National College of Ireland

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)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
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.	□

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.

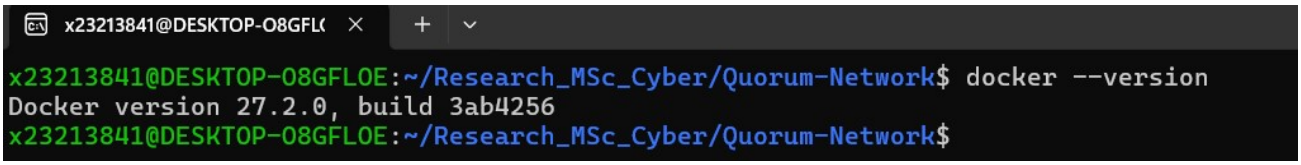


Fig 1. Docker version and build number

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

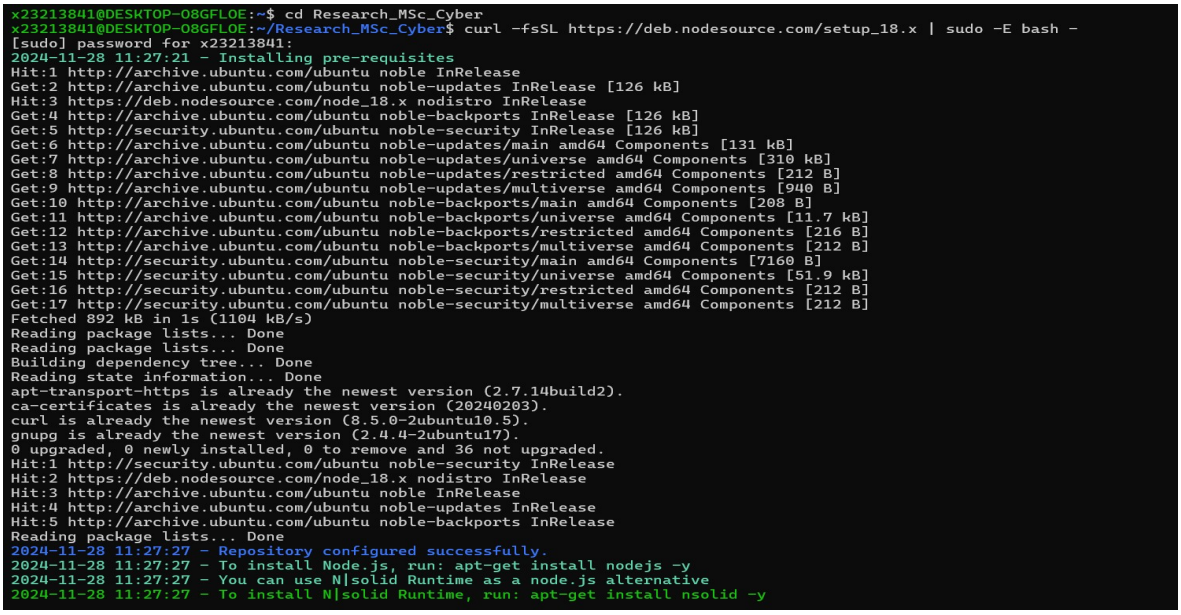


Fig 2. Installation of node.js

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

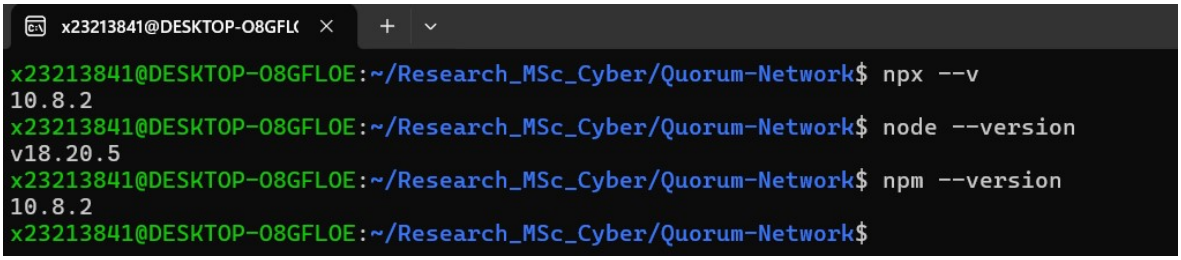


Fig 3. Version details of node, npx and npm

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)

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.

