SMU Data Science Review

Volume 3
Number 3 *Fall 2020*

Article 8

2020

Flow-Based and Packet-Based Intrusion Detection Using BLSTM

Brook Andreas
Southern Methodist University, bandreas@smu.edu

Jayaweera Dilruksha
Southern Methodist University, jdilruksha@smu.edu

Eric McCandless
Southern Methodist University, emccandless7@gmail.com

Flow-Based and Packet-Based Intrusion Detection Using BLSTM

[1]Brook Andreas, [1]Jayaweera Dilruksha, [1]Eric McCandless
[2]Shaibal Chakrabarty, [3]Omar Youssef
[1] Master of Science in Data Science, Southern Methodist University,
Dallas, TX 75275 USA
[2]AT&T, 208 S. Akard Street,
Dallas, TX 75202 USA
[3]Arab Academy for Science and Technology (AAST),
Cairo Egypt
[1]{bandreas, jdilruksha, emccandless}@smu.edu
[2]shaibalc@smu.edu
[3]dr_yaser-omar@yahoo.com

Abstract. Networks are always under the threat of malicious intrusions. Deep learning models are used to help identify and mitigate intrusions before damage can occur. Various types of deep learning models have been researched, built, and tested with the goal of improving intrusion detection and efficiencies. In this paper, a two-phase deep learning approach called a Hybrid Intrusion Detection System (HIDS) is proposed that uses Bi-Directional Long Short-Term Memory Neural Network (BLSTM) to assess both flow-based network data and packet-based data. This approach is unique because BLSTM is employed rather than a traditional Deep Neural Network (DNN) and two models are used to assess both flow-based and packet-based data, whereas typically only one type of data is assessed. The two models were tested using the UNSW-NB15 dataset and performance was evaluated using accuracy, precision, recall, and F1-measure. Accuracy of the models was compared to results generated using DNN models. The BLSTM flow-based model achieved an accuracy of 96% compared to 93% using DNN. However, the BLSTM packet-based model achieved 76% accuracy, which is slightly lower than 81% using DNN. The results suggest that BLSTM is more effective in predicting flow-based data, but DNN is more effective in predicting packet-based data. Future work will be to improve the BLSTM packet-based model so that it is better than or comparable to DNN. Once this is achieved, analyzing both flow-based and packet-based data in a hybrid fashion using BLSTM could provide an extra layer of reliable protection if built in a cascaded scenario.

1 Introduction

Computer networks are an invaluable asset within organizations that allow for seamless communications and the transfer and storage of critical information. Unfortunately, networks are always vulnerable to various types of evolving security intrusions by malicious attackers. Data breaches exposed 4.1 billion records in the

first half of 2019 [1]. Therefore, it is imperative that organizations have a flexible and adaptive Intrusion Detection System (IDS) in place to be able to quickly identify malicious traffic before theft or damage can occur. An IDS is not to be confused with a firewall. A firewall is typically a software program in a networked environment with a preset security policy; hence it does not possess an automated ability to search and identify evolving threats and anomalies. IDSs act as the first line of defense before the firewall and works in unison with the same purpose.

Organizations are investing heavily in deep learning models that act as IDSs to mitigate the risk. Worldwide spending on cybersecurity is predicted to reach $133.7 billion in 2022 [2]. Several well-known companies tap into this ever-increasing market by offering off-the-shelf purchasable solutions, including Axtent's Intruder Alert, Cisco's NetRanger, Computer Associate's E-Trust, ISS's ReaSecure, and Martin Roesch's SNORT [3]. These solutions utilize various models and algorithms. Research, development, and testing of the next generation of deep learning models continue. However, in order to develop new deep learning models, it is important to have a fundamental understanding of the different types of attacks and IDSs.

Network attackers' motive is typically ransom; however, they are just seeking to disrupt or cause damage in some cases. There are several types of malicious network intrusions that organizations need to be aware of and protect against. Malware attacks include spyware, ransomware, viruses, and worms that seek to destroy or gain unauthorized access to a network via users clicking an email, link, or downloadable file. Denial-of-service attacks flood networks with traffic to exhaust resources and bandwidth and bog down the system resulting in an inability to complete legitimate requests. Man-in-the-middle attacks allow attackers to eavesdrop on two-party transactions to steal data. Structured Query Language (SQL) attacks inject malicious code into a server that uses SQL and forces the server to reveal data [4].

There are several ways to break down the many different variations of IDSs. First, there are signature-based and anomaly-based systems. Signature-based systems look for traffic predetermined as outside a set of rules and create an alert. Anomaly-based systems model regular traffic and create an alert when the traffic deviates from normal traffic characteristics [5]. Second, there are shallow machine learning and deep learning systems that use various algorithms to flag malicious or unusual activity. Shallow machine learning systems rely on handcrafted data features to detect intrusion. Deep learning systems typically have several advantages such as the flexibility to evaluate incomplete or distorted data, predict attacks in the form of a probability, and the ability to "learn" from characteristics of attacks [5]. Disadvantages include the complexity of some training methods, the need for large datasets for statistical accuracy, and the 'Black Box Problem," which refers to not understanding what representations are learned from the data [5]. Typical types of shallow learning models include Artificial Neural Network (ANN), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Naive Bayes, Logistic Regression (LR), Decision Trees, and Clustering. In general, deep learning systems are preferred and more effective than shallow learning systems because they can self–learn and derive new data features through fitting and generalization capabilities; however, they tend to require more computing time and resources. Deep learning systems include supervised models such as Deep Belief Network (DBN), Deep Neural Network (DNN), Convolution Neural Network (CNN), and Recurrent Neural Network (RNN)

and unsupervised models such as Autoencoders, Restricted Boltzmann Machine (RBM), and Generative Adversarial Network (GAN) [5]. Different types of modifications can be made to these models. An example of a modification pertinent to this proposed work is using a version of RNN called Long Short-Term Memory (LSTM). LSTM models are a modified version of RNNs that make it easier for the model to assess past data.

Additionally, flow-based and packet-based data can be analyzed to detect intrusion. Packet-based data represents the entire packet payload besides the header, while flow-based data represents the aggregated information of related packets of network traffic in the form of flow [6]. A flow-based record typically contains the IP network addresses of the hosts, network ports, network protocol, amount of data, and the time when the flow occurred, while packet-based data records contain the raw data packet information themselves. The flow-based model captures traffic flow data from the traffic passing inward and outward via the router ports, and then it is sent to a live database to tabulate and generate input data for the models. The packet-based model does not rely on third-party components to generate meta or summary information of the network traffic. Instead, all analysis is on anomalies from the packet payload [7]. Typically, one type of network data is used in an IDS depending on the network architecture, deployment cost, available data source, the time lag for capturing data, and anomaly source tracing [7].

