

DDoS Attacks Detection with AutoEncoder

Kun Yang*
High Speed Network Lab
New York University
New York, USA
ky13@nyu.edu

Junjie Zhang*
Fortinet, Inc.
California, USA
jjzhang@fortinet.com

Yang Xu
School of Computer Science
Fudan University
Shanghai, China
xuy@fudan.edu.cn

Jonathan Chao
High Speed Network Lab
New York University
New York, USA
chao@nyu.edu

Abstract—Although many distributed denial of service (DDoS) attacks detection algorithms have been proposed and even some of them have claimed high detection accuracy, DDoS attacks are still a major problem for network security. The latent and inherent problems of these detection algorithms are 1) Requirement of both normal and attack data for building detection models, and 2) Almost inability to detect novel and unknown DDoS attacks. To conquer the problems, this paper proposes an AutoEncoder based DDoS attacks Detection Framework (AE-D3F), which only uses normal traffic to build the detection model and is able to update itself automatically as time goes. Experimental results on synthetic and public traffic show that our AE-D3F can not only achieve 82.00% detection rate (DR) with 0 false positive rate (FPR), better than classical anomaly detection approaches, but also detect novel and unknown attacks.

Index Terms—Machine Learning, Anomaly Detection, Deep Learning, DDoS.

I. INTRODUCTION

Distributed denial of service (DDoS) attacks have been an intriguing and challenging problem against the industrial and the academia since their first occurrence around the year 2000 [1], [2]. The goal of these attacks is to exhaust processing and connectivity resources of the targets to prevent legitimate users' access to the services provided by the corresponding company, thus causing partial or total unavailability and revenue loss [3], [4]. For example, according to Kaspersky Lab research [5], the financial impact of a DDoS attack is continuing to rise globally - totaling over \$120K for small and medium-sized businesses (SMBs) and costing enterprises over \$2M, per attack on average. Hence it is necessary to efficiently and effectively detect DDoS attacks in order to reduce revenue loss caused by the attacks. In recent years, more and more DDoS detection methods based on machine learning are being proposed [6]–[9] due to the fact that great achievements have been made by machine learning algorithms in various areas such as network security, credit card fraud detection, medical images diagnosis and so on [10]. And these detection methods can be mainly divided into two categories: supervised learning based detection methods and unsupervised learning based detection methods.

Supervised learning based detection methods build detection models based on both normal and attack data. Their objective

is to optimize the decision boundary so that the detection models can achieve low classification error on train set and good generalization capability on test set. These methods usually achieve expected high detection accuracy by being provided well labeled and balanced datasets [11]–[13]. However, it is very difficult, if not impossible, to collect and label attack traffic data in practice because attack data for training is very rare. Thus, supervised learning based methods suffer from the train set imbalance issues, when building their detection models. Some researchers try to augment train data by including synthetic attack data or attack data from other networks [14]. However, traffic patterns in different networks may vary significantly. The knowledge acquired from one network cannot be simply applied to another network. Experiments in Section IV-C2 show that training a supervised learning model by including attack data from other networks or synthetic attack data does not work. In addition, attacking manners and patterns used by attackers are volatile [15]. Novel and unknown attacks constantly occur. Unfortunately, supervised learning based methods cannot guarantee their performance on new attacks that did not appear in the train set. The above limitations reveal that supervised learning based methods are not suitable for the DDoS detection.

Unsupervised learning based detection methods build detection models solely based on normal data. This saves them from the hassle of collecting and labeling attack data. The basic idea of this type of detection methods is to separate anomalous traffic from normal traffic by profiling normal traffic behaviors. If a new sample deviates from normal traffic patterns, the sample would be classified as an outlier or an anomaly. Hawkins [16] defines an outlier as an observation that deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism. By measuring sample deviations, this type of method is able to detect new and unknown DDoS attacks. It appears more promising to utilize unsupervised learning based methods for DDoS attack detection. However, to make this type of detection methods work, two assumptions should hold true. (1) Normal data follows similar behaviors or patterns, so that machine learning models can be used to profile normal data behaviors; (2) Anomalous data is separable from normal data from the statistical point of view.

In this paper, we propose an AutoEncoder (AE) based DDoS Detection Framework (AE-D3F). AE is one of classical

* ** holds the same importance.

artificial neural networks (ANN) used to learn efficient data representation in an unsupervised manner [17]. AE consists of two components: encoder and decoder. The encoder encodes input data to a latent low dimensional space, then the decoder tries to recover or reconstruct the coding data back to the original input data by minimizing the "distance" (error) between the reconstructed data and input data. AE measures the sample deviations on the basis of their reconstruction errors.

The main contributions of this paper are summarized as follows:

1) By defining some key features (shown in Tab. I) to represent traffic behaviors, we show that normal traffic patterns are stable within a certain network. Thus, we can profile normal traffic behaviors using an AE model.

2) Extensive experiments show that the reconstruction errors of attack traffic is relatively larger than those of normal traffic. This indicates that the proposed AE model is able to separate attack traffic from normal traffic according to reconstruction errors.

3) To design an effective and practical DDoS detection framework, false positive rate (FPR), a key metric to measure the proportion of normal data recognized as attack data, should be kept below an acceptable limit. Meanwhile, to do real-time attack detection, sample inferences should be fast enough. Taking these two factors into account, we design a simple and effective neural network for the proposed AE model.