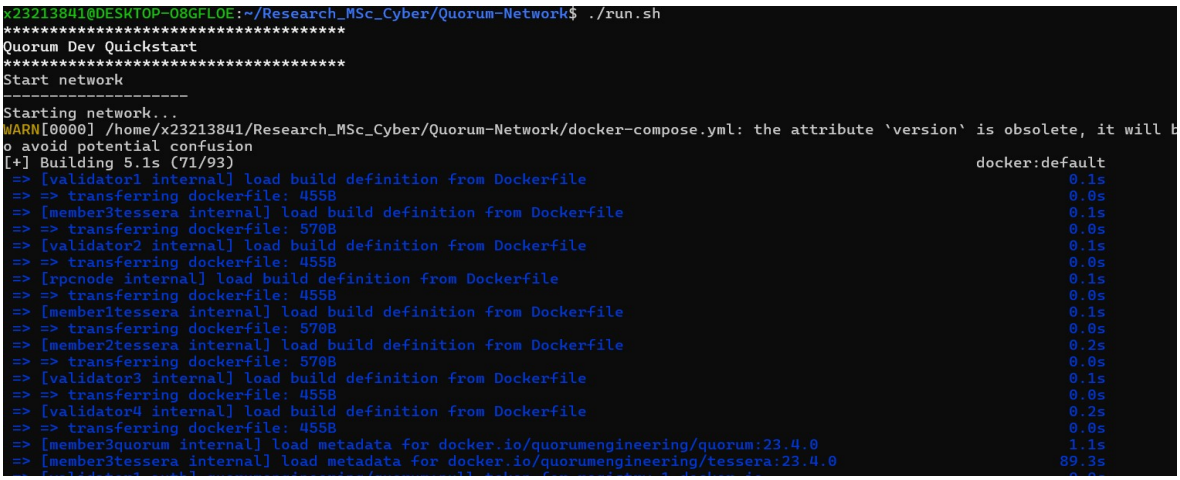


Fig 5. Creating images of Quorum nodes in Docker

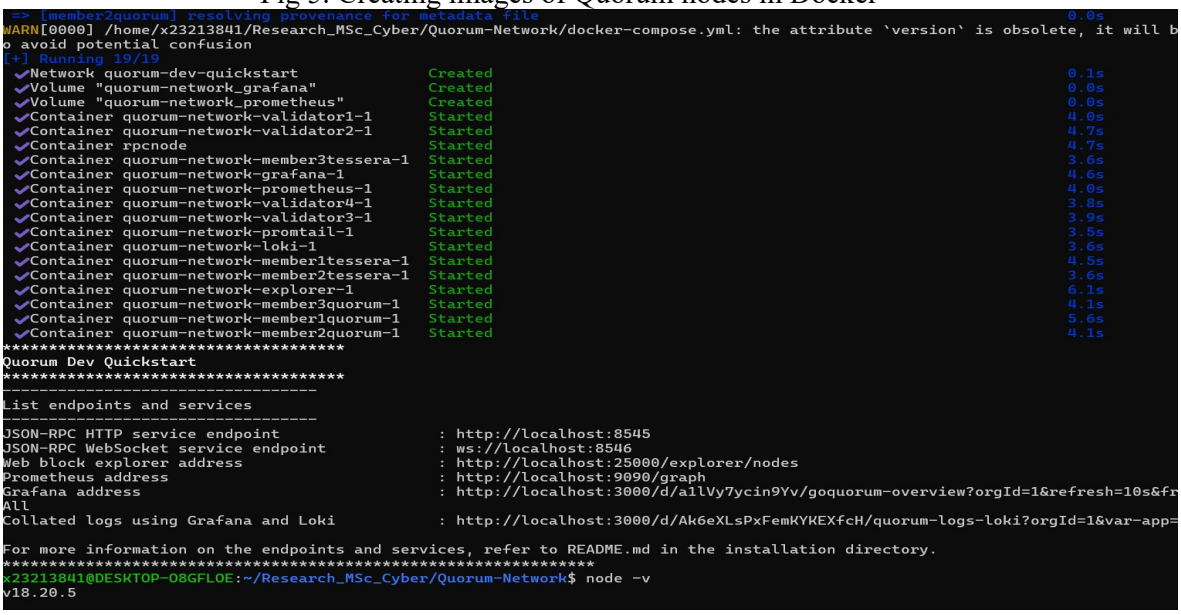


Fig 6. Docker containers are up for Quorum

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

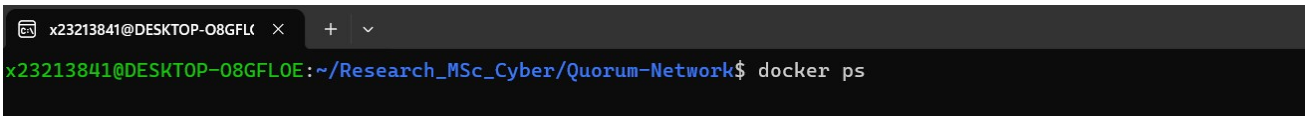


Fig 7. To validate container status of Quorum in Docker

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.

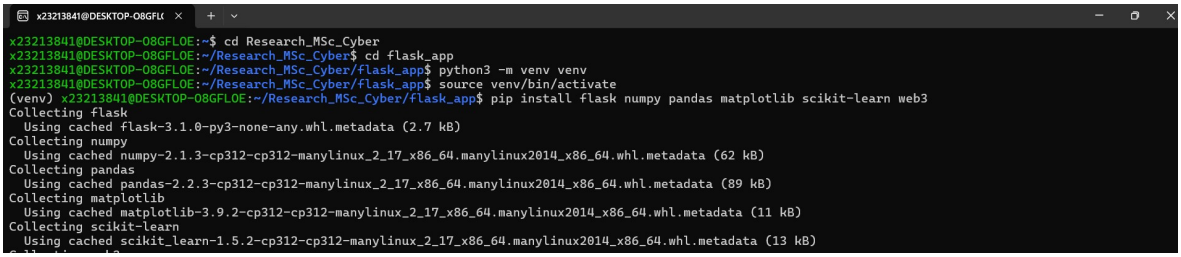


Fig 8. To create virtual environment for Flask App and Installation of required libraries

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)

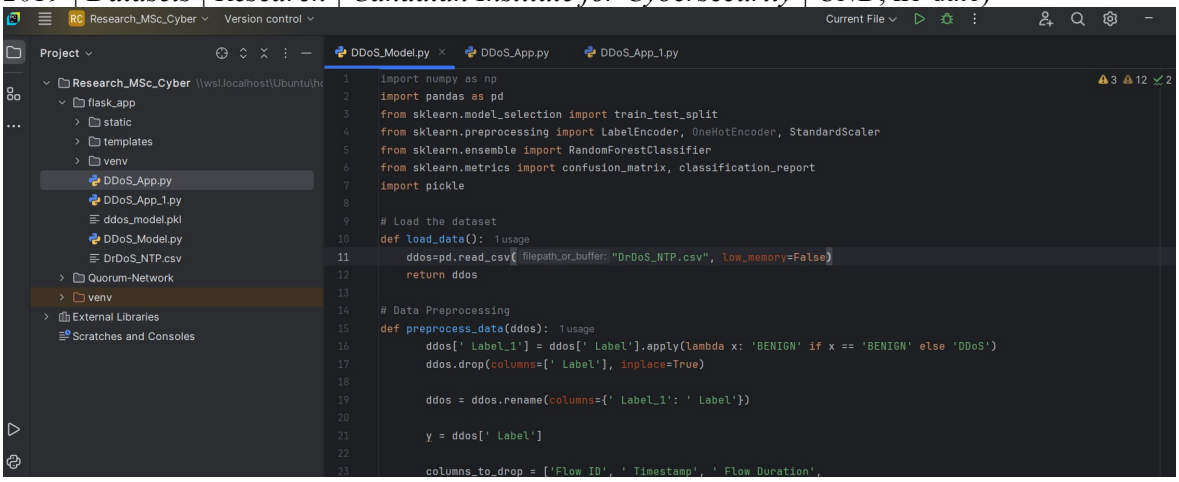


Fig 9. Code Snippet 1 of DDoS_Model.py

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

```python
# 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', 'SimillarHTTP', ' 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.

