IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received January 22, 2020, accepted February 10, 2020, date of publication February 17, 2020, date of current version March 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974293

RTVD: A Real-Time Volumetric Detection Scheme for DDoS in the Internet of Things

JIABIN LI[1], MING LIU[1], ZHI XUE[1], XIAOCHEN FAN[2], (Student Member, IEEE), AND XIANGJIAN HE[3,4]

[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University (SJTU), Shanghai 200240, China
[2]School of Electrical and Data Engineering, University of Technology Sydney (UTS), Ultimo, NSW 2007, Australia
[3]Computer Vision and Pattern Recognition Laboratory, Global Big Data Technologies Centre (GBDTC), University of Technology Sydney (UTS), Ultimo, NSW 2007, Australia
[4]Network Security Research Team, Center for Real-time Information Networks(CRIN), University of Technology Sydney (UTS), Ultimo, NSW 2007, Australia

Corresponding author: Ming Liu (liuming198904@sjtu.edu.cn)

This work was supported by the Foundation Item: Cyber Security from the National Key Research and Development Program of Shanghai Jiao Tong University under Grant 2017YFB0803203.

ABSTRACT Distributed Denial of Service (DDoS) attacks are increasingly harmful to the cyberspace nowadays. The attackers can now easily launch a bigger and more challenging DDoS attack both towards and with Internet-of-Things (IoT) devices, due to the fast popularization of them. Because of the characteristic of fast overwhelming, it is important to make fast as well as accurate response to DDoS attacks, and the real-time performance can be even more important to prevent and legitimate the attacks. Among the methods proposed by researchers, the entropy-based detection method provides a sensitive and reliable performance. However, the balance between computational complexity and recognition accuracy remains a challenge. In this paper, we propose a detection method that consists of 3 main parts in different aspects: a sliding time window to fasten the entropy calculation, a single-directional filter to realize early detection during the DDoS progress but not after the crash, and a quintile deviation check algorithm to optimize the detection result. These will eventually lead to a real-time and high-efficient performance to recognize IoT DDoS attacks as soon as possible.

INDEX TERMS DDoS detection, IoT security, joint entropy, quintile deviation check, real-time detection, sliding time window.

I. INTRODUCTION

It has been decades since the Distributed Denial-of-Service (DDoS) attack pattern first appeared [1], yet the DDoS attacks are still huge challenges. Traditionally, a Denial-of-Service (DoS) attack is characterized by an explicit attempt by attackers to prevent the legitimate use of a service, and a DDoS attack deploys multiple devices, usually millions of devices, to attain this goal [2]. In September 2016, the website of computer security consultant Brian Krebs was hit with 620 Gbps of traffic. The DDoS attacks can do much faster and more severe harm to the Internet, yet the attacks can be even stealthier than DoS attacks are. Due to the differences between DoS and DDoS, the defense strategies can be totally different.

Unfortunately, the security situation is even more severe especially in the era of Internet-of-Things (IoT) today. With the up growing number of IoT devices, attackers can easily take control of far more infected bots than ever before. What's more disturbing is that, the concepts of IoT, such as the modern Internet of Vehicles [3], [4], edge computing [7] and Information-Centric Networking (ICN) [5], [6] etc., are having increasingly stronger computing power [8]. In late 2016, Mirai [9], the ever largest IoT botnet, outbroke in 2016 and launched a DDoS attack on 20th Sept., with the maximum speed of 1.5 Tbps on OVH, the French web host and cloud service provider, and launched another DDoS attack on a DNS management service provider in America, which caused a large-area network paralysis in eastern United States. Many

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

American websites, including Twitter and Facebook, failed to be accessed by their domain names. In the offline event in Germany, attackers launched a flood scan on 7,547 port with Mirai, about 900,000 Deutsche Telekom routers crashed during the campaign, resulting in a large scale of limited network access. In late November 2017, a Mirai variant was found launching attacks on IoT devices in Argentina through port 23 and 2323 [10]. Besides Mirai, there already exist many IoT botnets such as QBOT, Luabot, Bashlight, Zollard, Remaiten, KTN-RM, etc., and most of them provide the DDoS service. In fact, few IoT viruses exclude the function of DDoS because IoT is perfect to launch a DDoS attack, making the DDoS more threatening.

In an original DDoS attack, the attacker controls a huge number of devices and threads to directly consume the resources of the victim servers. A Direct DDoS Attack itself is simple to understand and realize, but the attacker must prepare an attacking network to manage and control all the devices by adopting a botnet or a Trojan virus, which enriches the cost of an attack. To avoid this problem, or to make the attack more dangerous, the strategy of (Amplified) Reflective Attack is proposed. In a reflective DDoS attack, the attacker takes advantage of the vulnerabilities of those protocols which make more responding resources than requesting ones, such as DNS, NTP, SSDP, etc. The attacker designs and sends some fake request packets, which are claimed to be sent from the victims to the servers running those reflective protocols. Usually, the servers won't check whether the source IP addresses are valid or real and will send the query results to the victim IP addresses.

In another classification of flow quantity, DDoS attacks are divided into Volumetric attacks and Low and Slow attacks. Volumetric attacks are the traditional type of DDoS, and they overwhelm the targets by consuming network, storage or computing resources. Huge Volume DDoS attacks send a high amount of traffic or request packets to a targeted network or device to overwhelm its bandwidth capabilities, connection limit or other network resources. In contrast, Low and Slow attacks, also known as low-rate DDoS attacks (LDDoS) or Slow DDoS attacks, are mainly based on the vulnerabilities of layer-7 protocols and involve "what appears to be legitimate traffic at a very slow rate" [11] to occupy and consume the connections. The earliest Slow DDoS is SlowLoris [12], and it is first released by Robert "RSnake" Hansen in June 2009, which allows a single machine to take down another machine's web server with minimal bandwidth and side effects on unrelated services and ports.

Generally, Slow DDoS attacks can go through Volume DDoS detectors, because their packets are usually valid and real, and cause few volume or number of requests, so some researchers claim that Slow DDoS attacks are more technically superior, more dangerous and stealthier than volume attacks. We used to believe that as well, but the reports and data show the result. As is shown in the DDoS Threat Report 2018 Q3 [13], "The maximum attack size increased 139.84% Year-over-Year to 118Gbps", and in the Nexusguard Threat Report (DDoS) 2019 Q3 [14], the top attacking vectors are DNS Amplification, HTTP(S) Flood and TCP SYN Flood, as shown in Fig. 1. Since the Huge Volume DDoS attack is still one of the biggest threats in cyberspace and is turning its attacking mode into a brand-new faster one, it is necessary to make the detection more efficient and flexible.

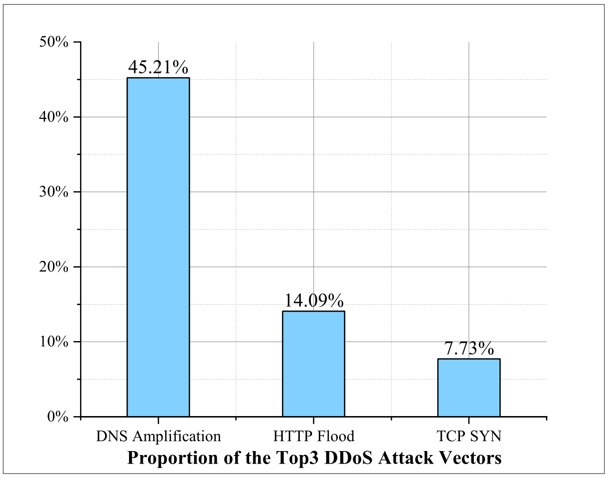


FIGURE 1. Ranking of different-type ddos attacking vectors [14].