Experimental results show that the proposed approach can not only achieve 82.00% detection rate (DR) with 0 false positive rate (FPR), better than other classical anomaly detection approaches (principal component analysis, isolation forest and one class support vector machine), but also detect unseen attacks on different test sets.

The remainder of this paper is organized as follows: Section II reviews the state-of-art works in this field. Section III explains the proposed framework in detail. Section IV presents experiment results based on different scenarios and analyzes the results. Lastly, Section V concludes our work and outlines future work.

II. RELATED WORK

This section focuses on the state-of-art works on DDoS detection proposed in the latest years, and almost all of them are based on machine learning algorithms. Chalapathy et al. [18] present a structured and detailed overview of research methods in deep learning-based anomaly detection. They also outline open issues in research and challenges faced while adopting deep anomaly detection techniques for real-world problems. Even though a reasonable amount of work has gone into detecting and defending against application layer DDoS, they still remain a major threat because of the difficulty in adopting the defenses.

Paffenroth et al. [19] use robust principal component analysis (RPCA) to detect anomalies. Experiments on DARPA dataset show that the proposed method achieves low FPR while maintaining reasonable true positive rates on individual

packets and further supports the hypothesis that the low dimensional subspace computed by RPCA is more representative of normal data. However, they use IP addresses and port numbers as part of features, which could be spoofed in practice. Latah et al. [20] propose a 5-level hybrid classification system based on flow statistics in order to attain an improvement in the overall accuracy of the detector. The results of experiments conducted on NSL-KDD dataset show that the proposed approach significantly enhanced the overall accuracy (84.29%). However, the proposed approach is one of the supervised classification approaches, i.e., it needs labeled data to build the detection model.

Pratomo et al. [21] propose the use of the autoencoder method for network intrusion detection. It does not need labeled datasets with both legitimate and malicious traffic for training purposes. Experiments show that the proposed approach is able to detect 100% of low rate attack traffic with an average false positive rate of 8.01%. Although the proposed approach is able to detect 100% of attacks, an average false positive rate of 8.01% is relatively high. Madani et al. [22] study the viability of using deep AE in the detection of anomalies in adaptive intrusion detection systems (IDSs), as well as their overall robustness against adversarial poisoning. Experimental results show that the proposed autoencoder based IDS outperforms a generic PCA-based counterpart by more than 15% in terms of detection accuracy. However, too many features (e.g., 122 features) are used to train their model, which will increase time complexity of the whole detection system, especially for feature extraction. Chen et al. [23] introduce AE ensembles for unsupervised outlier detection. Experiments to compare their proposed approach with state-of-the-art detectors are conducted on several benchmark data sets. The results show that the proposed approach achieves better detection accuracy than other detectors. However, their proposed approach integrates multi-autoencoders to a single model to detect anomaly samples, and hence, time complexity of training and updating the detection model is very high. Moreover, it will take more time to do the detection for each incoming flow. Shone et al. [24] propose a deep learning classification mode, which uses stacked AE to do feature reduction and Random Forest as a classifier. Experimental evaluations on the benchmark KDD Cup 99 and NSL-KDD datasets show the proposed approach offers up to 5% improvement in accuracy and up to 98.81% reduction in training time. However, the proposed approach is also one of the supervised classification approaches, i.e., it needs labeled data to build the detection model. Meidan et al. [25] propose a neural network-based anomaly detection method which extracts behavior snapshots of the network and uses deep AE to detect anomalous network traffic emanating from compromised Internet of Things (IoT) devices. This work mainly focuses on IoT anomaly detection, and it requires to use some expensive features, such as variance and correlation coefficient, which cannot be easily extracted from incoming packet stream.

As mentioned above, several AE-based detection models have been proposed for anomaly detection. However, most of

them focus on using AE to learn better and sparse feature representation, and then complement AE with other supervised machine learning algorithms, such as random forest, to build detection models. Essentially, these detection models treat DDoS detection problems as supervised classification tasks, and thus they cannot guarantee to get a good performance on zero-day or unknown attacks even though they are claimed to have high accuracy on attack data appears in train set. In contrast, we design an effective and unsupervised DDoS detection framework directly based on AE, which is easy to implement (only five layers in the AE model) and solely needs normal data to build the detection model. Moreover, our model is able to detect unknown attacks.

III. METHODOLOGY

In this section, we present the AE-based DDoS detection framework (AE-D3F) in detail. Our work is on per subflow basis, and the proposed detection framework, shown in Fig. 1, includes two components: feature extraction component (FEC) and online detection component (ODC). FEC is responsible for constantly and efficiently extracting all necessary traffic features from raw packets (data plane development kit (DPDK) [26] is used to speed up this process). ODC analyzes and identifies each anomaly sample on the basis of its reconstruction error generated by the AE model (more details can be seen in subsection III-B).

A. Feature Extraction Component

FEC extracts all necessary features from raw packets to keep track of traffic behaviors. Instead of analyzing network flows, we extract features and analyze traffic behaviors of network subflows to reduce the response time for DDoS attacks.

