2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)

# Adversarial Attack against LSTM-based DDoS Intrusion Detection System

Weiqing Huang*†, Xiao Peng*†, Zhixin Shi*‡, Yuru Ma*†
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{huangweiqing, pengxiao, shizhixin, mayuru}@iie.ac.cn, ‡corresponding author

***Abstract*—Nowadays, machine learning is a popular method for DDoS detection. However, machine learning algorithms are very vulnerable under the attacks of adversarial samples. Up to now, multiple methods of generating adversarial samples have been proposed. However, they cannot be applied to LSTM-based DDoS detection directly because of the discrete property and the utility requirement of its input samples. In this paper, we propose two methods to generate DDoS adversarial samples, named Genetic Attack (GA) and Probability Weighted Packet Saliency Attack (PWPSA) respectively. Both methods modify original input sample by inserting or replacing partial packets. In GA, we evolve a set of modified samples with genetic algorithm and find the evasive variant from it. In PWPSA, we modify original sample iteratively and use the position saliency as well as the packet score to determine insertion or replacement order at each step. Experimental results on CICIDS2017 dataset show that both methods can bypass DDoS detectors with high success rate.**

***Index Terms*—Adversarial samples, genetic algorithm, probability weighted, LSTM, DDoS detector**

## I. Introduction

Distributed Denial of Service (DDoS) is a kind of attacks in which an attacker seeks to make a victim machine or network resources unavailable to its intended users. Since it poses serious threat to network security, many technologies have been applied to detect DDoS traffic so far. Recently, with the development of machine learning technology, Long Short-Term Memory (LSTM) models have received a considerable attention in DDoS detection task. For example, recent studies [8, 11, 13, 14] all adopt LSTM model to distinguish DDoS from benign traffic and achieve over 96% accuracy. Radford et al. [18] use a bidirectional-LSTM model to detect traffic anomaly and achieve 84% AUC.

However, many machine learning models are vulnerable to adversarial samples: inputs that are specially crafted to cause a machine learning model to produce an incorrect output [5, 15]. Many works aim to fool machine learning models by crafting adversarial samples, which mainly focus on image recognition, attributed graphs, malware classification and so on. Nevertheless, for the DDoS intrusion detection task, this research topic is still at the initial stage.

This paper tries to validate the security of an LSTM-based DDoS detection model when it is attacked by adversarial samples. We assume this adversarial attack is in a black-box setting, where the attacker is only allowed to query the target detector and does not know the details of the learned model. Therefore, when generating DDoS adversarial samples, the attacker can only manipulate input samples by testing and observing the target model's outputs.

However, it is more challenging to generate DDoS adversarial sample due to its discrete property. The input sample in our attack refers to a sequence which comprises a series of packets. Thus, it is hard to optimize the distance between adversarial sample and the original sample while guaranteeing the correctness of each packet. In general, existing works designed for images cannot be applied to our attack directly.

Drawing ideas from generating adversarial texts [1, 19], we aim to generate high-quality DDoS adversarial sample by replacing or inserting partial packets. Then, we propose two attack methods: Genetic Attack (GA) and Probability Weighted Packet Saliency Attack (PWPSA). In GA, we formalize generating DDoS adversarial sample to a combinatorial optimization problem and use genetic algorithm to solve it. In this algorithm, GA continually updates a population which consists of many variants of the original sample, and then extracts DDoS adversarial samples from the population. In PWPSA, we break modifying the original sample down into small, single-operation steps. In each step, PWPSA first uses a scoring function to find the important position in packet sequence, and then finds a best packet to insert or replace at this position. Our experimental results show that both methods can yield high success rates.

The major contributions of this work are as follows:

- We give the very first study of a kind of DDoS adversarial attacks. A threat model for LSTM-based DDoS detector is introduced.
- We propose GA and PWPSA to generate DDoS adversarial samples. Both methods can effectively generate adversarial inputs by replacing or inserting partial packets under the black-box settings.
- We conduct a case study on a group of target detectors. Experimental results verify that our methods can fool the target detectors with high success rates. We also recognize that GA achieves fewer modifications while PWPSA needs less computational resources.

The paper is organized as follows: Section II reviews the related research. Then, we introduce the DDoS adversarial attack in Section III. Section IV shows the experimental evaluation of our methods. The conclusion is in Section V.

