electronics

MDPI

Article

MAN-EDoS: A Multihead Attention Network for the Detection of Economic Denial of Sustainability Attacks

Vinh Quoc Ta [1] and Minho Park [1,2,*]

[1] Department of Information Communication, Materials and Chemistry Convergence Technology, Soongsil University, Seoul 156-743, Korea; taquocvinh@soongsil.ac.kr
[2] School of Electronic Engineering, Soongsil University, Seoul 156-743, Korea
* Correspondence: mhp@ssu.ac.kr

Abstract: Cloud computing is one of the most modernized technology for the modern world. Along with the developments in the cloud infrastructure comes the risk of attacks that exploit the cloud services to exhaust the usage-based resources. A new type of general denial attack, called "economic denial of sustainability" (EDoS), exploits the pay-per-use service to scale-up resource usage normally and gradually over time, finally bankrupting a service provider. The stealthiness of EDoS has made it challenging to detect by most traditional mechanisms for the detection of denial-of-service attacks. Although some recent research has shown that multivariate time recurrent models, such as recurrent neural networks (RNN) and long short-term memory (LSTM), are effective for EDoS detection, they have some limitations, such as a long processing time and information loss. Therefore, an efficient EDoS detection scheme is proposed, which utilizes an attention technique. The proposed attention technique mimics cognitive attention, which enhances the critical features of the input data and fades out the rest. This reduces the feature selection processing time by calculating the query, key and value scores for the network packets. During the EDoS attack, the values of network features change over time. The proposed scheme inspects the changes of the attention scores between packets and between features, which can help the classification modules distinguish the attack flows from network flows. On another hand, our proposal scheme speeds up the processing time for the detection system in the cloud. This advantage benefits the detection process, but the risk of the EDoS is serious as long as the detection time is delayed. Comprehensive experiments showed that the proposed scheme can enhance the detection accuracy by 98%, and the computational speed is 60% faster compared to previous techniques on the available datasets, such as KDD, CICIDS, and a dataset that emerged from the testbed. Our proposed work is not only beneficial to the detection system in cloud computing, but can also be enlarged to be better with higher quality of training and technologies.

Keywords: network intrusion detection; cloud computing; economic denial of sustainability (EDoS); machine learning; deep learning; multihead attention network

check for updates

Citation: Ta, V.Q.; Park, M. MAN-EDoS: A Multihead Attention Network for the Detection of Economic Denial of Sustainability Attacks. *Electronics* **2021**, *10*, 2500. https://doi.org/10.3390/electronics10202500

Academic Editor: Mehdi Sookhak

Received: 4 September 2021
Accepted: 9 October 2021
Published: 14 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Problems Statements

In the past few years, cloud computing has been one of the fastest growing technologies of the IT industry. It provides a better business environment with elastic computing resource provisioning and powerful, high-availability clusters of virtual data centers. The advantages of the cloud computing are considered to be cost-savings, high-speed deployment, data back-up and restoration, on-demand self-service, allowing pay-per-use, etc. As the next generation architecture of the IT enterprise, cloud computing has unique features, such as the organization of dynamic resource and costs based on utilization [1,2]. Despite various benefits it might bring, cloud computing also possess vulnerabilities and is under security threats, like previous network technologies. The scalability feature of cloud computing may especially cause economic issues for the users, such as exorbitant costs [3,4].

The issues caused by attackers lead to many serious disadvantages in a variety of practical fields, such as healthcare monitoring systems [5], online marketing, managements and so on.

Economic denial of sustainability (EDoS) is a kind of denial-of-service (DoS) attack that focuses on the economic aspect [6]. The EDoS attack exploits the cloud computing feature of pay-per-use and causes an unnecessary scale-up of the service, resulting in financial waste for the cloud resource. Although EDoS attacks share similarities with conventional DoS threats, specifically those based on flooding [7] they often contain less traffic volume. Thus, EDoS is a form of low-rate DoS attack and is difficult to detect by traffic volume-based mechanisms. The EDoS attack is generated over thousands of attack packets, or even more attack packets per second sent to the victims. Consequently, a one-second delay in the detection would cause serious exhaustion to the victims. Thus, rapid detection in the system is very crucial. Furthermore, DoS attacks aim to bankrupt the service provider. Therefore, they can remain stealthy for a long time while wasting the cloud resource and deceiving most existing DoS defense threshold-based mechanisms.

To address the issue of EDoS detection, various methods [8] have been proposed. Among them, machine learning (ML) techniques, such as support vector machine (SVM) and self-organizing map (SOM), have been shown to detect EDoS attacks well [9,10]. However, because most traditional ML methods rely heavily on feature engineering and selection, they have limitations in effectively processing the massive EDoS data that arise in network application environments. To enhance the accuracy and robustness of the detection method and overcome the limitations of ML, deep learning (DL), a type of ML technique, is applied automatically to better extract features by using neural layers. As classification algorithms in DL, various neural network models mimic human nerves and use a large number of non-linear processing units to address complex problems. Because of the stealthiness of EDoS, many models and methods that apply flow-based detection methods require a time-series model, which can recall the latest input and forecast the output of the sequence data. The recurrent neural network (RNN) is a model for the sequence input that can successfully detect DoS attacks [11]. Another RNN scheme [12], called the long short-term memory (LSTM) model which can solve the vanishing gradient problem of RNNs by an internal mechanism called “memory gates” that can regulate the flow of the input sequence. Compared with previous models [11], this recurrent model faces a memory limitation and time delay in the training phase and detecting phase. Because of gradually computing the inputs over time of LSTM, stacking the memory cells caused the model to lose the information of the past input and easily vanish when the back propagation gradient is calculated. In addition, the time delay in model training is significant, especially when the input is long. These shortcomings of the recurrent models make them inefficient for processing the sequence data as EDoS data. In this research, to overcome the limitation in the time delay of recurrent models and increase the accuracy of EDoS detection, a new designed scheme is proposed.

1.2. Contribution

To counter the problems in recurrent models for the EDoS detection problem, a scheme is proposed that makes the training and predicting phase faster than the recurrent LSTM by parallel-calculating and keeps the data for forecasting the EDoS attack. In summary, the major contributions are as follows.

- An effective EDoS detection scheme is proposed, which applies the attention technique. Attention in DL can be interpreted as a vector of attraction score weights. To predict or comprehend one element, such as a pixel in an image or a word in a sentence, the attention vector is used to determine how strongly the unit is correlated with other units in the sequence data, and the sum of their values weighted by the attention vector is taken as the approximation of the target.
- In the present context, it is the relative scores of one feature to another features in the network packet and one packet to the others in the flow network. These scores help

to remember the history features of the input sequence. Furthermore, unlike most other statistical techniques that have to compute with the feature selection threshold, this scheme can compute the score of one feature that it observes and apply it to other features for calculating the relative of one feature to another.
- The attention score is applied in the model that is similar to the auto encoder–decoder network that has been introduced in the transformer model [13,14]. Instead of using both modules as the transformer, the proposed modified model includes only one encoder module block that can compute, in parallel, the inputs for accelerating the training time but still gain the best accuracy as a sequence model as in LSTM schemes.
- Moreover, the constructed attention model can address unsupervised problems by predicting the attack traffic using the attention score for classifying the zero-day attack output.
- In addition, the MAN mechanism speeds-up the time processing. The improvement in testing time will improve the detection time for the system, and the improvement in training time will help update the model by fine-tuning the system update immediately to get used to the change in the attacks.
- Common types of EDoS attacks in the cloud environment were investigated by adjusting the attack rate to be lower than that in popular flooding attacks in the cloud and network environments.

The rest of the article is organized as follows. In Section 2, some related works are reviewed. In Section 3, background knowledge is discussed. The proposed scheme and experimental setup are explained in Section 4. The performance results and evaluation are given in Section 5. The conclusion and future research directions are given in the final section.

