Journal of Science and Technology
ISSN: 2362-1443 (Print) 3059-9490 (Online)
2024, Vol. 4, No. 2 (Special Issue), 14-23
DOI : https://doi.org/10.3126/jost.v4i2.78944

Flow based Network Threat Detection using ANN and Socket in Real Time

Bishal Poudel
Nepal College of Information Technology
bishal.191318@ncit.edu.np

Aishu Gyawali
Nepal College of Information Technology
aishu.191306@ncit.edu.np

Roshan Chitrakar*
Nepal College of Information Technology
roshanchi@ncit.edu.np

* Corresponding author

Article History:
Received: 29 July 2024
Revised: 24 August 2024
Accepted: 17 December 2024

Keywords— Network threat detection, DDoS attack,Long short-term memory,Socket, Real- time.

Abstract*—Cyber attacks are increasingly frequent and sophis-ticated, posing severe risks to both individuals and organizations worldwide. Traditional rule-based or signature-based intrusion detection systems have proven inefficient in keeping up with the evolving tactics of attackers, leavingnetworks vulnerable. Our research focuses on improving the detection of network threats,with a specific focus on Distributed Denial of Service (DDoS)attacks for testing, through advanced deep learning techniques. Specifically, we utilize a Long Short-Term Memory (LSTM)neuralnetwork to detect attacks by analyzing network traffic in real time. We begin by collecting a complete dataset consisting of normal traffic and network attack scenarios including DDoS attacks. This data is carefully preprocessed using normalization and feature extraction to ensure accuracy. The LSTM model was then designed, trained, and tested, while tuning its parameters for optimal performance. The model was evaluated using metrics such as accuracy, precision,recall,and the F1 score. In real-time scenarios with socket communication, it demonstrated a total average delay of 86.2678 seconds and a processing delay of just 0.1102 seconds, reflecting its performance and efficiency in deployment. The LSTM model demonstrated strong perfor-mance, detecting DDoS attacks with an accuracy of 99.897%. It effectively identified attacks, achieving a recall of 99.953% and a precision of 99.794%,resulting in an overall F1 score of 99.874%. In practical scenario with Model 1, it detected 83.42% of DDoS attacks out of the total flow of 368. The LSTM's ability to capture temporal dependencies in network traffic makes it superior for complex attack detection compared to traditional methods. This study demonstrates the effectiveness of LSTM in enhancing real- time network security and addresses the limitations of rule-based systems.

I. INTRODUCTION

The increasing dependence of organizations on interconnected systems has, regrettably, amplified their vulnerability to cyberattacks. Among the various network threats, Distributed Denial of Service (DDoS) attacks stand out due to their ability to overwhelm servers with excessive malicious traffic, preventing authorized users from accessing vital services. These attacks have the potential to seriously interrupt vital services, resulting in monetary losses and harm to one's reputation.

The recent research–hackers target computers connected to the internet every 39 seconds [1], shows that the cyber attack and its complexity are increasing day by day. To detect such an attack, flow based detection system instead of rule or signature based plays a significant role. As the attacks are very dangerous and can destroy facilities, data center and network resources, it is important to detect such an attack in real time.

While traditional Intrusion Detection Systems (IDS) are designed to monitor network activities for malicious behavior, most of these systems rely on signature-based detection. This approach, although effective for known threats, struggles to keep up with the constantly evolving tactics of attackers, especially when it comes to real-time detection of new or emerging threats. Recent research proposes the detection of network threat based on the flow of packet which seems great advancement on the intrusion detection.

Our research builds on this foundation and takes it a step further by applying deep learning in a new way. Specifically, we use a Long Short-Term Memory (LSTM) neural network to detect DDoS attacks in real time. The use of LSTM is particularly well-suited for this task, as it excels at processing sequential data and retaining information over long periods. This capability allows the model to identify subtle traffic changes indicative of ongoing attacks, even as attack strategies evolve. Thus it allows us to detect more complex, multi-stage attacks with greater accuracy.

Moreover, our system has been evaluated in practical, real-time deployment scenarios, showcasing its ability to efficiently process network data with minimal delay, thereby providing timely detection of DDoS attacks. This research doesn't just introduce a more effective way to detect threats; it also shows how well the system works in real-world settings, where quick and accurate threat detection is essential.

II. LITERATURE REVIEW

In order to help to develop the proposed system, an extensive literature review has been conducted.

A. Related Research Work

Throughout the course of time, there have been many tries for the development of integrated tools having capability of IDS, IPS and SEIM. One of the important attempts can be considered by Muhammad et al. [2]. They integrated IDS and SEIM for live analysis using machine learning techniques. In this project they have used ELK stack for SEIM functionality, Zeek for IDS and Slip for machine learning analysis. The

proposed system by them was tested using Denial of Service (DoS) test with 344.1/sec packet.

ELK is the stack that comprises three popular projects: Elasticsearch, Logstash, and Kibana. It provides the search and analytics engine, data ingestion, visualization and thus SEIM capabilities. It can be considered as an open source alternative for SEIM [3]. Zeek on the other hand works as an IDS, It can create different log files based on the flow of traffic such as conn.log, dns.log, http.log etc. Which helps for data feeding to the next stage, Slips. Which is used for real time packet analysis using machine learning algorithm Support Vector Machine and some behavioral techniques.

