2024-2025 Academic Year

Istanbul Technical University

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

Bridging Machine Learning and Network Traffic Analysis

Course Name:

EHB 415E -Data Communication

Batu Burgu - 040210098
Javad Ibrahimli - 040210932
Kerem Karadeniz - 040210049

Supervised by:

İbrahim Hökelek, PhD.

Table of Contents

1 Motivation

2 Network Topology

3 Exporting Data

4 Data Preprocessing

5 Machine Learning Models

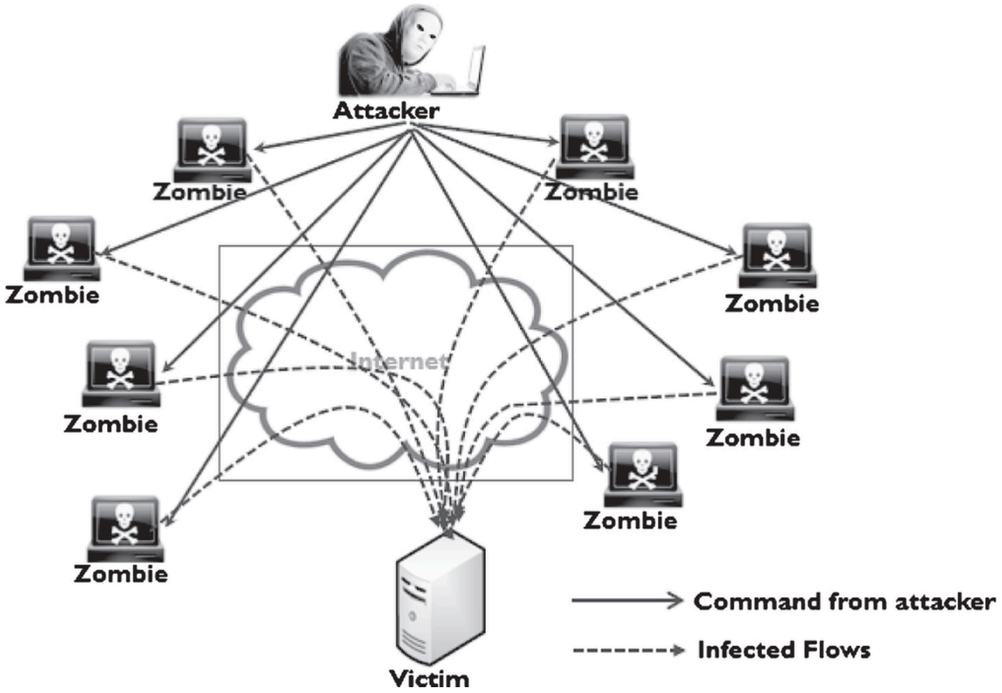
1. Motivation

1. Motivation

Detection of Malicious Actions

Ensuring Secure & Stable Communication

Automation of Classification Processes Inside Networks



2. Network Topology

2. Network Topology

Main object classes in the network:

