Received March 17, 2022, accepted April 14, 2022, date of publication April 22, 2022, date of current version April 29, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169616

LSTM-Based Collaborative Source-Side DDoS Attack Detection

SUNGWOONG YEOM, CHULWOONG CHOI, AND KYUNGBAEK KIM, (Member, IEEE)
Department of Artificial Intelligence Convergence, Chonnam National University, Gwangju 61186, South Korea

Corresponding author: Kyungbaek Kim (kyungbaekkim@jnu.ac.kr)

This work was supported by the Institute for Information & communications Technology Planning & Evaluation (IITP) and the Korea Government Ministry of Science and ICT (MSIT) under Grant 2019-0-01343, in part by the Regional strategic Industry convergence Security Core Talent Training Business, and in part by the Bio and Medical Technology Development Program of the National Research Foundation (NRF) through the Korean Government Ministry of Science and ICT (MSIT) under Grant NRF-2019M3E5D1A02067961.

ABSTRACT As denial of service attacks become more sophisticated, the source-side detection techniques are being studied to solve the limitations of target-side detection techniques such as delayed detection and difficulty in tracking attackers. Recently, some source-side detection techniques are being studied to use an adaptive attack detection threshold considering seasonal behavior of network traffic. However, because patterns of network traffic usage have become irregular with increased randomness and explosive traffic, the performance of the adaptive threshold technique has deteriorated. In addition, by limitations of the local view of a single site, distributed attacks from multiple sites may not be detected. In this paper, we propose a LSTM (Long Short Term Memory) based collaborative source-side DDoS (Distributed Denial of Service) attack detection framework which provides the attack detection result of a collaboration network in a global view. The proposed framework applies LSTM-based adaptive thresholds to each source-side network to mitigate performance degradation caused by irregular network traffic behavior. Also, in order to overcome the limitation of performance caused by the local view of single source-side network, the proposed framework constructs a collaborative network through multiple detection sites and aggregates feedback from each site, such as detection rates, local traffic patterns, and timestamp. The collaborative attack detection technique uses the aggregated feedback to determine whether the attack is finally detected and shares the finial detection results with multiple sites. Depending on this final detection result, the adaptive thresholds of each site are reset. Through extensive evaluation of actual network traffic data, the proposed collaborative source-side attack detection technique shows around 15% lower false positive rate than the single source-side attack detection technique while maintaining a high detection rate.

INDEX TERMS Network security, DDoS attack, SDN, LSTM, collaborative detection, traffic seasonality embedding.

I. INTRODUCTION

Because of the ubiquity and increasing popularity of IoT (Internet of Things), it contributes significantly to improving the quality of human life, from traditional equipment to general household goods. However, the threat of exposure to DDoS attacks that exploit vulnerabilities of IoT devices distributed in multiple regions is increasing rapidly [1]. A victim-side detection methods have disadvantages such as delay in detection and difficulty in tracking the attacker. To alleviate these disadvantages, a source-side detection method has been studied [2].

However, the volume of traffic observed in the source-side network is relatively small compared to the victim-side network, the normal traffic can be easily mixed with attack traffic. To separate a relatively small volume of attack traffic, an adaptive threshold method using the observed traffic volume was studied [3]. However, if the observed traffic is mixed with the attack traffic, the adaptive threshold method should separate the attack traffic from the observed traffic to calculate the next threshold. In order to detect the attack traffic, dynamic adaptive threshold methods based on observed traffic volumes are studied [4]. Recently, with the activation of time series deep learning neural networks, the short-term network traffic volume prediction methods based on time series deep learning neural networks are studied also [5]–[7].

In linear network traffic in which characteristics such as self-similarity [8], and seasonality [9] are observed, this method show high performance. However, because of unexpected behavior of network users, network traffic may have

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

high jitters with frequent explosive traffic and it show non-linear properties [10]. In non-linear network traffic where characteristics such as burstiness [11] and randomness [12] are observed, performance of these methods are degraded. To mitigate this degradation, we need to consider the relationship between traffic states such as distinguishing different network traffic in embedding spaces [13]–[16].

However, when a large DDoS attack is occurred on multiple sites, the source-side attack detection method in fixed single site may not detect the attack traffic by limited local view [17]. Also, the performance of source-side attack detection methods may be different depending on the network characteristics located on multiple regions [18], [19]. In order to improve the performance of source-side attack detection method, the collaborative attack detection methods which share attack detection result between source-side network of multiple regions are studied [20]–[23]. These methods calculate value that represent the performance of each site and these values are used to determine whether the attack is detected with the detection results of each site. However, because of irregular behavior patterns of network users if the detection results are aggregated from multiple source-side networks where non-linearity is observed, the performance of the collaborative attack detection methods may be degraded. Therefore, collaborative attack detection methods need to consider not only the attack detection results and performance of collaborative sites, but also the relationships between collaborative sites and time dependency of individual site characteristics.

In this paper, we propose an LSTM-based collaborative source-side DDoS attack detection framework. The proposed framework shares the detection results of source-side attack detection method which is located on multiple source-side networks. Through the statistics of the detection results which is shared in advance, the statistical weight of each source-side attack detection module is calculated by considering the probability of attack detection and the probability of false positive at the corresponding time index. The statistical weights can represent the performance of the source-side attack detection module, but cannot cope with false positives caused by irregular traffic patterns. The proposed framework aggregates the information such as the detection result, traffic change rate, and time index of each source-side DoS attack detection module located in different time zones. By using the aggregated information with the statistic weight of the source-side attack detection modules, the collaborative source-side attack detection module determines whether the attack is finally detected. The final detection result is shared to adjust the adaptive threshold of collaboration sites. In order to verify the effectiveness of the proposed framework, we evaluated the detection rate, false positive rate, and balanced accuracy of the proposed method according to the average traffic volume, jitters, and burst ratio based on actual DNS (Domain Name System) traffic data.

The rest of this paper is arranged as follows. Section 2 introduces the research progress of related techniques of adaptive threshold method for preventing DDoS attack by cooperating the source-side attack detection system. Section 3 describes the proposed collaborative source-side DoS attack detection system. Section 4 verifies the effectiveness of the algorithm though experiments. Section 5 summarizes conclusion and future research directions.

II. RELATED WORK

SDN can provide a more flexible, dynamic, manageable, and adaptive network [24]. SDN makes the network flexible by efficiently handling traffic congestion created by modern applications with dynamic behavior [25]. Because the network control logic can be programmed by SDN, the overall network enables easy configuration and quick optimization of network resources. In addition, the network administrators can dynamically adjust traffic flow to meet application demands. The dynamic network architecture provided by SDN makes it easier to detect and migrate DoS attacks [26]. An SDN based DoS defense mechanisms can be classified according to the location of deployment: victim-side defense mechanisms and source-side defense mechanisms.

The victim-side defense mechanism detects, filters, and limits malicious traffic at routers located close to the victim. Detecting DoS attacks on the victim-side is simple because the volume of malicious network traffic is extremely high and the attack traffic can be clearly distinguished from the observed traffic. However, the source-side defense mechanisms prevent generating attack traffic to a victim by identifying malicious packets that pass a gateway of a subnet where the attack sources reside [27]. The one of method of a source-side DoS attack detection mechanism is predicting the volume of normal network traffic. The volume of network traffic observed in the source-side network is relatively small, and more accurate prediction of network traffic is required in order to adjust threshold in a fine-tuned manner [3], [4].

In general, the statistical characteristics of traffic observed on the source-side network are time dependency, self-similarity, seasonality, non-linearity, randomness and burstiness. In the network where the linear trend is revealed, the statistical detection methods such as ARIMA(Auto-Regressive Integrated Moving Average) and exponential smoothing shows the high performance. Yaacob *et al.* used ARIMA method to detect potential attacks that may occur in the network [28]. Chan *et al.* used a neural network development approach based on an exponential smoothing method which aims at enhancing previously used neural networks for traffic flow forecasting [29]. In linear network traffic in which characteristics such as self-similarity, and seasonality are observed, these methods show high performance.

However, this approach does not consider the irregular usage patterns of network users such as non-linearity, randomness, and burstiness. Recently, there are few studies which uses LSTM to predict network traffic volume. Ramakrishnan *et al.* proposed RNN architectures as an approach towards solving the network volume prediction problems [5]. Lazaris *et al.* presented a network traffic prediction framework that uses real network traces to train LSTM and generate predictions at short time scales [7]. Geng *et al.* proposed LSTM and applied historical network traffic data to predict traffic flow [6]. For the stability of the prediction of LSTM by solving the long-term dependency gradient vanishing and exploding, a network traffic trend is distinguished from other network traffic trend. To solve this problem, the LSTM training model of network traffic trend can be efficiently customized by applying different embedding space [13]–[15].

