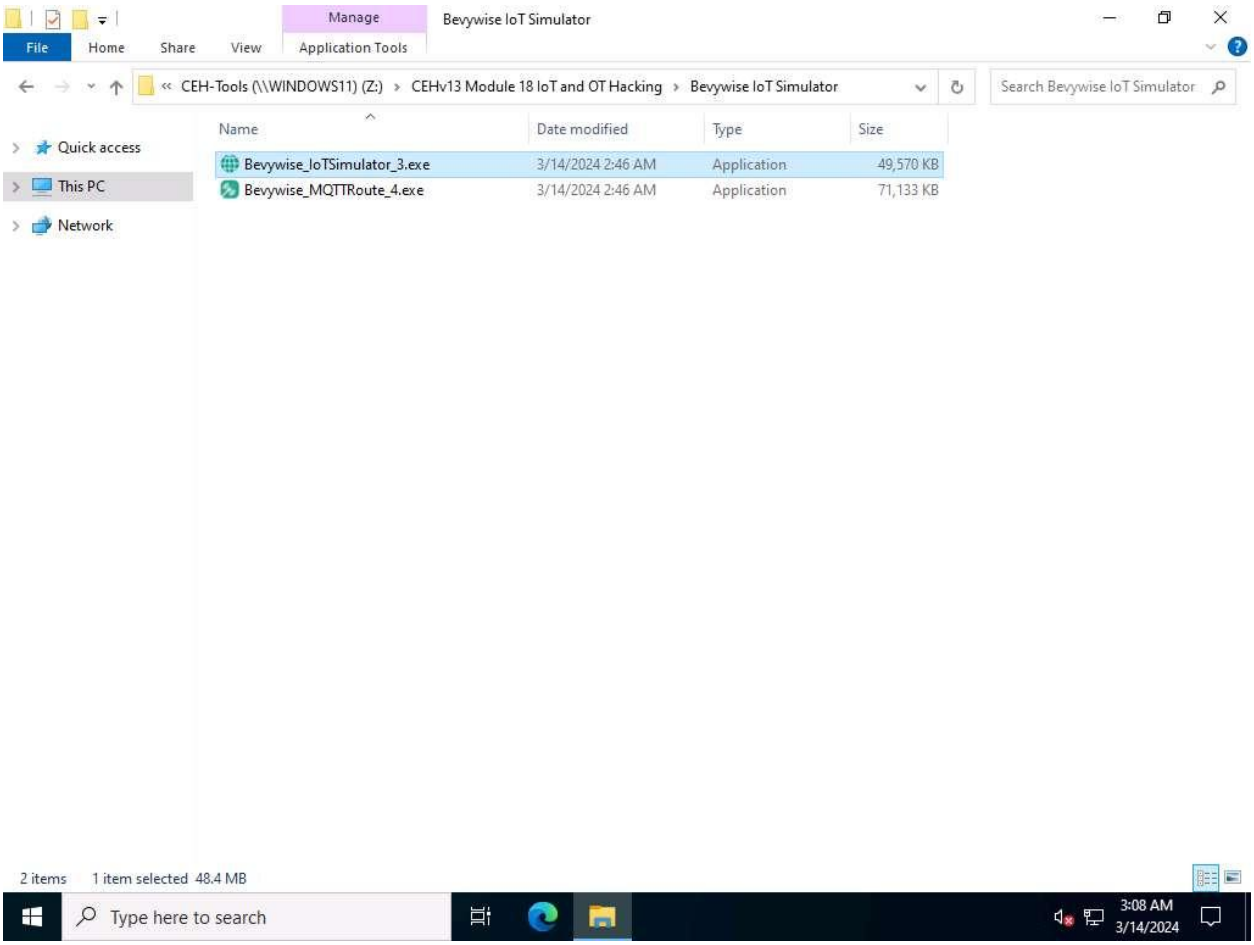
6. The MQTTRoute will execute and the command prompt will appear. You can see the **TCP** port using **1883**.



7. We have installed MQTT Broker successfully and leave the Bevywise MQTT **running**.
8. To create IoT devices, we must install the **IoT simulator** on the client machine.
9. Click Windows Server 2022 to switch to **Windows Server 2022** machine.
   Click Ctrl+Alt+Delete and login with **Administrator/Pa$$w0rd**.

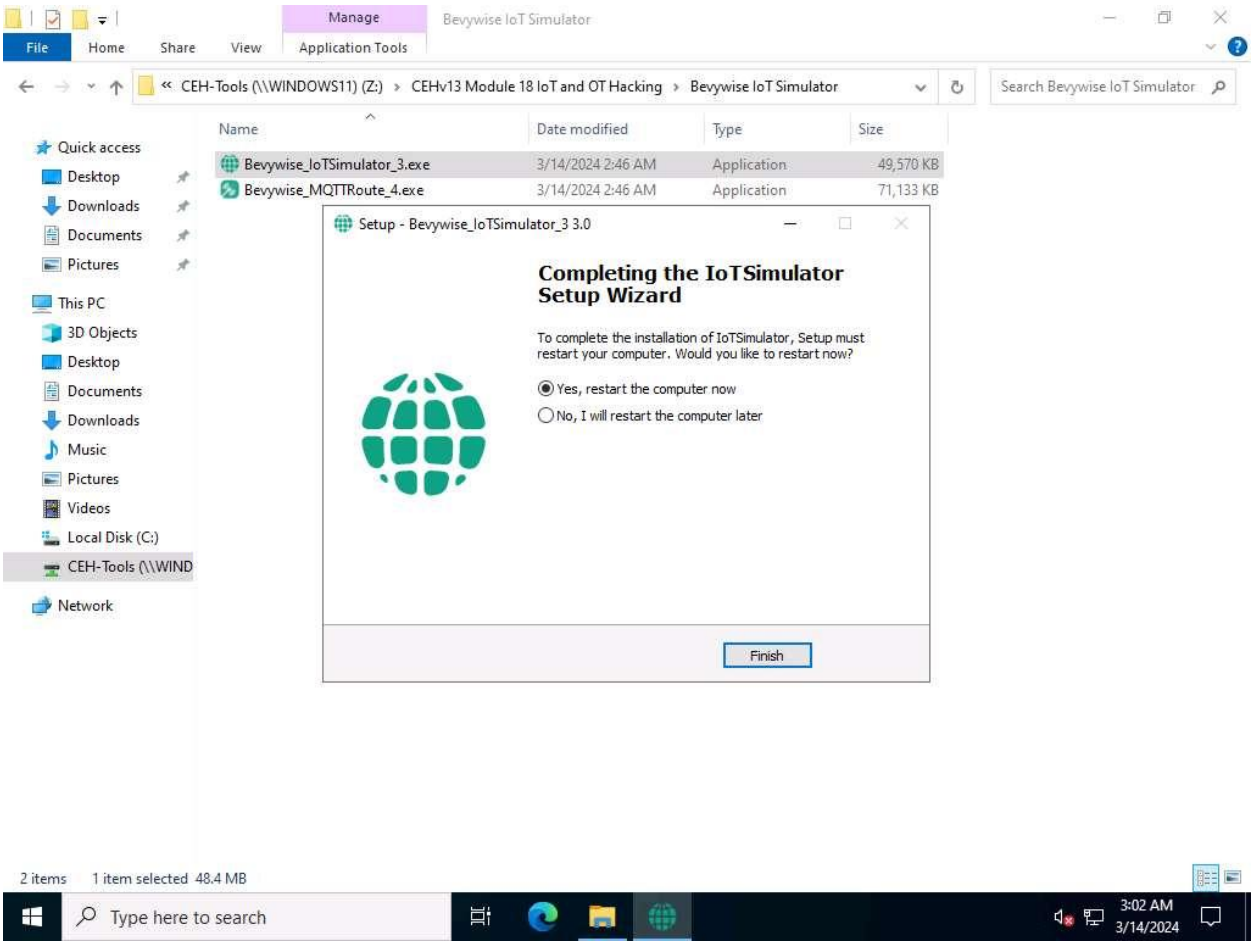
If the network screen appears, click **Yes**.

10. Navigate to **Z:\CEHv13 Module 18 IoT and OT Hacking\Bevywise IoT Simulator** folder and double-click on the **Bevywise_IoTSimulator_3.exe** file.

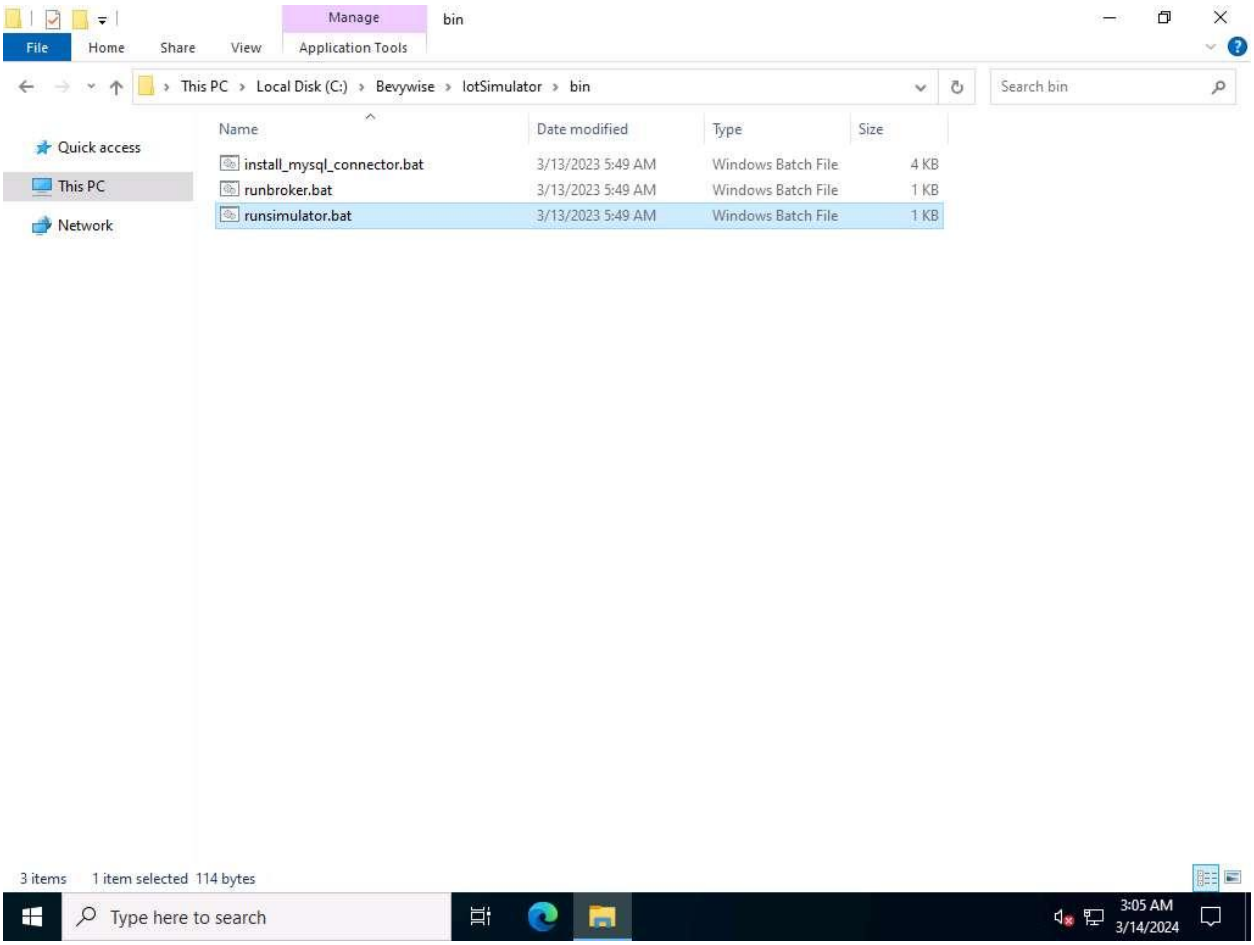


11. If **Open File - Security Warning** popup appears, click **Run**..
12. The **Setup-IoTSimulator_3 3.0** setup wizard opens. Select **I accept the agreement** and follow the wizard driven steps.
13. To complete the installation, select **Yes, restart the computer now** and click on **Finish** to complete the installation.

If restart computer option does not appear, then continue from **Step#16**.