```python
# 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.

```
from flask import Flask, request, jsonify, render_template, send_from_directory
import pickle
import numpy as np
from web3 import Web3
from web3.middleware.proof_of_authority import ExtraDataToPOAMiddleware  # Import the middleware for PoA

# Load the trained Random Forest model
with open('ddos_model.pkl', 'rb') as file:
    model = pickle.load(file)

# Initialize Flask app
app = Flask(__name__)

# Connect to Quorum node
web3 = Web3(Web3.HTTPProvider("http://localhost:8545"))  # Replace with your Quorum RPC endpoint
if not web3.is_connected():
    print("Failed to connect to Quorum")

# Add middleware for PoA compatibility
web3.middleware_onion.inject(ExtraDataToPOAMiddleware, layer=0)

# Set the default account (ensure this account exists in Quorum)
web3.eth.default_account = web3.eth.accounts[0]

# Define a route for the home page
@app.route('/')
def home():
    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.

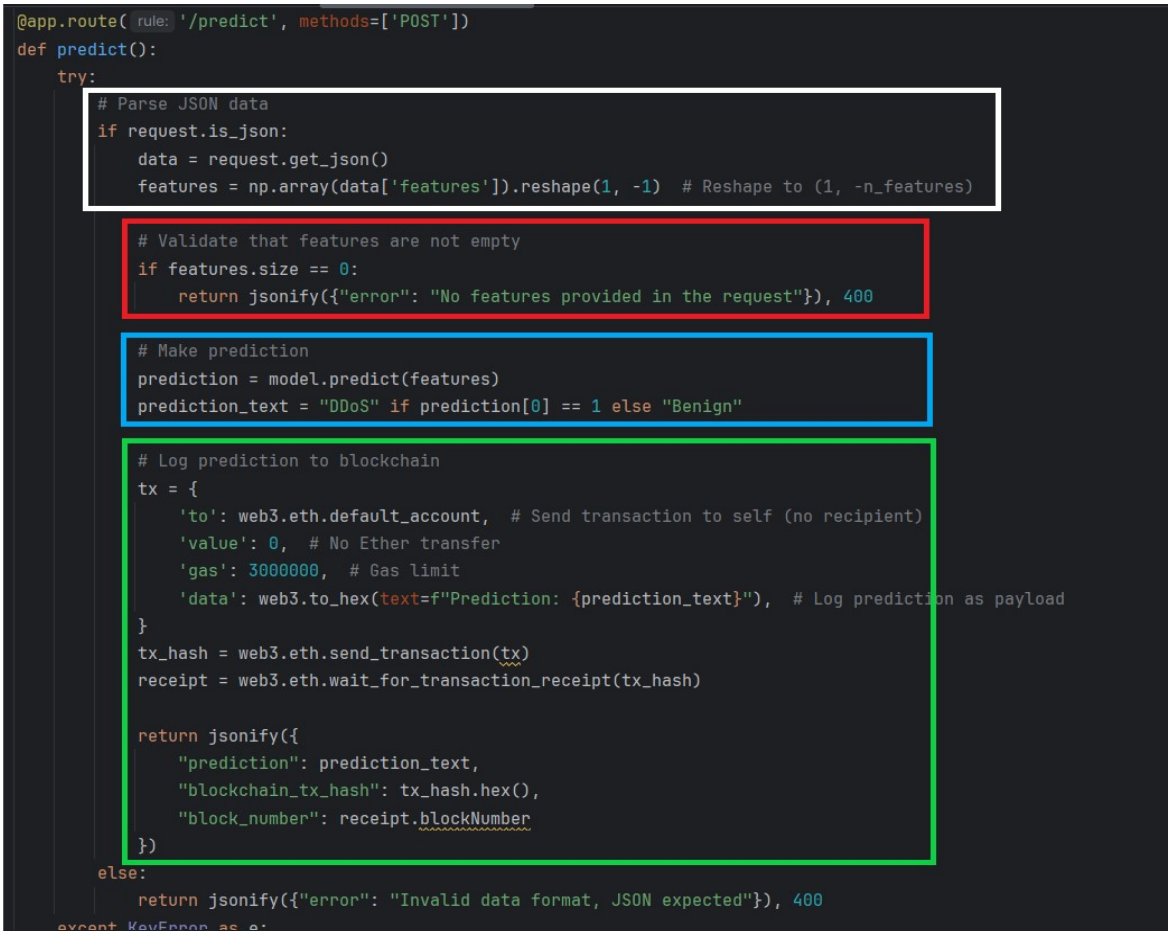


Fig 15. Code Snippet 2 of DDoS_App.py

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-O8GFLOE:~$ cd Research_MSc_Cyber
x23213841@DESKTOP-O8GFLOE:~/Research_MSc_Cyber$ python3 -m venv venv
x23213841@DESKTOP-O8GFLOE:~/Research_MSc_Cyber$ source venv/bin/activate
(venv) x23213841@DESKTOP-O8GFLOE:~/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, frozenlist, eth-hash, charset-normalizer, certifi, attrs, annotated-types, aiohappyeyeballs, yarl, types-requests, requests, pydantic-core, parsi
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
.27.1 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-O8GFLOE:~/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.

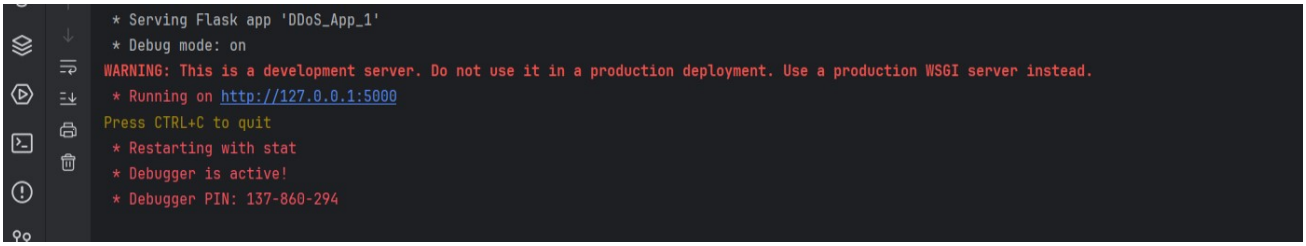


