Anomaly Detection in Network Traffic: A Machine Learning Approach with OMNeT++

EHB 415E

Simulation Experiment: Implementation of the proposed system using the selected simulator, running experiments, and collecting results

Prepared by:

Batu Burgu
Student ID: 040210098

Javad Ibrahimli
Student ID: 040210932

Kerem Karadeniz
Student ID: 040210049

2024-2025 Academic Year

Supervised by:
İbrahim Hökelek, PhD.

Contents

Contents

1 Introduction

Distributed Denial-of-Service (DDoS) attacks are a significant threat to network security, flooding systems with malicious traffic to disrupt legitimate services. To detect and mitigate such attacks, Intrusion Detection Systems (IDS) analyze network traffic for unusual patterns and alert administrators to potential threats. This report outlines the design and implementation of a network simulation, including a DDoS attack, and proposes novel machine-learning-based DDoS attack prevention strategies. OMNeT++ and the INET framework were utilized in order to realize the network topography and attack scenarios.

2 System Design

2.1 Network Topology

The network topology used in the simulation consists of a server, hosts, and zombie devices. An Backbone router is utilized as a backbone to connect the separate subnets. It is assumed that the zombie attacker hosts are located in a separate subnet and communicate with the server through this subnet. The ratio of zombie hosts to normal hosts during data transmission has been varied across different simulations to observe its impact, and data collection has been conducted accordingly.

Since the network being wireless does not have an impact on the algorithms designed for the network properties and traffic management being analyzed, the simulation is designed with devices communicating over a wired physical layer to ensure the simulation is more robust and easy to control. It is intended to simulate the traffic inside the network of the owner of the server, therefore the hosts symbolize data that came from various places and has made it into the network of the owner. Similar results are expected if the simulation network is enlarged and wireless communicating devices are implemented for a wireless network.

Figure 1 and 2 shows the network topology used for the simulations. Details of the data collection process and the simulation parameters will be explained in further sections.

