Intrusion Detection System: The Use of Neural Network Packet Classification

Nery Ruiz, Bryan Tavera, and Abdel-Shakour Abuzneid

Department of Computer Science and Engineering,
University of Bridgeport, Bridgeport, Connecticut, USA
e-mail: (neryruiz, btavera) @my.bridgeport.edu and abuzneid@bridgeport.edu

Abstract*-- Despite recent advances in cloud processing power and network connectivity to handle massive network traffic, networks are still vulnerable to Distributed Denial of Service (DDoS) attacks. With the recent proliferation of the Internet of Things (IoT), unsecured devices are fueling the ever-growing botnets, which allow creating larger malicious networks. Current mitigation techniques need to adapt to a new growing size of zero-day attacks to protect network services to consumers and block malicious connections. Deep learning enables machines to find the solution to many complex problems. This paper evaluates the performance of the Simple Neural Networks, Convolutional Neural Networks, and Recurrent Neural Networks in detecting DDoS attacks when trained with the CSE-CIC-IDS2018 Dataset. This research will discuss the presented datasets and the efficiency of the proposed networks. The trained data was obtained from a realistic dataset that holds different forms of intrinsic volume, protocol, and web-based attacks.

Keywords: DDoS, SDN, Deep Learning, RNN, LSTM, IDS/IPS, CNN, PCA

I. Introduction

A DDoS attack is best known to be a malicious attempt to disrupt normal traffic of a targeted server or network by overwhelming their network with unusual traffic. Its goal is to compromise the target system's ability to handle any request for its services. This has been further amplified by the exploitation of insufficiently protected machines and other network devices, such as devices in IoT. The attack will inevitably attempt to make an online service unavailable to its users, typically interrupting or suspending the services of its hosting servers. There are many types of DDoS attacks, and they all fall into three categories: volume, protocol, and web-based attacks.

Volume-based DDoS attacks employ brute force to send large amounts of data to a targeted system, causing network congestion. Network congestion occurs when the system reaches its limit of processing incoming network traffic. This attack doesn't just affect the target system, but it also affects the Internet Service Provider (ISP) of the target network. Most ISPs have ways to limit the amount of traffic that reaches the targeted network, which can mitigate a volume-based DDoS attack. Protocol-based attacks are hard to combat since they attack the fundamental protocols in the network layers of the Open System Interconnection (OSI) model. The protocols used within the network layers, such as IP, ICMP, TCP, UDP are standardized and used widely to be compatible to allow internetworking. Weaknesses of these protocols are well-known and easy to exploit.

Services on older platforms may have many vulnerabilities that were not necessarily identified at the time. With time, people with malicious intent find new vulnerabilities and exploit those systems. For example, some vulnerabilities affect the way that a web server handles connected users, like creating new threads per new connection. As an example, the SlowLoris DDoS attack will try to use up as many new additional threads as possible and slow down the host capability to handle them. This is one of the most challenging application level DDoS attacks to detect with reasonable means. The purpose of using neural networks in detecting such attacks is that they are flexible and typically produce a high accuracy classification. Compared to other machine learning algorithms, neural networks can learn abstract features that can differentiate new attacks autonomously. With these characteristics, neural networks should produce the best result for detecting a SlowLoris attack.

In summary, the main contributions in this work are the following: training and testing different neural network topologies with several attack DDoS datasets. We will evaluate the packet classification accuracy and compare the performance of different Neural Networks. Section II of this article is a survey of the past related work on training approaches, packet classifiers, and neural networks. Section III provides a background of different network topologies structures and known DDoS attacks. Section IV defines the methodology of designing the neural network and approaches to preprocess data. Section V contains the discussion of implementation approaches used. Section VI presents a discussion of the results and suggests improvements. The final section summarizes and discusses possible future applications.

II. Related Works

There has been much work done in comparing different methods for the detection of intrusive attacks and mitigation techniques. The most common methods used are machine learning techniques and deep learning algorithms. The authors in [1] use a simple neural network to detect DDoS attacks, where they examined the use of different back-propagation

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00239

algorithms and their impacts on classification performance. They also test different hidden layer sizes and record their effects on performance. Based on the sample size, the accuracy of the classifier was much higher with a smaller hidden layer size, than a larger hidden layer size in this application.

Researchers from [2] explored the use of Convolutional Neural Network (CNN) to predict traffic profiles by using just a small collection of packets. The authors collected 80 bytes from each packet and classified the traffic. Based on the results from the Dataset that they used, they received just about 100% accuracy. The researchers in [3] explored the use of an RNN with the implementation of a combination of Long-Short Term Memory (LSTM) and Bayes algorithm to detect DDoS attacks.