In 1980 the idea of intrusion detection was first suggested [4]. The goal of detecting attacks is to accurately recognize suspicious activity on the network For intrusion detection in 1998 Snort has developed by Martin Roesch,as a versatile significant tool in network security for two years. It is rule based detection mechanism. Snort is based on the packet capture library (libpcap), a system-independent interface for capturing traffic that is widely used in network analyzers [4]. However, a popular NIDS, relies on static rules and manual management, limiting its ability to adapt to new threats.

Similarly Suricata was developed which identifies threats by comparing network traffic patterns to pre-defined signatures of malicious activity [5]. This approach effectively detects known attacks but might struggle with zero-day vulnerabilities [5]. Suricata leverages multi-threading architecture for efficient processing of high-volume network traffic, making it suitable for demanding network environments [6]. Suricata, relies on static rules and manual management, limiting its ability to adapt to new threats. Similarly, Splunk is the world's leading operational data intelligence platform [7]. This platform is used to search, analyze and visualize the machine-generated data gathered from the websites, applications, sensors, devices etc. which make up IT infrastructure and business.

B. Artificial Neural Network (ANN)

Traditional Recurrent Neural Networks (RNNs) have difficulty capturing long-term dependencies in sequences because of the vanishing or exploding gradient problem [8]. This limitation makes RNNs less effective in tasks involving long sequences.As a result LSTM was introduced by Hochreiter and Schmidhuber [9] in 1997, designed to avoid the long-term dependency issue, unlike RNN, LSTM can remember data for a long period. LSTMs use an advanced gating mechanism to solve the vanishing gradient issue. Long-term dependencies within sequences can be learned and retained by LSTMs due to these gates, which regulate information flow within the cell [10].

The principal component of LSTM is the cell state. To add or remove information from the cell state, the gates are used to protect it, using sigmoid function (one means allows the modification, while a value of zero means denies the modification.). We can identify three different gates .

C. Forget Gate Layer

The forget gate layer decides which information to delete from the cell state. It analyzes the input data and the previous hidden state to determine what information should be discarded, using a sigmoid function. The forget gate's operation is expressed as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

where σ denotes the sigmoid function, W_f is the weight matrix, h_{t-1} is the previous hidden state, x_t is the current input, and b_f is the bias term.

D. Input/Update Gate Layer

This gate decides which new information to store in the cell state. It consists of two parts: the input gate and a candidate value created by a tanh layer. The input gate determines which information will be updated using a sigmoid function, and the tanh layer generates a candidate vector to add to the cell state. The combined operation is given by:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

and

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

where i_t is the input gate, $\tilde{C}_t$ is the candidate value, and W_i, W_C, b_i, and b_C are weights and biases. The cell state is updated as follows:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{4}$$

E. Output Gate Layer

This gate determines the output based on the updated cell state. It uses a sigmoid function to decide which part of the cell state to output, followed by a tanh function to scale the output between -1 and 1. The operations are:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

and

$$h_t = o_t \cdot \tanh(C_t) \tag{6}$$

where o_t is the output gate, and h_t is the new hidden state passed to the next time step.

Research has shown several advantages of LSTM networks for intrusion detection systems (IDS). Unlike signature-based IDS that struggle with novel attacks, LSTMs excel at analyzing sequential network traffic data to identify patterns deviating from normal behavior, potentially indicating an attack [11]. This is because LSTMs can capture the temporal relationships within the data, a key factor in intrusion detection. Additionally, LSTMs can learn complex relationships between events spread out over time, which is essential for detecting sophisticated attacks that unfold in stages [12]. For instance,

an attacker might perform reconnaissance activities before launching an exploit attempt. LSTMs can identify the connection between these seemingly unrelated events and raise a red flag.

Furthermore, LSTMs can be trained on normal network traffic patterns to detect anomalies or potential intrusions when deviations from these patterns occur. This is particularly useful for zero-day attacks that lack predefined signatures [13]. By adapting to evolving attack methods, LSTMs offer a more robust approach to intrusion detection. Finally, LSTMs can automatically learn relevant features from raw network trafficdata, eliminating the need for manual feature engineering, a complex process required by traditional methods [9]. This simplifies the process of identifying malicious activity within network traffic.

Increasing amount of cyber attack, its complexity and somehow lack in the development of highly performing real time Intrusion detection and prevention system provides motivation to conduct research in this fields.

III. METHODOLOGY

To successfully demonstrate the result of the research, a system architecture is constructed as shown in Fig. 1. The proposed architecture contains packet analyzer, trained model development, communication module and user interface.

Packet analyzer captures real time network packets, extracts packet specific information from different layers of TCP/IP architecture, creates a flow session and extracts the final flow features from the flow just before the session expires. It somehow works as CICFlowMeter [14] but fulfills specific requirements to carry out this research. The details of the packet analyzer are explained by Fig. 2. This analyzer only captures real time packets and extracts relevant flow features which are used while developing the threat detection model. These features are explained below in the model development section.

Most important part of the research is to develop a deep learning model that successfully detects the attacks. Since network attacks such as DDoS attacks evolve throughout the time. So we need to use time series data to detect attacks, alternatively we need to use those AI models, which should consider time series data. Artificial Neural Network (ANN) with Long-Short Term Memory ie. The LSTM model plays an important role while dealing with such a dataset.

Another important factor is the dataset. There are various datasets such as CIC DDoS 2019, CSE-CIC IDS 2018, CIC IDS 2017, kyoto 2006+ etc. All of them have their importance while detecting network attacks. Particularly in this research, Intrusion detection evaluation dataset (CIC-IDS2017) [15] is used. Which itself is considered an IDS evaluation dataset.