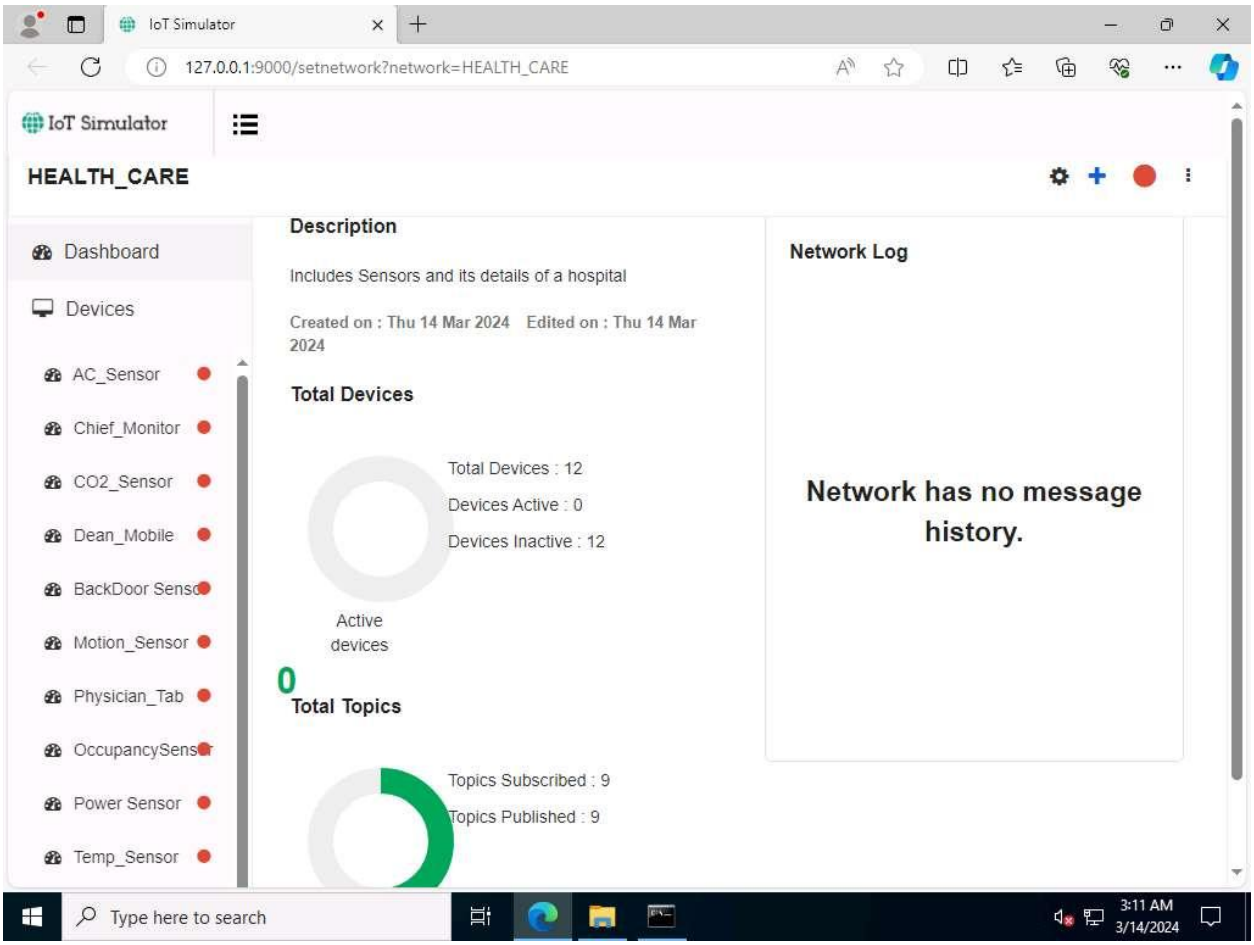
14. After restarting, Bevywise IoT Simulator is installed successfully. To launch the **IoT simulator**, navigate to the **C:\Bevywise\IotSimulator\bin** directory and double-click on the **runsimulator.bat** file.



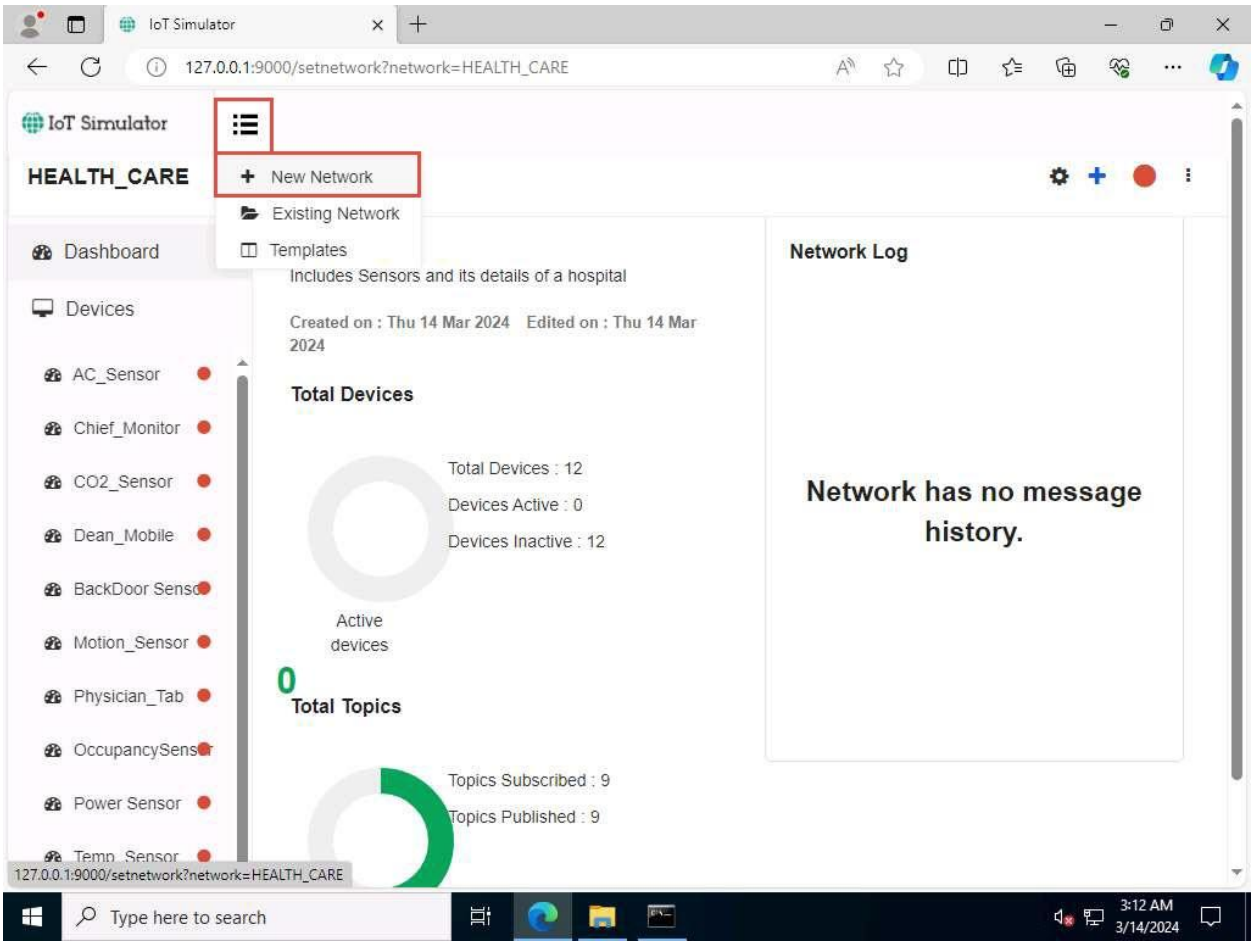
15. Upon double-clicking the **runsimulator.bat** file opens in the command prompt. If **How do you want to open this?** pop-up appears, select **Microsoft Edge** browser and click **OK** to open the URL **http://127.0.0.1:9000/setnetwork?network=HEALTH_CARE**.

If the URL directly opens in Microsoft Edge browser, then continue.

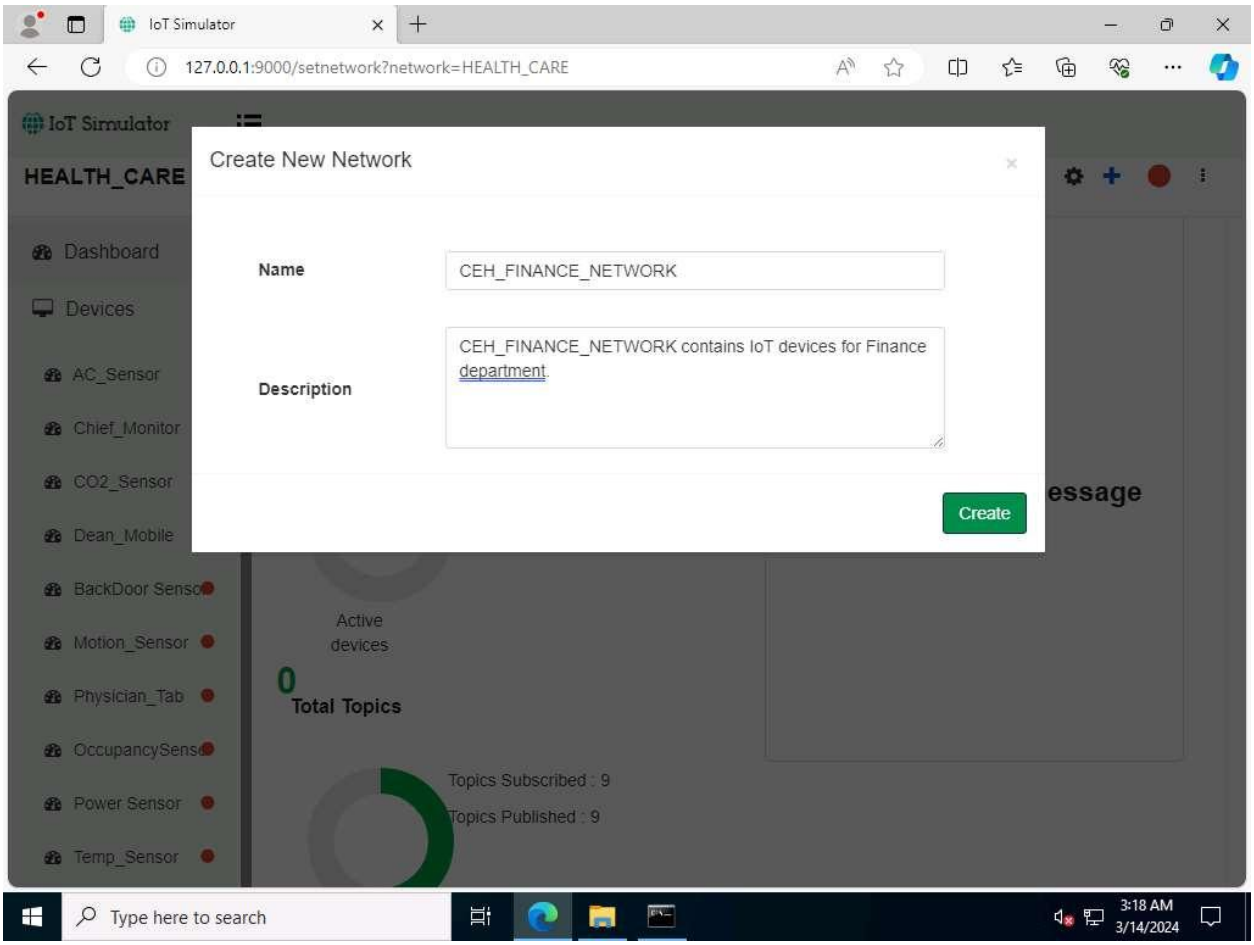
16. The web interface of the IoT Simulator opens in Edge browser. In the IoT Simulator, you can view the default network named **HEALTH_CARE** and several devices.



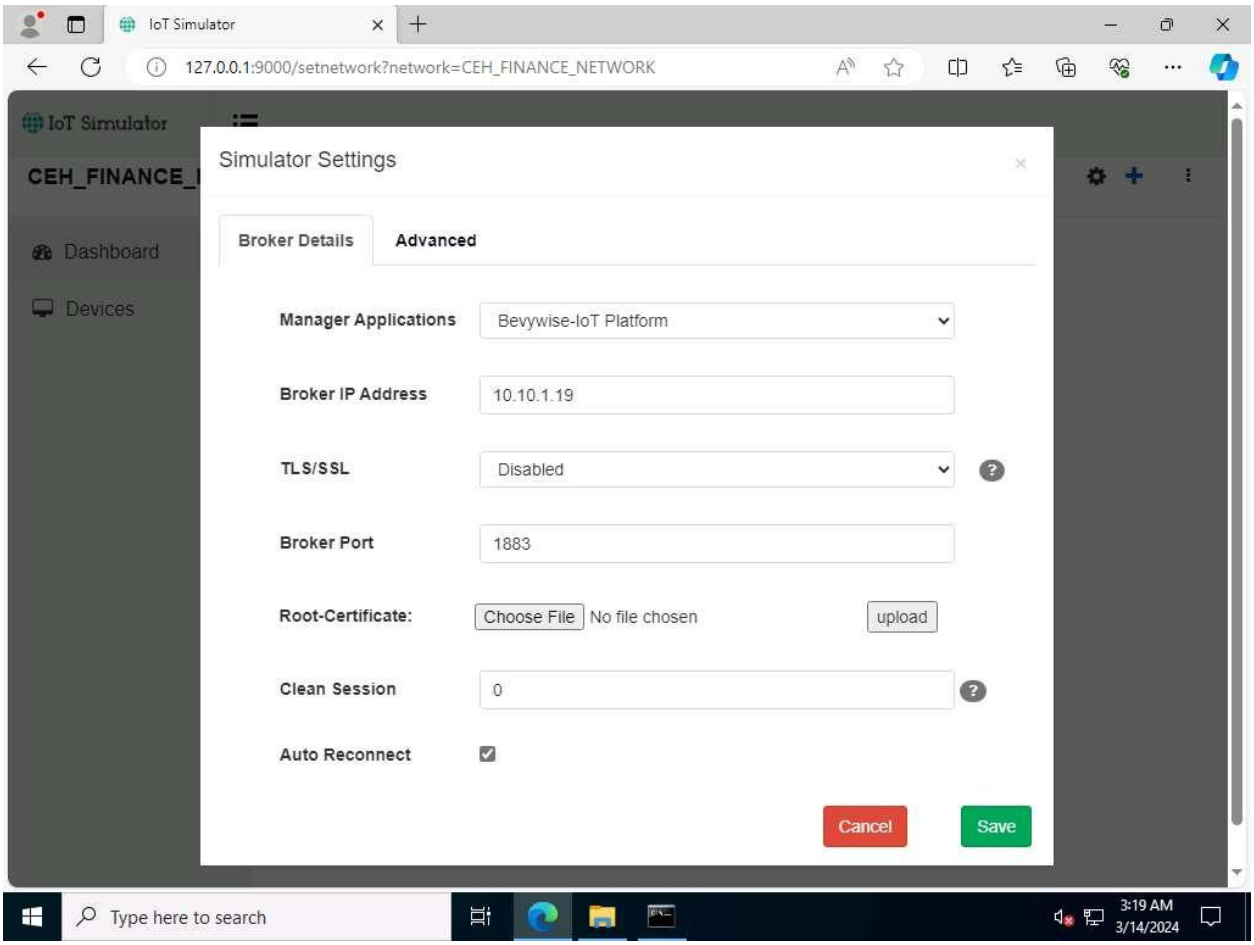
17. Next, we will create a **virtual IoT network** and **virtual IoT devices**. Click on the **menu** icon and select the **+New Network** option.



