

Configuration Manual

MSc Research Project
Cybersecurity

Rutwik Sayankar
Student ID: x23213841

School of Computing
National College of Ireland

Supervisor: Dr. Rohit Verma

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Rutwik Arvind Sayankar
Student ID: X23213841
Programme: MSc Cybersecurity **Year:** 2024
Module: MSc Research Project
Lecturer: Dr. Rohit Verma
Submission Due Date: 12 December 2024
Project Title: **Blockchain-Based Detection and Analysis of DDoS Attacks in Decentralized Networks**
Word Count: 1559 **Page Count:** 19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Rutwik Arvind Sayankar

Date: 12 December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rutwik Sayankar
Student ID: X23213841

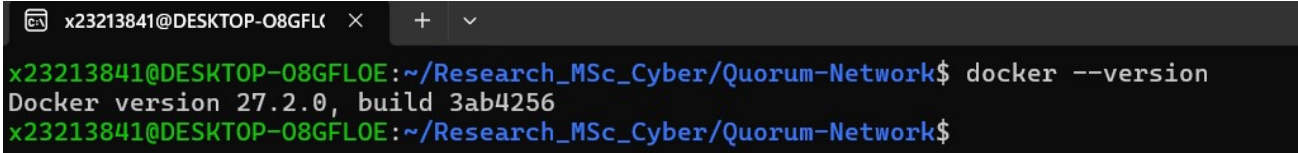
1 Introduction

This configuration manual is used as guide for setting up the basic environmental setup to implement and test a blockchain based DDoS detection system. This manual covers step by step instructions to establish solid infrastructure by using Docker, Quorum and Flask. These components are important in deployment of framework which helps to detect DDoS threats. By following the steps in manual, users can create a functional Quorum network and develop Flask application which is used as interface with this network.

2 Prerequisite Setup

1. Setting Up Docker Desktop :

- Downloaded Docker Desktop for Windows operating system(*Get Started | Docker*, 2021). As Windows OS is used, make sure to enable WSL2 (Windows Subsystem for Linux) during setup.
- After installation, confirm that Docker Desktop is operational by running the command in the terminal mentioned in figure 1.

A terminal window with a dark background. The title bar shows 'x23213841@DESKTOP-O8GFLC'. The prompt is 'x23213841@DESKTOP-O8GFLOE:~/Research_MSc_Cyber/Quorum-Network\$'. The command 'docker --version' has been executed, resulting in the output 'Docker version 27.2.0, build 3ab4256'. The prompt is now 'x23213841@DESKTOP-O8GFLOE:~/Research_MSc_Cyber/Quorum-Network\$' again.

```
x23213841@DESKTOP-O8GFLC x + v
x23213841@DESKTOP-O8GFLOE:~/Research_MSc_Cyber/Quorum-Network$ docker --version
Docker version 27.2.0, build 3ab4256
x23213841@DESKTOP-O8GFLOE:~/Research_MSc_Cyber/Quorum-Network$
```

Fig 1. Docker version and build number

2. Setup the NodeSource repository for installing Node.js version 18.x, as shown in figure 2.

```
x23213841@DESKTOP-08GFLOE:~$ cd Research_MSc_Cyber
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
[sudo] password for x23213841:
2024-11-28 11:27:21 - Installing pre-requisites
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 https://deb.nodesource.com/node_18.x nodistro InRelease
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [131 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [310 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:9 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:10 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:11 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.7 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:13 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:14 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7160 B]
Get:15 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 892 kB in 1s (1104 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.7.14build2).
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu0.5).
gnupg is already the newest version (2.4.4-2ubuntu17).
0 upgraded, 0 newly installed, 0 to remove and 36 not upgraded.
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 https://deb.nodesource.com/node_18.x nodistro InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
2024-11-28 11:27:27 - Repository configured successfully.
2024-11-28 11:27:27 - To install Node.js, run: apt-get install nodejs -y
2024-11-28 11:27:27 - You can use N|solid Runtime as a node.js alternative
2024-11-28 11:27:27 - To install N|solid Runtime, run: apt-get install nsolid -y
```

Fig 2. Installation of node.js

After installation, validate it by checking versions of Node.js, npm and npx, as shown in figure 3.

```
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber/Quorum-Network$ npx --v
10.8.2
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber/Quorum-Network$ node --version
v18.20.5
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber/Quorum-Network$ npm --version
10.8.2
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber/Quorum-Network$
```

Fig 3. Version details of node, npx and npm

3. Launch the Quorum Network by using NPX

Instead of cloning a repository, here npx command. This method allows to execute the Quorum Quickstart directly without downloading it first and minimizing manual configuration efforts.

Open terminal and execute following command to launch Quorum Quickstart, as shown in figure 4. (*Get started with Quorum Developer Quickstart | ConsenSys GoQuorum, 2023*)

```
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber$ npx quorum-dev-quickstart
```

Fig 4. Establishing Quorum setup

This command will guide through various configuration options like consensus mechanisms and setting up network.

4. Create images of quorum environment and containers by using `./run.sh` command in the folder where you installed Quorum quickstart, as shown in figure 5 & figure 6. Also check there is `.yaml` file present in that folder.

```

x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber/Quorum-Network$ ./run.sh
*****
Quorum Dev Quickstart
*****
Start network
-----
Starting network...
WARN[0000] /home/x23213841/Research_MSc_Cyber/Quorum-Network/docker-compose.yml: the attribute `version` is obsolete, it will be
o avoid potential confusion
[+] Building 5.1s (71/93)                                                                                               docker:default
=> [validator1 internal] load build definition from Dockerfile                                                       0.1s
=> => transferring dockerfile: 455B                                                                                  0.0s
=> [member3tessera internal] load build definition from Dockerfile                                                 0.1s
=> => transferring dockerfile: 570B                                                                                  0.0s
=> [validator2 internal] load build definition from Dockerfile                                                       0.1s
=> => transferring dockerfile: 455B                                                                                  0.0s
=> [rpcnode internal] load build definition from Dockerfile                                                         0.1s
=> => transferring dockerfile: 455B                                                                                  0.0s
=> [member1tessera internal] load build definition from Dockerfile                                                 0.1s
=> => transferring dockerfile: 570B                                                                                  0.0s
=> [member2tessera internal] load build definition from Dockerfile                                                 0.2s
=> => transferring dockerfile: 570B                                                                                  0.0s
=> [validator3 internal] load build definition from Dockerfile                                                       0.1s
=> => transferring dockerfile: 455B                                                                                  0.0s
=> [validator4 internal] load build definition from Dockerfile                                                       0.2s
=> => transferring dockerfile: 455B                                                                                  0.0s
=> [member3quorum internal] load metadata for docker.io/quorumengineering/quorum:23.4.0                             1.1s
=> [member3tessera internal] load metadata for docker.io/quorumengineering/tessera:23.4.0                           89.3s

```

Fig 5. Creating images of Quorum nodes in Docker

```

=> [member2quorum] resolving provenance for metadata file                                                             0.0s
WARN[0000] /home/x23213841/Research_MSc_Cyber/Quorum-Network/docker-compose.yml: the attribute `version` is obsolete, it will be
o avoid potential confusion
[+] Running 12/12
✔ Network quorum-dev-quickstart                                            Created          0.1s
✔ Volume "quorum-network-grafana"                                         Created          0.0s
✔ Volume "quorum-network-prometheus"                                      Created          0.0s
✔ Container quorum-network-validator1-1                                    Started          4.0s
✔ Container quorum-network-validator2-1                                    Started          4.7s
✔ Container rpcnode                                                        Started          4.7s
✔ Container quorum-network-member3tessera-1                               Started          3.6s
✔ Container quorum-network-grafana-1                                       Started          4.6s
✔ Container quorum-network-prometheus-1                                    Started          4.8s
✔ Container quorum-network-validator4-1                                    Started          3.8s
✔ Container quorum-network-validator3-1                                    Started          3.9s
✔ Container quorum-network-promtail-1                                       Started          3.5s
✔ Container quorum-network-loki-1                                          Started          3.6s
✔ Container quorum-network-member1tessera-1                               Started          4.5s
✔ Container quorum-network-member2tessera-1                               Started          3.6s
✔ Container quorum-network-explorer-1                                       Started          6.1s
✔ Container quorum-network-member3quorum-1                                Started          4.1s
✔ Container quorum-network-member1quorum-1                                Started          5.6s
✔ Container quorum-network-member2quorum-1                                Started          4.1s
*****
Quorum Dev Quickstart
*****
-----
List endpoints and services
-----
JSON-RPC HTTP service endpoint                                             : http://localhost:8545
JSON-RPC WebSocket service endpoint                                       : ws://localhost:8546
Web block explorer address                                                : http://localhost:25000/explorer/nodes
Prometheus address                                                         : http://localhost:9090/graph
Grafana address                                                            : http://localhost:3000/d/allVy7ycin9Yv/goquorum-overview?orgId=1&refresh=10s&fr
All
Collated logs using Grafana and Loki                                       : http://localhost:3000/d/Ak6eXLsPxFemkYKEXfch/quorum-logs-loki?orgId=1&var-app=
For more information on the endpoints and services, refer to README.md in the installation directory.
*****
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber/Quorum-Network$ node -v
v18.20.5