The method was to have the RNN with LSTM to classify the collected data and then pass it to a Bayes module to re-classify the output of the RNN. The result showed that the difference in performance is minimal, and for the cost of extra computation, it is not necessary for this application. The research done in [4], proposed a reduced CNN to save computational resources. This made the CNN capable of running on resource-constrained devices for DDoS attack detection. The author's results also showed that the reduced method provided more efficient results in this application.

III. Background

A. Network and Cloud Structure

In recent years, cloud as a service usage has grown in popularity for enterprises and governments. Most of these cloud service infrastructures use large networks with thousands of servers. Most of these networks are built on the traditional model networks, which are based on the Transmission Control Protocol and Internet Protocol (TCP/IP) model. The TCP/IP model traditionally resembles the seven layers listed under the OSI model: applications combined with presentation and session layer, transport, network, data link, and the physical layer. One must acknowledge that any traditional network model is limited by its hardware. In other words, a traditional network model is based on physical components like switches and routers. The network will inherit the physical limits of the components capability that the network is being executed on.

Large companies like Amazon and Google have cloud as a service business model. To implement these services, these companies are implementing software defined networks (SDN) to provide flexible network services that are not restricted by hardware. In other words, SDN are networks that do not depend on hardware like the traditional network architecture. SDN uses three software defined layers: application, control and infrastructure layer. Large cloud services like Amazon Web Services (AWS) and Google Cloud Platform (GCP) are built in a similar topology as an SDN. SDN allows more flexibility than traditional networks, which is key to the cloud as a service model. However, this topology can introduce application-based vulnerabilities that are hard to detect and resolve in traditional ways.

B. DDoS Attack Types

One kind of DDoS attack is a volume-based attack, which is one of the most common of all DDoS attacks. Attackers utilize many computers and internet connections to flood a website with overwhelming amounts of bandwidth. As a result, legitimate traffic is unable to pass through, and attackers can take down any website successfully. Some examples of a volume-based attack is a UDP flood in which an attacker overwhelms random ports on the targeted host.

Another kind of DDoS attack is a protocol attack, and this attack aims to exhaust server resources. Protocol attacks mainly attack between servers and websites, such as firewalls and load balancers. Attackers overwhelm websites and server's resources by making fraudulent protocol requests to consume the available resources. An example of a protocol-based attack would be a Smurf DDoS attack. The way that it is interpreted is that the attackers can exploit the Internet Control Message Protocol (ICMP) packets. Within these packets, it contains the end user's IP address, and then it broadcasts to the computer network. The victim's computer can immediately get flooded with traffic. This type of attack targets vulnerabilities within applications such as Apache, Windows, and OpenBSD. The way that this DDoS attack is that it initially takes down servers by making a large number of requests by mimicking the user's traffic behavior.

Lastly, application-layer attacks are another kind of DDoS attack, and they mainly look to disrupt specific features and or functions of a website, such as online transactions. What makes these attacks so unnoticeable is that the application layer targets specific application packets during the attack. An example of an application layer attack is SlowLoris. The way SlowLoris works is that it holds open many HTTP connections to the server for as long as possible. When establishing a connection, the attacker first opens to the targeted server by sending multiple partial HTTP request headers. By continually sending these requests, the system will eventually timeout the exceedingly long connection, freeing the thread up for the next request. To prevent the target from timing out the connections, the attacker will consistently send partial request headers to the target to keep the request alive. Overall, without the proper defense system being implemented, the system will result in DoS.

C. Intrusion Detection, Firewall and Prevention Tools

An intrusion detection system (IDS) is a device or software application that monitors a network for malicious activity. It can detect and report malicious activity or violation using security information and event management systems. Network, Host, Protocol-based, and application protocol-based intrusion detection system are the five types of IDS. Network intrusion detection system (NIDS) is a type of IDS that has network taps installed at critical locations on the network to examine traffic from all devices within the network. The host intrusion detection system (HIDS) runs on servers that are connected to the network and intercept traffic by routing traffic through it. Protocol-based intrusion detection systems (PIDS) are an agent that resides at the front end of a server. This system controls

and interprets the protocol between a device and a server. Application protocol-based intrusion detection system (APIDS) is defined as a system that monitors the communication within a group of servers to interpret the application's specific protocols.