Hosts # 1

Used for generating traffic across network

Each configered at a different data rate & package length

Attackers # 2

Used for generating malicious traffic across network

Each configered at a different data rate & package length

Network Devices # 3

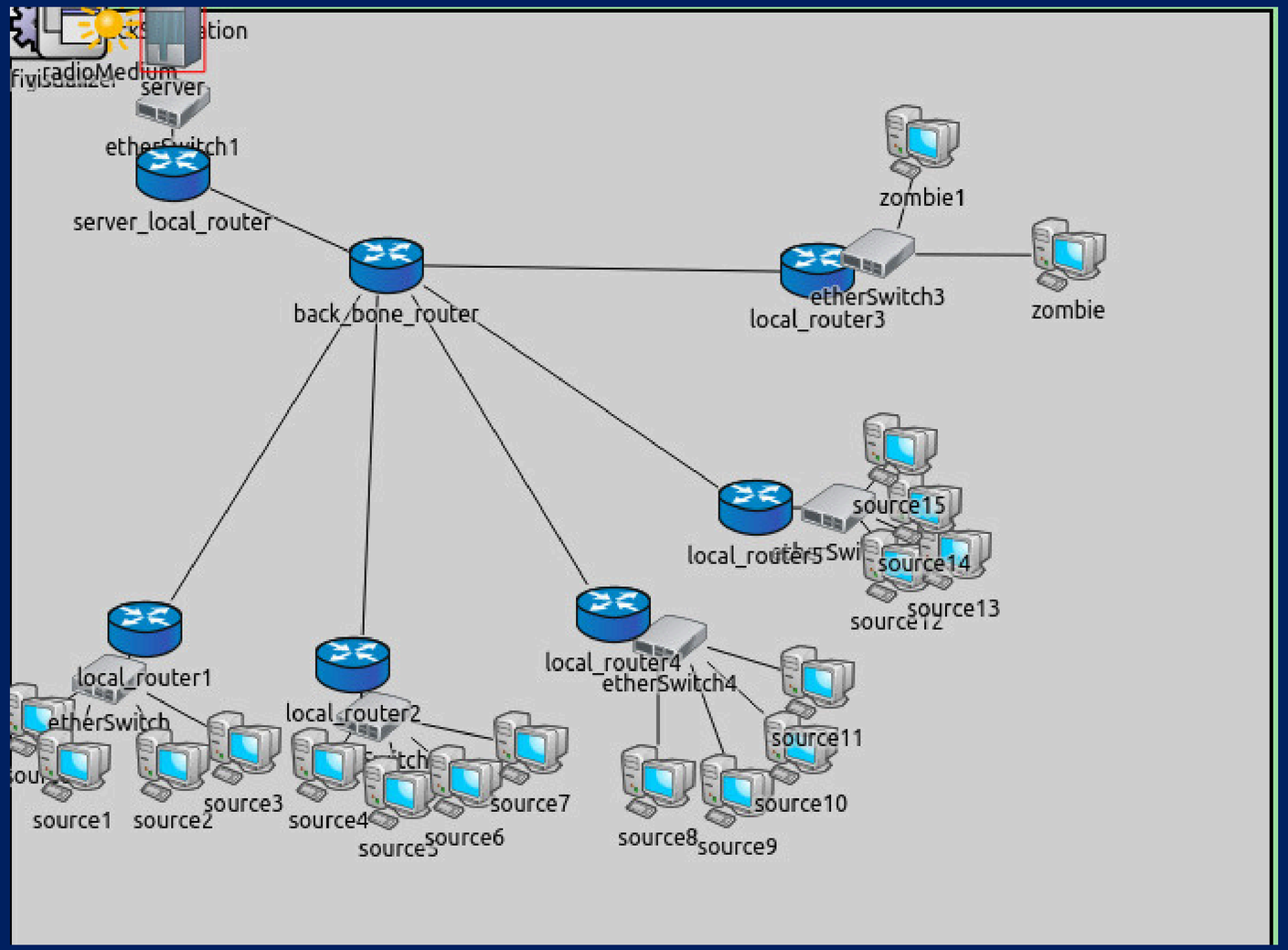
Used for carrying data across network