```

Fig 6. Docker containers are up for Quorum

5. Once setup is complete, check Docker Desktop to make sure all Quorum related containers are operational by executing command mentioned in figure 7.

```

x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber/Quorum-Network$ docker ps

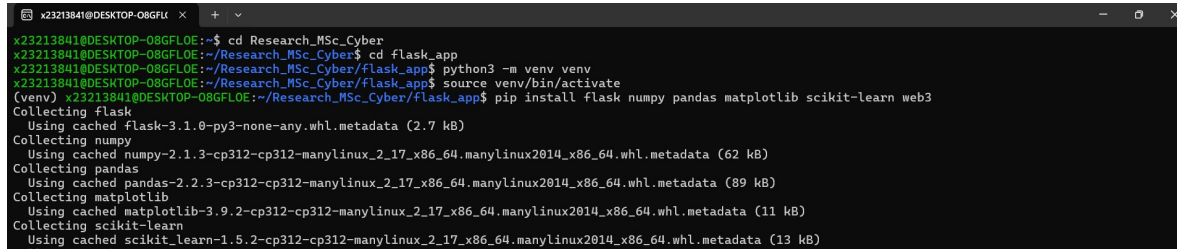
```

Fig 7. To validate container status of Quorum in Docker

This command confirms that the nodes validator nodes as well as member nodes and associated tools like Grafana, Prometheus and Loki are actively running.

3 Model Training

6. Created virtual environment and Install Flask and other required libraries, as shown in figure 8.

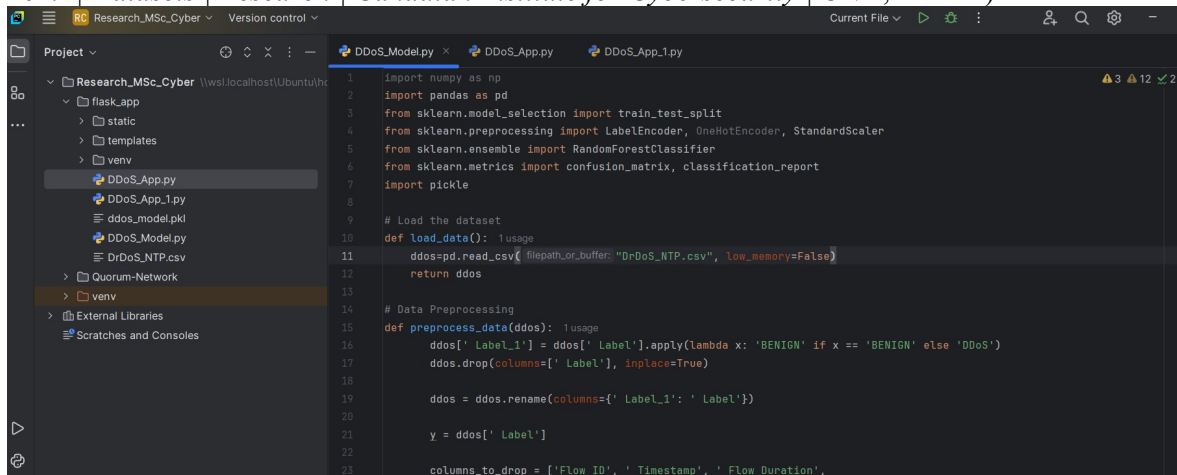


```
x23213841@DESKTOP-08GFLC ~$ cd Research_MSc_Cyber
x23213841@DESKTOP-08GFLC:~/Research_MSc_Cyber$ cd flask_app
x23213841@DESKTOP-08GFLC:~/Research_MSc_Cyber/flask_app$ python3 -m venv venv
x23213841@DESKTOP-08GFLC:~/Research_MSc_Cyber/flask_app$ source venv/bin/activate
(venv) x23213841@DESKTOP-08GFLC:~/Research_MSc_Cyber/flask_app$ pip install flask numpy pandas matplotlib scikit-learn web3
Collecting flask
  Using cached flask-3.1.0-py3-none-any.whl.metadata (2.7 kB)
Collecting numpy
  Using cached numpy-2.1.3-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (62 kB)
Collecting pandas
  Using cached pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (89 kB)
Collecting matplotlib
  Using cached matplotlib-3.9.2-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (11 kB)
Collecting scikit-learn
  Using cached scikit_learn-1.5.2-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (13 kB)
```

Fig 8. To create virtual environment for Flask App and Installation of required libraries

7. Data Preparation and Machine Learning Model Development by using Random Forest classifier, as shown in figure 9.

Here, the CIC-DDoS2019 dataset is used to generate network traffic data for analysis. (DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB, no date)



```
1 import numpy as np
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
5 from sklearn.ensemble import RandomForestClassifier
6 from sklearn.metrics import confusion_matrix, classification_report
7 import pickle
8
9 # Load the dataset
10 def load_data(): usage
11     ddos=pd.read_csv( filepath_or_buffer: "DrDoS_NTP.csv", low_memory=False)
12     return ddos
13
14 # Data Preprocessing
15 def preprocess_data(ddos): usage
16     ddos[' Label_1'] = ddos[' Label'].apply(lambda x: 'BENIGN' if x == 'BENIGN' else 'DDoS')
17     ddos.drop(columns=[' Label'], inplace=True)
18
19     ddos = ddos.rename(columns={' Label_1': ' Label'})
20
21     y = ddos[' Label']
22
23     columns_to_drop = ['Flow ID', ' Timestamp', ' Flow Duration',
```

Fig 9. Code Snippet 1 of DDoS_Model.py

For Data Preprocessing, follow the process feature selection as shown in snippet shown in figure 10.


```

# Data Preprocessing
def preprocess_data(ddos): 1 usage
    ddos['Label_1'] = ddos['Label'].apply(lambda x: 'BENIGN' if x == 'BENIGN' else 'DDoS')
    ddos.drop(columns=['Label'], inplace=True)

    ddos = ddos.rename(columns={'Label_1': 'Label'})

    y = ddos['Label']

    columns_to_drop = ['Flow ID', 'Timestamp', 'Flow Duration',
        'Fwd Packet Length Mean', 'Fwd Packet Length Std',
        'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s',
        'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
        'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std',
        'Fwd IAT Max', 'Bwd IAT Total', 'Bwd IAT Mean',
        'Bwd IAT Std', 'Bwd IAT Max', 'Fwd Packets/s',
        'Bwd Packets/s', 'Min Packet Length', 'Max Packet Length',
        'Packet Length Mean', 'Packet Length Std',
        'FIN Flag Count', 'SYN Flag Count', 'RST Flag Count',
        'PSH Flag Count', 'ACK Flag Count', 'URG Flag Count',
        'CWE Flag Count', 'ECE Flag Count',
        'Average Packet Size', 'Avg Fwd Segment Size',
        'Avg Bwd Segment Size', 'SimilarHTTP', 'Source IP', 'Destination IP', 'Unnamed: 0',
        'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Avg Bytes/Bulk',
        'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk',
        'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate']

    ddos = ddos.drop(columns = columns_to_drop).copy()

    label_encoder = LabelEncoder()

    y = label_encoder.fit_transform(y)