Volume DDoS attacks flood the target in the hopes of slowing or stopping their services. There are 3 main statistical characteristics of the DDoS traffic:

1) Huge traffic size. As in the big data era today, the in-out traffic of a personal web server can easily come to 1000's of Mbps, and when a DDoS attack occurred, typically, the request sizes are in the 100's of Gbps, and reports show that recent attacks have scaled to over 1Tbps.

2) Various source info. Not only does the source Port vary, but the source IP address also varies in a wide range. Usually, the attackers will use the fast-flux technology [15] and an IP/MAC pool for IP spoofing, or will simply generate fake packet heads for those non-source-IP-check protocols. Considering the huge size of network traffic, the range of values can be shocking.

3) Take the effect quickly. The DDoS attack will arouse a huge number of traffic or requests and crash the target victim in a short duration, which means there will be little interaction time for the detection and following defense against the attack.

In this paper, we focus on fastening the detection of volume type DDoS and designed a real-time alarming system. The contribution of this paper is described as follows:

1) We optimize the method of sliding time window to fasten the entropy calculation algorithm;

2) We design a Quintile Deviation Check algorithm to identify the attacks;

3) We redefine a temporal indicator to evaluate the time performance of DDoS detection.

With the 3 contributions, we realize a real-time volumetric detection scheme for DDoS in the Internet of Things.

The rest of this paper is organized as follows. In Section II, we describe the related works on the usage of the sliding time window and DDoS detection methods. In Section III, we list the nomenclature of all the terms, abbreviations and mathematical symbols used in the following paragraphs. In Section IV, we discuss the detection architecture as well as the mathematical theories that our detection is based on. And we set 3 subsections to explain the concepts of Traffic Processor, Entropy Calculator, and Detection & Alarm Module in detail. In Section V, we make simulations on 3 public datasets and a generated dataset to prove and evaluate the detection scheme we propose. In Section VI, we conclude the whole paper and point out the direction of future research.

II. RELATED WORKS

Research works have been done with various strategies to detect DDoS attacks with sliding time window or information entropy methods in recent years.

Early in 2006, the sliding window was used in DDoS detection in [16]. The authors use two adjacent non-overlapping windows: the referenced sliding window and the test sliding window, and use an autoregressive model to recognize anomalous traffic. Reference [18] mentioned a real-time DDoS detection technique, using a sliding window and source IP clusters, and using Spark to meet the distributed demands. In years before the DDoS attacks had the characteristics of a relatively fixed number of packets per second, packet bytes, gap periods, etc. However, the situation has changed and the technique of source IP cluster technique won't work as well as before, but other parts of the research provide many inspirations.

In [19], the authors propose a new two-step method for DDoS attack detection, which combines the approaches of network traffic entropy and the TSK Fuzzy System (TSK-FS), and shows that the TSK-FS DDoS detector reaches enhanced sensitivity and robustness in the detection process. Reference [20] shows a method to identify DDoS attacks by computing entropy and frequency-sorted distributions of selected packet attributes. The authors also use the sliding window to simplify the computation of the entropy, which fastens the calculation, but they didn't discuss the aspect of accuracy in detail. In [21] the authors study information-theoretic methods to detect intrusions in the network. They consider the usage of entropy, conditional entropy and information gain to help partition and setting parameters for existing detection models. On this basis, Virmani *et al.* proposed an entropy deviation method for network intrusion analysis [22]. They used the variation value of source IP entropy packet-by-packet to classify the benign (trusted) and malicious IP addresses. Reference [17] presented a DDoS attack detection and mitigation system with Software Defined Network (SDN). They collected the network traffic information using the SDN controller and sFlow agents in advance and then identified abnormal network traffic with an entropy-based method and a Support Vector Machine (SVM) classifier. Other researches also mentioned detection or mitigation methods such as Threat Intelligence (TI) information sharing [23] and command lines analysis [24].

Based on all the researches above, entropy works well in the DDoS detection scenario, but the balance between efficiency and accuracy is a big challenge. The mentioned reference [20] uses a sliding window to optimize the time performance, and base on that, our research in this paper does further progress.

TABLE 1. TFOR of different DDoS attacks using QuinDC.

Abbr. or Symbol	Description and Definition
(D)DoS	(Distributed) Denial of Service.
LDDOS	Low-rate Distributed Denial of Service.
5-Tuple	RFC 5766 defined basic information segements in the header of a network packet, which consists of "source IP, source port, destination IP, destination port and protocol".
s_IP / d_IP	Source/destination IP address.
s_Port / d_Port	Source/destination port number.
Src, Dst, P	The source segments, destination segments and protocol of a 5-tuple.
LSTM	Long Short-Term Memory, one of the Recurrent Neural Betwork (RNN) model, usually used in time sequence prediction and natural language processing (NLP).
QuinDC	Quintile Deviation Check.
I_{DU}, I_{SU}, I_C, I_{SD}, I_{DD}	5 subintervals in QuinDC, i.e., Divergent Interval (Upward), Shock Interval (Upward), Convergent Interval, Shock Interval (Downward) and Divergent Interval (Downward).
FPR, FNR, FOR	False Positive Rate, False Negative Rate and False Omission Rate. Popular indices of performance evaluation.
TFOR	Temporal False Omission Rate. Based on tratidional False Omission Rate, and take temporal into the evaluation.
S	A given time sequence.
n	The length of S.
w	The width of a sliding window.
S_x	The x^{th} w-length subsequence of S.
$H(x)$	The entropy value of S_x.
e	Element that appears in S_x.
e_{In}/e_{Out}	The read-in/pushed-out element.
Δx	Deviation value of x.
D	Array of dictionaries.
d_x	The x^{th} dictionary in D.
C	The matrix of value counters.
v_x	The value vector of d_x.
lb	The length of how many elements that the LSTM model will look back, also means the input length.
p	The predicted output value.
N	The length of sliding window in QuinDC.
k	The group length in QuinDC.
ϕ	The threshold parameter in QuinDC.
e_{max}, e_{min}, e	The maximum, minimum and average value of the first ϕ entropy values in the previous sliding window.
r	The radius of the changing interval.

III. NOMENCLATURE

To make a clearer expression, in this section, we list all the terms, abbreviations and mathematical symbols that are used in the following paragraphs, as well as their descriptions and definitions.

