IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 9, 2019, accepted October 24, 2019, date of publication October 31, 2019, date of current version November 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950820

Identifying Application-Layer DDoS Attacks Based on Request Rhythm Matrices

HUAN LIN[1], SHOUFENG CAO[2], JIAYAN WU[1], ZHENZHONG CAO[3], AND FENGYU WANG[1]
[1]School of Software, Shandong University, Jinan 250101, China
[2]National Computer Network Emergency Response Technical Team Coordination Center of China, Beijing 100029, China
[3]School of Software, Qufu Normal University, Qufu 273165, China

Corresponding author: Fengyu Wang (wangfengyu@sdu.edu.cn)

This work was supported in part by the Key Special Projects of the 13th Five Year Plan of the Ministry of Science and Technology of China under Grant 2016YFB0801502, and in part by the Key Special Projects of the Ministry of Science and Technology of China under Grant 2016YFB0801304.

ABSTRACT Application-layer distributed denial of service (AL-DDoS) attacks are becoming critical threats to websites because the stealth of AL-DDoS attacks makes many intrusion prevention systems ineffective. To detect AL-DDoS attacks aimed at websites, we propose a novel statistical model called the *RM* (rhythm matrix). Although the original features from the network layer are adopted, the access trajectory, including requested objects and corresponding dwell-time values, can be abstracted and accumulated into an *RM*. With an *RM*, we can almost losslessly compress complex features into a simple structure and characterize the user access behavior. We detect AL-DDoS attacks according to the increase of the abnormality degree in the *RM* and further identify malicious hosts based on change-rate outliers. In the experiments, we simulate three modes of AL-DDoS attacks with the latest popular DDoS attack tools: LOIC and HOIC. The results show that our method can detect these simulated attacks and identify the malicious hosts accurately and efficiently. For an AL-DDoS detection method, the ability to distinguish flash crowds is indispensable. We also demonstrate the excellent performance of our approach in distinguishing flash crowds from AL-DDoS attacks with two reconstructed public datasets.

INDEX TERMS Network security, application-layer DDoS attack, anomaly detection, rhythm matrix, outliers.

I. INTRODUCTION

Over the past two decades, distributed denial of service (DDoS) attacks have been a continuous critical threat to the Internet. Denial of service (DOS) attacks have been known to the network research community since the early 1980s. The first Distributed DoS (DDoS) attack incident was reported in the summer of 1999 and most of the DoS attacks since then have been distributed in nature [1]. Many tools are available to easily perpetrate DDoS attacks, and many cyberspace crimes are closely related to DDoS attacks. However, the attack modes of DDoS attacks have changed in the recent years. While most traditional attacks are still active, more application-layer traffic is emerging, such as HTTP, HTTPS and DNS queries. We adopt the taxonomy in [2] and call these attacks *application-layer DDoS (AL-DDoS)*.

Much research has been done on DDoS attack and its defense [1], [3]–[6]. The recent DDoS Threat Report [7] shows that the majority of DDoS attacks are short in duration, and high-volume, high-rate DDoS attacks are on the upswing. To thwart a DDoS attack, the detection of the events must be completed during the manifestation phase, in which the attack develops and finally compromises the availability of a legitimate service. Furthermore, the malicious hosts also need to be identified in order to mitigate the attack. Our objective in this paper is to develop a novel AL-DDoS detection method that meets these requirements.

In order to defend against AL-DDoS attacks effectively, we draw on the experiences of [8], which extracted rhythm patterns to represent the characteristics of music clips. When visiting a website, users open and browse web pages repeatly, which also has some rhythm patterns, both in content and dwell time. To catch these rhythm patterns, we adopt the *packet size* and the *interarrival time* of consecutive HTTP-request packets in a flow as our original features. After a series of transformation, the relative relations are extracted from feature sequences to construct the *RM* (rhythm matrix). For website users, the *RM* can characterizes the distribution of their access trajectory fragments, including the order of visiting pages and the time spent on each page. It is difficult

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

for an attacker to mimic these characteristics of legitimate access. On the contrary, mass similar access controlled by one attacking master will markedly change some elements in the *RM*. So we examine the change-rate abnormality in the *RM* to detect AL-DDoS attacks, and furhter identify the malicious hosts according to their droppoints in the *RM*. Our method provides a new perspective for both characterizing group behavior and identifying AL-DDoS attacks. Analysis and experiments show that our method performs well in terms of both timeliness and accuracy.

The main contributions of this paper are listed as follows:

- We define a noval abstract model, named *RM* (Rhythm Matrix). For one website, an *RM* can depict the distribution of user access trajectories, which is a whole new perspective. Statistics show that the *RM* can characterize the legitimate access and is sensitive to the change of group behavior.
- We propose an approach to detect AL-DDoS attacks. Based on the fact that the distribution of droppoints in *RM*s remains relatively stable under legitimate access, we count the variation degree of droppoints distribution, and detect AL-DDoS attacks from its remarkble growth.
- We also propose an approach to further identify malicious hosts. After an AL-DDoS attack is detected, we find out the change-rate outliers from the subsequent *RM*s, and track the associations between hosts' droppoints and these outliers. If a large percentage of it's droppoints fall on the outliers, one host can be determined as a malicious host.
- We simulate three modes of AL-DDoS attacks with popular DDoS attack tools, and evaluate the accuracy and timeliness of our detection approach. We also verify the effect of malicious-hosts recognition with these simulated datasets.
- We reconstructed two public flash crowd datasets and demonstrate that an *RM* would not bothered by the drastic changes in legitimate traffic. This advantage ensures that our detection method can effectively distinguish flash crowds from AL-DDoS attacks.