The source-side attack detection mechanisms usually detect the traffic flow fluctuations in Internet routers or gateways near the sources, but have limitations in terms of detection accuracy and delay.

Moreover, this mechanism cannot capture the characteristics of large-scale distributed attacks. BH Song *et al.* proposed an effective DDoS defense system using a collaborative method between distributed IDRSs(Intrusion Detection and Response Systems) located around the attack sources or victim network [30]. However, because the scale of end-to-end service becomes larger, IDRS located on the source-side needs to consider computing resources for packet sampling. At this time, in order to reduce computing resources in IDRS, it is necessary to use the network traffic volume instead of packet sampling for detecting DDoS attacks. Yeom *et al.* proposed a collaborative source-side attack detection method to more accurately detect DDoS attack in multiple networks, taking into account the detection performance in different time zones [23]. This method detects DoS attacks by applying a margin to the adaptive threshold at each source-side, and shares the detection results and the weights which represent the performance of each source-side attack detection. Sharing the statistical weight with each source-side attack detection system can alleviate the side-effect of the adaptive threshold where the detection rate and the false positive rate are increased. However, when the traffic pattern is changed on the source-side, the optimal margin applied in adaptive threshold will also be changed. Then, the simulation for calculating statistical weight should be processed. Therefore, in collaborative attack detection method, it should be possible to consider the degree of traffic congestion on each source-side. If the collaborative source-side attack detection techniques consider not only the statistical weight but also the traffic pattern, then the performance can be improved. In this paper, we propose a LSTM-based collaborative source-side attack detection framework considering the performance and traffic patterns which are cooperative with source-side attack detection systems.

III. COLLABORATIVE SOURCE-SIDE ATTACK DETECTION FRAMEWORK

The source-side attack detection methods use the adaptive threshold to detect attack traffic mixed with a relatively small amount of traffic volume, where non-linearity features such as high jitter and high burst rates are observed in the source-side network, the adaptive thresholds may have low performance. In order to improve the performance of adaptive threshold, it is necessary to predict network traffic using time series deep learning techniques after attack detection. In addition, it is required to share the detection results of collaborators to overcome the limitation caused by the local view of single site. At this time, the collaborative attack detection module considers the relationship between source-side networks and the time dependency between the detection information aggregated from each sites. Figure 1 shows the proposed collaborative source-side attack detection framework that determines whether attack is detected or not by aggregating the detection information from collaborators. The proposed framework consists of four components as follows: source-side attack detection system, event handler module, trust management module and collaborative attack detection module.

A. SOURCE-SIDE ATTACK DETECTION SYSTEM

The proposed source-side attack detection system can be deployed on the gateway to detect DoS attack traffics which flow from subnet to the victim. This system captures network traffic from the gateway through samplers such as a DNS sampler and a NTP sampler by using SDN [4]. When the amount of captured traffic is rapidly increased by being mixed with attack traffic, the adaptive threshold can be used to effectively detect the attack. The proposed system consists of two modules as follows: the adaptive threshold module, the LSTM-based normal traffic prediction module

1) ADAPTIVE THRESHOLD MODULE

Because the volume of traffic observed in the source-side network is relatively small compared to the traffic observed in the victim-side network, the observed traffic volume during the unit time is very sensitive. The adaptive threshold is used to detect attack traffic mixed with normal traffic. In every time windows with constant size T_w, the network traffic is captured and the volume of the observed traffic in the z^{th} time window is defined as s_z. Whenever the network traffic is captured, the adaptive threshold θ_z is applied to s_z in order to determine whether the observed traffic contains malicious attack traffic or not. The detection threshold, θ_z, is set dynamically by adding margin, δ, to the volume of forecasted traffic, $\bar{s}_{z+1}$, as shown in the equation 1.

$$\theta_{z+1} = (1+\delta) * \bar{s}_{z+1} \tag{1}$$

This detection threshold, θ_{z+1}, is dynamically adjusted by using the volume of forecasted traffic, $\bar{s}_{z+1}$. The volume of forecasted traffic $\bar{s}_{z+1}$ is calculated to an exponential smoothing algorithm as shown in the equation 2.

$$\bar{s}_{z+1} = \alpha * s_z + (1-\alpha) * \bar{s}_z \tag{2}$$

However, when an attack is detected, it is necessary to separate the attack traffic from the observed traffic to calculate the next threshold. To separate attack traffic from observed traffic, the predicted volume of normal traffic in the z^{th} time

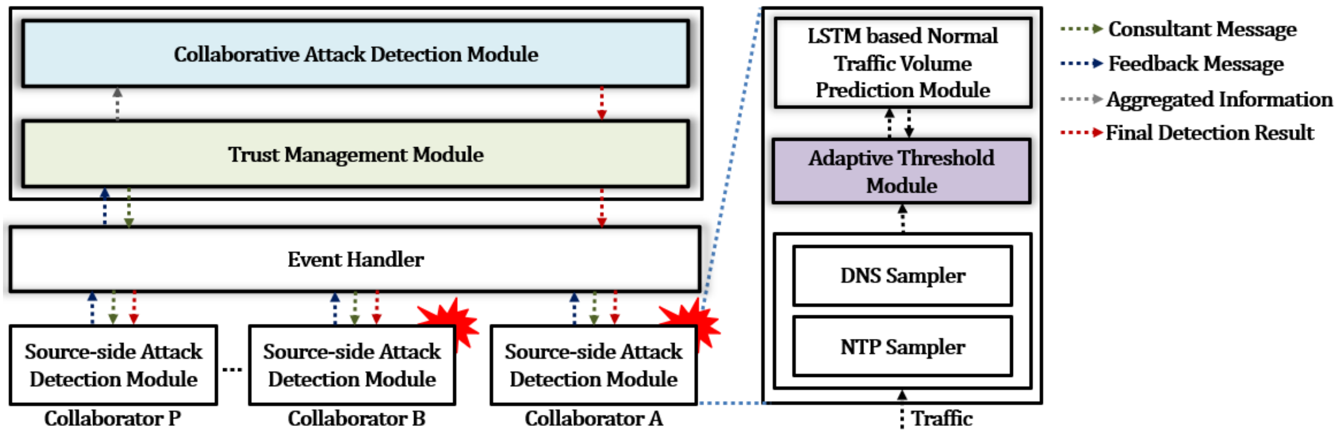

FIGURE 1. Collaborative source-side attack detection framework.

window, $s_{predict_z}$, is required as shown in equation 3.

$$\bar{s}_{z+1} = \alpha * s_{predict_z} + (1-\alpha) * \bar{s}_z \tag{3}$$

$s_{predict_z}$ is set by multiplying the volume of observed traffic in the previous $(z-1)^{th}$ time window, s_{z-1}, to the traffic changing rate in the z^{th} time window, ch_z, as shown in the equation 4.

$$s_{predict_z} = s_{z-1} * ch_{predict_z} \tag{4}$$

When the attack is detected by comparing the detection threshold θ_z with the observed traffic s_z, the predicted changing rate $ch_{predict_z}$ is transmitted with the changing rate of observed traffic ch_z and timestamp t_z from the LSTM-based normal traffic prediction module.

The proposed adaptive threshold θ_z is set on the source-side attack detection systems that is located on each side. And each of source-side attack detection system share the detection results SR_z with the collaborative source-side attack detection framework. The detection result SR_z is calculated as shown in the equation 5.

$$SR_z = \begin{cases} 1 & \text{if } s_z \geq \theta_z \\ 0 & \text{if } s_z < \theta_z \end{cases} \tag{5}$$

Later, this detection result SR_z becomes the component of the feedback messages with the changing rate of observed traffic ch_z and timestamp t_z after receiving the consultant message from the proposed trust management module of the collaborative source-side attack detection framework. When the final detection result FR_z comes out from the collaborative attack detection module of the proposed framework, it should be transmitted to the source-side attack detection systems to calculate the detection threshold θ_z. The value of final detection result FR_z will be 0 (normal) to 1 (attack). Finally, the adaptive threshold calculated by determining whether attack is detected or not in the source-side attack detection system is as shown in the equation 6.

$$\bar{s}_{z+1} = \alpha * ((1-FR_z) * s_z + FR_z * s_{predict_z}) + (1-\alpha) * \bar{s}_z \tag{6}$$

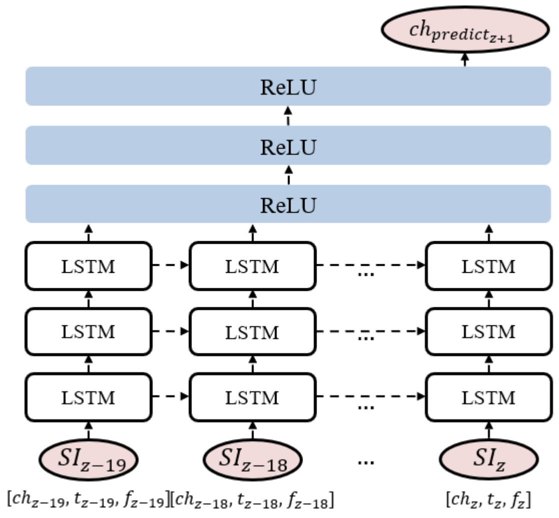

FIGURE 2. Architecture of LSTM-based traffic volume prediction model.