A. About the dataset

CICIDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter [14] with labeled flows based on the timestamp, source, and destination IPs, source and destination ports, protocols and attack (CSV files).

CICFlowMeter can be used to generate bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence more than 80 statistical network traffic features such as Duration, Number of packets, Number of bytes, Length of packets, etc. can be calculated separately in the forward and backward directions.

B-Profile system [15] is used to profile the abstract behavior of human interactions and generates naturalistic benign background traffic. Abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols are built to develop this dataset.

The data capturing period started at 9 a.m., Monday, July 3, 2017 and ended at 5 p.m. on Friday July 7, 2017, for a total of 5 days. Monday is the normal day and only includes the benign traffic. The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. They have been executed both morning and afternoon on Tuesday, Wednesday, Thursday and Friday.

Among different attacks, this research takes DoS/DDoS attacks, carried out Wednesday in the data collection scenario to develop the threat detection model. We developed and tested the various models by considering different parameters such as sample size, no of epoch, model architecture, data scaling, data splitting etc.

B. Training the model

In the initial dataset, there were 84 columns. The overall flowchart for training the model is explained by Fig. 3. The following features were removed as part of the preprocessing step:

- Flow ID
- Source IP
- Source Port
- Destination IP
- Bwd PSH Flags
- Fwd URG Flags
- Bwd URG Flags
- CWE Flag Count
- Fwd Avg Bytes/Bulk
- Fwd Avg Packets/Bulk
- Fwd Avg Bulk Rate
- Bwd Avg Bytes/Bulk
- Bwd Avg Packets/Bulk
- Bwd Avg Bulk Rate
- Fwd Header Length.1

Network centric features are removed as they vary over the network. Other features are removed due to constant value, irrelevance, and duplication.

The timestamp plays an important role while creating temporal dependencies in the data items. So data are sorted in ascending order according to timestamp. After the removal of null and infinite data items, data are applied with and without scaling. For scaling MinMax scalar is used. Also the model is