```

Fig 10. Code Snippet 2 of DDoS_Model.py

After training the data, save the trained model as a **pickle file** (ddos_model.pkl) using Python's pickle, as shown in figure 11.

```

# Step 4: Save the Model
def save_model(model, filename='ddos_model.pkl'): 1 usage
    with open(filename, 'wb') as file:
        pickle.dump(model, file)
    print(f"Model saved to {filename}")

```

Fig 11. Code Snippet 3 of DDoS_Model.py

4 Development of Flask Application

8. Develop Generic Frontend by using “index2.html” is provided to create a simple frontend for users to insert input traffic data and view predictions, as shown in figure 12.

```
<body>
  <h1>DDOS Attack Prediction</h1>
  <form id="prediction-form">
    <input type="text" id=" Source Port" name=" Source Port" placeholder="Source Port">
    <input type="text" id=" Destination Port" name=" Destination Port" placeholder="Destination Port">
    <input type="text" id=" Protocol" name=" Protocol" placeholder="Protocol">
    <input type="text" id=" Total Fwd Packets" name=" Total Fwd Packets" placeholder="Total Fwd Packets"><br><br>
    <input type="text" id=" Total Backward Packets" name=" Total Backward Packets" placeholder="Total Backward Packets">
    <input type="text" id="Total Length of Fwd Packets" name="Total Length of Fwd Packets" placeholder="Total Length of Fwd Packets">
    <input type="text" id=" Total Length of Bwd Packets" name=" Total Length of Bwd Packets" placeholder="Total Length of Bwd Packets">
    <input type="text" id=" Fwd Packet Length Max" name=" Fwd Packet Length Max" placeholder="Fwd Packet Length Max"><br><br>
    <input type="text" id=" Fwd Packet Length Min" name=" Fwd Packet Length Min" placeholder="Fwd Packet Length Min">
    <input type="text" id="Bwd Packet Length Max" name="Bwd Packet Length Max" placeholder="Bwd Packet Length Max">
    <input type="text" id=" Bwd Packet Length Min" name=" Bwd Packet Length Min" placeholder="Bwd Packet Length Min">
    <input type="text" id=" Flow IAT Min" name=" Flow IAT Min" placeholder="Flow IAT Min"><br><br>
    <input type="text" id=" Fwd IAT Min" name=" Fwd IAT Min" placeholder="Fwd IAT Min">
    <input type="text" id=" Bwd IAT Min" name=" Bwd IAT Min" placeholder="Bwd IAT Min">
    <input type="text" id="Fwd PSH Flags" name="Fwd PSH Flags" placeholder="Fwd PSH Flags">
    <input type="text" id=" Fwd Header Length" name=" Fwd Header Length" placeholder="Fwd Header Length"><br><br>
    <input type="text" id=" Bwd Header Length" name=" Bwd Header Length" placeholder="Bwd Header Length">
    <input type="text" id=" Packet Length Variance" name=" Packet Length Variance" placeholder="Packet Length Variance">
    <input type="text" id=" Down/Up Ratio" name=" Down/Up Ratio" placeholder="Down/Up Ratio">
    <input type="text" id=" Fwd Header Length.1" name=" Fwd Header Length.1" placeholder="Fwd Header Length.1"><br><br>
    <input type="text" id="Subflow Fwd Packets" name="Subflow Fwd Packets" placeholder="Subflow Fwd Packets">
    <input type="text" id=" Subflow Fwd Bytes" name=" Subflow Fwd Bytes" placeholder="Subflow Fwd Bytes">
    <input type="text" id=" Subflow Bwd Packets" name=" Subflow Bwd Packets" placeholder="Subflow Bwd Packets">
    <input type="text" id=" Subflow Bwd Bytes" name=" Subflow Bwd Bytes" placeholder="Subflow Bwd Bytes"><br><br>
    <input type="text" id="Init_Win_bytes_forward" name="Init_Win_bytes_forward" placeholder="Init_Win_bytes_forward">
    <input type="text" id=" Init_Win_bytes_backward" name=" Init_Win_bytes_backward" placeholder="Init_Win_bytes_backward">
    <input type="text" id=" act_data_pkt_fwd" name=" act_data_pkt_fwd" placeholder="act_data_pkt_fwd">
```

Fig 12. Code Snippet 1 of index.html

Initiated JSON script in HTML file to process a form submission dynamically without reloading the page, as shown in figure 13.

It also used to collect input data, converts it into a JSON structure, and sends it to a Flask API for prediction and display the prediction result or an error message on the webpage.


```
<script>
  async function sendData(event) {
    event.preventDefault(); // Prevent the form from refreshing the page

    const formData = {};
    const inputs = document.querySelectorAll('input');
    inputs.forEach(input => {
      const value = parseFloat(input.value.trim());
      formData[input.name.trim()] = isNaN(value) ? 0 : value; // Use 0 if the field is empty
    });

    try {
      // Log the data being sent (for debugging)
      console.log("Sending data:", formData);

      // Send JSON data to Flask API
      const response = await fetch('/predict', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ features: Object.values(formData) }) // Convert form data to an array
      });

      // Parse the JSON response
      const result = await response.json();
      if (response.ok) {
        document.getElementById('result').innerText = `Prediction: ${result.prediction}`;
      } else {
        document.getElementById('result').innerText = `Error: ${result.error}`;
      }
    }
  }

```

Fig 13 . Code Snippet 2 of index.html

9. Import libraries like Flask, Pickle, Web3 in the flask app file, as shown in figure 14.

```
1  from flask import Flask, request, jsonify, render_template, send_from_directory
2  import pickle
3  import numpy as np
4  from web3 import Web3
5  from web3.middleware.proof_of_authority import ExtraDataToPOAMiddleware # Import the middleware for PoA
6
7  # Load the trained Random Forest model
8  with open('ddos_model.pkl', 'rb') as file:
9      model = pickle.load(file)
10
11 # Initialize Flask app
12 app = Flask(__name__)
13
14 # Connect to Quorum node
15 web3 = Web3(Web3.HTTPProvider("http://localhost:8545")) # Replace with your Quorum RPC endpoint
16 if not web3.is_connected():
17     print("Failed to connect to Quorum")
18
19 # Add middleware for PoA compatibility
20 web3.middleware_onion.inject(ExtraDataToPOAMiddleware, layer=0)
21
22 # Set the default account (ensure this account exists in Quorum)
23 web3.eth.default_account = web3.eth.accounts[0]
24
25 # Define a route for the home page
26 @app.route('/')
27 def home():
28     return render_template('index2.html')
```

Fig 14. Code Snippet 1 of DDoS_App.py

Red Box:

- The code snippet mentioned in this box explains that the ExtraDataToPOAMiddleware is a middleware from Web3.py that ensures compatibility with Proof of Authority (PoA) networks.
- It preprocesses the extra data in block headers, allowing seamless interaction with PoA blockchains like Quorum.
- This is used to avoid errors related to block header parsing while working with PoA consensus.

Blue Box : To retrieve a previously saved model from a .pkl file and load it into memory for use in this app for making predictions and evaluating its performance.

Green Box : This Python code snippet uses the web3.py library to connect to a blockchain node (like Quorum node) via an HTTP provider.

10. This python code defines a Flask API endpoint (/predict) for handling POST requests, as shown in figure 15. This endpoint is used to make predictions using a machine learning model, log the prediction to a blockchain and return the result along with blockchain transaction details.

```
@app.route(rule: '/predict', methods=['POST'])
def predict():
    try:
        # Parse JSON data
        if request.is_json:
            data = request.get_json()
            features = np.array(data['features']).reshape(1, -1) # Reshape to (1, -n_features)

        # Validate that features are not empty
        if features.size == 0:
            return jsonify({"error": "No features provided in the request"}), 400

        # Make prediction
        prediction = model.predict(features)
        prediction_text = "DDoS" if prediction[0] == 1 else "Benign"

        # Log prediction to blockchain
        tx = {
            'to': web3.eth.default_account, # Send transaction to self (no recipient)
            'value': 0, # No Ether transfer
            'gas': 3000000, # Gas limit
            'data': web3.to_hex(text=f"Prediction: {prediction_text}"), # Log prediction as payload
        }
        tx_hash = web3.eth.send_transaction(tx)
        receipt = web3.eth.wait_for_transaction_receipt(tx_hash)

        return jsonify({
            "prediction": prediction_text,
            "blockchain_tx_hash": tx_hash.hex(),
            "block_number": receipt.blockNumber
        })
    except KeyError as e:
        return jsonify({"error": "Invalid data format, JSON expected"}), 400
except KeyboardInterrupt as e:
```

Fig 15. Code Snippet 2 of DDoS_App.py

White Box : Code snippet mentioned in this box verify whether the incoming request contains valid JSON data using `request.is_json`. It then parses the JSON payload into a dictionary and extracts the features array and convert it into NumPy array and reshape it (1, -n_features) format for ML model predictions.