The rest of this paper is organized as follows. In Section II, we present related work. Section III presents the construction of the *RM*s and our detection approach. In Section IV, the experiments and analyses are described. Finally, Section V discusses some open issues, and Section VI concludes this paper.

II. RELATED WORK

Generally, the word "rhythm" is used to refer to all the temporal aspects of a musical work. In the field of music signal processing, rhythm has been extracted as a feature for detecting music mood or classifying the genre of the music content [8]–[10]. Based on the rhythmic characteristics of network flow, we propose *RM* and use it to identify AL-DDoS attacks.

Researchers have studied DDoS attacks from different perspectives and proposed many approaches to detect and mitigate them [1], [3]–[6], [11]–[18]. Recent research interests have been more concentrated on AL-DDoS attack [2], [7], [19]–[30] for its harmfulness and concealment. In this section, we focus on works that are similar to our approach in terms of feature selection or model construction.

Some studies adopted *packet size* or *interarrival time* as features in DDoS detection. Li [31] employed the packet size discrete series to study how the Hurst parameter H of traffic varies under DDoS flood attacks. A method presented in [32] made use of the mean packet interarrival time to construct a fuzzy estimator to detect DDoS attacks. Zhou *et al.* [33] used the expectation of packet size to distinguish two typical low-rate DDoS attacks, constant attack and pulsing attack, from legitimate traffic. Although *packet size* and *interarrival time* are network-layer properties, a wealth of application-layer information is implied by them. Our method utilizes the rhythm matrix to depict the relative relations of consecutive packets and to capture the application-layer information.

Some related publications defined a matrix to characterize the patterns of traffic in DDoS detection. One paper [34] adopted a covariance matrix for the detection of AL-DDOS attacks, but implementing such a matrix for each user on a popular website has a high cost. Xie and Yu [35] constructed an *access matrix* to describe document popularity by extending the traditional definition, i.e., the *request hit rate*, and then used a stochastic process to model the variety of document popularity. Lee *et al.* [36] built a *traffic matrix* by extracting source IP addresses from an inbound traffic stream and used the variance computed from the *traffic matrix* to detect DDoS attacks. Another study [37] designed a Bayesian network structure to model the causal relationships between network traffic and constructed a *traffic matrix*. Luo *et al.* [38] constructed a behavior feature matrix based on nine user behavior features, and outliers from user browsing behavior patterns were used to recognize normal users and attackers. Compared with the *access matrix*, our rhythm matrix can also capture the spatial-temporal patterns of access behaviors with lower complexity. The rhythm matrix is constructed at the packet level, as is the *traffic matrix*, but the features employed by the rhythm matrix can depict a much richer application-layer information.

III. DESCRIPTION OF THE PROPOSED METHOD

In this paper, we focus on AL-DDoS attacks targeting websites and consider HTTP-flooding attacks as the principal experimental subject. Our main objective is to build an accurate model of legitimate traffic, i.e., the target class, then try to detect whether the subsequent traffic is adulterated with attack flows by comparing the behavior to that of the target class.

A. REQUEST RHYTHM MATRIX

Fig.1 intuitively depicts the construction scheme of the rhythm matrix. First, we extract the flows from inbounding traffic, and denote the aggregation of flows as $\mathcal{F}$. Then,

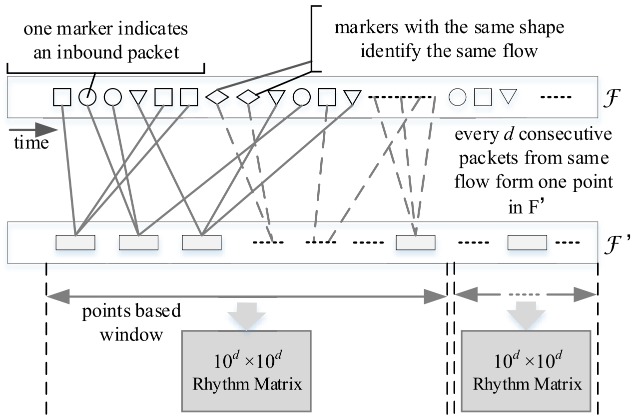


FIGURE 1. Construction scheme of the rhythm matrix.

we discrete and encapsulate the feature sequences of these flows, and $\mathcal{F}$ is transformed into $\mathcal{F}'$. Finally, we set a window on $\mathcal{F}'$ and produce the rhythm matrix. Next, we describe the construction process in greater detail.

1) FEATURE EXTRACTION

Based on the analogy between music and network flow, we consider the *packet size* and *interarrival time* between consecutive packets in a flow as our features to create the *RM*. The packet size implies the location of the requested object, so the packet size sequence depicts the access trajectory in the spatial dimension. The interarrival time implies the dwell time on one object, so the interarrival time sequence depicts the access trajectory in the temporal dimension. These appearently network layer features contain a wealth of application-layer information.

Our method uses *RM*s to profile the access behaviors. An *RM* is constructed from the inbound traffic flows. We define flow F as the unidirectional, ordered sequence of IP packets produced either by one client towards the server or by the server towards the client during a period of time. Our detection system only pays attention to the flows from client to server, and we denote the aggregation of flows from all clients towards the server in one period as $\mathcal{F}$. Each flow F can be represented as an ordered sequence of pairs $(s, \Delta t)$, where s represents the packet size and Δt represents the interarrival time between the current packet and the previous one.