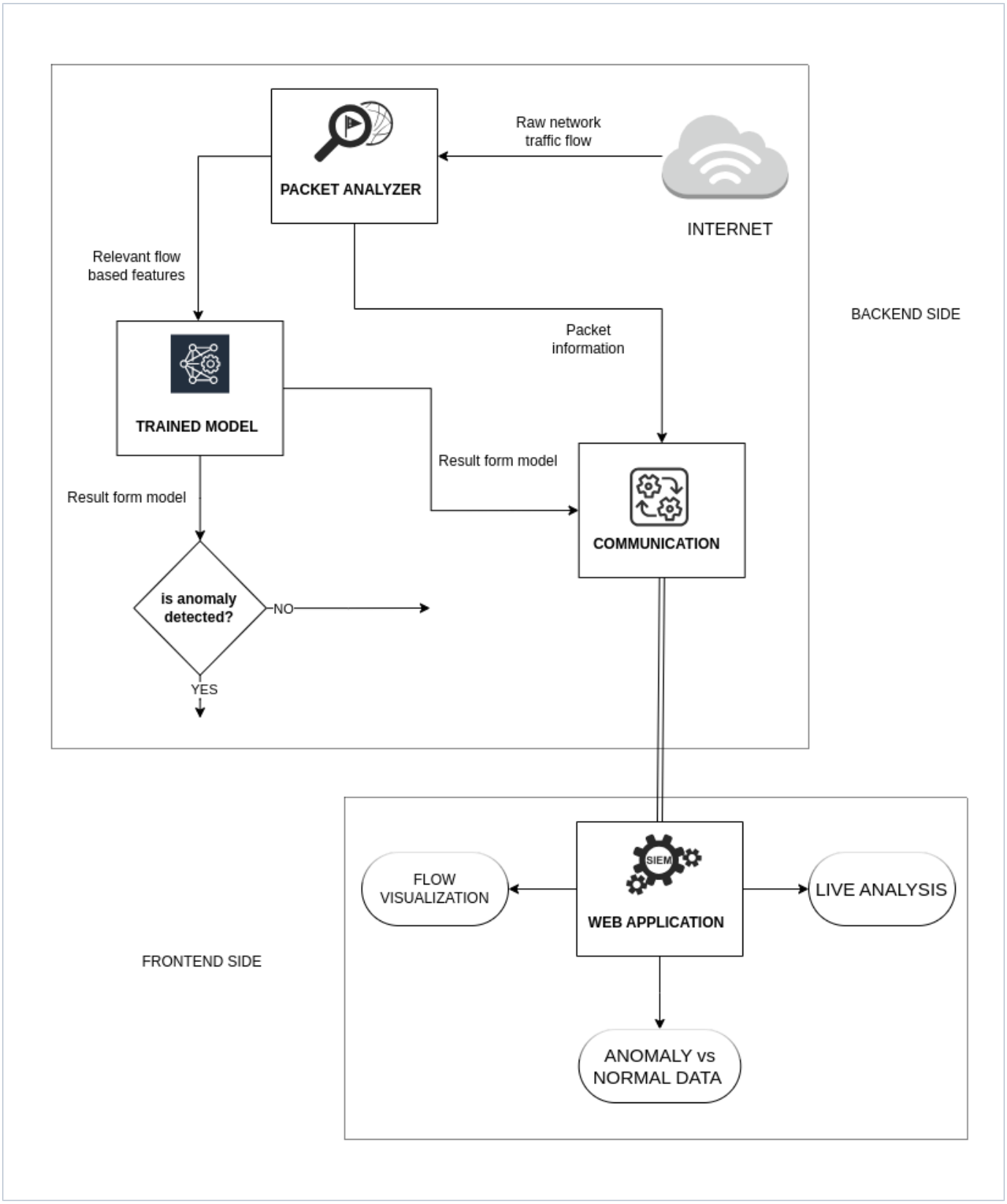


Fig. 1: Generalized system architecture

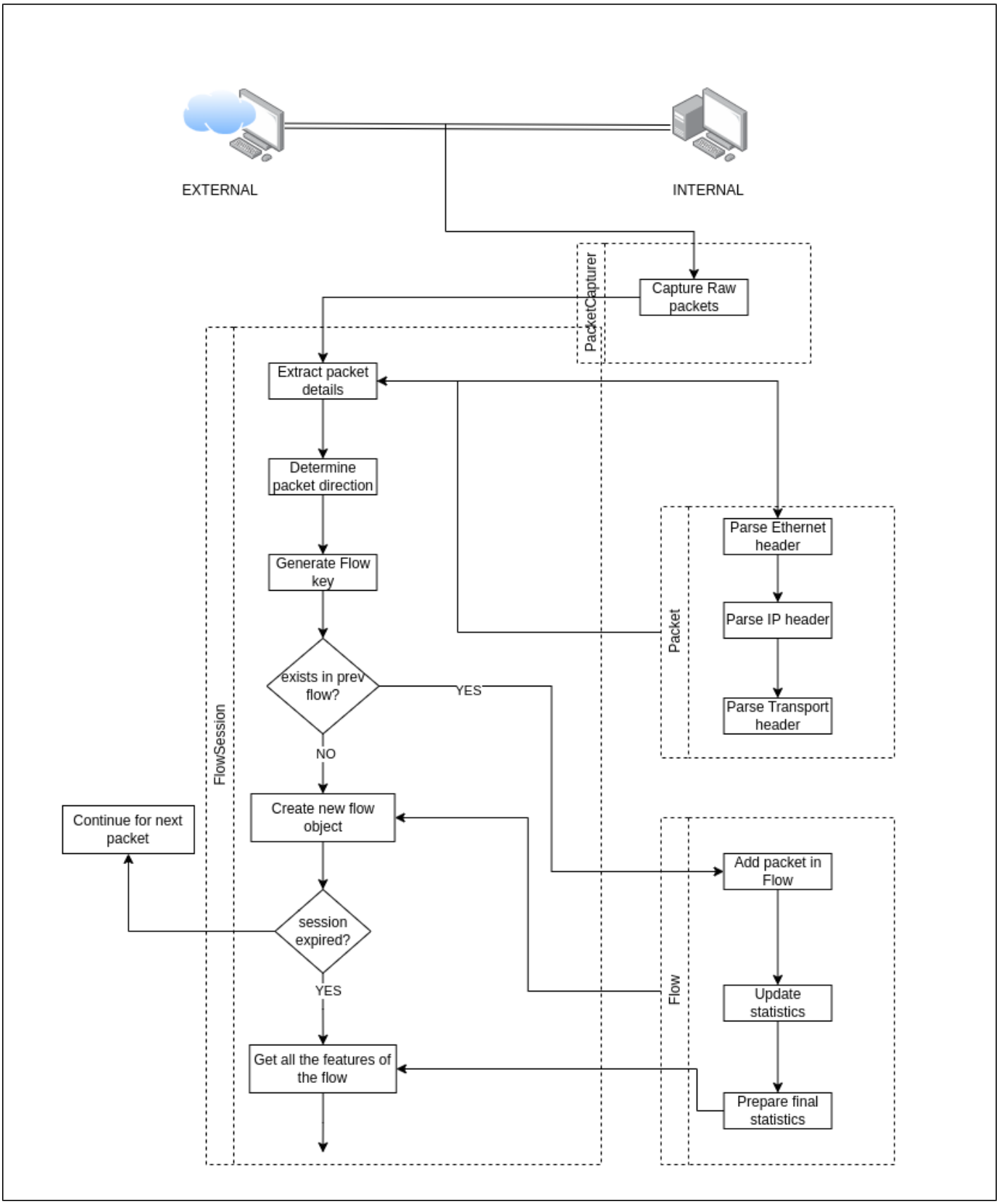


Fig. 2: Packet analyzer

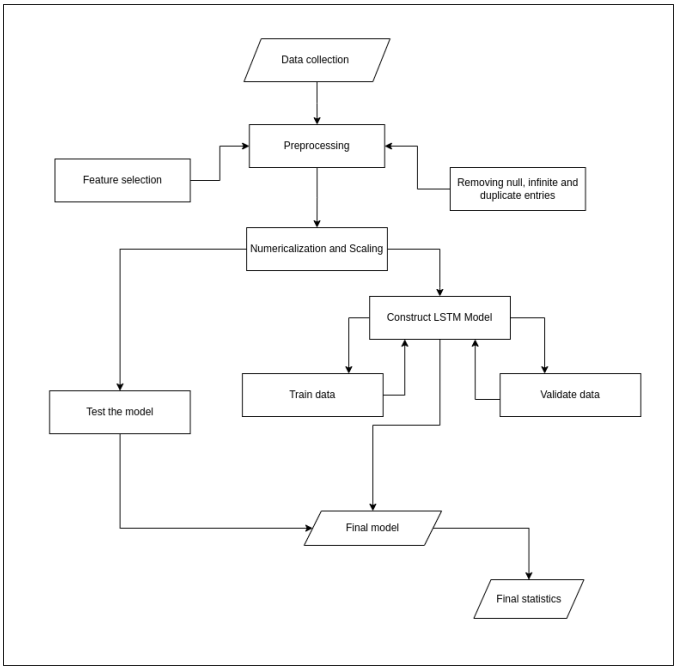


Fig. 3: Model training flowchart

trained with and without timestamp after creating a temporal sequence of 10 items. Finally 68 features are supplied to the LSTM model. Some cases early stop meahanism is applied to stay one better result.

In this research, we tested two model architectures, illustrated in Fig. 4(a) and 4(b). Architecture 1 is a basic model comprising a single Long Short-Term Memory (LSTM) layer, dropout, and two dense layers. It is designed to capture simple temporal patterns, making it suitable for less complex data sequences.

In contrast, Architecture 2 is a more advanced model that features two LSTM layers, dropout, and two dense layers. This architecture is capable of capturing more intricate temporal relationships, which is crucial for accurately detecting and responding to complex network threats.

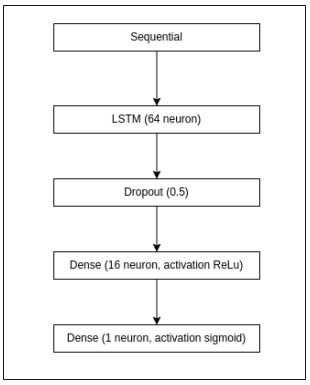
The main distinction between these two architectures lies in their complexity. The additional LSTM layer in Architecture 2 allows for a deeper understanding of temporal dynamics, enhancing the model's predictive capabilities. Moreover, Architecture 2 employs varying dropout rates across its layers, providing different levels of regularization. This strategy can significantly improve performance by mitigating the risk of overfitting, ultimately leading to more robust and reliable predictions.