Red Box : Code snippet mentioned in this box validates that the features array is not empty and returns a 400 Bad request error with error message if no features provided.

Blue Box : The `model.predict(features)` method uses the pretrained machine learning model to predict whether the input indicate a DDoS attack. The result is then converted into numeric predictions (e.g., 1 for DDoS, 0 for Benign) into readable string.

Green Box :

- The code mentioned in green box, defines and sends a blockchain transaction that logs the prediction result as a payload.
- This is done with no ether transfer and a specified gas limit, then waits for the transaction to be mined and returns a receipt.
- The JSON response includes the prediction result whether it is DDOS or Benign, transaction hash and block number where the transaction is recorded.

11. This python code defines a Flask API endpoint (`/get-logs`) that retrieves and returns transactions from the last 10 blocks of blockchain, filtering for transactions sent to node's default account, shown in figure 16.

```
# Endpoint to retrieve logged transactions from the blockchain
@app.route(rule: '/get-logs', methods=['GET']) 1 usage (1 dynamic)
def get_logs():
    try:
        # Fetch the latest N blocks (adjust as needed)
        block_number = web3.eth.block_number
        logs = []
        for i in range(max(0, block_number - 10), block_number + 1): # Fetch the last 10 blocks
            block = web3.eth.get_block(i, full_transactions=True)
            for tx in block.transactions:
                if tx.to == web3.eth.default_account: # Check transactions sent to the default account
                    logs.append({
                        "block_number": i,
                        "transaction_hash": tx.hash.hex(),
                        "data": web3.to_text(tx.input),
                    })
            return jsonify({"logs": logs})
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

Fig 16. Code Snippet 3 of DDoS_App.py

The code mentioned in red box, gathers relevant transaction details like block number, transaction hash and input data which is converted to text and after that it responds with JSON object containing the logs.

12. This python code defines a Flask API endpoint (/matrix) that handles GET requests and serves a file (specifically an image file) from server's static directory, shown in figure 17.

```
@app.route(rule: '/matrix', methods=['GET'])
def confusion_matrix():
    return send_from_directory(directory: 'static', path: 'confusion_matrix.png')
```

Fig 17. Code Snippet 4 of DDoS_App.py

13. Created virtual environment to install web3, this will help to integrate flask application with quorum, as shown in figure 18 & figure 19.

```
x23213841@DESKTOP-08GFLOE:~$ cd Research_MSc_Cyber
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber$ python3 -m venv venv
x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber$ source venv/bin/activate
(venv) x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber$ pip install web3
Collecting web3
  Using cached web3-7.6.0-py3-none-any.whl.metadata (5.4 kB)
Collecting eth-abi>=5.0.1 (from web3)
  Using cached eth_abi-5.1.0-py3-none-any.whl.metadata (5.1 kB)
Collecting eth-account>=0.13.1 (from web3)
  Using cached eth_account-0.13.4-py3-none-any.whl.metadata (5.3 kB)
Collecting eth-hash>=0.5.1 (from eth-hash[pycryptodome]>=0.5.1->web3)
  Using cached eth_hash-0.7.0-py3-none-any.whl.metadata (5.4 kB)
Collecting eth-typing>=5.0.0 (from web3)
  Using cached eth_typing-5.0.1-py3-none-any.whl.metadata (5.1 kB)
Collecting eth-utils>=5.0.0 (from web3)
  Using cached eth_utils-5.1.0-py3-none-any.whl.metadata (5.3 kB)
Collecting hexbytes>=1.2.0 (from web3)
  Using cached hexbytes-1.2.1-py3-none-any.whl.metadata (3.7 kB)
Collecting aiohttp>=3.7.4.post0 (from web3)
  Downloading aiohttp-3.11.8-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (7.7 kB)
Collecting pydantic>=2.4.0 (from web3)
  Downloading pydantic-2.10.2-py3-none-any.whl.metadata (170 kB)
  170.8/170.8 kB 2.6 MB/s eta 0:00:00
Collecting requests>=2.23.0 (from web3)
```

Fig 18. Installation of Web3 in quorum directory

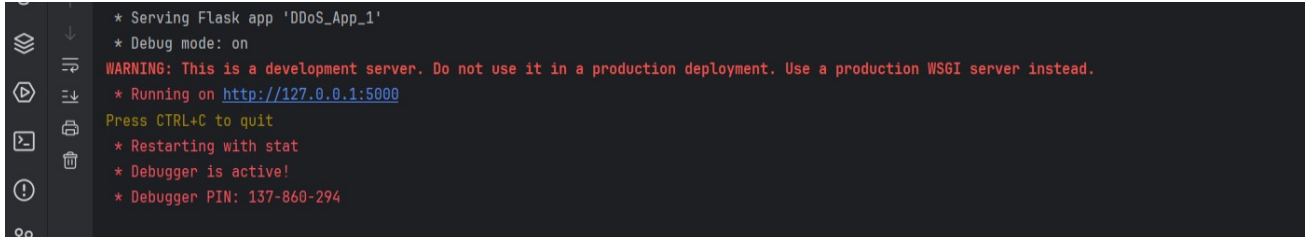
```
Using cached toolz-1.0.0-py3-none-any.whl (56 kB)
Installing collected packages: ckzg, bitarray, websockets, urllib3, typing-extensions, toolz, regex, pyunormalize, pycryptodome, propcache, multidict, idna,
hexbytes, frozendict, eth-hash, charset-normalizer, certifi, attrs, annotated-types, aiohappyeyeballs, yarl, types-requests, requests, pydantic-core, parsim
monious, eth-typing, cytoolz, aiosignal, pydantic, eth-utils, aiohttp, rlp, eth-keys, eth-abi, eth-rlp, eth-keyfile, eth-account, web3
Successfully installed aiohappyeyeballs-2.4.3 aiohttp-3.11.8 aiosignal-1.3.1 annotated-types-0.7.0 attrs-24.2.0 bitarray-3.0.0 certifi-2024.8.30 charset-nor
malizer-3.4.0 ckzg-2.0.1 cytoolz-1.0.0 eth-abi-5.1.0 eth-account-0.13.4 eth-hash-0.7.0 eth-keyfile-0.8.1 eth-keys-0.6.0 eth-rlp-2.1.0 eth-typing-5.0.1 eth-u
tils-5.1.0 frozenlist-1.5.0 hexbytes-1.2.1 idna-3.10 multidict-6.1.0 parsimonious-0.10.0 propcache-0.2.0 pycryptodome-3.21.0 pydantic-2.10.2 pydantic-core-2
.7.1 pyunormalize-16.0.0 regex-2024.11.6 requests-2.32.3 rlp-4.0.1 toolz-1.0.0 types-requests-2.32.0.20241016 typing-extensions-4.12.2 urllib3-2.2.3 web3-7
.6.0 websockets-13.1 yarl-1.18.0
(venv) x23213841@DESKTOP-08GFLOE:~/Research_MSc_Cyber$
```

Fig 19. Installation Completed

5 Workflow

14. Make sure that the containers of validator nodes, member nodes, Grafana, Loki are in running state.

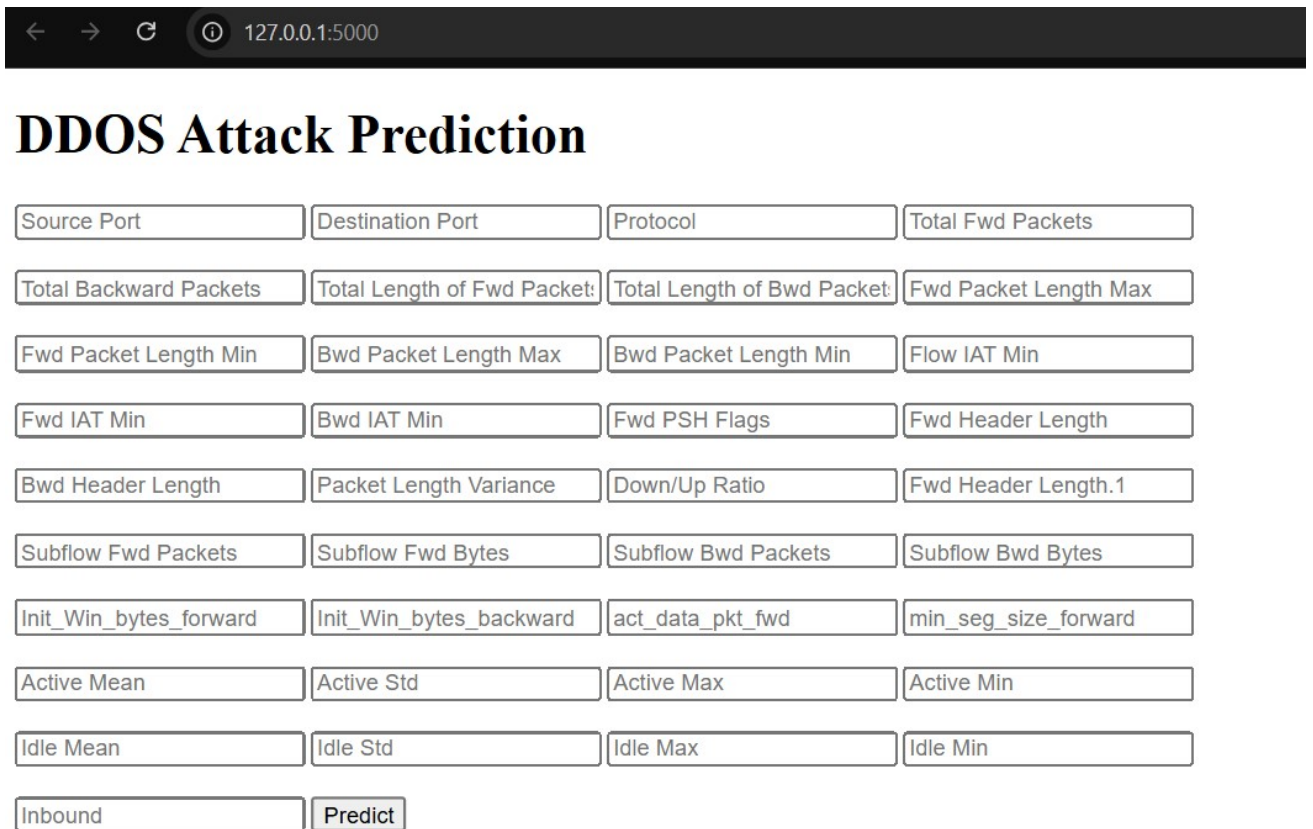
15. Run the flask app and you will get outcome mentioned in the figure, click on the localhost address to access the Flask application, as shown in figure 20.