Some incoming packets should be filtered out from $\mathcal{F}$. The packets that carry empty TCP payload are not considered, except for those for which SYN/FIN TCP-flags are set. The statistical behavior of TCP packets of zero payload size does not depend on the behavior of the users but is a function of the TCP/IP stack of the end hosts and the status of the network. So empty TCP packets are of little use in the characterization of application-layer traffic. A side benefit of ignoring zero-payload-size packets is that it is helpful to reduce the consumption of computational resources. In practice, a simple BPF expression, as below, can help us to filter the useless packets.

- `BPF expression example:`
  (dst host *host* and tcp port *port*) and \
  ((tcp[tcpflags] & (tcp-syn | tcp-fin) != 0) or \
  ((ip[2:2] − (ip[0]&0xf)<<2 − (tcp[12]&0xf0)>>2) != 0))

When matched packets arrive at the detecting module, we group them into different inbound flows by maintaining a hash table. With the hash value of the source IP address, one packet is finally mapped and appended to a list of $(s, \Delta t)$. The collision problem is negligible in this hash table because the window restricts the number of points in $\mathcal{F}'$ and, in turn, the number of packets for one rhythm matrix.

Assuming there are N flows during one window, we set F to be one of these flows with size of L and denote it as follows:

$$F = \begin{pmatrix} s(1) & s(2) & \dots & s(L) \\ \Delta t(1) & \Delta t(2) & \dots & \Delta t(L) \end{pmatrix} \tag{1}$$

Then $\mathcal{F}$ consists of these N flows:

$$\mathcal{F} = \bigcup_{i=0}^{N-1} F^i, \ where F^i \ denotes \ the \ i\_th \ flow \ of \ \mathcal{F} \tag{2}$$

2) DATA TRANSFORMATION

The packet size s is approximately equal to the sum of the length of the encapsulation headers and the payload. The encapsulation headers always hold a fixed length for a client-server pair, and the payload is the main variable for different application protocols. For HTTP requests, the size of payload is the sum of the lengths of the URL and HTTP protocol extra messages. For a given website, the only variation is the extra messages, which depend on some factors with a small variable quantity. Thus, the packet size s corresponds to the length of the URL.

The interarrival time between consecutive packets is affected by many factors, such as the distance of the peers, network congestion, and user access behavior. We argue that, within a short period, the relative interarrival times of HTTP requests are mainly influenced by the page components and user browsing duration. We adopt a logarithmic transformation to truncate the range of possible values to a limited interval. Commonly, the grain of the time-stamp recorded by a network traffic capture tool is fixed. Hence, the observed variable $\log_{10}[\Delta t]$ is discrete and limited.

We denote the cardinality of observable values for the two variables as $B(s)$ and $B(\Delta t)$, respectively. In reality, $B(s)$ and $B(\Delta t)$ always hold great values. Since our detection model must run in a real-time environment, the rhythm matrix should be abstracted into a reasonable space size to construct it fast enough, while at the same time covering as many incoming packets as possible. Therefore, we quantize the variable $\log_{10}[\Delta t]$ with step size of one and s with a step size of $B(s)/10$, where $B(s) = \sup(s) - \inf(s)$. The symbols "sup" and "inf" indicate the supremum and infimum limits of the variables immediately following with them. According to these rules, we obtain the formula (3). In a practical process, we can modify the supremum and infimum limits of s

and Δt to different values based on the actual situation. For simplicity, we set $\lfloor \log_{10}[\sup(\Delta t)] - \log_{10}[\inf(\Delta t)] \rfloor$ to be the constant 9. As mentioned above, s is quantized with a step size of $B(s)/10$. Thus, both s' and $\Delta t'$ are discrete integers in [0, 9].

$$(s', \ \Delta t') = \left(\lfloor \frac{s - \inf(s)}{(B(s)/10)} \rfloor, \ \lfloor \log_{10}[\Delta t] - \log_{10}[\inf(\Delta t)] \rfloor \right) \tag{3}$$

To encapsulate the HTTP request rhythm, we integrate d pairs $(s', \Delta t')$ into one pair and transform the flow into:

$$F' = \begin{pmatrix} x(1) \ x(2) \ \ldots \ x(L') \\ y(1) \ y(2) \ \ldots \ y(L') \end{pmatrix} \tag{4}$$

in which

$$\begin{pmatrix} x(j) \\ y(j) \end{pmatrix} = \begin{pmatrix} \sum_{k=0}^{d-1} (s'_{i+k} \times 10^k) \\ \sum_{k=0}^{d-1} (\Delta t'_{i+k} \times 10^k) \end{pmatrix} \tag{5}$$

where $L' = \lfloor \frac{L}{d} \rfloor$, $i = dj + 1$ and $0 \leq j < L'$. With variable d, the rhythm matrix possesses a flexible resolution ratio. If the length of flow F is not an integer multiple of d, the remaining packets will fit into the next window. Once the array of $(s, \Delta t)$ for each flow is filled up with $d + 1$ pairs, the transformation in (5) is performed exactly once. Then, the first d elements of the array are cleared and the tail is moved to the head to calculate Δt of the next incoming packet. This procedure is executed repeatedly during the module runtime. However, we should also pay attention to the flows with lengths less than d. These flows are never transformed to the format of F', so we must set a session timeout value and release the buffers of the useless flows periodically.