The method used by IDS to detect abnormal or malicious traffic is either by signature-based or anomaly-based. The signature-based method is to detect specific patterns of bytes. The IDS could have a library of known signatures that are from known malicious traffic to compare to the incoming traffic. Anomaly-based detection method is an IDS that uses machine learning techniques to detect unknown attacks. The use of machine learning techniques can create a model to classify normal traffic patterns and signal against abnormal traffic. These models are trained according to network applications and network configurations. The primary function of a firewall is to control and monitor incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and untrusted external networks. Firewalls work by trying to match the network traffic against the ruleset that is defined within its table.

D. Deep Learning and Multivariable Classification

Artificial neural networks are a concept to mimic the brain's neural function. The concept of artificial neural networks is also known as deep learning. The brain is made up of biological neurons to process data, which then passes it to the next neuron. In comparison to the artificial neural networks, this network is built upon mathematical functions that are stacked in layers. In biology, a neuron has three basic parts to carry out the functions of integration: dendrites, cell body, axons, and axons terminals. Artificial neural networks have similar parts, just like a biological neuron. Dendrites collect data from input nerves or other neurons. An artificial neuron mimics this functionality by weighting the input values by applying trained weights to each input value. When the input values are preprocessed, the artificial neuron sums up all the values with a trained bias value (1).

$$Sum\ of\ all\ inputs\ =\ Bias\ +\sum_{i=1}^{n}\quad W_i I_i \qquad (1)$$

The job of a biological neuron's cell body is to figure out if the data is important enough to pass a strong signal. If the input of that neuron is important, the body cell will send a larger signal to the rest of the other neurons in the link. The same happens in the artificial neuron. However, the output of the sum from the "dendrites" is passed to an activation function. This activation function will return the sum of a standardized value between 0 to 1 (or between other values depending on the type of activation function used as shown in Table I).

Table I: Common Activation Functions.

Sigmoid	$Activation\ (x)\ = \frac{1}{1+e^{-x}}$
Hyperbolic Tangent	$Activation\ (x)\ =\ tanh(x)$
ReLU	$Activation\ (x)\ =\ max\ (0,x)$
Leaky ReLU	$Activation\ (x)\ =\ max\ (0.1x,x)$

The axons portion of the neuron collects the processed input and sends a meaningful signal to the axon's terminals. The axon terminals act as a distributor of the output to other neurons. When more groups of artificial neurons are stacked together, they form "hidden" layers. They are called hidden because the values of those neurons are not measured as final outputs. The last layer of the network is the output layer. To use neural networks as a binary classification, the output layer will have only one output neural node. This node could have the same activation function as the rest of the network. However, if the neural network will be classifying different classes, the output layer will have the same number of neural nodes as the number of different classes. The final layer will use a different activation function called the SoftMax function. The SoftMax function will create the desired output to be differentiated from the rest of the outputs. SoftMax Equation is defined as (2).

$$Activation(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K}\ e^{x_i}} \qquad (2)$$

Convolutional Neural Networks (CNN) are neural networks that can handle multi-dimensional input like images. The common application of a CNN is image recognition. The purpose of the convolutional layer is to reduce the size of the data by filtering out noise and sharpening its features. The convolutional layer has a filter matrix parameter called kernels. This matrix has trained values to filter the input data and extract data that is needed. The size of the kernel matrix also ends up affecting the output size of that layer. For example, we can have the input data with dimensions of 100x100; the kernel could be 5x5; therefore, the output would be 98x98. Convolution Equation is defined as (3).

$$(f\ \cdot\ k)(t) = \int_{-\infty}^{\infty}\quad f(t-\tau)k(\tau)d\tau \qquad (3)$$

The output of the convolution layer passes along to a pooling layer. Common pooling functions are averaging and max functions. The way it works is that an average pooling helps function return the average of overlapping sections of the input matrix, which further reduces the size of the input matrix. Now, the way max pooling function works, it returns the maximum value of overlapping sections of the input matrix. For network traffic classification, CNN should be able to classify large groups of packets and collect the most significant features that are distinctive.

However, some neural networks have special properties for different applications like time sequence classification. Recurrent neural network (RNN) is a time sequence classifier that loops back from its past output and takes that as part of its new input. This characteristic gives this type of neural network the ability to retain a memory of the past inputs and then use it with the current input to predict the next likely output. With this feature, it gives the RNN the ability to classify sequences of data. Some of the common uses of RNN are voice and video classification. Long Short-Term Memory (LSTM) is a type of RNN that introduces filtering to the feedback loop. This behavior adds more computational overhead but allows the

classification of complex sequences. Some of the common applications of LSTM is language recognition.