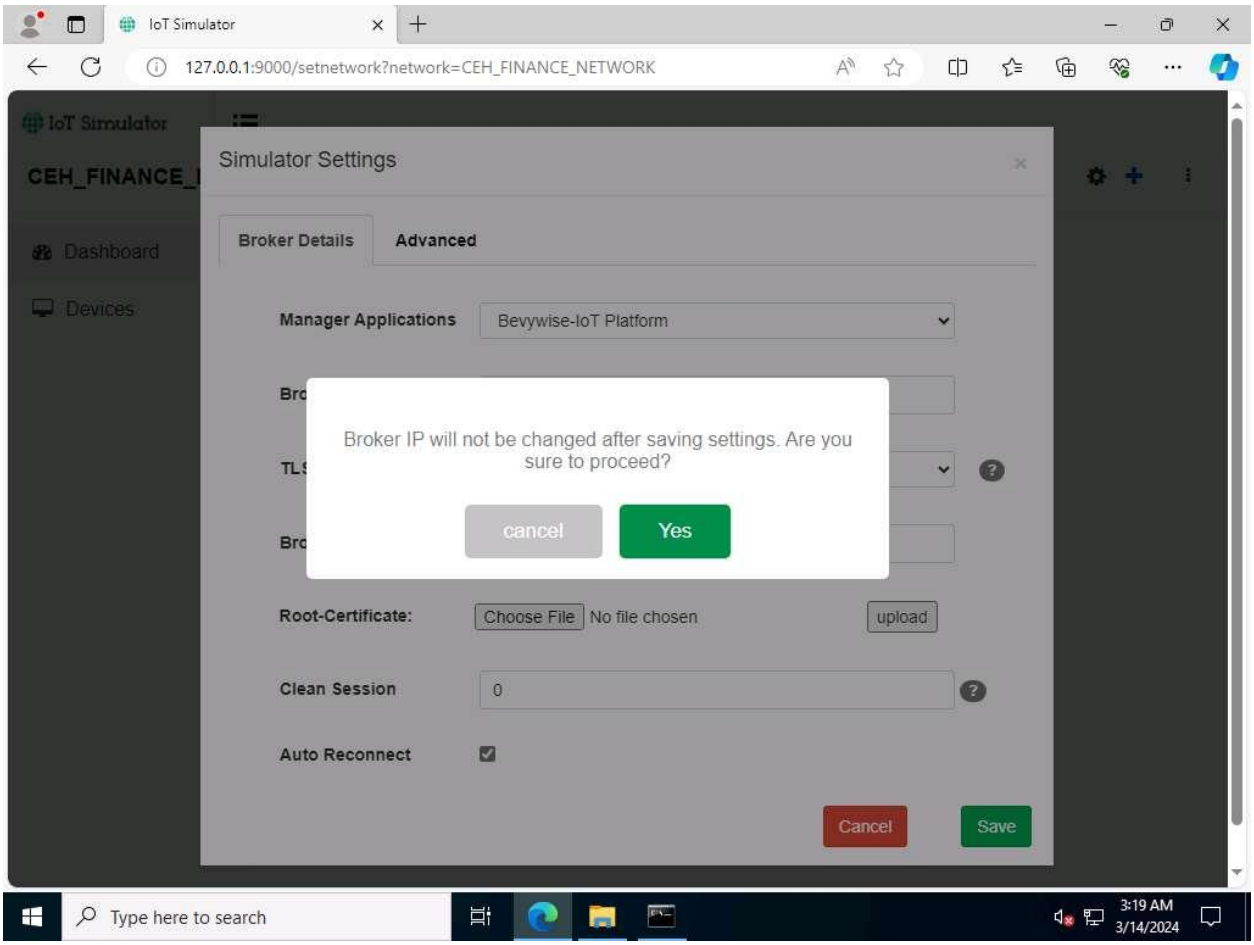
18. The **Create New Network** popup appears. Type any name (here, **CEH_FINANCE_NETWORK**) and description. Click on **Create**.



19. In the next screen, we will setup the **Simulator Settings**. Set the **Broker IP Address** as **10.10.1.19** (the IP address of the **Windows Server 2019**). Since we have installed the Broker on the web server, the created network will interact with the server using MQTT Broker. Do not change default settings and click on **Save**.

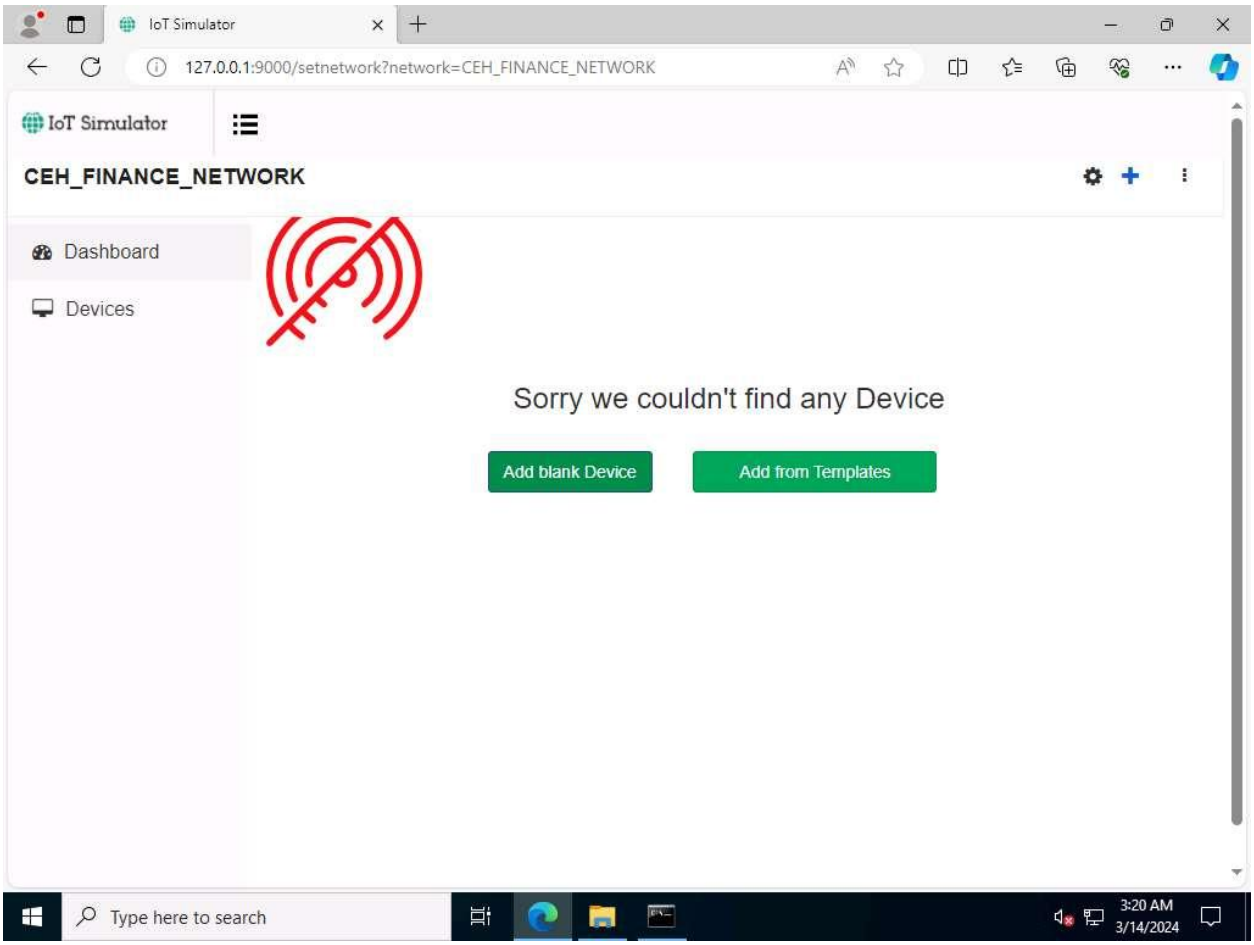


20. To proceed with the network creation, click on **Yes**.

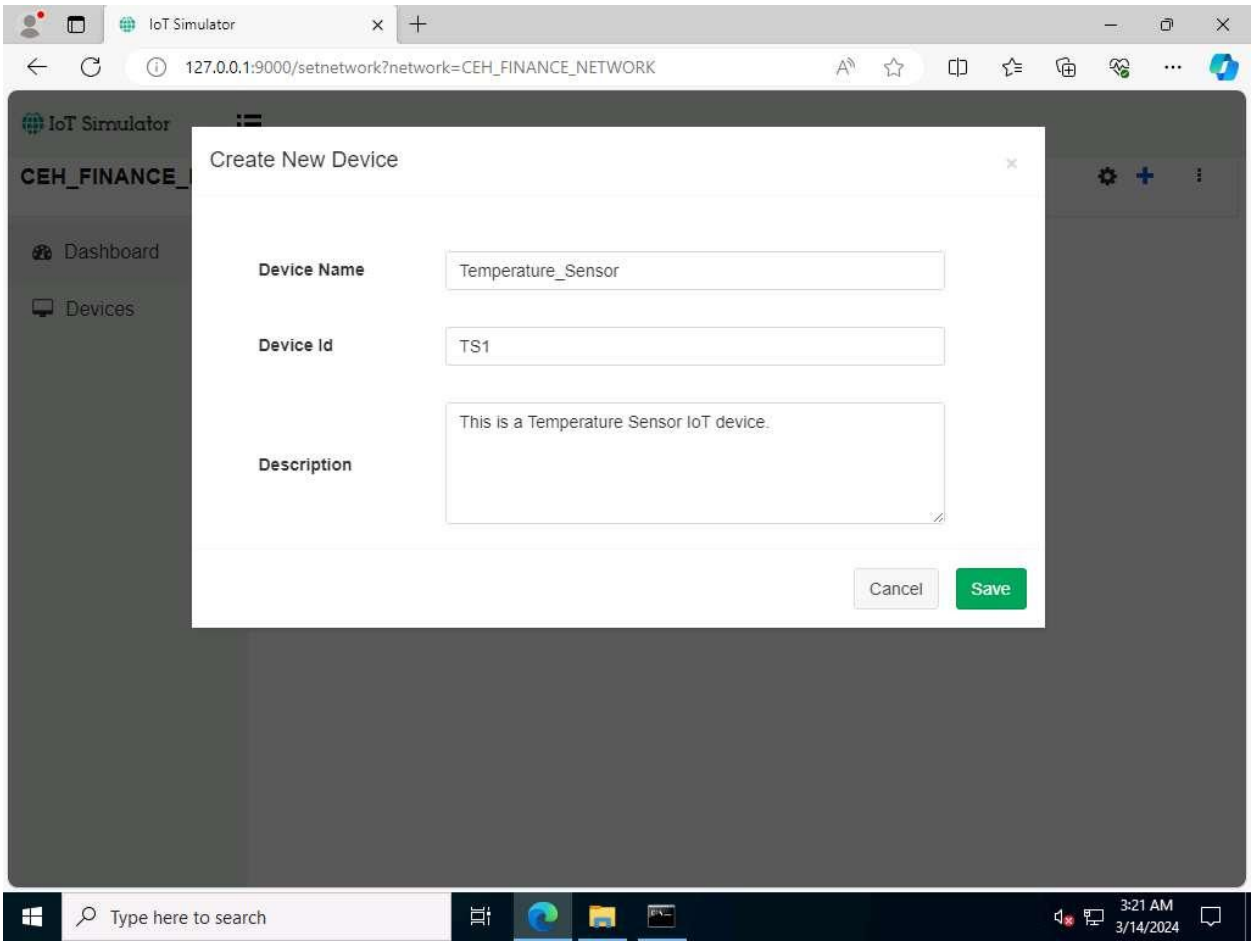


If **Configuration Saved** pop-up appears. Click on **OK** to continue. This step completes the creation of the virtual IoT network.

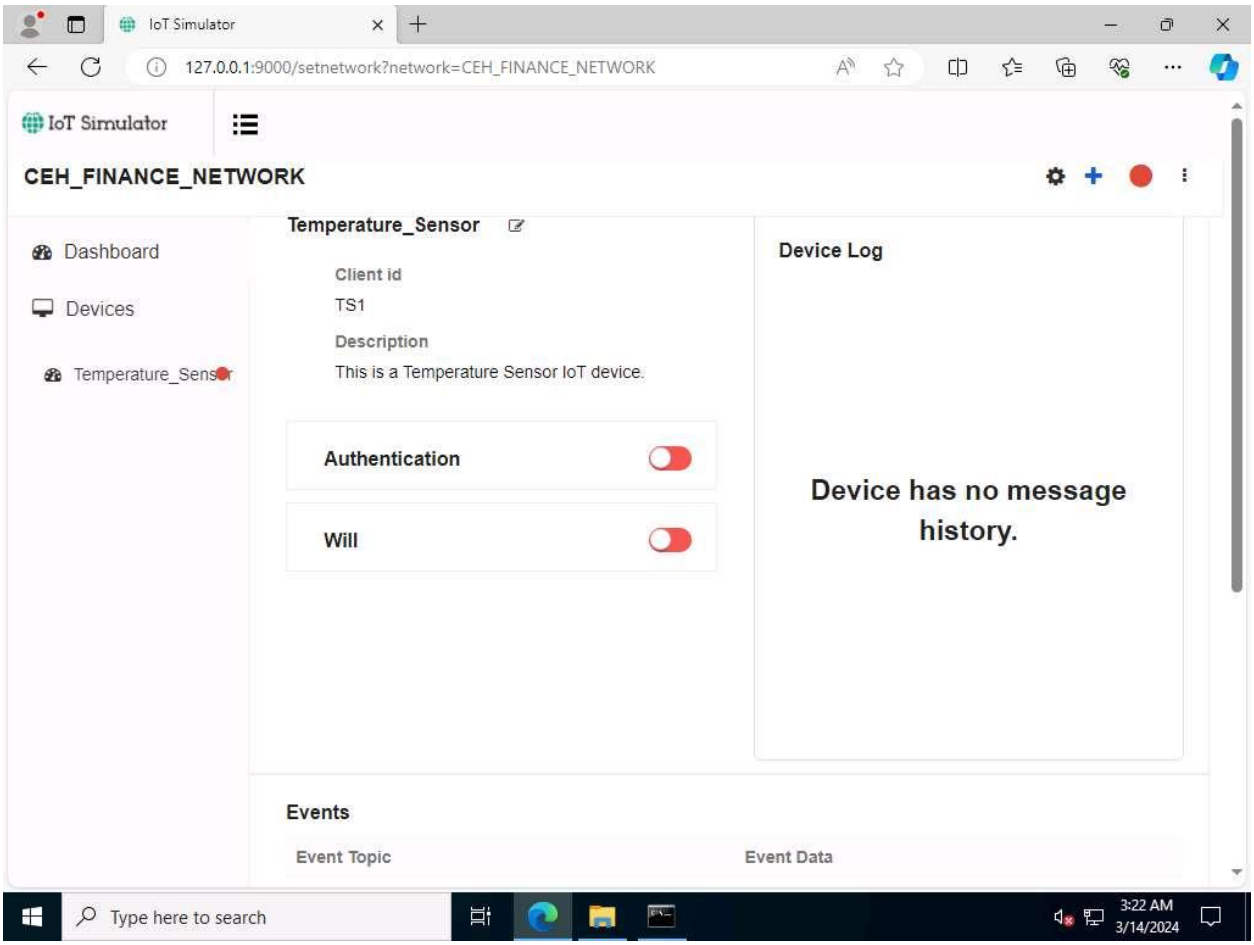
21. To add IoT devices to the created network, click on the **Add blank Device** button.