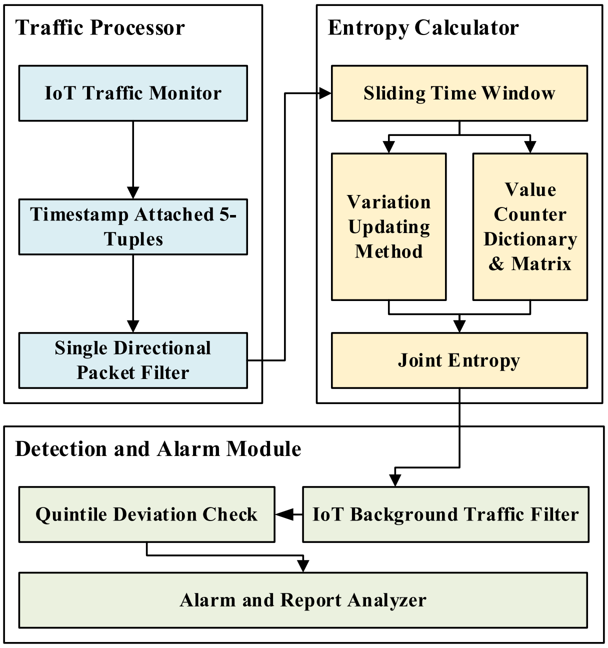


FIGURE 2. DDoS detection architecture.

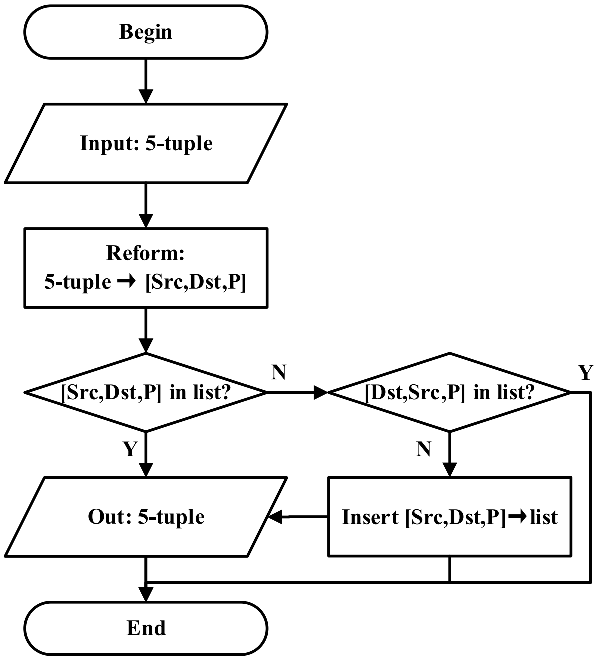


FIGURE 3. Flow chart of single directional packet filter.

IV. DETECTION ARCHITECTURE AND MATHEMATICAL THEORIES

The detection system consists of 3 parts: a traffic processor, an entropy calculator, and a detection module, and the architecture is shown in Fig. 2. The traffic processor monitors the traffic in the whole network, preprocesses the traffic data and then forwards the data to the entropy calculator. The entropy calculator calculates the real-time entropy values and sends the values to the detection module. The detection module matches the entropy values with the given risk models and decides whether to send out an alarm. In the following sections, we will introduce the mechanism and theories of the 3 parts in detail.

A. TRAFFIC PROCESSOR

1) MONITORING AND THE TIMESTAMP ATTACHED 5-TUPLES

Due to the existence of security vulnerabilities in traditional network architectures, DDoS attacks are becoming increasingly dangerous. To contain the consequence and also to trace the attacks, the industry and academia proposed the concept of software-defined network (SDN), whose centralized control network provides support for the defense against DDoS attacks [25]. Within an SDN environment, we can trace and filter the DDoS traffic, can get the traffic packet in real-time and can make real-time detection and defense.

Since we can get real-time traffic packets, the next thing that matters most is to extract the concerned segments from the input data. The traditional Request-For-Comments (RFC)-defined traffic identifier is the 5-Tuple [26], [27]: [Source IP, Source Port, Destination IP, Destination Port, Protocol]. Analysts use 5-tuples to distinguish between different network flows. Using 5-tuple only will lose time-domain characteristics. To solve this problem, we also append the timestamp of each packet to the tuple as the expiring threshold variable.

2) SINGLE DIRECTIONAL PACKET FILTER

A complete DDoS attack can be divided into 3 stages of preparing, attack accumulating, and saturation. The attackers collect the necessary information and intelligence of the target victims and deploy botnets of Trojans during the preparation stage. After the preparation, the attackers send out a command to launch the attack. As the connection numbers, network resources or computing resources are being gradually occupied, the service capacity keeps declining, and the victims will finally be unable to respond. Most detection methods are based on this characteristic, but this means the attackers have already fulfilled their goals. If we want to prevent the attack before the victims paralyzed, we must further shorten the responding time and latency. However, another problem is that, during the accumulating stage. the victim server can still make responses to the packets, including the malicious ones. We will not only see the victims' IP addresses in the destination IP segment but also see as many attackers' and legal users' IP addresses as well. This will lead to a failure in applying entropy detection until the attack enters the saturation stage.

To eliminate the influences from the victims' response packets, a popular approach is to use flow information instead of packet information, because the network traffic flow will keep only the first packet in the flow. However, this will also lead to false statistical characteristics. Popular flow extractors (e.g., bro and tshark) usually work on a captured pcap file and

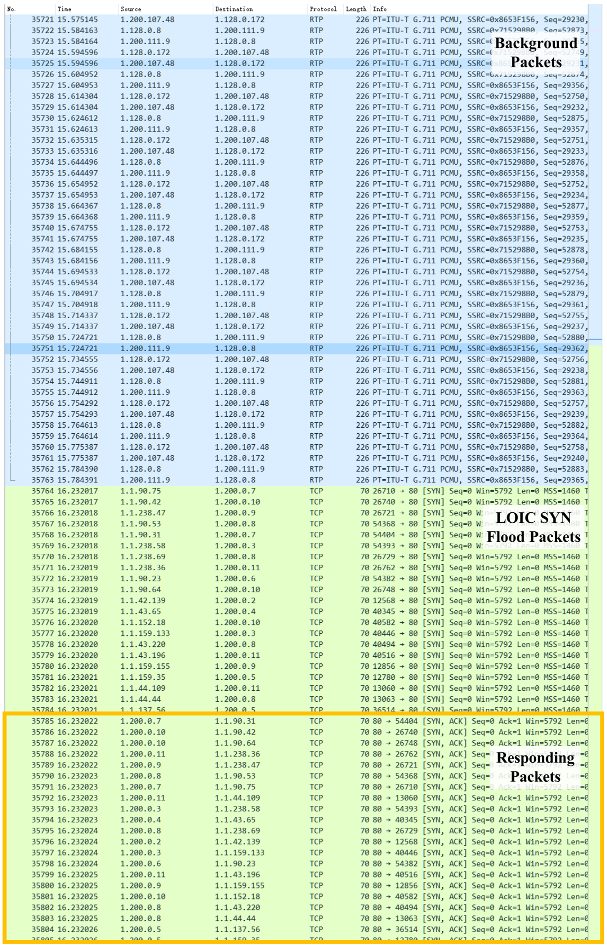


FIGURE 4. LOIC-TCP-SYN flood packets (screenshot from Wireshark).

loses real-time capability. In this paper, we propose a single directional packet filter and tried to advance the detection to the attack accumulation stage. Fig. 3 describes the flow chart of the filter.