A flow represents a conversation between a client and a server. In our design, a flow is divided into several subflows according to a certain threshold, such as a fixed interval (e.g., 10 ms) or the number of packets (e.g., 200), which can be obtained by analyzing historical traffic. Thus, a subflow sample we used can be represented as follows:

$$subflow_i := \langle ts_i, flow_id, features_i \rangle \quad (1)$$

where, ts_i is the start time of the i -th subflow of the corresponding flow, $flow_id$ is a five-tuple, and $features_i$ are extracted from the raw packets belonging to the i -th subflow.

1) *Features*: Many features have been introduced to represent traffic behaviors of a flow/subflow [27]. However, to make sure FEC processes feature extraction in time, some features with high complexity, such as standard deviation, median, and quantiles, are excluded in our work. Table I lists all features we used in this paper, which are $prtl$, $dura$, $pkts$, $bytes$, pkt_size , $itvl$, tll , num_flg , is_new , $state$, and $state_pre$. There are 27 feature, in which, 5 features are floating-point number and the rest are an integer.

2) *Feature Extraction*: A flow table $\langle key, value \rangle$ is designed to obtain the value of each feature in Table I, where key is the five-tuple of each subflow, and $value$ is the features listed in Table I. Once the duration of a subflow reaches a

TABLE I: Features

Feature	Type	Num.	Description
$prtl$	I	1	Protocol, TCP is 6 and UDP is 17.
$dura$	F	1	Flow duration.
$pkts$	I	2	Total number of packets sent from the client to the server (forwards) or vice versa (backwards).
$bytes$	I	2	Total number of bytes sent from the client to the server (forwards) or vice versa (backwards).
pkt_size	I	4	Minimum and maximum packet size in two directions.
$itvl$	F	4	Minimum and maximum interval between two adjacent packets in two directions.
tll	I	4	Minimum and maximum ttl in two directions.
num_flg	I	6	The number of packets in the subflow with URG, ACK, PSH, RST, SYN and FIN flag.
is_new	I	1	Whether it is the first time the subflow is being recorded.
$state$	I	1	The current subflow state.
$state_pre$	I	1	The previous subflow state.
Total		27	

Type- I: Integer, F: Floating-Point Number

predetermined time interval Δ_{itvl} (e.g., $\Delta_{itvl} = 10s$), a sample represented by the features will be generated from the subflow and fed into latter components. For instance, the value of the feature ($pkts$) can be obtained according to the following equation,

$$pkts_{s \leftarrow c} = \frac{num_{c \leftarrow s}}{\Delta_{itvl}} \quad (2)$$

$$pkts_{c \rightarrow s} = \frac{num_{c \rightarrow s}}{\Delta_{itvl}} \quad (3)$$

, in which, $c \leftarrow s$ means a server sends packets to a client, $num_{c \leftarrow s}$ is the number of packets sent from the server to the client in the time interval Δ_{itvl} .

More details of the feature extraction process can be seen in Procedure 1.

B. Online Detection Component

ODC infers the samples (subflows) generated by Procedure 1 and identifies them as normal samples or attack samples on the basis of their reconstruction errors generated by the AE model.

1) *Autoencoder*: Traditionally, autoencoder (AE) is used for dimension reduction or feature learning [17]. However, this paper uses it to profile normal traffic and detect anomaly samples. Fig. 2 shows an example of the classical AE architecture. The encoder part maps an input sample $x \in R^d$ (d -dimensional space) to a latent representation $z \in R^t$ (t -dimensional space). In general, $d > t$ using nonlinear mapping $z = f(Wx + B)$, where f is an activation function (e.g., $Relu$), W is the weights matrix, and B is the bias vector. The decoder part reconstructs the input from the latent representation. The output of AE, $x' \in R^d$, is the "reconstructed" input, $x' = \phi(W'z + B')$, where ϕ is an activation function. By

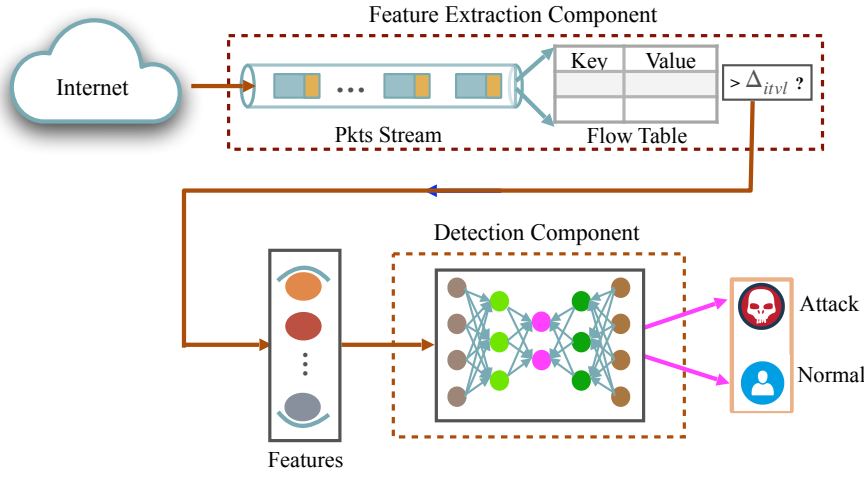


Fig. 1: DDoS detection framework.

```

Function extract_feature (packets,  $\Delta_{itvl} = 10s$ ):
    X = Queue()
    while packet in packets do
        key, value  $\leftarrow$  packet # value will be accumulated
            and updated.
        # key: five-tuple.
        # value: 27 features updated by flow table.
        if key not in flow_table.keys() then
            flow_table[key]  $\leftarrow$  value
        end
        else
            if flow_table.interval >  $\Delta_{itvl}$  then
                X += flow_table[key] # add the sample
                    (made of the features) to Queue().
                flow_table[key]  $\leftarrow$  0
            end
            else
                flow_table[key]  $\leftarrow$  value # update the
                    value of each feature.
            end
        end
    end
    return X
End Function