Several metrics are utilized by modelers to measure the effectiveness of network IDSs. The accuracy metric assesses the ratio of correctly identified records to total records. Additionally, the precision metric assesses the ratio of true positive records to predicted positive records, recall assesses the ratio of true positive records to total positive records, and the F-measure assesses the harmonic average of precision and recall.

New and innovative deep machine learning approaches such as DNN and RNN are continually being explored and tested by organizations, technology suppliers, and academia to improve intrusion detection performance. These approaches must account for the type of intrusion to be detected, the variation of deep learning model needed, and the type of data analyzed. The contribution of this paper is focusing on detecting malicious activity by employing BLSTM models. While most approaches analyze either flow-based or packet-based data, this paper uses both in a two-phase deep learning model. The idea being that network administrators deploy a HIDS using BLSTM for both flow-based and packet-based models. Analyzing both flow-based and packet-based data in a hybrid fashion using BLSTM could provide an extra layer of reliable protection for networks beyond the firewall.

2 Background on BLSTM

It is important to have a background on the key aspects of the HIDS model proposed in this paper. Specifics of the model used in the analysis are discussed later in the Method section.

DNNs can be applied to problems where input is encoded with fixed dimensionality vectors. This causes significant limitations as many problems are best

explained with sequences whose lengths are not known in advance. Network traffic flow and packet flow are sequences of data flow, which poses a challenge for DNN. In network traffic and packet flow, the input and output cannot be fixed, so using a BLSTM [8] [9] [10] model to detect the traffic and packet flow is a significant improvement when compared to typical DNN.

LSTM models consist of three gates; the input gate, the forget gate, and the output gate. There is also a memory cell, which is the same as the hidden cell. This hidden state is designed to record additional information.

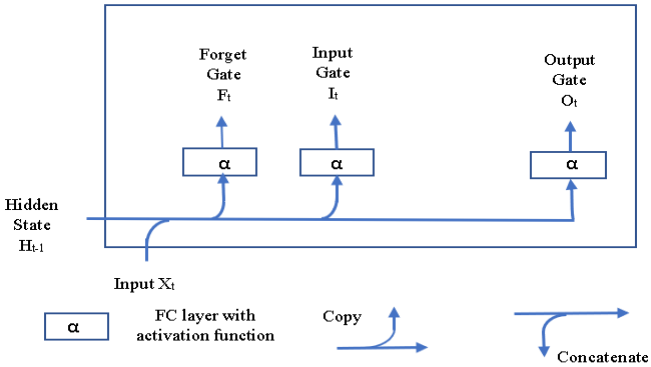


Fig. 1. Calculation of input, forget, and output gates in an LSTM.

The network traffic flow data feeds into the LSTM gates with the current timestamp $X_{t,}$ and a hidden state of the previous timestamp is H_{t-1}. A Sigmoid activation function with a completely connected layer is used to compute the values for the input, forget, and output gates. The output value of these three gates ranges from 0 to 1, as illustrated in Figure 1. If there are h hidden units of size n and the number of inputs is d, the inputs can be expressed as $X_t = \in R^{n*h}$ and calculated using the formulas below:

$$I_t = \alpha(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \quad \textbf{(1)}$$
$$F_t = \alpha(X_t W_{xf} + H_{t-1} W_{hf} + b_f). \quad \textbf{(2)}$$
$$O_t = \alpha(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \quad \textbf{(3)}$$

Where W_{xi}, W_{xf}, $W_{xo} \in R^{d\times h}$ and W_{hi}, W_{hf}, $W_{ho} \in R^{h\times h}$ are weight parameters and b_i, b_f, $b_o \in R^{1\times h}$ are bias parameters.

The candidate memory cell is shown in Figure 2. The $C\sim_t \in R^{n\times h}$ computation is the same as the three gates described above, and it uses the **tanh** function with a value range of -1 to 1 as the activation function. This generates the equation below with timestamp t.

$$C\sim = \textbf{tanh}(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \quad \textbf{(4)}$$

$W_{xc} \in R^{d \times h}$ and $W_{hc} \in R^{h \times h}$ are weight parameters and $b_c \in R^{1 \times h}$ is a bias parameter.

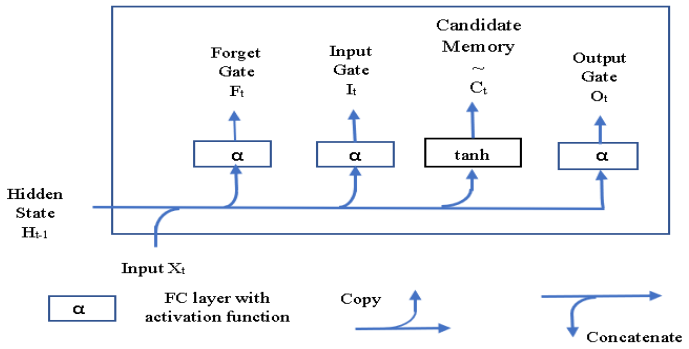


Fig 2. Computation of candidate memory cells in LSTM.

LSTM has two parameters. I_t controls how much new data is included via $C\sim_t$ and F_t is the forget parameter that addresses how much of the old memory cell content $C_{t-1} \in R^{n \times h}$ is retained. The following equation is derived using the same pointwise multiplication.

$$C_t = F_t \odot C_{t-1} + I_t \odot C\sim_t \quad \textbf{(5)}$$

When the forget gate value is approximately 1 and the input gate is approximately 0, the past memory cell C_{t-1} is saved and moves to the current timestamp. This design solves the vanishing gradient issue and efficiently captures the dependencies for time-series with long-range dependencies.

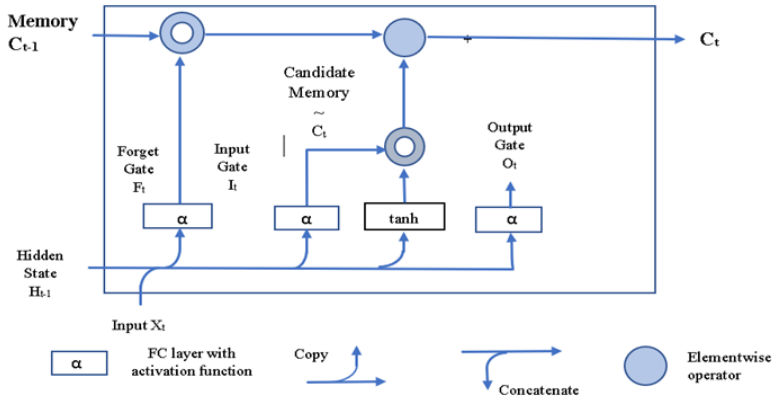


Fig 3. Computation of memory cells in an LSTM. Multiplication is carried out elementwise.