2. Related Works

There have been several works on EDoS detection techniques. Al-Haidari et al. [15] introduced EDoS-shield, which contains virtual firewalls (VFs) and verified cloud nodes (V-nodes). The VFs work by filtering mechanisms that allow the packets originating from the whitelist IP addresses to pass and dropping other packets originating from blacklist IP addresses. These lists are frequently updated by the V-nodes according to the results of the detection EDoS attacks module. Although EDoS-shield worked as the detection module, it is easily bypassed by the spoofed IP addresses of attackers if the spoofed IP is not in, or not updated to, the lists. Self-verifying proof of work (sPoW) [16] is an on-demand cloud-based and application layer mitigation scheme. The main function of this method is to judge the attack traffic before it starts pledging the resources. It transforms the network-level traffic to distinguish the traffic that matches the attack pattern. However, one of this framework's requirements is the high computation power needed to solve crypto-puzzles for clients, and the attackers can start a puzzle accumulation attack based on its vulnerability. Moreover, the complexity of the puzzle can increase the false-positive rates because some users are excluded from the network traffic or the cloud. EDoS armor is a multi-layered defense system [17]. It is composed of an admission control phase and a congestion control phase. The admission control restricts the number of end-users, while the congestion control sets the priorities for the client based on past browsing behavior that is benign or anomalous using a decision tree algorithm. A limitation of this technique is the adaptability of the model because the site could be complicated for potential new users, who might then lose interest. Game theory is the study of mathematical models of communicating strategically among rational decision-makers. Chowdhury et al. [18] applied game theory to construct a scenario among attackers and defenders. They obtained the optimal threshold value by calculating the Nash equilibrium and incorporated honeypot to minimize false rates. However, the game theory failed to operate reliably in the case of massive matrices of payment. The use of a honeypot is unnecessary if it has to extract new signatures, so it is not beneficial for detection.

In recent years, ML techniques have attracted much attention based on their efficiency and accuracy in various fields, including computer vision, healthcare, image processing, and intrusion detection. Ghanem et al. [9] introduced the SVM technique as the detection module to reduce the number of false alarms in distributed denial-of-service (DDoS) detection applications. In [10] Trung et al. proposed an efficient solution to counter DDoS attacks by using a hybrid ML model based on SVM and SOM algorithms to improve the traffic classification. The SVM acts as a high-speed classification based on hyperplanes in a high dimensional space and SOM increases the accuracy of vague data points that are considered as suspicious points. Although ML worked for DDoS detection, ML is limited because of the vast quantity of data with a vulnerable threshold, especially unfeasible for EDoS attacks, because the vast quantity of data results in false-positive errors. A branch of ML-DL has become popular and has been applied for DDoS attack detection. Many studies have shown that DL can achieve much better accuracy with increasing datasets than ML. Shaaban et al. [19] applied a convolution neural network (CNN), which technically mimics the human brain. The difference of CNN is that it learns directly from image-like input samples, not handcrafted features. In addition to the packet-based method to detect DDoS and EDoS attacks, the flow-based method can also be used. Yin et al. [11] proposed a DL approach for intrusion detection using RNNs. With continuous traffic flow, the RNN is applied as the classification technique for sequential input. By calculating input sequentially, the recurrent model can remember the input of the previous character and predict the next input so as to make a decision about the entire flow sequence. If the sequence is extremely long, the RNN model is limited in memory because the features are stacked in memory cells. LSTM is another recurrent model that can address the RNN memory problem. In previous studies [12,20,21], LSTM showed major improvements over what RNNs could accomplish. LSTM is designed to avoid long dependency problems and can remember long historical information and gain high accuracy in EDoS detection with a sequence flow-based method. Nevertheless, with the activation functions in recurrent gates, the computations slow down the training time and prediction time of the scheme and affect the multivariate-real-time with a long input sequence. According to one report [22], the concept of bi-directional LSTM (BiLSTM) is enhanced from a bidirectional RNN that processes sequence inputs in forward as well as backward directions by utilizing two different hidden layers. BiLSTM joins both the hidden layers to the same output layer. BiLSTM runs the inputs in two ways: one from the past to the future and one from the future to the past. What distinguishes this approach from a uni-directional one is that the LSTM that runs backward preserve information from the future and uses the two combined hidden states, leading to the ability at any point in time to preserve information from both the past and future. However, by calculating the inputs in two ways, backward and forward, the BiLSTM is even slower than LSTM in training time and predicting time.

Because of the issues of recurrent models in detecting the intrusion network, specifically in EDoS attack detection, a multi-head attention mechanism working with an encoder model was proposed as the Transformer model [13,14]. To the best of our knowledge, most of the research works have been focused on improving the accuracy, while the time processing of the detection system is significant so that it affects the EDoS detection. Hence, our proposal can achieve the best results for sequence issues in EDoS attack detection, and also increase the quality of time processing.

3. Background Knowledge

3.1. EDoS Attack on Network

In general, a DDoS attack is a targeted attack by multiple compromised computers called “botnets” or “zombies”, focusing on a single system network [23]. Their purposes are to exhaust the objective system, thus making it unavailable for registration. Similarly to DDoS attacks, in EDoS attacks, the network systems are attacked by many botnets that are spoofed by attackers. However, EDoS attacks are stealthier and gradually exhaust the system by pushing illegitimate traffic over a longer period of time. The motivation of EDoS attackers is to increase the costs that are incurred by the users through the system or the cloud. The users are then forced to rent extra computing resources by manipulating the auto-scaling engine in the cloud. As a result, the users have to pay more money.

As in Figure 1, EDoS attacks have a lower region of attack rate and intensity threshold value than DDoS attacks, thus, an EDoS attack can easily be ignored and pass the DDoS defense mechanisms. EDoS is described as a specific family of attacks and one kind of low-rate attack against the cloud computing platforms detection system, where the attacker aims at increasing the economic costs derived from both maintenance and provision of the services offered, hence making the support less applicable, even achieving denial. In general terms, EDoS attacks have similarities to regular DDoS threats, especially those based on flooding. However, EDoS raise a significantly different problem. The characteristics and impact of EDoS are illustrated in Figure 2.

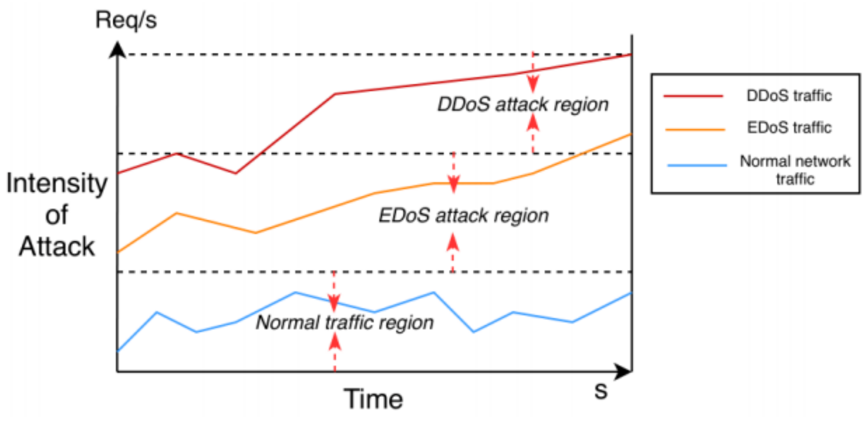

Figure 1. DDoS and EDoS attacks regions.

