

## Research Article

# Detection of Bruteforce Attacks on the MQTT Protocol Using Random Forest Algorithm

Galuh Muhammad Iman Akbar<sup>1</sup>, Mokhamad Amin Hariyadi<sup>2</sup>, Ajib Hanani<sup>3\*</sup> 

<sup>1,2,3</sup>Department of Computer Engineering, State Islamic University of Maulana Malik Ibrahim Malang, Indonesia

\*Corresponding author: [ajib@uin-malang.ac.id](mailto:ajib@uin-malang.ac.id)



**Citation:** G.M.I.Akbar, M.A.Hariyadi, A.Hanani "Detection of Bruteforce Attacks on the MQTT Protocol Using Random Forest Algorithm". *Iota*, 2023, ISSN 2774-4353, Vol.03, 03. <https://doi.org/10.31763/iota.v3i3.630>

Academic Editor : Adi.P.D.P

Received : June, 23 2023

Accepted : June, 16 2023

Published : August, 02 2023

**Publisher's Note:** ASCEE stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2023 by authors. Licensee ASCEE, Indonesia. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution Share Alike (CC BY SA) license(<https://creativecommons.org/licenses/by-sa/4.0/>)

## Abstract:

Bruteforce is a hacking technique that launches an attack by guessing the username and password of the system that is the target of the attack. The Bruteforce attack on the MQTT protocol is an attack that often occurs on the IoT, so it is necessary to detect attacks on the MQTT protocol to find out normal traffic and brute force traffic. Random Forest was chosen because this method can classify a lot of data in a relatively short time, and the results from Random Forest can improve accuracy and prevent overfitting in the data classification process. This study uses two types of data: primary data from the hacking environment lab and secondary data from the IEEE Data Port MQTT-IOT-IDS2020 dataset. Trials on primary data and the results obtained are accuracy of 99.55%, precision of 100%, recall of 99.54%, and f-measure of 99.77%, the duration needed to get these results with 1796 data lines, i.e., for 0 seconds. As for the secondary data, the researcher obtained an accuracy of 99.77%, a precision of 100%, a recall of 99.43%, and an f-measure of 98.71%, the duration required to obtain these results with 85002 data lines, i.e., for 62 seconds.

**Keywords:** Bruteforce, Random Forest, IoT Attack Detection, Protocol MQTT

## 1. INTRODUCTION

Technology is a tool humans use to assist in the survival and comfort of life. Current technological developments will experience very rapid growth in the future. Currently, technological developments lead to an internet base that aims to increase the benefits of a connection useful for sharing files, performing remote control, and conducting analysis. The technology leading to the Internet base is IoT (Internet of Things). The IoT is a concept in which an object is given access to the internet, which is useful for transferring data, controlling the system, and other things helpful in easing work. The Internet of Things (IoT) is a concept that uses an internet network as the main infrastructure network that connects certain objects [1]. Message Queue Telemetry Transport (MQTT) is a messaging protocol that is lightweight enough to be supported by the smallest devices yet robust enough to ensure that important messages get to their destinations every time. MQTT devices such as smart energy meters, cars, trains, satellite receivers, and personal healthcare devices can communicate with each other and with other systems or applications [2].

## 2. THEORY

### 2.1 Protocol MQTT

The MQTT (Message Queue Telemetry Transport) protocol is a publish-subscribe-based lightweight messaging protocol used on top of the TCP/IP protocol suite. The following are the features contained in the MQTT protocol [3].

1. Publish/Subscribe is a pattern for distributing messages from one-to-many systems.
2. Messaging transport is agnostic to the content of the payload.
3. TCP/IP as network connectivity.
4. Three levels of QoS (Quality of Service):
  - QoS - 0: Data will be communicated at most once.
  - QoS - 1: Data will be communicated at least once.
  - QoS - 2: Data will be sent exactly once.

### 2.2 Random Forest

Random Forest in Figure 1 is a machine learning algorithm with an ensemble method that can be used for classification and regression [9-12]. A Random Forest consists of a collection of decision trees associated with a bootstrap sample from a dataset[4]. This method creates a decision tree consisting of internal, root, and leaf nodes by randomly taking some attributes and data according to the provisions in force. The internal node is a branching node, where this node has at least two outputs and only one input. The root node is the node that is located at the top, commonly referred to as the root of the decision tree[5]. At the same time, the leaf node or terminal node is the last node with only one input without output. The entropy value is calculated when the decision tree is made and is used to determine the level of attribute impurity and the importance of information gain. There are three essential aspects of a random forest method [6] :

1. Perform bootstrap sampling.
2. Make predictions on each tree with a random predictor.
3. Combine the results from each decision tree by selecting the average value for the regression.

Based on a complete study of previous researchers using random forest techniques on this type of dataset to solve cyber-attacks on IoT networks because random forests predict Data Type Probing (DTP), Malicious Control (MC), Malicious Operation (MO), Scan (SC) attacks, Spying (SP), Wrong Settings (WS). Detection results that can be achieved using the random forest algorithm obtained an accuracy of 99.4% [7].

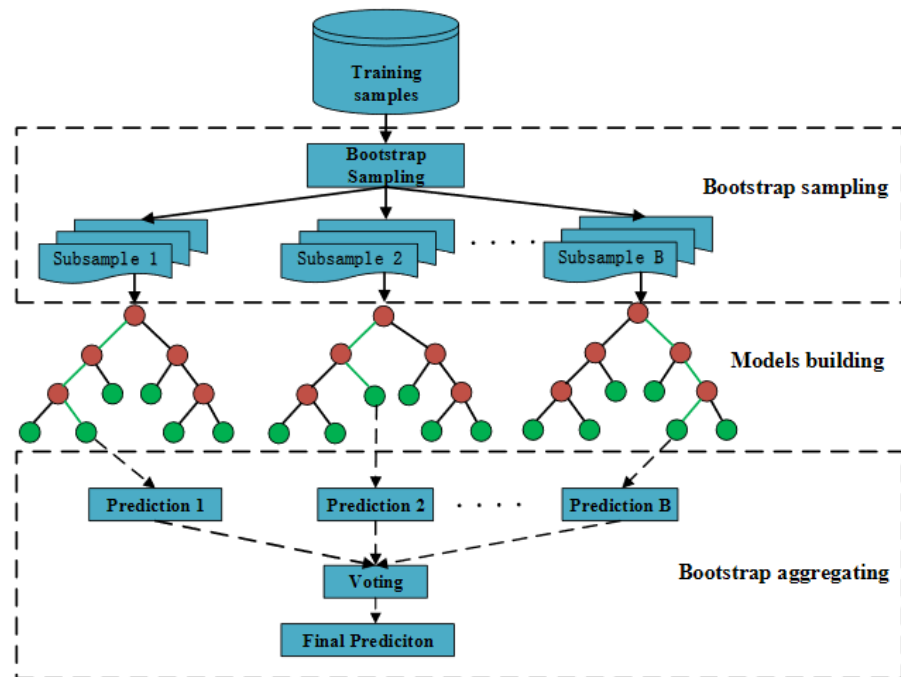


Fig 1. Flowchart Random Forest

### 3. METHOD

#### 3.1 System Design

In this chapter, the author needs to explain the methods used, e.g., Algorithm, Flowchart, Design., Etc. Paragraphs must be regular. All sections must be aligned, that is, both left and right. In the system design block diagram presented in Figure 2, the researcher separates primary data and secondary data; if the data that is processed as training is taken from primary data, then the testing data will be taken from primary and vice versa if the training data is taken from secondary data then data testing is also taken from secondary data, primary data will be divided into two, i.e., training data and testing data, as well as secondary data. In the first step, the researcher collects secondary datasets and uses used for training; then, at the pre-processing stage, which consists of a collection of such data with irregular ranges, then normalization is carried out, namely collecting data with the same range. Next, the bootstrap stage is to re-sample the original data in a new sub-sample. This new sample created a decision tree that was used as a training model.