2) LSTM-BASED NETWORK TRAFFIC VOLUME PREDICTION MODEL

For improving the performance of adaptive threshold, the accurately estimating change rate of normal traffic ch_z in z^{th} time window is important. For this estimation, the proposed approach uses LSTM which is a time series neural network model. Figure 2 depicts the architecture of the proposed LSTM-based traffic volume prediction model. This model consists of three LSTM layers and three dense layers. Among these three dense layers, the first two layers use ReLU (Rectified Linear Unit) activation function and the last layer uses Linear activation function. Each layer includes 20 nodes.

For training and testing the LSTM-based traffic volume prediction model, it is necessary to manage observed traffic to the matrix. The matrix of observed traffic to collect the volume of observed traffic s_{ij} in the i^{th} time window on j^{th} day from the adaptive threshold module is represented as $\mathbf{S} = \begin{bmatrix} s_{11} & \cdots & s_{j1} \\ \vdots & \ddots & \vdots \\ s_{1i} & \cdots & s_{ji} \end{bmatrix}$, where $i \in [1, \frac{1440}{n}]$ at a given constant time interval n and $j \in [1, m]$ for m day. However, when observed traffic is mixed with attack traffic, the predicted volume of normal traffic $s_{predict_z}$ is managed instead of the observed traffic s_z.

With this matrix of observed traffic, the input vector for the LSTM model is generated in order to ensure the stability of the prediction. The input vector consists of three features including the changing rate of observed traffic ch_z, the time window index t_z, and the traffic trend v_z which are corresponding to the z^{th} time window. The input vector at the z^{th} time window represents as $SI_z = (ch_z, t_z, v_z)$.

The LSTM model trains with the input vectors to predict the changing rate of normal traffic. In detail, the LSTM model uses 20 continuous input vectors $SI_{z-19}, SI_{z-18}, \ldots, SI_z$ from $(z-19)^{th}$ to z^{th}, and this model provides output as the predicted changing rate at the $(z+1)^{th}$ time window, $ch_{predict_{z+1}}$.

The first domain of the input vector is the changing rate of observed traffic ch_z, which is the changing rate between two continuous the volume of observed traffic. Because the volume of observed traffic is managed in the matrix of observed traffic S, the changing rate of observed traffic is also managed in the matrix. The matrix of changing rate of observed traffic C which collects the changing rate of observed traffic ch_{ij} in the i^{th} time window on j^{th} day is represented as $\mathbf{C} = \begin{bmatrix} ch_{11} & \cdots & ch_{j1} \\ \vdots & \ddots & \vdots \\ ch_{1i} & \cdots & ch_{ji} \end{bmatrix}$. Accordingly, the changing rate of observed traffic in the z^{th} time window on the y^{th} day, ch_{yz}, can be calculated as the equation 7.

$$ch_{yz} = \frac{s_{yz}}{s_{yz-n}} - 1 \tag{7}$$

The second domain of the input vector is the time window index for the changing rate of observed traffic. The matrix of time window index T which collects the time window index t_{ij} in the i^{th} time window on j^{th} day is represented as $\mathbf{T} = \begin{bmatrix} t_{11} & \cdots & t_{j1} \\ \vdots & \ddots & \vdots \\ t_{1i} & \cdots & t_{ji} \end{bmatrix}$. The time interval n is set to five minutes equal to the size of time window T_w, and the length of time window on one day becomes 288. Accordingly, the time window index t_{yz} in the z^{th} time window on the y^{th} day has values between 1 and 288.

The third domain of the input vector is the traffic trend, which represents the characteristics of network traffic including seasonality, burstiness, and randomness. The traffic trend is categorized by considering various characteristics of network traffic. The categorized traffic trend v_z is embedded through one-hot encoding as shown in the equation 8, where o is the number of the traffic trend which can be categorized.

$$v_z = \{e_1, e_2, \ldots, e_o\}, \quad v \in \{0, 1\} \tag{8}$$

When the predicted change rate $ch_{predict_z}$ comes out from the proposed LSTM-based traffic prediction volume module, it is sent to the adaptive threshold module along with the change rate of observed traffic ch_z and timestamp t_z.

3) TRAFFIC SEASONALITY EMBEDDING

The network traffic trend is necessary to express the relationship between the seasonal patterns of traffic observed per unit time in order to clearly improve the performance of the LSTM model. This embedding method can be defined as a static embedding method and a dynamic embedding method.

The static embedding method categorizes the traffic trend into a given number of states in a static manner. That is, a day is divided into a given number of time zone by considering the seasonal pattern of network traffic. For example, in general, traffic trends increase during the morning when people start their activity, fluctuate from the afternoon to early evening when people are active, and decrease during the subsequent hours. Then, the network trend can be categorized into three distinguished trends such as increasing trend, fluctuating trend and decreasing trend. Once the trend category is determined by using the time index, the traffic trend corresponding to the z^{th} time window of a day is embedded with one-hot vector encoding as $v_z = \{e_1, e_2, e_3\}$, $v \in \{0, 1\}$. For example, {1,0,0} denotes the increasing trend, {0,1,0} denotes the fluctuating trend and {0,0,1} denotes the decreasing trend.

The dynamic embedding method categorizes the traffic trend into a given number of states in a dynamic manner by considering the traffic changing rate of a given time window. Figure 3 shows the state selection mechanism based on the traffic changing rate, and this mechanism calculates the changing rate between observed traffic volumes of z^{th} and $(z-n)^{th}$ time window. The length of embedding space depends on the granular expression of traffic trend, and it may affect to the performance of LSTM-based network traffic prediction. If n is bigger, the embedded traffic trend represents longer period of time. Through the degree of the changing rate, the state of traffic trend can be categorized into five states such as Further Increasing, Increasing, Fluctuation, Decreasing, and Further Decreasing. With this five states, we may use different embedding space such as $v_z = \{e_1, e_2\}$ (Increasing, Decreasing), $v_z = \{e_1, e_2, e_3\}$ (Increasing, Fluctuating, Decreasing), $v_z = \{e_1, e_2, e_3, e_4, e_5\}$ (Further Increasing, Increasing, Fluctuating, Decreasing, Further Decreasing). For two states embedding, the traffic trend is distinguished by checking whether the changing rate ch_z is positive or negative. For three states embedding, the traffic is used to categorize the traffic trend. If the changing rate ch_z is larger than τ the traffic trend is considered as Increasing state. If the changing rate ch_z smaller than τ the traffic trend is considered as Decreasing state. Otherwise, the traffic trend is considered as Fluctuation state. For five states embedding, two traffic trend lines, τ and 2τ, are considered to categorize the traffic trend.