C. *Evaluation of the models*

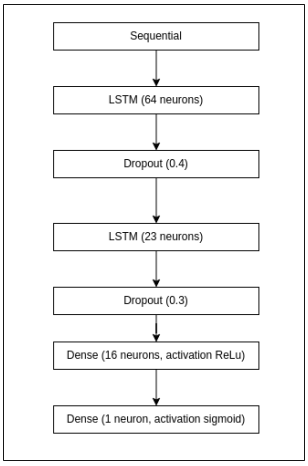
Result of various modes are shown below.

a) First model:

- Total data: 300000 (166286:133714)
- Training data: 191992
- Validation data: 47999
- Testing data: 60000



(a) Model architecture (1)

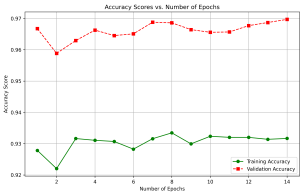


(b) Model architecture (2)

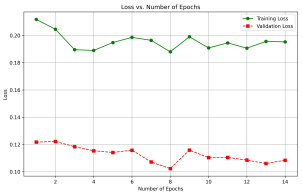
Fig. 4: Model architectures

- Model architecture: 1
- Epochs: 14
- Batch size: 64 (No scaling)

The parameters listed above were used to train the first model. The learning accuracy and loss for this model are shown in Fig. 5(a) and 5(b). The evaluation metrics are displayed as a confusion matrix in Fig. 6.



(a) Accuracy



(b) Loss

Fig. 5: Model learning curves (first model)

First model Metrics:

- Accuracy: 0.9545431814772216
- Precision: 0.9525436807584192
- Recall: 0.9972302365839585
- F1 Score: 0.9743748766667606

b) Second model:

- Total data: 300000 (166286:133714)
- Training data: 191992
- Validation data: 47999
- Testing data: 60000
- Model architecture: 1
- Epochs: 30
- Batch size: 64 (MinMax scaling)

The parameters mentioned above were used to train the second model. Fig. 7(a) and 7(b) show the model's learning accuracy and loss. The evaluation metrics are shown as a confusion matrix in Fig. 8.

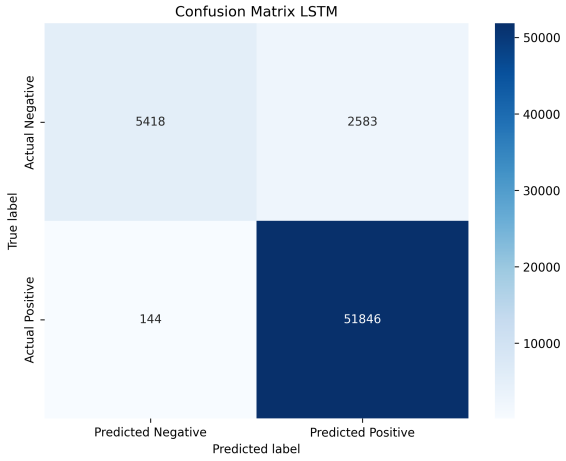
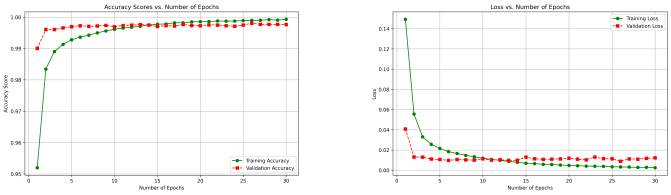