Test data is entered from a collection of decision trees that have been formed. The same thing is done at the testing stage, namely with the same normalization as in the training data process. Test data that has been normalized is entered into the training model. The results to be obtained from the testing process are in the form of leaf values in each decision tree. This leaf value will be collected as the majority of values that often appear the majority of these values will then be concluded as prediction results in the system.

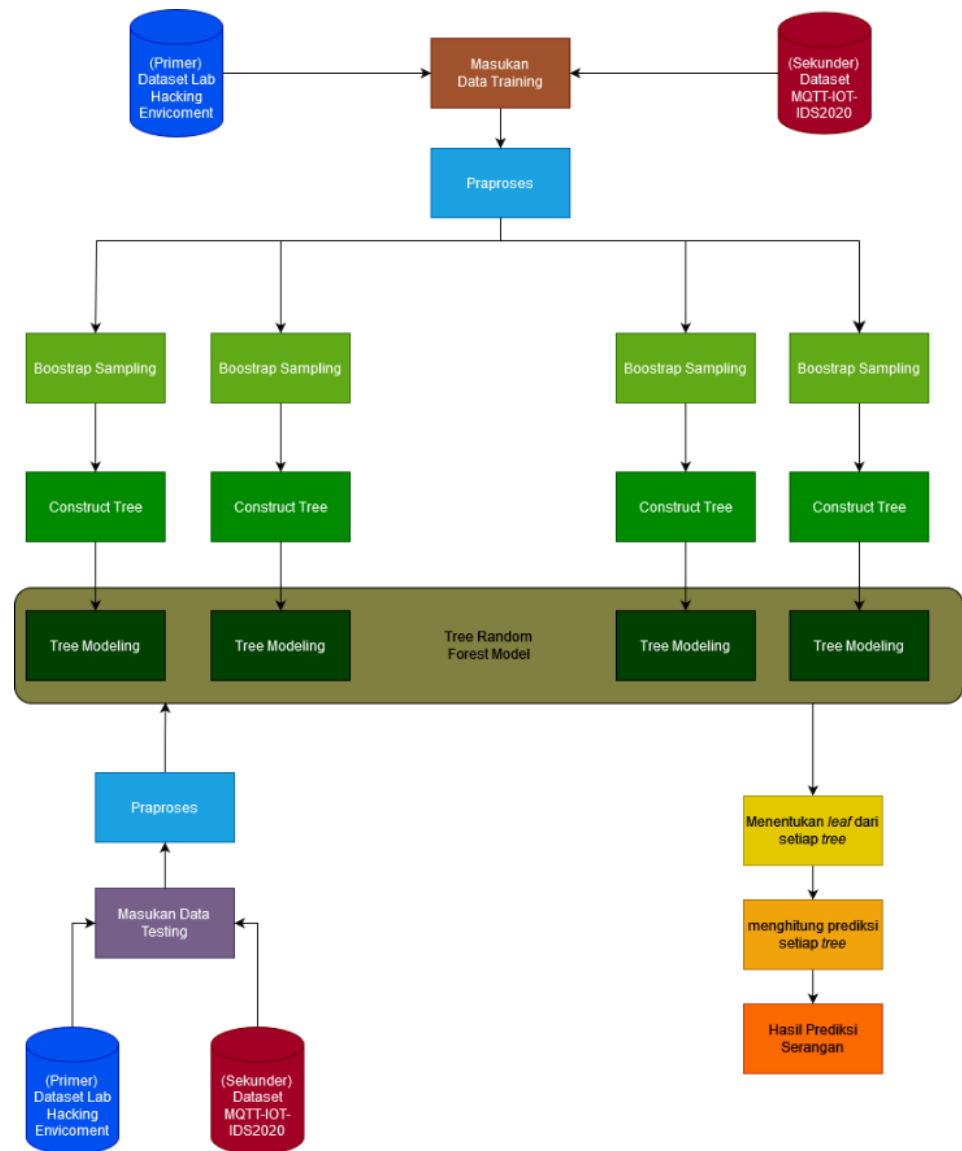


Fig 2. System Design Block Diagram

### 3.2 Data Pre-processing

The data obtained in the study cannot be directly processed using the random forest method; it is necessary to have model training first. So before doing the modeling, the researcher uses the primary data that the researcher has obtained from the results of data retrieval on network traffic for normal data as many as 46 lines and brute force data as many as 1750 lines. In this case, the proportion used for training data is 75%, while for data testing as much as 25%. As for the secondary data that researchers got from the MQTT-IOT-IDS2020 dataset that will be used, namely normal data 52082 lines and for brute-force data as many as 32920 lines, the proportion used for training data is 75%, while for testing data, it is 25%. From the two datasets that the researcher has made, an examination will be carried out first so that all data can work according to the procedures made by the researcher. In the data pre-processing, the numerical data normalization stage

is carried out. Normalization, namely changing numeric data to the same scale, calculates numeric data as equation 1.