2375-0197/20/$31.00 ©2020 IEEE
DOI 10.1109/ICTAI50040.2020.00110

## II. Related Work

### A. LSTM-based DDoS Detection

Recently, LSTM models are widely used in DDoS detection. Kim et al. [8] and Le et al. [10] all use LSTM classifier as intrusion detection system and achieve good experimental results. In these works, the input sample is a vector which is composed of 41 flow features that are provided by KDDCup99 dataset. In 2017, Yuan et al. [26] leverage different deep learning models, which contain Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), to detect DDoS attack. They extract 20 features to represent a packet, and then establish input samples by transformation and reshaping. Meanwhile, they compare the performance of four different RNN models and experimental results that LSTM model shows higher accuracy. Radford et al. [18] use an LSTM model to detect malicious activity from network traffic and achieve 84% AUC. In this work, the input sample is an ordered sequence of flows per IP-pair which are formed with a rolling window. Li et al. [13] also use RNN models (including LSTM model) to classify. When establishing input sample, they use the features that provided in KDDCup99 dataset and use PCA algorithm to reduce the dimensions of the features. Li et al. [14] also use LSTM as a classifier with combining with other machine learning model to detect DDoS. In this work, they extract 10 features from the traffic to form input sample.

### B. Adversarial attack LSTM model

Existing works of adversarial attack LSTM models mainly focus on natural language processing (NLP) and speech recognition. There are two kinds of attack: white-box setting and black-box setting. In white-box setting, the attackers have complete knowledge of the target model. Some methods which are proposed to mislead feed-forward neural networks, such as FGSM [16], JSMA [16], C&W [23], can also be adapted to attack LSTM models. Additionally, researchers proposed other adversarial attack methods according to the property of input samples. For example, Ebrahimi et al. [3] and Gong et al. [4] propose a gradient-based optimization method to generate adversarial texts respectively, which can decrease the accuracy of LSTM model greatly. However, the black-box setting is closer to realistic scenarios. In recent work, there have been a few attempts [24, 27]. Kuleshov et al. [9] propose a greedy optimization strategy for finding adversarial samples. It is an iterative procedure that considers at each step all valid one-word changes to a sequence. Besides, Li et al. [12] propose TEXTBUGGER to generate adversarial texts, in which they first find the important sentences and then use a scoring function to find important words to manipulate. On this basis, Ren et al. [20] propose a greedy algorithm called PWWS, in which the manipulation order determined by both the position saliency and the word Importance. In addition, Alzantot et al. [1] propose an attack method that based on genetic algorithm. Although they are applied to NLP and limited to semantic characteristics, these works offer thoughts for our GA and PWPSA methods.

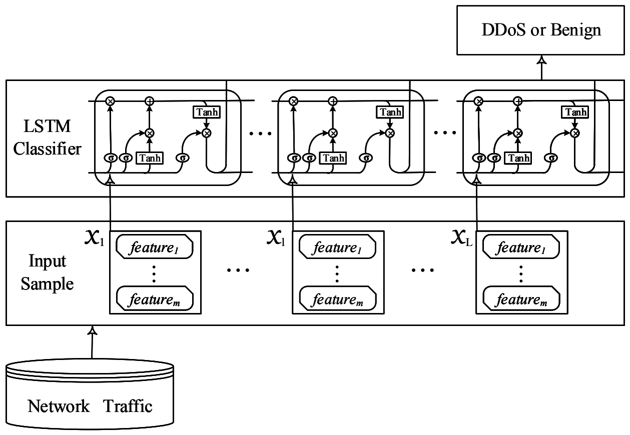


Fig. 1. An LSTM-based DDoS Intrusion Detection System

### C. DDoS Adversarial Samples

There are a few literatures on adversarial samples targeted at deep learning based DDoS detection. Ibitoye et al. [7] and Warzynski et al. [25] analyze adversarial attacks against deep learning based IDS. However, they use FGSM, BIM, PGD to generate adversarial samples, which belong to white-box attacks. [17] presents a black-box attack and utilize an optimization method to generate DDoS adversarial samples. However, since the input sample is not a kind of sequence, the method cannot be applied to LSTM-based detector. [6] is perhaps most closely related to our work. They perform delaying, splitting and injecting packets until the input sample can evade the target detector. However, it is hard to perform multiple operation to the original sample in practice.

## III. Attack Design

### A. Threat Model & Initial Formulation

*1) Target Detector:* Throughout this paper, a DDoS detector refers to an external system which adopts LSTM-based classification technology. As shown in Fig.1, to distinguish DDoS attack from benign activity, the detector traces back multiple network packets. After feature extraction, each packet is represented as a specially-designed vector. Packets within a time window form a sequence, which is taken as an input sample of the LSTM classifier.