When preparing an input vector, the traffic trend is embedded as an encoded hot-vector. If the traffic trend is categorized in three states, the length of the encoded hot-vector is three. In here, we may focus state transitions rather than the state itself. The state transition means that the state changes

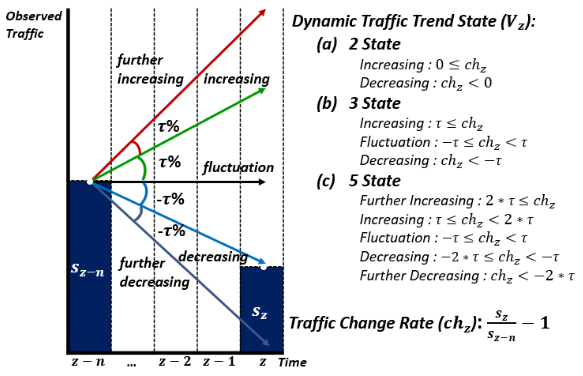
Observed Traffic
further increasing
increasing
τ%
τ%
fluctuation
-τ%
-τ%
further decreasing
decreasing
s_{z-n}
s_z
$z-n$ ... $z-2$ $z-1$ z Time
Dynamic Traffic Trend State (V_z):
(a) 2 State
Increasing : $0 \leq ch_z$
Decreasing : $ch_z < 0$
(b) 3 State
Increasing : $\tau \leq ch_z$
Fluctuation : $-\tau \leq ch_z < \tau$
Decreasing : $ch_z < -\tau$
(c) 5 State
Further Increasing : $2 * \tau \leq ch_z$
Increasing : $\tau \leq ch_z < 2 * \tau$
Fluctuation : $-\tau \leq ch_z < \tau$
Decreasing : $-2 * \tau \leq ch_z < -\tau$
Further Decreasing : $ch_z < -2 * \tau$
Traffic Change Rate (ch_z): $\frac{s_z}{s_{z-n}} - 1$

FIGURE 3. **Dynamic traffic state mechanism.**

between the previous time window and the current time window. If traffic trend has three states, trend state transition has nine states. Therefore, if there are two, three and five states, we can consider four, nine and twenty five states transitions, respectively. Through this traffic trend state embedding method, the overall performance of the LSTM-based source-side DoS detection method can be improved. However, the source-side attack detection system located in a single site lacks a global view for detecting attacks that are incoming from multiple sites simultaneously.

B. TRUST MANAGEMENT MODULE

In order to improve performance by detecting attacks coming from multiple sites and sharing detection results, the proposed source-side attack detection system is located in multiple sites. However, if the detection result of low-performance source-side attack detection systems are also aggregated, the performance of collaborative attack detection can be degraded. In order to mitigate the performance degradation of collaborative attack detection, it is necessary to manage the collaborators located in multiple sites. Before the collaborative source-side attack detection framework is activated, the proposed trust management module selects the collaborators according to the weight that is calculated through the simulation. In order to evaluate the performance of all source-side attack detection systems, the trust management module periodically sends test messages to each site and aggregates the detection results. According to the accumulated detection results of simulation by sending the test messages, the high-performance source-side attack detection system is registered in the list of collaborator. On the other side, the low-performance source-side attack detection systems are removed from the list of collaborators. The trust management module sends the consultant messages to collaborators which are recorded on the list. Then, the trust management module receives the feedback messages from each collaborators.

1) TEST MESSAGE

In order to verify the performance of source-side attack detection system of collaborators, the trust management module sends the test messages periodically to all of the sites. The test message is a "fake" consultant message for simulation and includes the attack ratio a^{ratio} estimated from the source-side attack detection systems. After receiving the result of test message, the collaborators execute the detection simulation and the false positive simulation. For the detection simulation in collaborators, the volume of observed traffic s_z which is managed from the matrix S applies the attack ratio a^{ratio} to generate the volume of attack traffic s_{test_z} in z^{th} time window for test as the following equation 9.

$$s_{test_z} = s_z * a^{ratio} \tag{9}$$

In the z^{th} time window, the detection simulation d_z is determined by comparing the detection threshold θ_z with the volume of attack traffic s_{test_z} as shown in the equation 10.

$$d_z = \begin{cases} 1 & \text{if } s_{test_z} \geq \theta_z \\ 0 & \text{if } s_{test_z} < \theta_z \end{cases} \tag{10}$$

The detection simulation d^l_{ij} from collaborator l at the i^{th} time window on j^{th} day is aggregated on the trust management module and managed in the detection matrix $\mathbf{D_l} = \begin{bmatrix} d^l_{11} & \cdots & d^l_{j1} \\ \vdots & \ddots & \vdots \\ d^l_{1i} & \cdots & d^l_{ji} \end{bmatrix}$. This detection matrices of collaborators are managed in the set of detection matrices, $D = \{D_A, D_B, \ldots, D_P\}$. Because of the traffic characteristics observed in the network of collaborators, the false positive can occur from source-side attack detection modules. It is necessary to simulate whether the false positive occur or not. In the z^{th} time window, the false positive simulation f_z is determined by comparing the threshold θ_z with the observed traffic s_z which is managed from matrix S as shown in the equation 11.

$$f_z = \begin{cases} 1 & \text{if } s_z \geq \theta_z \\ 0 & \text{if } s_z < \theta_z \end{cases} \tag{11}$$

The false positive simulation f^l_{ij} from collaborator l at the i^{th} time window on j^{th} day is also aggregated on the trust management module and managed in the matrix of false positive $\mathbf{F_l} = \begin{bmatrix} f^l_{11} & \cdots & f^l_{j1} \\ \vdots & \ddots & \vdots \\ f^l_{1i} & \cdots & f^l_{ji} \end{bmatrix}$. This false positive matrices of collaborators are also managed in the set of false positive matrices, $F = \{F_A, F_B, \ldots, F_P\}$. This simulation results are used to calculate the weight representing the performance of collaborators.

2) COLLABORATORS LIST MANAGEMENT

In order to improve the performance of collaborative detection, it is necessary to exclude a collaborator whose performance is relatively low by network characteristics from the list of collaborators. The performance of collaborators can be represented by weights calculated through statistics of simulation results. To calculate the statistical weight, the probability of detection $p^d_{z,k}$ and false positive $p^f_{z,k}$ of collaborator k in z^{th} time window during the last N days are represented as shown in the equation 12 and equation 13.

$$p^d_{z,k} = P(f^k_z = 1) = \sum_{j=1}^{N} \frac{d^k_{jz}}{N}, \quad k \in l \tag{12}$$

$$p^f_{z,k} = P(f^k_z = 1) = \sum_{j=1}^{N} \frac{f^k_{jz}}{N}, \quad k \in l \tag{13}$$

Then, the statistical weight w^k_z of collaborator k in z^{th} time window is estimated by adding the probability of detection $p^d_{z,k}$ and the probability of false positive $p^f_{z,k}$. The statistical weight w^k_z is represented as shown in the equation 14.

$$w^k_z = \beta * p^f_{z,k} + (1-\beta) * p^f_{z,k}, \quad k \in l \tag{14}$$

The coefficient β represents the specific importance of $p^d_{z,k}$ and $p^f_{z,k}$. The coefficient β has a constant value between 0 and 1. It means that the closer the value of coefficient β is to 1, the greater probability of detection $p^d_{z,k}$ from all of the source-side attack detection systems. We set the coefficient β to 0.5 to equalize the probability of detection and the probability of false positive.

3) CONSULTANT MESSAGE AND FEEDBACK MESSAGE

When the source-side attack detection systems detect the attack, the trust management module sends the consultant messages to the collaborators which are recorded in the list. This consultant message is not for simulation. When the attack occur in reality, the trust management module sends the consultant message to all of the collaborators.

After the collaborators receive the consultant messages, each of the source-side attack detection systems start to detect the attack following equation 5. When the attack detection of collaborators are finished, the trust management module receives the feedback message of collaborators for the determination of attack detection. The feedback message of collaborator k in the z^{th} time window represents as $FM^k_z = (SR^k_z, t^k_z, ch^k_z)$. The feedback message FM^k_z of the collaborator k in z^{th} time window consists of the detection result SR^k_z, the time window index t^k_z, and the changing rate of observed traffic volume ch^k_z from collaborators. However, if one of the collaborators fails to receive a feedback message by a network problem, the collaborative detection can be delayed.

C. EVENT HANDLER MODULE

For communication between the collaborative detection framework and the collaborators, the event handler module needs to monitor the end-to-end network link state. However, when links and devices errors are caused by unexpected traffic volumes, network communication cannot work smoothly. In order to reduce the recovering time about the failures of links and devices, the proposed event handler module uses the trustiness of links and switches derived from location trustiness [31]. The event handler module communicates with the received requests to the corresponding components of framework and routes requests and messages that are sent to the collaborators. These messages managed by the event handler module are as follows: test messages, consultant messages and feedback messages.

D. COLLABORATIVE ATTACK DETECTION MODULE

A collaborative attack detection module is needed to finally determine whether there is an attack using the aggregated feedback messages from the collaborators. However, if non-linearity such as high randomness and high burst rate are observed in the network of collaborators, the performance of collaborative attack detection methods that aggregate false positive results show a relatively low detection rate. In order to mitigate this performance degradation, we need to consider the relationship between collaborators and the time dependence between individual source-side attack detection results. There are two proposed collaborative detection method: Weight-based attack detection method and LSTM-based collaborative attack detection method.