$$\text{scaled}(x) = \frac{x_i - x_{i\min}}{x_{i\max} - x_{i\min}} \quad (1)$$

### 3.3 Random Forest for Model Building

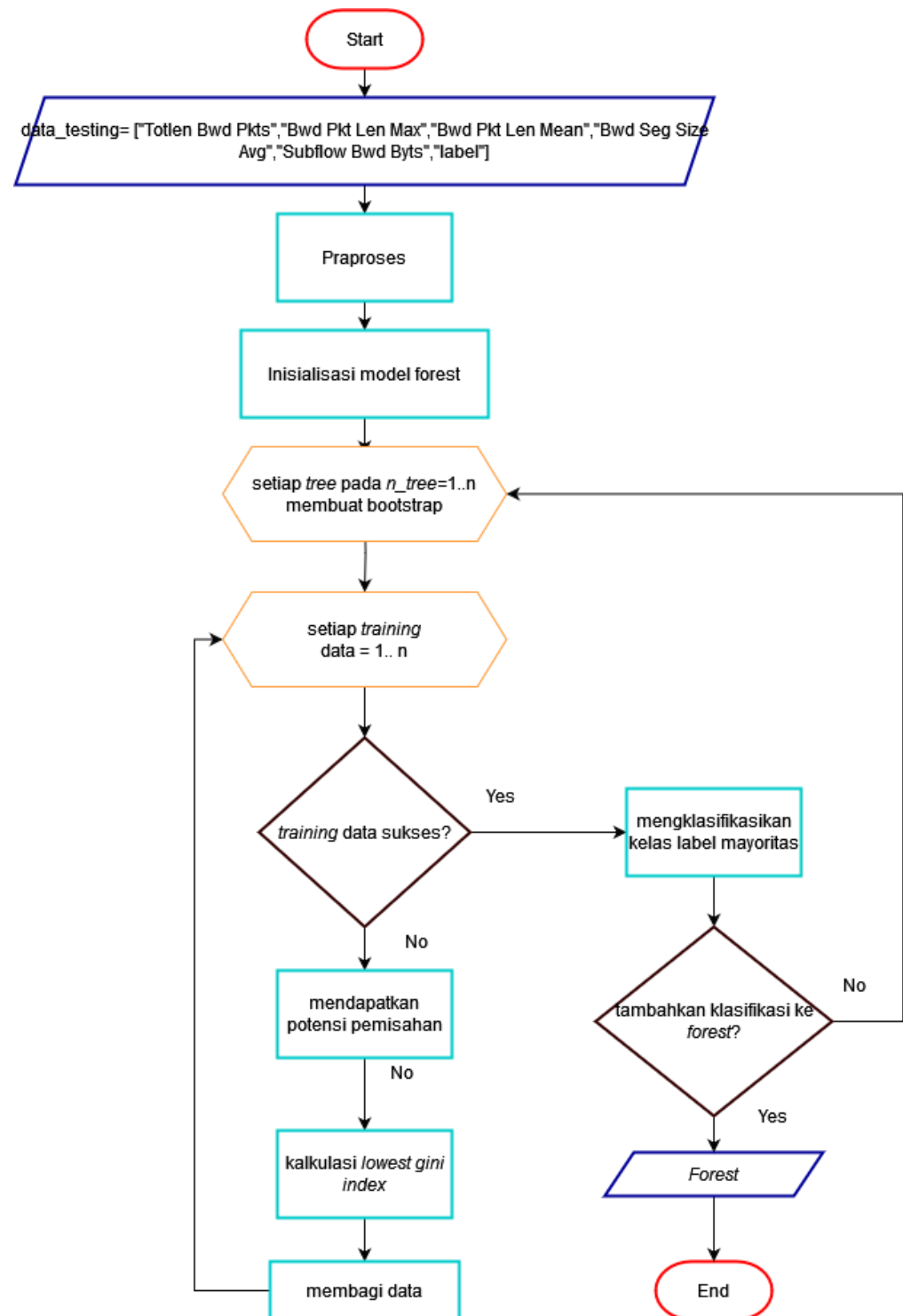


Fig 3. Flowchart Model Random Forest Data Training

In Figure 3, the steps taken by the researcher are explained to create training data for the system. The initial stage is to input training data and then take random samples in the dataset. The dataset must be prepared when bootstrapping, which is then taken randomly. The dataset column will be kept in accordance with the original, while the rows in the dataset are allowed to be randomized and take the same row to be placed in the new sample. Bootstrap is obtained by re-sampling the original data placement in a new sub-sample, after obtaining the formation in the form of bootstrap or sample data, then forming a tree for each bootstrap. Like a Decision tree, a tree is formed with three things, i.e., the root node (tree top), internal node (tree branch), and leaf node (tree root). To build a tree in this study, researchers used an additional method, namely the Gini Index method. In making a decision tree, a method is needed to build the tree. The researchers used the Gini Index method. In making a decision tree, a method is needed to build the tree. The researchers used the Gini Index method. The following is the formula for obtaining the impurity value, as following equation 2.

$$Impurity = 1 - \sum P_i^2 \quad (2)$$

Where is the value of  $P_i$  as the probability value of the label value that appears in the column concerned? The  $P$  value is obtained from the selected nodes, then divided by the total of all nodes. Meanwhile, the Gini Index value can be calculated using Equation 3.

$$Gini(x) = 1 - \sum \frac{n_i}{n} * Impurity \quad (3)$$

The value  $n_i$  is the number of partitioned data in the set, and  $n$  is the total number of data sets. This value is obtained from the number of tuples belonging to class  $i$  in the child, while  $n$  is the number of nodes  $n$  the child.

Furthermore, in Figure 4 of the MQTT attack detection system, entering a data test is carried out first in the model tree created earlier. The testing data entered must go through the pre-processing normalization process. The data that has been entered in the detection model that has been created will produce a list of decision tree leaves. After that, when the process starts, there will be a set time. Then do the separation between Normal data and MQTT brute force data on the leaf. After that, there will be a process of calculating the time generated during the testing process. Then the final result is determined by the leaf that often appears as a result of traffic detection using Random Forest to perform MQTT brute force detection, and the duration is also required during the testing time.

### 3.4 Random Forest MQTT Attack Detection

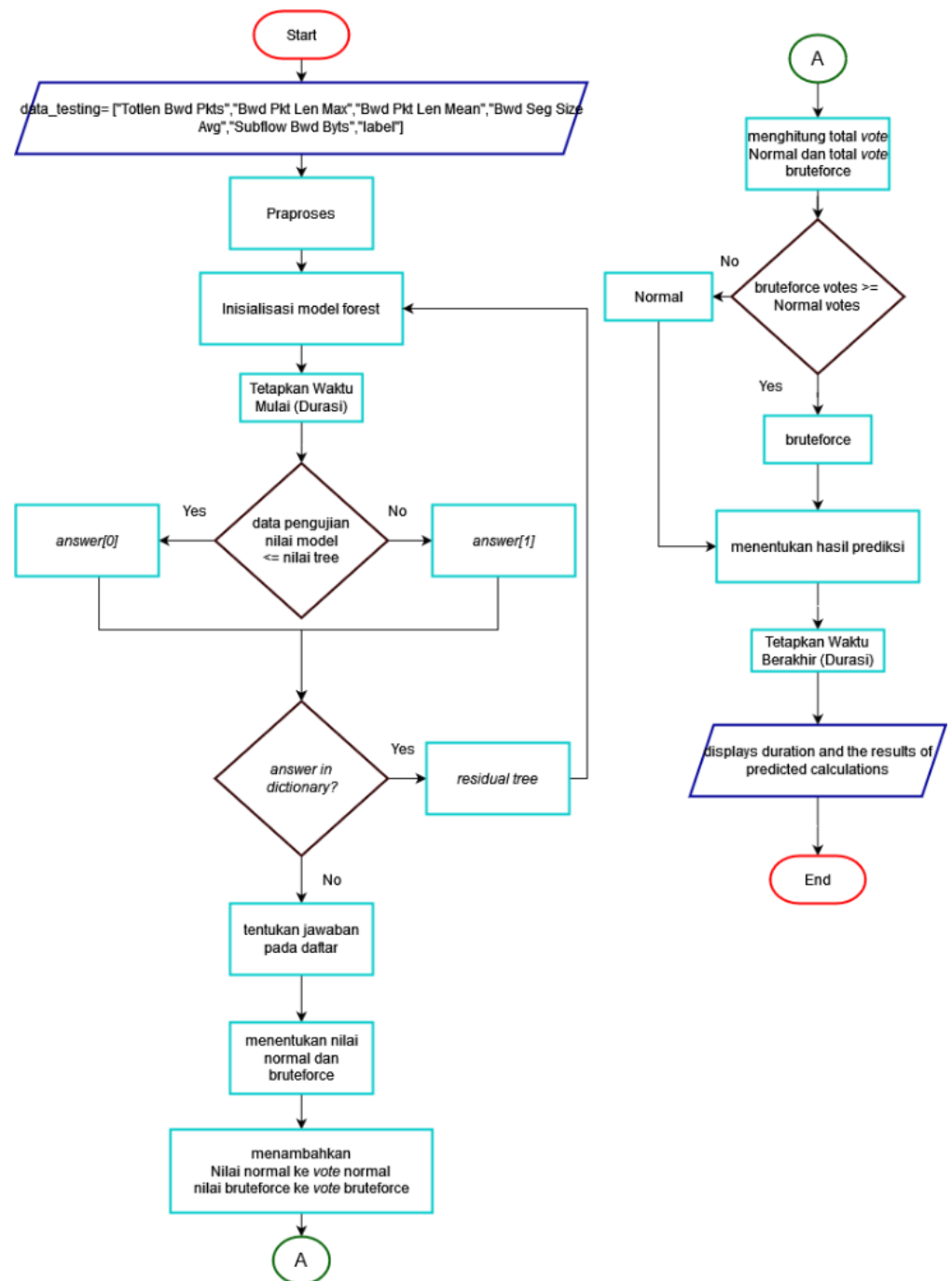
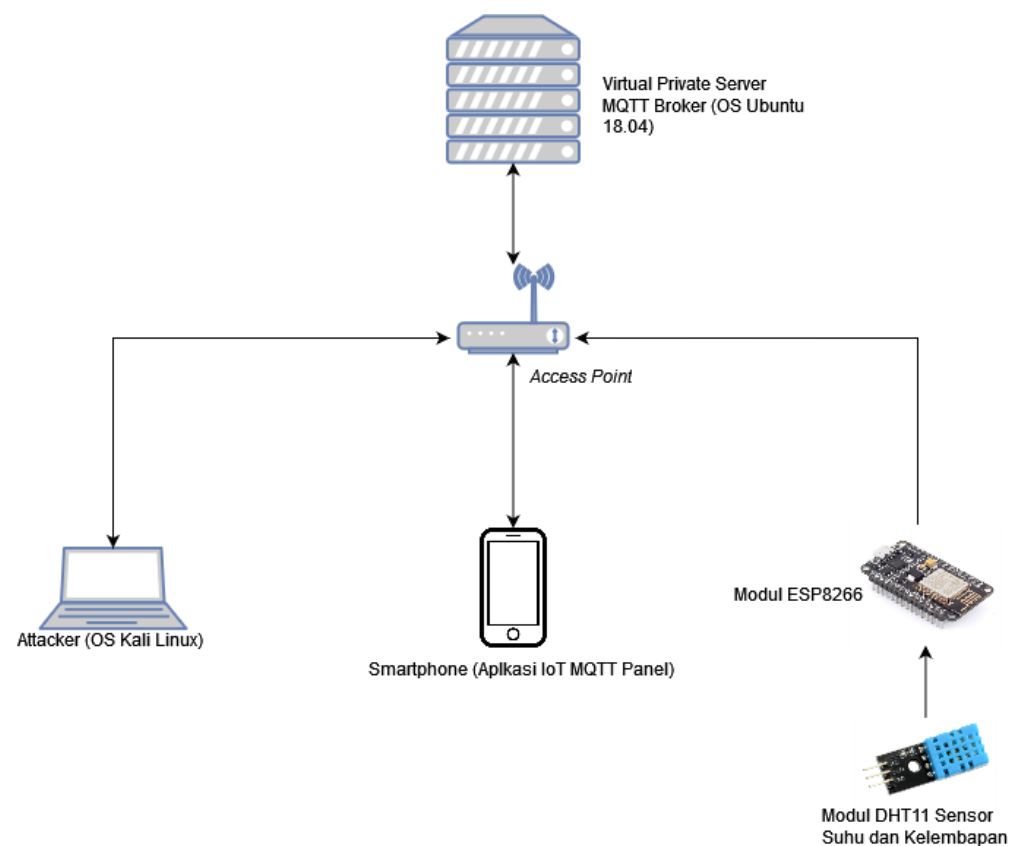