```

Procedure 1: Feature extraction from raw packets stream.

optimizing the parameters of AE (e.g., W , W' , B , and B'), the "error" between input x and reconstructed input x' is minimized, in which, the "error" is usually referred to as "reconstruction error" (δ), which is given by:

$$RE(x, x') = \|x' - x\|_2 \quad (4)$$

The training phase of AE is shown in Procedure 2.

2) *Online Detection*: The AE model is trained by using latest historical normal samples. Once training is done, the AE model can be used to detect anomaly samples. The online detection algorithm is shown in Procedure 3. FEC

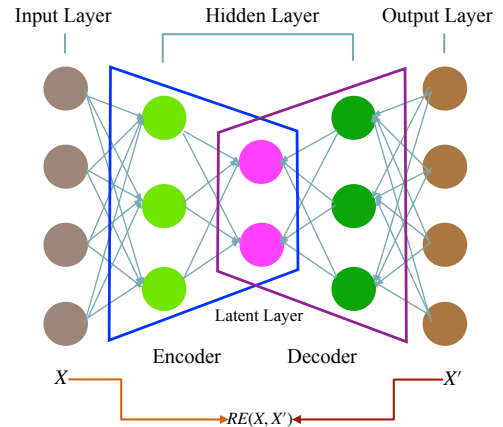


Fig. 2: Autoencoder architecture.

```

Function train_autoencoder (samples):
    X  $\leftarrow$  samples
    AE_model  $\leftarrow$  random values # Initialization.
    batch_size  $\leftarrow$  32
    epochs  $\leftarrow$  100 # the training times.
    while epochs > 0 do
        batches  $\leftarrow$  (X, batch_size)
         $\delta_{avg} = 0$  # average reconstruction error.
        for X_batch in batches do
            Z  $\leftarrow$  AE_model.encoder(X_batch)
            X'_batch  $\leftarrow$  AE_model.decoder(Z)
            loss  $\leftarrow$  RE(X'_batch, X_batch)
             $\delta_{avg} += \text{mean}(\text{loss})$ 
        end
         $\delta_{avg} \leftarrow \frac{\text{loss}}{\text{len}(\text{batches})}$  # len(batches) is the number
            of batch.
        epochs -= 1
    end
    return AE_model,  $\delta_{avg}$ 
End Function

```

Procedure 2: Autoencoder's training phase

constantly extracts features from raw packets and generate samples (subflows), then these samples are forwarded to the AE model. The model outputs the reconstructed input value X' , and the reconstruction error δ is calculated according to Eq.(4). Once the *error* δ exceeds the preset threshold

$$\Delta_{re} = c * \delta_{avg}, \quad (5)$$

where c is a constant and δ_{avg} is the average of reconstruction error of normal samples when the model converges in the training phase. An alarm would be raised and the corresponding flow would be classified as attack traffic. Packets belong to this flow would be dropped.

```

Function detect_anomaly(samples, AE_model):
     $\Delta_{re} \leftarrow 1.0$  # the threshold for RE, e.g.,  $1.0 = 7 * 0.15$ ,  $c = 7$ .
    while  $x$  in samples do
         $x' \leftarrow AE\_model(x)$ 
         $\delta \leftarrow RE(x', x)$ 
        if  $\delta \leq \Delta_{re}$  then
            | pass
        else
            alarm("An attack happened")
            drop(packets $x$ ) # drop all packets that have
                           the same 5 tuple with  $x$ .
        end
    end
End Function

```

Procedure 3: Online detection phase.

IV. EVALUATION

In this section, we evaluate the proposed framework (AE-D3F) and compare it with a classical supervised machine learning approach: decision tree (DT), and three classical anomaly detection approaches: principal component analysis (PCA), isolation forest (IF) and one class support vector machine (OC-SVM) on three datasets - one synthetic dataset and two public datasets

A. Dataset

The synthetic dataset (SYNT) is generated by a commercial simulation tool, which supports more than 38, 000 attacks and malwares, such as DDoS attacks, exploits, malware, and so on. Two public datasets, UNB 2017 [28] and MAWI [29], are downloaded from public repositories, which are intended to allow the readers to be confident on the datasets used for evaluation - that is fair and representative of contemporary Internet traffic. All the raw packets are stored in .pcap files. We replay the .pcap files and obtain the samples made of the features by FEC from raw packets in the experimental environment described in Section IV-A. Tab. II shows the details of all samples. The synthetic dataset (SYNT) includes 77, 988 normal samples and 74, 242 attack samples, in which, the attack samples include 36, 000 attack samples of DDoS "excessive get post attack", 37, 000 attack samples of DDoS "recursive get attack", 1, 000 DDoS "slowLoris attack", and

TABLE II: Dataset details of experiment I

Data source	Normal	Attack
SYNT	77, 988	74, 242
UNB	59, 832	35, 953
MAWI	62, 000	0

242 attack samples of DDoS "slow post attack". UNB includes 59, 832 normal samples and 35, 953 attack samples obtained by FEC. Four types of DDoS attack are included in the UNB dataset, i.e., 6, 414 attack samples of "slow HTTP attack", 11, 530 attack samples of DoS "hulk attack", 16, 741 attack samples of "slowloris attack", and 1, 268 attack samples of "golden eye". MAWI only includes 62, 000 normal samples.