```
* Serving Flask app 'DDoS_App_1'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 137-860-294
```

Fig 20. Terminal after running Flask Application

16. After accessing <http://127.0.0.1:5000>, you will find dashboard with features mentioned in boxes which are needed to predict the result, as shown in figure 21.



DDoS Attack Prediction

Source Port	Destination Port	Protocol	Total Fwd Packets
Total Backward Packets	Total Length of Fwd Packet:	Total Length of Bwd Packet:	Fwd Packet Length Max
Fwd Packet Length Min	Bwd Packet Length Max	Bwd Packet Length Min	Flow IAT Min
Fwd IAT Min	Bwd IAT Min	Fwd PSH Flags	Fwd Header Length
Bwd Header Length	Packet Length Variance	Down/Up Ratio	Fwd Header Length.1
Subflow Fwd Packets	Subflow Fwd Bytes	Subflow Bwd Packets	Subflow Bwd Bytes
Init_Win_bytes_forward	Init_Win_bytes_backward	act_data_pkt_fwd	min_seg_size_forward
Active Mean	Active Std	Active Max	Active Min
Idle Mean	Idle Std	Idle Max	Idle Min
Inbound	Predict		

Fig 21. Home page of Flask Application

17. For input details, I have used DDoS data mentioned in DrDoS_NTP.csv from CIC-DDoS2019, highlighted in figure 22.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W		
1	Unnamed: Flow ID	Source IP	Source Po	Destinatio	Destinatio	Protocol	Timestamp	Flow Dura	Total Fwd	Total Back	Total Leng	Total Leng	4290	509	0	160.6667	237.8782	1073	0	476.6667	565.2572	1190.988	4.022313	2	
2	0	172.16.0.5	172.16.0.5	60675	192.168.5	80	6	17:11.2	5220876	12	9	1928	4290	509	0	160.6667	237.8782	1073	0	476.6667	565.2572	1190.988	4.022313	2	
3	7	172.16.0.5	172.16.0.5	60675	192.168.5	80	6	17:11.2	12644252	5	2	0	0	0	0	0	0	0	0	0	0	0	0	0.553611	2
4	12858	192.168.5	65.55.163.	443	192.168.5	50458	6	17:12.6	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	666666.7
5	10191	192.168.5	65.55.163.	443	192.168.5	50465	6	17:13.5	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	666666.7
6	239	192.168.5	192.168.5	0	224.0.0.5	0	0	17:13.5	1.14E+08	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.454827
7	64	172.17.3	192.168.5	56085	172.217.3	80	6	17:13.9	34847273	9	8	8	0	1	0	0.888889	0.333333	0	0	0	0	0	0	0	0.229573
8	255	192.168.5	192.168.5	56131	23.194.14	443	6	17:14.1	1.2E+08	29	30	5844	876	2910	0	201.5172	641.5538	438	0	29.2	111.1242	56.07338	0.492311	2	
9	69	192.168.5	192.168.5	56123	72.21.91.2	80	6	17:14.2	35613140	9	8	8	0	1	0	0.888889	0.333333	0	0	0	0	0	0	0	0.224636
10	68	192.168.5	192.168.5	56133	8.43.72.9E	443	6	17:14.2	35550443	11	8	114	0	53	0	10.36364	21.08209	0	0	0	0	0	0	0	3.206711
11	5	172.16.0.5	172.16.0.5	53948	192.168.5	80	6	17:16.2	5008872	8	7	754	2146	377	0	94.25	174.5171	1073	0	306.5714	523.5704	578.9727	2.994686	3	
12	1	172.16.0.5	192.168.5	80	172.16.0.5	60675	6	17:16.4	176	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	14296	172.16.0.5	172.16.0.5	60675	192.168.5	80	6	17:16.4	95	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	242	192.168.5	192.168.5	0	224.0.0.5	0	0	17:16.8	1.12E+08	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	13987	192.168.5	65.55.163.	443	192.168.5	50466	6	17:17.4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000000
16	11917	189.168.5	65.55.163.	443	189.168.5	50467	6	17:18.0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000000

Fig 22. DrDoS_NTP.csv file data

Here is the input dashboard after putting all required feature values in the dedicated boxes, as shown in figure 23.

← → ↻
127.0.0.1:5000

DDoS Attack Prediction

60675	80	6	0
2	0	0	0
0	0	0	3
0	3	0	32
64	0	2	32
0	0	2	0
64	243	0	32
0	0	0	0
0	0	0	0
1	Predict		

Prediction: DDoS

Fig 23. DDoS data inputs in Flask Application

18. After clicking predict button, it will take 1 to 2 minutes to predict that data is DDoS or Benign. We also get dedicated log for the predicted result as shown in figure 24. Here we get block number, prediction of data and transaction hash for our result.