Fig 20. Terminal after running Flask Application

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.

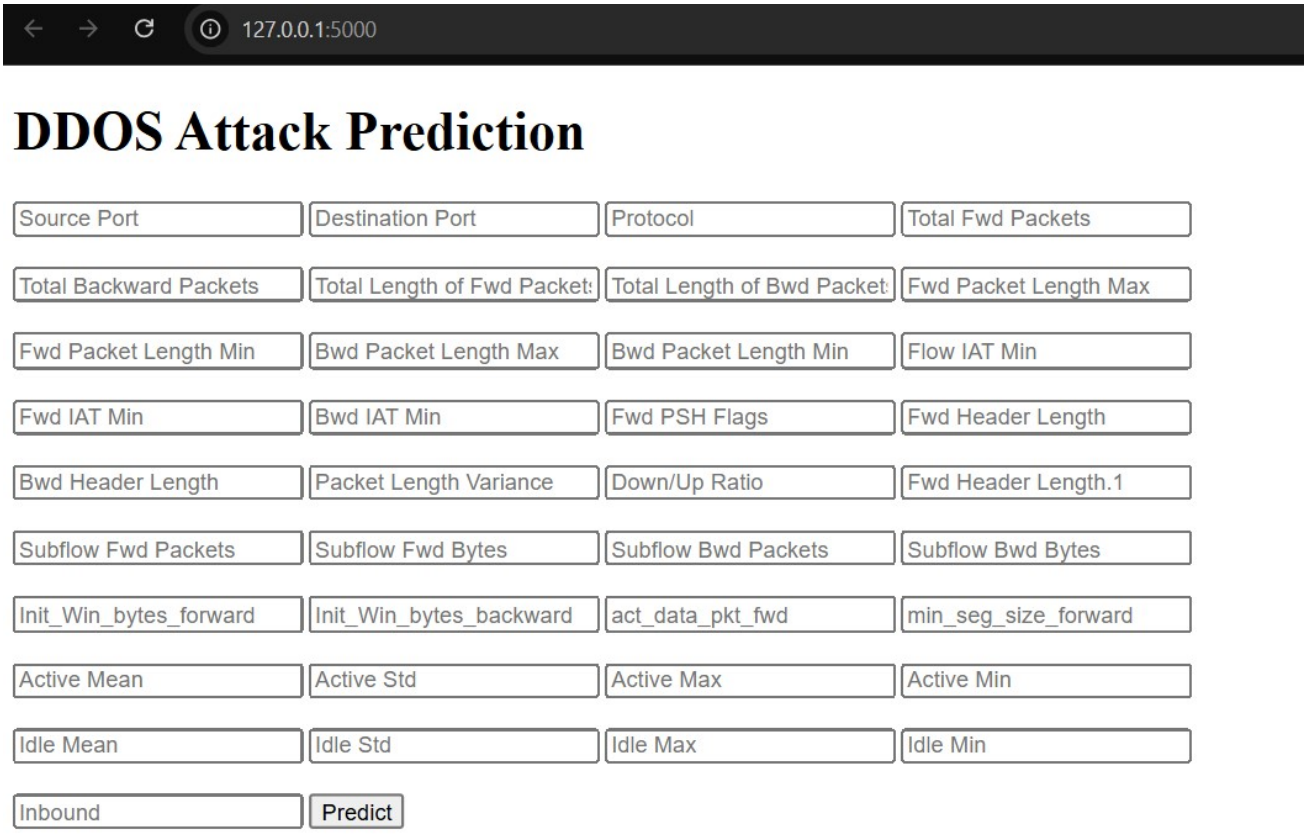
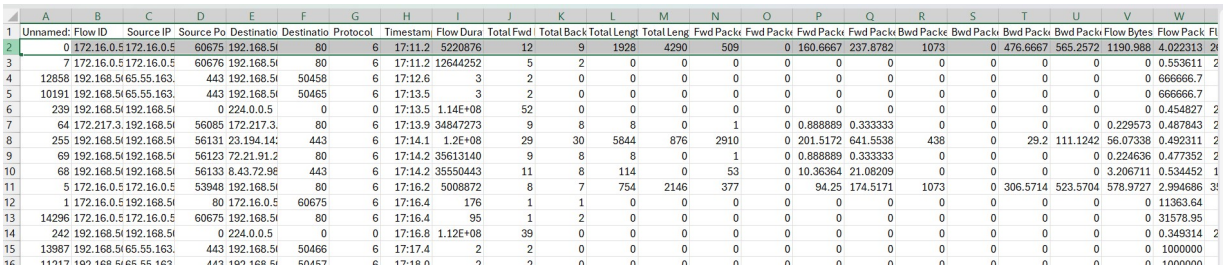


Fig 21. Home page of Flask Application

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	Timestam	Flow Dura	Total Fwd	Total Back	Total Lengt	Total Leng	Fwd Packe	Fwd Packe	Fwd Packe	Fwd Packe	Bwd Packe	Bwd Packe	Bwd Packe	Bwd Packe	Flow Bytes	Flow Pack
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
3	7	172.16.0.5	172.16.0.5	60676	192.168.5(	80	6	17:11.2	12644252	5	2	0	0	0	0	0	0	0	0	0	0	0	0.553611
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	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	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	0.454827
7	64	172.217.3.	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	0.229573	0.487843
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