The filter takes every sent-in 5-tuple as an input, and only allows the tuples whose direction is the same with the first tuple of a flow to pass. The filter maintains a list to record the identifier of every flow. We reform each 5-tuple into a 3-part array[Src:[S_IP, S_Port], Dst:[D_IP, D_Port], P:Protocol], and whenever a 5-tuple is sent in, the filter first check whether [Src, Dst, P] or [Dst, Src, P] is already in the list. If [Src, Dst, P] doesn't exist, append [Src, Dst, P] into the list; else if [Src, Dst, P] exists, let the tuple pass the filter; else drop it.

Figure 4 shows a beginning part of an accumulation stage of a LOIC TCP SYN Flood DDoS attack. The attackers' IP range is 1.1.0.0/16, while the victim servers' IP fall in 1.200.0.0/24. In the example, we can easily notice that the distribution of source information and destination information becomes unexpectedly the same, and we are unable to locate the attacking event until we remove the response packets.

B. ENTROPY CALCULATOR

Entropy theory is effective in DDoS detection because when an attack occurs, the IP addresses and port numbers of the malicious traffic are provided by thousands of millions of different hosts that are fake, virtual or illegally controlled, while under legal usage. However, they would be confined to a relatively small range, and that will show a sharp contrast from the entropy perspective.

1) OPTIMIZED SLIDING TIME WINDOW

$$H(X) = E[-\log p(x_i)] = -\sum_{i=1}^{n} p(x_i)\log p(x_i) \tag{1}$$

The entropy value can directly represent the distribution of information, but according to Shannon's original entropy formula (1), if we take all accumulated data as the input, whenever we receive new input, we have to recalculate the entropy. The time complexity of the original algorithm is $O(n^2)$, which will cause unacceptable responding time in DDoS detection. To avoid this disadvantage, Feinstein *et al.* use the sliding window while computing [20] and reduce the time complexity extremely from $O(n^2)$ to $O(w^2) = O(1)$, where w is a constant value and is less than n. Because the series of entropy value describes the trend of the value distribution, the input values that have arrived too long before should have few influences on the current input. Therefore, we can set a proper constant width of a sliding window, i.e. 10^4 packets, and apply the sliding window to the input values to calculate the entropy. Comparing with (1), the upper bound of i is limited to be less than or equal to the width w. The method makes it possible to apply entropy to DDoS detection, but the response time can be further reduced. Taking the result as a basis, we propose 2 methods to further reduce the time complexity.

a: VARIATION UPDATING METHOD

The calculation of the entropy is accumulative. If we apply a sliding time window, we can only recalculate the information content of both the slide-out and slide-in elements, and keep most other unaffected elements the same, thus simplifying the calculation. Supposing that there is an input sequence $s[0:5] = [x_1, x_2, x_1, x_1, x_3, x_4]$, and the width of the sliding window is $w = 5$. Then it is easy to calculate the entropy of the first sliding window:

$$\begin{aligned} &H(x[0:4]) \\ &\quad = h(x_1) + h(x_2) + h(x_3) \\ &\quad = -\frac{3}{5}\cdot\log\frac{3}{5} - \frac{1}{5}\cdot\log\frac{1}{5} - \frac{1}{5}\cdot\log\frac{1}{5} \end{aligned} \tag{2}$$

When the window slides forward by a step, the entropy becomes:

$$\begin{aligned} &H(x[1:5]) \\ &\quad = h(x_1) + h(x_2) + h(x_3) + h(x_4) \\ &\quad = -\frac{2}{5}\cdot\log\frac{2}{5} - \frac{1}{5}\cdot\log\frac{1}{5} - \frac{1}{5}\cdot\log\frac{1}{5} - \frac{1}{5}\cdot\log\frac{1}{5} \end{aligned} \tag{3}$$

Giving a subscript of [0 : 4] to the information contents in (2) and [1 : 5] to those in (3), we can get the relation between (2) and (3):

$$H(x[1:5]) = H(x[0:4]) - h_{[0:4]}(x[0]) + h_{[1:5]}(x[0]) - h_{[0:4]}(x[5]) + h_{[1:5]}(x[5]) \quad (4)$$

Let $\Delta h(x) = h_{[1:5]}(x) - h_{[0:4]}(x)$:

$$H(x[1:5]) = H(x[0:4]) + \Delta h(x[0]) + \Delta h(x[5]) \quad (5)$$

Using this method, the only thing to do is just to count how many times do the slide-in-and-out elements appear within the sliding time window, and the worst time complexity is $O(2*w)$. Then we calculate the variation of their information contents to update the entropy value, thus getting the generalized (5) with the following hypothesis.

Given an n-length sequence S and a preset sliding window width w ($w < n$), let $H(x)$ be the entropy value of the x^{th} w-length subsequence of S (marked as S_x), and $h_x(e)$ be the information content of an element e in S_x. The recurrence formula of entropy writes:

$$H(x) = \begin{cases} \sum_{e \in S_1} h(e) = -\sum p(e)\log p(e), & x = 1 \\ H(x-1) + \Delta h_x(e_{Out}) + \Delta h_x(e_{In}), & x > 1 \end{cases}$$
$$\Delta h_x(e) = h_x(e) - h_{x-1}(e), \quad 1 \le x \le n - w + 1 \quad (6)$$

b: VALUE COUNTER DICTIONARY AND COUNTER MATRIX

Another realization is to build a dictionary to record the count numbers of every distinct element at every sliding since the very beginning. Taking the previous example, we now record all the count numbers: 3 x_1s, 1 x_2 and 1 x_3 and record the result as a dictionary $d = \{x_1 : 3, x_2 : 1, x_3 : 1\}$ to make the result formatted. Now we slide the time window, recount the elements and append the new dictionary to the former one:

$$D = \begin{bmatrix} \{x_1 : 3, x_2 : 1, x_3 : 1\}, \\ \{x_1 : 2, x_2 : 1, x_3 : 1, x_4 : 1\} \end{bmatrix} \quad (7)$$

Because we preset the window width to be 5, it's simple to count the number of all elements. But the situation could be worse as the w goes larger. So we set another variable X to record the to-be-slide-out element. In the example, we set $X = x_1$, and after we slide we update the counting number of x_1 from 3 down to 2, and then check whether the newly slide-in element x_4 already exists.

$$D = \begin{bmatrix} d_1 = \{x_1 : 3, x_2 : 1, x_3 : 1\}, \\ d_2 = d_1 + \{x_1 : -1, x_4 : 1\} \end{bmatrix} \quad (8)$$

The dictionaries of the whole sliding window can be described in an array: $D = \begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_w \end{bmatrix}$. Furthermore, D can be transformed into a counting matrix of $C = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_w \end{bmatrix}$, where v_i is the value vector of d_i. In the example above, the matrix is $C = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 \end{bmatrix}$. The entropy calculation formula can now be written in a matrix format:

$$H[i] = -\frac{v_i}{w} \cdot \log \frac{{v_i}^T}{w}, \quad 1 \le i \le w \quad (9)$$

Or:

$$H[C] = -\frac{C}{w} \cdot \log \frac{C^T}{w} \quad (10)$$

2) JOINT ENTROPY

When a DDoS attack occurs, we usually see a sharp climbing-up of the entropy of source IP and Ports because of the huge amount of distributed attacking devices. But the situation may be opposite if the target devices are so weak that they can be easily taken down by a few attacking nodes, or the attack is launched by some old-but-still-effective DDoS tools such as ddosim [28]. On the other point of view, we will see a sharp decline of the entropy of destination IP until the number of attack packets reaches the width of the entropy sliding window, and this is a traditional and the most popular detection basis. But actually, the result will be interfered with by some indicators such as the target numbers, policies that motivate traffic to burst up, which is called "a benign flash crowd" in Zhao's research [19], etc.