*2) Threat Model:* We assume that an attacker attempts to launch a desired DDoS attack and aims to fool the DDoS detector. It is a black-box attack. The structure and parameters of the target detector are not revealed to the attacker, as well as the training data that are used. However, we assume the attacker can obtain the length of input samples, as well as the classification score of each input sample. The classification score is a predicted number which indicates the probability of maliciousness. Additionally, we assume the attacker is capable of manipulating network traffic and can replay network traffic to the target DDoS detector. Under this threat model, the strategy of the attacker is to craft DDoS variants iteratively

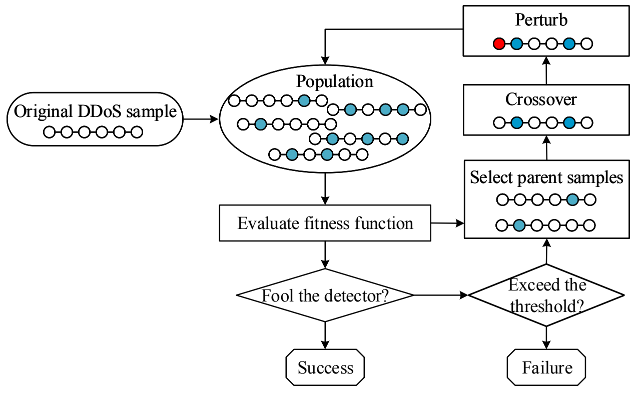


Fig. 2. The Illustration of Genetic Attack

until find the optimal variants. In a word, the core of conducting DDoS adversarial attack is to generate DDoS adversarial samples efficiently.

*3) Initial Formulation:* Let $X = (x_1, \dots, x_L)$ denote an input sample, where $L$ is the sample length. Let $F(X) : X \in \mathcal{X} \rightarrow Y \in \mathcal{Y}$ denotes the target detector. Specially,$Y_{adv}$ indicates that a DDoS sample is classified as benign. Besides, let $\mathrm{P}(Y_{adv}|X)$ represents the classification score of an input sample. Given an original DDoS Sample $X_0$, the generating DDoS adversarial sample can be expressed as: $\widetilde{X} = \arg\min_{X^*} \mathrm{d}(X_0, X^*)\ s.t.\ F(X^*) == Y_{adv}$, where $\mathrm{d}$ represents the distance between two samples.

## B. Genetic Attack

In this subsection, we apply genetic algorithm to generate high-quality DDoS adversarial samples. It is used to solve such a combinatorial optimization problem: (a). minimize the number of replaced/inserted packets; (b). keep the replaced/inserted packet close to original packet. The solution is illustrated in Fig.2. It begins with a population, which is a set of variants of the original sample. After selecting more than one parent variant with certain probability, crossover and perturb operators are carried out to update the whole population. Additionally, a fitness function is used to evaluate each variant in population and judge whether the DDoS adversarial sample is found or not.

*1) Perturb Operation:* It refers to performing a replace/insert operation for an input sample, which can be denoted as $\mathrm{Perturb}(X) : X \rightarrow X_l^*$, where

$$X = (x_1, \dots, x_l, \dots, x_L),$$

$$X_l^* = (x_1, \dots, x_l^*, \dots, x_L)\ or\ X_l^* = (x_1, \dots, x_l^*, x_l, \dots, x_L).$$

There are two requirements for Perturb Operation: (a). the replaced/inserted packet should be close to original packet; (b). the perturbed sample oughts to increase the classification score. Therefore, the function $\mathrm{Perturb}(X)$ contains the following steps:

- Select a position $l$ randomly, and repeat this step until the packet $x_l$ in this position hasn't been replaced/inserted;
- Obtain $N$ nearest neighbors of the selected packet $x_l$, and find the best replacement/insertion $x_l^*$ which $\mathrm{P}(Y_{adv}|X_l^*)$ reaches a maximum;
- Replace/Insert with $x_l^*$ at position $l$, and return the perturbed sample.

There are several details when implementing the second step. Firstly, although $L_p$ norms with $p = 0, 2, \infty$ are often used as the distance metrics, we employ the Mahalanobis distance to quantify the similarity between two packets. This is because the extracted features of packet have different range of values and there is a correlation between different features. Besides, Since the number of packets in the database is huge, we suggest that the attacker do a random sampling and then calculate the Top$N$ nearest packets. Next, the attacker can determine the best replacement/insertion by querying for the target detector $N$ times.

*2) Crossover Operation:* It refers to generate a child sample from a population, which can be denoted as $\mathrm{Crossover}(POP) : POP \rightarrow X_{child}^*$, where $POP$ represents a population. It contains two steps:

- Select two samples $X_{parent1}^*$ and $X_{parent2}^*$ with a certain probability vector $Select\_Prob$;
- Merge $X_{parent1}^*$ and $X_{parent2}^*$ into one sample $X_{child}^*$, and the probability for each token in $X_{child}^*$ coming from one parent sample is 50%.