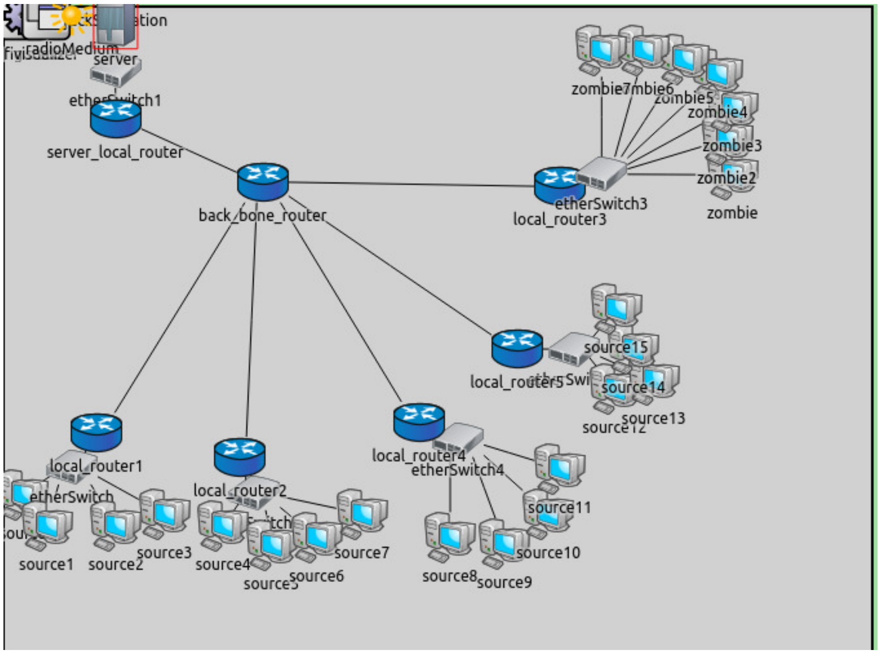


Figure 1: Network Topology 1

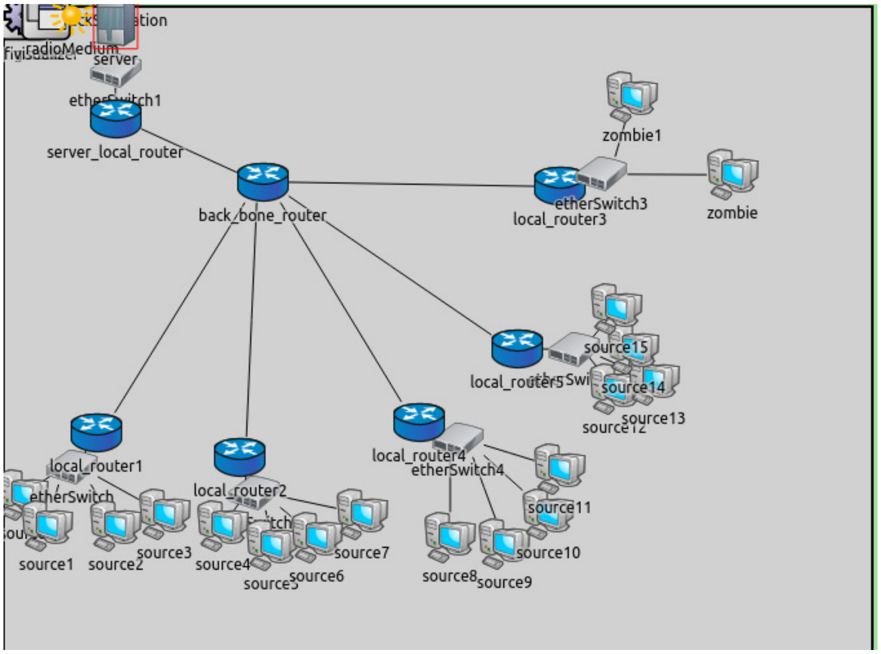


Figure 2: Network Topology 2

2.2 DDoS Attack Realization

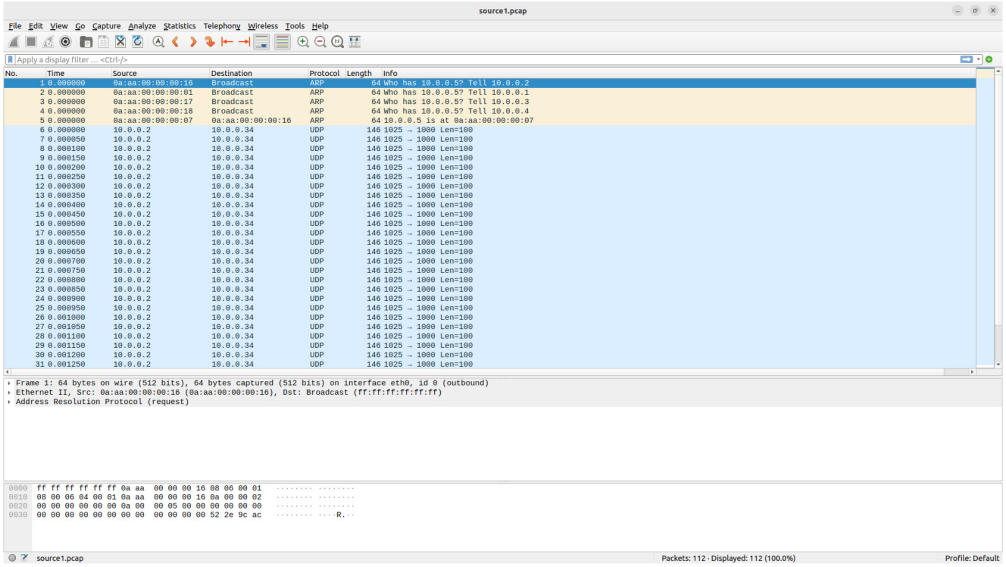
In order to test different attack scenarios, the behavior of the zombie attackers was varied between simulations. In one simulation, the zombies were configured to burst data for 1.5 ms and remain idle for the next 0.5 ms. In another simulation, they operated continuously without any pauses. The UDP protocol was utilized for all communications.