The hidden state is computed by $H_t \in R^{n \times h}$ and the output gate is activated. The LSTM is a gated version of the **tanh** for the memory cell and this ensures the value of H_t is always in the range of -1 to 1. When the output gate is 1 the memory information

is passed through the predictor. If 0, it retains the memory information within the cell and halts further processing. Figure 4 illustrates this scenario.

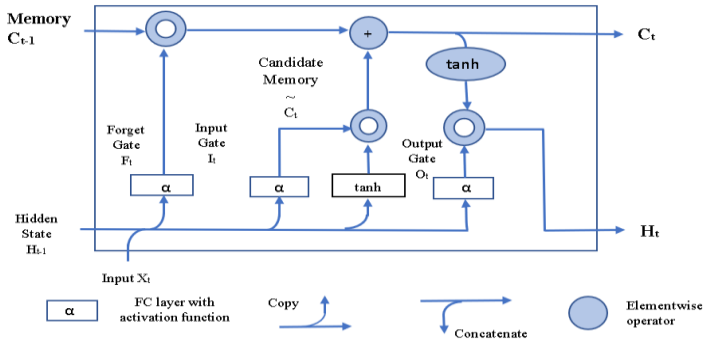


Fig 4. Computation of the hidden state. Multiplication is elementwise.

BLSTMs [11] are ideal for network traffic classification because they can learn fast and propagate more information. BLSTMs consist of two LSTM layers [12] running side by side, configured as forward pass and backward pass. The forward pass performs on a positive time dimension, and the backward pass performs on a negative time dimension. The final output is a concatenation of both passes. BLSTMs help accurately classify the network data flow pattern by running two passes. A single pass may not identify the anomalies of continuously flowing network traffic accurately. Having two passes enables the model to accurately classify and reduce the false positive rate and improve the ability to detect true positives. This is important as minor misclassification may increase the vulnerability of the entire network.

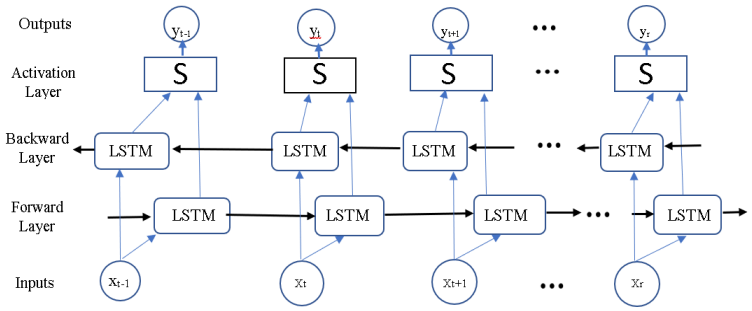


Fig 5. BLSTM Process.

Vulnerable devices sit outside the firewall, including Internet routers and switches. It is crucial to have filters on the router to prevent unauthorized users from logging in to the router; or sending management traffic to the router. Conventionally, this is prevented by deploying an Access Control List (ACL), which may be effective only for blocking the intrusions conducted by signature or internal attacks. The real-time novel attacks may pass through the router and the firewall as an authorized packet

without being detected. Adding ACL entry for every suspicious activity once worked as an effective technique for blocking unauthorized access; however, it creates massive ACL flooding with higher traffic networks.

Introducing an additional layer of deep learning-based HIDS over the traditional ACL and signature-based IDS to detect incoming and outgoing anomalies is an intelligent and efficient approach that will not compromise the productivity of a network. This hybrid approach consists of two deep learning models: 1) Flow-Based Intrusion Detection (FBID) that predicts a binary classification of traffic as attack or not attack and 2) Packet-Based Intrusion Detection (PBID) that predicts multiclass classification of traffic in terms of type of attack.

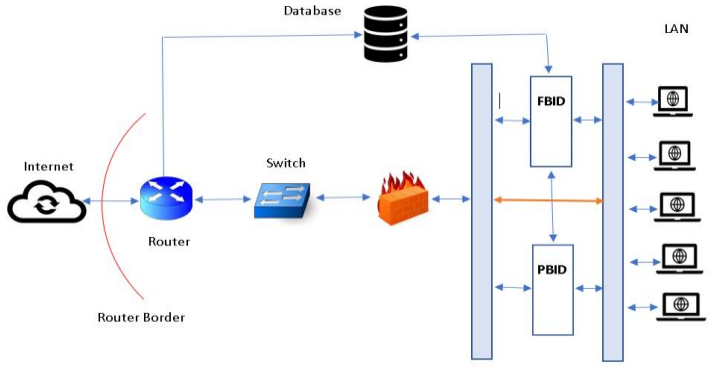


Fig 6. The Proposed Network Architecture for HIDS

3 Related Work on Deep Learning Methods

IDSs have been the subject of research on many fronts in the wake of cyber-attacks on corporations resulting in data loss and financial damage. IDSs are used for "the process of identifying and responding to malicious activity targeted at computing and networking resources" [13]. IDSs monitor any exploitation of the computing system either through the internet or intranet. The literature review for this paper focused on multiple journal articles related to different types of IDS approaches and machine learning features with an emphasis on approaches similar to the one proposed in this paper.

For example, research was done by Aslam et al. (2017) regarding hybrid Network Intrusion Detections Systems (NIDS) using a machine learning classification and rule-based learning system dual model to offset the high false positive rate generated from rule-based NIDS [14]. The method used in their paper employed a rule-based approach to identify incoming network packets as intrusions or normal packets and then used a trained model of machine learning classifiers with logistic regression as a further layer of validation. The final classification decision used in the approach was a gate logic of "OR." In their result, they were able to identify intrusion with a minimum false positive rate. The researchers used the KDD dataset with all associated features. However, the limitation of the paper was that the researchers

selected only a few features of the KDD dataset that were already pre-selected from the SNORT technique from the rule-based approach, such as protocol, flag, count, duration, and class.

In a paper written by Alaidaros et al. (2011) on the impact of high-speed networks on intrusion detection techniques using a flow-based and packet-based approach [15], the authors found that the packet-based NIDS processed every payload with high processing time, though it produced low false alarms. However, the flow-based approach had low processing time but suffered from high false alarms. Therefore, the recommended approach was to use a hybrid model to address the high processing time and high false alarm rate. This approach directly relates to this proposed paper's approach, except they used a machine learning approach to address the scenario. The article also showed how the NIDS performance and accuracy were affected by the threats and attacks within the high-speed network environment.