```

127.0.0.1:5000/get-logs
Pretty-print [ ]
{
  "logs": [
    {
      "block_number": 3073,
      "data": "Prediction: DDoS",
      "transaction_hash": "6a69288ded0ddefe88a64bbe932d78763bd6c2576f4fb2f960fa801ac2486b59"
    }
  ]
}

```

Fig 24. Output logs in /get-logs endpoint

After DDoS traffic data send to the Quorum through Flask Application.

19. The visualizations of traffic data is displayed over the **Grafana** and Logging data is displayed over the **Loki**.

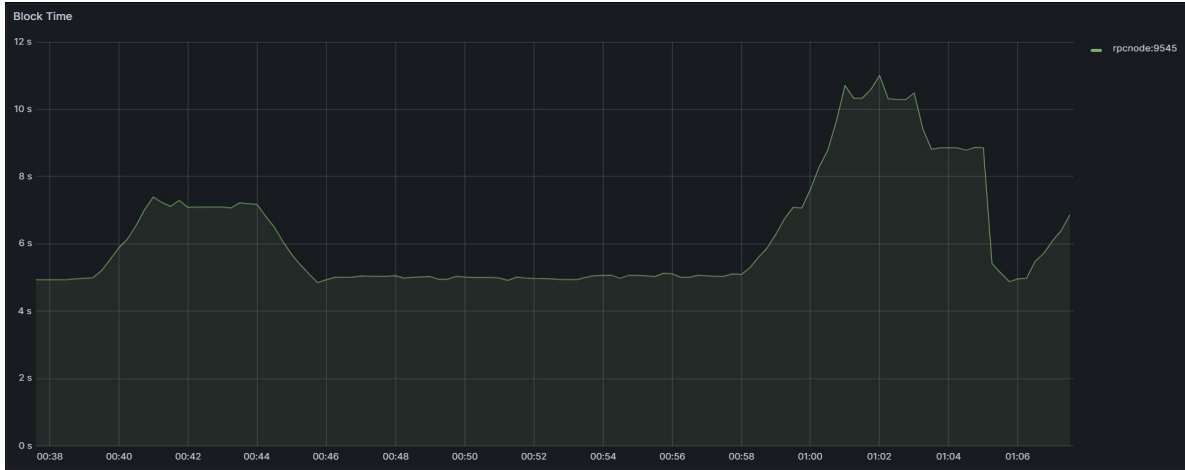


Fig 25. Variations in Block time transaction graph of rpcnode.



Fig 26. Variations in Network Traffic Graph of rpcnode.

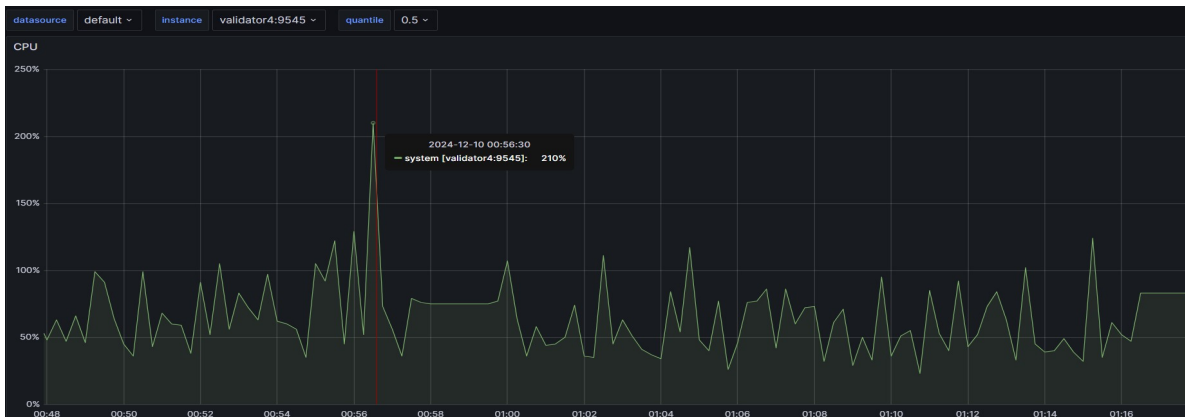


Fig 27. Variations in CPU Usage Graph of validator4 node.

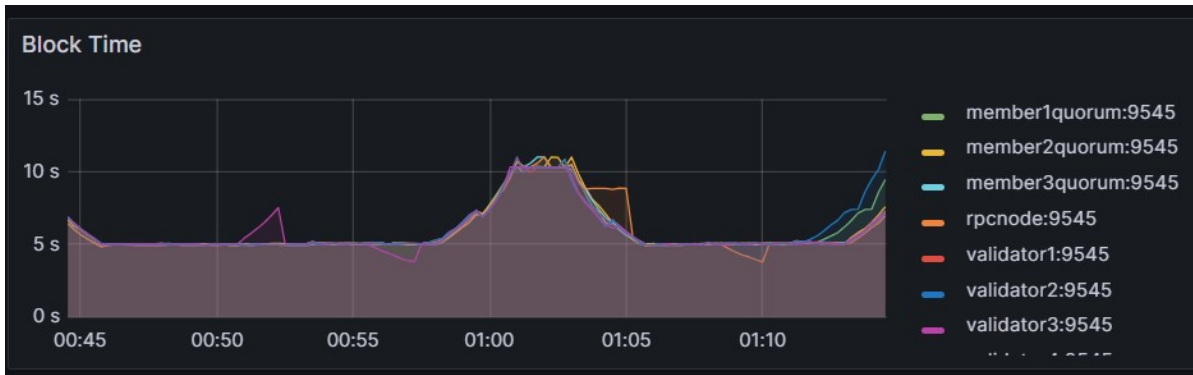


Fig 28. Variations in Block time Graph of all nodes.



Fig 29. Variations in Transaction propagation Graph of all nodes.

```

> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.058] QBFT: handle final committed address=0x13A52aa8092e1322e08b52506276363d4754c122e current.round=0 current.sequence=3073 state="Accept request"
> 2024-12-10 01:02:22.571 INFO [12-10]01:02:22.500] QBFT: PRE-PREPARE block proposal is in the future (will be treated again later) address=0x93917cAD8ace5dFCE132B991732c6Cda9bcC588a current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:22.571 INFO [12-10]01:02:22.499] QBFT: handle block proposal request address=0x93917cAD8ace5dFCE132B991732c6Cda9bcC588a current.round=0 current.sequence=3073 state="Accept request"
> 2024-12-10 01:02:22.571 INFO [12-10]01:02:22.500] QBFT: PRE-PREPARE block proposal is in the future (will be treated again later) address=0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18 current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:22.571 INFO [12-10]01:02:22.500] QBFT: handle PRE-PREPARE message address=0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18 current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:22.571 INFO [12-10]01:02:22.499] QBFT: handle block proposal request address=0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18 current.round=0 current.sequence=3073 state="Accept request"

Fields
├ INFO
├ QBFT_
├ _12_10_01_02_22_499_
├ address
│   0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18
├ block
├ current_round
│   0
├ current_sequence
│   3073
├ filename
│   /var/log/quorum/geth-172.16.239.12.log
├ handle
├ job
│   quorum
├ proposal
├ request
├ state
│   Accept request

```

Fig 30. Log Details - 1 of block where DDoS data stored.