Fig. 6: Confusion matrix



(a) Accuracy (b) Loss

Fig. 7: Model learning curves (second model)

Second model Metrics:

- Accuracy: 0.9958160390725276
- Precision: 0.9964878610498032
- Recall: 0.9986920561646471
- F1 Score: 0.9975887410538451

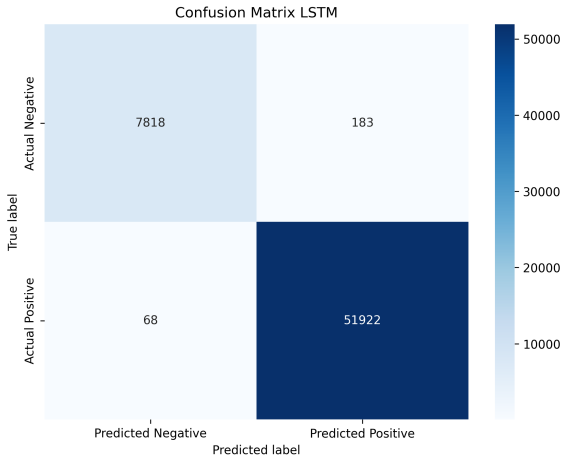
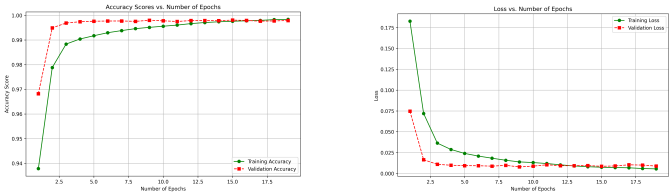


Fig. 8: Confusion matrix (second model)

c) Third model:

- Total data: 300000 (166286:133714 [attack:normal])
- Training data: 191992
- Validation data: 47999
- Testing data: 60000
- Model architecture: 2 (32 neurons in second LSTM layer)
- Epochs: 50 (Early stopping)
- Batch size: 64 (MinMax scaling)

The parameters listed above were used to train the third model. The learning accuracy and loss for this model are shown in Fig. 9(a) and 9(b). The evaluation metrics are displayed as a confusion matrix in Fig. 10.



(a) Accuracy (b) Loss

Fig. 9: Model learning curves (third model)

Third model Metrics:

- Accuracy: 0.9957326932373189
- Precision: 0.9960495531776167
- Recall: 0.9990382765916522
- F1 Score: 0.9975416762694937

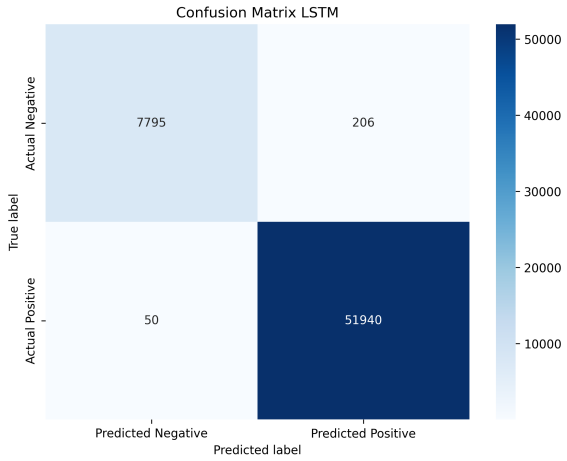


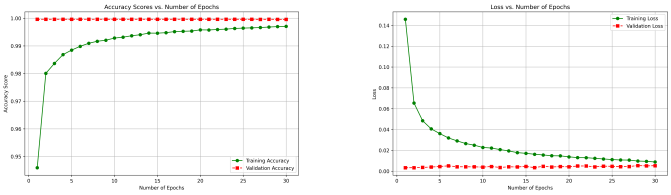
Fig. 10: Confusion matrix (third model)

d) Fourth model:

- Total data: 645286
- Training data: 105629 (120035:12011 for train-validation)
- Validation data: 26408
- Testing data: 512931 (369188:143743)
- Model architecture: 1
- Epochs: 30
- Batch size: 64 (MinMax scaling)

The parameters listed above were used to train the fourth model. The learning accuracy and loss for this model are

shown in Fig. 11(a) and 11(b). The evaluation metrics are displayed as a confusion matrix in Fig. 12.



(a) Accuracy (b) Loss

Fig. 11: Model learning curves (fourth model)

Fourth model Metrics:

- Accuracy: 0.9961066204998031
- Precision: 0.9980954121218937
- Recall: 0.9879924587632093
- F1 Score: 0.9930182392817561

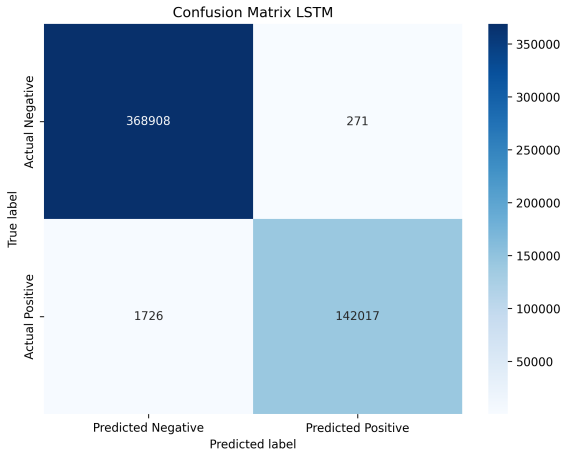


Fig. 12: Confusion matrix (fourth model)