In a journal article written by Pramita et al. (2020) using LSTM-RNN to classify network attacks, findings showed deep learning as a reliable approach in IDSs with higher accuracy and distinctive learning mechanisms [16]. The researchers used the NSL-KDD dataset and RNN for optimal feature selection. They generated model measures such as accuracy, recall, precision, F-score, and confusion matrix with a binary classification accuracy of normal versus abnormal binary categories and multiclass categories (Normal, DoS, Probing, U2R, and R2L) sets. The researchers also used traditional ML models like support vector machine (SVM) and random forest (RF) as the benchmark comparison with the LSTM-RNN, which proved to be a robust classifier of intrusion features. They showed that the proposed model produced the highest accuracy rate of 96.51% and 99.91% for binary classification using 122 features and an optimal set of 99 features. The researchers did not show the impact of CPU or GPU performance as the number of neurons increases for the LSTM-RNN model using the refined dataset. This paper proposes a hybrid flow-based and packet-based approach using the same technique on a relatively newer dataset.

Yin et al. (2017) explored how to model an intrusion detection system based on deep learning and proposed an approach using recurrent neural networks (RNN-IDS) [17]. The authors studied the performance of the model in binary classification and multiclass classification along with the number of neurons and different learning rate impacts on the performance of the proposed model. They compared the results with ANN, random forest, support vector machine, and other machine learning methods proposed by previous researchers as benchmarks. The experimental results showed that RNN-IDS is very suitable for modeling a classification model with high accuracy and that its performance is superior to that of traditional machine learning classification methods in both binary and multiclass classification. The RNN-IDS model improves the accuracy of intrusion detection and provides a new research method for intrusion detection. The researchers used the NSL-KDD dataset generated in 2009 containing 41 continuous and symbolic features. The researchers used the NSL-KDD as the benchmark dataset, which effectively solved the inherent redundant records problems of the KDD Cup 1999 dataset and made the number of records reasonable in the training set and testing set that the classifier did not favor more frequent records. The proposed model achieved 97% accuracy on the testing dataset.

Kaur and Singh (2019) proposed hybrid intrusion detection and signature generation using deep recurrent neural networks and a deep learning-based system for

hybrid intrusion detection and signature generation of unknown web attacks referred to as D-Sign [18]. D-Sign used both a misuse detection approach and an anomaly-based detection approach. The method detects attacks, generates signatures, and updates the signature repository of IDSs proactively. Critical components of D-Sign include the honeypot server, Misuse Detection Engine (MDE), Anomaly Detection Engine (ADE), and Signature Generation Engine (SGE). The paper presented the idea of containing all intrusion features to be filtered and processed in the signature generation engine without human interference, prep the landscape for more classification tasks deployed, and update the signature repository of intrusion detection systems proactively. The ML technique used was RNN-LSTM. The approach rendered good accuracy results based on the NSL-KDD dataset in binary and multiclass classification. However, performance overhead was still an issue along with the vanishing gradient effect in the deep learning model. The true positive rate, false positive rate, precision, recall, F-measure, and ROC of the model generated reasonable values.

Hon et al. (2018) researched a deep learning approach for intrusion detection in the Internet of Things (IoT) using BLSTM-RNN and addressed the scope of RNN in intrusion detection [19]. A novel deep learning technique for detecting attacks within the IoT network using BLSTM-RNN was the core element. A multi-layer deep learning neural network was trained using the novel benchmark dataset UNSW-NB15. The researchers highlighted the binary classification of normal and attack patterns on the IoT network. The experimental outcomes showed the model's potential efficiency in this proposed paper in terms of precision, recall, and F-1 score. Their proposed BLSTM model achieved over 95% accuracy in attack detection. The experimental outcome showed that BLSTM RNN is highly efficient for building high-accuracy intrusion detection models and offers a novel research methodology. This paper showed the effectiveness of the model to detect intrusion in IoT network.

Kwon and Roy B. (2019) presented an overview of deep learning methodologies, including restricted Boltzmann machine-based deep belief network, deep neural network, recurrent neural network, and machine learning techniques relevant to network anomaly detection [20]. The authors used the 1999 KDD-Cup dataset and NSL-KDD datasets as training and testing datasets. Since the KDD-Cup 1999 dataset contains a considerable amount of redundant records, it makes the learning algorithm biased. To solve the issue, NSL-KDD, a new version of the KDD-Cup 1999 dataset, is widely adopted for anomaly detection. In the research, the authors pointed out the advantage of using deep learning models and dimensionality reduction techniques on a fully connected model over conventional ML methods such as SVM, random forest, and Adaboosting. Deep learning methods like RNN have shown better model benchmarks compared to the techniques mentioned above.

In a journal article written by Ralf (2015), the researcher applied LSTM-RNN to intrusion detection using the KDD dataset [21]. The modeling treated the network traffic as time-series data. The researcher also applied various network topologies to identify suitable LSTM-RNN network parameters. Using a configuration that deploys networks with four memory blocks, the author demonstrated that LSTM has a good learning rate compared to traditional ML methods for intrusion detection with better accuracy and computational learning cost. The paper's strength is that the researcher was able to produce a distinct time-series event while detecting DOS attacks and

network probes. The researcher measured the crafted model's performance in terms of mean-squared error, confusion matrix, accuracy, receiver operating characteristic (ROC) curve, and the corresponding AUC value. The research limitation was that feature selection from the dataset was minimized by focusing on DOS attacks and network probes. In the proposed paper, more features and attributes are included in the analysis and use the more recent and complete UNSW-NB15 dataset.

In a related journal article written by Sheikhan et al. (2012), intrusion detection using reduced-size RNN based on feature grouping was explored [22]. The researchers deployed a three-layer RNN architecture with categorized features as inputs and attack types as outputs. The intrusion features assessed were anomaly-based and misuse-based, where the attack types classified were Denial-of-Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R). Using 41 features of the KDD dataset, the researchers' model showed that RNN offers a better detection rate than other benchmark models. However, how a reduced size neural network or partially connected RNN model improves classification rate is not discussed in detail. RNN proved to be a robust technique in intrusion detection, even in more feature groups.

Li et al. (2018) used RNN and Restricted Boltzmann Machines (RBM) for Malicious Traffic Detection [23]. The paper addressed RBM as a hybrid solution for detecting intrusions using the ISCX-2012 and KDD dataset from an ever-increasing machine learning algorithm. The RBM allowed the researchers to take byte-level raw data as input without feature engineering. Additionally, the RBM model was used to extract the feature vectors of the network packets, and an RNN model was used to extract the flow feature vector. Finally, the flow vectors were sent to the SoftMax layer to obtain the detection result. The research emphasizes the use of temporal information between network packets in one micro-flow. The paper's hybrid solution claimed greater detection accuracy, recall rate, and lower false alarm rate. Though the paper did not address the computed overhead caused by the additional layer of the RBM model, it showed the RNN method's efficacy in packet-level intrusion detection.