1) WEIGHT-BASED COLLABORATIVE ATTACK DETECTION METHOD

In order to statistically represent the performance of the source-side DoS attack detection module, the proposed weight-based attack detection method needs the performance value that calculated the probability of detecting an attack and the probability of falsely detecting the attack at a specific time index. For the performance of the collaborators, the weight of the collaborators are calculated from the trust management module. The proposed weight-based attack detection method calculates the weighted arithmetic mean WA_z by using the detection result SR_z collected from the feedback aggregation module and the statistics weight w^k_z in z^{th} time window at k site. The weighted arithmetic mean WA_z formula in the weight-based attack detection method is shown as shown in the equation 15.

$$WA_z = \frac{\sum_{k \in l}^{N} w^k_z * SR^k_z}{\sum_{k \in l}^{N} w^k_z} \tag{15}$$

If this weighted arithmetic average WA_z is greater than the specified threshold th, the final detection result FR_z is determined to have been detected as shown in the equation 16.

$$FR_z = \begin{cases} 1 & \text{if } WA_z \geq th \\ 0 & \text{if } WA_z < th \end{cases} \tag{16}$$

However, the performance of weighted collaborative attack detection method is affected by the performance of the source-side attack detection system. That is, if false positives are observed frequently in the source-side attack detection system, the performance of collaborative detection method is also degraded. In order to improve the performance of collaborative attack detection, it is necessary to find patterns of

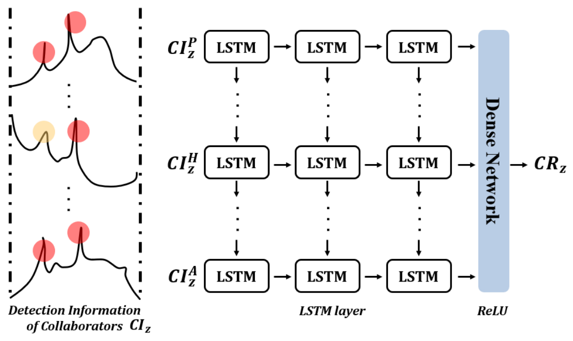

FIGURE 4. Architecture of LSTM-based collaborative attack detection model.

TABLE 1. Components of DoS attack detection methods.

Detection Method	Source-side Detection	Traffic Prediction	Collaborative Detection
LC_LBAT	Adaptive Threshold	LSTM	LSTM
LC_STBAT	Adaptive Threshold	Statistic	LSTM
WC_LBAT	Adaptive Threshold	LSTM	Weight
WC_STBAT	Adaptive Threshold	Statistic	Weight
LBAT [16]	Adaptive Threshold	LSTM	X
STBAT [4]	Adaptive Threshold	Statistic	X

the feedback messages and the statistical weights aggregated from collaborators. To recognize and learn patterns between factors well, the proposed collaborative detection method can also use LSTM.

2) LSTM-BASED COLLABORATIVE ATTACK DETECTION METHOD

The proposed LSTM-based collaborative source-side attack detection method learns the time dependency of detection results for individual collaborator, the relationship between collaborators with performance, and finally determines whether attack is happened or not. Figure 4 depicts the proposed LSTM-based collaborative attack detection model architecture. This model consists of three LSTM layers and dense layers. Dense layer uses the ReLU activation function. The number of nodes in each layer is the number of collaborators. The input vector consists of four features including the detection rate SR_z^k, the statistical weight w_z^k, the time window index t_z^k, and the changing rate of observed traffic volume ch_z^k at z^{th} time window of collaborator k. The input vector of collaborator k in the z^{th} time window is given as $CI_z^k = (SR_z^k, t_z^k, ch_z^k, w_z^k)$. The proposed LSTM model learns the input vector $CI_z^A, CI_z^B, \ldots, CI_z^P$ consisting of the detection information aggregated from collaborators from A to P. This LSTM model regresses to collaboration result CR_z between 0 (normal) and 1 (attack). The finial detection result FR_z is determined by comparing the calculated collaboration result CR_z and the static threshold th as shown in the equation 17.

$$FR_z = \begin{cases} 1 & \text{if } CR_z \geq th \\ 0 & \text{if } CR_z < th \end{cases} \tag{17}$$

When the attack is detected by the collaborative attack detection module, the final detection result FR_z is sent to the all of collaborators through the list of collaborator from trust management module. According to the final detection result FR_z received from the collaborative attack detection module, the source-side attack detection module of all collaborators need to set the adaptive threshold of the $(z+1)^{th}$ time window as shown in the equation 6.

IV. EVALUATION

A. DATASETS AND EVALUATION OVERVIEW

In order to evaluate the DoS attack detection method, we compare the performance of the proposed collaborative source-side attack detection method and the single source-side attack detection method. Table 1 shows the components of DoS attack detection method for performance comparison. In the single-source-side attack detection methods, there are seasonality-aware adaptive threshold (STBAT) [4] and LSTM-based adaptive threshold (LBAT) [16]. When the attack is detected, each method uses the adaptive threshold and predicts traffic for next threshold setting. In order to predict traffic on source-side network, the STBAT uses traffic statistic and the LBAT uses the LSTM model. This LBAT can use the static embedding and the dynamic embedding which are represented as S(N) and D(N), where N is the number of traffic states. Through the evaluation, the dynamic embedding uses n and τ set to 1 and 1%, respectively. The STBAT and the LBAT can be used as source-side attack detection method for the collaborative attack detection. In the collaborative source-side attack detection methods, there is a weight-based collaborative method (WC) and a LSTM-based collaborative method (LC). Depending on the source-side attack detection method used, these methods can be represented differently as shown in table 1. In order to determine the detection result finally, the WCs use the weight and the LCs uses the LSTM model. Through the evaluation, the static threshold of WCs set to 0.6 and the static threshold of LCs set to 0.1.

We evaluated the proposed method with the real-world DNS request traffic collected from DNS-STAT:Hedgehog which is operated by ICANN (Internet Corporation for Assigned Names and Numbers). We collect DNS request traffic from sixteen cities in three different countries: China, Brazil and USA. Each dataset has DNS request traffic for 30 days between September thirteen and October twelve of year of 2020. In order to analyze the relationship between the performance of the proposed model and the extracted non-linear features in detail, it is necessary to define it as an equation. Non-linearity caused by irregular user behavior can be distinguished into two categories. The first feature is characterized by high jitter due to the continuously fluctuating

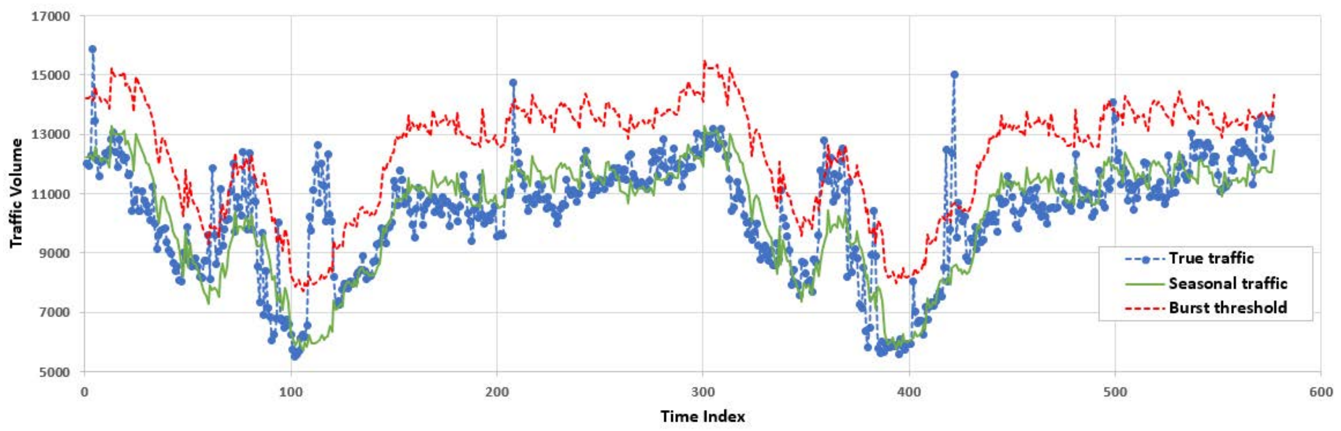

FIGURE 5. Actual Shanghai DNS request traffic volume and non-linear feature extraction using STL decomposition.