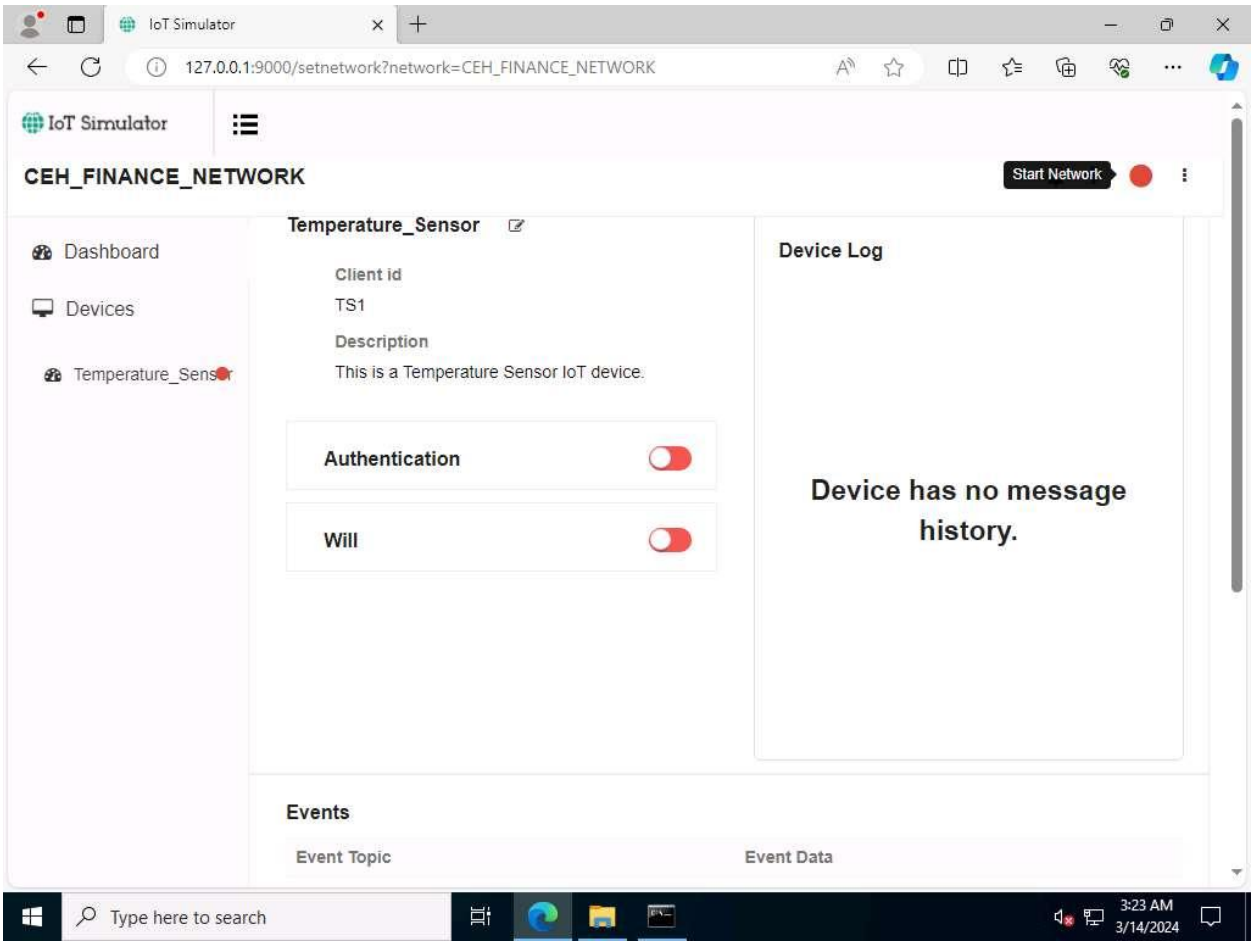
22. The **Create New Device** popup opens. Type the device name (here, we use **Temperature_Sensor**), enter Device Id (here, we use **TS1**), provide a **Description** and click on **Save**.



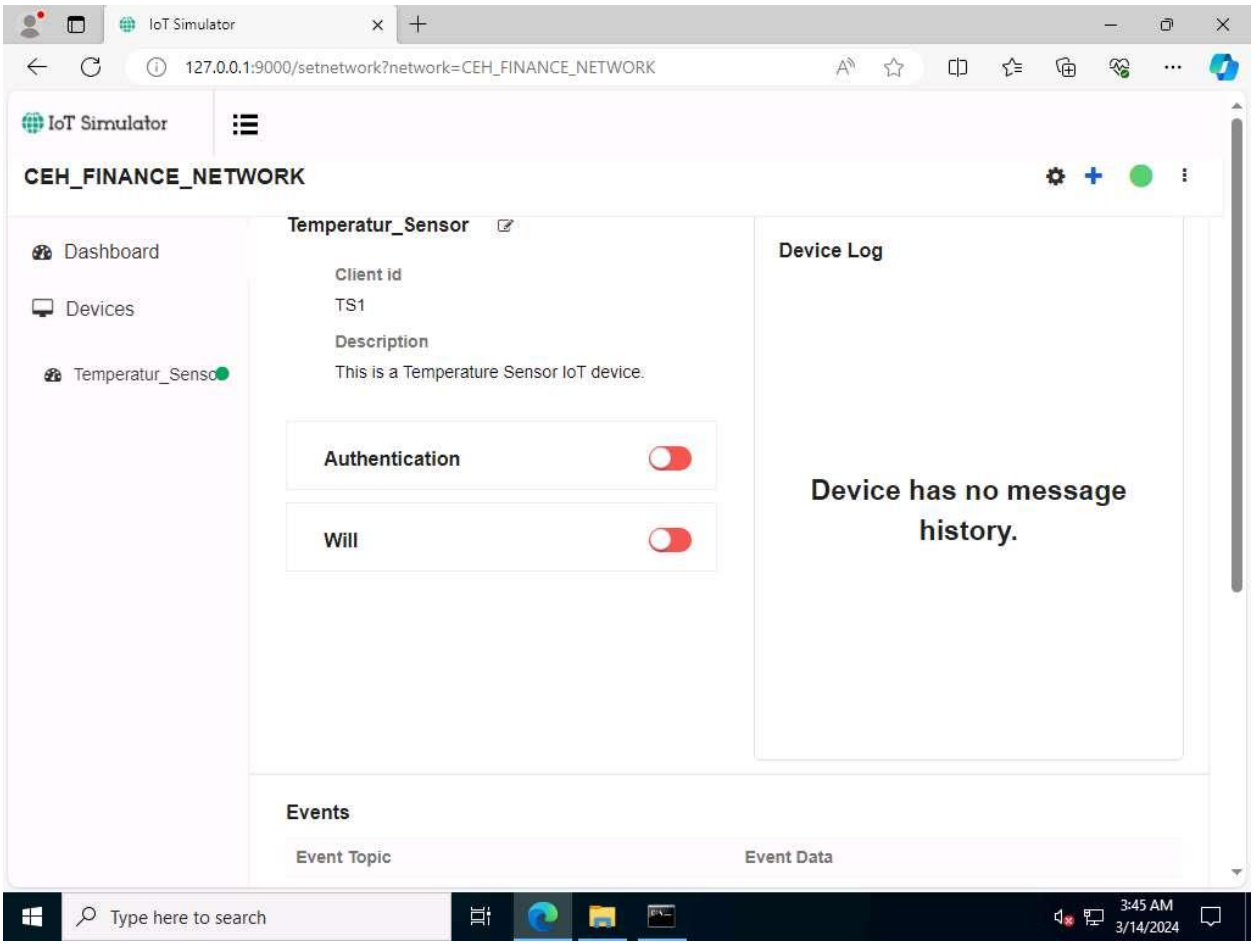
23. The device will be added to the **CEH_FINANCE_NETWORK**.



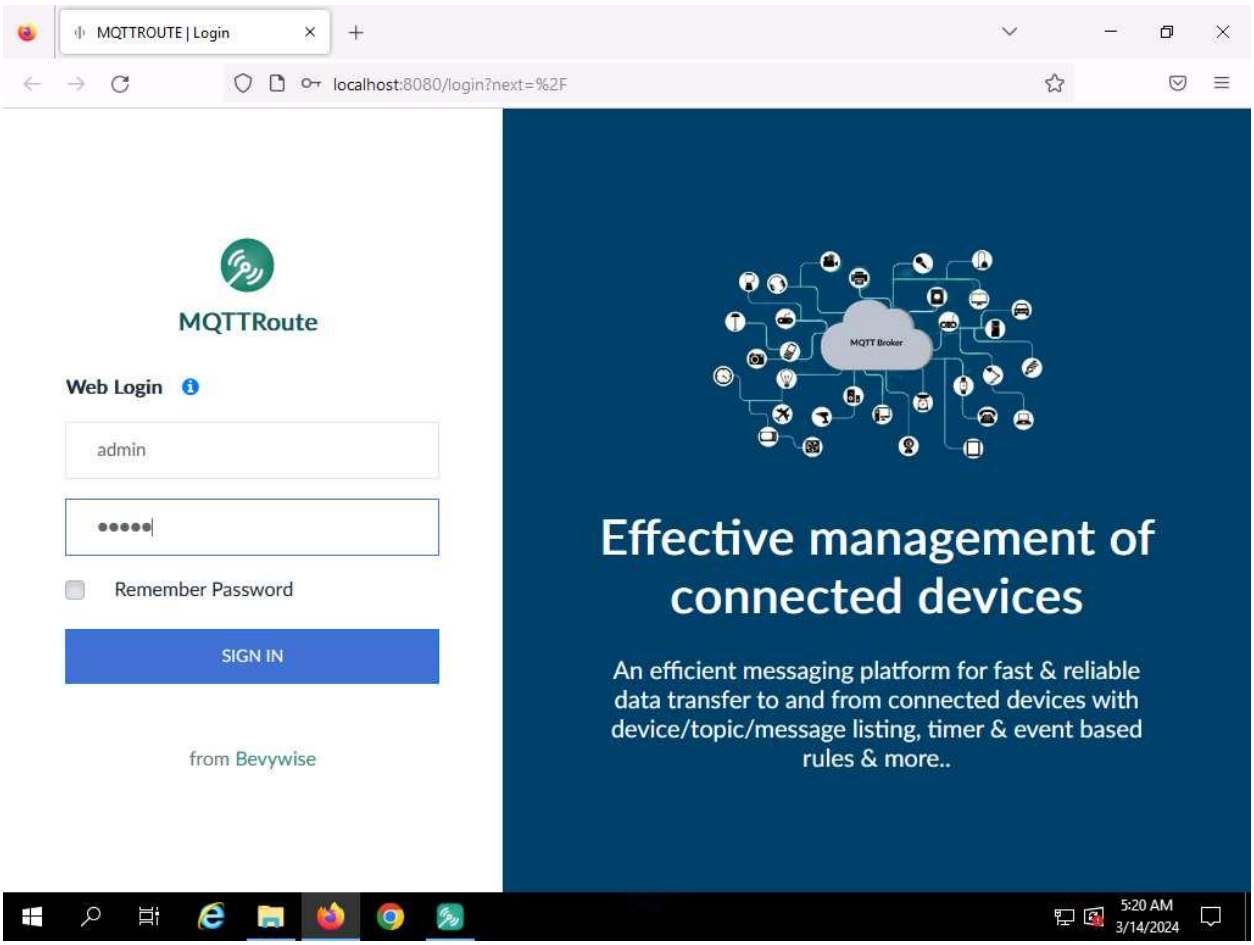
24. To connect the Network and the added devices to the server or Broker, click on the **Start Network** red color circular icon in right corner.