To solve these problems, we combine both source and destination segments and apply joint entropy to the timestamp attached 5-tuples. We use the joint couple of (*s_IP*, *s_Port*) instead of the source IP and source port, and then make it joint with destination segments as our final detection basis. We mark (*s_IP*, *s_Port*) as s and (*d_IP*, *d_Port*) as d, and the final detection basis can be written as (s, d), so that the fake going-down of source IP can be eliminated.

C. DETECTION AND ALARM MODULE

1) BACKGROUND TRAFFIC FILTER

We can now easily find the beginning of attacks even by our eyes, but in practice, the terrible jitters can confuse the computers. It is necessary to eliminate the influences of background noise. Because the entropy values are continuous time series, we can use a Long Short-Term Memory (LSTM) model to predict the following entropy values. The LSTM [29], [30] is an artificial Recurrent Neural Network (RNN) architecture. Unlike standard Feedforward Neural Networks, LSTM has feedback connections and can take past inputs into the calculation, which makes it capable to deal with sequence prediction missions. Bloomberg Business Week wrote: "These powers make LSTM arguably the most commercial AI achievement, used for everything from predicting diseases to composing music" [31]. We preset lb as the number of the previous values that LSTM model looks back, send the first lb values as the input sequence, and get a predicted output value p. Then we repeat the process, pump p into the bottom of lb sequence and pop out the top value, send it to LSTM and get the second p value, and so on.

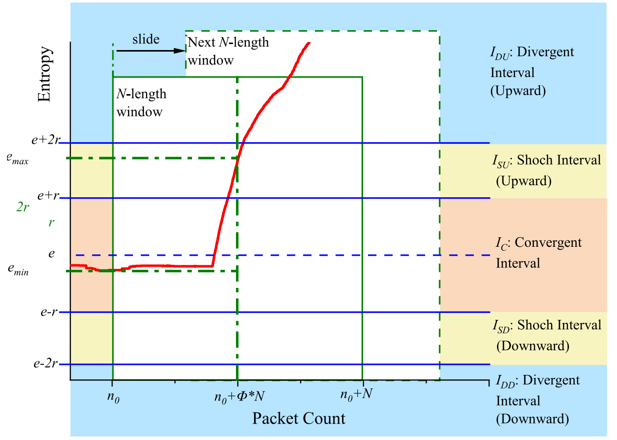


FIGURE 5. Interval division of QuinDC.

2) QUINTILE DEVIATION CHECK ALGORITHM

As we've got the pre-processed and real-time entropy values, the next thing to do is to make an immediate check and alarm for the suspicious packets. There are many deviation check algorithms: Quartile Deviation [32], [33], Generalized Extreme Studentized Deviate (GESD) [34], Linear Regression, Confidence Interval, etc. In this paper, we concern more about the sensitivity, and the values are nearly continuous, so we decide to redesign the Quartile Deviation algorithm. Quartile Deviation departs ascending sequence values into 4 equal-length subintervals and marks the 3 quartile points as Q1(25%), Q2(50%) and Q3(75%). The deviation value is Q3-Q1, which means 50% of the values are in [Q1, Q3], and the maximum deviation is Q3-Q1.

We made a great modification to the original Quartile Deviation algorithm, and name it as Quintile Deviation Check (abbr. QuinDC). Since we have to consider both directions of the curve trend, we extended the number of subintervals to 5 unequal-length subintervals and named them as Divergent Interval (Upward), Shock Interval (Upward), Convergent Interval, Shock Interval (Downward) and Divergent Interval (Downward) as shown below. The core parameters of QuinDC is the previous average value e and the previous radius value r (half of the value of which in traditional Quartile Deviation algorithm). We set e as the central basis of every test data, and determine the division points with $e, e \pm r$ *and* $e \pm 2r$. If a value falls in the interval of $[e - r, e + r]$, we mark it as 'convergent'; if it falls in the interval of $[e - 2r, e - r]$ or $[e + r, e + 2r]$, we mark it as 'shock'; and if it is larger than $e + 2r$ or less than $e - 2r$, then we mark it as 'divergent'. Those divergent values are what we concern about, and we can then turn to research what kind of data shows abnormal behavior. We introduce 3 new parameters, N, k and ϕ to solve this problem.

As is shown in Fig. 5, we apply an N-length sliding window here and divide the N-length sequence into k-length groups in the pattern shown in Fig. 6. Generally, N cannot be divided by k, so we just get rid of the beginning $N\%k$ values. Because the sequence or array is checked with a

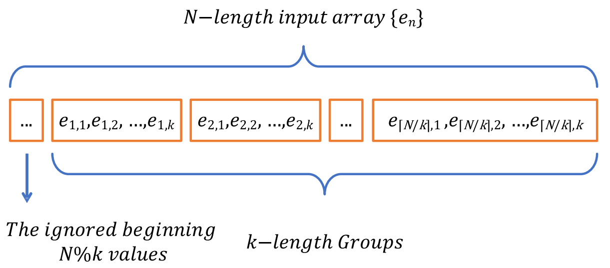


FIGURE 6. The group division diagram of QuinDC.

sliding window, this operation won't cause a bad affect on the result.

Finally, we set ϕ as the threshold of the alarm trigger, and summarize the following two alarm principles:

1) Monotonicity: The average value of all groups must be monotonic.

2) Divergency: ϕ of the values in the sliding window should be in the divergent intervals determined by the first ϕ of the sliding window in the previous step. The system will send out an alarm as soon as the 2 principles are both met.

V. SIMULATION AND EVALUATION

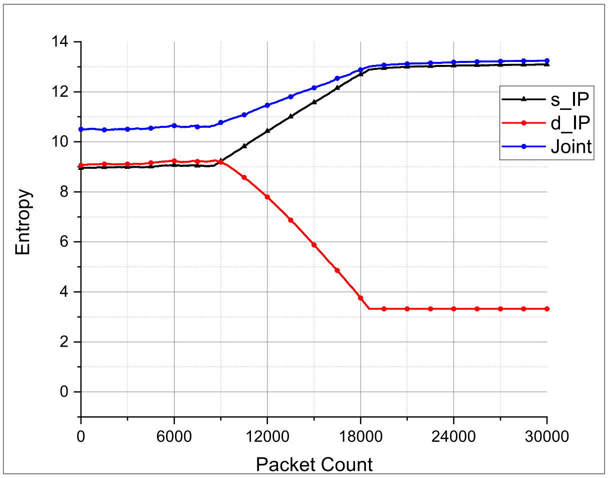
A. DATASETS SELECTION

In this paper, we use 3 public datasets and a generated dataset for testing. The public datasets are the 1999 DARPA Intrusion Detection Evaluation Data Set [35]–[37], 2009 DARPA DDOS Dataset [38], and the UNB CIC DDoS 2019 Evaluation Dataset [39]. The generated dataset is an emulation SDN-environment DDoS attack traffic generated by IXIA traffic simulator [40]. The 1999 & 2009 DARPA datasets are classical in DDoS evaluation. The CIC DDoS 2019 dataset is the newest dataset that including more modern attacking methods than the previous two datasets. The IXIA dataset is used as a supplement for testing the real-time performance in the SDN environment.