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	0.224636	0.477352
10	68	192.168.5(	192.168.5(	56133	8.43.72.98	443	6	17:14.2	35550443	11	8	114	0	53	0	10.36364	21.08209	0	0	0	0	3.206711	0.534452
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
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	11363.64
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	31578.95
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	0.349314
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	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.

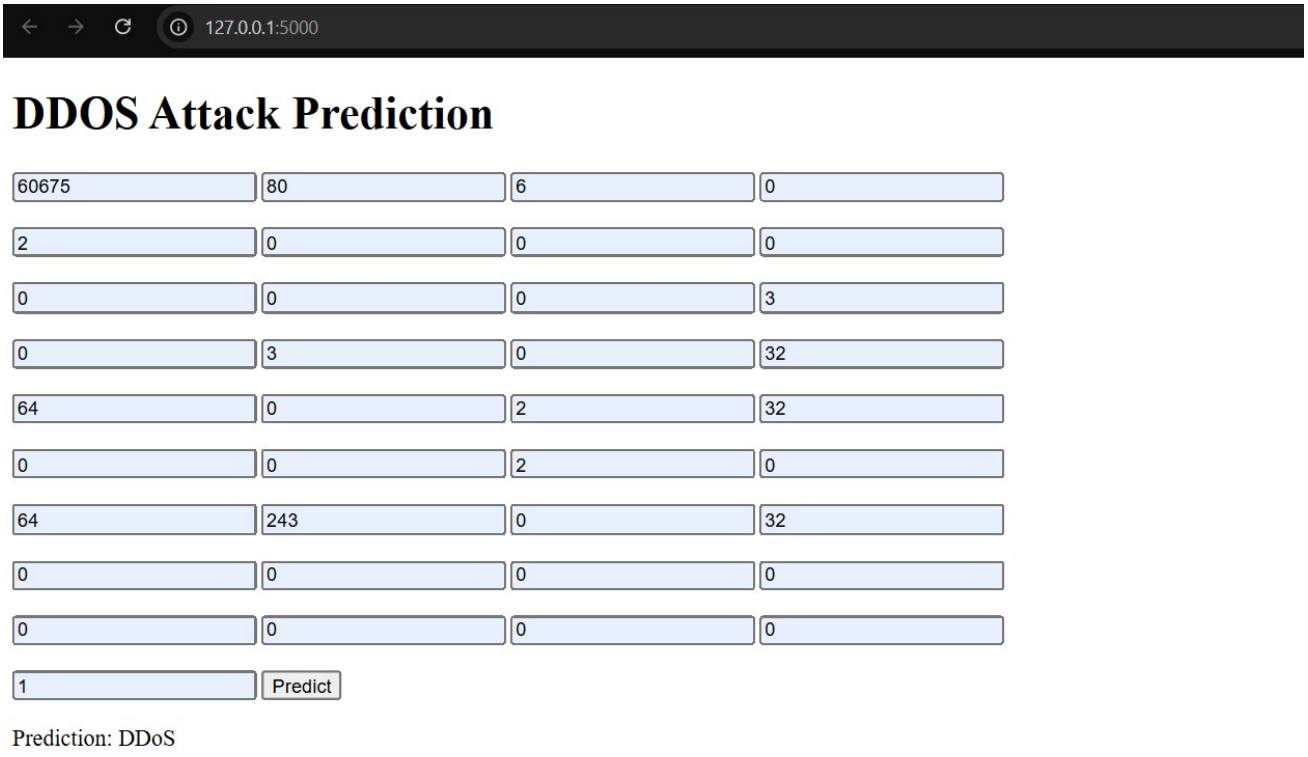


Fig 23. DDoS data inputs in Flask Application

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 got block number, prediction of data and transaction hash for our result.

Fig 24. Output logs 1 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**.