B. Parameter and Architecture

Recall that one of our objectives is to detect real-time DDoS attacks. Hence, the AE model of AE-D3F should be simple and effective. The neural network of the AE model consists of one input layer, three hidden layers, and one output layer. The number of neurons for each layer is 27-24-16-24-27, respectively. The activation function we used is leaky relu. Adam optimizer is used for optimization, and mean squared error (MSE) is used as the loss function. Batch size is set to 32. In addition, the preset threshold of the reconstruction error is calculated according to the Eq. (5) in Section III-B. We set $c = 5$ for the following experiments and δ_{avg} is obtained from the training phase of each experiment.

C. Experiments

Each dataset is split into train set, validation set, and test set, using a 70-10-20 ratio. Train set is used to train models, and hyper parameters are tuned based on validation set. Finally, the models are evaluated on test set.

1) *Experiment I:* In this experiment, we demonstrate that normal traffic patterns are stable within a certain network environment. AE-D3F is able to profile normal traffic behaviors, and thus able to detect new and unknown attacks.

We train AE-D3F on three mentioned datasets and the data details are shown in Tab. III. Normal samples of each dataset is split into train set, validation set, and test set according to the ratio mentioned above. Note that AE-D3F is trained solely based on normal samples. Thus, all attack samples are included in test set. For example, for SYNT, train set includes 54, 592 (77, 988 * 70%) normal samples; validation set includes 7, 799 (77, 988 * 10%) normal samples; Test set includes the last 15, 598 (77, 988 * 20%) normal samples and 73, 242 attack samples.

The results in Tab. IV show that AE-D3F can achieve 98.32% DR on SYNT with only 0.38% FPR. For UNB, it can achieve more than 94.00% DR with a slightly higher FPR (1.88%). For MAWI, FPR is still low, only 0.84%. DR is not available, due to no attack samples included in the test set.

Fig. 3 shows the reconstruction errors generated by AE-D3F on the normal and attack samples of UNB test set (note that we resample the same number of normal and attack samples

TABLE III: Dataset details of experiment I

Data source	Train set (70% normal)	Validation set (10% normal)	Test set (20% normal + 100% attack)
SYNT	54, 592	7, 799	15, 598 + 74, 242
UNB	41, 881	5, 984	11, 967 + 35, 953
MAWI	43, 400	6, 200	12, 400 + 0

TABLE IV: Results of experiment I

Test set	Metrics	
	DR	FPR
SYNT	98.32%	0.38%
UNB	94.10%	1.88%
MAWI	-	0.84%

to draw the figure.). More than 98% reconstruction errors of normal samples is very small, less than 1.00. However, for attack samples, 94% reconstruction errors are more than 1.00, larger than that of normal samples.

Experiment I's results reveal that normal traffic patterns are stable within a network environment (the reconstruction errors of normal samples is very small.), and AE-D3F is able to separate attack samples from normal samples. The results also demonstrate the effectiveness of AE-D3F, which can achieve more than 94.00% DR with less than 2% FPR.

2) *Experiment II*: In this experiment, we show that the knowledge acquired from one network environment cannot be applied to another network environment. Hence, each network environment needs to build its own detector, while it would be more feasible for unsupervised approaches, which solely require normal samples.

We train AE-D3F, DT, PCA, IF, and OC-SVM models based on UNB dataset, then test them on UNB (partial), SYNT and MAWI datasets, respectively (other train set and test set combinations show the similar results. Due to the space limit, we only show the experiment results of this combination.). Since DT (one of supervised machine learning algorithms) requires normal samples and attack samples together to build its detection model, we split the attack samples of UNB dataset according to the default split ratio. Tab. V shows the details about the datasets used in this experiment. Note that unsupervised learning approaches (AE-D3F, PCA, IF, and OC-SVM) are all trained solely based on normal samples. For DT, we do data sampling for the train set to balance attack and normal samples, i.e., randomly sampling 25, 166 normal samples from the train set to make normal samples is equal to attack samples in the train set.

Table VI shows the results of experimental II. On UNB test set, PCA and OC-SVM achieve best DR (100%). However, their FPR are more than 9.50%. AE-D3F achieves a better FPR (below 2.00%), with a slightly lower DR (94.14%). DT achieves the lowest FPR (0.38%) of all. IF achieves the worst result, below 50% DR. The results show that all approaches except IF can work well within a certain network environment (more than 94% DR), by being provided

sufficient train samples. However, when applying to different network environments (i.e., SYNT and MAWI Test set), none of the detection models can achieve good performance. These detection models classified most of samples (normal and attack samples) as attack samples, since the traffic patterns in SYNT and MAWI are different from that in UNB. Thus, they achieve both high DR and FPR (note that DR is unavailable for MAWI test set, since it does not include attack samples). IF is an exception. it achieves a low FPR (6.40%) in MAWI test set. However, it performs worst in UNB test set in terms of DR. One possible reason is that IF does not learn useful knowledge from UNB train set, so the FPRs achieved by IF on all test sets are not credible.