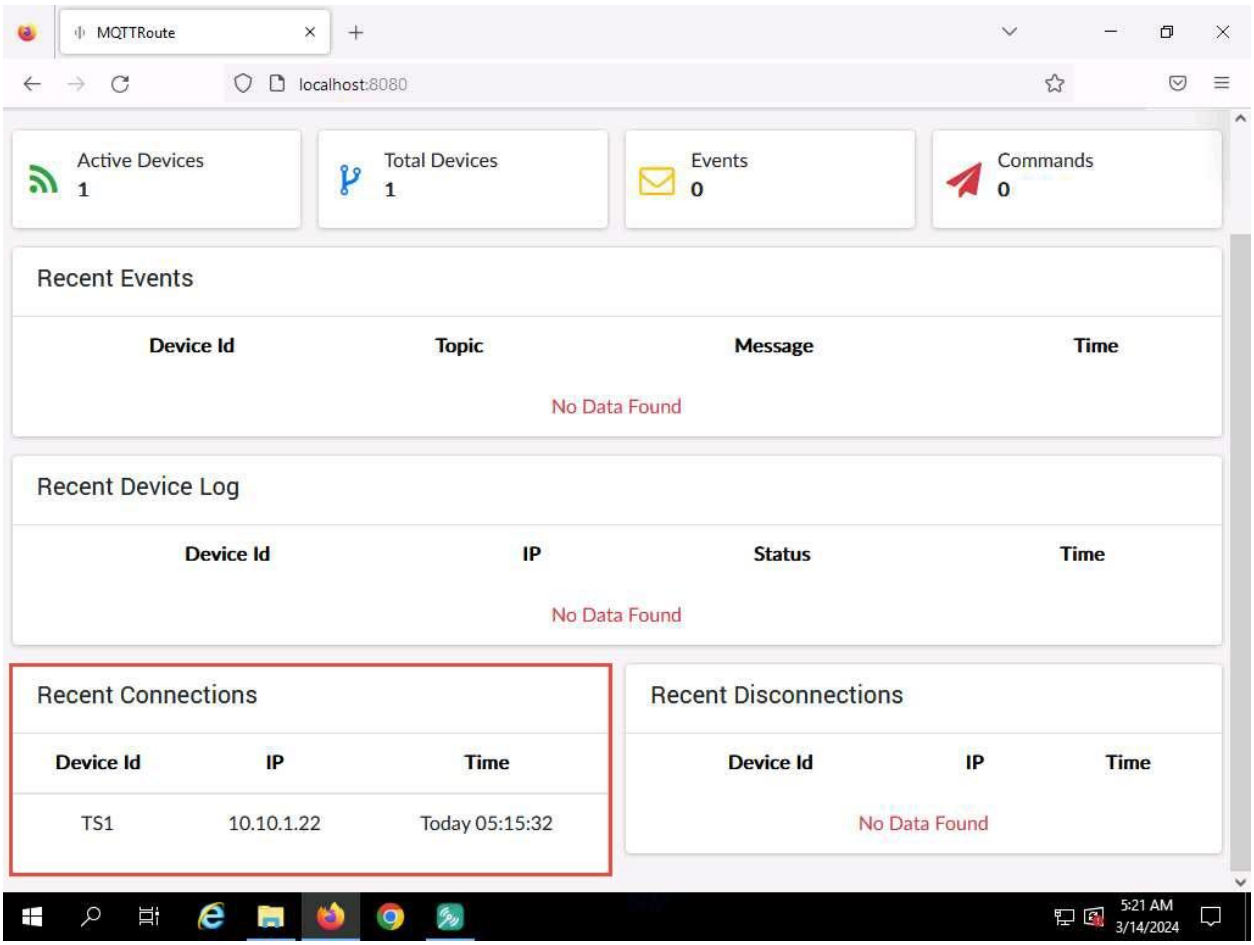
25. When a connection is established between the network and the added devices and the web server or the MQTT Broker, the red button turns into **green**.



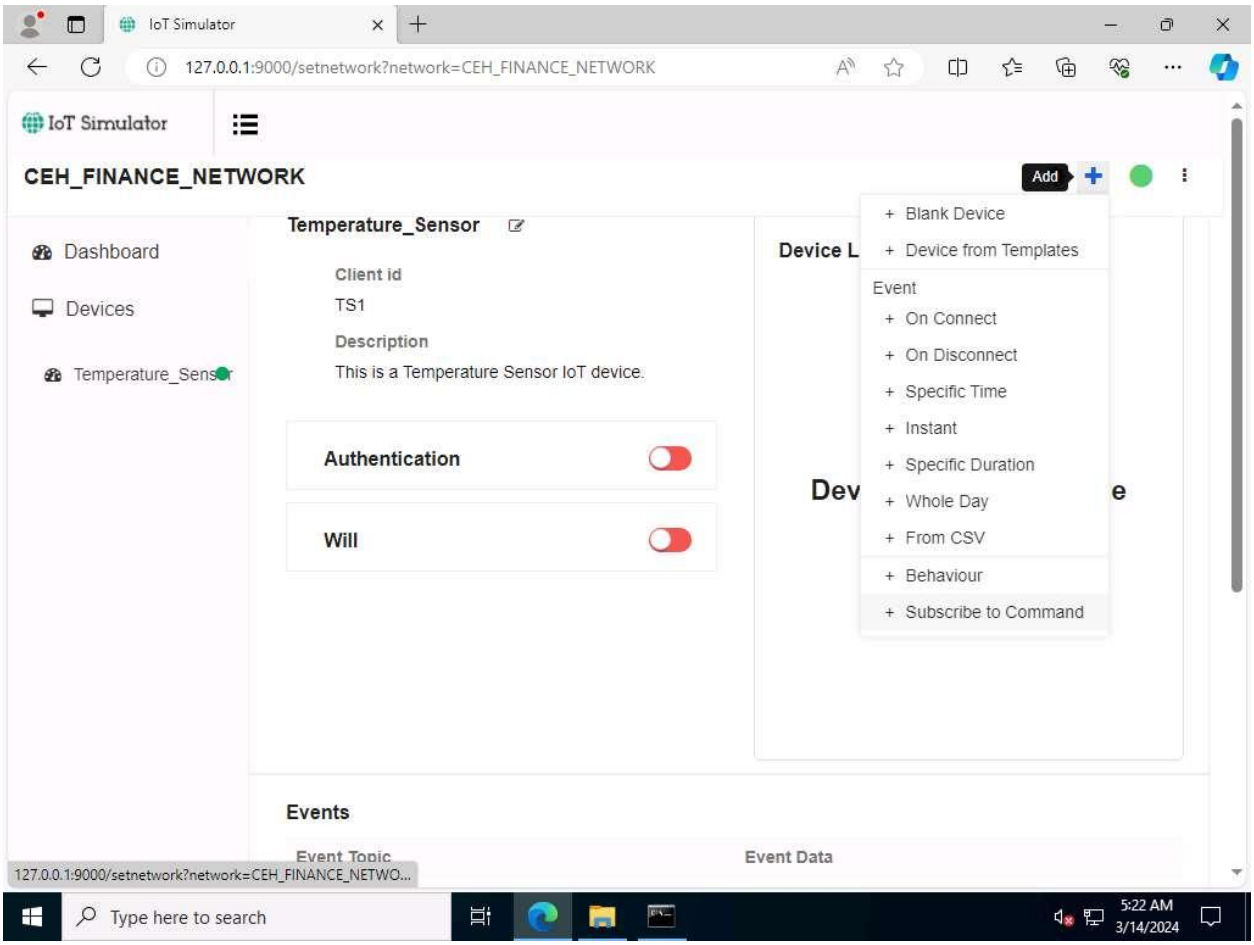
26. Next, switch to the Windows Server 2019 machine. Open a web browser, and go to **http://localhost:8080** and login using **admin/admin** (here, we are using **Firefox** Browser).



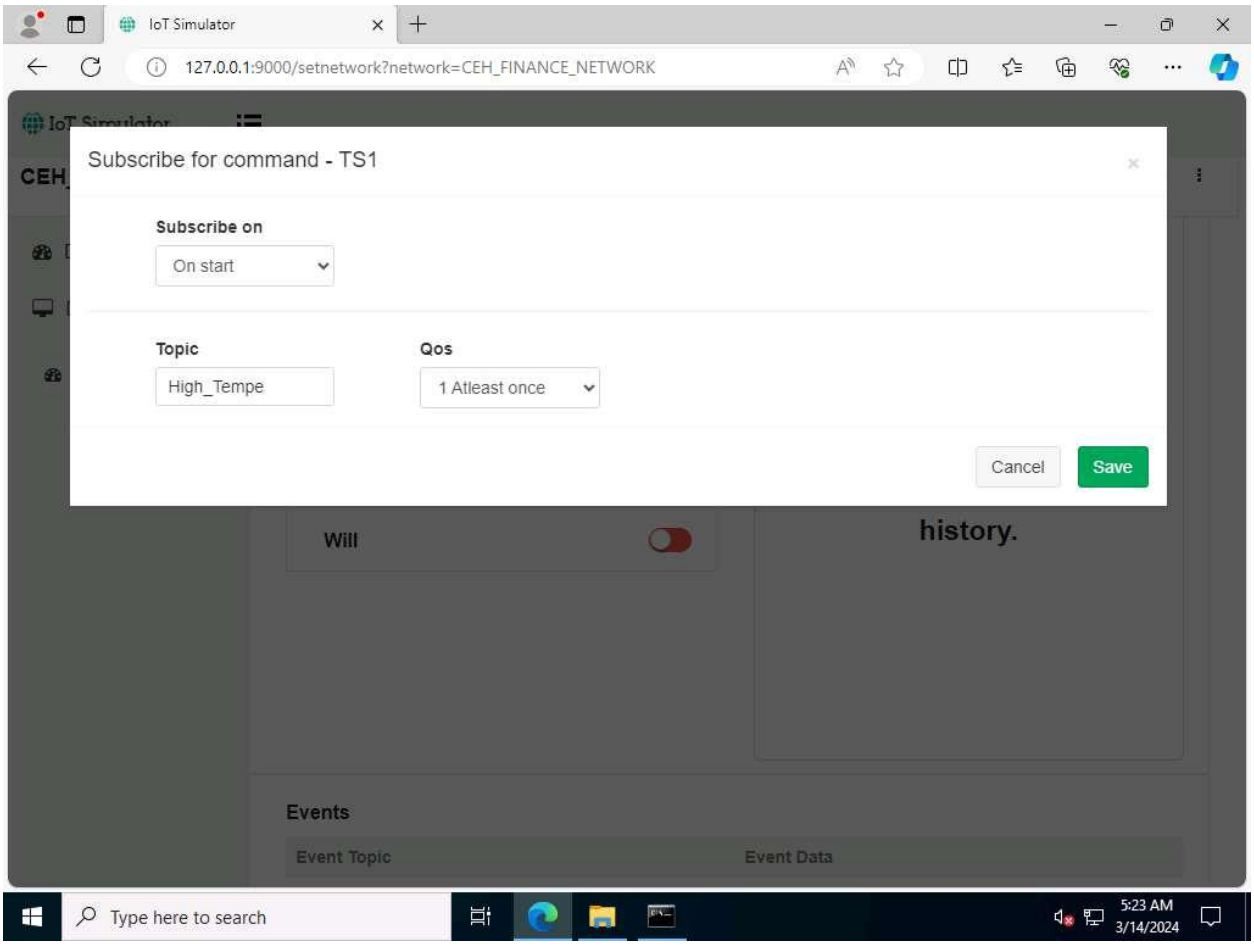
27. Since the Broker was **left running**, you can see a connection request from machine **10.10.1.22** for the device **TS1** under **Recent Connections** section.



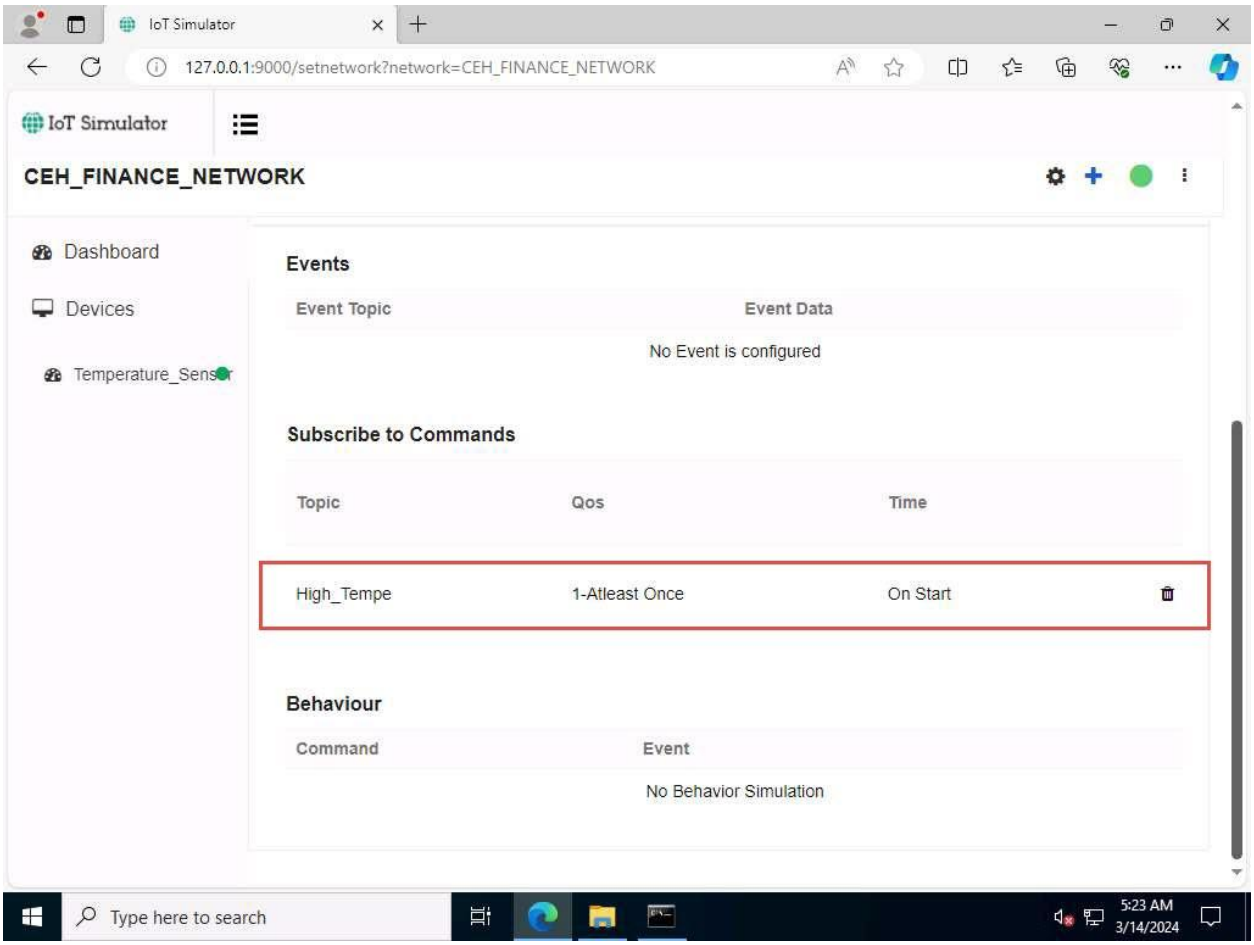
28. Switch back to Windows Server 2022 machine.
29. Next, we will create the **Subscribe command** for the device Temperature_Sensor.
30. Click on the **Plus** icon in **the top right corner** and select the **Subscribe to Command** option.



31. The **Subscribe for command - TS1** popup opens. Select **On start** under the **Subscribe on** tab, type **High_Tempe** under the **Topic tab**, and select **1 Atleast once** below the **Qos** option. Click on **Save**.