DDoS attacks were simulated in three scenarios with durations of 12 ms, 12 ms, and 25 ms, respectively. In these simulations, the relative number of zombie attackers and their attack patterns were varied according to the descriptions provided above. This approach ensured that the data collected included a diverse range of scenarios, which is essential for training and testing the AI model effectively.

3 Simulation

3.1 Data Export and Wireshark Implementation

To extract the required data from the simulation, the Wireshark PCAP Manager was integrated into the OMNeT++ application, and the data was exported using Wireshark. As shown in Figure 2, ARP requests for generating the ARP tables at simulation time zero can be observed in Wireshark, capturing the necessary packet exchanges.


source1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
source1.pcap
Packets: 112 · Displayed: 112 (100.0%)
Profile: Default

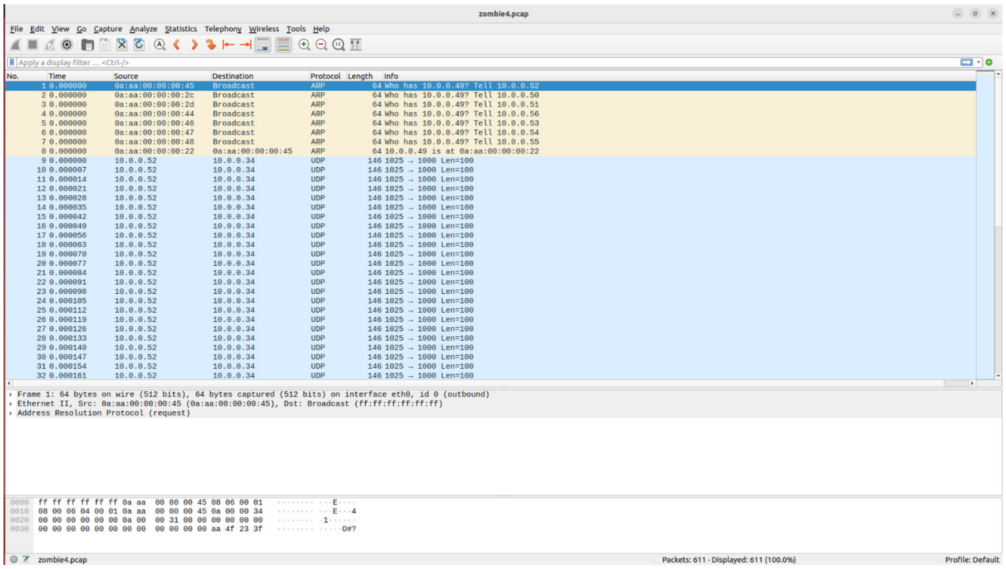
```
No.  Time     Source             Destination        Protocol Length Info
   1 0.000000 0a:aa:00:00:00:16  Broadcast          ARP          64 Who has 10.0.0.5? Tell 10.0.0.2
   2 0.000000 0a:aa:00:00:00:01  Broadcast          ARP          64 Who has 10.0.0.5? Tell 10.0.0.1
   3 0.000000 0a:aa:00:00:00:17  Broadcast          ARP          64 Who has 10.0.0.5? Tell 10.0.0.3
   4 0.000000 0a:aa:00:00:00:18  Broadcast          ARP          64 Who has 10.0.0.5? Tell 10.0.0.4
   5 0.000000 0a:aa:00:00:00:07  0a:aa:00:00:00:16  ARP          64 10.0.0.5 is at 0a:aa:00:00:00:07
   6 0.000000 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
   7 0.000050 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
   8 0.000100 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
   9 0.000150 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  10 0.000200 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  11 0.000250 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  12 0.000300 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  13 0.000350 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  14 0.000400 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  15 0.000450 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  16 0.000500 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  17 0.000550 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  18 0.000600 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  19 0.000650 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  20 0.000700 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  21 0.000750 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  22 0.000800 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  23 0.000850 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  24 0.000900 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  25 0.000950 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  26 0.001000 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  27 0.001050 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  28 0.001100 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  29 0.001150 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  30 0.001200 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
  31 0.001250 10.0.0.2           10.0.0.34          UDP         146 1025 → 1000 Len=100
```

```
› Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface eth0, id 0 (outbound)
› Ethernet II, Src: 0a:aa:00:00:00:16 (0a:aa:00:00:00:16), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 0a aa  00 00 00 16 08 06 00 01   ········ ········
0010  08 00 06 04 00 01 0a aa  00 00 00 16 0a 00 00 02   ········ ········
0020  00 00 00 00 00 00 0a 00  00 05 00 00 00 00 00 00   ········ ········
0030  00 00 00 00 00 00 00 00  00 00 00 00 52 2e 9c ac   ········ ····R.··
```