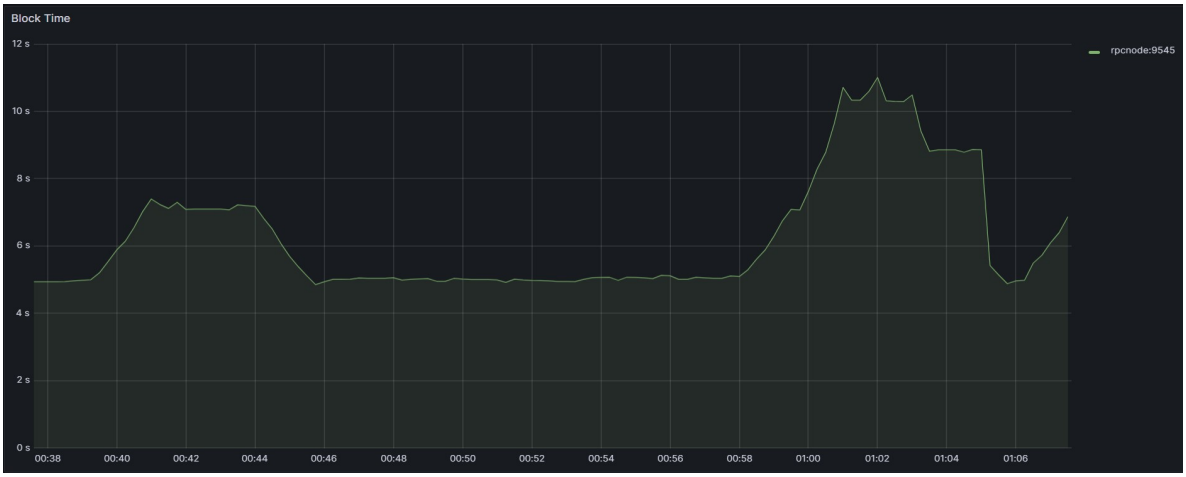


Fig 25. Variations in Block time transaction graph of rpcnode.

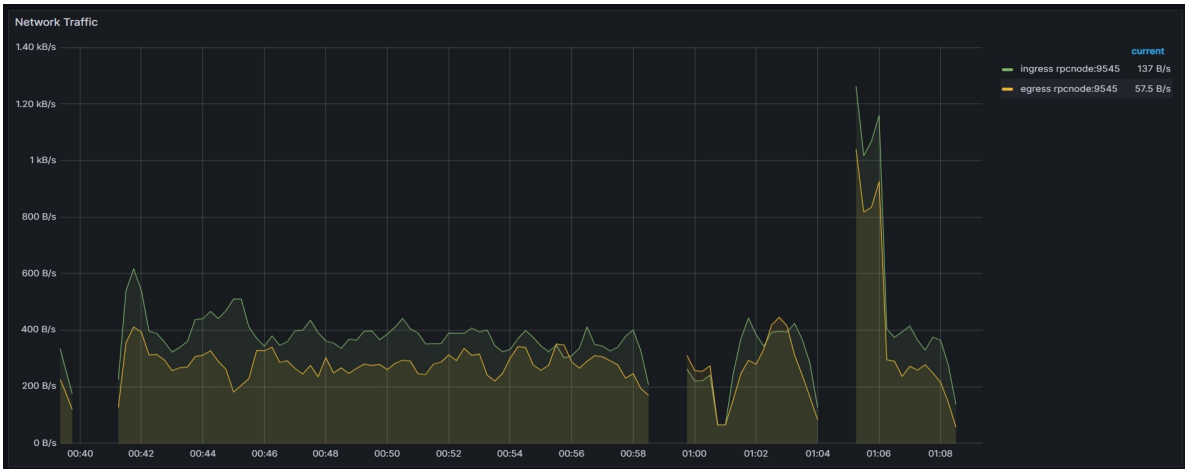


Fig 26. Variations in Network Traffic Graph of rpcnode.

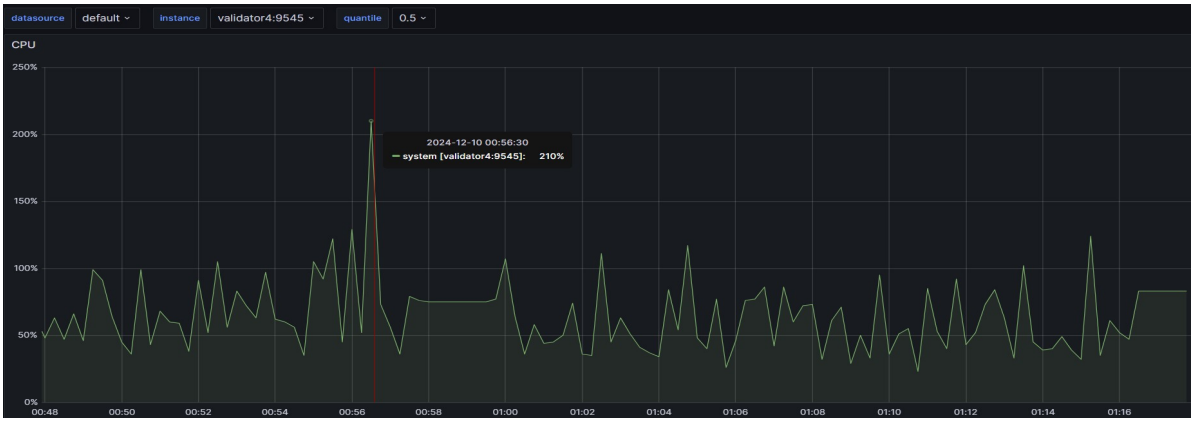


Fig 27. Variations in CPU Usage Graph of validator4 node.

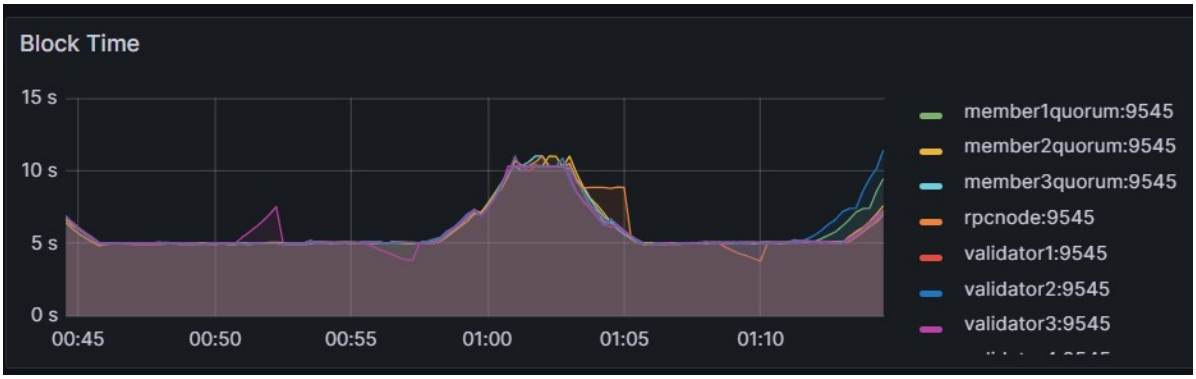


Fig 28. Variations in Block time Graph of all nodes.

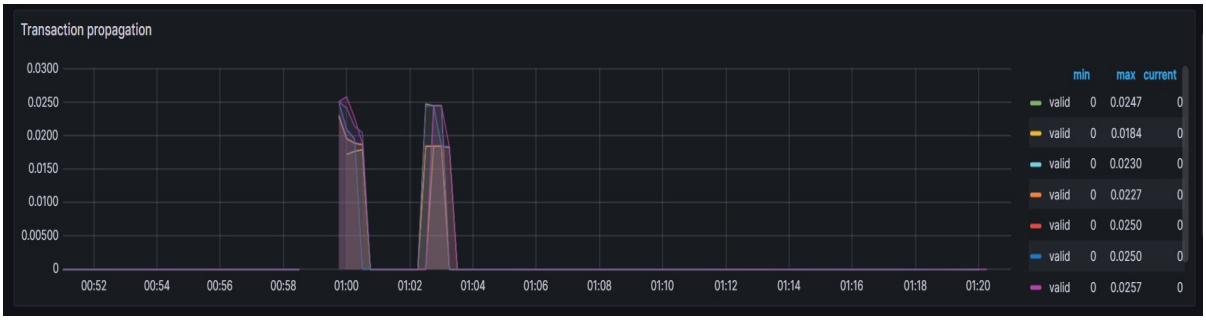


Fig 29. Variations in Transaction propagation Graph of all nodes.

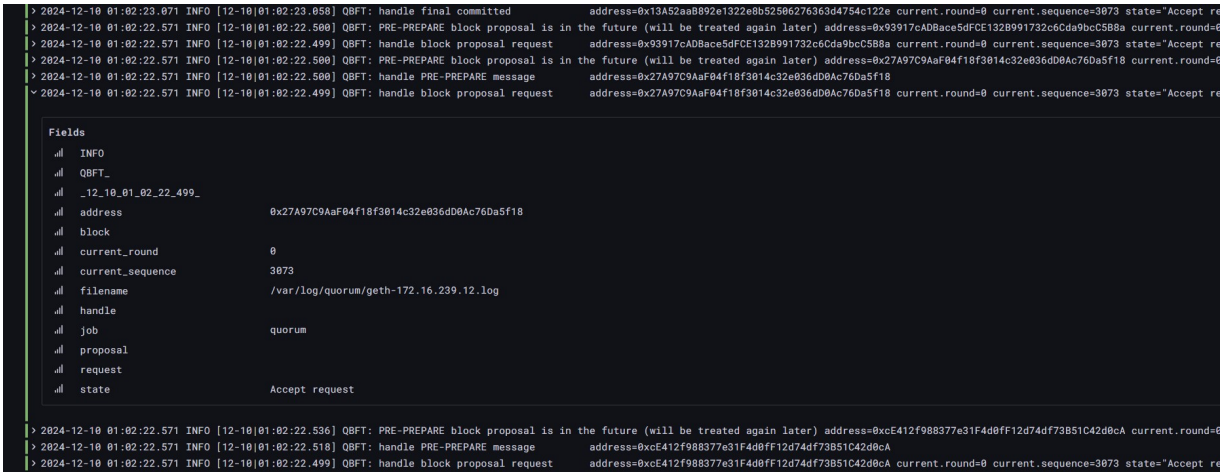


Fig 30. Log Details - 1 of block where DDoS data stored.

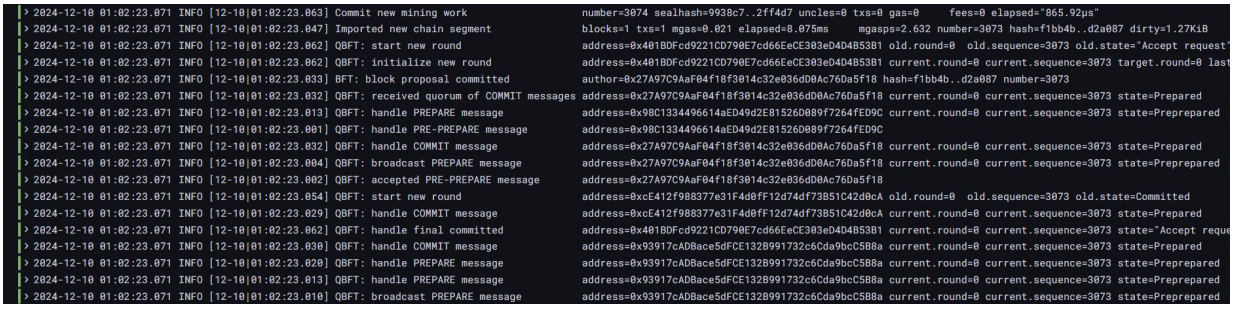


Fig 31. Log Details - 2 of block where DDoS data stored.

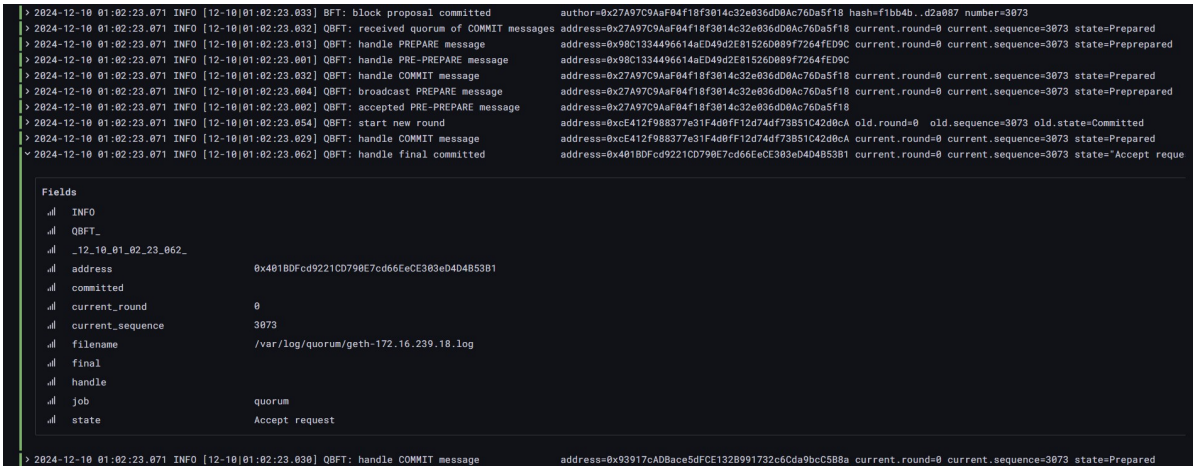


Fig 32. Log Details – 3 of block where DDoS data stored.

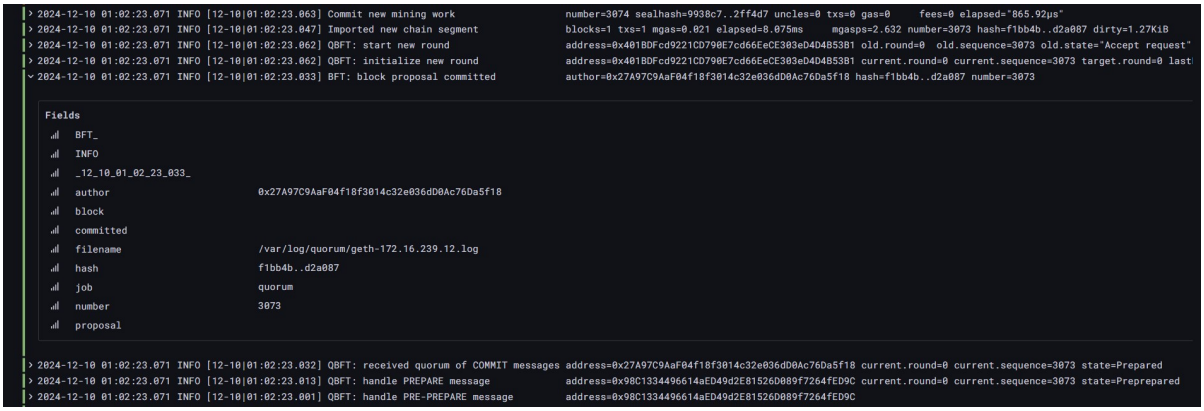


Fig 33. Log Details-4 of block where DDoS data stored.