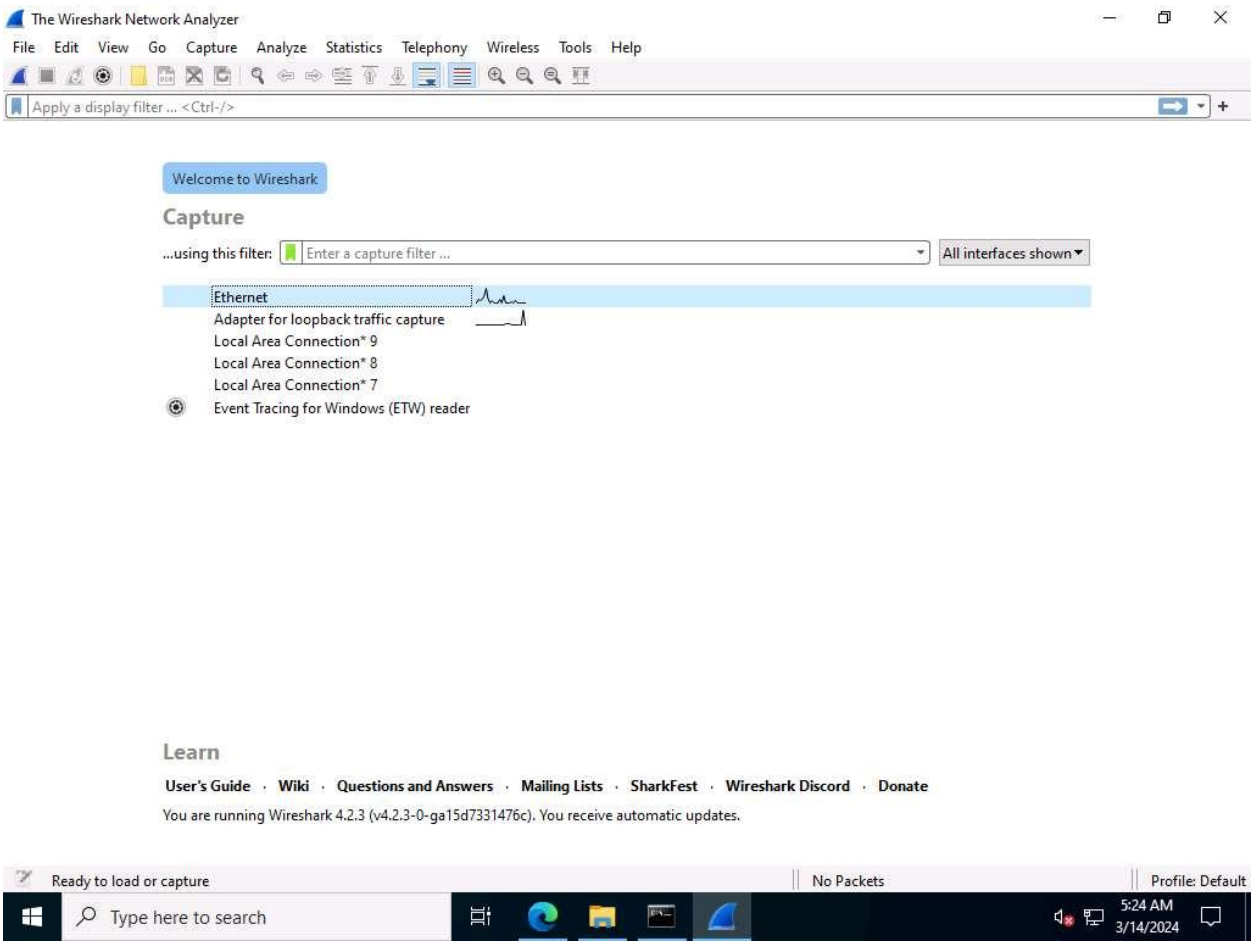
32. Scroll down the page, you can see the **Topic** added under the **Subscribe to Commands** section.



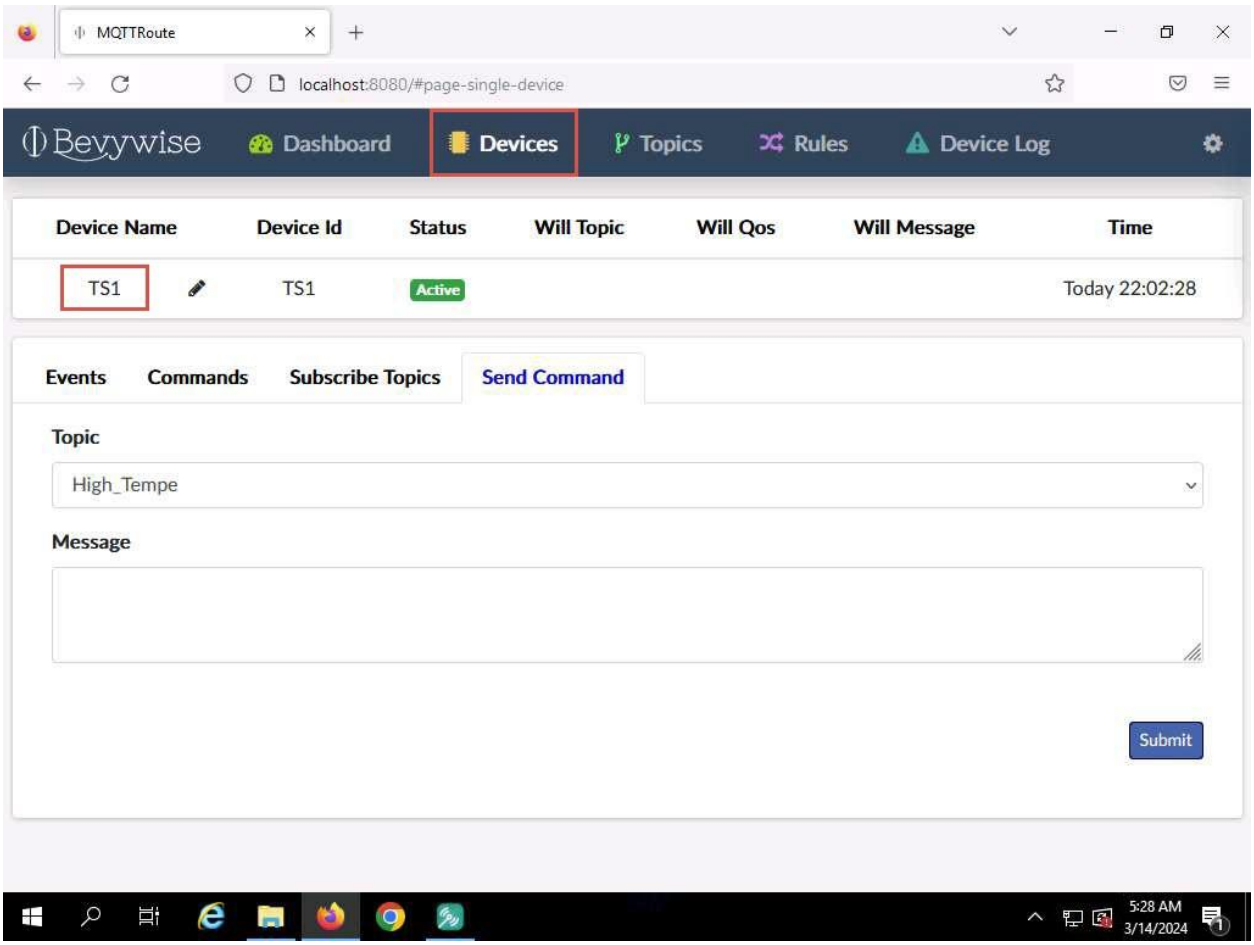
33. Next, we will capture the traffic between the **virtual IoT network and the MQTT Broker** to monitor the secure communication.
34. Minimise the Edge browser. Click **Type here to search** field on the **Desktop**, search for **wireshark** in the search bar and select **Wireshark** from the results.
35. The Wireshark Application window appears, select the **Ethernet** as interface.

Make sure you have selected interface which has **10.10.1.22** as the IP address.

If Software update popup appears click on **Skip this version**.

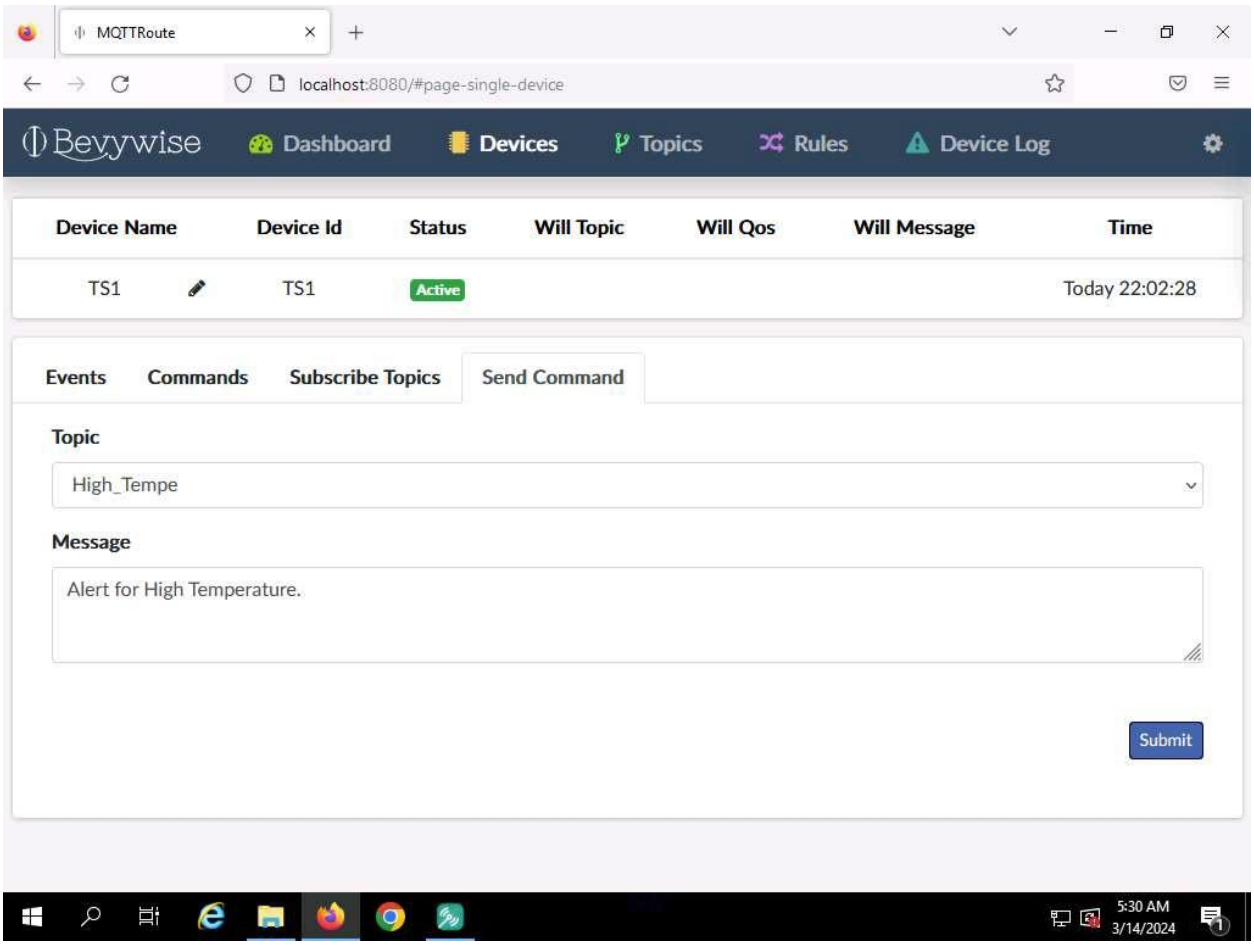


36. Click on the **Start Wireshark** icon to start the capturing packets, leave the Wireshark running.
37. Leave the IoT simulator running and switch to the Windows Server 2019 machine.
38. Navigate to **Devices** menu and click on connected device i.e.**TS1**.