Fig 4. Flowchart Model Random Forest Data Testing

### 3.5 Lab Hacking Environment

Furthermore, Figure 5 is a Hacking Environment Lab Design. In the design of the Lab Hacking Environment on the server, the MQTT broker will run using Mosquitto and also capture data traffic on the server. The access point is used to connect devices on a network. Attackers will carry out attacks on services on the MQTT protocol using tools, namely the mqttsa tools used to brute force the MQTT broker. The smartphone device will send a subscription to the MQTT service and read the temperature and humidity sensor results. The ESP8266 module will do the programming to read the temperature and humidity sensors and publish them to the MQTT Broker service via the access point. Moreover, The material requirements for making the Lab Hacking Environment are presented in Table 1.



**Fig 5.** Hacking Environment Lab Design



**Table 1.** Material requirements in making Lab Hacking Environment

No	Role	specification
1	VPS MQTT (Broker Mosquitto)	- Ubuntu 18.04 - 1 GB RAM
2	IoT Device	- ESP8266 module - DHT11 Module (Temperature and Humidity Sensor)
3	Smartphone	Apps IoT MQTT Panel
5	OS Kali Linux	- Nmap - Mqttsa is used to brute force the MQTT broker

The following is Table 1 of the scope of devices and networks in the Lab hacking environment.

**Table 2.** Scope of Devices and Networks

No	Device Name	Device	Role	IP address
1	VPS Ubuntu 18.04	Virtual machine	MQTT (Broker Mosquitto)	117.53.144.151
2	OS Kali Linux	Virtual machine	Attacker	192.168.43.160
3	ESP8266 and DHT11	Modul	Sensor data sender	192.168.43.44
4	Mobile (IoT MQTT Panel)	Smartphone	Data receiver	192.168.43.2

### 3.6 Data Acquisition

There are two types of data, e.g., primary and secondary data. Both of these data have the same function; primary data is data built by researchers, while secondary data is datasets from experts. Some of the secondary data will be used to build research models, and some will be used for data testing, which functions to test the system or model scenarios that are built. The following is the Lab Hacking Environment configuration for data acquisition:

#### 3.6.1 Configure Lab Hacking Environment

Data acquisition begins with building an MQTT service on the Ubuntu server and configuring the Ubuntu server. This MQTT service will connect the broker with the client. In hacking, researchers will use the Debian-based Kali Linux operating system as a hacker who will attack the MQTT service. Moreover, Figure 6 is a Password setting for the MQTT broker on the Ubuntu server by adding the lines `password_file` and `allow_anonymous`; `password_file` refers to the location of the password path that the researcher has made while `allow_anonymous false` is a configuration so that all users must be authenticated first to access the data on the broker MQTT as follows.

```

GNU nano 2.9.3 /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

password_file /etc/mosquitto/passwd
allow_anonymous false

```

Fig 6. File mosquito Configuration

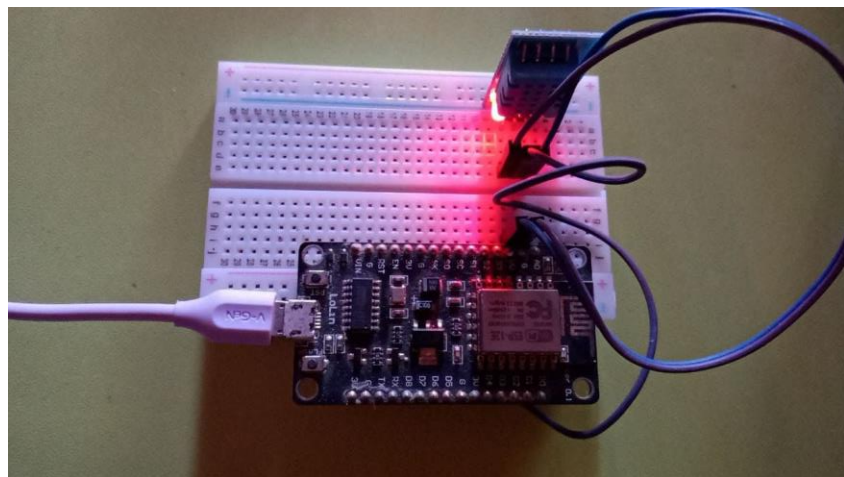


Fig 7. ESP8266 and DHT11 Connectivity

The following is a configuration from the DHT11 sensor side and the ESP8266, as shown in Figure 7. The data taken from the sensor is JSON, which will be sent to the MQTT broker on the Ubuntu server. The following is the MQTT Panel application configuration as the Figure 8; the value at index 0 is the temperature value, while for index 1 is the humidity value.

**Left Panel (Kelembapan):**

- Panel name: Kelembapan
- Disable dashboard prefix topic: ☐
- Topic: v2/IOT2021/HP/json
- Payload min: 10, Payload max: 200
- Arc color range: Green (80), Yellow (150), Red
- Unit: RH, Factor: 1
- Enable notification: ☐
- Payload is JSON Data: ☒
- JsonPath for subscribe: \$.cikarang.suhu\_kelembapan[1].value
- Show received timestamp: ☐
- QoS: 0

**Right Panel (Suhu):**

- Panel name: Suhu
- Disable dashboard prefix topic: ☐
- Topic: v2/IOT2021/HP/json
- Payload min: 10, Payload max: 200
- Arc color range: Green (80), Yellow (180), Red
- Unit: °C, Factor: 1
- Enable notification: ☐
- Payload is JSON Data: ☒
- JsonPath for subscribe: \$.cikarang.suhu\_kelembapan[0].value
- Show received timestamp: ☒
- QoS: 0

Fig 8. MQTT Temperature and Humidity Configuration

### 3.6.2 Hacking Simulation on the MQTT service

After configuring the lab hacking environment, the next step is to simulate hacking into the MQTT service by carrying out a Bruteforce attack. The Ubuntu server runs the MQTT mosquito service, and the hacker operating system uses Kali Linux. When the broker service has been run, the researcher will see the service on the mosquito. On the Kali Linux operating system, the function of mqttsa tool runs Bruteforce Credentials by shooting credentials on the MQTT service using a directory attack. Moreover, The hacking simulation is carried out by running the mqttsa script as the executor, who will carry out the attack by guessing the username and password in the Mosquitto service. The first step is to scan with the help of nmap tools to find out which services are running on the server; with the command "nmap -p 1883 <ip-server>" as in the table 2, the server ip has a value of 117.53.144.151. The following is the enumeration result. Nmap scanning tools results can be seen specifically in Figure 9.