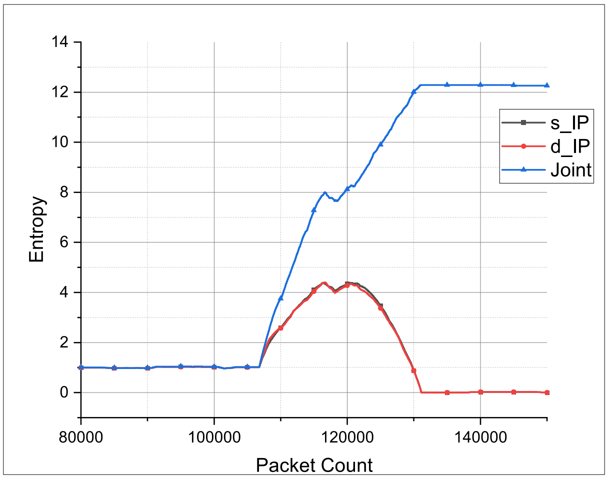
The DARPA datasets mainly provide TCP SYN Flood attack traffic, while in the CIC dataset, there are much more types of DDoS traffic: Network Time Protocol (NTP), Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), MSSQL, NetBIOS, Simple Network Management Protocol (SNMP), Simple Service Discovery Protocol (SSDP), User Datagram Protocol (UDP), UDP-Lag, WebDDoS, SYN, Trivial File Transfer Protocol (TFTP), etc. The IXIA BreakingPoint network tester simulates different DDoS attack scenarios and sends out the traffic packets. In the dataset, the types of DDoS attacks consist of generated simple attacks (such as SYN Flood, UDP Flood, and HTTP Flood) and simulated toolkit attacks (such as TCP and UDP attacks with different frequencies, background traffics or traffic-peak values).

B. SIMULATION OF RTVD DETECTION

Traditionally, we will see the entropy curves of source IP and destination IP in a volumetric DDoS like the black and



(a) Entropy Curves of most DDoS.



(b) Entropy Curves in DARPA 09 DDoS.

FIGURE 7. Joint entropy curves.

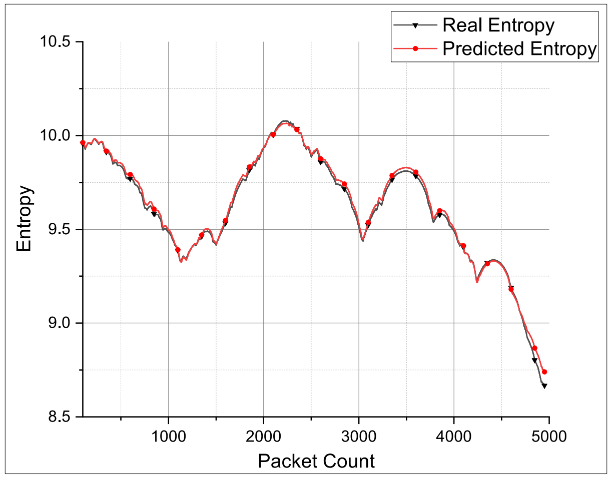


FIGURE 8. Background entropy filter using LSTM.

red curves shown in Fig. 7a, which describe the entropy changes from the occurrence of the attack to its saturation status. But if there are some indicators of specific attacking methods or legal flash crowd etc., like the black and red curves shown in Fig. 7b, it can be difficult to tell when to use the source IP or the destination IP for detection. But after we apply the joint entropy, as the blue curves shown in Fig. 7a and Fig. 7b, the situations become unified. So far, the data preparation has been completed.

Now we realize the IoT background packet filter. Fig. 8 shows the real entropy data and its LSTM-predicted data. We can see irregular but legal (for these entropy comes from benign background traffic without any attack) serrations in the real blue entropy line and the highly-coincident predicted red line. We let the LSTM prediction model continuously predict the future entropy with a benign input. Then, we still use a sliding time window, pump the predicted value into the input sequence, then pop out the oldest value. Therefore, the prediction keeps going on.

Since we've got the prediction entropy values of the background traffic, we can make a subtraction to eliminate the background noise, thus cutting down the false alarm rate.

C. EVALUATION

We've already been able to receive the traffic from an IoT network and calculate the real-time entropy values of that traffic. The only thing to do is to use QuinDC to find when the attacks occur. Before the final detection, we first made several tests to decide the optimal parameters N, k, ϕ in QuinDC. We select the traditional Receiver Operating Characteristic (RoC) indices of False Positive Rate (FPR) and False Positive Rate (FPR) to evaluate the QuinDC performance under different parameters.

$$\begin{cases} TPR = \dfrac{TP}{TP + FN} \\ FPR = \dfrac{FP}{FP + TN} \end{cases} \tag{11}$$

Generally, the TPR will raise together with the FPR from both 0% to near both 100%, and we hope to find the point where the TPR is as high as possible, and the FPR is as low as possible in the meanwhile. That is to say, what we want is the certain combination of (N, k, ϕ) that makes the point of (TPR, FPR) closest to the top-left point in the coordinates. In Fig. 9, there are three RoC curves represent IXIA SYN Flood, CIC 2019 DDoS Dataset and DARPA 1999 DDoS Dataset. Since We succeeded in detecting all the DDoS attacks listed in the dataset providers' attacking-timeline files, so we turn to evaluate the TPR and FPR with how many 'attacking packets' will QuinDC identify or omit. According to the result shown in Fig. 9, the best parameter combination is: $N = 500$, $k = 100$, $\phi = 0.4$, which means the width of a QuinDC sliding window is 500 packets, the group size is 100 packets, and at least 40% of the values in a sliding window should fall in the I_{DU} which is determined by the first 40% values in the previous sliding window.

TABLE 2. TFOR of different DDoS attacks using QuinDC.