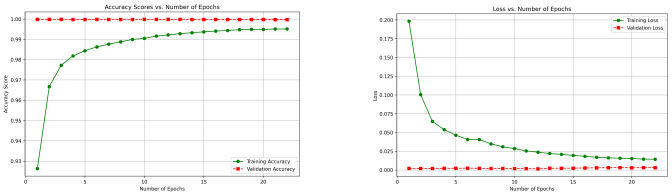
e) Fifth model:

- Total data: 645286
- Training data: 123204 (16000: 107204 [A:N])
- Validation data: 26408 (11:52791)
- Testing data: 100000
- Model architecture: 2
- Epochs: 50 (Early stopping)
- Batch size: 64 (MinMax scaling)

The parameters listed above were used to train the fifth model. The learning accuracy and loss for this model are shown in Fig. 13(a) and 13(b). The evaluation metrics are displayed as a confusion matrix in Fig. 14.

Fifth model Metrics:

- Accuracy: 0.99897
- Precision: 0.9995346216964264
- Recall: 0.9979458084710946
- F1 Score: 0.998739583205864



(a) Accuracy (b) Loss

Fig. 13: Model learning curves (fifth model)

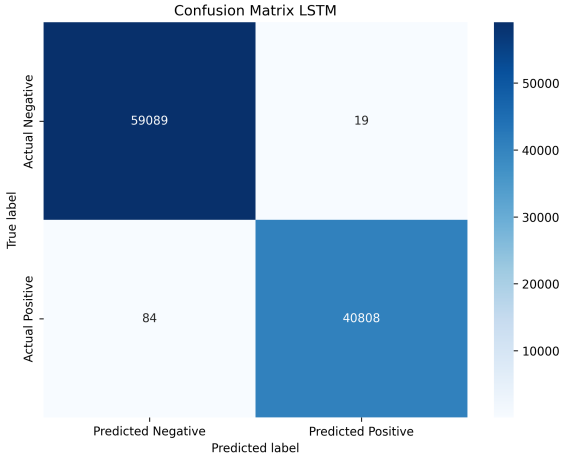


Fig. 14: Confusion matrix (fifth model)

D. Model deployment and testing

Once the development of model, best one is deployed in the architecture and tested for real time real traffic. Then the predicted result along with packet flow passed to the UI through websocket and provide good interface for testing the model. Fig. 15 illustrates the websocket architecture for the communication between sender and receiver.

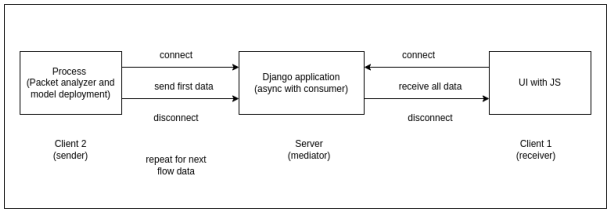
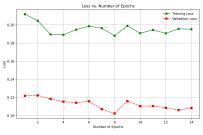


Fig. 15: Testing with real time packet flow

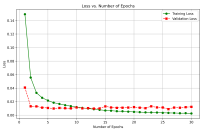
For the testing, we used a Dell laptop as the attacker machine. It had an Intel i5 8th Gen processor, 8 GB of RAM, and ran Kali Linux. We conducted DDoS attacks using HULK and SlowHTTPTest. The target was a virtual machine set up for the Mr. Robot CTF challenge, which ran on an Ubuntu server with 512 MB of RAM. This setup allowed us to effectively evaluate how well the model could detect attacks in real time.

IV. RESULT AND DISCUSSION

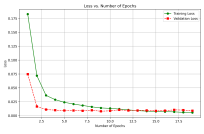
This section provide analysis of model and result after model deployment. we consider loss of model learning curve and evaluation metrix for the best selection among the five models. Fig. 16 displays the loss for all five models, while Fig. 17 provides a comparative analysis of accuracy, precision, recall, and F1 score for each model.



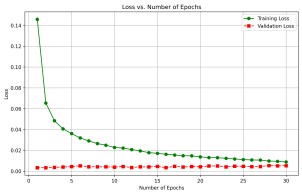
(a) Loss (model 1)



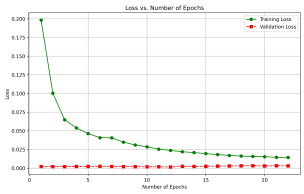
(b) Loss (model 2)



(c) Loss (model 3)



(d) Loss (model 4)



(e) Loss (model 5)

Fig. 16: Model learning curves (Loss)