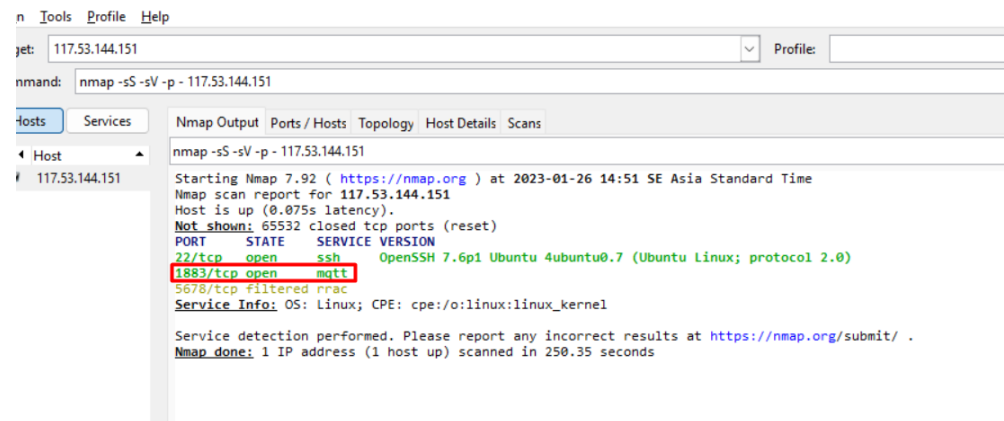


Fig 9. Nmap scanning tool results

Based on the scanning results, it is known that the server has a running MQTT service, but the service has credentials that the attacker must know. So the brute force technique was used; the researcher made a wordlist of 500 words and then ran the mqttsa script with the command "python3 mqttsa.py 117.53.144.151 -p 1883 -u administrator -w wordlist.txt". Following is the result in Figure 10.

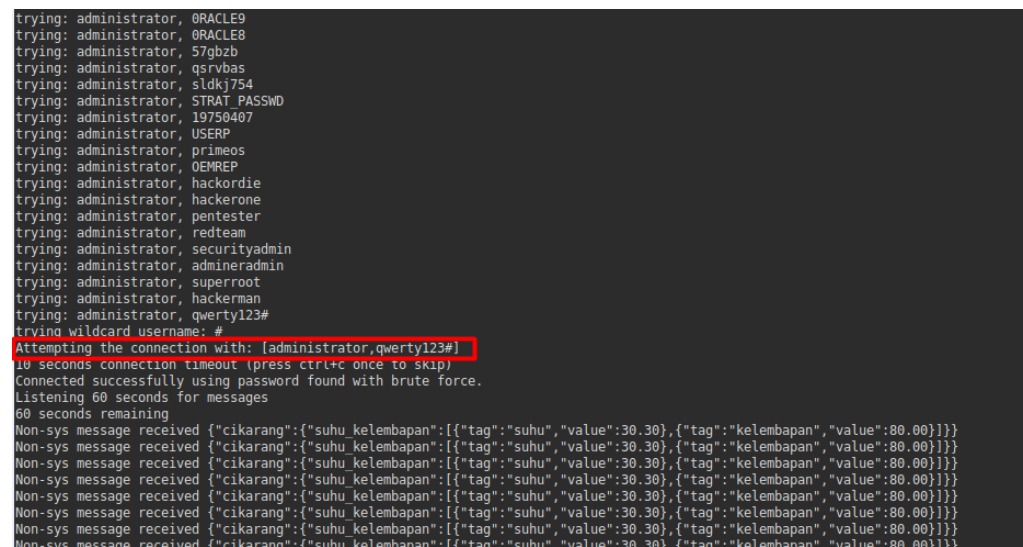
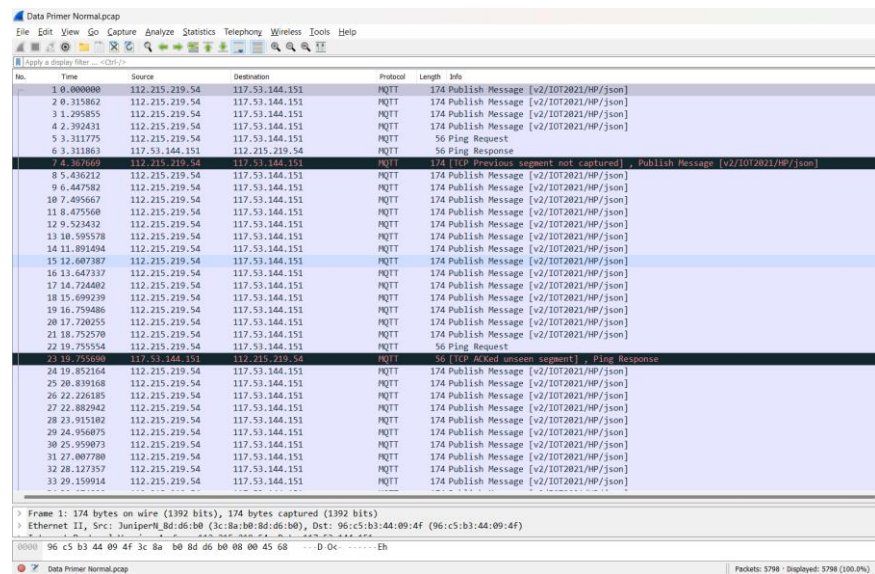


Fig 10. Result Bruteforce

### 3.6.3 Network Traffic Recording

The process of recording network traffic using the Tools App Wireshark. When the MQTT service hacking process takes place, hackers use mqttsa tools. The Wireshark application will record network traffic running on the MQTT service during the hacking process. The configuration is set to run traffic capturing on devices connected to WiFi, and this is because the MQTT service will connect to WiFi. Wireshark will record network traffic between the MQTT service and the client, in this case, the mqttsa tools that act as clients, so these tools carry out a dual role, namely as a hacker.

Recording network traffic using Wireshark tools aims to retrieve test data because it requires data from hackers, responses from the MQTT service, and normal data. Normal data in Figure 11 is obtained from normal client subscriptions to the MQTT service and responses from the MQTT service without using MQTTSa tools. The traffic is recorded by Wireshark, which is then used as normal traffic on the Network. All data that has been successfully captured will be stored in the PCAP extension format. Researchers retrieve IoT network traffic data from normal traffic. The figure shows that the attack recording process has not been launched.