Figure 3: Legitimate Host Communication Packets

The data collected via Wireshark includes key metrics such as packet size, source and destination ports, IP addresses, data transmission frequency, and data protocols. These metrics were exported as CSV files and utilized for analyzing network traffic and training the AI model to detect attackers. Figure 3 shows the packets captured with Wireshark which have been sent from the zombie attacker clients.


zombie4.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0a:aa:00:00:00:45	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.52
2	0.000000	0a:aa:00:00:00:2c	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.50
3	0.000000	0a:aa:00:00:00:2d	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.51
4	0.000000	0a:aa:00:00:00:44	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.56
5	0.000000	0a:aa:00:00:00:46	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.53
6	0.000000	0a:aa:00:00:00:47	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.54
7	0.000000	0a:aa:00:00:00:48	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.55
8	0.000000	0a:aa:00:00:00:22	0a:aa:00:00:00:45	ARP	64	10.0.0.49 is at 0a:aa:00:00:00:22
9	0.000000	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
10	0.000007	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
11	0.000014	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
12	0.000021	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
13	0.000028	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
14	0.000035	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
15	0.000042	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
16	0.000049	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
17	0.000056	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
18	0.000063	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
19	0.000070	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
20	0.000077	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
21	0.000084	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
22	0.000091	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
23	0.000098	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
24	0.000105	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
25	0.000112	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
26	0.000119	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
27	0.000126	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
28	0.000133	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
29	0.000140	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
30	0.000147	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
31	0.000154	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100
32	0.000161	10.0.0.52	10.0.0.34	UDP	146	1025 → 1000 Len=100

```
▸ Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface eth0, id 0 (outbound)
▸ Ethernet II, Src: 0a:aa:00:00:00:45 (0a:aa:00:00:00:45), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 0a aa  00 00 00 45 08 06 00 01   ········ ···E····
0010  08 00 06 04 00 01 0a aa  00 00 00 45 0a 00 00 34   ········ ···E···4
0020  00 00 00 00 00 00 0a 00  00 31 00 00 00 00 00 00   ········ ·1······
0030  00 00 00 00 00 00 00 00  00 00 00 00 aa 4f 23 3f   ········ ····O#?
```

zombie4.pcap Packets: 611 · Displayed: 611 (100.0%) Profile: Default

Figure 4: Zombie Attacker Packets

4 Artificial Intelligence Model

The artificial intelligence model developed in this project is designed to detect anomalies in network traffic, with a primary focus on identifying Distributed Denial-of-Service (DDoS) attacks. The AI framework employs machine learning algorithms to analyze patterns in simulated network traffic and classify them as either legitimate or malicious.

4.1 Feature Selection and Data Preparation

Key features used to train the AI models were extracted from the network simulation and include:

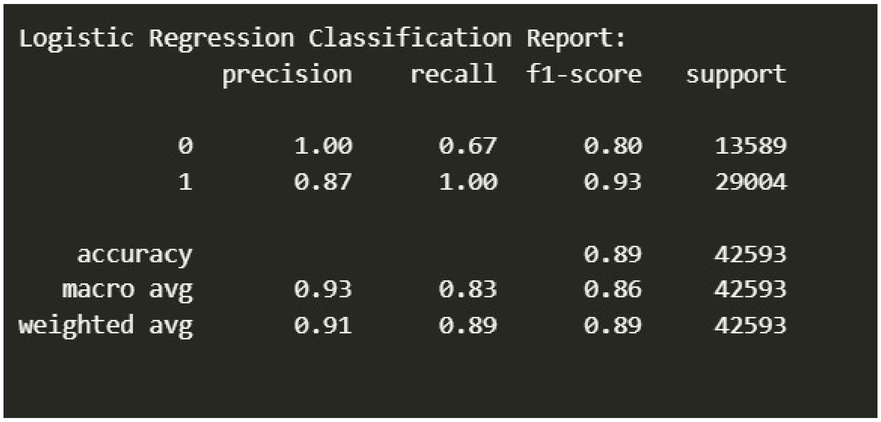
- Packet size
- Source and destination ports
- IP addresses
- Transmission frequency
- Protocols

These features were collected using Wireshark and exported as CSV files for preprocessing. The preprocessing pipeline involved scaling, normalization, and addressing class imbalances to enhance model accuracy.

4.2 Machine Learning Algorithms

Several machine learning models were implemented and tested:

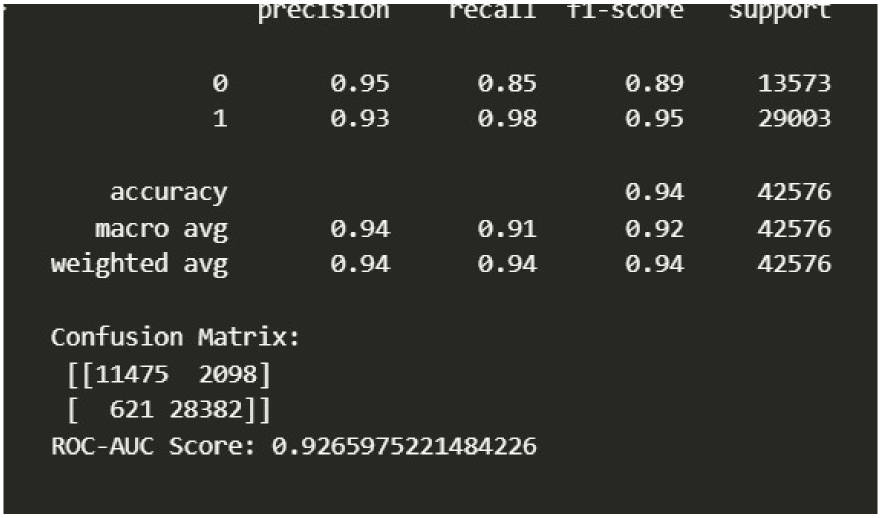
- **Logistic Regression:** A baseline model achieving an accuracy of 89%, as shown in Figure 5.



```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.67      0.80     13589
           1       0.87      1.00      0.93     29004

    accuracy                           0.89     42593
   macro avg       0.93      0.83      0.86     42593
weighted avg       0.91      0.89      0.89     42593
```

Figure 5: Logistic Regression Model Results

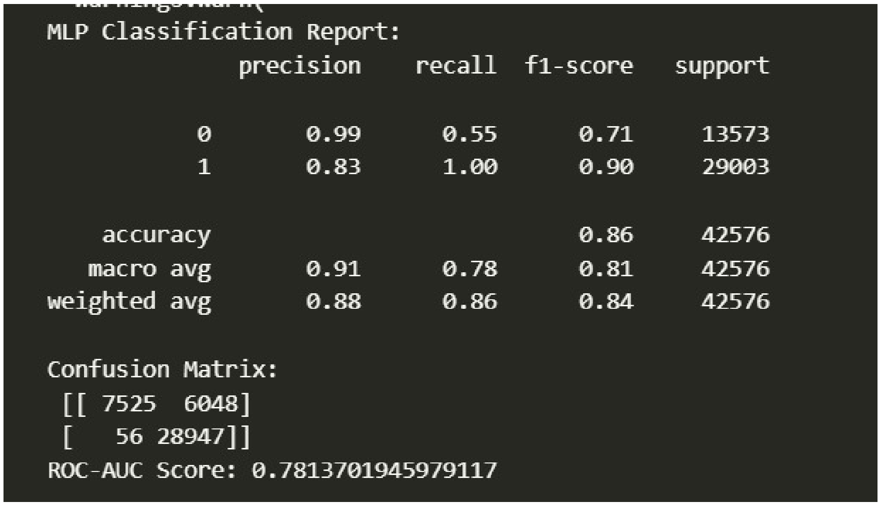
- **Random Forest:** A robust ensemble-based model, achieving the highest accuracy (94%) and an ROC-AUC score of 0.927, as shown in Figure 6.



```
              precision    recall  f1-score   support

           0       0.95      0.85      0.89     13573
           1       0.93      0.98      0.95     29003

    accuracy                           0.94     42576
   macro avg       0.94      0.91      0.92     42576
weighted avg       0.94      0.94      0.94     42576

Confusion Matrix:
 [[11475  2098]
 [  621 28382]]
ROC-AUC Score: 0.9265975221484226
```