Prior work on this topic was used to help develop a framework for the models used in this paper. One of the common features of the prior work examined was that most of the studies used similar datasets, such as KDD Cup 1999 and NSL-KDD. Additionally, only flow-based or packet-based data were analyzed. Different types of IDS scenarios were discussed, such as host-based/network-based, signature-based/anomaly-based, and packet-based/flow-based. This paper examines using a BLSTM approach for both flow-based and packet-based data to identify malicious network traffic.

4 Method

This section provides an overview of the dataset, exploratory data analysis, steps to prepare the data for the models, and the general model architecture.

4.1 Data

Several datasets were evaluated to test the proposed HIDS. The HIDS required a dataset for training and testing that allowed for the identification of both flow-based and packet-based intrusions and included data representing network attacks in addition to normal network traffic. The KDDCUP-99 [24] and NSL-KDD [25] datasets were considered, but the UNSW-NB15 dataset was chosen because it contained a more representative set of present-day attacks, normal network behavior data, payload and packet headers to represent the packets, and complete information.

The raw network packets in the UNSW-NB15 dataset [26] were created by the IXIA Perfect Storm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) in order to generate a hybrid of real modern normal network activities and synthetic contemporary attack behaviors [27]. The Tcp dump tool was utilized to capture 100 GB of raw traffic (e.g., Pcap files).

There were two million records in the UNSW-NB15 dataset. Partitions of 175,341 records for data training and 82,332 records for data testing were provided. The Argus and Bro-IDS tools were used to develop twelve algorithms and generate forty-seven features with class labels (see Table A in Appendix). The dataset had nine families of attacks and one family of normal traffic as shown in Table 1.

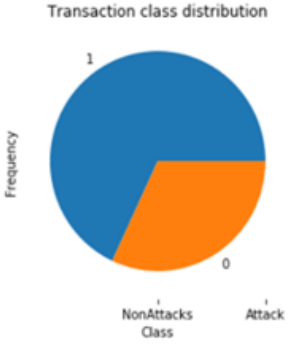
Table 1. UNSW-NB15 general summary of attack families.

Category	Training Set	Testing Set
Normal	56,000	37,000
Analysis	2,000	677
Backdoor	1,746	585
DoS	12,264	4,089
Exploits	33,393	11,132
Fuzzers	18,184	6,062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1,133	378
Worms	130	44
Total Records	175,341	82,332

4.2 Exploratory Data Analysis (EDA)

The UNSW-NB15 dataset was explored to understand the characteristics of the data and identify features that should be utilized in the models to identify network intrusion.

Figure 7 shows the initial breakdown of data within the UNSW-NB15 dataset in terms of records classified as attacks and normal. The analysis determined there were no duplicate records and no null values.



Name	Type	Description
attack_cat	N	Nine types of attack categories
Label	B	0 for normal data flow and 1 for attack records
Type: N: nominal, I: integer, F: float, T: timestamp and B: binary		

	label	count	percent
0	1	119341	68.062233
1	0	56000	31.937767

Fig.7. UNSW-NB15 label features.

The models used in this paper examine both flow and packet data. A packet undergoes flow-based processing after any packet-based filters have been applied to it. A flow is a stream of related packets that meet the same matching criteria and share the same characteristics [28]. Table 2 and Table 3 show key flow-based and packet-based features in the dataset.

Table 2. UNSW-NB15 flow-based features.

Name	Description
Srcip	Source IP address
Sport	Source port number
Dstip	Destination IP address
Dsport	Destination port number
proto	Transaction protocol

Table 3. UNSW-NB15 packet-based features.

Name	Description
Swin	Source TCP window advertisement
Dwin	Destination TCP window advertisement
Stcpb	Source TCP sequence number
Dtcpb	Destination TCP sequence number
Smeanz	Mean of the flow packet size transmitted by the src
Dmeansz	Mean of the flow packet size transmitted by the dst
trans_depth	the depth into the connection of http request/response transaction
Res_bdy_len	The content size of the data transferred from server's http service

4.3 Data Preprocessing

As part of data preprocessing, label encoding and hot encoding were explored to address the feature variables. Transformation and normalization of data were implemented to streamline the dataset for feature variable selection.

Additionally, a correlation heat map shown in Figure 8 was constructed to explore the correlation among variables within the dataset. This analysis was important to explore the effectiveness of feature selection on our models.

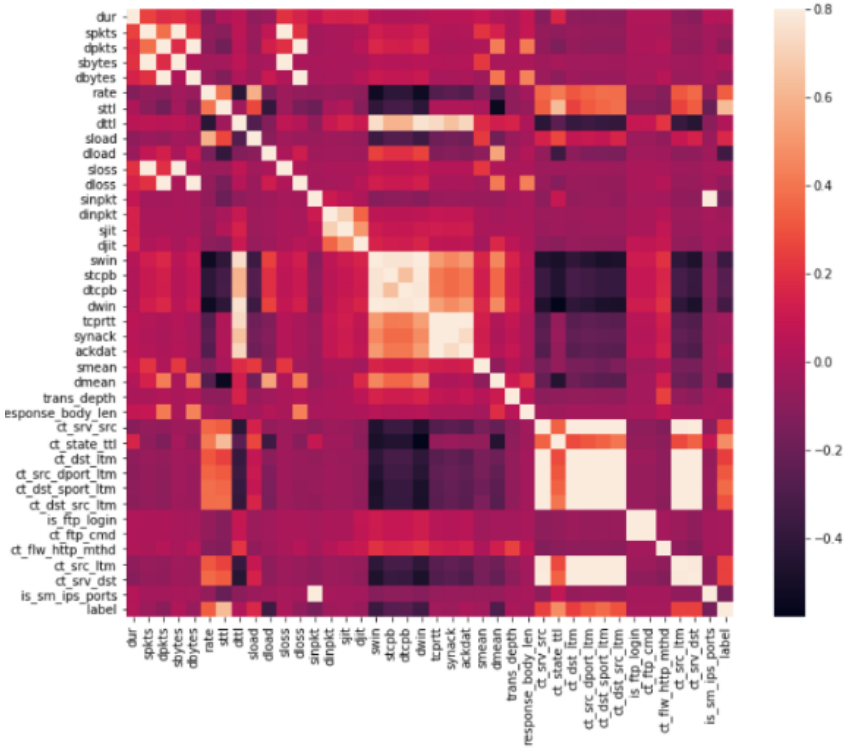


Fig 8. UNSW-NB15 Correlation heat map of variables.

The feature selection exploration identified several independent variables as not having a relationship with the dependent variables and the variables were subsequently removed from the models. Additionally, independent variables with low standard deviation were removed. These independent variables were removed because they have low prediction power. The dataset was reduced from 47 independent variables to 33 variables.