In the first step, $Select\_Prob = (P_1, \dots, P_s, \dots, P_S)$, where $S$ is the size of population and $P_s$ denotes the probability of selecting sample $X_s^*$ from the population. $P_s$ is defined as:

$$P_s = \frac{\mathrm{P}(Y_{adv}|X_s^*)}{\sum_{X^* \in POP} \mathrm{P}(Y_{adv}|X^*)}. \qquad (1)$$

*3) Optimization Procedure:* Now we combine these genetic operators into an iterative algorithm, as described in Alg.1. In each iteration, the population will be evaluated and updated. By querying for the target detector, we can extract the best variant from a population which has the highest classification score. Then, if its predicted label is equal to the target label $Y_{adv}$, terminate the iteration. If not, reserve the best variant and perform *Crossover* and *Perturb* operation to update the population. As mentioned above, classification scores represent the fitness scores of each variant, which will direct the update. What's more, in order to guarantee the diversity of the population, each child has different parents when performing *Crossover* operation in practice. In short, the optimization procedure is to evolve the population and the DDoS adversarial sample is derived from the population.

## C. Probability Weighted Packet Saliency Attack

We propose another iterative method to generate DDoS adversarial sample, which is illustrated in Fig.3. In each iteration, unlike genetic attack, this method modifies current

**Algorithm 1** Genetic Attack

```
Require: The target detector $F(X)$, population size $S$, the number of iterations $T$, original DDoS sample $X_0$.
Ensure: DDoS adversarial sample $\widetilde{X}$.
1: Initialize the population $POP_0 \leftarrow \emptyset$.
2: for $s = 1, 2, \ldots S$ do
3:     $POP_0 \leftarrow POP_0 \cup \{\text{Perturb}(X_0)\}$
4: end for
5: for $t = 0, 1, \ldots T$ do
6:     $\widetilde{X}_t \leftarrow \arg\min_{X^* \in POP_t} \text{P}(Y_{adv}|X^*)$
7:     if $F(\widetilde{X}_t) == Y_{adv}$ then
8:         return $\widetilde{X}_t$
9:     else
10:        $POP_{t+1} \leftarrow \{\widetilde{X}_t\}$
11:        for $s = 1, 2, \ldots S - 1$ do
12:            $X^*_{child} = \text{Crossover}(POP_t)$
13:            $POP_{t+1} \leftarrow POP_{t+1} \cup \{\text{Perturb}(X^*_{child})\}$
14:        end for
15:    end if
16: end for
```

input sample at a certain position with a specific packet. In order to determine this position and this packet, we first estimate the saliency of each position. And then we select the best replacement/insertion packet for each position. After determining the priority of position for replacement/insertion, we can modify a sample in an exact manner. The key operations are introduced below.

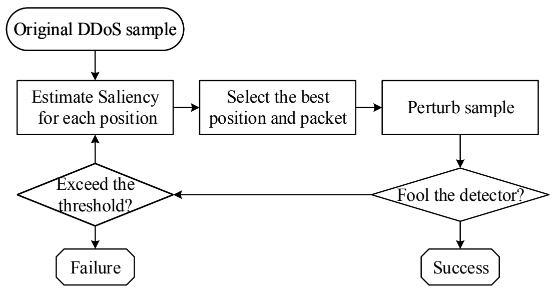


Fig. 3. The Illustration of Probability Weighted Packet Saliency Attack

*1) Estimate Saliency Operation:* The Saliency of a position represents how a position $l$ impacts the output of an input sample $X$, which is denoted as $\text{Saliency}(X, l)$. The value can be estimated by:

$$S_l = \text{P}(Y_{adv}|\widehat{X}_l) - \text{P}(Y_{adv}|X), \quad (2)$$

where

$$X = (x_1, \ldots, x_l, \ldots, x_L),$$

$$\widehat{X}_l = (x_1, \ldots, \Delta, \ldots, x_L).$$

In this equation, the packet at position $l$ is set to unknown, which is denoted as $\Delta$. In practice, $\Delta$ can be a packet which does not exist in the database.

*2) Perturb Operation:* It refers to performing a replace/insert operation for an input sample, which is same with the introduction in Section.III-B1 except that the position to modify is given. Let $\text{Perturb}(X, l)$ denote the operation. Then, the result is defined as:

$$X^*_l = \arg\max_{X'_l} \text{P}(Y_{adv}|X'_l), \quad (3)$$

where

$$X'_l = (x_1, \ldots, x'_l, \ldots, x_L) \ or \ X'_l = (x_1, \ldots, x'_l, x_l, \ldots, x_L)$$

and $x'_l$ is close to $x_l$.