Now, flow F is transformed into F'. Accordingly, $\mathcal{F}$ is transformed into $\mathcal{F}'$, which is an aggregation of F':

$$\mathcal{F}' = \bigcup_{i=0}^{N-1} F'^{i}; \tag{6}$$

where N is the number of flows in $\mathcal{F}'$ and F'^{i} is the transformed sequence of the *i-th* flow in $\mathcal{F}$.

3) RHYTHM MATRIX CONSTRUCTION

The construction of an *RM* is shown in Fig. 2. We use x and y as the indexes of the row and column of the matrix respectively. Clearly, the values of x and y in formula (4) are integers in $[0, 10^d)$, so the scale of the rhythm matrix is $10^d \times 10^d$. In terms of space complexity, we need to limit the value of d. If the data of an *RM* can all reside in memory, the efficiency of the algorithm can be effectively improved. In general, the recommended value range of d is between 2 and 4. Typically, we set d as 3 in the later experiments. The element (x,y) in the *RM* records the times that (x,y) is generated from $\mathcal{F}'$. We present the definition of the rhythm matrix below.

Definition 1: **Rhythm Matrix** Let $\mathcal{F}$ be the packet sequence of a traffic trace, the transformed sequence of which is $\mathcal{F}'$. The *RM* is a $10^d \times 10^d$ matrix, with R_{xy} denoting the element at location (x,y) of the matrix. If the value of each element R_{xy} is equal to the frequency of (x,y) in $\mathcal{F}'$, we call the *RM* the rhythm matrix of this trace.

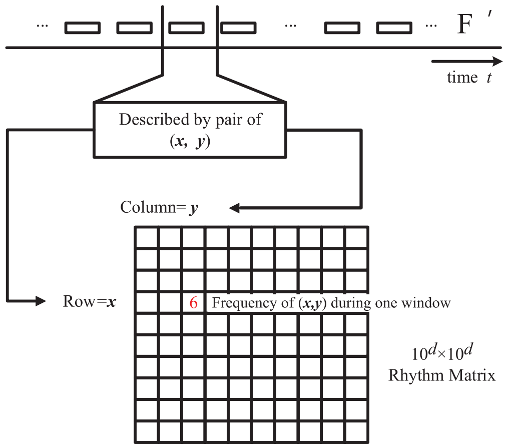


FIGURE 2. Construction of the rhythm matrix from the transformed sequence.

We did not set the window on $\mathcal{F}$ but on $\mathcal{F}'$, and the size of the window is based on the number of droppoints. On the one hand, a time-based window on $\mathcal{F}$ would lead to extra computational overhead in the absence of DDoS attacks and infrequent access. On the other hand, a window size based on the number of packets would generate dissimilar *RMs* in the case of flash crowd traffic. As an extreme example of adopting the window size based on the number of packets, when the number of concurrent sessions is equal to the window size, no droppoints can be produced and the *RM* is filled with zero elements. Such an *RM* cannot represent any information about the traffic. In the following, we will denote the window size based on the number of droppoints as *PktW*.

To describe the construction more clearly, we illustrate the matrix generation process with an simple example. Suppose a flow F exists, from which a fragment of packet-size series s and interarrival time series Δt are extracted as follows:

$$\begin{pmatrix} \mathrm{s} \\ \Delta \mathrm{t} \end{pmatrix} = \begin{pmatrix} 40 & 186 & 332 & 186 & 518 \\ 0.000001 & 0.00001 & 0.1 & 1 & 10 \end{pmatrix}$$

We discretize above flow features with formula (3) and yield:

$$\begin{pmatrix} \mathrm{s}' \\ \Delta \mathrm{t}' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 1 & 3 \\ 0 & 1 & 5 & 6 & 7 \end{pmatrix}$$

Then this flow can be added into the rhythm matrix. We set d to 3 and calculate the droppoint coordinate for every three continuous packets with formula (5). For example, the x-coordinate for packets 1˜3 is $0 * 100 + 1 * 10 + 2 = 12$, and the y-coordinate is $0 * 100 + 1 * 10 + 5 = 15$. Thus, the droppoint coordinate of packets 1˜3 in the rhythm matrix is (12,15), and we add 1 to the element at this location. Other flows can be processed concurrently in the same way, until the number of droppoints reaches window size *PktW*.

	SUF	MUF	RUF	WEB-1	WEB-2	WEB-3	WEB-4	WEB-5	WEB-6	WEB-7
SUF	**29.63**	95.12	82.67	129.33	121.71	100.16	85.33	93.48	85.69	126.19
MUF	95.12	**22.47**	113.52	150.75	143.12	126.27	115.80	122.50	115.94	148.65
RUF	82.67	113.52	**27.68**	117.77	110.11	86.19	68.84	81.25	68.11	111.03
WEB-1	129.33	150.75	117.77	**49.65**	142.12	129.51	117.48	124.91	119.26	143.68
WEB-2	121.71	143.12	110.11	142.12	**25.88**	122.50	110.84	109.85	112.40	135.50
WEB-3	100.16	126.27	86.19	129.51	122.50	**25.32**	86.21	94.69	87.59	126.60
WEB-4	85.33	115.80	68.84	117.48	110.84	86.21	**28.05**	82.16	69.28	115.07
WEB-5	93.48	122.50	81.25	124.91	109.85	94.69	82.16	**29.70**	83.40	121.12
WEB-6	85.69	115.94	68.11	119.26	112.40	87.59	69.28	83.40	**42.47**	117.02
WEB-7	126.19	148.65	111.03	143.68	135.50	126.60	115.07	121.12	117.02	**33.30**

FIGURE 3. Distance matrix of 10 groups of *RM*s.