Feature reduction methods were implemented to further reduce the number of independent variables. Techniques explored to achieve this reduction were Decision Tree Classifier (DTC), Random Forest Classifier (RFC), and Extra Tree Classifier

(ETC) with a gini criteria. A Voting Classifier was used to combine these feature reduction techniques into a single model, which is typically stronger than any individual technique. Ultimately, this classification process reduced the number of independent variables from 33 to 10. Preliminary assessments of the FBID and PBID models using the reduced dataset produced very low accuracy scores. The decision was made to proceed with the 33 features.

4.3 Proposed Model

Two BLSTM models were implemented. A FBID model was implemented to predict whether network traffic is normal or abnormal. A PBID model was implemented to predict the type of attack. The models allowed output from the previous step to be fed as input to the current step.

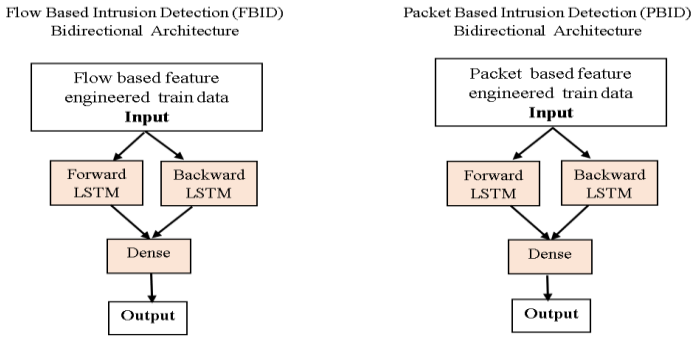


Fig.9. FBID and PBID models

The Keras Tensorflow framework [29] was used to train the FBID and PBID models individually. The original training dataset consisting of 175,341 training records and test dataset consisting of 82,332 test records were combined and then randomly split into 70% training data and 30% testing data.

The FBID model was used to predict binary classification (Attack, Not Attack). The PBID model was used to predict 10 attack classifications (Normal, 'Shellcode', 'Analysis', 'Fuzzers', 'DoS', 'Backdoor', 'Worms', 'Reconnaissance', 'Exploits', 'Generic'). Each model's performance was measured using accuracy, precision, recall, and F1-score.

5 Results

Results for the models proposed in this paper were initially generated using a Dell XPS personal notebook, which has an Intel Core i7-5200U CPU@2.6 GHz configuration, four core physical processor, a 32 GB memory, and Skylake GT2 GPU acceleration. The BLSTM models caused a performance bottleneck. As a result, the

models were run using Google Colab. The remainder of this section shows results for the BLSTM flow-based and packet-based models.

5.1 BLSTM Flow-Based Binary Classification (FBID)

Using the BLSTM framework to evaluate the model's performance in binary classification, the model used 33 independent variables. Data were trained using 30, 50, and 60 steps with 60 steps yielding the best results. Three hidden layers and 1 output layer were used with each hidden layer containing 50 neurons and a dropout rate of 0.2. The model was run through 30 epochs with a batch size of 32. The learning rate was 0.001. A 3D input array was fed into the model and utilized the ReLU activation function for the inner layers and Sigmoid for the outer layer. The model was optimized using the 'adam' optimizer and the binary crossentropy loss function was employed.

Evaluation of the model metrics show the final model accuracy achieved 96%. The weighted average precision, recall, and F1 achieved 96%, 96%, and 96%. Results are shown in Table 4.

Table 4. Model metrics for FBID binary classification.

	Precision	Recall	f1-score	Support
Attack	1.00	0.92	0.96	36757
Not Attack	0.93	1.00	0.96	40545
Accuracy			0.96	77302
Macro Avg	0.97	0.96	0.96	77302
Weighted Avg	0.96	0.96	0.96	77302

Figure 10 shows the corresponding confusion matrix. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

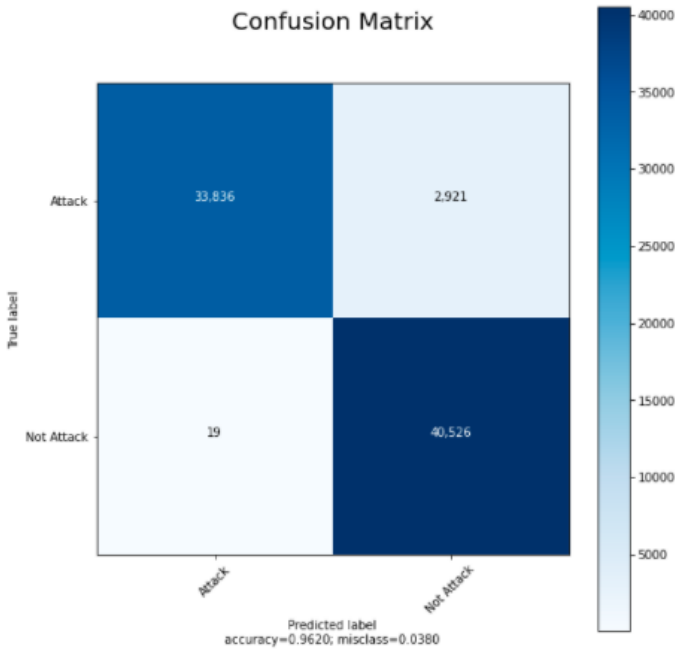


Fig.10. Confusion matrix for FBID binary classification model.

A DNN model was also run with similar parameters to compare to the performance of the BLSTM model. Results show that the accuracy of the DNN model was 93% compared to 96% for the BLSTM model.

5.2 BLSTM Packet-Based Multiclass Classification (PBID)

Using a similar BLSTM framework to evaluate the model's performance in multiclass classification, the model used 33 independent variables. Data were trained using 30, 50, and 60 steps with 30 steps yielding the best results. Three hidden layers and 1 output layer were used with each hidden layer containing 50 neurons and dropout rate of 0.2. The model was run through 30 epochs with a batch size of 32. The learning rate was 0.001. A 3D input array was fed into the model and utilized the ReLU activation function for the inner layers and SoftMax for the outer layer. The model was optimized using the 'softmax' optimizer and the categorical crossentropy loss function was employed.

Evaluation of the model metrics shows the final model accuracy achieved 76%. The weighted average precision, recall, and F1 achieved 70%, 76%, and 72%. Results are shown in Table 5.

Table 5. Model metrics for PBID multiclass classification model.