*3) Determine Priority Operation:* This operation provides a basis for selecting the best position and packet for replacement/insertion. To determine the replacement/insertion order, we consider the saliency of each position as well as the perturb result for each position. Let $\text{Priority}(X, l)$ denote the priority of position $l$ for input sample $X$. It is calculated by:

$$\text{Priority}(X, l) = \frac{S_l}{\sum_{i=1}^{L} S_i} \cdot \text{P}(Y_{adv}|X^*_l), \quad (4)$$

where $S_l = \text{Saliency}(X, l)$ and $X^*_l = \text{Perturb}(X, l)$.

*4) Optimization Procedure:* The final Algorithm is shown in Alg.2. Given an input sample, this algorithm only extracts one position to perturb in each iteration. To achieve this, this algorithm first performs $Saliency$ and $Perturb$ operation to calculate the $Priority$ for each position. Then, perturbs the sample according to the value of $Priority$. Finally, repeat this process until a DDoS adversarial sample is found.

**Algorithm 2** Probability Weighted Packet Saliency Attack

```
Require: The target detector $F(X)$, the number of iterations $T$, original DDoS sample $X_0$, sample length $L$.
Ensure: DDoS adversarial sample $\widetilde{X}$.
1: for $t = 0, 1, \ldots T$ do
2:     for $l = 1, 2, \ldots L$ do
3:         $S_l \leftarrow \text{Saliency}(X_t, l)$
4:     end for
5:     for $l = 1, 2, \ldots L$ do
6:         $X^l_t \leftarrow \text{Perturb}(X_t, l)$
7:         $W_l \leftarrow \text{Priority}(X_t, l)$
8:     end for
9:     $BestPosition_t \leftarrow \arg\max_{l \in [1,L]} W_l$
10:    $X_{t+1} \leftarrow \text{Perturb}(X_t, BestPosition_t)$
11:    if $F(X_{t+1}) == Y_{adv}$ then
12:        return $X_{t+1}$
13:    end if
14: end for
```

## IV. Experiments

### A. Dataset and Setup

To evaluate the effectiveness of the proposed methods, we use CICIDS2017 dataset [22]. Specifically, we extract DDoS traffic from the $Wednesday$ data and regard the $Monday$ data as benign traffic. Referring to [26], we extract 8 features to represent a packet which are listed in Tab.I. In this case, we obtain 3166007 different packets. Then, we separate the data into training set and testing set with ratio 8:2. In addition, because most traffic are benign, we randomly choose benign samples to match the number of DDoS samples when building training set in each experiment.

TABLE I
FEATURES THAT ARE EXTRACTED FROM PACKET

| Feathure | Example | Type |
|---|---|---|
| frame.len | 74 | Integer Numeric |
| frame.protocols | eth:ethertype:ip:tcp | Enumerated Text |
| tcp.ack | 1 | Integer Numeric |
| tcp.analysis.ack_rtt | 0.211084 | Floating-point Numeric |
| tcp.analysis.bytes_in_flight | 0 | Integer Numeric |
| tcp.len | 231 | Integer Numeric |
| tcp.srcport | 53058 | Enumerated Numeric |
| tcp.window_size | 29200 | Integer Numeric |

We establish 5 different LSTM models as target detectors which is outlined in Tab.II. These models have different hidden sizes and different keep probabilities for dropout regularization. However, the sample length $L$ for each model is set to 20, which indicates that all detectors trace back 20 packets for DDoS detection. As shown in Tab.II, the accuracy of each target detector is up to 93.97% and each average confidence is about 99.70%. We emphasize that detector **E** has the highest accuracy while detector **D** is the lowest one.

TABLE II
THE TARGET DETECTOR

| Detector Name | A | B | C | D | E |
|---|---|---|---|---|---|
| **Hidden Size** | 64 | 128 | 256 | 128 | 128 |
| **Keep-Prob** | 0.85 | 0.85 | 0.85 | 0.65 | 1.00 |
| **Accuracy** | 95.39% | 95.80% | 96.04% | 93.97% | 97.21% |
| **AC** | 99.81% | 99.72% | 99.76% | 99.70% | 99.96% |

When implementing adversarial attacks, we set $T$=100, $S$=100, $N$=50 in all experiments. It is worth noting that the original DDoS samples are from testing set and are correctly classified by target detector. For simplicity, we abbreviate the method of generating DDoS adversarial samples by replacing packets with Genetic Attack algorithm as ”GA-Replace”. And the remaining three methods are shorted as ”GA-Insert”, ”PWPSA-Replace”, ”PWPSA-Insert” respectively.

## *B. Results Analysis*

We use four metrics to measure the effectiveness of adversarial attacks:

**Success Rate:** The percentage of generated adversarial samples which are misclassified under the condition that the modification percentage is less than 30%.

**Average Modification Percentage (AMP):**. The average ratio of replaced or inserted packets to all packets for adversarial samples.