Automatic IPv4 address configurator is used in network

2. Network Topology

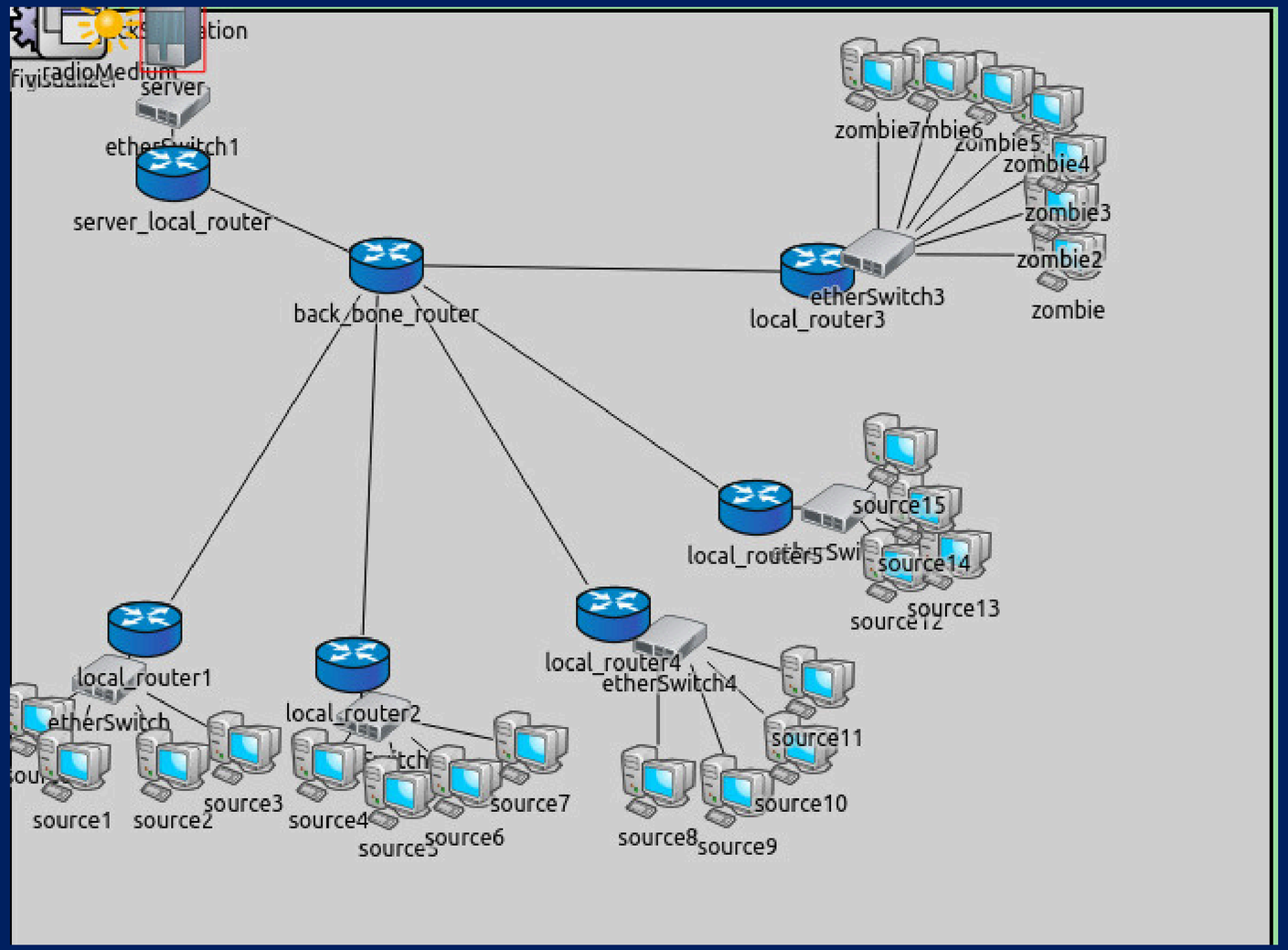
- # 1 Two network topologies are used to gather transmission data
- # 2 Each topology has 15 valid hosts, one has 2 attackers and the other has 8
- # 3 Runtimes for simulations are 15ms, 24ms and 30ms
- # 4 Wired connections since the data transmission method is irrevelant