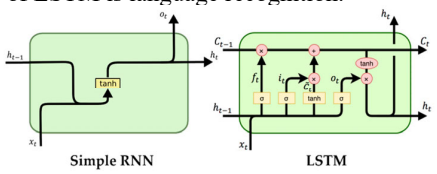

Figure 1. Functional diagram of RNN and LSTM.

IV. Methodology

The main problem that needs to be addressed is how DDoS attacks can be minimized or at least be detected. There are specific measures that should be taken into perspective when detecting such activity: DDoS attack is a kind of network intrusion attack that congests the target network or brings the targeted network down to a stop. The goal of an efficient intrusion detection system is to detect patterns of the attack by analyzing the incoming network traffic behavior.

A. The Dataset

The Dataset that we focused on was the CSE-CIC-IDS2018[5, 6]. This data set was generated to represent real world traffic. The developers created servers and users on the AWS cloud services. The goal was to generate a diverse and well comprehended model for intrusion detection based on the creation of various user profiles. These user profiles contained abstract representations of events and behaviors that were seen through the network. In this case, the profiles were combined to help generate a diverse set of unique features within the Dataset for best practical use. The Dataset is made up of raw data captures and the total size of the Dataset compressed, exceeded a total amount of 486.1 GB.

The Dataset also portrays multiple different attack scenarios. The total seven listed attacks demonstrated in the Dataset are brute-force, Heartbleed, botnet, DoS, DDoS, web attacks and infiltration attacks. This Dataset even includes the captures of network traffic and system logs of each machine. Attacks such as SlowLoris were utilized in the Dataset. The capturing and final data was calculated through a list of daily attacks, specifically on Fri-16-02-2018. The Dataset is broken up by dates, while the attacks separated by time.

A suitable method of feature extraction within any dataset particularly would be the CICFlowMeter[6]. This program is a network traffic flow generator that helps choose various features in which a user decides to calculate their data. One is able to select features, calculate them, and even add new ones if needed, all in favor of having better control of duration during flow timeout. Some of these features are listed as duration, number of packets, number of bytes, and length of the packet. The output of the application is in a CSV file format with each row labeled by FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol, each having more than 80 network traffic features.

B. Data Preprocessing

Data preprocessing is an important topic when training any machine learning algorithm. The method chosen to process the data will inevitably impact the machine learning performance. In the past research, it explored training neural networks with raw packets with labels. A few obstacles when giving raw packet data to a neural network is finding the best size for the application and the style of padding. This is because a trained neural network will have a fixed size input, and when packets are smaller than the input size, padding becomes an important factor within the preprocess. There are three ways of padding raw packets. One padding method is to pad at the beginning and end of the packet with zeros. The second padding method is to pad from the end of the packet with zeros. The final padding method is to repeat the packet over until it reaches the correct size.

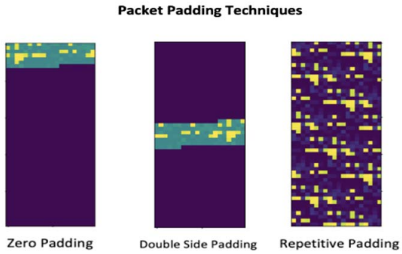

Figure 2. Visualizations of packet padding techniques.

Another approach to preprocess network traffic data is to collect the overall specification of each connection. Some aspects that can be obtained from the captured traffic are connection durations, the rate of transfer, average packet length, and network flags. This approach of processing the network data creates a table of the overall characteristics of the network connections. Taking a data mining approach will help allow the traffic of characterized clusters to classify specific network behavior. The CICFlowMeter-V4[6] tool is capable of mining features from targeted network captured files. This tool returns a Comma-Separated Values (CSV) file with collected features.

C. Neural Network Models

The Dataset that was used was CSE-CIC-IDS2018. These datasets generate data that simulate different DDoS or other intrusive attacks and this Dataset in particular, represents real-world scenarios. Three Neural Networks will be tested to evaluate their performance against each other. Simple Neural Network and Convolutional Neural Network were heavily researched in the past and proven to be acceptable classifiers for this application. On the account that RNN includes previous generated results in order to define the subsequent results. Recurrent neural networks as a network behavior classifier should provide better performance for analyzing network traffic data. A sequence of packets is an essential factor to classify traffic behavior. While training the RNN the data cannot be shuffled at each epoch. Epochs are best defined as the number of iterations or steps that are needed to partition batches of training data.