In general, the results indicate that traffic patterns in different network environments vary significantly. Thus, to make the detection models work, training samples should be collected from the network being protected. With sufficient training samples, DT achieves the lowest FPR (0.38%) on UNB. However, except normal samples, training a DT also requires attack samples, which might not be always available. In contrast, AE-D3F is able to achieve comparable performance (exceeds 94% DR and belows 2% FRP) solely based on normal samples.

3) *Experiment III*: In this experiment, we show that DT, one of supervised machine learning approaches, cannot effectively detect new attacks which do not appear in its train set. In contrast, AE-D3F performs consistently well on new and unknown attacks.

We train DT and AE-D3F based on the SYNT dataset and the details of the datasets used for this experiment are shown in Tab. VII. The normal samples and attack samples of SYNT dataset are split into three sets according the default split ratio. To test unknown attacks, we introduce another test set (test set 2) including all attacks samples (35, 953) of UNB and the same normal samples as test set 1 (15, 598: 77, 988 * 20%). Note that the attack samples of test set 2 are not included in train set. For DT, we also do data sampling for train set to balance attack and normal samples as we did in experiment II. Note that AE-D3F still only uses normal samples in train set and validation set during the training phase.

The evaluation results shown in Tab. VIII indicate that DT performs well on known attacks, nearly 100% DR. However, it is unable to effectively detect new and unknown attacks (only 1.60% DR on test set 2) which do not appear in the train set. In contrast, AE-D3F can achieve nearly 100% DR on both known and unknown attacks test sets, with less than 0.5% FPR. In conclusion, AE-D3F is a promising solution for DDoS attack detection.

D. Feature Reduction

Generally speaking, extracting more features for AE-D3F achieves higher detection accuracy, but it also makes the system more complex and thus increases the time for training and inference. In some cases, extracting and analyzing more features could even have a negative impact on the model's performance because features may interfere with each other.

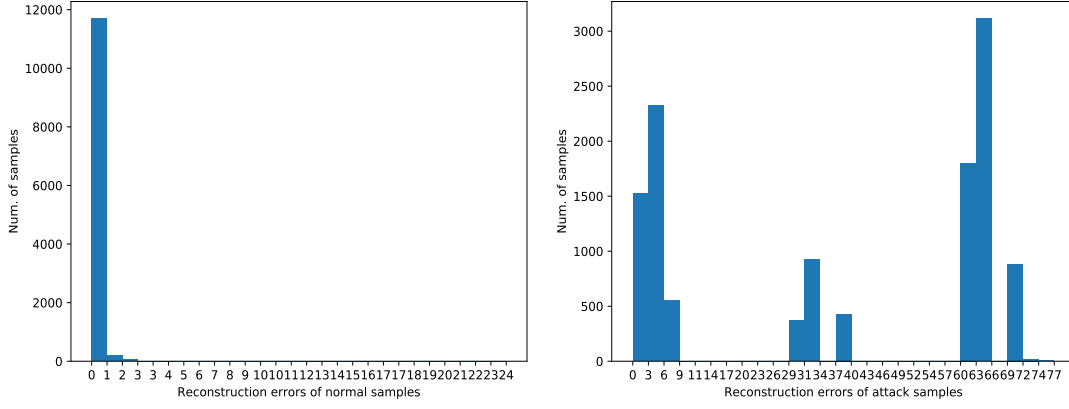


Fig. 3: Reconstruction errors of normal and attack samples on UNB test set.

TABLE V: Dataset details of experiment II

Data source	Train set (70%) (normal + attack)	Validation set (10%) (normal + attack)	Test set (normal + attack)		
			UNB Test set (20%)	SYNT Test set	MAWI Test set
UNB	41, 881+25, 166	5, 984 +3, 596	11, 967 + 7, 191	77, 988 + 74, 242	62, 000 + 0

TABLE VI: Results of experiment II

Model	Test Set (normal + attack)					
	UNB Test set		SYNT Test set		MAWI Test set	
	DR	FPR	DR	FPR	DR	FPR
AE	94.14%	1.91%	100%	100%	-	100%
DT	99.83%	0.38%	99.27%	80.18%	-	87.00%
PCA	100%	9.61%	100%	96.85%	-	100%
IF	49.06%	9.62%	99.38%	65.94%	-	6.40%
OC-SVM	100%	9.48%	100%	96.80%	-	100%

TABLE VII: Dataset details of experiment III

Data source	Train set (70%)	Validation set (10%)	Test set (normal + attack)	
			Test set 1 (normal + known attacks)	Test set 2 (normal + unknown attacks)
SYNT	54, 591 + 51, 968	7, 799 + 7, 425	15, 598 + 14, 849	15, 598 + 35, 953

TABLE VIII: Results of experiment III

Model	Test set 1		Test set 2	
	DR	FPR	DR	FPR
DT	99.96%	0%	1.60%	0%
AE	100%	0.49%	99.99%	0.49%

In order to reduce system complexity, this section aims to reduce the features as many as possible without decreasing DR significantly. Instead of combining some features into new or latent features, our objective is to select a subset of the original 27 features. Although the former may reduce feature dimension as PCA does, it does not reduce the feature extraction complexity. After comparing different feature selection algorithms [30], we choose pearson's correlation coefficient (PCC) [31], one of classical feature selection algorithms.