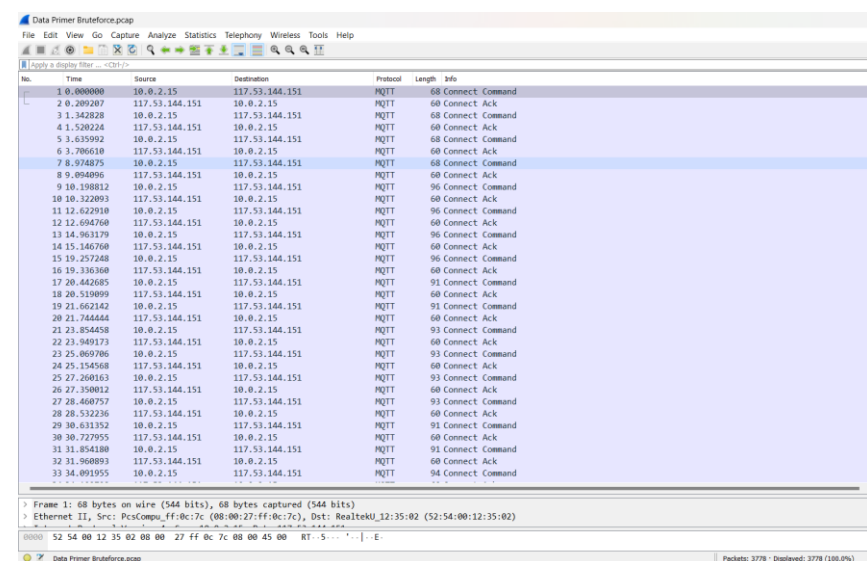


The screenshot shows a Wireshark capture titled 'Data Primer Normal.pcap'. The packet list on the left shows a series of MQTT publish messages and ping requests. The packet details pane on the right shows the structure of an MQTT publish message, including the Topic Name, Payload, and QoS. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
2	0.115862	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
3	1.295855	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
4	2.392431	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
5	3.311775	112.215.219.54	117.53.144.151	MQTT	56	Ping Request
6	3.311863	117.53.144.151	112.215.219.54	MQTT	56	Ping Response
7	4.405664	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
8	5.436212	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
9	6.447582	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
10	7.495667	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
11	8.475568	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
12	9.520432	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
13	10.595578	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
14	11.891494	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
15	12.687387	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
16	13.647337	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
17	14.724482	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
18	15.699239	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
19	16.759486	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
20	17.749255	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
21	18.752570	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
22	19.755554	112.215.219.54	117.53.144.151	MQTT	56	Ping Request
23	19.755600	117.53.144.151	112.215.219.54	MQTT	56	Ping Response
24	19.832164	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
25	20.839168	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
26	22.226185	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
27	22.882942	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
28	23.915182	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
29	24.956875	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
30	25.959073	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
31	27.007780	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
32	28.127357	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]
33	29.159914	112.215.219.54	117.53.144.151	MQTT	174	Publish Message [v2/1072021/HP/Json]

Fig 11. Normal primary data

When tools Wireshark is running, the attacker tries to attack the MQTT broker by carrying out a brute-force attack, and the following is Figure 12 of a brute-force attack.



The screenshot shows a Wireshark capture titled 'Data Primer Bruteforce.pcap'. The packet list on the left shows a series of MQTT connect commands and acknowledgments. The packet details pane on the right shows the structure of an MQTT connect command, including the Client ID, Username, Password, and Clean Session flag. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.15	117.53.144.151	MQTT	68	Connect Command
2	0.209287	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
3	1.242828	10.0.2.15	117.53.144.151	MQTT	68	Connect Command
4	1.520224	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
5	3.635992	10.0.2.15	117.53.144.151	MQTT	68	Connect Command
6	3.706610	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
7	8.3974875	10.0.2.15	117.53.144.151	MQTT	68	Connect Command
8	9.004096	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
9	10.198812	10.0.2.15	117.53.144.151	MQTT	96	Connect Command
10	10.322893	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
11	12.622010	10.0.2.15	117.53.144.151	MQTT	96	Connect Command
12	12.694768	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
13	14.963179	10.0.2.15	117.53.144.151	MQTT	96	Connect Command
14	15.146768	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
15	19.257248	10.0.2.15	117.53.144.151	MQTT	96	Connect Command
16	19.336368	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
17	20.442685	10.0.2.15	117.53.144.151	MQTT	91	Connect Command
18	20.519099	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
19	21.662142	10.0.2.15	117.53.144.151	MQTT	91	Connect Command
20	21.744444	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
21	23.854458	10.0.2.15	117.53.144.151	MQTT	93	Connect Command
22	23.949173	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
23	25.869786	10.0.2.15	117.53.144.151	MQTT	93	Connect Command
24	25.154568	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
25	27.260163	10.0.2.15	117.53.144.151	MQTT	93	Connect Command
26	27.358012	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
27	28.460757	10.0.2.15	117.53.144.151	MQTT	93	Connect Command
28	28.532236	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
29	30.631352	10.0.2.15	117.53.144.151	MQTT	91	Connect Command
30	30.727955	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
31	31.854188	10.0.2.15	117.53.144.151	MQTT	91	Connect Command
32	31.960893	117.53.144.151	10.0.2.15	MQTT	68	Connect Ack
33	34.091955	10.0.2.15	117.53.144.151	MQTT	94	Connect Command

Fig 12. Bruteforce primary data

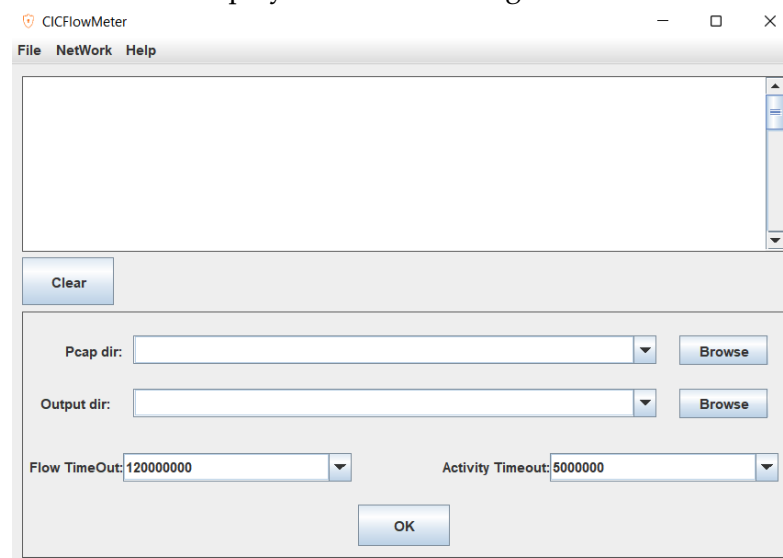
Researchers do not combine normal data and attack data to get precise and pure data. For normal data, the first data is collected until it is finished, then stored under the name "Primary Normal Data" and data retrieval on attacks is separated between attacks "Bruteforce Primary Data." Moreover, Table 3 presents the number of lines that were successfully recorded.

**Table 3.** Primary Data

Primary data		
No	Network Traffic	Data Row
1	Normal	5798
2	Bruteforce	3778

### 3.7 Network traffic extraction

After the network traffic recording process is carried out, all data will be stored in a file with the PCAP extension format. However, PCAP is still not in accordance with the data training format, so it is necessary to extract data into data that is suitable for data training, namely files with the CSV extension format. The tool that functions as a converter to convert PCAP data into CSV data is CICFlowMeter, which is a generator and analyzer for detecting anomalies in the Network. These tools have been widely used in cybersecurity datasets such as IPS/IDS (CICIDS2017), Android Adware-General Malware dataset (CICAAGM2017), and Android Malware dataset (CICAndMal2017). At this stage, the researcher extracts the data using CICFlowMeter. Moreover, Wireshark data is required in the form of the .pcap extension to perform the extraction, which will be extracted and stored as the .csv extension. There is Network>Offline on the main menu for researchers to extract data, then select the data to extract. The offline menu display can be seen in Figure 13.



**Fig 13.** Offline Menu Display

Table 4 presents the network traffic extraction results using the CICFlowMeter tool; this table shows the extraction results from the primary data.

**Table 4.** Primary Data Extraction Results

Primary data		
No	Network Traffic	Data Row
1	Normal	46
2	Bruteforce	1750

### 3.8 Labeling

Traffic record data that has been processed and converted into CSV format will be converted back into XLSX. The data obtained from the results of recording traffic on the Network still does not have a label to distinguish between normal data traffic and hacking data traffic. The researcher did the labeling manually by considering the time the simulation process was carried out. Researchers carry out separate simulations on normal traffic and also hacking traffic, so it is hoped that researchers will get traffic that matches the activity; labeling is the last step in network acquisition.

Secondary data is data obtained from existing sources. In this research, the researcher decided to use a Dataset from the IEEE Data Port (MQTT-IOT-IDS2020: MQTT Internet of Things Intrusion Detection Dataset). The dataset contains normal network traffic as well as general cyber-attacks, which have the same real-world data as PCAP and CSV formats. Researchers will use data in CSV/XLSX format per the requirements in the next process.