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

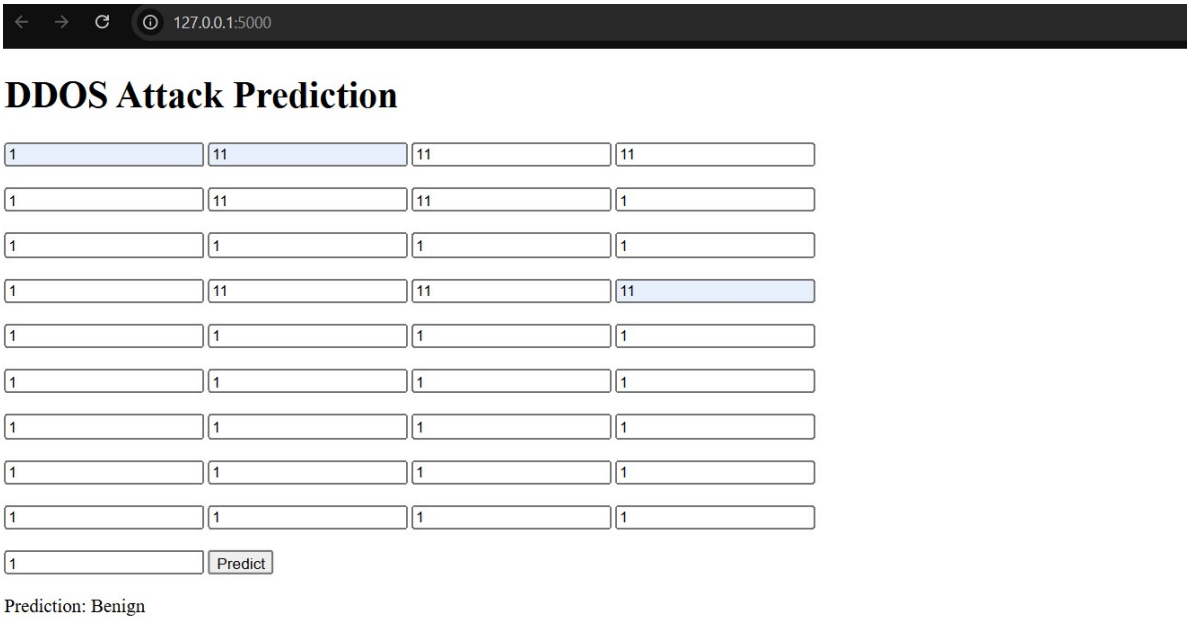


Fig 34. Random / Benign data inputs in Flask Application

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

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

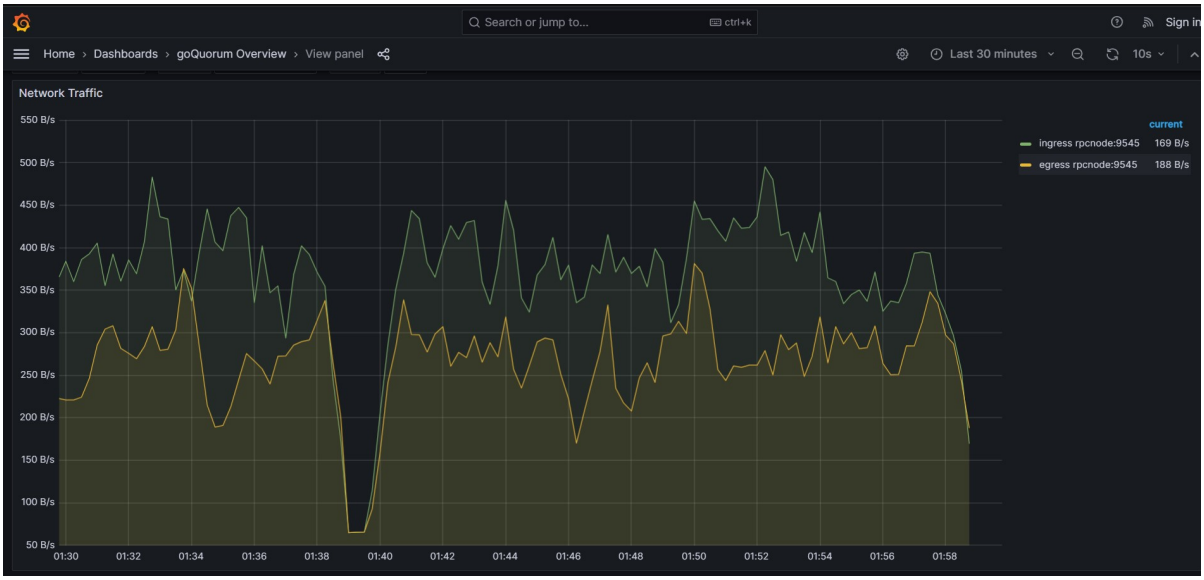


Fig 37. Variations in Network Traffic graph of rpcnode.

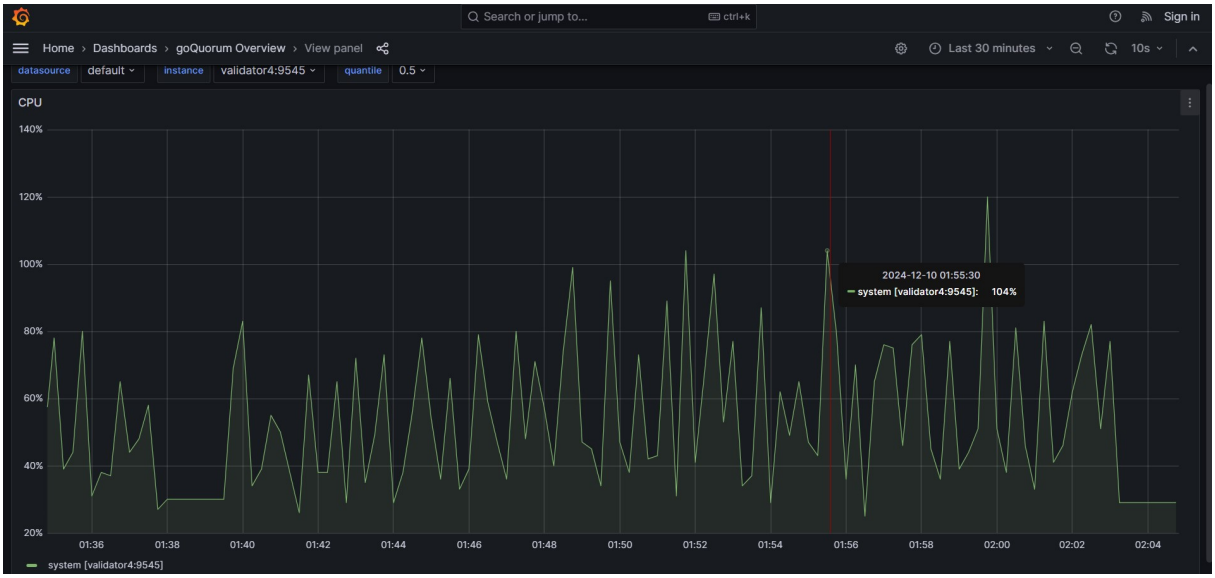


Fig 38. Variations in CPU Usage graph of rpcnode.

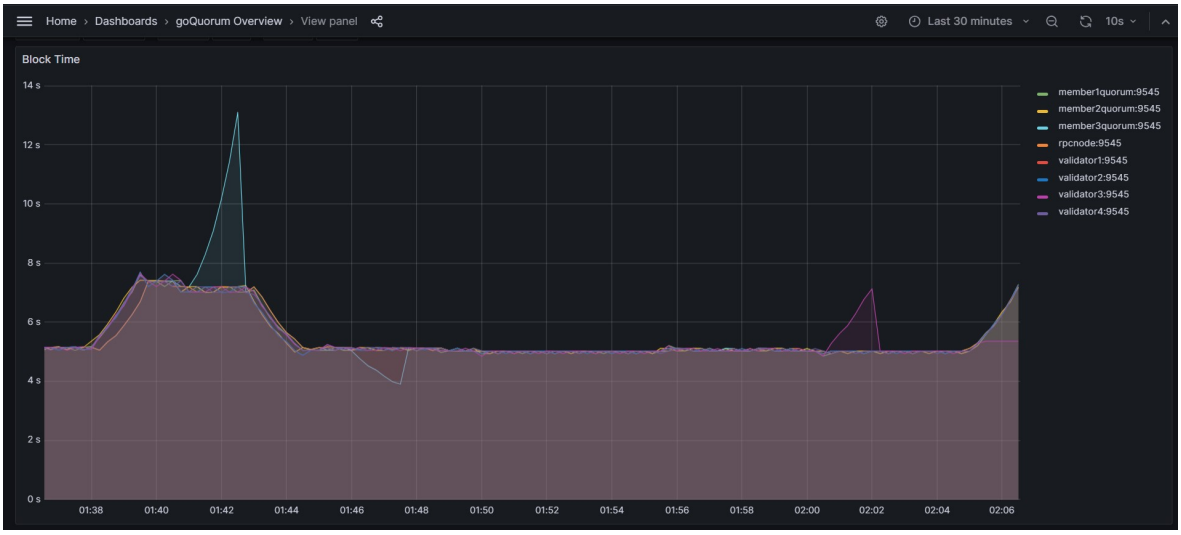


Fig 39. Variations in Block time transaction graph of all nodes.

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

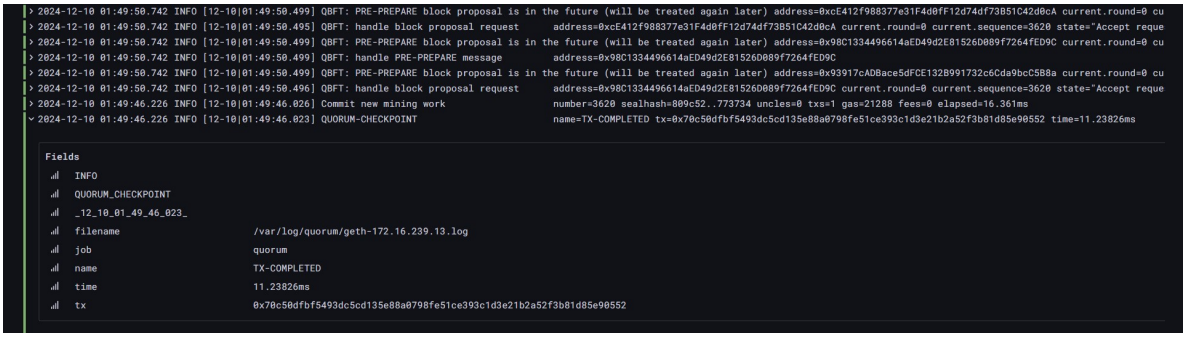


Fig 41. Log Details - 1 of block where Benign data stored.

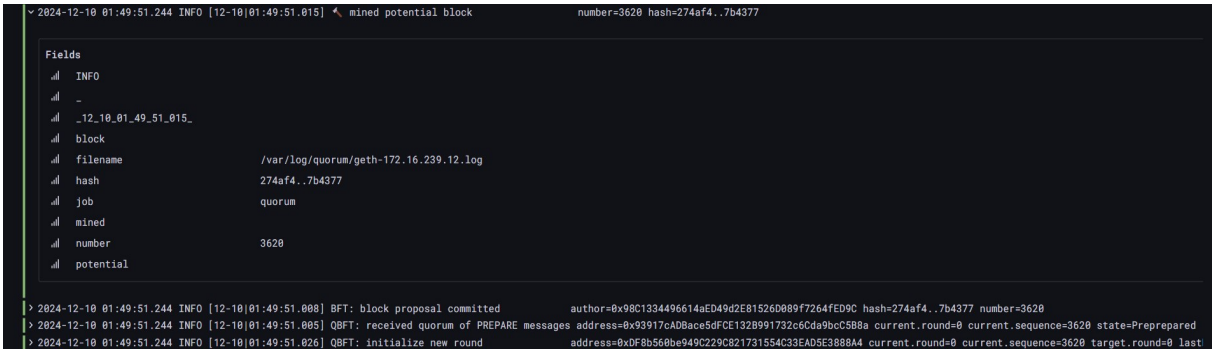


Fig 42. Log Details - 2 of block where Benign data stored.

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).