V. Implementation

We explored different preprocessing methods to validate the performance of each neural network. We tested various styles of padding for each neural network like zero, double side and repetitive padding in order to train the data. Afterward, we followed the data mining approach and extracted features from the raw traffic captures and used the data to train the neural networks. While we were performing the analysis of the raw data, we came across an algorithm to reduce the dimensionality of the collected data. To represent the data, we used the Principal Component Analysis (PCA.)[7] This is a linear dimensionality reduction technique that is used for the extraction of information from a high-dimensional into a lower-dimensional subspace. The primary usage of this technique is for leveraging and speeding up our machine algorithm's training and testing time.

Our code primarily used the open-source library TensorFlow, Scikit-learn, and Keras. Scikit-learn is a python machine learning library that implements a range of machine learning, preprocessing, cross-validation, and visualization algorithms. This library is made up of key components, such as featuring various classification, regression, and clustering algorithms. TensorFlow is a python-friendly open-source library for numerical computation, which helps make machine learning faster and a lot easier to use. TensorFlow is capable of training and running deep neural networks for data classification and sequence-to-sequence models. We used Keras library to create the layers of the neural networks. This sped up development when adjusting the size and structure of the neural networks. TensorFlow has CUDA support to use GPU compute power to train the NNs. Using hardware acceleration for training reduces waiting time to see results.

The hardware we used was an Intel Core i7 6700K with 16GB of DDR4 RAM and a Nvidia GTX 1080 GPU with 8 GB GDDR5 RAM to train the NNs. We also used an enterprise-level server to handle multiple training sessions at the same time. The server had two Xeon E5-2620 and 64 ECC RAM. After training, we collected the outputs from the neural networks for the classification analysis. We evaluated the packet classification accuracy and the effect of normal traffic flow according to the rate of true and false positives. We analyzed the performance of each neural network by using a confusion matrix. A confusion matrix is a technique that helps visualize the performance of a classification algorithm. A classification algorithm is a function that weighs the input features and sorts it according to its class. We calculated our confusion matrix by collecting the actual outputs from the classifier and comparing them to the expected outputs. DDoS attacks were detected between within assigned time intervals. Then, we were able to compare the binary output to the expected output within a confusion matrix.

VI. Result and Discussion

Three models were created and trained with the CSE-CIC-IDS2018 Dataset [5]. The output was collected from each trained model and analyzed it with the expected output with a confusion matrix. Some of the key points from our results consisted of *precision*, *recall*, and *F1-score*. *Precision* is defined as the number of true positives divided by the sum of true positives and false positives. *Recall*, on the other hand, reveals the ability to find all relevant instances in a data set. It is defined as the number of true positives divided by the sum of true positives and false negatives. *F1-score* is the measure of our test's accuracy. Mathematically, the F1-score (4) is the mean of the test's precision and recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \qquad (4)$$

TABLE II Zero Padding: Conventional Neural Network

Traffic Type	Precision	Recall	F1-Score
Normal	0.98	0.92	0.95
Attack	0.27	0.68	0.39

Simple Neural Network total accuracy = 0.9044%

TABLE III Zero Padding: Convolutional Neural Network

Traffic type	Precision	Recall	F1-Score
Normal	1.00	0.90	0.95
Attack	0.08	1.00	0.15

Convolutional Neural Network total accuracy =0.8979%

TABLE IV Repetitive Padding: Conventional Neural Network

Traffic type	Precision	Recall	F1-Score
Normal	0.88	0.92	0.90
Attack	0.42	0.31	0.35

Simple Neural Network total accuracy = 0.8298%

TABLE V Repetitive Padding: Convolutional Neural Network

Traffic type	Precision	Recall	F1-Score
Normal	0.94	0.92	0.93
Attack	0.33	0.42	0.37

Convolutional Neural Network total accuracy =0.8748%

Reviewing the data that we collected in tables [II, III, IV, V], we concluded that the Simple Neural Network performed best for both types of padding. The overall accuracy of the classifiers was high but, when comparing zero and double-sided zero paddings, they produced similar results with both classifiers. However, the simple and convolutional neural networks trained with repetitive padding had lower overall accuracy in contrast to the classifiers trained with zero padding. Although, the repetitive padding improved the classifier's precision performance for classifying attack scenarios.