Furthermore, the researcher separates network traffic from attack traffic at this stage. Researchers use separation to maintain the purity of traffic data and not mix it with other data. There are 36 data taken from normal network traffic and 1751 data from attack traffic. The two traffic are then combined into one XLSX file as the primary data, which will be used as the algorithm used. As secondary research data, researchers used the MQTT-IOT-IDS2020 dataset. The researchers use several parameters from the MQTT-IOT-IDS2020 dataset, as shown in Table 5. The following research parameters.

**Table 5.** Research Parameters

No	Feature Dataset	Research Parameters
1	TotLen Bwd Pkts	TotLen Bwd Pkts
2	Bwd Pkt Len Max	Bwd Pkt Len Max
3	Bwd Pkt Len Mean	Bwd Pkt Len Mean
4	Bwd Seg Size Avg	Bwd Seg Size Avg
5	Subflow Bwd Byts	Subflow Bwd Byts
6	Init Bwd Win Byts	Init Bwd Win Byts
7	label	label

The following is the total data that will be used in the research. The secondary data is taken from the MQTT-IOT-IDS2020 dataset, which has passed the extract data phase. The total data used in the research is shown in Table 6.

**Table 6.** Total Data in Research

Data Type	Network Traffic Type	Number of Data Rows	Total (Row x Column)
Primary data	Normal	46	1796 x 7
	Bruteforce	1750	
Secondary Data	Normal	52082	85002 x 7
	Bruteforce	32920	

## 4. RESULT AND ANALYZES

### 4.1 Data Pre-processing

Before entering the learning data stage using Random Forest, the researcher pre-processes the data first; the researcher performs data rescaling using the min-max method; the goal is to make the value of each variable have a minimum and maximum value, which value will be converted into a value range between 0 to 1, the following are the results displayed in rescaling numeric data according to Figure 14 Rescaling numeric data.

```
[[6.13585854e-03 9.00439239e-02 1.20000000e-15 7.72909091e-02
 6.13585854e-03 1.00000000e+00]
[6.52236931e-03 9.00439239e-02 4.80000000e-16 7.67546218e-01
 6.52236931e-03 0.00000000e+00]
[6.42574162e-03 9.00439239e-02 1.13196000e-12 7.73538462e-01
 6.42574162e-03 0.00000000e+00]
...
[1.20784617e-03 1.83016105e-02 7.50000000e-16 7.50000000e-16
 1.20784617e-03 1.00000000e+00]
[1.15953232e-03 1.75695461e-02 7.20000000e-16 7.20000000e-16
 1.15953232e-03 1.00000000e+00]
[1.15953232e-03 1.75695461e-02 7.20000000e-16 7.20000000e-16
 1.15953232e-03 1.00000000e+00]]
```

**Fig 14.** Numerical Data Rescaling

The algorithm in the Random Forest method at this stage will be used to create a training model and test the data the researcher has obtained. The first step taken by the researcher is to make re-sampling randomize the training data in a way; this stage is called bootstrapping. Bootstrapping is made repeatedly for the specified number of trees. For this stage, the researchers used a predetermined number of trees. The bootstrapping data obtained will be used to build a collection of Random Forests. The built model will be used as the primary data row training model. Following Figure 15, the following shows an example of the Random Forest Model.



```

{'TotLen_Bwd_Pkts <= 0.0017392984829452123': ['Bruteforce',
{'Subflow_Bwd_Byts <= 0.006522369311044545': ['Normal',
'Bruteforce']}]},
{'Bwd_Seg_Size_Avg <= 9.710619600000045e-10': ['Bruteforce',
{'Bwd_Pkt_Len_Max <= 0.016105417276720352': ['Bruteforce',
'Normal']}]},
{'Subflow_Bwd_Byts <= 0.0017392984829452123': ['Bruteforce',
{'Bwd_Pkt_Len_Mean <= 1.131960000000052e-12': ['Normal',
'Bruteforce']}]},
{'TotLen_Bwd_Pkts <= 0.0017392984829452123': ['Bruteforce', 'Normal']},
{'Bwd_Pkt_Len_Max <= 0.02635431918008785': ['Bruteforce',
{'Bwd_Pkt_Len_Max <= 0.09004392386530015': ['Normal',
'Bruteforce']}]},
{'Subflow_Bwd_Byts <= 0.0017392984829452123': ['Bruteforce',
{'TotLen_Bwd_Pkts <= 0.006522369311044545': ['Normal',
'Bruteforce']}]},
{'Bwd_Seg_Size_Avg <= 9.710619600000045e-10': ['Bruteforce',
{'Bwd_Pkt_Len_Max <= 0.016105417276720352': ['Bruteforce',
'Normal']}]},
{'Subflow_Bwd_Byts <= 0.0017392984829452123': ['Bruteforce', 'Normal']},

```

**Fig.15** Sample Model Random Forest

This is also known as the prediction stage or the network traffic detection stage as a result of building a training model in the form of a collection of trees (forest). The Random Forest model created from secondary training data is used to detect testing data on secondary data. Likewise, the Random Forest Model, built from training data from primary data, is used to detect testing data from primary data. Then while the detection process is running, the examiner also gives time to be used as a reference for the duration of the process. After the detection process is complete, the researcher provides output in the form of benchmarks in the form of accuracy, precision, recall, and f-measure.

## 4.2 Model Evaluation

In this sub-chapter, the researcher will explain the process for obtaining accuracy, precision, recall, and f-measure in Random Forest to receive the output. Look at the prediction results and the accuracy of the time received when testing and training data.

The accuracy value calculates the ratio between the correct predictions made by the model and the total number of forecasts [8]. The researcher estimated the accuracy value according to Equation 4.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) \times 100\% \quad (4)$$

The Precision value calculates the ratio between correctly predicted positive samples and the total number of optimistic predictions[8]—precision calculation according to equation 5.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \times 100\% \quad (5)$$

The Recall value calculates the ratio between positive samples that are correctly predicted and the total number of positive samples[8]—calculation of recall according to equation 6.

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN}) \times 100\% \quad (6)$$

The f-measure value calculates the test's effectiveness by looking at the precision and recall values[8]. The calculation of the f-measure value is in accordance with equation 7.

$$\text{f-measure} = 2.(\text{prec. recall})/(\text{precision} + \text{recall}) \times 100\% \quad (7)$$

### 4.3 Trial Results

In the trial results stage, the researcher provides the results in accordance with the process and stages described by the examiners in the previous sub-chapter. System testing was carried out on primary data and secondary data. Table 7 presents the testing of the Random Forest method using primary testing data.

**Table 7.** Trial data testing on primary data

Data Line Number	Predictive Data	Actual Data
758	Bruteforce	Bruteforce
345	Bruteforce	Bruteforce
1645	Bruteforce	Bruteforce
1566	Bruteforce	Bruteforce
1551	Bruteforce	Bruteforce
.....	.....	.....
809	Bruteforce	Bruteforce
909	Bruteforce	Bruteforce
1782	Bruteforce	Bruteforce
1488	Bruteforce	Bruteforce
649	Bruteforce	Bruteforce
Duration : 0 second		
Accuracy : 99.55456570155901		
Precision : 100.0		
Recall : 99.54648526077098		
f-measure: 99.7727272727273		

From the test results using the Random Forest method using data testing from primary data, the researchers draw conclusions from the trial results in Table 8. The results of the primary trials are presented.

**Table 8.** Trial results on Primer

Random Forest Detection Duration	0 second
Accuracy	99,55%
Precision	100%
Recall	99.54%
f-Measure	99,77%

Furthermore, researchers conducted trials using data testing using secondary data using the Random Forest method. The following in Table 9 is presented with testing data on secondary data.

**Table 9.** Trial with testing data on secondary data

Data Line Number	Predictive Data	Actual Data
72457	Normal	Normal
36287	Normal	Normal
65354	Normal	Normal
12682	Bruteforce	Bruteforce
40409	Normal	Normal
.....	.....	.....
23579	Bruteforce	Bruteforce
4802	Bruteforce	Bruteforce
16435	Bruteforce	Bruteforce
84572	Normal	Normal
58549	Normal	Normal

Duration : 62 second

Accuracy : 99.77883393722648

Precision : 100.0

Recall : 99.43407585791691

f-measure: 99.71623498158544

The researchers draw conclusions from the results of testing using the Random Forest method using testing data from primary data from the trial results presented in Table 10.