It calculates the correlation coefficient between every two features using equation (6) and decides which features should be grouped together based on the value of the coefficients. The higher correlation coefficient, the higher linear relation between two features, i.e., the higher probability one feature can be represented by the other. Hence, only one of them needs to be included in the subset. The correlation coefficient between two features is given by

$$\rho_{x,y} = \frac{Cov(X,Y)}{\sigma_X * \sigma_Y}, \quad (6)$$

where $Cov(X,Y)$ is the covariance between X feature and Y feature. σ_X is the standard deviation of X , and σ_Y is the standard deviation of Y .

The smaller correlation threshold is set, the more features will be removed from the original 27 features. Fig. 4 shows

the relation between FPR/FNR and the number of features on SYNT test set. When the number of features is equal to 16 (the corresponding correlation threshold is set to 0.95), FPR and FNR are both low at the same time. The corresponding features are *prtl*, *dura*, *pkts* (2 features including forwards and backwards packets), *bytes* (1 feature only including forwards bytes), *pkt_size* (4 features including min and max packets in forwards and backwards), *itvl* (4 features including the interval in forwards and backwards), *tll* (1 feature only including the minimal *tll* in forwards), *ack*, *state*.

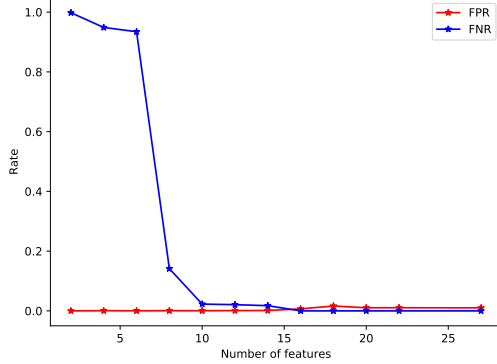


Fig. 4: Relation between the number of selected features and the detection performance on SYNT test set.

TABLE IX: Results of experiment I on different features

Test set	Metrics (27 features)		Metrics (16 features)	
	DR	FPR	DR	FPR
SYNT	98.32%	0.32%	99.99%	0.71%
UNB	94.10%	1.88%	93.51%	1.04%
MAWI	-	0.84%	-	1.03%

Table IX shows the results of AE-D3F with all features (27) and the selected features (16) on the datasets used in experiment I. The results indicate that AE-D3F achieves comparable performance while using fewer features.

E. Reconstruction Error Threshold

Reconstruction error threshold is an important parameter for AE-D3F. The threshold determines how sensitive we wish to detect outliers. Fig. 5 shows the relation between the model's performance and reconstruction error thresholds on UNB test set. There is a trade-off between FPR and FNR. The larger threshold is set, the more attack samples would be misclassified as normal samples. In other words, our model would become less sensitive and be more tolerant to attack samples, and thus FPR would decrease and FNR would increase. In contrast, the smaller threshold results in a higher FPR and a lower FNR. It is impossible to select a fixed threshold for all security objectives. A good threshold should be chosen based on the requirement of the specific security objective. To achieve low false alarm rate, we are more concerned with FPR than with FNR. Thus, a slightly larger threshold is preferred

(such as 0.95), rather than using the intersection point in the figure (0.6). When the threshold is set to 2.5, AE-D3F is able to reduce FPR to 0 and still achieve 82% DR.

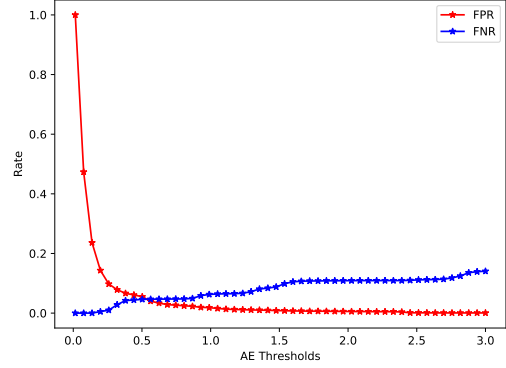


Fig. 5: Relation between the model's performance and reconstruction error thresholds on UNB test set.

V. CONCLUSION AND FUTURE WORKS

In this paper, we propose an AE-based DDoS detection framework (AE-D3F), which trains the AE model using only normal traffic from a protected network, and then uses the model to detect any attack traffic differing from the normal traffic. Experiments across different datasets (synthetic and public datasets) show that it can reduce FPR to 0 and still obtain 82% DR. Furthermore, it can detect new and unknown DDoS attacks in which supervised machine learning algorithms fail. Although this paper mainly focuses on DDoS attack detection, we believe that this AE-based framework can be extended for other scenarios of anomaly detection, such as datacenter network system failure detection.

According to the evaluation results, the proposed framework is promising. There are a few concerns should be further investigated. 1) To make our framework more robust, our model needs to be updated with time goes. Networks are highly dynamic systems: concept drift can occur at any time when a new host connects or disconnects to the network. It would be interesting to see if our framework still works with continuous update. 2) With continuous update, does our framework survive under poisoning attacks [32]? In addition, 3) it also will be interesting to investigate if our framework can be deployed in DDoS open threat signaling environments [33], in which each smart agent will have its own detection model to detect anomaly traffic. Our future works will focus on addressing the above concerns.