behavior of users, and the second feature is characterized by high burst ratio due to the sudden behavior of users. In order to extract these jitter and burst rates from the observed traffic, it is necessary to separate the seasonal traffic and the residuals caused by irregular user behavior. Figure 5 shows actual DNS traffic volume and non-linear feature extraction from Shanghai. We extract non-linear features using STL decomposition from traffic volume [32]. The STL decomposition method decomposes the observed traffic S_i by additive decomposition of seasonality ST_i, trend TR_i, and residual R_i, and it is represented as equation 18.

$$S_i = ST_i + TR_i + R_i \tag{18}$$

These seasonality ST_i, trend TR_i, and residual R_i are used to express burst rate BR and traffic jitter J. The burst ratio BR_i is the probability that a burst occurs in the z^{th} time window for a specific period, and the burst B_i and burst ratios BR are represent as equations 19 and 20.

$$B_i = \begin{cases} 1 & \text{if } R_i \geq 2 * \sum_{i=1}^{N} \frac{R_i}{N} \\ 0 & \text{if } R_i < 2 * \sum_{i=1}^{N} \frac{R_i}{N} \end{cases} \tag{19}$$

$$BR = \sum_{i=1}^{N} \frac{B_i}{N} * 100 \tag{20}$$

The average of jitter J is a characteristic in which a certain pattern is periodically repeated, and represents the ratio of the residual excluding seasonality and trend from the traffic volume. The average of jitter J is represented as equation 21.

$$J = \sum_{i=1}^{N} \left| \frac{R_i}{(ST_i + TR_i) * N} \right| * 100 \tag{21}$$

Table 2 shows the average of traffic volume, the average of traffic jitter, and the burst ratio of traffic for each DNS request traffic. This burst rate BR and average of jitter J are extracted from normal traffic and non-stationary traffic, and the performance of the proposed method can be compared according to the degree. When the burst ratio and the average of jitter is relatively high, the traffic behaves more in a non-linear, random and bursting manner. That is, traffic with relatively low jitter and low burst ratio behaves more in a linear and seasonal manner, while traffic with relatively high jitter and high burst ratio behaves more in a non-linear, random, and burst manner.

TABLE 2. Datasets of DNS request traffic.

Country	City	Arg.Volume	Arg.Jitter	Burst ratio
China	Xining	130.5	10%	8%
	Zhengzhou	9854	9%	9%
	Shanghai	10770.38	8%	13%
	Wuhan	6555.24	5%	10%
Brazil	Rio de Janeiro	1179	9%	13%
	Uberlandia	7539.81	8%	11%
	Sao Paulo	6416.57	6%	11%
	Fortaleza	1305.7	6%	12%
USA (East)	Lawrence	525.78	13%	11%
	Wilmington	756.39	10%	11%
	Denver	1069	5%	13%
	Reston	5342.99	6%	9%
USA (West)	Portland	78.2	9%	8%
	Los Angeles	3999.22	7%	11%
	San Jose	1040.11	7%	11%
	Anchorage	165.05	4%	9%

The traffic of the first 10 days of each dataset is used to train both of STBAT and LBAT, and the following 10 days of each dataset are used to calculate statistic weight for WC_LBAT, and the following 8 days of each dataset are used to train LC_LBAT, and the last 2 days of each dataset are used as a test set for evaluating how effectively each system detects the attack traffic mixed in legitimate traffic. For evaluation, we select 36 time window (20%) indices randomly from the test set, and infuse the attack traffic by increasing the volume of the traffic up to 10%. For each dataset, we evaluate the performance of STBAT and LBAT and summarize the result. We use a detection rate, false positive rate and balanced accuracy to evaluate each method. The detection rate is the percentage of detected attack when

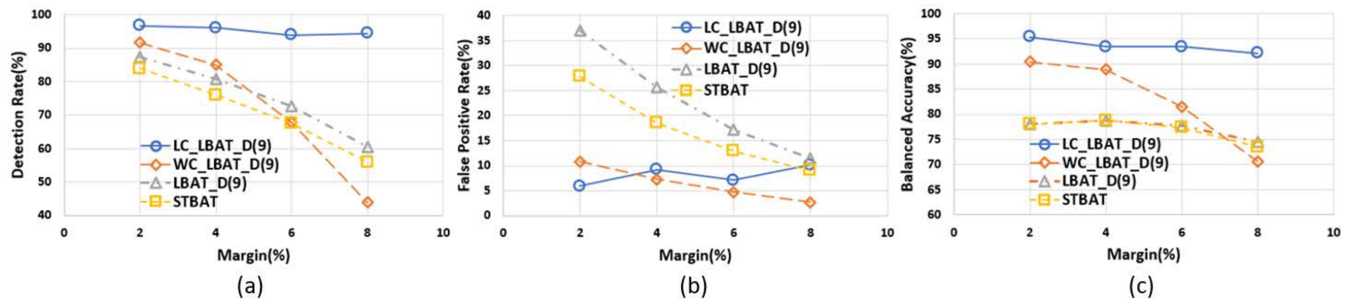

FIGURE 6. Performance comparison of collaborative source-side attack detection methods and source-side attack detection methods under highest burst ratio(Shanghai). (a) Detection Rate. (b) False Positive Rate. (c) Balanced Accuracy.

an attack is granted. The higher the detection rate, the higher the performance of the method. The false positive rate is the percentage of falsely detected attacks when the attack does not be granted. The higher the false positive rate, the lower the performance of the method. The balanced accuracy is the arithmetic mean of the detection rate and true negative rate. The main reason of using balanced accuracy is to see at a glance the difference in performance of the compared methods using the detection rate and the false positive rate. If the balance accuracy is high, the accuracy of the method is high.

B. PERFORMANCE COMPARISON

In order to understand the impact of burstiness to the proposed method, we evaluate the performance with various margins for the souce-side attack detection methods: LC_LBAT_D(9), WC_LBAT_D(9), LBAT_D(9), and STBAT. Figure 6 shows the performance comparison of the collaborative source-side attack detection methods and the single source-side attack detection methods under 2020 Shanghai DNS request traffic which has highest burst ratio between datasets. Because these methods detect the attack by comparing the observed traffic with the adaptive threshold to which the margin is applied, the detection rate and false positive rate are measured according to the percentage of margin.

First, we compare the performance between LBAT_D(9) and STBAT which are the single source-side attack detection methods. LBAT_D(9) shows relatively higher detection rate and false positive rate than STBAT. That is, LBAT_D(9) aggressively adjusts the detection threshold to detect attack traffic based on the fine grained traffic trend embedding, and it also increases the false positive rate as a side effect. Because of this, LBAT_D(9) and STBAT achieve the similar balanced accuracy when the margin is 4%. However, if the single source-side attack detection get support from multiple sites, the performance of attack detection can show higher performance than the single source-side attack detection method.

Second, we compare the performance of the weight-based collaborative source-side attack detection method WC_LBAT_D(9) and the single source-side attack detection methods. At the efficient margin 4%, WC_LBAT_D(9) shows around 3% higher the detection rate and shows around 12% lower the false positive rate than LBAT_D(9) and STBAT. That is, WC_LBAT_D(9) can effectively reduce the false positive rate by sharing the detection results of collaborators. However, if the accuracy of the final detection result which is shared with the collaborators can be increased, the performance of collaborative attack detection methods can be improved. By using an LSTM model that can learn time dependency and the relationship of collaborators from aggregated information of collaborator, the performance of collaborative attack detection can be improved.

Third, we compare the performance WC_LBAT _D(9) and LC_LBAT_D(9). LC_LBAT_D(9) shows about 8% higher detection rate than WC_LBAT_D(9) by using various information for the final attack detection. However, because of slight irregular pattern between information when they are learned on LSTM of collaborative attack detection module, the false positive rate increases up to around 3%.

Finally, if the performance of each method is compared at a glance as shown in Figure 6, the performance is good in the order of LC_LBAT_D(9), WC_LBAT_D(9), LBAT_D(9), and STBAT. However, depending on the type of source-side attack detection methods in multiple sites, the performance of the collaborative source-side attack detection methods may differ.

Figure 7 shows the performance comparison of the collaborative source-side attack detection according to individual source-side attack detection methods under 2020 Shanghai DNS request traffic. The collaborative source-side attack detection methods WCs and LCs are represented differently depending on whether STBAT, LBAT_S(9), and LBAT_D(9) methods are applied. In the case of WCs, these methods are represented as WC_STBAT, WC_LBAT_S(9), and WC_LBAT_D(9), respectively. And In the case of LCs, these methods are represented as LC_STBAT, LC_LBAT_S(9), and LC_LBAT_D(9), respectively.