Simulation	Timestamp of Attack Started	Timestamp of Attack Detected	Alarm Delay (second)	TFOR(%)
SYN Flood with no Background Traffic	1558359825.41888	1558359825.42462	0.00574	0.019133%
SYN Flood with Single Background Traffic	1558448841.30895	1558448841.31410	0.00515	0.017167%
SYN Flood with Mixed Background Traffic	1558364138.11221	1558364138.11660	0.00439	0.014633%
DNS Amplification	1558519298.38008	1558519298.38314	0.00306	0.010200%
LOIC HTTP Flood with Background Traffic	1558445608.31978	1558445608.32563	0.00585	0.019499%
LOIC HTTP Flood	1558445361.56058	1558445361.58692	0.02634	0.087800%
LOIC TCP UDP Mixed Attack	1558518501.53957	1558518501.61041	0.07084	0.236133%
LOIC TCP Flood	1558360113.90172	1558360113.90173	0.00001	0.000033%
DARPA 1999 Neptune TCP SYN Flood	921168256.32867	921168256.32867	0.00000	0.000000%
DARPA 1999 Neptune TCP SYN Flood	921255612.09387	921255612.13946	0.04559	0.101313%
DARPA 2009 DDoS TCP SYN Flood	1257425451.46404	1257425451.58030	0.11626	0.258356%
CIC 2019 DNS Amplification	1541249612.64374	1541249613.36564	0.72190	1.604222%
CIC 2019 LDAP Amplification	1541251314.31492	1541251314.34196	0.02704	0.060089%
CIC 2019 NetBios Amplification	1541253013.39668	1541253013.42064	0.02396	0.053244%
CIC 2019 SSDP Amplification	1541255357.76567	1541255357.78082	0.01515	0.033666%
CIC 2019 Web DDoS	1541258171.78800	1541258171.85230	0.06430	0.142888%
CIC 2019 SYN Flood	1541258870.05899	1541258871.06661	1.00762	2.239156%
CIC 2019 TFTP	1541259206.51490	1541259207.10361	0.58871	1.308245%
Average			0.015172	0.344765%

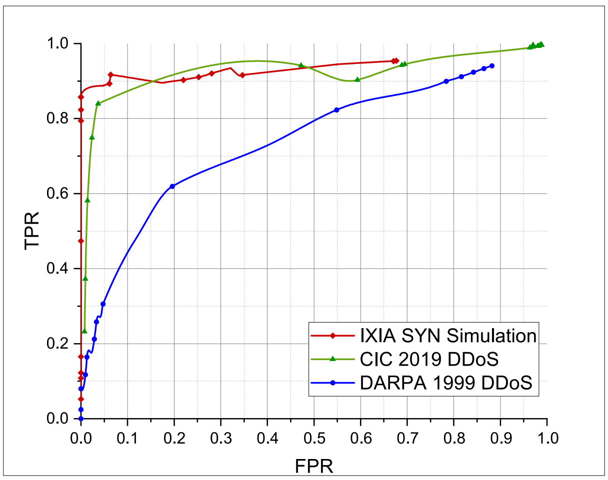


FIGURE 9. The RoC curves of datasets under different N, k, ϕ.

The TPR of modern DDoS detection is 100%, and the TPR of attacking packets detection is over 90%, with a low FPR of less than 3%. Furthermore, we redesigned another indicator of Temporal False Omission Rate (TFOR) (10) to evaluate the time performance of response. Traditionally, researchers will use a False Omission Rate (FOR) to evaluate the rate of omitted positive samples.

$$FOR = \frac{FN}{FN + TN} \tag{12}$$

However, in this paper, we do not care about the other positive samples but want to know how fast can we find the first true attack packet. So we modified (12) and added the temporal variables, and get the TFOR formula as (10).

$$TFOR = \frac{Alarm\ Delay}{Alarm\ Delay + Elapsed\ Benign\ Duration} \tag{13}$$

With this revision, we can evaluate a detection scheme to be sensitive and fast with a lower TFOR value. If the system recognizes an attack at its first attacking packet, then the value of *Alarm Delay* is 0, and the TFOR will be 0%. Otherwise, with the sliding window running, if the system fails to recognize any of the attacking packets, then the TFOR value will raise up and finally reach 100% because there are no packets before the attack and the value of *Elapsed Benign Duration* is 0. We simulate different types of DDoS attacks from the 4 datasets, and the TFOR results are shown in Fig.10 and listed in detail in Table 2. Most TFOR are less than 0.5% and have an average TFOR of 0.344765%. Also, the scheme can recognize all the volumetric DDoS attacks in an average delay of 0.015172 second.

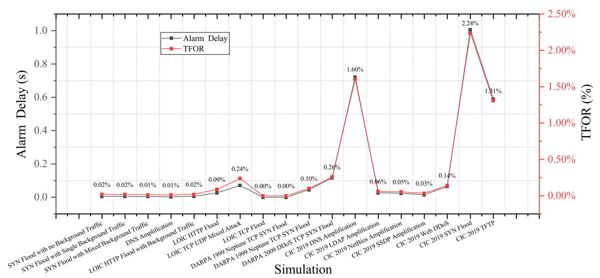


FIGURE 10. TFOR of different DDoS attacks using QuinDC.

VI. CONCLUSION

In this paper, we optimized the joint entropy calculation process and designed the algorithm of Quintile Deviation Check (QuinDC) to realize RTVD: a real-time volumetric detection scheme for DDoS in the Internet of Things. The techniques of sliding window, timestamp attached 5-tuple, and single directional packet filter make it possible to do entropy calculation in real-time. Furthermore, the QuinDC provides a low-latency and accurate performance, so that it can be applied in systems with the real-time requirement, such as intrusion defense systems in IoT environments. In future research, we will try to use LSTM or GRU for directly IoT DDoS detection, and focus on the Slow DDoS, as well as real-time volumetric DDoS taxonomy.

REFERENCES

[1] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, ''Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling,'' *Comput. Netw.*, vol. 121, pp. 25–36, Jul. 2017.

[2] J. Mirkovic and P. Reiher, ''A taxonomy of DDoS attack and DDoS defense mechanisms,'' *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, p. 39, Apr. 2004.

[3] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, ''Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty,'' *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8322–8335, Sep. 2019, doi: 10.1109/TVT.2019.2926732.

[4] Z. Zhou, B. Wang, M. Dong, and K. Ota, ''Secure and efficient vehicle-to-grid energy trading in Cyber physical systems: Integration of blockchain and edge computing,'' *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 43–57, Jan. 2020, doi: 10.1109/TSMC.2019.2896323.

[5] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, ''Fog-computing-enabled cognitive network function virtualization for an information-centric future Internet,'' *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 48–54, Jul. 2019.

[6] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan, ''FCSS: Fog-computing-based content-aware filtering for security services in information-centric social networks,'' *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 4, pp. 553–564, Oct. 2019.

[7] X. Lin, J. Li, J. Wu, H. Liang, and W. Yang, ''Making knowledge tradable in Edge-AI enabled IoT: A consortium blockchain-based efficient and incentive approach,'' *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6367–6378, Dec. 2019.

[8] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, and A. K. Bashir, ''Learning-based context-aware resource allocation for edge computing-empowered industrial IoT,'' *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2019.2963371.

[9] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ''DDoS in the IoT: Mirai and Other Botnets,'' *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[10] J. Li and Z. Xue, ''Distributed threat intelligence sharing system: A new sight of P2P Botnet detection,'' in *Proc. 2nd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, May 2019, pp. 1–6, doi: 10.1109/CAIS.2019.8769511.