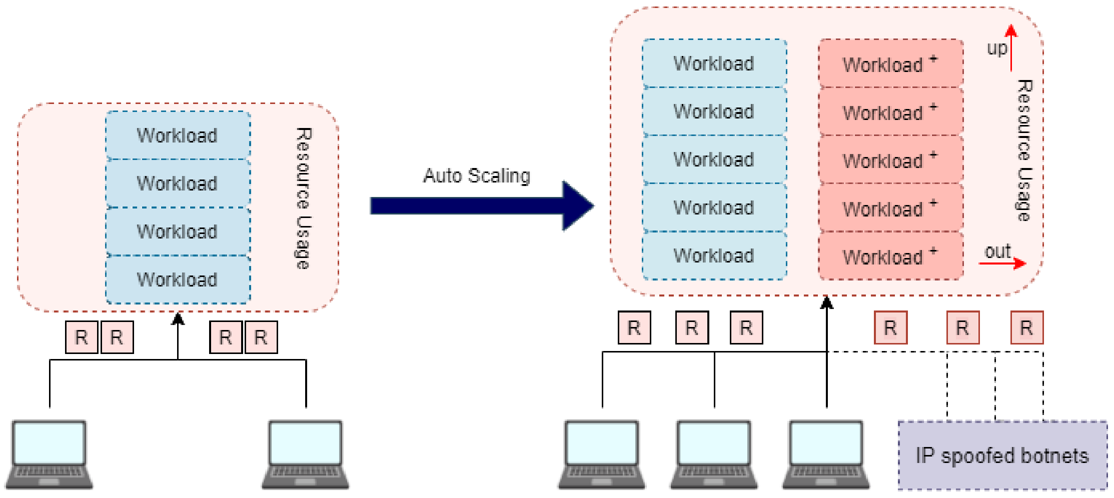

Figure 2. Auto-scaling triggered by EDoS attacks.

An effect of EDoS attacks is that it becomes necessary to scale-up and scale-out the deployed environment resource by adding additional computational resources when the request packets of the attacker are consequently generated, flooding the resource [24]. Moreover, Figure 3 illustrates the CPU utilization of EDoS attacks in multi-variate time series. Some characteristics, such as CPU utilization, are changed by time, this multi-time variate feature is significant in the flow-based method rule instead of the packet-based rule. As in Figure 3, some parameters change extremely during the attack time. Following this alternation, we use these two extremely high changes in values of CPU usage and memory usage parameters to collect and process data in the Data Preparation section. However, the CPU and memory usage features are the sequential features that are multivariate time in series. Therefore, a mechanism is proposed that is not only based on sequence multi-variate time data, as a flow-based method to detect EDoS attacks, but also overcomes the shortcomings of previous approaches. In the following subsections, we discuss the varieties of EDoS attacks, as mentioned in previous research [23].

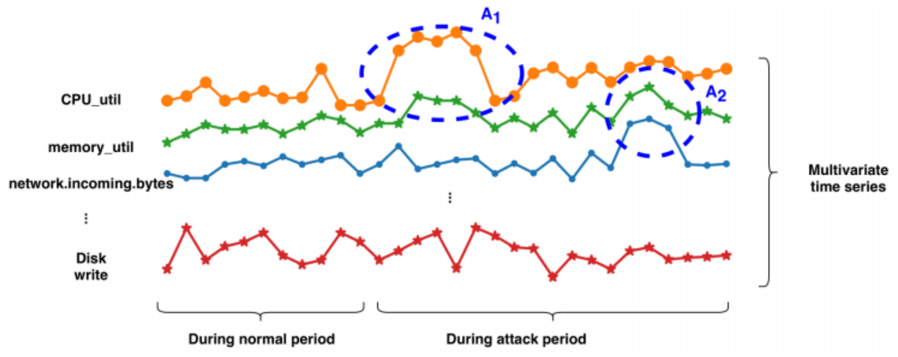

Figure 3. Suspected regions occurring during normal and EDoS attacks periods.

3.1.1. TCP SYN Flooding Attack

Transmission Control Protocol (TCP) is a connection-adapted protocol that happens on the transport layer of the model stack. The connection is settled by the three-way handshaking method by sending the SYN and ACK packets and receiving response ones respectively. The clients send an SYN packet to the server. Upon receiving the SYN packet, the server acknowledges the SYN-ACK packet and allocates the TCP stack for the new connection request, and it comes into the listening state. In this type of attack, the attackers send repeated SYN packets to ports on the headed server, often using fake IP addresses. The server unfamiliar with the attack receives multiple, apparently appropriate, requests to establish communication. It responds to each attempt with an SYN-ACK packet from each open port. The attacked client either does not send the expected ACK or the IP address is spoofed, and the SYN-ACK is never received in the first place. Either way, the server under attack will wait for acknowledgment of its SYN-ACK packet for a bit of time. The server cannot close down the connection by sending an RST packet and the connection stays open. Before time is out, another SYN packet is reached. This leads to an increase in the large number of connections being half-open connections. Eventually, due to the connection overflow tables, services of legitimate clients are denied and the server may even malfunction or crash [25].

3.1.2. UDP Flooding attack

The User Datagram Protocol (UDP) is produced on top of the Internet Protocol (IP) to broadcast datagrams over the network interface. UDP does not require the source and destination to build the three-way handshaking like TCP. Moreover, it is not crucial for an end-to-end connection [26]. The minor of the authentication mechanism and end-to-end connections make UDP vulnerable to attacks. The application is reached directly with the spoofed IP packets that are generated by the attackers. In the variety of ports on a single server, the flood of UDP contains the huge volumes of packets. The server reacts to all the requests with response messages, overwhelming its resources. In addition, in the EDoS UDP flood attack, the attackers send the packets frequently, but with a low-rate request to slowly congest the network traffic.

3.1.3. ICMP Flooding attack

The Internet Control Message Protocol (ICMP) is a network layer protocol used by network devices to analyze network communication problems. ICMP is used to determine whether data reach their destinations in a timely manner. Commonly, the ICMP is used on network devices, such as routers. ICMP is crucial for error recording and verification, but it can also be used in DoS attacks, especially EDoS attacks. The ICMP flood attack can be broken down into two repeating steps.

- The attacker sends many ICMP request packets to the aimed server using multiple botnets.
- The victim server then sends an ICMP echo-response packet back respectively to the request one to the botnet's IP address as an answer. The effect on the network traffic is damaging when the number of requests made to the targeted server increase.

3.2. Attention Technique

The attention technique in the self-attention layer helps the model look at other input sentences as it encodes a specific feature input. As discussed in [13], self-attention is a mechanism that relates different points of a single sequence to compute a representation of the sequence. The first step in calculating self-attention is to create three vectors from each of the input vectors of the encoder. For each input $\mathbf{X}_i$ in window slot $t \leq i \leq d$, three matrices, Query $\mathbf{Q}$, Key $\mathbf{K}$ and Value $\mathbf{V}$, are created by multiplying the embedding three matrices that are in training process.

$$\mathbf{Z} = \sigma(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}}). \tag{1}$$

The Query, Key, and Value are useful for calculating and understanding attention. Secondly, self-attention is calculated as the $\mathbf{Z}$ scores. The scores determine the concentration to place on other parts of the inputs sequence. The scores are calculated by computing the dot-product of the $\mathbf{Q}$ matrix and the $\mathbf{K}$ transpose matrix. Then the results are divided by the square root of the dimension of the key matrix $\sqrt{d_k}$. This leads to more stable gradients. Then, the result is passed through a softmax operation σ. Softmax normalizes the scores that are between 0 and 1 so that they can be interpreted as probability and sum up to 1.

$$\sigma(z) = \frac{e^z}{\sum_{j=1}^{d} e^z}. \tag{2}$$

The softmax score determines how much each input packet is expressed at this position in one flow traffic window. These scores also are significant to weights on the values. The dot-product is more rapid and more space-efficient in practice because it can be achieved using highly optimized matrix multiplication code. The dot-product advances large in weight, pushing the softmax function into regions where it has extremely small grades, and $\sqrt{d_k}$ outperforms dot-product attention and scales the dot-product, preventing the vanishing in gradient.

$$Attention = A(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{Z} \times \mathbf{V}. \tag{3}$$

The third step is to multiply each value matrix $\mathbf{V}$ by the softmax scores. The intent is to keep the values of the packets and the features in the flow that one wants to focus on intact, and track the gradual change of the packets in the flow that affects much of the attack. Finally, the weighted value matrices are summed up. This produces the output of the self-attention layer at the position of these packets in the window flow.