	Precision	Recall	f1-score	Support
Analysis	0.07	0.00	0.01	588
Backdoor	0.02	0.00	0.00	554
DoS	0.00	0.00	0.00	2871
Exploits	0.43	0.63	0.51	9115
Fuzzers	0.25	0.20	0.22	5294
Generic	0.79	0.81	0.80	18871
Normal	0.92	1.00	0.95	36757
Reconnaissance	0.00	0.00	0.00	2897
Shellcode	0.00	0.00	0.00	318
Worms	0.00	0.00	0.00	37
Accuracy			0.76	77302
Macro Avg	0.25	0.26	0.25	77302
Weighted Avg	0.70	0.76	0.72	77302

Figure 11 shows the corresponding confusion matrix. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

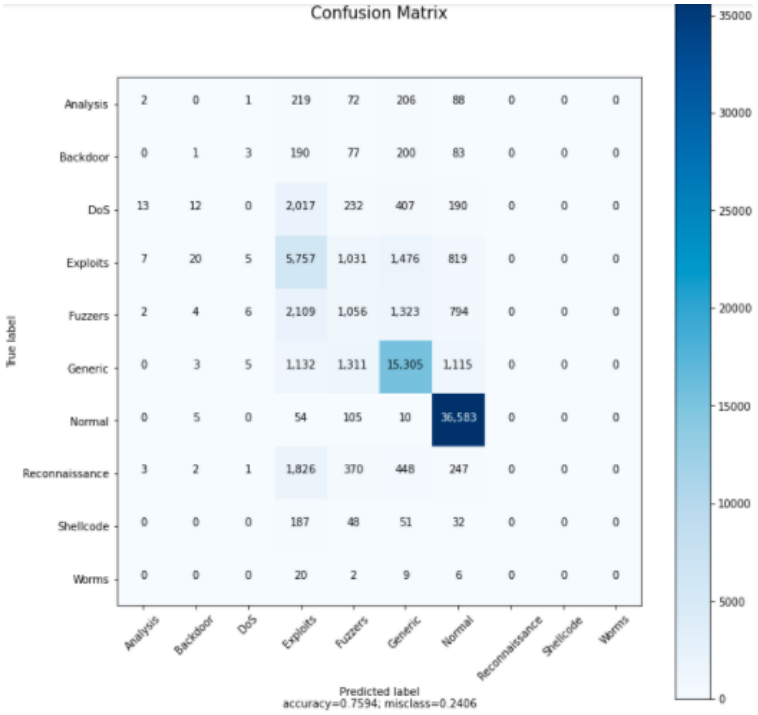


Fig.11. Confusion matrix for PBID multiclass classification model.

A DNN model was also run with similar parameters to compare to the performance of the BLSTM model. Results show that the accuracy of the DNN model was 81% compared to 76% for the BLSTM model.

6 Discussion

For this paper, BLSTM models were used to classify normal and abnormal network traffic using the UNSWNB-15 dataset. Flow-based (binary classification) and packet-based (multiclass classification) data were used in two separate FBID and PBID models to detect network intrusion.

The Keras Tensorflow framework was used to train the FBID and PBID models individually. The original training dataset consisting of 175,341 training records and test dataset consisting of 82,332 test records were combined and then randomly split into 70% training data and 30% testing data.

Feature reduction techniques were explored, but the models were more effective when 33 features were included. The BLSTM FBID model was trained using 60 steps and the packet-based model was trained using 30 steps. Three hidden layers and 1 output layer were used with each hidden layer containing 50 neurons and a dropout rate of 0.2. The model was run through 30 epochs with a batch size of 32. The learning rate was 0.001. A 3D input array was fed into the model and utilized the ReLU activation function for the inner layers and sigmoid and softmax activation functions for the outer layer. The models were optimized using the 'adam' optimizer for the FBID model and the 'softmax optimizer' for the PBID model. Binary crossentropy and categorical crossentropy loss functions were employed.

Evaluation of the model metrics shows the model accuracy achieved for the BLSTM FBID model was 96%. The weighted average precision, recall, and F1 achieved 96%, 96%, and 96%. For the BLSTM PBID model, the final model accuracy achieved 76%. The weighted average precision, recall, and F1 achieved 70%, 76%, and 72%.

The results suggest that BLSTM is more effective in predicting flow-based data, but DNN is more effective in predicting packet-based data.

7 Conclusion

The results of the analysis suggest that using BLSTM to predict network intrusion for flow-based data is a suitable method. However, DNN performs better with packet-based data. Future work will be to improve the BLSTM packet-based model so that it is comparable to or better than DNN. Once this is achieved, analyzing both flow-based and packet-based data in a hybrid fashion using BLSTM could provide an extra layer of reliable protection if built in a scenario where both models work in a cascading fashion through a self-activating algorithm. The FBID detects any unusual traffic flow and sends an alert to the PBID model to identify the type of attack packet. This would be an extra layer of defense against attackers.

8 Ethics

Computer networks are an invaluable asset within organizations that allow for seamless communications and critical information transfer and storage. However, these networks are always vulnerable to various types of evolving security intrusions by malicious attackers. Machine learning can be applied to mitigate the risk of these intrusions. It is the ethical responsibility of model administrators to "contribute to society and human well-being," as stated in the ACM Code of Ethics and Professional Conduct [30]. However, machine learning techniques cannot be trusted to be 100% accurate. The risk of an attack will always be present. It is the responsibility of the administrator to continually reassess methods to prevent attacks, adjust accordingly, and communicate that intrusion detection models are not perfect. Additionally, machine learning models have access to extensive data, some of which could be deemed confidential. It is the responsibility of the model administrator to evaluate data for sensitivities and ensure the confidentiality of individuals' information when necessary.