B. DESCRIPTING TRAFFIC WITH RHYTHM MATRICES

The key point of our model is that it can effectively sense the change of access trajectory distribution. In this section, we present how the *RM* is able to describe the characteristic of various types of traffic. Ten traces, including seven different sites' normal traffic and three DDoS attack modes' traffic,[1] were used to build the *RM*s. The WebDDoS attack modes are classified based on the taxonomy presented by [39] and [40], i.e., Single-URL Flood (SUF), Multi-URL Flood (MUF) and Random-URL Flood (RUF). We built a group of 4 *RM*s with the same *PktW* on each trace and computed the Euclidean distance between all pairs of *RM*s. Then, the mean Euclidean distance between the *RM*s in two groups composed the distance matrix, as shown in Fig.3.

Fig.3 provides at least 3 viewpoints about *RM*s. First, the *RM*s have a high similarity if they are built based on the same-source traffic but are dissimilar if built on different-source traffic. Second, the *RM*s built on DDoS traffic are dissimilar to those built on normal traffic, even traffic from different sites. Thus, the rhythm matrix has the ability to discriminate DDoS attack traffic from normal traffic through some reasonable measures. Finally, the modes of DDoS attack traffic can be classified by the *RM*s, although this is not a requirement for this paper. Later in the experiments, we will show *RM*'s perception ability to AL-DDoS attacks in a more comprehensive way.

C. DEFENDING AGAINST AL-DDoS ATTACKS

According to the previous analysis, we conclude that an *RM* can accurately characterize the access behaviors of website users. Under an AL-DDoS attack, many malicious hosts access a website via a similar mode simultaneously, which would lead some elements in the rhythm matrix to increase disproportionately. According to these abnormal elements, we can detect the DDoS attacks and identify the malicious hosts.

[1]DDoS attack traffic is captured by simulating on our campus website; normal traffic is extracted from the dataset http://wand.net/wits/waikato/8/

1) CHANGE RATES AND DDoS DETECTION

For a website, the relative relations among elements in the rhythm matrix remain roughly stable over time. Suppose we have sampled i RMs, $\vec{RM} = \{RM_1, RM_2, ..., RM_i\}$ for the normal traffic of one site. RM_i is the current rhythm matrix, and we select one previous rhythm matrix as its *base matrix*. Let $R_{x,y}$ be one element of RM_i, and let $R'_{x,y}$ be the corresponding element in the *base matrix*. Then, we can define the *change-rate* for $R_{x,y}$ as:

$$\alpha_{x,y} = \frac{R_{x,y}}{R'_{x,y}} \tag{7}$$

When $R'_{x,y}$ is 0, we set it to be 1. Normally, the values of change-rate in one rhythm matrix would be very similar.

However, for the aggregation of similar access behavior under an AL-DDoS attack, this balance in rhythm matrix can be broken and some of the elements can increase unusually. In practice, some accidental events could also cause a few elements to increase unusually; thus, AL-DDoS attacks should not be identified based solely on the appearance of abnormal growth elements. To eliminate the distraction of accidental abnormal growth, we introduce a new measure named the *abnormality degree*. For the elements whose change-rate is greater than 1 in RM_i, we suppose the average change-rate is $\bar{\alpha}$. We define the abnormality degree of element $R_{x,y}$ as $max(0, R_{x,y} - R'_{x,y} * \bar{\alpha})$. Then, the abnormality degree of RM_i is:

$$\gamma = \sum max(0, R_{x,y} - R'_{x,y} * \bar{\alpha}) \tag{8}$$

We can track the normal access traffic of one website and obtain the maximum value of γ, then set a threshold Γ slightly higher than γ. For subsequent access traffic, if the abnormality degree of an *RM* exceeds Γ, then the website is under AL-DDoS attack.

In general, we use RM_{i-2}, rather than RM_{i-1}, as the *base matrix* of RM_i. Because if an AL-DDoS attack starts in the second half of the window of RM_{i-1}, the attack can probably escape detection in RM_{i-1} for an insufficient abnormality degree. Then, using RM_{i-1} as the *base matrix* will weaken the abnormality degree of RM_i and result in a false negative. Selecting RM_{i-2} as the *base matrix* of RM_i can prevent this problem. If the DDoS attack is detected in one rhythm matrix, for instance, RM_i, then the *base matrix* of subsequent *RM*s should remain unchanged until the attack disappears.

2) OUTLIERS AND MALICIOUS HOST IDENTIFICATION

As previously mentioned, the similar access behavior of a vast number of malicious hosts results in abnormal growth elements, and vice versa, the majority droppoints of one malicious host would fall on abnormal growth elements. Thus, if the abnormal growth elements are found out, we can identify the malicious hosts.

By considering the change-rate outliers, we can locate the abnormal growth elements in RM_i. In statistics, an outlier is an observation point that is distant from other observations. The simplest way to identify outliers is with the quartile method, which flags observations based on the interquartile range. Suppose Q_1 and Q_3 are the lower and upper quartiles of the change-rates in an *RM*, respectively, then we could define an outlier of change-rates to be any observation outside the range:

$$[Q_1 - k(Q_3 - Q_1), Q_3 + k((Q_3 - Q_1)] \tag{9}$$

where $k = 1.5$ indicates an "outlier".