**Table 10.** Test results on Secondary

<b>Random Forest Detection Duration</b>	<b>62 second</b>
<b>Accuracy</b>	99.77%
<b>Precision</b>	100%
<b>Recall</b>	99.43%
<b>f-Measure</b>	98.71%

#### 4.4 Discussion of trial results

After going through the testing process that the researcher has carried out, the results will be seen from testing two sets of data, namely secondary data and primary data. The researcher first displays these results in the confusion matrix table of each trial result. The results of tests conducted on primary and secondary data based on the test results did not find any False Positives (FP) because the Random Forest method can overcome the overfitting of the test data. The following are the results for each data set listed in Table 11, e.g, the confusion matrix results.

**Table 11.** Confusion matrix results

<b>Confusion Matrix on Primary Data</b>		
<b>Actual / Predictions</b>	<b>Bruteforce</b>	<b>Normal</b>
Bruteforce	439	0
Normal	2	8
<b>Confusion Matrix on Secondary Data</b>		
<b>Actual / Predictions</b>	<b>Bruteforce</b>	<b>Normal</b>
Bruteforce	8258	0
Normal	47	12946

The following is a comparison of the results of testing the two data sets, in the form of accuracy, precision, recall, f-measure, and also the duration generated using the Random Forest method using testing data from the primary and also the secondary data sets, explaining that the Random Forest algorithm can be

trusted to detect attacks on a network. The following shows accuracy, precision, recall, and f-measure, as shown in Table 12.

**Table 12.** Results of Random Forest measurements of secondary data and primary data

Parameter	Primary data	Secondary Data
Duration	0 second	62 second
Accuracy	99,55%	99.77%
Precision	100%	100%
Recall	99.54%	99.43%
f-measure	99,77%	98.71%

The emergence of errors in measuring the accuracy results in the Random Forest is due to the lack of unbalanced class representation. The Random Forest model tends to have difficulty predicting the minority class if the target data has an unstable class. This can cause errors in predicting the minority class. As the data presented by the testers above, it is known that in the primary data, there are normal data, namely 46 rows, and brute force data, namely 1750 rows. As for the secondary data, the number of normal data is 52082 lines, while the brute force data is 32920 lines. Therefore, the primary and secondary data do not have a balanced class between normal and brute force data.

## 5. CONCLUSION

Researchers have researched attack detection on the MQTT IoT protocol using the Random Forest data method, used as research material using normal network traffic data and brute force attack data. In this study, two data were used, namely primary data taken from the Hacking Environment Lab that the researchers created, and for secondary data, researchers used datasets from IEEE MQTT-IOT-IDS2020. The data was collected from expert hacking labs for research purposes in IoT security. The two data will be divided into training and testing data for each dataset. Primary data will be split into two parts, namely training data with a proportion of 75% and testing data with a proportion of 25%, while secondary data will be split into two parts, namely training data with a proportion of 75% and testing data with a proportion of 25%. There are seven parameters used in this study, namely TotLen Bwd Pkts, Bwd Pkt Len Max, Bwd Pkt Len Mean, Bwd Seg Size Avg, Subflow Bwd Byts, Init Bwd Win Byts and labels. In the training model built by the researcher, each primary and secondary data in the training data will be used as a model for data testing.

The researcher conducted trials on primary data, and the results obtained were an accuracy of 99.55%, precision of 100%, recall of 99.54%, and f-measure of 99.77%; the duration required to get these results with as many data rows as 1796 lines in only 0 seconds. As for the secondary data, the researcher obtained an accuracy of 99.77%, a precision of 100%, a recall of 99.43%, and an f-measure of 98.71%, the duration needed to get these results with 85002 data lines in just 62 seconds.

### ACKNOWLEDGMENTS

Thank you to all the teachers, lecturers, practitioners, and anyone who wants to develop an attack detection on the MQTT protocol.

### AUTHOR CONTRIBUTIONS

Conceptualization; Galuh Muhammad Iman Akbar [G.M.I.A], Mokhamad Amin Hariyadi [M.A.H], Ajib Hanani [A.H], Methodology; [G.M.I.A],[M.A.H],[A.H], validation; [G.M.I.A],[M.A.H],[A.H], formal analysis; [G.M.I.A],[M.A.H],[A.H], investigation; [G.M.I.A],[M.A.H],[A.H], data curation; [G.M.I.A],[M.A.H],[A.H], writing—original draft preparation; [G.M.I.A],[M.A.H],[A.H], writing—review and editing; [G.M.I.A],[M.A.H],[A.H], visualization; [G.M.I.A],[M.A.H],[A.H], supervision project administration; [G.M.I.A],[M.A.H],[A.H], funding acquisition; [G.M.I.A],[M.A.H],[A.H], have read and agreed to the published version of the manuscript.

### CONFLICTS OF INTEREST

The authors declare no conflict of interest.

### REFERENCES

1. B. Widagdo and M. Rofik, "Internet of Things as Engine of Economic Growth in Indonesia," 2019. [Online]. Available: <https://journal.uniku.ac.id/index.php/ijbe>
2. V. Lampkin, W. T. Leong, L. Olivera, S. Rawat, N. Subrahmanyam, and R. Xiang, "Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry," *ibm.com/redbooks First Edition, September 2012*
3. C. Patel and N. Doshi, "A Novel MQTT Security Framework in Generic IoT Model," in *Procedia Computer Science, Elsevier B.V., 2020, pp. 1399–1408. doi: 10.1016/j.procs.2020.04.150.*
4. S. Amalia, I. Deborah, and I. N. Yulita, "Comparative analysis of classification algorithm: Random Forest, SPAARC, and MLP for airlines customer satisfaction," *SINERGI, vol. 26, no. 2, p. 213, Jun. 2022, doi: 10.22441/sinergi.2022.2.010.*

5. Y. Sulistyo Nugroho and dan Nova Emiliyawati, "Sistem Klasifikasi Variabel Tingkat Penerimaan Konsumen Terhadap Mobil Menggunakan Metode Random Forest." [Online]. Available: <http://archive.ics.uci.edu/ml/>
6. A. Primajaya and B. N. Sari, "Random Forest Algorithm for Prediction of Precipitation," *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIMD)*, vol. 1, no. 1, pp. 27–31, 2018.
7. M. Hasan, M. Milon Islam, M. Ishrak Islam Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," 2019, doi 10.1016/j.iot.2019.10.
8. O. Elnakib, E. Shaaban, M. Mahmoud, and K. Emara, "EIDM: deep learning model for IoT intrusion detection systems," *Journal of Supercomputing*, pp. 1–21, Mar. 2023, doi: 10.1007/S11227-023-05197-0/TABLES/4.
9. P. D. Reddy and L. R. Parvathy, "Prediction Analysis using Random Forest Algorithms to Forecast the Air Pollution Level in a Particular Location," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022, pp. 1585-1589, doi: 10.1109/ICOSEC54921.2022.9952138.
10. R. Mittal, V. Malik, V. Singh, J. Singh, and A. Kaur, "Integrating Genetic Algorithm with Random Forest for Improving the Classification Performance of Web Log Data," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), Wanknaghat, India, 2020, pp. 177-181, doi: 10.1109/PDGC50313.2020.9315807.
11. H. Yiling and H. Shaofeng, "A Short-Term Load Forecasting Model Based on Improved Random Forest Algorithm," 2020 7th International Forum on Electrical Engineering and Automation (IFEEA), Hefei, China, 2020, pp. 928-931, doi 10.1109/IFEEA51475.2020.00195.
12. P. Mekha and N. Teeyasuksaet, "Image Classification of Rice Leaf Diseases Using Random Forest Algorithm," 2021 Joint International Conference on Digital Arts, Media, and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering, Cha-am, Thailand, 2021, pp. 165-169, doi: 10.1109/ECTIDAMTNCON51128.2021.9425696.