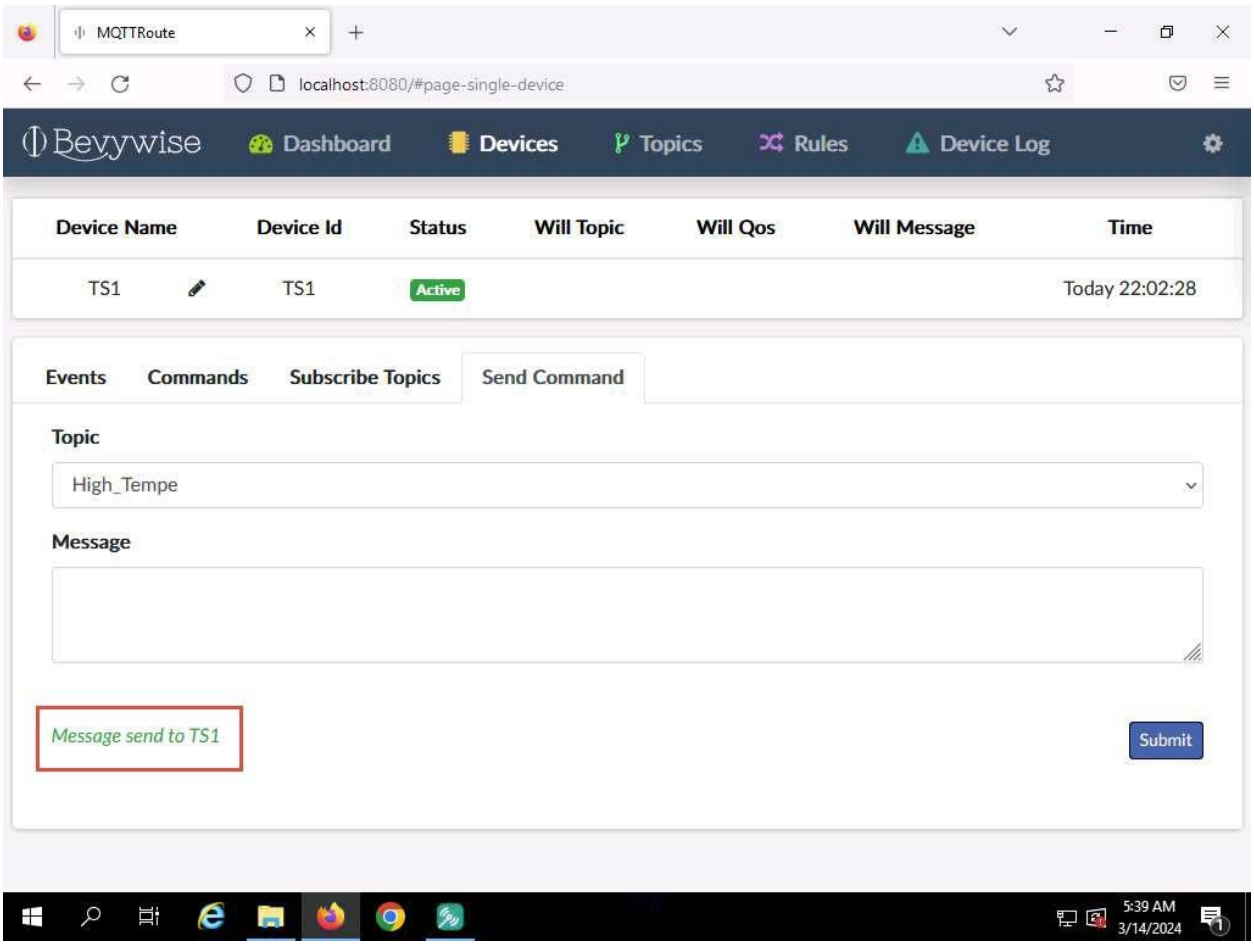


39. Now, we will send the command to **TS1** using the **High_Tempe** topic.

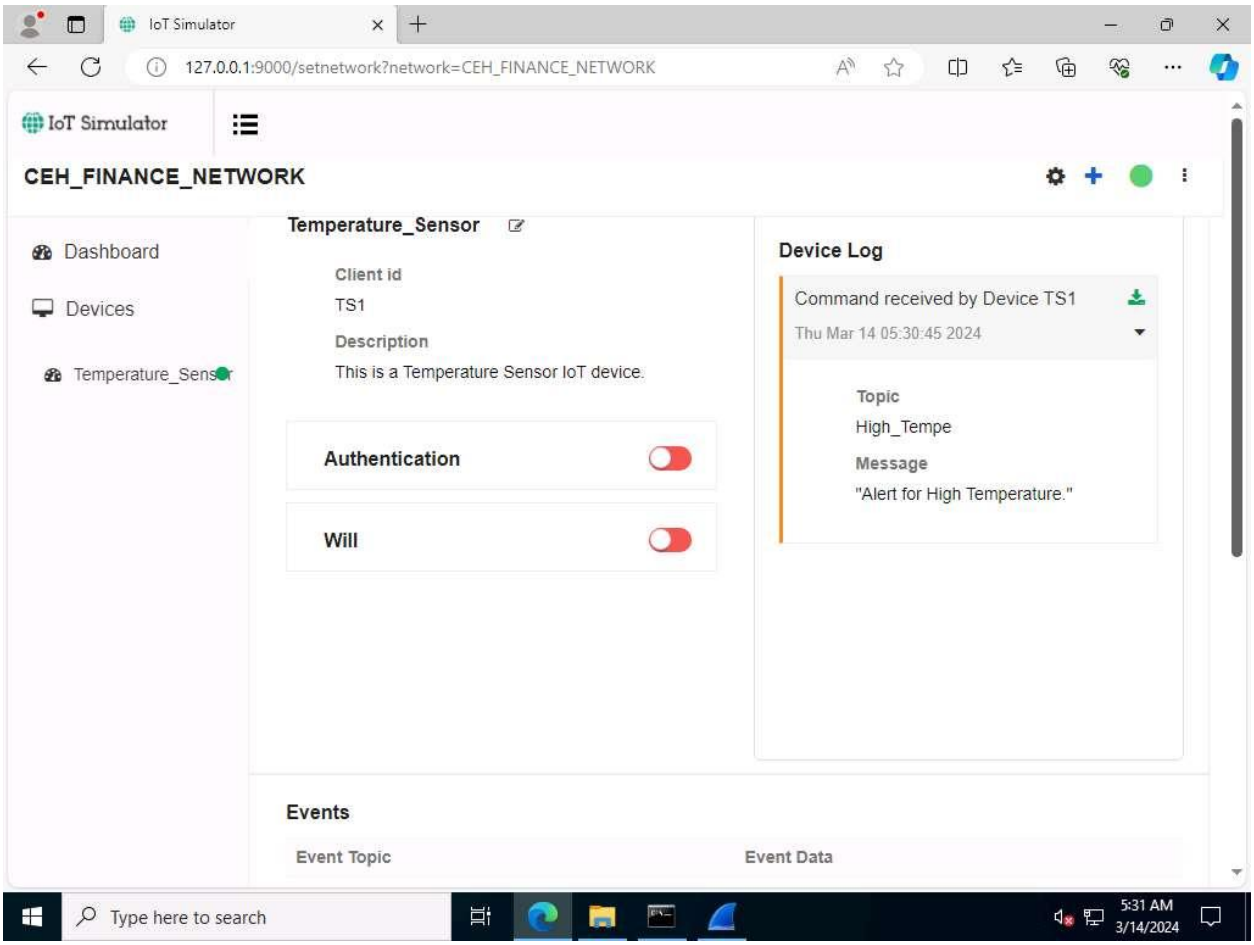
40. In **Send Command** section, select **Topic** as **High_Tempe**, type **Alert for High Temperature** in **Message** field and click on the **Submit** button.



41. **Message sent to TS1** appears under **Message** box which indicates that the message was successfully sent to TS1.

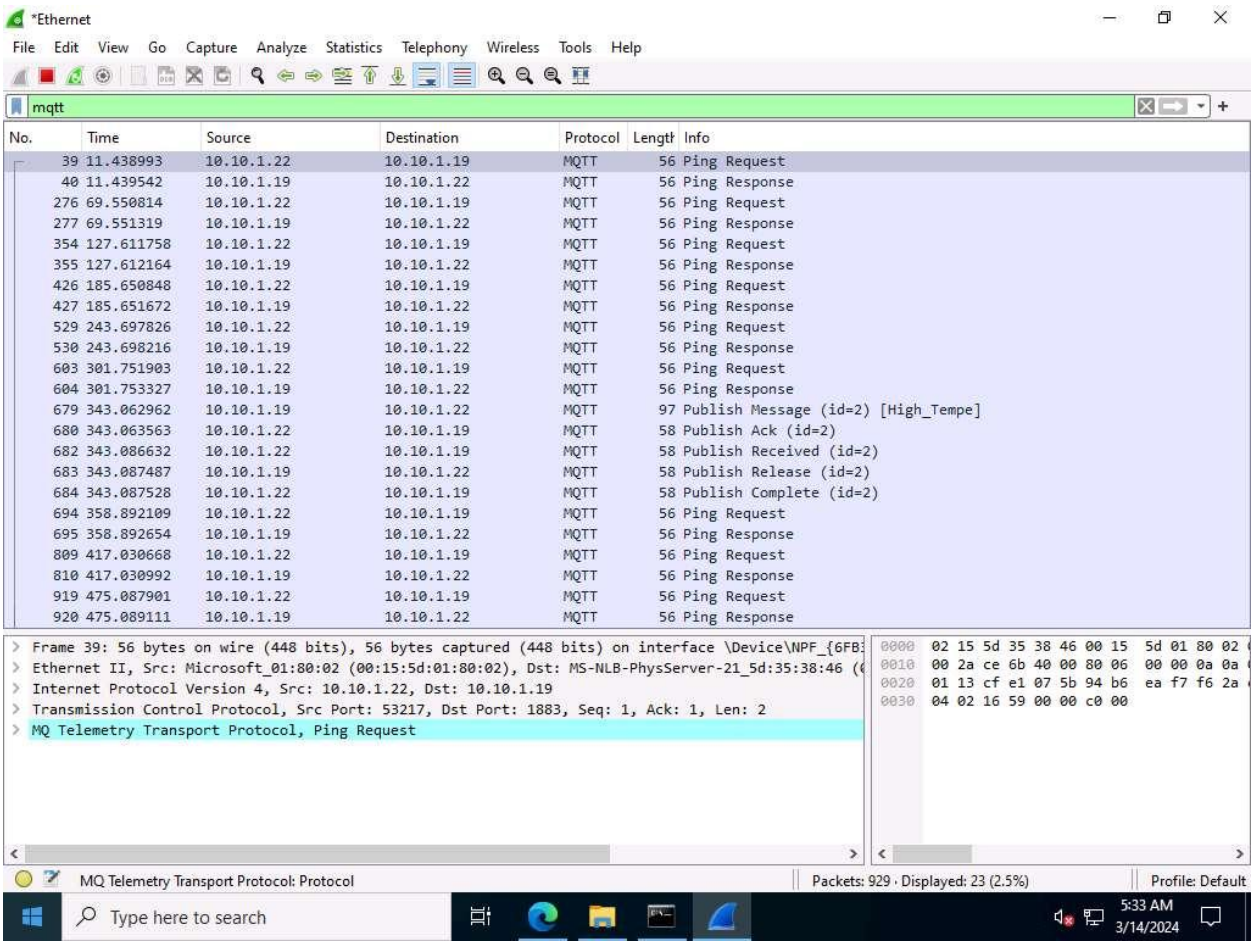


42. The message has been sent to the device using this topic.
43. Next, switch to Windows Server 2022 machine.
44. We have left the IoT simulator running in the web browser. To see the alert message, maximise the Edge browser and expand the arrow under the connected **Temperature_Sensor**, **Device Log** section.
45. You can see the alert message "**Alert for High Temperature**"



46. To verify the communication, we have executed **Wireshark** application, switch to the Wireshark traffic capturing window.

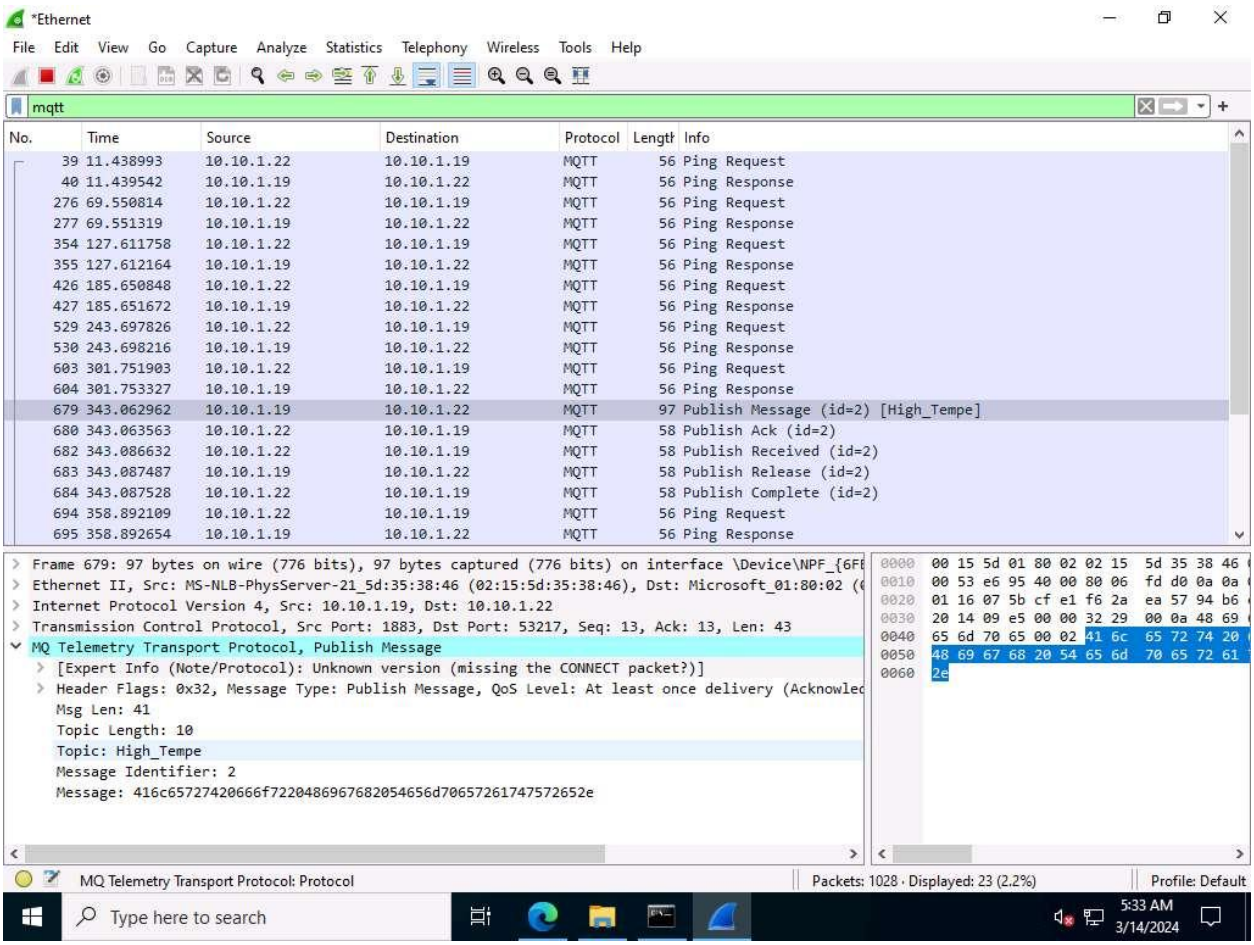
47. Type **mqtt** under the **filter** field and press **Enter**. To display only the MQTT protocol packets.



48. Select any **Publish Message** packet from the **Packet List** pane. In the **Packet Details** pane at the middle of the window, expand the **Transmission Control Protocol**, **MQ Telemetry Transport Protocol**, and **Header Flags** nodes.

49. Under the **MQ Telemetry Transport Protocol** nodes, you can observe details such as **Msg Len**, **Topic Length**, **Topic**, and **Message**.