References

1. Security, R. (n.d.). Request the 2019 Q1 Data Breach QuickView Report. Retrieved July 05, 2020, from https://pages.riskbasedsecurity.com/2019-midyear-data-breach-quickview-report
2. Gartner Forecasts Worldwide Information Security Spending to Exceed $124 Billion in 2019. (n.d.). Retrieved July 05, 2020, from https://www.gartner.com/en/newsroom/press-releases/2018-08-15-gartner-forecasts-worldwide-information-security-spending-to-exceed-124-billion-in-2019
3. Raghudharan, R. (n.d.). Intrusion Detection Systems: Beyond the first line of defense. Retrieved June 15, 2020, from http://www.networkmagazineindia.com/200109/security1.htm
4. Cyber Attack - What Are Common Cyberthreats? (2020, March 30). Retrieved July 05, 2020, from https://www.cisco.com/c/en/us/products/security/common-cyberattacks.htmlAppendix: Springer-Author Discount
5. Hongyu Liu, & Bo Lang. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. Applied Sciences, 9(20). https://doi.org/10.3390/app9204396
6. An Overview of Flow-Based and Packet-Based Intrusion Detection Performance in High Speed Networks Retrieved July 05, 2020, from https://core.ac.uk/download/pdf/80743019.pdf
7. Nguyen, Huy, and Deokjai Choi. "Network Anomaly Detection: Flow-Based or Packet-Based Approach?" (2010): n. pag. Print.
8. Gers, F., Schraudolph, N., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research, 3, 115–143.
9. Long short-term memory S Hochreiter, J Schmidhuber - Neural computation, 1997 - MIT Press.
10. Gers, F et al. "Learning Precise Timing with LSTM Recurrent Networks." Journal of Machine Learning Research 3.1 (2003): 115143. Web.
11. Yang, Su. "Research on Network Behavior Anomaly Analysis Based on Bidirectional LSTM." 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE, 2019. 798–802. Web.
12. Graves, Alex, and Jürgen Schmidhuber. "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures." Neural Networks 18.5-6 (2005): 602–610. Web.
13. Amoroso E., "Intrusion Detection: An Introduction to Internet Surveillance," Correlation, and Response, New Jersey, 1999.
14. Aslam, Urooj & Batool, Ezzat & Ahsan, Syed & Sultan, Abdullah. (2017). "Hybrid Network Intrusion Detection System Using Machine Learning Classification and Rule-Based Learning System." International Journal of Grid and Distributed Computing. 10. 51-62. 10.14257/ijgdc.2017.10.2.05.
15. Alaidaros Hashem, Mahmuddin, Massudi & Al Mazari Ali. (2011). "An Overview of Flow-based and Packet-based Intrusion Detection Performance in High-speed Networks." School of Computing, University Utara Malaysia, Malaysia (2011)
16. Pramita Sree Muhuri, Prosenjit Chatterjee, Xiaohong Yuan, Kaushik Roy, & Albert Esterline. (2020). Using a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) to Classify Network Attacks. Information (Basel), 11(243)
17. Chuanlong Yin, Yuefei Zhu, Jinlong Fei, & Xinzheng He. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access, 5, 21954–21961.
18. Sanmeet Kaur, Maninder Singh. (2019) .Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks. Neural Computing & Applications, 32(12), 7859–7877.

19. Cheung, Hon & Roy, B. (2018). A Deep Learning Approach for Intrusion Detection in the Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), 1–6
20. Kwon, D., Kim, H., Kim, J., Suh, S., Kim, I., & Kim, K. (2019). A survey of deep learning-based network anomaly detection. Cluster Computing, 22(Supplement 1), 949–961.
21. Ralf C. Staudemeyer. (2015). Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal = Suid-Afrikaanse
22. Sheikhan, M., Jadidi, Z., & Farrokhi, A. (2012). Intrusion detection using reduced-size RNN based on feature grouping. Neural Computing and Applications, 21(6), 1185–1190.
23. Li, C., Wang, J., & Ye, X. (2018). Using a Recurrent Neural Network and Restricted Boltzmann Machines for Malicious Traffic Detection. NeuroQuantology, 16(5), 823–831.
24. KDDCup1999.Available-on: http://kdd.ics.uci.edu/databases/kddcup99/KDDCUP99.html, 2007.
25. NSLKDD. Available on: http://nsl.cs.unb.ca/NSLKDD/, 2009.
26. The UNSW-NB15 data set files: https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEE
27. Moustafa, Nour, Jill Slay, and Nour Moustafa. "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)." The Institute of Electrical and Electronics Engineers, Inc. (IEEE) Conference Proceedings. N.p., 2015. 1–6. Web.
28. Flow-Based and Packet-Based Processing - Technical Documentation - Support. (2020). Juniper Networks. https://www.juniper.net/documentation/en_US/junos/topics/reference/general/security-feature-flow-based-packet-based-processing-support.html#:%7E:text=A%20flow%20is%20a%20stream,and%20share%20the%20same%20characteristics.&text=Packet%2Dbased%20processing%20applies%20stateless,egress%20interface%2C%20or%20to%20both
29. F. Chollet et al. (2015). Keras. [Online]. Available: https://github.com/fchollet/keras.
30. The Code affirms an obligation of computing professionals to use their skills for the benefit of society. (2020). Acm.Org. https://www.acm.org/code-of-ethics

Appendix

Below is a list of variables included in the UNSW-NB15 dataset.

Table A. UNSW-NB15 variables.

Feature	Data Type	Definition
srcip	nominal	Source IP address
sport	integer	Source port number
dstip	nominal	Destination IP address
dsport	integer	Destination port number
proto	nominal	Transaction protocol
state	nominal	Indicates to the state and its dependent protocol
dur	Float	Record total duration
sbytes	Integer	Source to destination transaction bytes
dbytes	Integer	Destination to source transaction bytes
sttl	Integer	Source to destination time to live value
dttl	Integer	Destination to source time to live value
sloss	Integer	Source packets retransmitted or dropped
dloss	Integer	Destination packets retransmitted or dropped
service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
Sload	Float	Source bits per second
Dload	Float	Destination bits per second
Spkts	integer	Source to destination packet count
Dpkts	integer	Destination to source packet count
swin	integer	Source TCP window advertisement value
dwin	integer	Destination TCP window advertisement value
stcpb	integer	Source TCP base sequence number
dtcpb	integer	Destination TCP base sequence number
smeansz	integer	Mean of the low packet size transmitted by the src

dmeansz	integer	Mean of the low packet size transmitted by the dst
trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
res_bdy_len	integer	Actual uncompressed content size of the data transferred from the server's http service.
Sjit	Float	Source jitter (mSec)
Djit	Float	Destination jitter (mSec)
Stime	Timestamp	Record start time
Ltime	Timestamp	Record last time
Sintpkt	Float	Source interpacket arrival time (mSec)
Dintpkt	Float	Destination interpacket arrival time (mSec)
tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
is_sm_ips_ports	Binary	If source and destination IP addresses equal and port numbers equal then, this variable takes value 1 else 0
ct_state_ttl	Integer	No. for each state according to specific range of values for source/destination time to live.
ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
ct_ftp_cmd	integer	No of flows that has a command in ftp session.
ct_srv_src	integer	No. of connections that contain the same service and source address in 100 connections according to the last time.

ct_srv_dst	integer	No. of connections that contain the same service and destination address in 100 connections according to the last time.
ct_dst_ltm	integer	No. of connections of the same destination address in 100 connections according to the last time.
ct_src_ ltm	integer	No. of connections of the same source address in 100 connections according to the last time.
ct_src_dport_ltm	integer	No of connections of the same source address and the destination port in 100 connections according to the last time.
ct_dst_sport_ltm	integer	No of connections of the same destination address and the source port in 100 connections according to the last time.
ct_dst_src_ltm	integer	No of connections of the same source and the destination address in in 100 connections according to the last time.
attack_cat	nominal	The name of each attack category. In this data set, nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
Label	binary	0 for normal and 1 for attack records