VI. ACKNOWLEDGMENTS

We would like to thank Fortinet for sponsoring the project, Sarjak pankaj Thakkar for conducting part of experiments, and Kartik Aggarwal for his help on extracting features from raw packets. We also would like to thank anonymous reviewers whose comments helped us improve the paper.

REFERENCES

- [1] Dhruba Kumar Bhattacharyya and Jugal Kumar Kalita. *DDoS attacks: evolution, detection, prevention, reaction, and tolerance*. Chapman and Hall/CRC, 2016.
- [2] Yuan Cao, Yuan Gao, Rongjun Tan, Qingbang Han, and Zhuotao Liu. Understanding internet ddos mitigation from academic and industrial perspectives. *IEEE Access*, 6:66641–66648, 2018.
- [3] Akami. DDoS report. <https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/state-of-the-internet-security-ddos-and-application-attacks-2019.pdf>, 2019. Accessed: 2019-01-20.
- [4] Matthijs Vos. Characterizing infrastructure of ddos attacks based on ddosdb fingerprints, July 2019.
- [5] kaspersky. DDoS report. https://usa.kaspersky.com/about/press-releases/2018_ddos-breach-costs-rise-to-over-2m-for-enterprises-finds-kaspersky-lab-report, 2019. Accessed: 2019-01-20.
- [6] Mousa Taghizadeh Manavi. Defense mechanisms against distributed denial of service attacks : A survey. *Computers & Electrical Engineering*, 72:26 – 38, 2018.
- [7] Liu Liu, Olivier De Vel, Qing-Long Han, Jun Zhang, and Yang Xiang. Detecting and preventing cyber insider threats: a survey. *IEEE Communications Surveys & Tutorials*, 20(2):1397–1417, 2018.
- [8] Nour Moustafa, Jiankun Hu, and Jill Slay. A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, 128:33–55, 2019.
- [9] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16, 2018.
- [10] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, pages 1–43, 2018.
- [11] Nazrul Hoque, Hiral Kashyap, and DK Bhattacharyya. Real-time ddos attack detection using fpga. *Computer Communications*, 110:48–58, 2017.
- [12] Rohan Doshi, Noah Aporthe, and Nick Feamster. Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018*, pages 29–35, 2018.
- [13] Amit Praseed and P Santhi Thilagam. Ddos attacks at the application layer: Challenges and research perspectives for safeguarding web applications. *IEEE Communications Surveys & Tutorials*, 2018.
- [14] Dusan Stevanovic and Natalija Vlajic. Next generation application-layer ddos defences: applying the concepts of outlier detection in data streams with concept drift. In *2014 13th International Conference on Machine Learning and Applications*, pages 456–462. IEEE, 2014.
- [15] Sunny Behal and Krishan Kumar. Characterization and comparison of ddos attack tools and traffic generators: A review. *I. J. Network Security*, 19(3):383–393, 2017.
- [16] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.
- [17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [18] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.
- [19] Randy Paffenroth, Kathleen Kay, and Les Servi. Robust pca for anomaly detection in cyber networks. *arXiv preprint arXiv:1801.01571*, 2018.
- [20] Majd Latah and Levent Tokar. An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks. *arXiv preprint arXiv:1806.03875*, 2018.
- [21] Baskoro Adi Pratomo, Pete Burnap, and George Theodorakopoulos. Unsupervised approach for detecting low rate attacks on network traffic with autoencoder. In *2018 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2018, Glasgow, Scotland, United Kingdom, June 11-12, 2018*, pages 1–8, 2018.
- [22] Pooria Madani and Natalija Vlajic. Robustness of deep autoencoder in intrusion detection under adversarial contamination. In *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security*, page 1. ACM, 2018.
- [23] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 90–98. SIAM, 2017.
- [24] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, 2018.
- [25] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiotnetwork-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.
- [26] Intel. DDPK. <https://www.dpdn.org/>, 2019. Accessed: 2019-01-20.
- [27] Andrew Moore, Denis Zuev, and Michael Crogan. Discriminators for use in flow-based classification. Technical report, ”, 2013.
- [28] UNB. IDS 2017 dataset. <https://www.unb.ca/cic/datasets/ids-2017.html>, 2019. Accessed: 2019-02-30.
- [29] MAWI. MAWI dataset. <http://mawi.wide.ad.jp/mawi/samplepoint-F/2019/>, 2019. Accessed: 2019-02-30.
- [30] Pradnya Kumbhar and Manisha Mali. A survey on feature selection techniques and classification algorithms for efficient text classification. *International Journal of Science and Research*, 5(5):1267–1275, 2016.
- [31] Alexander Ly, Maarten Marsman, and Eric-Jan Wagenmakers. Analytic posteriors for pearson’s correlation coefficient. *Statistica Neerlandica*, 72(1):4–13, 2018.
- [32] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.
- [33] Tirumaleswar Reddy.K, Mohamed Boucadair, Prashanth Patil, Andrew Mortensen, and Nik Teague. Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification. Internet-Draft draft-ietf-dots-signal-channel-39, Internet Engineering Task Force, November 2019. Work in Progress.