According to [13] to improve the performance of the attention, the mechanism "Multi-Head Attention" has been introduced. This gives the attention layer multiple representation subspaces. The multihead of attention are multiple sets of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ weight matrices. Each of these sets is randomly initialized. After training, each set is used to project the input embedding into a different representation subspace.

$$MultiHead = Concatenate(A(\mathbf{Q}_j, \mathbf{K}_j, \mathbf{V}_j)). \tag{4}$$

Multihead attention is a method to concatenate the attention j-th results where j is the number of heads in the attention layers. Then, the concatenated result is the matrix $MultiHead \in \mathbb{R}^{T \times j}$.

Figure 4 is an example of the computation of the multi-head attention scores of three embedded inputs. To earn these representations, every input is multiplied with a set of weights for keys, a set of weights for queries, and a set of weights for values. In a neural network setting, these weights are usually small numbers, initialized randomly using an appropriate random distribution such as Gaussian distribution. This initiation is done once before training. After obtaining the key **K**, query **Q** and value **V** representations for every input with the same dimension size, to obtain the attention score, the dot product starts between the input's query with all keys of other inputs, including itself. The softmax scores of each dot product result are then multiplied by its corresponding value. The outputs for each input are then added up together. The calculation is repeated for every input to gain the result matrix attention scores.

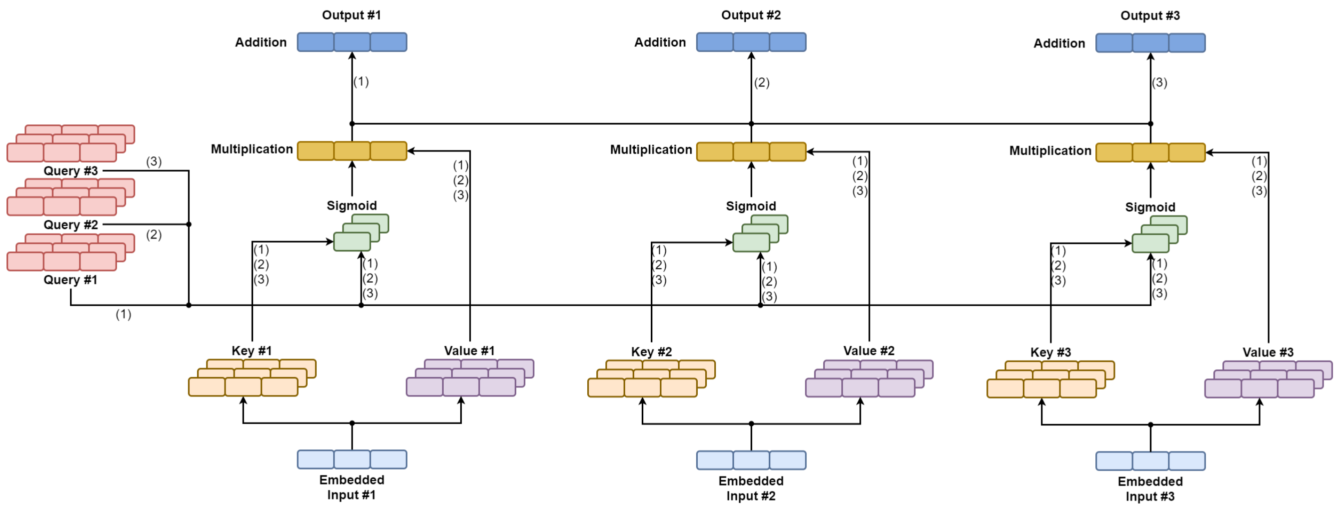

Figure 4. Attention score example of three embedded inputs.

4. Proposed Scheme and Experiment

4.1. Testbed Scheme for Network Communication

To construct the network traffic testbed, the virtual machine (VM) and Open vSwitch tools were used, and port mirroring was set up to capture the packets from the attacker to the VMs of the victim [27,28] as shown in Figure 5. The VMs attackers deploy different flooding attacks by running an attack script. The open-source software switch, called Open vSwitch, provides network connections between VMs inside the VirtualBox virtualization platform. In order to capture the packets from attackers to the victims, a third party was set up to run the Open vSwitch and port mirroring bridge.

The packets captured from the interface by Open vSwitch and port mirroring are translated to Data preprocessing modules before being fed into the model for training. The model decides whether the input flow window is an EDoS flow attack or benign flow. The defense system works as Virtual Firewall (VF) in [29] that contains the blacklist and whitelist, while the Virtual Firewall can be implemented in the cloud as a VM that has filtering and routing capability. The whitelist is used to track the authenticated source IP addresses that are in benign flow as decided by the trained model, and the blacklist is used to hold those unauthenticated source IP addresses that indicate EDoS flow attacks, which are excluded from the service and the incoming packet flows are dropped. These two lists are updated periodically by the time running the interface network traffic. After training the model, the predicting model is used in the mitigation system with the defense system. In the experiment in the next section, it is shown that this novel solution to EDoS has an influencing effect. Algorithm 1 describes the workflow of the testbed in Figure 5 of the EDoS detection and mitigation system.

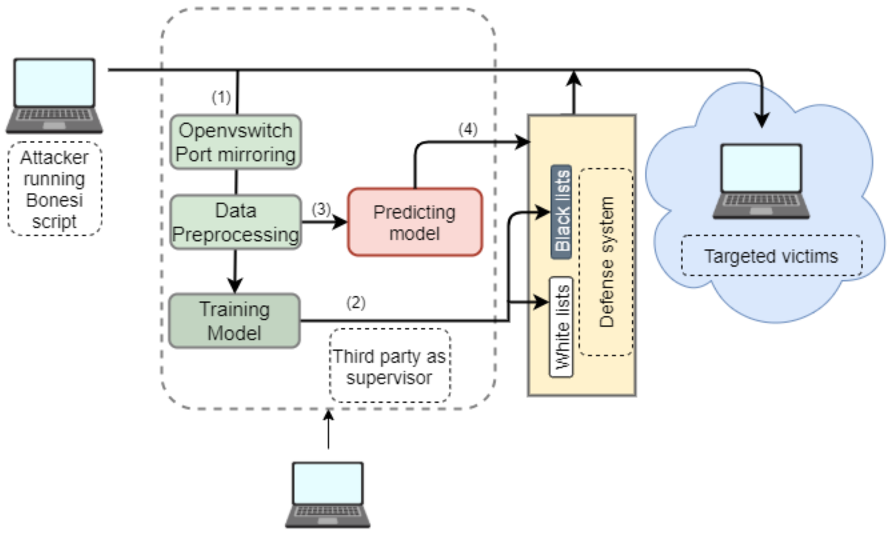

Figure 5. Testbed set up for capturing network packets.

Algorithm 1 Workflow of testbed for EDoS detection and mitigation

```
$\overline{Capture} \longleftarrow$ running Open vSwitch and port mirroring script for capturing packets from the interface.
$\overline{Pre-Process} \longleftarrow$ Processing data.
$\overline{Model} \longleftarrow$ Multihead Attention Model (MAN).
$\overline{Defense} \longleftarrow$ The defense system that contains black lists and white lists.
loop
    x $\longleftarrow \overline{Capture}$
    X $\longleftarrow \overline{PreProcess}$
    Training: output $\longleftarrow \overline{Model} \longleftarrow$ X
    Updating:
      if output is normal:
        $\overline{Defense} : White-lists \longleftarrow$ output
      else output is EDoS:
        $\overline{Defense} : Black-lists \longleftarrow$ output
      end if
    Predicting:
      output $\longleftarrow \overline{Model} \longleftarrow$ X
      if Output is EDoS
        $\overline{Defense} \longrightarrow$ Drop
      else output is benign
        $\overline{Defense} \longrightarrow$ Pass
      end if
end loop
```

4.2. EDoS Attack Performance

In this section, first a scheme for performing an EDoS attack based on DDoS attack application script—BoNeSi [30] is constructed. As mentioned, an EDoS attack has a lower rate in a longer period of time.