[11] NetScout. *Low and Slow DDoS Attacks*. Accessed: Dec. 11, 2019. [Online]. Available: https://www.netscout.com/what-is-ddos/low-slow-attack

[12] Wikipedia. *SlowLoris*. Accessed: Dec. 11, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Slowloris_(computer_security)

[13] Nexusguard. (2018). *DDoS Threat Report 2018 Q3*. [Online]. Available: https://www.nexusguard.com/threat-report-q3-2018

[14] Nexusguard. (2019). *DDoS Threat Report 2019 Q3*. [Online]. Available: https://www.nexusguard.com/threat-report-q3-2019

[15] C.-H. Hsu, ''Fast-flux Bot detection in real time,'' in *Proc. RAID*, 2010, pp. 464–483.

[16] W. Qing-tao and S. Zhi-qing, ''Detecting DDoS attacks against web server using time series analysis,'' *Wuhan Univ. J. Nat. Sci.*, vol. 11, no. 1, pp. 175–180, Jan. 2006.

[17] D. Hu, P. Hong, and Y. Chen, ''FADM: DDoS flooding attack detection and mitigation system in software-defined networking,'' in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–7.

[18] F. Feng, ''Adaptive technique for real-time DDoS detection and defense using spark streaming,'' *J. Frontiers Comput. Sci. Technol.*, vol. 10, no. 5, pp. 601–611, 2016.

[19] Y. Zhao, W. Zhang, Y. Feng, and B. Yu, ''A classification detection algorithm based on joint entropy vector against application-layer DDoS attack,'' *Secur. Commun. Netw.*, vol. 2018, pp. 1–8, 2018.

[20] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, ''Statistical approaches to DDoS attack detection and response,'' in *Proc. DARPA Inf. Survivability Conf. Expo.*, Mar. 2004, pp. 303–314.

[21] W. Lee and D. Xiang, ''Information-theoretic measures for anomaly detection,'' in *Proc. IEEE Symp. Secur. Privacy.*, Nov. 2002, pp. 130–143.

[22] D. Virmani, S. Taneja, T. Chawla, R. Sharma, and R. Kumar, ''Entropy deviation method for analyzing network intrusion,'' in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Apr. 2016, pp. 515–519.

[23] M. Liu, Z. Xue, X. He, and J. Chen, ''Cyberthreat-intelligence information sharing: Enhancing collaborative security,'' *IEEE Consum. Electron. Mag.*, vol. 8, no. 3, pp. 17–22, May 2019.

[24] C. Mou, Z. Xue, and Y. Shi, ''BiLSTM and attention based anomaly detection of user shell commands,'' *Commun. Technol.*, vol. 52, no. 12, pp. 3016–3020, 2019.

[25] M. H. H. Khairi, ''A review of anomaly detection techniques and distributed denial of Service (DDoS) on software defined network (SDN),'' *Eng., Technol. Appl. Sci. Res.*, vol. 8, no. 2, pp. 2724–2730, 2018.

[26] *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*, document RFC 5766, 2010.

[27] D. Lee and N. Brownlee, ''Passive measurement of one-way and two-way flow lifetimes,'' *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, p. 17, Jul. 2007.

[28] A. Furtuna. *DDOSIM*. [Online]. Available: https://sourceforge.net/projects/ddosim/

[29] F. A. Gers, J. Schmidhuber, and F. Cummins, ''Learning to forget: Continual prediction with LSTM,'' *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, Oct. 2000.

[30] Y. Bengio, P. Simard, and P. Frasconi, ''Learning long-term dependencies with gradient descent is difficult,'' *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[31] A. Vance. (2018). *This Man Is the Godfather the AI Community Wants to Forget*. [Online]. Available: https://www.bloombergquint.com/businessweek/google-amazon-and-facebook-owe-j-rgen-schmidhuber-a-fortune

[32] X. Zhang and H. E. Zhongtang, ''Two-phase placement algorithm with energy efficiency optimization for virtual machins based on data center,'' *J. Comput. Appl.*, vol. 62, pp. 167–170, Jun. 2014.

[33] A. T. Buller and J. Mcmanus, ''The quartile-deviation/median-diameter relationships of glacial deposits,'' *Sedimentary Geol.*, vol. 10, no. 2, pp. 135–146, Oct. 1973.

[34] S. Srinu, A. K. Mishra, and S. Farooq, ''Improved GESD test for cooperative sensing over impaired cognitive radio networks,'' in *Proc. Annu. IEEE India Conf. (INDICON)*, Dec. 2014, pp. 1–5.

[35] P. R. Lippmann, K. R. Cunningham, J. D. Fried, I. Graf, R. K. Kendall, E. S. Webster, and A. M. Zissman, ''Results of the DARPA 1998 offline intrusion detection evaluation,'' presented at the RAID Conf., West Lafayette, IN, USA, Sep. 1999. [Online]. Available: https://archive.ll.mit.edu/ideval/files/RAID_1999a.pdf

[36] P. R. Lippmann and K. R. Cunningham, ''Using key-string selection and neural networks to reduce false alarms and detect new attacks with sniffer-based intrusion detection systems,'' presented at the RAID Conf., West Lafayette, IN, USA, Sep. 1999. [Online]. Available: https://archive.ll.mit.edu/ideval/files/RAID_1999b.pdf

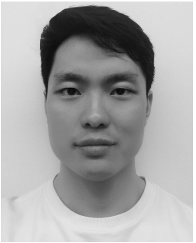
[37] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman, ''Evaluating intrusion detection systems without attacking your friends: The 1998 DARPA intrusion detection evaluation,'' in *Proc. SANS*, 1999, pp. 1–6, [Online]. Available: https://archive.ll.mit.edu/ideval/files/Evaluating_IDs_DARPA_1998.pdf

[38] M. Gharaibeh and C. Papadopoulos, ''DARPA 2009 intrusion detection dataset,'' Colorado State Univ., Tech. Rep., Aug. 2014. [Online]. Available: http://www.darpa2009.netsec.colostate.edu/

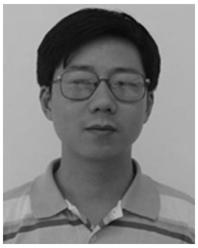
[39] I. Sharafaldin, ''Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy,'' in *Proc. IEEE ICCST*, OCT. 2019, pp. 1–8.

[40] Ixia. (2019). *Network Security Testing—BreakingPoint*. [Online]. Available: https://www.ixiacom.com/products/network-security-testing-breakingpoint

JIABIN LI is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. His research interests include intrusion detection systems, cyber threat intelligence, neural networks, and big data analytics.



MING LIU is currently a Lecturer with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. His research interests include cyber threat intelligence, intrusion detection systems, and scalable data analytics.



ZHI XUE is currently a Professor with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. His research interests include wireless network security, cloud security, cryptography, and cyber threat intelligence.

XIAOCHEN FAN (Student Member, IEEE) received the B.S. degree in computer science from the Beijing Institute of Technology, China, in 2013. He is currently pursuing the Ph.D. degree with the School of Electrical and Data Engineering, University of Technology Sydney (UTS), Australia. His research interests include mobile and pervasive computing, deep learning, and the IoT.

XIANGJIAN HE is currently a Professor and also the Director of the Computer Vision and Pattern Recognition Laboratory, Global Big Data Technologies Centre (GBDTC), and also a Co-Leader of the Network Security Research Team, Center for Real-time Information Networks (CRIN), University of Technology Sydney.