Because WCs use the detection result and the weight of collaborators for the final detection result, the performance of the WCs are affected by the type of source-side attack detection methods. In other words, WCs can reduce false positive rate while maintaining high detection rate. Therefore, if the source-side attack detection method with the high detection rate is adopted as collaborator of WCs regardless of

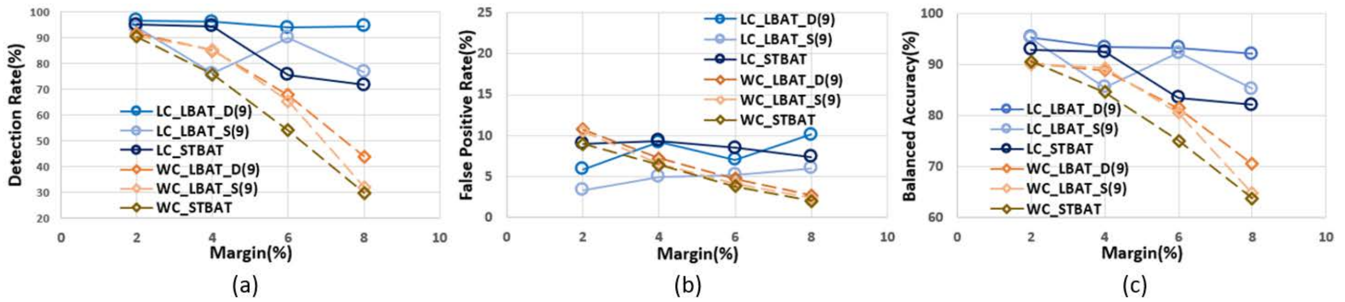

FIGURE 7. **Performance comparison of collaborative source-side attack detection methods according to source-side DoS attack detection method under highest burst ratio(Shanghai). (a) Detection Rate. (b) False Positive Rate. (c) Balanced Accuracy.**

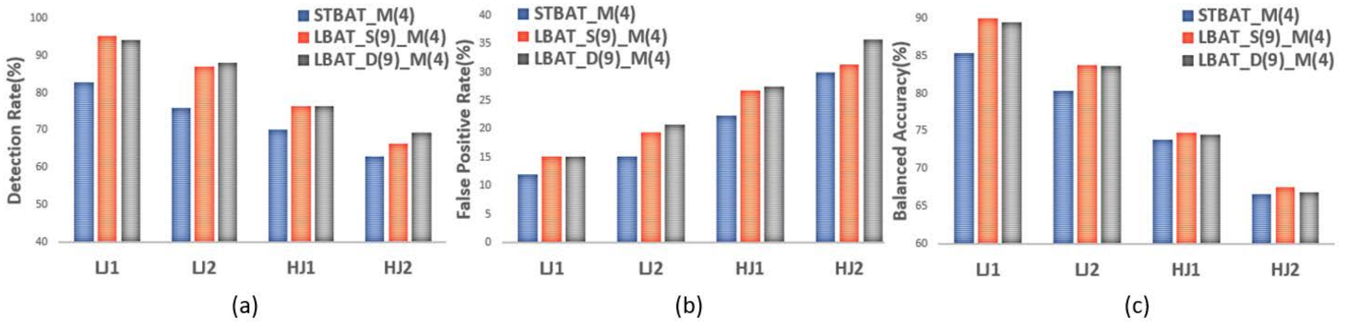

FIGURE 8. **Performance comparison of source-side DoS attack detection methods under different Avg. Jitter (LJ1:Denver, LJ2:Anchorage, HJ1:Portland, HJ2:Wilmington). (a) Detection Rate. (b) False Positive Rate. (c) Balanced Accuracy.**

the side effects, overall performance will be improved. At the efficient margin 4%, WC_LBAT_D(9) and WC_LBAT_S(9) show around 10% higher the detection rate than WC_STBAT, and the false positive rates of WCs are similar.

For the final attack detection, LSTM models of LCs learn the relationship and the time dependency between collaborators, so LCs show higher detection rate than WCs. However, because LSTM models of LCs learn irregular pattern between information of collaborators, the false positive rates of LCs can be higher. LC_LBAT_D(9) and LC_STBAT show around 10% higher detection rate but show around 4% lower than WC_LBAT_D(9) and WC_LBAT_S(9) when the margin is 4%. In the case of LC_LBAT_S(9), the source-side attack detection result patterns may become somewhat irregular due to static seasonal embedding. LC_LBAT_S(9) shows around 5% lower detection rate, but shows around 2% lower false positive rate. Through this evaluation, it can be seen that LC_LBAT_D(9) shows the highest performance. However, the performance of the LBAT_D(9) may change according to jitter, which is one of the non-linear characteristics.

Figure 8 shows the performance comparison with datasets which have different jitter level. As we described earlier, datasets for Denver and Anchorage have low jitter and datasets for Portland and Wilmington have high jitter. In Figure 8, LJ and HJ stands for low and high jitter, respectively. Each detection method, we use the same value of margin as 4 which is used for the adaptive threshold, and it is represented as M(4) suffix. We select a dataset with a relatively low traffic volume in order to focus on the change according to the jitter level. It is observed that the overall detection rate decreases and the false positive rate increases as the average of jitter increases. At this time, as shown in the balanced accuracy, the performance of LBAT_S(9) and LBAD_(9) is similar. Because LSTM can learn and mitigate irregular non-linear characteristics in traffic, LBAT can keep its detection rate from around 95% down to around 69%, while STBAT drops its detection rate from 82% down to 63%. Especially, the performance degradation happens significantly when the network traffic has high randomness by the high jitter. Though the randomness cause the performance degradation, the LBAT mitigates this degradation. This endurance against the randomness is observed in the balanced accuracy result as well. In this way, the performance of the collaborative source-side attack detection module, LC_LBAT, may be lowered by multiple sites that represent high jitter. In order to improve and maintain the performance of the proposed framework, it is necessary to manage the configuration of the collaboration network.

Figure 9 shows the performance comparison of the LC_LBAT_D(9)_M(4) method according to the number of collaborators which have different jitter. In the same way as above, we select the effective margin of LC_LBAT_D(9) as 4%. At this time, when we construct Low Jitter Collaborative network(LJCN) and High Jitter Collaborative network(HJCN), we selected relatively low or high jitter datasets as shown in Table 2. In the collaborative network composed

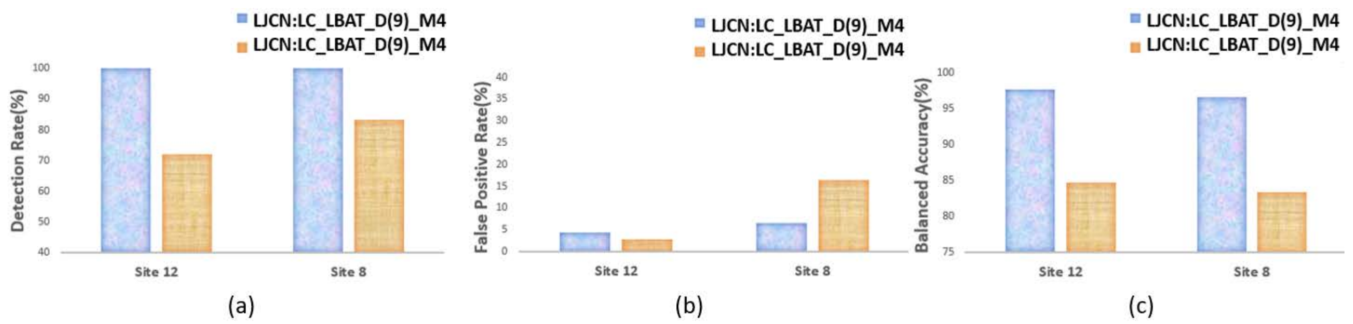

FIGURE 9. **Performance comparison of LC_LBAT_D(9)_M(4) methods by the number of collaborators according to jitter. (a) Detection Rate. (b) False Positive Rate. (c) Balanced Accuracy.**

of relatively low jitter, LJCN, LC_LBAT_D(9)_M(4) shows high detection rate and around 4% false positive rate regardless of the number of collaborators. However, because of the increased overall randomness in the collaborative network composed of relatively high jitter, HJCN, the performance of LC_LBAT_D(9)_M(4) becomes unpredictable. When the number of collaborators which configure HJCN is 12, the detection rate is 72% and the false positive rate is around 3% in LC_LBAT_D(9)_M(4). When the number of collaborators which configures HJCN is 8, the detection rate is round 85% and the false positive rate is round 15% in LC_LBAT_D(9)_M(4). In performance comparison with the balanced accuracy, the fewer the number of HJCNs, the lower the performance of LC_LBAT_D(9)_M(4). Therefore, when constructing a cooperative network, it is recommended to exclude sites with relatively high jitter and high burst rate from the list of collaborators.