Table 1 shows the different kinds of EDoS attacks, classified by the BoNeSi tool. In the TCP SYN flooding attack, the attackers generate many botnets that carry the spoofed IP addresses (100 botnets in the experiment) to send SYN packets to the victims and exhaust the network traffic. The UDP and ICMP flooding attacks differ from the TCP one, by sending a large number of attack packets, which eventually causes the system traffic network to be unreachable. The EDoS attack is different from a DDoS one in that it is stealthier, and the intensity is under the threshold value of the attack traffic rate.

To generate EDoS attack, the attack rate in the BoNeSi script was adjusted to 2000 and 3000 request packets per second, which is different from the default DDoS attack of more than 50,000 requests per second.

Table 1. EDoS attack by BoNeSi.

EDoS Attack	Rate (pkts/s)	Botnets	Times (s)
TCP SYN flooding	2000	100	3600
ICMP flooding	3000	-	3600
UDP flooding	3000	-	3600

4.3. Preprocessing and Model Work Flow

In this section, the workflow of the scheme is discussed, including: data capture, data preprocessing and model architecture as shown in Figure 6.

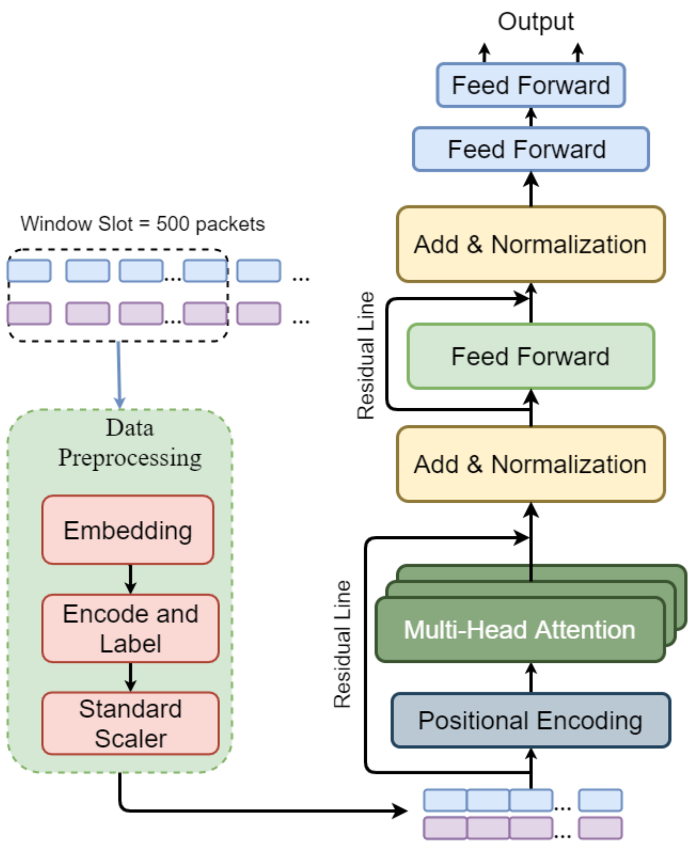

Figure 6. Model flow architecture.

4.3.1. Data Preparation

- Data capture: This module runs in basic network traffic. After collecting generated packets from BoNeSi attackers, the packets inflow is captured by the Wireshark tool which can split flow by the number of packets per flow and by the time per flow. Referring to a previous study [21], the sequence length is 250 packets per flow. To overcome the loss of information and memory vanishing of LSTM, a longer sequence length of the window slot, that is 500 packets per flow sequence, is proposed. For a sequence flow that does not have enough packets of protocol, the proposed methods generate fake packets in the flow, which contain zero values to fill up the sequence window slot.

- Data preprocessing: The embedding module transforms the categorical features into numerical form by using the one-hot encoding technique. Encoding and labeling are used to label flows generated from the window slot. The Standard Scaler normalizes the data that are significant to the model for calculating and accelerating the time in training and predicting the phase. The formula can be expressed as:

$$z = \frac{x - min}{max - min}, \tag{5}$$

where x is the value that is standardized, min and max are the minimum and maximum values of every features in dataset.
- Data description: For the training phase and testing phase, two datasets are used from the training scheme and the UNSW-NB15 dataset [31,32]. The UNSW-NB15 dataset contains pcap files that can extract the packets from the flows by the timestamp. This dataset can represent actual situations in the real network and overcome the shortcomings of KDDCUP'99 [33]. Moreover, to evaluate the model prediction results, the NSL-KDD and CICIDS dataset introduced by the Canadian Institute are used with the same features and behaviors [33,34]. They have gradually become one of the benchmark datasets in the field of network security. Table 2 shows the features of packets that are generated from the system and from the testing UNSW-NB15 dataset.

Table 2. Key features.

Features	Description
ip_src	Source IP address
ip_dst	Destination IP address
proto	Network protocol type
port_src	Source port
port_dst	Destination port
dur	Flow duration
sttl	Source time to live
dttl	Destination time to live
time_s	Time stamp packet captured
length	Packet length bytes
spkts	Source to destination packets count
dpkts	Destination to source packets count
sload	Source bits per second
dload	Destination bits per second
stcpb	Source TCP base
cpu_util	CPU utilization
memory_usage	Memory usage
label	Label attack

4.3.2. Model Architecture

- POSITIONAL ENCODING: To make use the order of the sequence input, instead of recurrence and no convolution, the positional encoding adds up the encoding of the order of the sequence in the window slot and the input sequence:

$$p^t_{(pos,2i)\in d} = sin(pos/500^{2i/dim_{model}}) \tag{6}$$

$$p^t_{(pos,2i+1)\in d} = sin(pos/500^{2i/dim_{model}}) \tag{7}$$

$$P^t_d \in [p^t_0, p^t_1, p^t_2, ..., p^t_d] \tag{8}$$

$$Input = X^t_d + P^t_d \tag{9}$$

The parameter pos is the position and i is the dimension. Each dimension of the positional encoding harmonizes to a sinusoidal. The wavelengths form a geometrical advancement from 2π to $500 \times 2\pi$. Connecting these values to the embeddings provides full meanings of distances between the embedded vectors once they are projected into the **Q**, **K** and **V** vectors and during dot-product attention.

- Multihead Attention enables the model to attend to information from different positions of the sequence inputs jointly. The Residual Line is introduced from ResNet [35] to prevent the loss of information through network layers and improve the propagation of useful gradient information.
- Add and Normalization: add up the input information to the output from the attention layers. The normalization layer normalizes the activation of the previous layer for each given example in a batch independently.

$$\mathbf{Z} = Normal(MultiHead + \mathbf{X}_d), \tag{10}$$

where $\mathbf{X}_d$ is the input sequence in the window slot that uses the residual to add to the output of the multihead layers.
- Feed-Forward Network (FFN) is used to process the output of the previous layers that compute the feature maps as Fully Connected Feed-Forward networks and enforce a Sigmoid activation function:

$$Output = \sigma(\mathbf{Z}). \tag{11}$$

The calculating and training flow are shown in Algorithm 2 with two-phase data preprocessing and training phases. The data preparation phase cleans the dataset when extracting the packets network x_d^t from the interface in the size of window slot d by the time t. The embedding layer makes it possible to convert categorical data into a fixed-length vector of a defined size. The consequent vector is a dense one with real values instead of just 0 and 1. The fixed length of categorical data represents data in a better way with reduced dimensions. The one-hot-encoding embeds works just as a look-up table does. The categories are the keys in this table, while the dense word vectors are the values. The standard input that re-scales data from a range of 0 to 1 in order to not cause the gradient propagation to vanish. The experiment was performed on a local CPU (Intel Core i7 8700 3.20 GHz) and GPU NVIDIA GeForce GTX 1060 3 GB (32 GB memory). To train the Multihead Attention model, the model parameters were constructed as in Table 3.