If an AL-DDoS attack is detected in RM_i, we track the droppoints of every host in RM_{i+1}. After identifying the outliers in RM_{i+1}, we can identify the malicious hosts by checking their droppoints. Suppose that for one host, p percent of its droppoints fall on outliers. If p exceeds a threshold Θ, then the host is considered to be malicious. The threshold Θ can be set experimentally. A low Θ can ensure a high malicious host identification rate; however, this would inevitably lead to false positives. Therefore, we need to find the optimal range of Θ. In practice, we can adopt a dynamic adjustment scheme: Θ is initialized with a high value, such as 80%, and once an AL-DDoS attack is detected, the value of Θ is gradually reduced until the target website under attack can function normally.

D. COMPLEXITY ANALYSIS

We now consider the computational cost of the discussed algorithm. The computational complexity of the generation of an *RM* is $O(n)$, where n is the number of droppoints in the *RM* and is no more than $PktW$. A hash table is used to record the list of packet lengths and interarrival times $(s, \Delta t)$, which is preallocated and does not increase, so the spatial complexity is $O(1)$. To generate the abnormality degree of each *RM* in the detecting phase, we need to calculate the change-rates and count them. The number of elements in an *RM* is constant, so the time complexity is $O(1)$. When we identify malicious hosts, we first perform outlier detection. The time complexity to detect outliers is related to the sorting algorithm: we use quick-sort with time complexity $O(n * log_2 n)$. After an attack is detected, we track every IP and calculate its percentage of droppoints falling on outliers. All the hosts must be recorded,

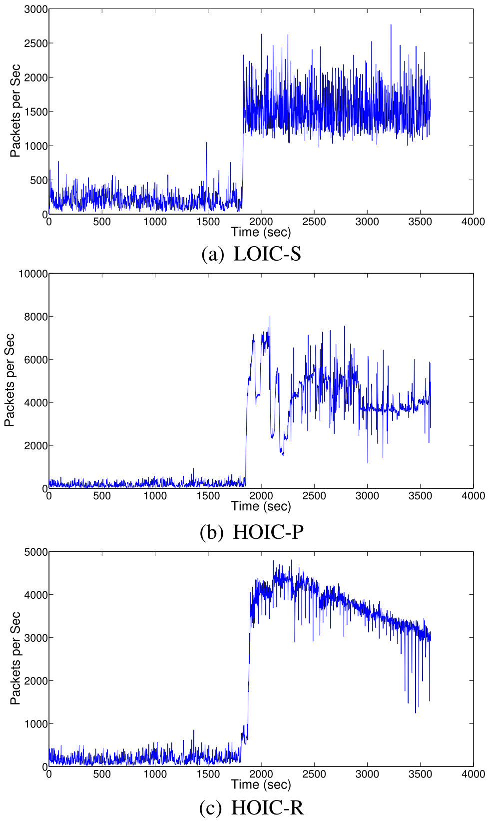


FIGURE 4. Profiles of the different AL-DDoS datasets.

so the space complexity is $O(n)$, where n is the number of hosts.

IV. EXPERIMENTS

A. DATASETS

1) DDOS DATASET

In this section, we employed two practical DDoS attack tools, HOIC [41] and LOIC [42], to generate our own DDoS attack datasets [12]. The primary data samples were generated by a small-scale simulation in which our campus website was the main protection target. We evaluated the effectiveness and timeliness of the proposed method.

The data were collected by mirroring the server's Ethernet port and capturing the inbound traffic on ports 80 and 443. The PF_RING ZC [43] was employed in the capture process to avoid packet dropping. We launched three attacks with different settings to generate three datasets. The traffic profiles are shown in Fig. 4, where 4(a) shows the *LOIC-S* of which the target is a single page without parsing the mainframe of the page, 4(b) is *HOIC-P* of which the target is one page with parsing of the mainframe of the page, while 4(c) is *HOIC-R* of which the target is randomly chosen from a list of pages and with parsing of the mainframe of each page. These three datasets correspond to the three modes introduced in Section III.

We recorded the start-time and the onset-time of each attack mode. The onset-time stands for the time when

TABLE 1. Duration of the manifestation phase.

Mode	ART-Duration(sec)	ECN-Duration(sec)
LOIC-S	300	912
HOIC-P	100	820
HOIC-R	120	206

the DDoS attack took effect, which we identified by two approaches. The first approach is artificial recognition, that is, continuously visiting and refreshing the page of the victim server on one PC connected in the same subdomain as the victim server. The time when the page no longer displayed was recorded as the onset-time (ART). The other approach extracted the time-stamp of the first packet with the ECN (Explicit Congestion Notification) TCP-flag as the onset-time (ECN). The ECN is an extension of the TCP/IP that allows end-to-end notification of network congestion without dropping packets. Therefore, we can consider the time-stamp of the first packet with the ECN setting coming out from the server as the DDoS onset-time. By subtracting the start-time from the onset-time, we can obtain the duration of the DDoS manifestation phase of each mode, as shown in Table 1.

The duration times in Table 1 show that legitimate users are influenced more immediately by DDoS attacks than are network devices because the servers always attempt to serve every incoming request and will not signal impending congestion until the limited resources are exhausted. However, an end-user can feel the network abnormality before that point.