Topology 1



- 15 Hosts
- 2 Attackers
- 400 Gbps Eth Connections Except the Connection Between the Server and the Server Switch
- 100 Gbps Eth Connection Between the Server and the Server Switch

Topology 2



- 15 Hosts
- 8 Attackers
- 400 Gbps Eth Connections Except the Connection Between the Server and the Server Switch
- 100 Gbps Eth Connection Between the Server and the Server Switch

3. Exporting Data

3. Exporting Data

 # 1 The data is exported as .pcap file

 # 2 PcapRecorder node from INET is utilized to create .pcap file

 # 3 Traffic information is gathered from all devices seperately

 # 4 The traffic data is exported from Wireshark as .csv file

4. PCAP MANAGERS

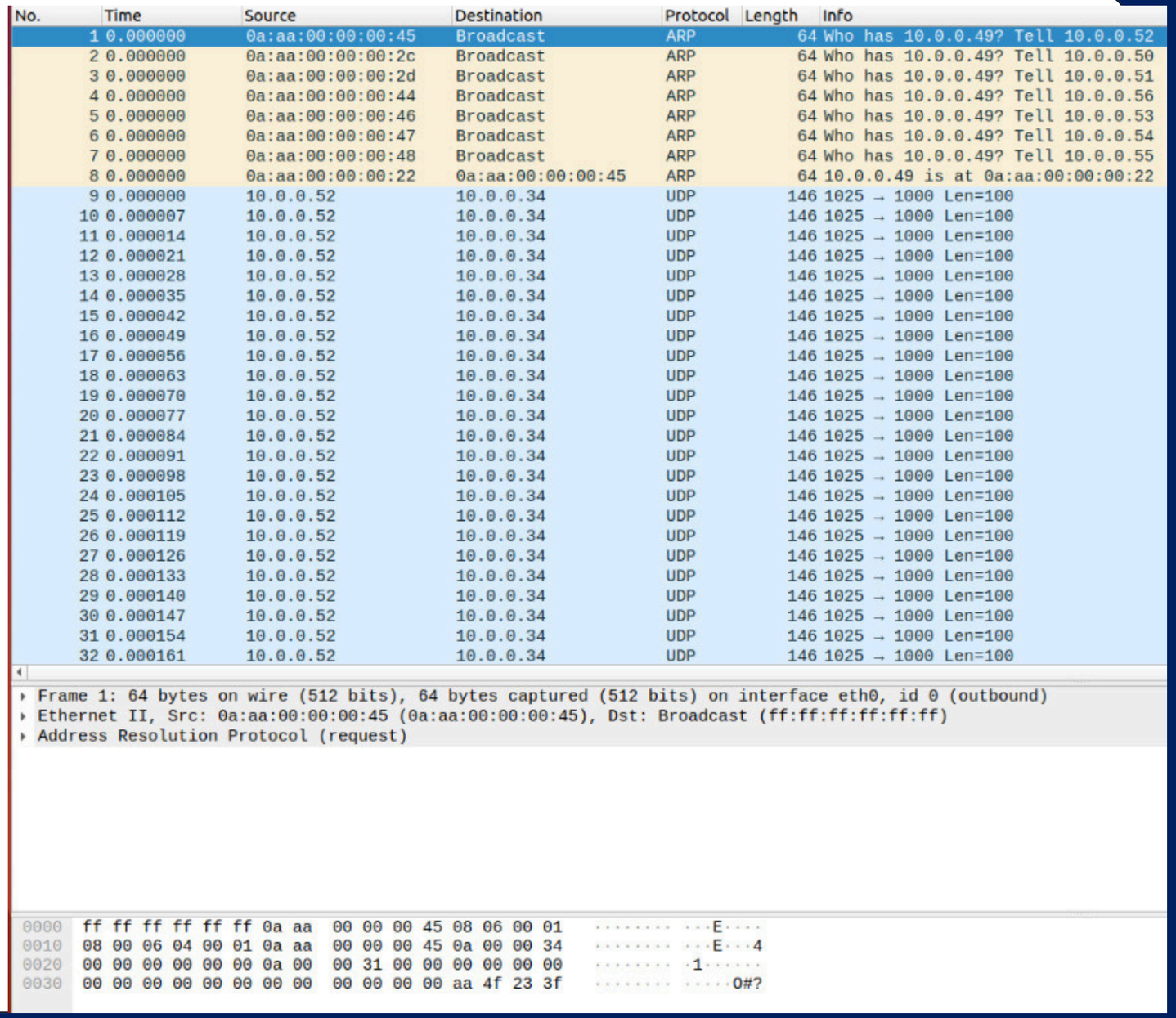
1 Wireshark, tcpdump, Moloch

2 Stands for Packet Capture

3 They are used in order to inspect and debug the networks

4 Creates files consisting of information regarding the network like IP, Protocol...

Wireshark Host



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

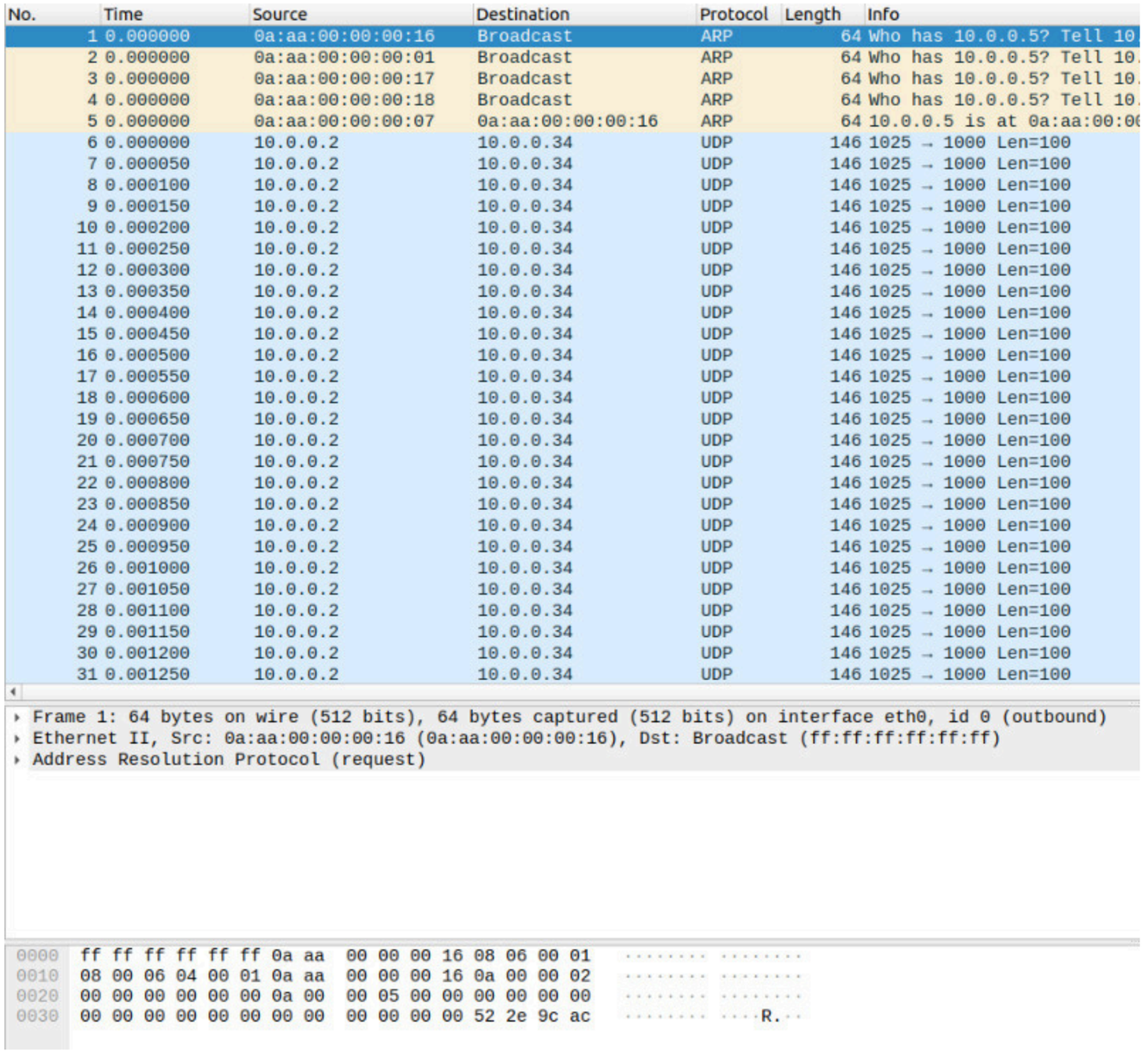
```
▸ Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface eth0, id 0 (outbound)
▸ Ethernet II, Src: 0a:aa:00:00:00:45 (0a:aa:00:00:00:45), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