While training, overfitting was an issue with this data. This stems from the fact that the data had a larger set of normal data traffic than it did attack data. To mitigate overfitting, noise

layers were deployed at the beginning of each network. The dropout layers were also deployed between each layer and regularizers were deployed for each NN. The Gaussian noise layer and the L1 regularizers were used from the Keras library. We trained our Recurrent Neural Network with different styles of padding techniques, but the RNN did not produce consistent results. The RNN, with this style of preprocessing, was not able to classify the raw packet sequences. We concluded that the RNN could not be trained with this type of approach to preprocessing the input data. We explored a data mining tool called CICFlowMeter, this tool pre-extracted features that can be used on the RNN. Analyzing the CSE-CIC-IDS2018 Dataset, gave us a lot of scenarios to work with and each file has captures from different IP addresses. This lowered a chance of the NN to overfit the model from memorizing the IPs and ports used.

VII. Conclusion

After exploring different deep learning mechanisms to implement packet classification, we concluded that using simple neural networks produced more effective results, thus confirming the result stated in [1]. The simple neural network provided an accuracy of 82% along with a percentage of 42% precision for detecting malicious packets. We were able to explore different approaches that included convolutional neural networks, simple neural networks, and recurrent neural networks. Multitude of tests gave us insights on how differently these algorithms process the data and information.

For future research, it is vital to investigate better data mining approaches that could better extract significant features before the data is used for training. The reason for this is because we noticed the approach of providing raw packets to these classifiers often did not give expected results and produced many overfitting issues. The accuracy was only able to reach a certain percentage because of overfitting. Overfitting resulted in overtraining the data for the normal scenario which resulted in high false positives and negative rates. In conclusion, better representation of training data can improve the performance and accuracy of network traffic classification with neural networks

[1] O. Ali and P. Cotae, "Towards DoS/DDoS Attack Detection Using Artificial Neural Networks," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2018: IEEE, pp. 229-234.

[2] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection," *IEEE Access,* vol. 8, pp. 30387-30399, 2020.

[3] Y. Li and Y. Lu, "LSTM-BA: DDoS Detection Approach Combining LSTM and Bayes," in *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, 2019: IEEE, pp. 180-185.

[4] R. Doriguzzi-Corinα, S. Millarβ, S. Scott-Haywardβ, J. Martınez-del-Rincónβ, and D. Siracusaα, "LUCID: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," 2019.

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018, pp. 108-116.

[6] A. H. L. Iman Sharafaldin, and Ali A. Ghorbani. "CSE-CIC-IDS2018 on AWS." https://www.unb.ca/cic/datasets/ids-2018.html (accessed January 10th, 2020).

[7] R. Abdulhammed, M. Faezipour, H. Musafer, and A. Abuzneid, "Efficient network intrusion detection using pca-based dimensionality reduction of features," in *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, 2019: IEEE, pp. 1-6.

[8] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics,* vol. 8, no. 3, p. 322, 2019.

[9] Y. Afek, A. Bremler-Barr, and S. L. Feibish, "Zero-day signature extraction for high-volume attacks," *IEEE/ACM Transactions on Networking,* vol. 27, no. 2, pp. 691-706, 2019.

[10] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE sensors letters,* vol. 3, no. 1, pp. 1-4, 2018.

[11] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. Alessa, "Effective features selection and machine learning classifiers for improved wireless intrusion detection," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 2018: IEEE, pp. 1-6.

[12] N. G. Dharma, M. F. Muthohar, J. A. Prayuda, K. Priagung, and D. Choi, "Time-based DDoS detection and mitigation for SDN controller," in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2015: IEEE, pp. 550-553.

[13] V. Kanimozhi and T. P. Jacob, "Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, 2019: IEEE, pp. 0033-0036.

[14] A. Luo, W. Huang, and W. Fan, "A CNN-based Approach to the Detection of SQL Injection Attacks," in *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*, 2019: IEEE Computer Society, pp. 320-324.

[15] A. Maslan, K. M. Mohammad, F. B. M. Foozy, and S. N. Rizki, "DDoS Detection on Network Protocol Using Neural Network with Feature Extract Optimization," in *2019 2nd International Conference on Applied Information Technology and Innovation (ICAITI)*, 2019: IEEE, pp. 60-65.

[16] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019: IEEE, pp. 1-8.

[17] M. Essaid, D. Kim, S. H. Maeng, S. Park, and H. T. Ju, "A Collaborative DDoS Mitigation Solution Based on Ethereum Smart Contract and RNN-LSTM," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019: IEEE, pp. 1-6.

[1-5, 7-17]