50. Publish Message can be used to obtain the message sent by the MQTT client to the broker.

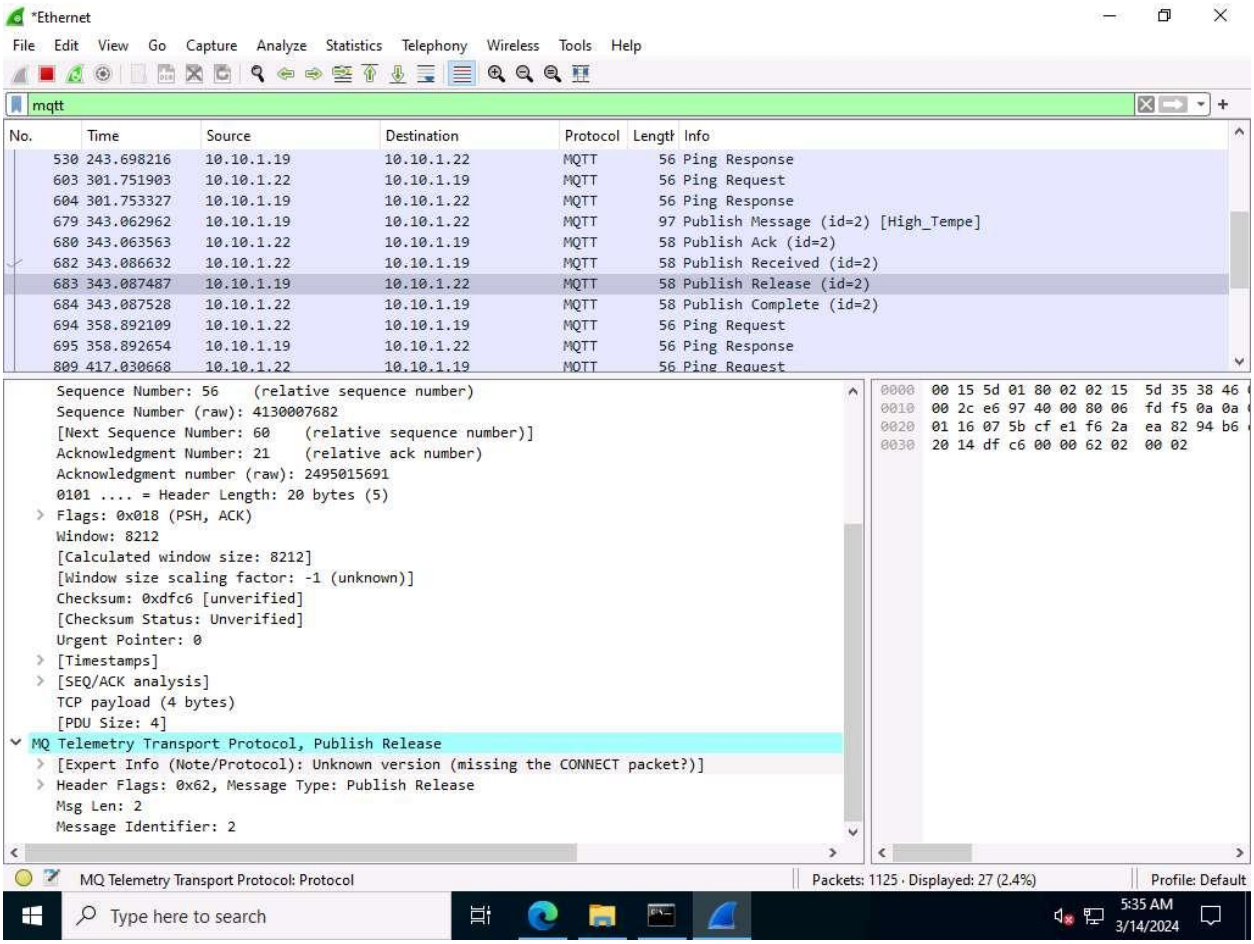


Note: After establishing a successful connection with the MQTT broker, the MQTT client can publish messages. The headers in the Publish Message packet are given below:

- Header Flags: Contains information regarding the MQTT control packet type.
- DUP flag: If the DUP flag is 0, it indicates the first attempt at sending this PUBLISH packet; if the flag is 1, it indicates a possible re-attempt at sending the message.
- QoS: Determines the assurance level of a message.
- Retain Flag: If the retain flag is set to 1, the server must store the message and its QoS, so it can cater to future subscriptions matching the topic.
- Topic Name: Contains a UTF-8 string that can also include forward slashes when it needs to be hierarchically structured.
- Message: Contains the actual data to be transmitted.
- Payload: Contains the message that is being published.

51. Select any **Publish Release** packet from the **Packet List** pane. In the **Packet Details** pane at the middle of the window, expand the **Transmission Control Protocol**, **MQ Telemetry Transport Protocol**, and **Header Flags** nodes.

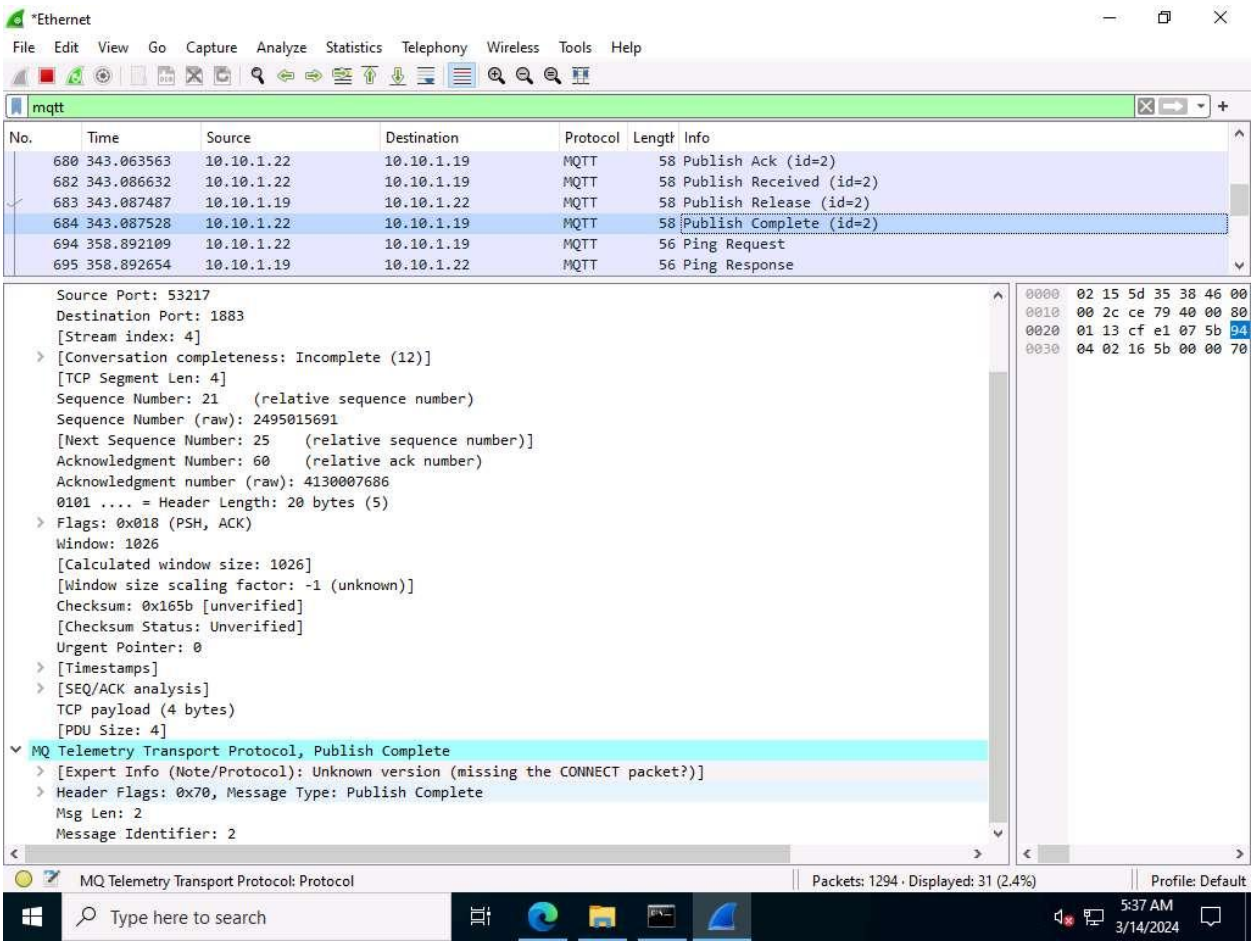
52. Under the **MQ Telemetry Transport Protocol** nodes, you can observe details such as **Msg Len**, **Message Type**, **Message Identifier**.



Note: A Publish Release (PUBREL) packet is the response to a Publish Received (PUBREC) packet.

53. Now, scroll down, look for the **Publish Complete** packet from the **Packet List** pane, and click on it. In the **Packet Details** pane at the middle of the window, expand the **Transmission Control Protocol**, **MQ Telemetry Transport Protocol**, and **Header Flags** nodes.

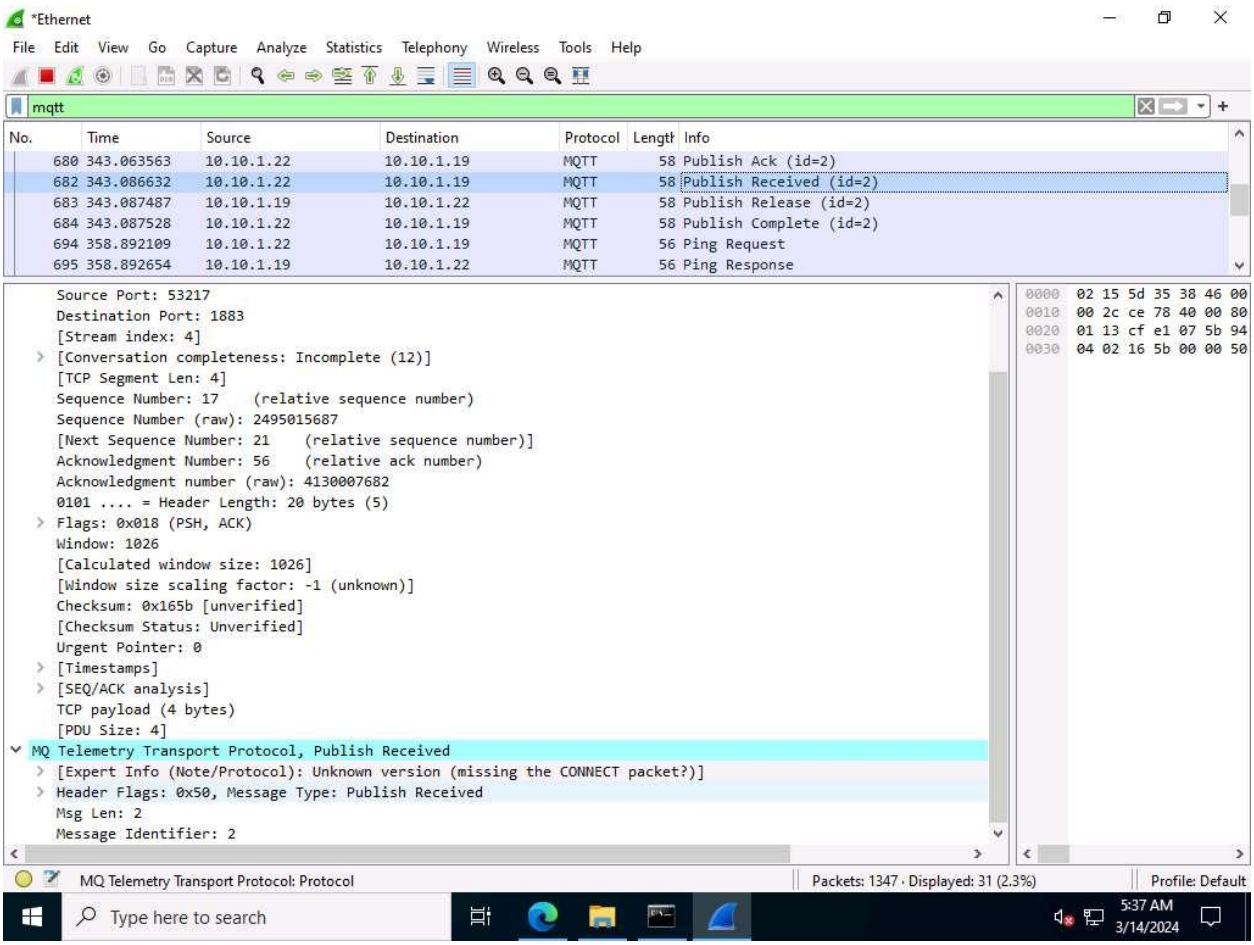
54. Under the **MQ Telemetry Transport Protocol** nodes, you can observe details such as **Msg Len** and **Message Identifier**.



Note: The Publish Complete (PUBCOMP) packet is the response to a Publish Release (PUBREL) packet.

55. Now, scroll down, look for the **Publish Received** packet from the **Packet List** pane, and click on it. In the **Packet Details** pane at the middle of the window, expand the **Transmission Control Protocol**, **MQ Telemetry Transport Protocol**, and **Header Flags** nodes.

56. Under the **MQ Telemetry Transport Protocol** nodes, you can observe details such as **Message Type**, **Msg Len** and **Message Identifier**.



57. Similarly you can select **Ping Request**, **Ping Response** and **Publish Ack** packets and observe the details.

58. This concludes the demonstration of capturing and analyzing MQTT protocol packets. Here, we analyzed different processes involved in the communication between an MQTT client and an MQTT broker using Wireshark. Understanding these metrics as well as the workflow can help you in quickly identifying the MQTT-related issues.

59. Close all open windows and document all the acquired information.

Question 18.2.1.1

Use Wireshark and Bevywise MQTT Route and Bevywise IoT Simulator to capture and analyze traffic between IoT devices. What is the default TCP port used by Bevywise MQTT Route to establish connection with Bevywise IoT Simulator?

Question 18.2.1.2

Use Wireshark and Bevywise MQTT Route and Bevywise IoT Simulator to capture and analyze traffic between IoT devices. What is the default WebSocket port used by Bevywise MQTT IoT Simulator to establish connection with Bevywise MQTT Route?