Table 3. Model parameters.

Number of attention heads	8
Sequence length	500
Embed and Feed forward dimension	64
Dropout	0.1
Batch size	512
Epochs	500
Optimizer	SGD

Algorithm 2 Multihead attention network for EDoS attack scheme

```
$\{x_0^t, x_1^t, x_2^t, ..., x_d^t\} \longrightarrow$ A set of packets that captured from interface.
$\overline{Embedd} \longrightarrow$ Embedding layers for encoding the categorical data.
$\overline{Standard} \longrightarrow$ Standard Scaler for the huge value data.
while True:
    for x in range from t to t+d do
        $\overline{Embedd}, \overline{Standard} \longleftarrow x$
        $X_d^t = [x_0^t, x_1^t, x_2^t, ..., x_d^t]$
    end for
end while
Output: $\mathbf{X} = [X_d^t, X_d^{t+1}, X_d^{t+3}, ..., X_d^{t+T}]$
Attention model training:
$\mathbf{P}_d^t \longrightarrow$ Positional encoding for the order of the window slot.
$W_j^{\mathbf{Q}}, W_j^{\mathbf{K}}, W_j^{\mathbf{V}}, \mathbf{Q}_j, \mathbf{K}_j, \mathbf{V}_j \longrightarrow$ Weight matrices that are trained and calculated Query, Key and Value matrices.
$\overline{ADDNormal} \longrightarrow$ Add up layers and Normalization layers.
$\overline{MultiHead} \longrightarrow$ MultiHead Attention layer calculation.
$\overline{FFN} \longrightarrow$ Feed-Forward Network layers for linear calculation.
loop forward and backward propagation
    for $X_d^t$ in X do:
        Input = $\mathbf{P}_d^t + X_d^t$
        $Q_j, K_j, V_j = (W_j^{\mathbf{Q}}, W_j^{\mathbf{K}}, W_j^{\mathbf{V}}) \times$ Input
        $Z_1 \longleftarrow \overline{MultiHead} \longleftarrow \mathbf{Q}_j, \mathbf{K}_j, \mathbf{V}_j$
        $Z_2 \longleftarrow \overline{FFN} \longleftarrow \overline{ADDNormal} \longleftarrow \mathbf{Z}_1, X_j^t$
        Output $\longleftarrow \overline{FFN} \longleftarrow \overline{ADDNormal} \longleftarrow \mathbf{Z}_2$
    end for
end loop
```

5. Results and Evaluation

5.1. Results

During the training phase, the used training dataset, generated from the testbed and testing validation dataset, was UNSW-NB15.

Figures 7 and 8 illustrate the training and testing results of the multihead attention model in detecting the EDoS attacks. To evaluate the model results, we use metric accuracy and sparse category cross-entropy loss were used.

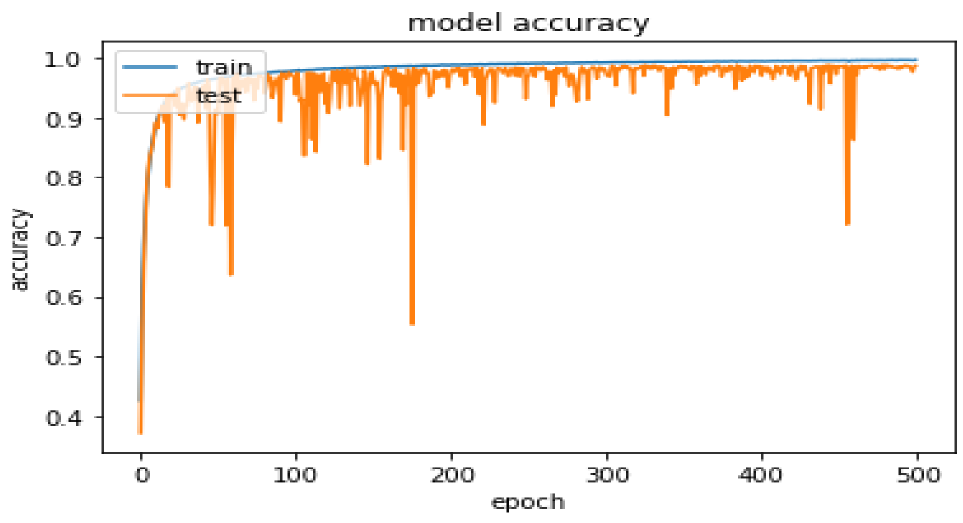

Figure 7. Model accuracy in training and testing phases.

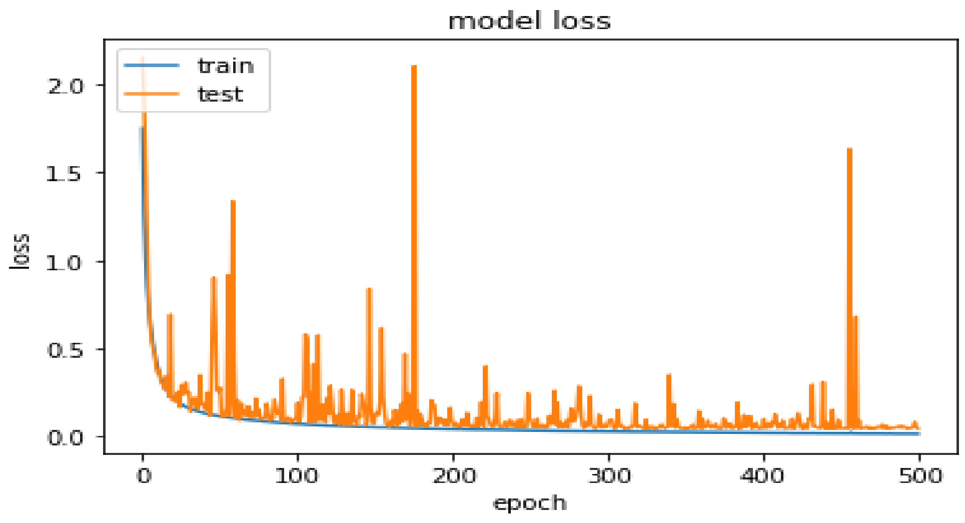

Figure 8. Model loss in training and testing phases.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Here, TP is the true positives, TN is true negatives, FP is false positives and FN is false negatives of trained data. The sparse categorical cross-entropy loss function is a measure from the field of information theory, building upon entropy and commonly calculating the characteristic between two probability distributions. The difference between sparse categorical cross-entropy and categorical cross-entropy is the format of true labels. In multi-class classification issues, the labels are commonly absolute for each datum. Then, one can represent true labels using one-hot embedding.

$$J(\omega) = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)], \tag{13}$$

where ω refers to the model parameters, y_i is the true label, $\hat{y}_i$ is the predicted label. During the training phase, the accuracy and loss have the intensity to converge in about hundreds of epochs through the test accuracy and test loss which fluctuate in time owning to the difference between the training dataset from the testbed and available validation dataset. However, the validation accuracy and test gradually increase and decrease, respectively, and gain more than 98.6% in accuracy and 0.012 in loss while the validation accuracy is nearly 98% and validation loss is 0.015. These results can be compared with the results of some related works. In a study [36], the best result was 97.3% with the neuron network attention technique. Likewise, our results are compared to the hierarchical attention technique in RNN in the work [37], which gained 98.76% on the UNSW-NB15 dataset. The comparisons reveal that the present method achieved comparable results to those of recent researches. The proposed scheme not only had better results but also increased the results of such metrics as time in training and predicting precision scores, and f1 scores.