Figure 6: Random Forest Model Results

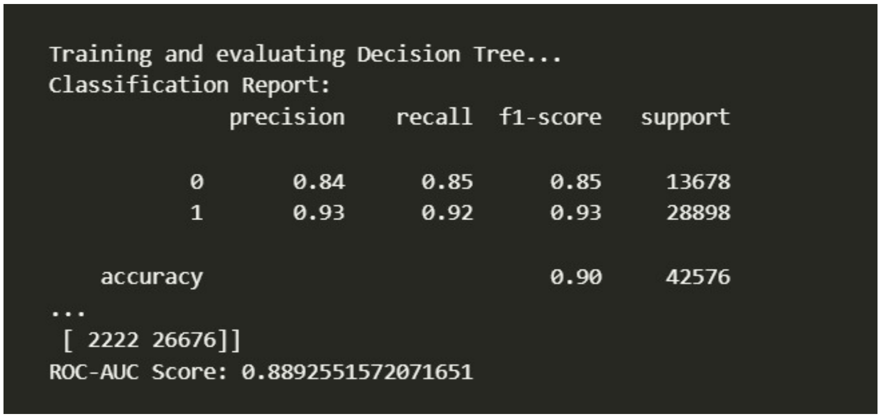
- **Multi-Layer Perceptron (MLP):** A neural network-based model showing moderate performance, achieving 86% accuracy (Figure 7).



```
  warnings.warn(
MLP Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.55      0.71     13573
           1       0.83      1.00      0.90     29003

    accuracy                           0.86     42576
   macro avg       0.91      0.78      0.81     42576
weighted avg       0.88      0.86      0.84     42576

Confusion Matrix:
 [[ 7525  6048]
 [   56 28947]]
ROC-AUC Score: 0.7813701945979117
```

Figure 7: Multi Layer Perception Model Results

- **Decision Tree:** A straightforward and interpretable model, with an accuracy of 90% (Figure 8).



```
Training and evaluating Decision Tree...
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.85      0.85     13678
           1       0.93      0.92      0.93     28898

    accuracy                           0.90     42576
...
 [ 2222 26676]]
ROC-AUC Score: 0.8892551572071651
```

Figure 8: Decision Tree Model Results

4.3 Detailed Analysis of Model Performances

The performance of the machine learning models used for detecting Distributed Denial-of-Service (DDoS) attacks was evaluated comprehensively using multiple metrics. These metrics include accuracy, precision, recall, F1-score, and ROC-AUC. This section provides an in-depth analysis of each model's performance based on the results obtained during the evaluation phase.

4.3.1 Logistic Regression

The Logistic Regression model served as a baseline algorithm for anomaly detection. It demonstrated an accuracy of 89%, as shown in Figure 5, and achieved reasonable precision and recall values:

- **Precision:** 100% for class 0 (legitimate traffic) and 87% for class 1 (malicious traffic).
- **Recall:** 67% for class 0 and 100% for class 1.
- **F1-Score:** The weighted F1-score of 0.89 indicates balanced performance across classes.