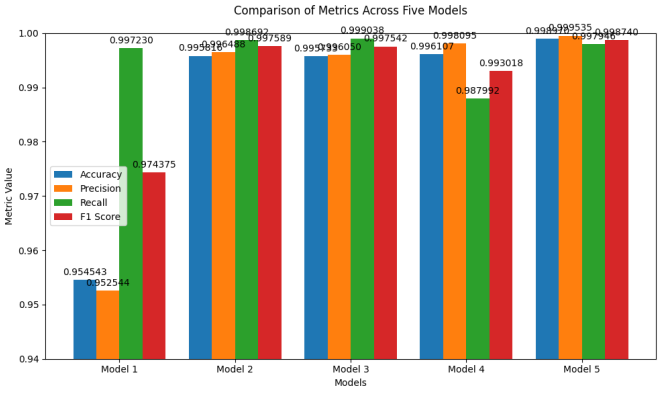


Fig. 17: Model matrics comparision

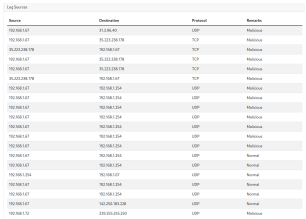
The evaluation revealed that Model 5 outperformed all other models in detecting Distributed Denial of Service (DDoS) attacks, achieving impressive metrics: 99.897% accuracy, 99.953% precision, 99.794% recall, and a 99.863% F1 score. Its architecture, which includes two LSTM layers, allows it to effectively capture complex patterns in time series data. Furthermore, thorough hyperparameter tuning and careful feature selection contributed to its robust performance, making it the most reliable choice for DDoS detection in this study.

In our real-time testing with Model 1, we observed an average total delay of 86.27 seconds, which includes a very short processing delay of just 0.11 seconds.

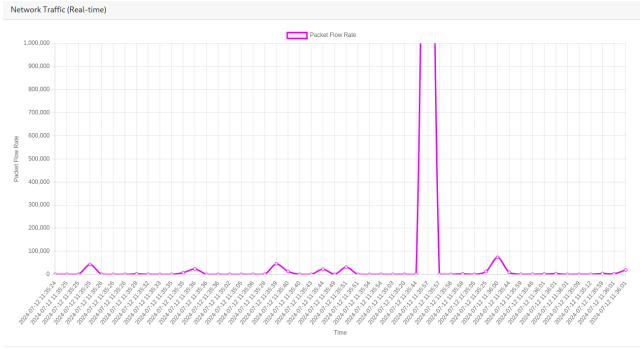
Notably, Model 1 was able to successfully identify 83.42% of DDoS attacks from a total of 368 flow sessions, showcasing its effectiveness in real-world traffic scenarios.

The output after prediction and detection is shown in Fig. 18. Here Fig. 18 (a) is the bar chart of the number of flow sessions counted as attack and normal. Fig. 18 (b) provides the log of communication and Fig. 18 (c) shows the network traffic flow in bytes per second.

(a) Attack vs normal traffic flow



(b) Log sources



(c) Network traffic flow

Fig. 18: Real-time testing results

V. CONCLUSION AND FURTHER WORKS

This research demonstrates significant results in network threat detection majorly in DDoS attacks. Furthermore, above provided result reflects the model's impressive efficiency and effictiveness when deployed in real-world conditions. It also shows that the sequential model with the LSTM layer is good for time series data.

However, this research does not cover all types of network threat detection, leaving room for future work to expand itsscope to include other threats. Once an attack isdetected, a key response is to prevent traffic flow, a mechanism known as intrusion prevention, which can also be further developed. Additionally, in real-world implementations, it is essential to design a robust architecture with low latency. The proposed design can be easily scaled and enhanced to improve effectiveness as needed.

REFERENCES

[1] University of North Georgia. ”Cyber Operations.” [Online]. Available: https://ung.edu/cyber-operations/index.php [Accessed: April 19, 2024].

[2] Muhammad, Adabi & Sukarno, Parman & Wardana, Aulia. (2023). Integrated Security Information and Event Management (SIEM) with Intrusion Detection System (IDS) for Live Analysis based on Machine Learning. Procedia Computer Science. 217. 1406-1415. 10.1016/j.procs.2022.12.339.

[3] Vazão, Ana, et al. ”SIEM open source solutions: a comparative study.” 2019 14th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2019.

[4] ”How Snort Works,” CrowdStrike, Available: https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/snort-rules/ ,[Accessed :14 Apr, 2024].

[5] J. Nderson, “Computer Security Threat Monitoring and Surveillance, Rapport Technique”, Fort Washington, Pennsylvania, 1980.

[6] Šelšek, Andrej, et al. ”Suricata review and attack sceneries.” 2014.

[7] Subramanian, Karun, and Karun Subramanian. ”Introducing the Splunk Platform.” Practical Splunk Search Processing Language: A Guide for Mastering SPL Commands for Maximum Efficiency and Outcome (2020): 1-38.

[8] Long Short-Term Memory networks - Fundamentals of Deep Learning [online course]. Coursera. Retrieved July 20, 2024, from [website hosting Coursera deep learning courses ON deeplearning.ai]

[9] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735-80

[10] Houdt, G., Mosquera, C., & Nápoles, G. (2020, May 13). A review on the long short-term memory model. https://link.springer.com/article/10.1007/s10462-020-09838-1

[11] Intrusion detection systems using long short-term memory (LSTM) Journal of Big Data [journal of big data ON Springer journalofbig-data.springeropen.com].

[12] Yu, Y., Li, H., Xu, Z., Li, X., Zhou, Y., & Zhao, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Computation, 31(7), 1239-1280. https://www.researchgate.net/publication/320970831_Recurrent_Neural Network_Architectures

[13] LBDMIDS: LSTM Based Deep Learning Model for Intrusion Detection Systems for IoT Networks - arXiv [arxiv.org]

[14] CICFlowMeter, 2017. https://github.com/ISCX/CICFlowMeter.

[15] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.