5.2. Evaluation with Recurrent Models

Table 4 illustrates that the proposed scheme can improve the time in training and prediction compared with the recurrent models such as LSTM and RNN. According to the theory of recurrent models, the input that fed one by one with the order of input into the memory cells of the model to remember the important information of previous input and predict the output for the sequence. This causes a delay in training time and predicting time. The attention model not only feeds the inputs and computes in parallel, just as a neural network, but can also retain the information in the sequence by the attention techniques to predict the output.

Table 4. Model results and evaluation.

Metrics	MAN	BiLSTM	LSTM	RNN
Training time (s/epoch)	11	50	32	26
Predicting time (s)	2.40	3.12	2.77	2.56
Accuracy	0.98	0.99	0.98	0.95
Precision	0.989	1	0.981	0.95
Recall	0.983	0.967	0.98	0.958
F1_score	0.986	0.99	0.978	0.965

The results in the table show that the proposed attention model was much faster in training and testing times than the bidirectional LSTM, LSTM [21] and RNN [37] models. For each iteration epoch in the training time, the attention model was twice as fast as the RNN model and three times faster than the LSTM model, and its accuracy was as high as that of these other models. Moreover, the performance for the accuracy and f1 score metrics was superior, demonstrating that the proposed approach is better for EDoS attack detection. The performance metrics used were: accuracy, precision (the rate of the time slots anticipated as the attack that is truly an attack), recall (the rate at which the true attack slots are accurately anticipated as attacks) and f-score (the adjustment mean of the precision and recall).

$$Precision = \frac{TP}{TP + FN} \tag{14}$$

$$Recall = \frac{TP}{TP + FP} \tag{15}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{16}$$

The results in Figures 9 and 10 are the comparisons of the evaluation of the proposed model to the recent works that are considered as the state-of-the-art for intrusion detection and NLP issues. The accuracy shown on different popular datasets by different methods illustrates that the proposed approach produced better results than other methods. The evaluation results were obtained from four models; RNN, LSTM, BiLSTM and multihead attention network. The CICIDS-2019 dataset [34] is introduced by the Canadian Institute for Cybersecurity that contained captured data with timestamps of benign and attack networks. Another compared dataset is NSK-KDD, which is a dataset suggested to solve some of the inherent problems of the KDD'99 dataset and provided by the Canadian Institute [33]. To evaluate the proposed scheme, it was applied to three different datasets and three different models mentioned in Figures 9 and 10. For each dataset, the accuracy of every model was gained precisely with a high score. The multihead attention network scheme worked as well as BiLSTM on every dataset, but had faster training time and predicting time. RNN and LSTM had lower accuracy and longer times. This evaluation experiment verified the efficiency of the proposed multihead attention network scheme in the EDoS attack detection issue.

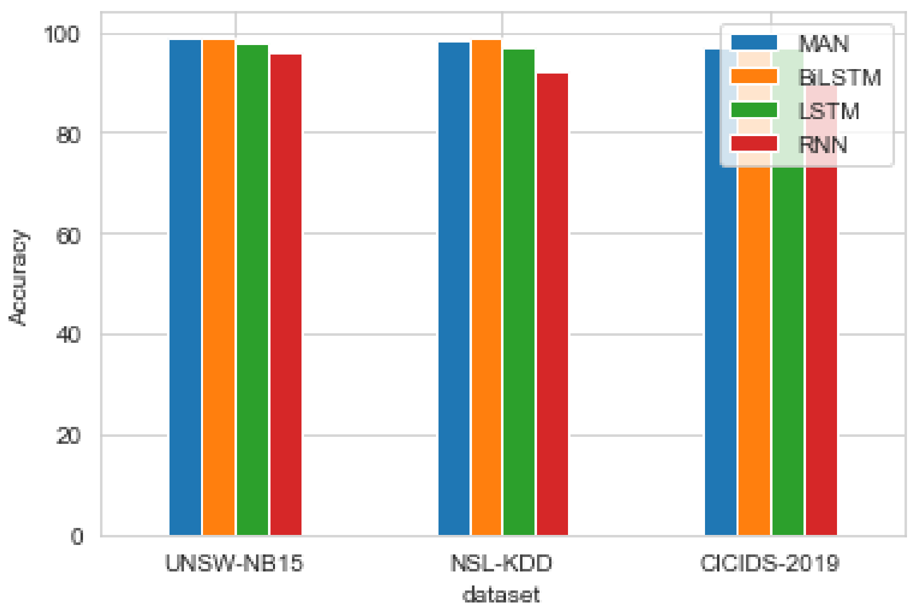

Figure 9. Accuracy evaluation on different datasets.

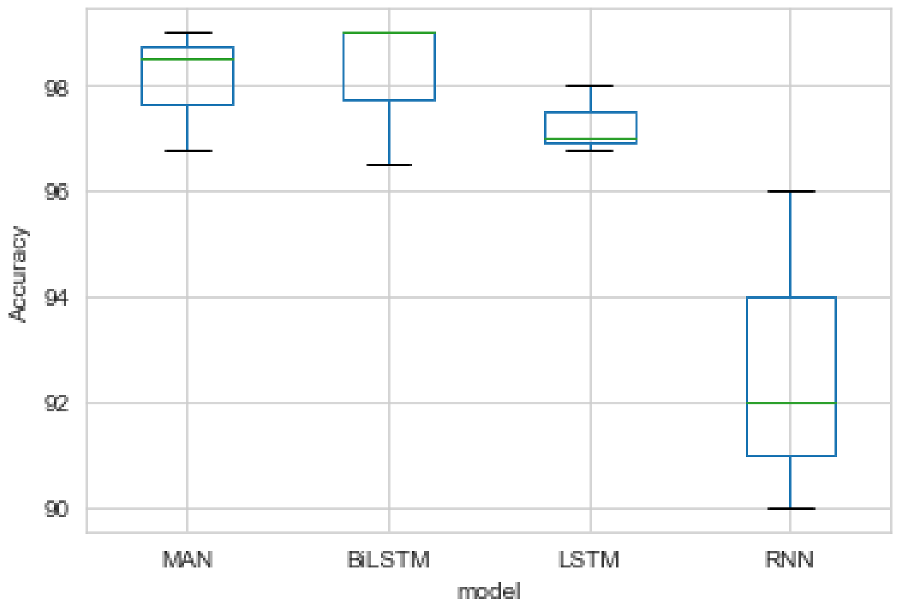

Figure 10. Distribution of quantitative of accuracy for different models in different datasets.

6. Conclusions and Future Works

We proposed a testbed scheme, approached by a multihead attention model, which works with a multivariate time series for EDoS attack detection in the network traffic environment. By calculating the query, key and value matrices, the multihead attention matrices scores are applied with neuron networks to be trained with the EDoS data. The proposed idea is established by improving the recent works of multivariate time series models applied for EDoS attacks in cloud computing. The results and evaluation in Section 5 showed that the proposed scheme is very efficient in terms of accuracy and accelerates the time required for training and predicting. The results have shown that our proposal gained not only better results, as well as a state-of-the-art mechanism, but also increased the training time and predicting time, though the gap in predicting time values was not large. This improvement speeds-up the processing time of the detection system of cloud computing, which is beneficial for the real-time detection of the system while the flooding attack is serious to the interface for an instance of time. Predicting time is very important in the detection system of cloud computing, since, in an instance of time, the EDoS attackers generate a thousand packets. Besides, the twice-as-fast training time is very significant when we want to update the system with new types of attacks by fine-tuning or training with newly-generated data. This time advantage comes from the proposed attention mechanism. With our proposed work, the detection system in cloud computing is able to prevent the EDoS attacks which have constituted emergencies for cloud service payments of many enterprises.