```
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#?
```

Figure showing the information regarding one of the hosts with the ip address 10.0.0.52. Sending Packets to 10.0.0.34

Wireshark Attacker

Figure showing the information regarding one of the hosts with the ip address 10.0.0.2. Sending Packets to 10.0.0.34



Exported CSV Files

A	B	C	D	E	F	G	H	I
No.	Time	Source	Destinatio	Protocol	Length	Info		
1	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.5? Tell 10.0.0.1		
2	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.5? Tell 10.0.0.2		
3	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.5? Tell 10.0.0.3		
4	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.5? Tell 10.0.0.4		
5	0.000000	0a:aa:00:0	0a:aa:00:0	ARP	64	10.0.0.5 is at 0a:aa:00:00:00:07		
6	0.000000	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
7	0.000020	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
8	0.000040	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
9	0.000060	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
10	0.000080	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
11	0.000100	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
12	0.000120	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
13	0.000140	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
14	0.000160	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
15	0.000180	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		
16	0.000200	10.0.0.1	10.0.0.34	UDP	446	1025 > 1000 Len=400		

A	B	C	D	E	F	G	H	I
No.	Time	Source	Destinatio	Protocol	Length	Info		
1	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.		
2	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.		
3	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.		
4	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.		
5	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.		
6	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.		
7	0.000000	0a:aa:00:0	Broadcast	ARP	64	Who has 10.0.0.49? Tell 10.0.0.		
8	0.000000	0a:aa:00:0	0a:aa:00:0	ARP	64	10.0.0.49 is at 0a:aa:00:00:00:2		