2) FLASH CROWD DATASET

To demonstrate the performance of our proposed method in the case of flash crowds, we used the 1998 FIFA World Cup dataset [44] as a representation of flash crowds. This dataset represents a highly reliable flash crowd and has been widely used for recent high-quality publications, such as [35], [45]. To better reflect the real situation of the network, we chose only the requests to the servers in PLANO during two time periods that correspond to two knockout games, Romania versus Croatia (RC for short) and Argentina versus England (AE for short), as shown in Fig. 5.

We reconstructed the data to meet the requirements of our experiments. First, we set the session-timeout value to 100 seconds so that the maximum number of active sessions would not be too great to exhaust the resources of the servers, as discussed in [46]. Second, a log item with the SYN flag was inserted in the front of each session while an item with the FIN flag was set at the end. Finally, the interarrival times between consecutive log items of the same session were generated from a log-normal distribution with parameters μ and σ taken from paper [47]. To more clearly represent the actual situation, we calculated the packet size for each item on the basis of the URL length.

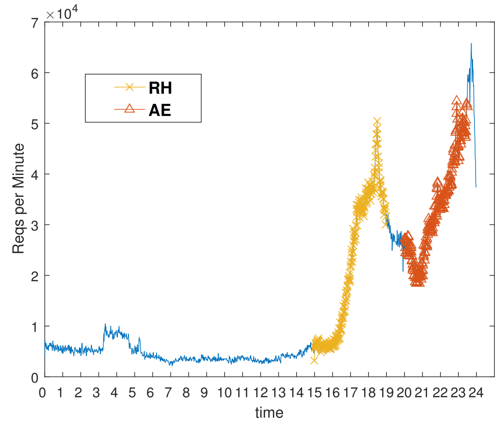


FIGURE 5. Profiles of the flash crowds dataset.