**Queries:** The average number of accessing target detector when generating adversarial samples.

**Average Confidence of Adversarial Class (ACAC):** The average prediction confidence towards the misclassified class.

The results are reported in Tab.III–Tab.V. Since the ACACs are all greater than 80%, we did not list them for the limited space. The results show that the success rates range from 67% to 99%. Additionally, we find the following phenomena:

- No matter using GA or PWPSA, replacing packets seems to contribute to a higher success rate and a less AMP than inserting packets.
- No matter replacing or inserting packets, GA algorithm usually leads to a higher success rate and a less AMP than PWPSA.
- The queries of PWPSA is far less than GA.

TABLE III
THE SUCCESS RATE FOR DIFFERENT ATTACK METHODS

| Detector / Attack | A | B | C | D | E |
|---|---|---|---|---|---|
| **GA-Replace** | 92.00% | 93.23% | 94.20% | 91.37% | 99.09% |
| **GA-Insert** | 72.50% | 88.24% | 91.80% | 74.50% | 99.90% |
| **PWPSA-Replace** | 88.89% | 92.86% | 91.37% | 88.90% | 95.35% |
| **PWPSA-Insert** | 69.96% | 77.46% | 76.94% | 67.17% | 93.85% |

TABLE IV
THE AMP FOR DIFFERENT ATTACK METHODS

| Detector / Attack | A | B | C | D | E |
|---|---|---|---|---|---|
| **GA-Replace** | 17.55% | 16.36% | 15.42% | 20.20% | 7.24% |
| **GA-Insert** | 22.63% | 18.26% | 16.85% | 22.88% | 6.89% |
| **PWPSA-Replace** | 18.99% | 17.76% | 16.15% | 19.59% | 10.92% |
| **PWPSA-Insert** | 25.07% | 23.65% | 22.77% | 26.68% | 11.03% |

TABLE V
THE QUIRIES FOR DIFFERENT ATTACK METHODS

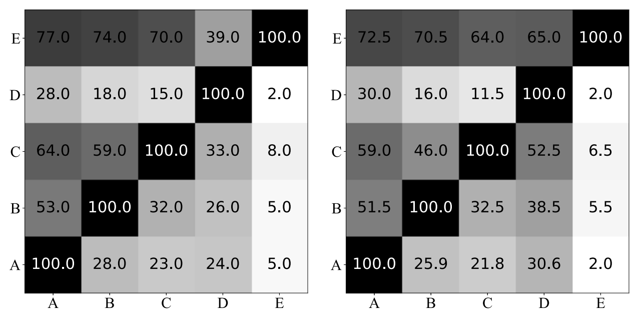
| Detector / Attack | A | B | C | D | E |
|---|---|---|---|---|---|
| **GA-Replace** | 22722 | 11523 | 10992 | 13546 | 6584 |
| **GA-Insert** | 14741 | 12372 | 11682 | 14679 | 5200 |
| **PWPSA-Replace** | 1619 | 1541 | 1670 | 1721 | 948 |
| **PWPSA-Insert** | 3382 | 3130 | 3014 | 3580 | 1520 |

Here is an explanation for the second phenomenon. In each iteration, PWPSA can exactly elect the best variant from $L \cdot N$ alternative samples. However, before GA exactly electing the best variant from $S$ alternative samples, it has $\binom{S}{2} \cdot 2^m \cdot L$ possibilities for each alternative sample. Here we use $m$ to represent the number of differences between two selected samples when a $Crossover$ operation is performed. In short, GA has a poorer stability but has a larger search space than PWPSA.

In addition, the query of GA is equal to $S \cdot N + t \cdot [S + (S-1) \cdot N]$ while the query of PWPSA is $L + t \cdot L \cdot N$, where $t$ refers to the number of iterations when the algorithm was terminated. It is worth stressing that, in the same condition, $t$ of GA is usually greater than $t$ of PWPSA. Actually, $t$ of PWPSA is no more than the sample length $L$. Yet, the decrease of $S$ is usually accompanied by the increase of $t$. Their relationship will be discussed in the next subsection. All in all, the queries of GA are usually greater than that of PWPSA.