```

> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.063] Commit new mining work number=3074 sealhash=9938c7.2ff4d7 uncles=0 txs=0 gas=0 fees=0 elapsed="865.92µs"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.047] Imported new chain segment blocks=1 txs=1 mgas=0.021 elapsed=8.075ms mgasps=2.632 number=3073 hash=f1bb4b...d2a087 dirty=1.27KiB
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.062] QBFT: start new round address=0x4818DFcd9221CD790E7cd66EeCE308eD4D485381 old.round=0 old.sequence=3073 old.state="Accept request"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.062] QBFT: initialize new round address=0x4818DFcd9221CD790E7cd66EeCE308eD4D485381 current.round=0 current.sequence=3073 target.round=0 last.author=0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18 hash=f1bb4b...d2a087 number=3073
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.033] BFT: block proposal committed address=0x98C1334496614aED49d2E81526D089F7264FED9C current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.013] QBFT: handle PRE-PREPARE message address=0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18 current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.032] QBFT: handle COMMIT message address=0x98C1334496614aED49d2E81526D089F7264FED9C current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.004] QBFT: broadcast PRE-PREPARE message address=0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18 current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.002] QBFT: accepted PRE-PREPARE message address=0x27A97C9AaF04f18F3014c32e036dD0Ac76Da5f18 current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.054] QBFT: start new round address=0xcE412f988377e31F4d0ff12d74d773B51C42d0Ac old.round=0 old.sequence=3073 old.state="Committed"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.029] QBFT: handle COMMIT message address=0xcE412f988377e31F4d0ff12d74d773B51C42d0Ac current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.062] QBFT: handle final committed address=0x4818DFcd9221CD790E7cd66EeCE308eD4D485381 current.round=0 current.sequence=3073 state="Accept request"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.030] QBFT: handle COMMIT message address=0x93917cAD8ace5dFCE132B991732c6Cda9bcC588a current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.020] QBFT: handle PRE-PREPARE message address=0x93917cAD8ace5dFCE132B991732c6Cda9bcC588a current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.013] QBFT: handle PRE-PREPARE message address=0x93917cAD8ace5dFCE132B991732c6Cda9bcC588a current.round=0 current.sequence=3073 state="Prepared"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.010] QBFT: broadcast PRE-PREPARE message address=0x93917cAD8ace5dFCE132B991732c6Cda9bcC588a current.round=0 current.sequence=3073 state="Prepared"

```

Fig 31. Log Details - 2 of block where DDoS data stored.

```

> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.033] BFT: block proposal committed author=0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18 hash=f1bb4b...d2a087 number=3073
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.032] QBFT: received quorum of COMMIT messages address=0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18 current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.013] QBFT: handle PREPARE message address=0x98C1334496614aED49d2E81526D0889f7264fED9C current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.001] QBFT: handle PRE-PREPARE message address=0x98C1334496614aED49d2E81526D0889f7264fED9C current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.032] QBFT: handle COMMIT message address=0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18 current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.004] QBFT: broadcast PREPARE message address=0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18 current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.002] QBFT: accepted PRE-PREPARE message address=0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18 current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.054] QBFT: start new round address=0xcE412f988377e31fAd0ff12d74d7f3B51C42d0cA old.round=0 old.sequence=3073 old.state=Committed
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.029] QBFT: handle COMMIT message address=0xcE412f988377e31fAd0ff12d74d7f3B51C42d0cA current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.062] QBFT: handle final committed address=0x4018DFcd9221CD798E7cd66EecE383eD4D4853B1 current.round=0 current.sequence=3073 state=Accept request

Fields
└─ INFO
└─ QBFT_
└─ _12_10_01_02_23_062_
└─ address 0x4018DFcd9221CD798E7cd66EecE383eD4D4853B1
└─ committed
└─ current_round 0
└─ current_sequence 3073
└─ filename /var/log/quorum/geth-172.16.239.18.log
└─ final
└─ handle
└─ job quorum
└─ state Accept request

> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.030] QBFT: handle COMMIT message address=0x93917cAD8ace5DFCE1328991732c6Cda9bcC58Ba current.round=0 current.sequence=3073 state=Prepared

```

Fig 32. Log Details – 3 of block where DDoS data stored.

```

> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.063] Commit new mining work numbers=3074 sealhash=9938c7...2ff4d7 uncles=0 txs=0 gas=0 fees=0 elapsed="865.92µs"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.047] Imported new chain segment blocks=1 txs=1 mgas=0.021 elapsed=8.075ms mgasps=2.632 number=3073 hash=f1bb4b...d2a087 dirty=1.27KiB
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.062] QBFT: start new round address=0x4018DFcd9221CD798E7cd66EecE383eD4D4853B1 old.round=0 old.sequence=3073 old.state="Accept request"
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.062] QBFT: initialize new round address=0x4018DFcd9221CD798E7cd66EecE383eD4D4853B1 current.round=0 current.sequence=3073 target.round=0 last
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.033] BFT: block proposal committed author=0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18 hash=f1bb4b...d2a087 number=3073

Fields
└─ BFT_
└─ INFO
└─ _12_10_01_02_23_033_
└─ author 0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18
└─ block
└─ committed
└─ filename /var/log/quorum/geth-172.16.239.12.log
└─ hash f1bb4b...d2a087
└─ job quorum
└─ number 3073
└─ proposal

> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.032] QBFT: received quorum of COMMIT messages address=0x27A97C9AaF04f18f3014c32e036d08Ac76Da5f18 current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.013] QBFT: handle PREPARE message address=0x98C1334496614aED49d2E81526D0889f7264fED9C current.round=0 current.sequence=3073 state=Prepared
> 2024-12-10 01:02:23.071 INFO [12-10]01:02:23.001] QBFT: handle PRE-PREPARE message address=0x98C1334496614aED49d2E81526D0889f7264fED9C current.round=0 current.sequence=3073 state=Prepared

```

Fig 33. Log Details-4 of block where DDoS data stored.

After Random traffic data send to the Quorum through Flask Application.

DDOS Attack Prediction

1	11	11	11
1	11	11	1
1	1	1	1
1	11	11	11
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

1

Prediction: Benign

Fig 34. Random / Benign data inputs in Flask Application