As future work, we plan to implement the scheme for multi-class attack detection, modify the model in a cloud computing environment, such as the Software Defined Network (SDN), and improve it for comparison with previous works using more evaluation criteria. Based on the characteristic of the SDN, the separate control plane helps us monitor the model to train and predict more precisely. The controller of SDN can extract the flow features easily and progress the detection system in real-time conveniently. On another hand, using other metrics and computed decision thresholds will increase the quality of the model to adapt better with other types of data that are related to the DoS attacks .

Author Contributions: Conceptualization, V.Q.T.; Project administration, M.P.; Supervision, M.P.; Writing—original draft, V.Q.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Research Foundation of Korea (NRF) grant number 2020R1F1A1076795.

Acknowledgments: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2020R1F1A1076795).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rambabu, M.; Gupta, S.; Singh, R.S. Data Mining in Cloud Computing: Survey. In *Innovations in Computational Intelligence and Computer Vision*; Advances in Intelligent Systems and Computing; Sharma, M.K., Dhaka, V.S., Perumal, T., Dey, N., Tavares, J.M.R.S., Eds.; Springer: Singapore, 2021; Volume 1189. [CrossRef]
2. El Kafhali, S.; El Mir, I.; Hanini, M. Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing. *Arch. Comput. Methods Eng.* **2021**, 1–24. [CrossRef]
3. Kuyoro, S.O.; Ibikunle, F.; Awodele, O. Cloud Computing Security Issues and Challenges. *Int. J. Comput. Netw.* **2011**, *3*, 247–255.
4. Sabahi, F. Cloud computing security threats and responses. In Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 27–29 May 2011; pp. 245–249. [CrossRef]
5. Liagkou, V.; Kavvadas, V.; Chronopoulos, S.K.; Tafiadis, D.; Christofilakis, V.; Peppas, K.P. Attack Detection for Healthcare Monitoring Systems Using Mechanical Learning in Virtual Private Networks over Optical Transport Layer Architecture. *Computation* **2019**, *7*, 24. [CrossRef]
6. Singh, P.; Rehman, S.U.; Manickam, S. Comparative Analysis of State-of-the-Art EDoS Mitigation Techniques in Cloud Computing Environment. *arXiv* **2019**, arXiv:1905.13447.
7. Zargar, S.T.; Joshi, J.; Tipper, D. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 2046–2069. [CrossRef]
8. Shawahna, A.; Abu-Amara, M.; Mahmoud, A.S.H.; Osais, Y. EDoS-ADS: An Enhanced Mitigation Technique Against Economic Denial of Sustainability (EDoS) Attacks. *IEEE Trans. Cloud Comput.* **2020**, *8*, 790–804. [CrossRef]
9. Ghanem, K.; Aparicio-Navarro, F.J.; Kyriakopoulos, K.G.; Lambotharan, S.; Chambers, J.A. Support Vector Machine for Network Intrusion and Cyber Attack Detection. In Proceedings of the 2017 Sensor Signal Processing for Defence Conference (SSPD), London, UK, 6–7 December 2017; pp. 1–5. [CrossRef]
10. Phan, T.V.; Park, M. Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud. *IEEE Access* **2019**, *7*, 18701–18714. [CrossRef]
11. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
12. Liang, X.; Znati, T. A Long Short-Term Memory Enabled Framework for DDoS Detection. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]

13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I.I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
14. Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; Kaiser, L. Universal Transformers. *arXiv* **2019**, arXiv:1807.03819.
15. Al-Haidari, F.; Salah, K.; Sqalli, M.; Buhari, S.M. Performance Modeling and Analysis of the EDoS-Shield Mitigation. *Arab. J. Sci. Eng.* **2017**, *42*, 793–804. [CrossRef]
16. Khor, S.H.; Nakao, A. Spow On-Demand Cloud-based EDDoS Mitigation Mechanism. In Proceedings of the 5th Workshop on Hot Topics in System Dependability, Lisbon, Portugal, 29 June 2009; pp. 1–6.
17. Masood, M.; Anwar, Z.; Raza, S.A.; Hur, M.A. EDoS Armor: A cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments. In Proceedings of the INMIC, Lahore, Pakistan, 19–20 December 2013; pp. 37–42. [CrossRef]
18. Chowdhury, F.Z.; Idris, M.Y.I.; Kiah, L.M.; Manazir Ahsan, A.M. EDoS eye: A game theoretic approach to mitigate economic denial of sustainability attack in cloud computing. In Proceedings of the 2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia, 4–5 August 2017; pp. 164–169. [CrossRef]
19. Shaaban, A.R.; Abd-Elwanis, E.; Hussein, M. DDoS attack detection and classification via Convolutional Neural Network (CNN). In Proceedings of the 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 8–10 December 2019; pp. 233–238. [CrossRef]
20. Li, Y.; Lu, Y. LSTM-BA: DDoS Detection Approach Combining LSTM and Bayes. In Proceedings of the 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD), Suzhou, China, 21–22 September 2019; pp. 180–185. [CrossRef]
21. Dinh, P.T.; Park, M. Dynamic Economic-Denial-of-Sustainability (EDoS) Detection in SDN-based Cloud. In Proceedings of the 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020; pp. 62–69. [CrossRef]
22. Roy, B.; Cheung, H. A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. In Proceedings of the 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 21–23 November 2018; pp. 1–6. [CrossRef]
23. Dong, S.; Abbas, K.; Jain, R. A Survey on Distributed Denial of Service (DDoS) Attacks in SDN and Cloud Computing Environments. *IEEE Access* **2019**, *7*, 80813–80828. [CrossRef]
24. Monge, M.A.S.; Vidal, J.M.; Perez, G.M. Detection of economic denial of sustainability (EDoS) threats in self-organizing networks. *Comput. Commun.* **2019**, *145*, 248–308. [CrossRef]
25. Geetha, K.; Sreenath, N. SYN flooding attack—Identification and analysis. In Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, 27–28 February 2014; pp. 1–7. [CrossRef]
26. Boro, D.; Basumatary, H.; Goswami, T.; Bhattacharyya, D.K. UDP Flooding Attack Detection Using Information Metric Measure. In *Proceedings of International Conference on ICT for Sustainable Development*; Advances in Intelligent Systems and Computing; Satapathy, S., Joshi, A., Modi, N., Pathak, N., Eds.; Springer: Singapore, 2016; Volume 408. [CrossRef]
27. Open vSwitch. Available online: https://www.openvswitch.org/ (accessed on 9 August 2016).
28. VirtualBox Oracle. Available online: https://www.virtualbox.org/ (accessed on 13 December 2012).
29. Sqalli, M.H.; Al-Haidari, F.; Salah, K. EDoS-Shield—A Two-Steps Mitigation Technique against EDoS Attacks in Cloud Computing. In Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing, Melbourne, Australia, 5–8 December 2011; pp. 49–56. [CrossRef]
30. BoNeSi, The DDoS Botnet Simulator. 2015. Available online: https://github.com/Markus-Go/bonesi (accessed on 2 December 2018).
31. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
32. Moustafa, N.; Slay, J. The evaluation of Net-work Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset. *Inf. Secur. J. Glob. Perspect.* **2016**, *25*, 18–31. [CrossRef]
33. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A. A Detailed Analysis of the KDD CUP 99 DataSet. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6. [CrossRef]
34. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Portugal, 22–24 January 2018.
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
36. Tan, M.; Iacovazzi, A.; Cheung, N.M.; Elovici, Y. A Neural Attention Model for Real-Time Network Intrusion Detection. In Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 14–17 October 2019; pp. 291–299. [CrossRef]
37. Liu, C.; Liu, Y.; Yan, Y.; Wang, J. An Intrusion Detection Model with Hierarchical Attention Mechanism. *IEEE Access* **2020**, *8*, 67542–67554. [CrossRef]