However, the model showed weaknesses in detecting legitimate traffic due to relatively lower recall for class 0, suggesting susceptibility to false negatives.

4.3.2 Random Forest

The Random Forest model achieved the best performance among all tested algorithms, with an overall accuracy of 94%, as illustrated in Figure 6. Key performance metrics are as follows:

- **Precision:** 95% for class 0 and 93% for class 1.
- **Recall:** 85% for class 0 and 98% for class 1.
- **F1-Score:** The macro-average F1-score of 0.91 highlights excellent performance across both classes.
- **ROC-AUC:** The ROC-AUC score of 0.927 indicates strong discriminatory power.

This model's balanced precision and recall values make it the most suitable choice for detecting anomalies in network traffic.

4.3.3 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron (MLP) model, while offering a non-linear approach to classification, performed moderately well with an accuracy of 86% (Figure 7). The results indicate:

- **Precision:** 99% for class 0 and 83% for class 1.
- **Recall:** 55% for class 0 and 100% for class 1.
- **F1-Score:** The macro-average F1-score of 0.78 reflects inconsistencies in detecting legitimate traffic.
- **ROC-AUC:** The ROC-AUC score of 0.781 indicates moderate ability to distinguish between classes.

While MLP achieved perfect recall for malicious traffic, it suffered from poor recall for legitimate traffic, resulting in a high number of false positives.

4.3.4 Decision Tree

The Decision Tree model provided interpretability while maintaining strong performance, with an overall accuracy of 90% (Figure 8). Detailed metrics include:

- **Precision:** 84% for class 0 and 93% for class 1.
- **Recall:** 85% for class 0 and 92% for class 1.
- **F1-Score:** The macro-average F1-score of 0.85 indicates consistent performance across both classes.
- **ROC-AUC:** The ROC-AUC score of 0.889 suggests a robust capability to distinguish between legitimate and malicious traffic.

Although the Decision Tree performed well, it was slightly less accurate than the Random Forest model due to its susceptibility to overfitting.

4.3.5 Comparison and Insights

A comparative analysis of the models is summarized in Table 1:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**ROC-AUC**
Logistic Regression	89%	91%	83%	86%	0.85
Random Forest	94%	94%	91%	91%	0.927
MLP	86%	91%	78%	81%	0.781
Decision Tree	90%	89%	85%	85%	0.889

Table 1: Performance Metrics Comparison

From the analysis, the Random Forest model is the most reliable due to its balanced performance across all metrics and its ability to generalize effectively.

4.4 Future Work on Model Optimization

Future work will focus on:

- **Hyperparameter Tuning:** Further optimization of model parameters to enhance performance.
- **Advanced Architectures:** Incorporating ensemble stacking and deep learning models for better generalization.
- **Real-Time Deployment:** Integrating the best-performing model into the OMNeT++ framework for live traffic analysis.

The final version of the paper will provide **more detailed visualizations and improved analysis**, highlighting additional insights into the models' performance under varying network conditions.

5 Conclusion

It has been observed that the attacker detection algorithms created with different models work with an accuracy of approximately 90%

6 References

1. Legitimate Client App: `https://doc.omnetpp.org/inet/api-current/neddoc/inet.applications.udpapp.UdpBasicApp.html`
2. Zombie Client App: `https://doc.omnetpp.org/inet/api-current/neddoc/inet.applications.udpapp.UdpBasicBurst.html`
3. Server App: `https://doc.omnetpp.org/inet/api-current/neddoc/inet.applications.udpapp.UdpSink.html`
4. Ethernet Switch: `https://doc.omnetpp.org/inet/api-current/neddoc/inet.node.ethernet.EthernetSwitch.html`
5. Router: `https://doc.omnetpp.org/inet/api-4.4.0/neddoc/inet.node.inet.Router.html`
6. PCAP Recording : https://inet.omnetpp.org/docs/showcases/general/pcaprecording/doc/index.htm

7. Kralevska, K., Garau, M., Førland, M., & Gligoroski, D. (2019). Towards 5g intrusion detection scenarios with omnet++. Proceedings of 6th International OM, 66, 44-51.