```
{
  "logs": [
    {
      "block_number": 3620,
      "data": "Prediction: Benign",
      "transaction_hash": "70c50dfbf5493dc5cd135e88a0798fe51ce393c1d3e21b2a52f3b81d85e90552"
    }
  ]
}
```

Fig 35. Output logs-2 in /get-logs endpoint



Fig 36. Variations in Block time transaction graph of rpcnode.

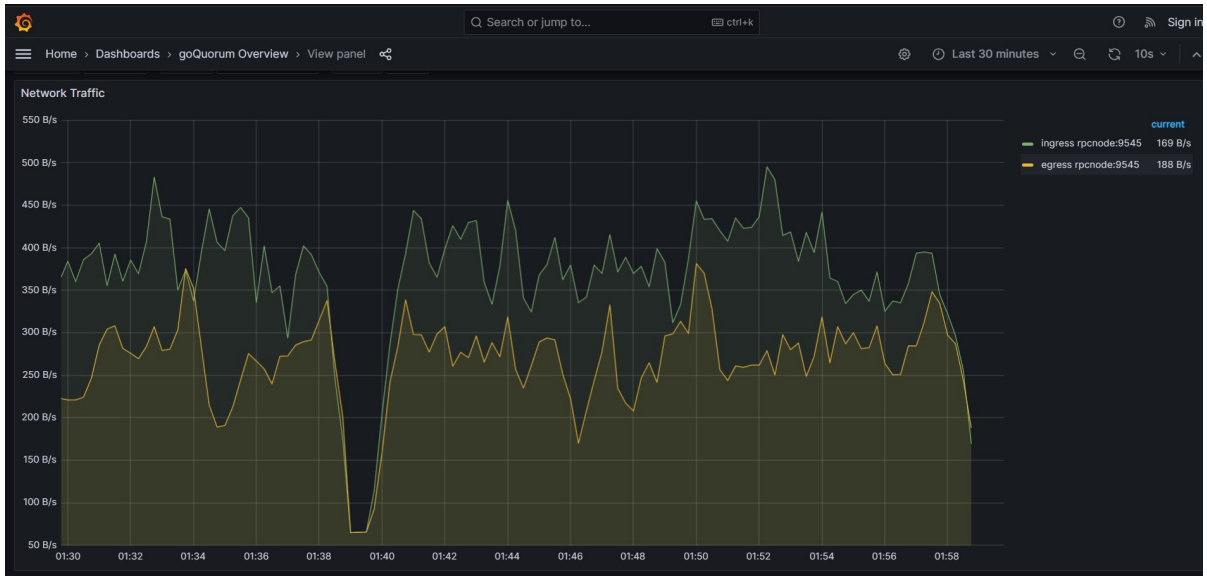


Fig 37. Variations in Network Traffic graph of rpcnode.

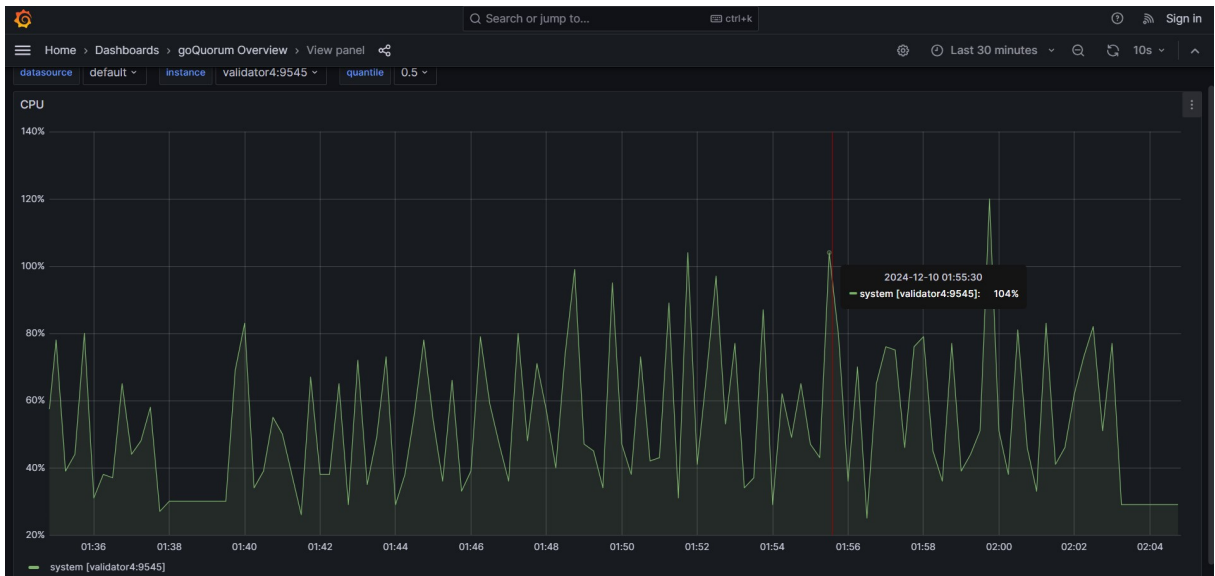


Fig 38. Variations in CPU Usage graph of rpcnode.

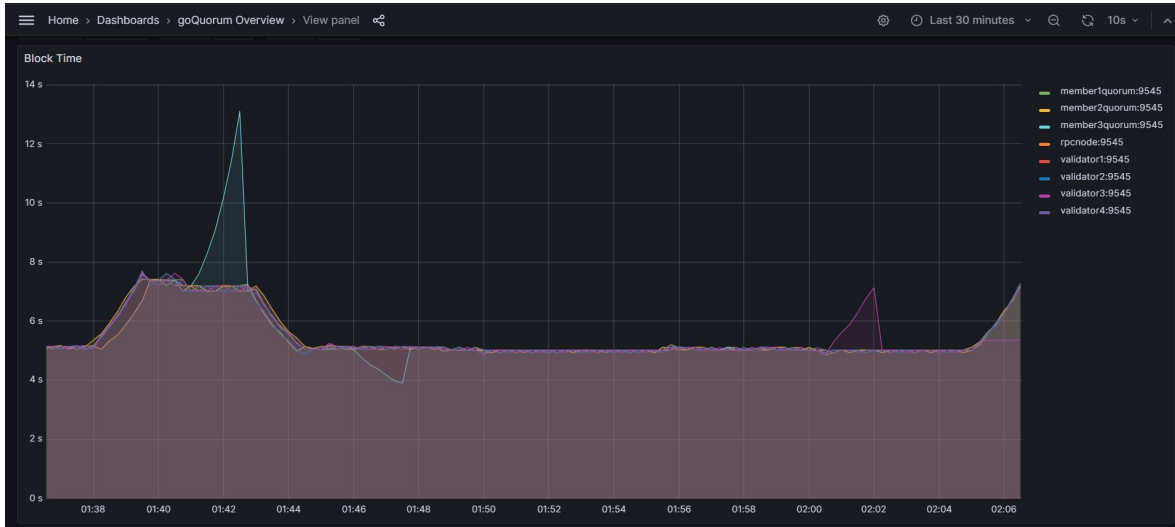


Fig 39. Variations in Block time transaction graph of all nodes.



Fig 40. Variations in Transaction propagation graph of all nodes.

```

> 2024-12-10 01:49:50.742 INFO [12-10]01:49:50.499] QBFT: PRE-PREPARE block proposal is in the future (will be treated again later) address=0xcE412f988377e31F4d8ff12d74df73B51C42d8cA current.round=0 cu
> 2024-12-10 01:49:50.742 INFO [12-10]01:49:50.495] QBFT: handle block proposal request address=0xcE412f988377e31F4d8ff12d74df73B51C42d8cA current.round=0 current.sequence=3628 state="Accept reque
> 2024-12-10 01:49:50.742 INFO [12-10]01:49:50.499] QBFT: PRE-PREPARE block proposal is in the future (will be treated again later) address=0x98C1334496614aED49d2E81526D889F7264FED9C current.round=0 cu
> 2024-12-10 01:49:50.742 INFO [12-10]01:49:50.499] QBFT: handle PRE-PREPARE message address=0x98C1334496614aED49d2E81526D889F7264FED9C
> 2024-12-10 01:49:50.742 INFO [12-10]01:49:50.499] QBFT: PRE-PREPARE block proposal is in the future (will be treated again later) address=0x93917cADBace5dFCE132B991732c6da9bcC588a current.round=0 cu
> 2024-12-10 01:49:50.742 INFO [12-10]01:49:50.496] QBFT: handle block proposal request address=0x98C1334496614aED49d2E81526D889F7264FED9C current.round=0 current.sequence=3628 state="Accept reque
> 2024-12-10 01:49:46.226 INFO [12-10]01:49:46.826] Commit new mining work number=3628 sealhash=889c52.773734 uncles=0 txs=1 gas=21288 fees=0 elapsed=16.361ms
> 2024-12-10 01:49:46.226 INFO [12-10]01:49:46.823] QUORUM-CHECKPOINT name=TX-COMPLETED tx=0x78c58dfbf5493dc5cd135e88a0798fe51ce393c1d3e21b2a52f3b81d85e98552 time=11.23826ms

Fields
┆ INFO
┆ QUORUM_CHECKPOINT
┆ _12_10_01_49_46_823_
┆ filename /var/log/quorum/geth-172.16.239.13.log
┆ job quorum
┆ name TX-COMPLETED
┆ time 11.23826ms
┆ tx 0x78c58dfbf5493dc5cd135e88a0798fe51ce393c1d3e21b2a52f3b81d85e98552

```

Fig 41. Log Details - 1 of block where Benign data stored.

```
2024-12-10 01:49:51.244 INFO [12-10|01:49:51.015] mined potential block number=3620 hash=274af4..7b4377

Fields
├ INFO
├ -
├ _12_10_01_49_51_015_
├ block
├ filename /var/log/quorum/geth-172.16.239.12.log
├ hash 274af4..7b4377
├ job quorum
├ mined
├ number 3620
├ potential

> 2024-12-10 01:49:51.244 INFO [12-10|01:49:51.008] BFT: block proposal committed author=0x98c1324496614aED49d2E81526D889f7264fED90 hash=274af4..7b4377 number=3620
> 2024-12-10 01:49:51.244 INFO [12-10|01:49:51.005] QBFT: received quorum of PREPARE messages address=0x93917cADbace5DFCE132B991732c6Cda9bcC5B8a current.round=0 current.sequence=3620 state=Prepared
> 2024-12-10 01:49:51.244 INFO [12-10|01:49:51.026] QBFT: initialize new round address=0xDF8b560be949c229c821731554c33EAD5E3888A4 current.round=0 current.sequence=3620 target.round=0 last
```

Fig 42. Log Details - 2 of block where Benign data stored.

References

DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB (no date). Available at: <https://www.unb.ca/cic/datasets/ddos-2019.html> (Accessed: 12 December 2024).

Get Started | Docker (2021). Available at: <https://www.docker.com/get-started/> (Accessed: 12 December 2024).

Get started with Quorum Developer Quickstart | ConsenSys GoQuorum (2023). Available at: <https://docs.goquorum.consenSys.io/tutorials/quorum-dev-quickstart> (Accessed: 12 December 2024).