9	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
10	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
11	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
12	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
13	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
14	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
15	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
16	0.000000	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
17	0.000001	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		
18	0.000001	10.0.0.50	10.0.0.34	UDP	146	1025 > 1000 Len=100		

Above: Legitimate Host Packets
Below: Zombie Attacker Packets

Data Collected

1 Source and Destination IP Addresses

2 Transmission Time, Transmission Frequency

3 Transmisson Protocol

4 Packet Size, Label as Host and Attacker

Data Cleaning

1 CSV files exported for each host and zombie client from the simulations

2 ARP packets has been removed from data set

3 Labeled the packets as host and zombie in order to train the model

4 Merged all of the files except for some in order to test the model

4. Machine Learning Models

Data Augmentation

Purpose:

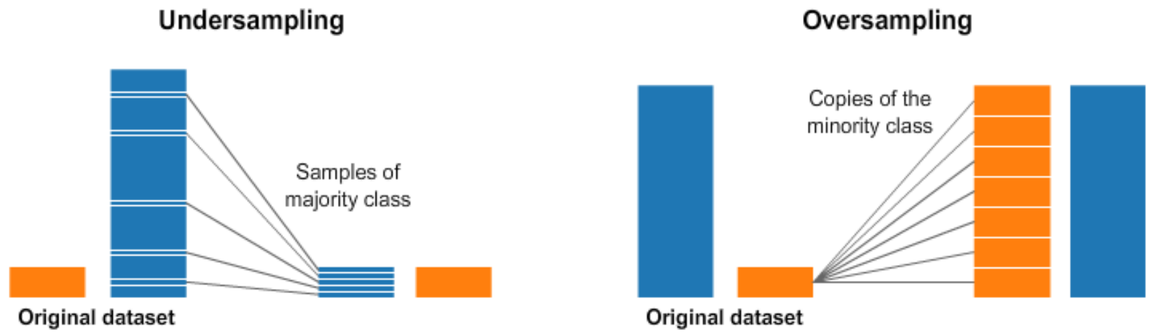
Addressed class imbalance by creating synthetic data for underrepresented classes (e.g., malicious traffic).

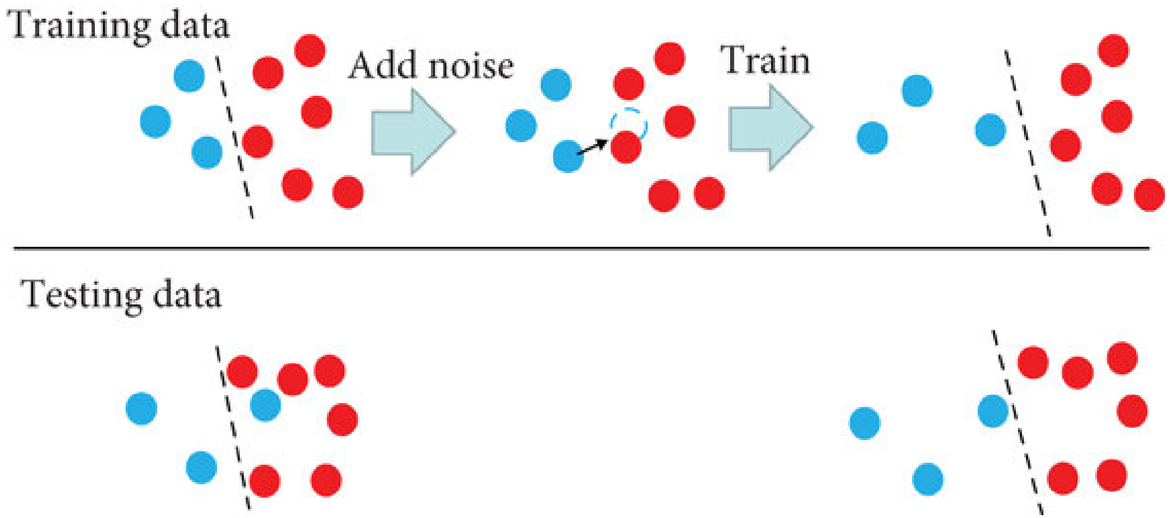
Techniques Used:

- **Oversampling**: Duplicated minority class samples to balance the dataset.
- **Synthetic Data Generation:** Applied transformations like random noise addition and feature scaling to simulate diverse traffic patterns.

Impact:

- Improved the robustness and generalization of machine learning models by providing more representative training data

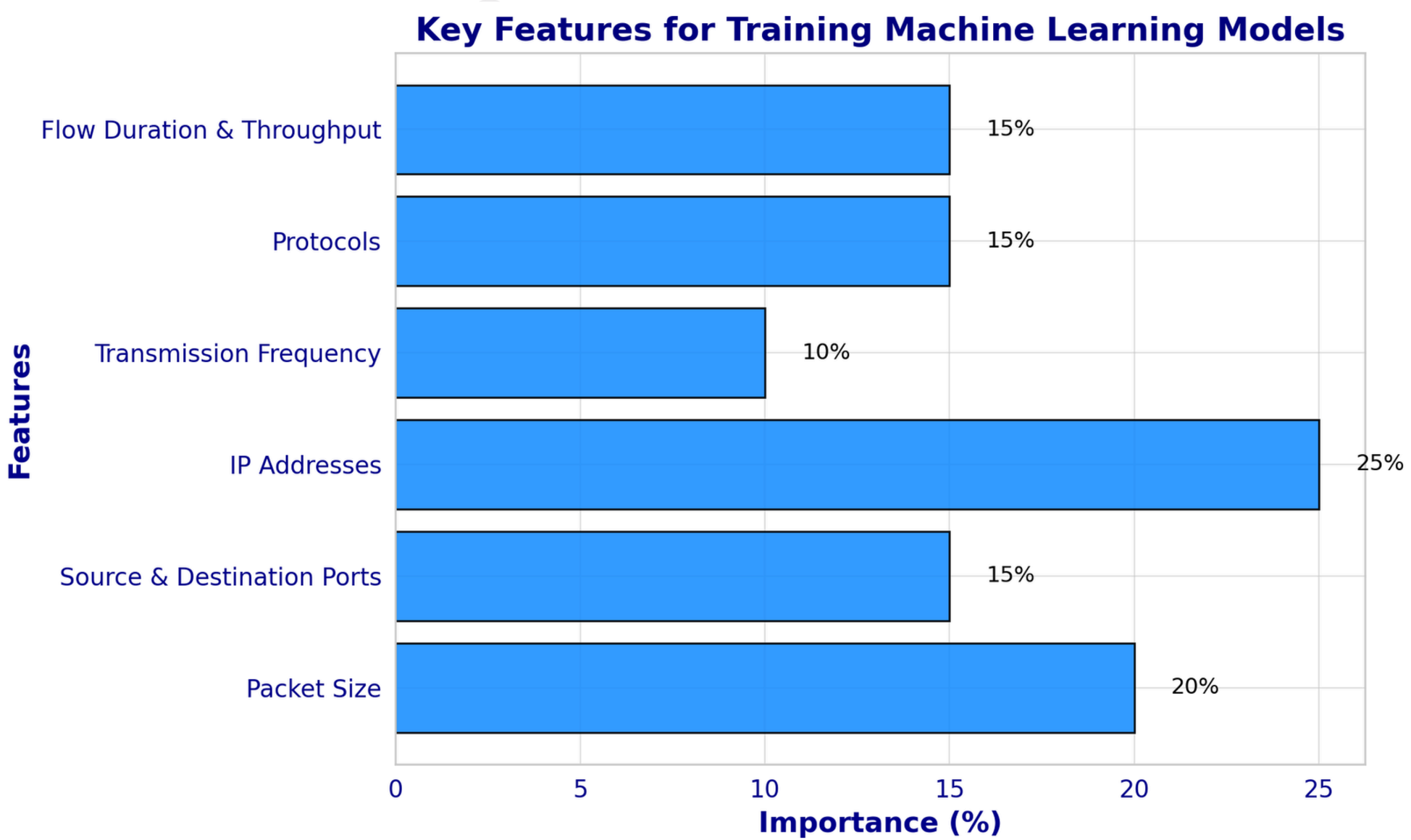




Key Features for Training

- **Packet Size:** Identifies unusual packet behavior.
- **Source & Destination Ports:** Tracks communication endpoints for suspicious activity.
- **IP Addresses:** Flags unknown or blacklisted addresses.
- **Transmission Frequency:** Detects traffic spikes and irregular patterns.