V. CONCLUSION

In this paper, we propose the LSTM-based collaborative source-side attack detection framework. In order to improve the performance of source-side attack detection system located in the target subnet, the proposed framework aggregates the weight and traffic pattern from collaborators and implement the LSTM based collaborative attack detection to finally determine the attack detection. For the verification and management of collaborator, the trust management module in the proposed framework sends test messages to collaborator in list periodically.

Through extensive evaluation of actual DNS request traffic, the proposed LSTM-based collaborative source-side attack detection method outperforms the previous single-source-side attack detection method and the weight-based collaborative source-side attack detection method. At an effective margin of 4%, The proposed method shows the 25% higher detection rate and a 10% lower false positive rate compared to the single-source attack detection method at a valid margin of 4%. The performance of the proposed collaborative source-side attack detection method may vary depending on the source-side attack detection method. If dynamic seasonality embedding is applied, the performance of source-side attack detection method can be improved. The performance of the proposed method can show higher performance by excluding collaborators with high burst rate and jitter from the collaborator list.

However, in the proposed framework, the availability of the collaborative detection method can be changed depending on the network link state. In addition, it is necessary to understand the complex relationship of each site, not as a network having a graph structure. For efficient link state prediction in large-scale graph networks, it is necessary to study a heterogeneous graph embedding method that represents complex relationships between links.

REFERENCES

[1] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: Perspectives and challenges," *Wireless Netw.*, vol. 20, no. 8, pp. 2481–2501, Nov. 2014.

[2] Z. He, T. Zhang, and R. B. Lee, "Machine learning based DDoS attack detection from source side in cloud," in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Jun. 2017, pp. 114–120.

[3] S.-N. Nguyen, V.-Q. Nguyen, G.-T. Nguyen, J. Kim, and K. Kim, "Source-side detection of DRDoS attack request with traffic-aware adaptive threshold," *IEICE Trans. Inf. Syst.*, vol. E101.D, no. 6, pp. 1686–1690, 2018.

[4] G.-T. Nguyen, V.-Q. Nguyen, S.-N. Nguyen, and K. Kim, "Traffic seasonality aware threshold adjustment for effective source-side dos attack detection," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 5, pp. 2651–2673, 2019.

[5] N. Ramakrishnan and T. Soni, "Network traffic prediction using recurrent neural networks," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 187–193.

[6] Y. Geng and S. Li, "A LSTM based campus network traffic prediction system," in *Proc. IEEE 10th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2019, pp. 327–330.

[7] A. Lazaris and V. K. Prasanna, "An LSTM framework for modeling network traffic," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 19–24.

[8] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, Feb. 1994.

[9] F. K. Oduro-Gyimah and K. O. Boateng, "Investigating the influence of seasonality and structural breakpoint on telecommunication network traffic prediction," in *Proc. IEEE 7th Int. Conf. Adapt. Sci. Technol. (ICAST)*, Aug. 2018, pp. 1–9.

[10] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, O. Hohlfeld, and G. Smaragdakis, "The lockdown effect: Implications of the COVID-19 pandemic on internet traffic," in *Proc. ACM Internet Meas. Conf.*, Oct. 2020, pp. 1–18.

[11] H. Jiang and C. Dovrolis, "Why is the internet traffic bursty in short time scales," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, 2005, pp. 241–252.

[12] F. Ciucu and J. Schmitt, "On the catalyzing effect of randomness on the per-flow throughput in wireless networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 2616–2624.

[13] H. Gwon, C. Lee, R. Keum, and H. Choi, "Network intrusion detection based on LSTM and feature embedding," 2019, *arXiv:1911.11552*.

[14] M. Bhanu, J. Mendes-Moreira, and J. Chandra, "Embedding traffic network characteristics using tensor for improved traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3359–3371, Jun. 2021.

[15] J. Cui, J. Long, E. Min, and Y. Mao, "WEDL-NIDS: Improving network intrusion detection using word embedding-based deep learning method," in *Proc. Int. Conf. Modeling Decisions Artif. Intell.*, Cham, Switzerland: Springer, 2018, pp. 283–295.

[16] S. Yeom, C. Choi, and K. Kim, "Source-side DoS attack detection with LSTM and seasonality embedding," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, Mar. 2021, pp. 1130–1137.

[17] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 3, pp. 38–47, Jul. 2001.

[18] M. A. Lopez, R. S. Silva, I. D. Alvarenga, G. A. F. Rebello, I. J. Sanz, A. G. P. Lobato, D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "Collecting and characterizing a real broadband access network traffic dataset," in *Proc. 1st Cyber Secur. Netw. Conf. (CSNet)*, Oct. 2017, pp. 1–8.

[19] M. Mantere, M. Sailio, and S. Noponen, "Network traffic features for anomaly detection in specific industrial control system network," *Future Internet*, vol. 5, no. 4, pp. 460–473, Sep. 2013.

[20] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, "A trust-aware, P2P-based overlay for intrusion detection," in *Proc. 17th Int. Conf. Database Expert Syst. Appl. (DEXA)*, 2006, pp. 692–697.

[21] C. J. Fung, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Jun. 2009, pp. 33–40.

[22] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[23] S. Yeom and K. Kim, "Improving performance of collaborative source-side DDoS attack detection," in *Proc. 21st Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2020, pp. 239–242.

[24] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[25] Y. Zhang, "An adaptive flow counting method for anomaly detection in SDN," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, Dec. 2013, pp. 25–30.

[26] R. F. Fouladi, O. Ermiş, and E. Anarim, "A DDoS attack detection and defense scheme using time-series analysis for SDN," *J. Inf. Secur. Appl.*, vol. 54, Oct. 2020, Art. no. 102587.

[27] J. Mirkovic and P. Reiher, "D-WARD: A source-end defense against flooding denial-of-service attacks," *IEEE Trans. Depend. Secure Computing*, vol. 2, no. 3, pp. 216–232, Jul. 2005.

[28] A. H. Yaacob, I. K. T. Tan, S. F. Chien, and H. K. Tan, "ARIMA based network anomaly detection," in *Proc. 2nd Int. Conf. Commun. Softw. Netw.*, Feb. 2010, pp. 205–209.

[29] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, "Traffic flow forecasting neural networks based on exponential smoothing method," in *Proc. 6th IEEE Conf. Ind. Electron. Appl.*, Jun. 2011, pp. 376–381.

[30] B. Song, J. Heo, and C. S. Hong, "Collaborative defense mechanism using statistical detection method against DDoS attacks," *IEICE Trans. Commun.*, vols. E90–B, no. 10, pp. 2655–2664, Oct. 2007.

[31] V.-Q. Nguyen, S. N. Nguyen, and K. Kim, "Enabling disaster-resilient SDN with location trustiness," in *Proc. 4th Int. Conf. Comput. Appl. Inf. Process. Technol. (CAIPT)*, Aug. 2017, pp. 1–4.

[32] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition," *J. Off. Statist.*, vol. 6, no. 1, pp. 3–73, 1990.

SUNGWOONG YEOM received the B.S. and M.S. degrees from the Department of Electronics and Computer Engineering, Chonnam National University, South Korea, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree in artificial intelligence convergence. His research interests include implementations of network anomaly detection, flow control, routing, GRID/cloud systems, and software-defined networking with deep learning.

CHULWOONG CHOI received the B.S. and M.S. degrees in computer statistics and software convergence engineering from Chosun University, South Korea, in 2014, and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Artificial Intelligence Convergence, Chonnam National University, South Korea. His research interests include distributed systems, recommended systems, and deep learning. His current focus is on the medical bigdata analysis and distributed deep learning framework.

KYUNGBAEK KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 1999, 2001, and 2007, respectively. He is currently a Professor with the Department of Artificial Intelligence Convergence, Chonnam National University. Previously, he was a Postdoctoral Researcher with the Department of Computer Sciences, University of California at Irvine, Irvine, CA, USA. His research interests include intelligent distributed systems, software-defined networks/infrastructure, big data platform, GRID/cloud systems, social networking systems, AI applied cyber-physical systems, blockchain, and other issues of distributed systems. He is a member of ACM, IEICE, KIISE, KIPS, KICS, KIISC, and KISM.