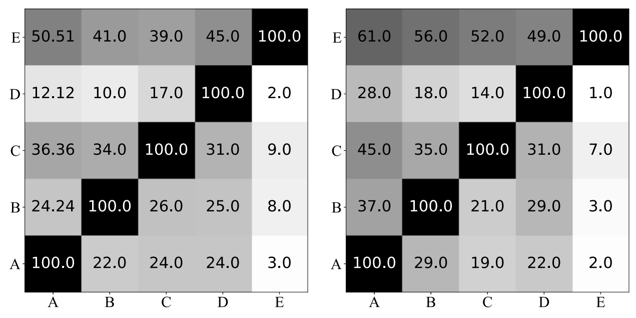
## C. Transferability Analysis

The transferability refers to the ability of adversarial samples which are produced to mislead a specific model to mislead other models [5, 15]. Fig.4 and Fig.5 depict the transferability for GA and PWPSA respectively. In each confusion matrix, cell(**x**,**y**) indicates the percentage of DDoS adversarial samples which are generated on target detector **x** and are misclassified as benign by detector **y**. The four confusion matrices are similar on the whole. We can see that detector **E** and **C** are more susceptible to adversarial attacks despite having high accuracy. Meanwhile, the most robust detector is **D**, closely followed by detector **A**. It is worth noting that the **D** and **A** have the fewest hidden size or the fewest keep probabilities for dropout regularization. This indicates that the generalization ability of the model has a significant influence on the effectiveness of adversarial attack.



(a) GA-Replace (b) GA-Insert

Fig. 4. The Transferability for Genetic Attack



(a) PWPSA-Replace (b) PWPSA-Insert

Fig. 5. The Transferability for Probability Weighted Packet Saliency Attack

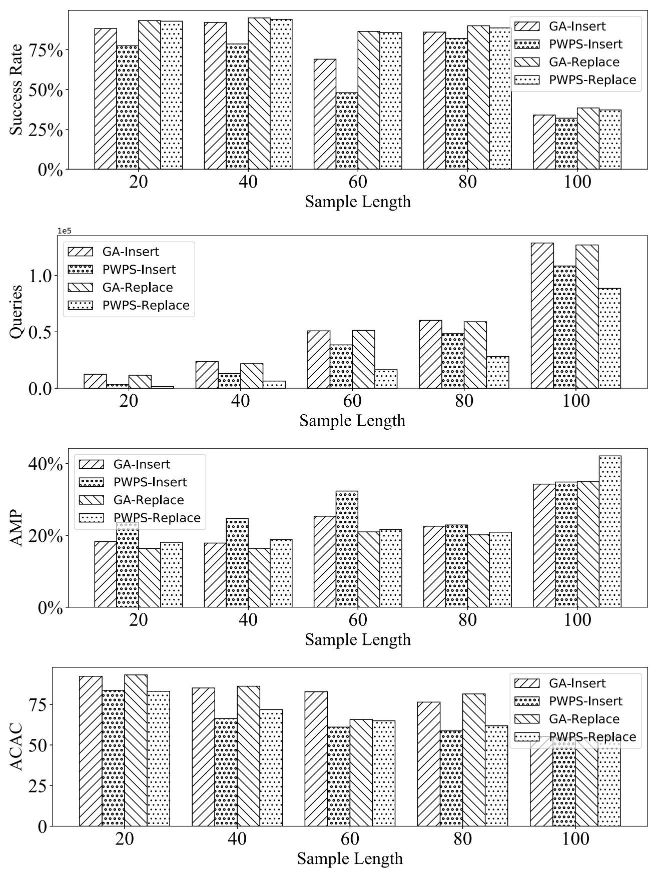


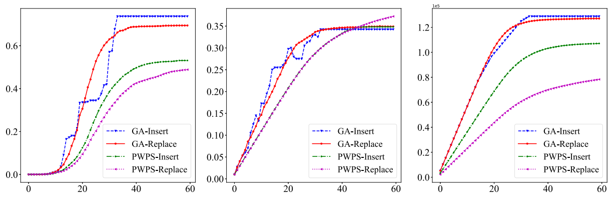
Fig. 6. Impact of Sample Length $L$

## D. Parameters Evaluation

In this subsection, we analyze how the parameters impact the performance of adversarial attacks.

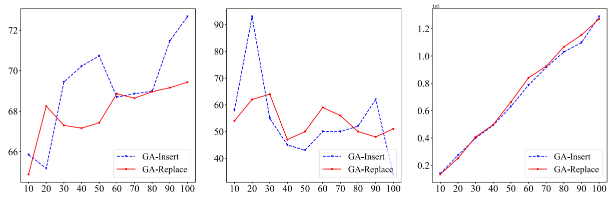
*1) Impact of Sample Length:* We vary the length of input sample length. Fig.6 presents the evaluation results. There is no doubt that the queries monotonically increase with the increase of sample length. Meanwhile, it is shown that the sample length has a negative impact on the results of success rate and AMP. It demonstrates that a larger number of captured packets not only reflect more attack activities but also make an LSTM-based DDoS detector more robust. However, as mentioned in [26], researchers seldom traced back more than 100 packets for DDoS detection.