- **Protocols:** Differentiates between traffic types (e.g., TCP, UDP).
- **Flow Duration & Throughput**: Measures data flow consistency and volume.

Data was collected via Wireshark during OMNeT++ simulations and preprocessed with scaling, normalization, and balancing for effective model training.

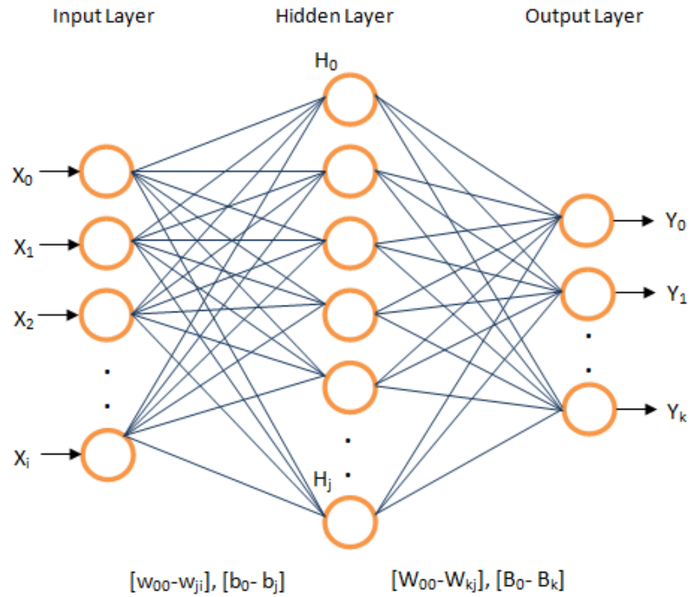

Key Features for Training Machine Learning Models
Features
Flow Duration & Throughput
Protocols
Transmission Frequency
IP Addresses
Source & Destination Ports
Packet Size
15%
15%
10%
25%
15%
20%
0
5
10
15
20
25
Importance (%)

Model Training and Evaluation

Model Implementation:

Utilized machine learning algorithms, including Logistic Regression, Random Forest, Decision Tree, and Multi-Layer Perceptron (MLP).

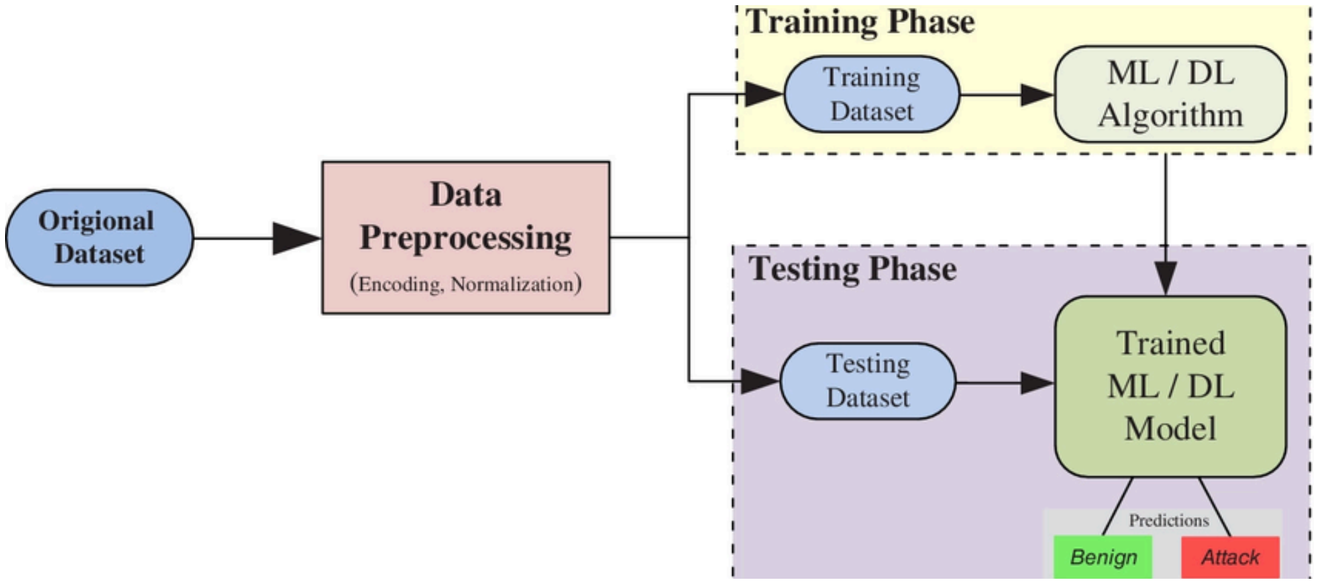
Training was conducted on **augmented datasets** to improve detection accuracy for anomalies.



Optimization:
Hyperparameter tuning was performed to refine models and improve performance metrics.

Results and pipeline

Random Forest outperformed other models with the highest accuracy (94%) and ROC-AUC (0.927). However, the ultimate goal is to enhance the Multi-Layer Perceptron (MLP) model, enabling it to achieve better accuracy and dynamically analyze more complex patterns in network traffic.



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

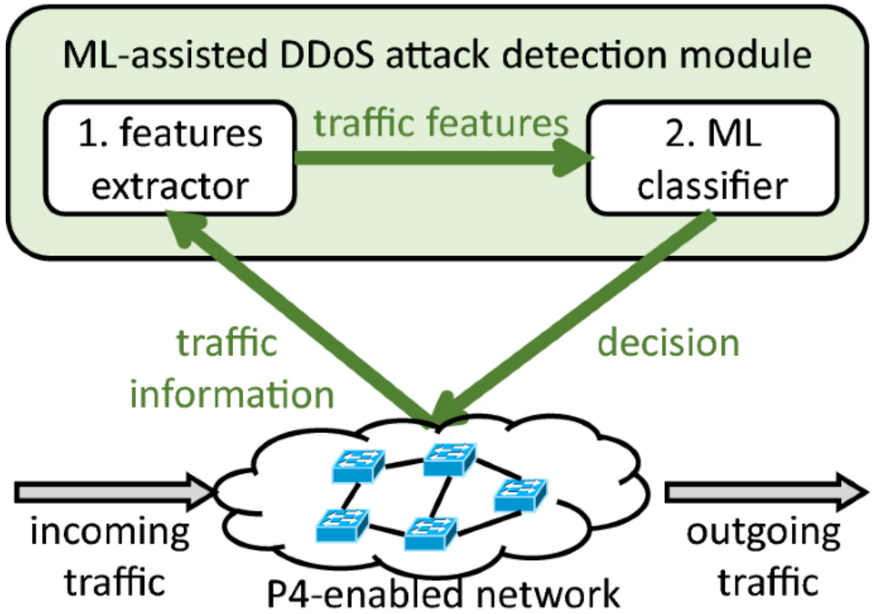
Future Work and Improvements

Goals for Enhancement

- Improving MLP Model:
- Enhance the Multi-Layer Perceptron (MLP) to improve accuracy, recall, and training efficiency.
- Dynamic Analysis Capability:
- Adapt the MLP to analyze complex traffic patterns and respond to real-time network variations.

Proposed Approaches

- Hyperparameter Tuning: Optimize configurations like learning rate and number of layers.
- Advanced Architectures: Explore CNNs or RNNs for better pattern recognition.
- Real-World Deployment: Implement the improved MLP model in live traffic monitoring systems



İTÜ

İSTANBUL TEKNİK ÜNİVERSİTESİ 1773

Thanks for watching!!!