*2) Impact of Iterations:* To reflect the impact of iterations adequately, we set the sample length $L$ to 100 in this experiment. The experimental results are shown in Fig.7. We can see that the ACAC of GA grows faster than PWPSA. In other words, GA achieves faster convergence. However, we stress that results like this the relies on a reasonable population size. It is also worth noting that the AMP of PWPSA increases with the iteration definitely monotonically. And yet, in terms of GA, the AMP fluctuates while increasing. It demonstrates that there are more probabilities for GA to obtain the optimal



(a) Variations of ACAC in increasing $T$ gradually (b) Variations of AMP in increasing $T$ gradually (c) Variations of Quries in increasing $T$ gradually

Fig. 7. Impact of Iterations $T$



(a) Variations of ACAC in increasing $S$ gradually (b) Variations of Iterations in increasing $S$ gradually (c) Variations of Quries in increasing $S$ gradually

Fig. 8. Impact of Population Size $S$

samples.

*3) Impact of Population Size:* Population size $S$ is an important parameter for GA. In this experiment, $L$ is also set to 100. As shown in Fig.8, a large $S$ would increase the population diversity which contributes to a high ACAC of generated adversarial samples. However, as other studies have pointed out [2, 21], a large $S$ can be unhelpful on the genetic algorithm. We can see that the queries are ascending notably with the increase of $S$. Besides, although the increase of $S$ contributes to the decrease of $T$ on the whole, the negative correlation between them is not obvious in our case. That is, the use of a larger $S$ does not improve the effectiveness of adversarial attack and only increases the needed computational resources. Comprehensively taking all these factors into account, we suggest that $S$ is set to a number ranging from 5 to 30.

## V. Conclusion

In this paper, we study adversarial attack against LSTM-based DDoS detector under a black-box setting. In addition, we propose GA and PWPSA to generate DDoS adversarial samples. Our experimental results show that both methods are powerful and effective. Furthermore, we discuss the relationship between the effectiveness of adversarial attack and the generalization ability of the target detector. Our results can encourage building more robust LSTM-based DDoS detector in the future.

## References

[1] M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. Srivastava, and K. Chang, "Generating natural language adversarial examples," pp. 2890–2896, 2018.

[2] T. Chen, K. Tang, G. Chen, and X. Yao, "A large population size can be unhelpful in evolutionary algorithms," *Theoretical Computer Science*, vol. 436, pp. 54–70, 2012.

[3] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[4] Z. Gong, W. Wang, B. Li, D. Song, and W.-S. Ku, "Adversarial texts with gradient methods," *arXiv preprint arXiv:1801.07175*, 2018.

[5] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv: Machine Learning*, 2014.

[6] M. J. Hashemi, G. Cusack, and E. Keller, "Towards evaluation of nidss in adversarial setting," in *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019, pp. 14–21.

[7] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in iot networks," *arXiv preprint arXiv:1905.05137*, 2019.

[8] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," pp. 1–5, 2016.

[9] V. Kuleshov, S. Thakoor, T. Lau, and S. Ermon, "Adversarial examples for natural language classification problems," 2018.

[10] T. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," pp. 1–6, 2017.

[11] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of ddos attack–based on deep learning in openflow-based sdn," *International Journal of Communication Systems*, vol. 31, no. 5, 2018.

[12] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.

[13] Q. Li, L. Meng, Y. Zhang, and J. Yan, *DDoS Attacks Detection Using Machine Learning Algorithms*, 05 2019, pp. 205–216.

[14] Y. Li and Y. Lu, "Lstm-ba: Ddos detection approach combining lstm and bayes," pp. 180–185, 2019.

[15] N. Papernot, P. Mcdaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv: Cryptography and Security*, 2016.

[16] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, 2016, pp. 49–54.

[17] X. Peng, W. Huang, and Z. Shi, "Adversarial attack against dos intrusion detection: An improved boundary-

based method," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 1288–1295.

[18] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks." *arXiv: Computers and Society*, 2018.

[19] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, 2019.

[20] ——, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 1085–1097.

[21] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," pp. 371–376, 2013.

[22] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," pp. 108–116, 2018.

[23] M. Sun, F. Tang, J. Yi, F. Wang, and J. Zhou, "Identify susceptible locations in medical records via adversarial attacks on deep predictive models," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 793–801.

[24] W. Wang, B. Tang, R. Wang, L. Wang, and A. Ye, "A survey on adversarial attacks and defenses in text," *arXiv preprint arXiv:1902.07285*, 2019.

[25] A. Warzyński and G. Kołaczek, "Intrusion detection systems vulnerability on adversarial examples," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018, pp. 1–4.

[26] X. Yuan, C. Li, and X. Li, "Deepdefense: Identifying ddos attack via deep learning," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2017.

[27] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 3, pp. 1–41, 2020.