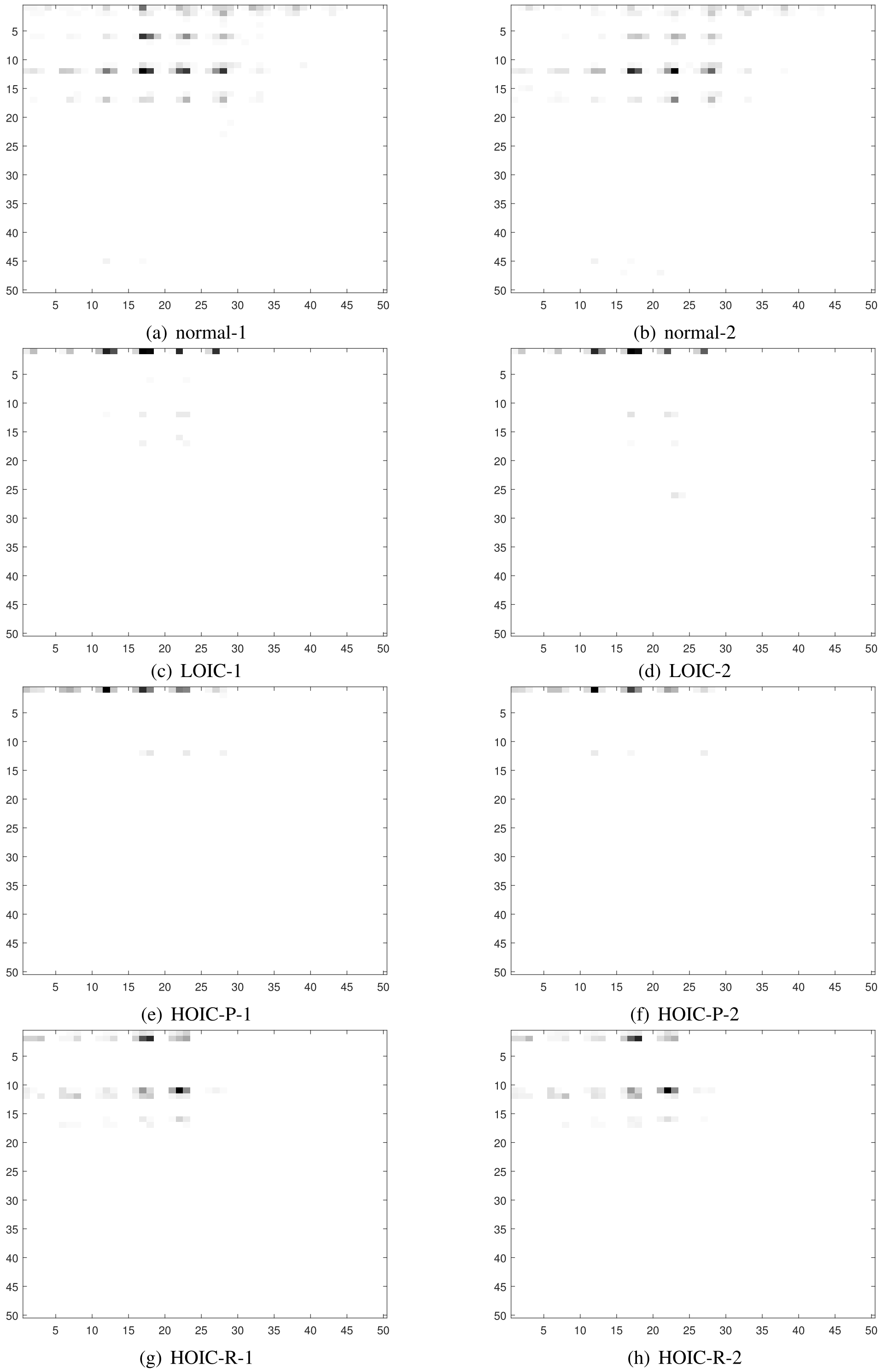
B. EMPIRICAL RESULTS

In this section, we perform evaluations of our proposed method using the aforementioned datasets. Since our algorithm only considers the relative packet size, for simplicity, we use the size of the Ethernet frame instead and limit the packet size s in the range [40,1500] in all the experiments. Although the interarrival times between consecutive packets are not generally limited, we restrict them so that the minimum value is fixed to 10^{-6} s and the maximum is fixed to 10^3 s. Since we apply a log transformation, the variable $\log_{10}(\Delta t)$ ranges in $[-6,3]$, which satisfies the condition $\lfloor \log_{10}[\sup(\Delta t)] - \log_{10}[\inf(\Delta t)] \rfloor \equiv 9$, as mentioned in Section III. For simplicity, we set the parameter $d = 3$. In other words, we adopt the 1000×1000 rhythm matrices for the experiments.

1) PERFORMANCE OF DETECTING AL-DDOS

Our AL-DDoS detecting method is based on two assumptions, one is that the *RM*s of traffic targeted at the same website are similar, and the other is that AL-DDoS attacks would cause obvious changes in the *RM*s. We set *Pktw* to be 2000 and chose two *RM*s at random from normal traffic, *LOIC-S*, *HOIC-P* and *HOIC-P*, respectively. The 2000 drop-points are sparsely distributed in 1000*1000 matrix, which is difficult to be observed by the unaided eye. So we compacted these 1000×1000 matrices into 50×50 matrices through accumulating adjacent elements. The gray-scale maps of these 50×50 matrices are shown in Fig 6. As we can see, the differences between *RM*s from normal traffic, *LOIC-S*, *HOIC-P* and *HOIC-P* are obvious, and the matrices in the same group are very similar. These statistical results strongly support our assumptions.

To evaluate our detection method, we calculate the true positive rate (*TPR*) and false positive rate (*FPR*). A larger *Pktw* could delay the detection time, but a small window size could also weaken the statistical characteristics of the rhythm matrix. Taking these factors and the dataset size into account, we set *Pktw* in the range 800~2200 with step 200. Before attack detection, the threshold abnormality degree, i.e., Γ,



(a) normal-1 (b) normal-2 (c) LOIC-1 (d) LOIC-2 (e) HOIC-P-1 (f) HOIC-P-2 (g) HOIC-R-1 (h) HOIC-R-2

FIGURE 6. Gray-scale map of *RM*s for normal and three types of AL-DDoS.

must be set. For each *Pktw*, we determine Γ by raising the maximum value of the abnormality degree on normal data by 5%. We replay one hour of normal traffic data and three different types of attack traffic to the detection system. From the normal and one type of DDoS traffic, we gather *P* and *Q* *RM*s respectively, which denoted by RM_{norm} and RM_{DDoS}. Then, we perform detection and calculate the *TPR* and *FPR* via formula (10), as shown at the bottom of the next page.

As shown in Table 2, for three types of AL-DDoS attacks, our method performed well in terms of both *TPR* and *FPR*.

TABLE 2. Performance in detecting DDoS attacks.

PktW	TPR			FPR		
	LOIC-S	HOIC-P	HOIC-R	LOIC-S	HOIC-P	HOIC-R
800	1.00	1.00	1.00	0.00	0.00	0.01
1000	1.00	1.00	1.00	0.00	0.01	0.00
1200	0.99	0.99	1.00	0.00	0.01	0.01
1400	1.00	1.00	1.00	0.01	0.00	0.00
1600	0.99	1.00	1.00	0.00	0.00	0.00
1800	0.99	1.00	0.99	0.00	0.00	0.00
2000	0.99	0.99	1.00	0.00	0.00	0.00
2200	1.00	1.00	1.00	0.00	0.00	0.00

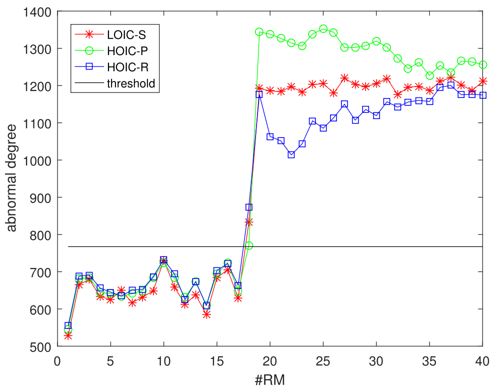


FIGURE 7. Abnormity degree and threshold for different types of attack traffic.

Occasionally, in some *RM*s, our method failed to detect the DDoS attack because the startup point of the attack fell in the second half of the window and the abnormality degree of that *RM* was not accumulated enough. The attack was then detected in the next *RM*. Regarding the false positives, it is inevitable that a small number of extreme exceptions exist in normal data. By increasing the quantity of training data, we can obtain a more appropriate Γ and maintain a relatively low false positive rate.

In order to show the detection effect more intuitively, we checked the abnormality degrees for three AL-DDoS attacks with the *Pktw* set to 1400 and the Γ set to 730. The results are shown in Fig. 7. In all three DDoS attacks, the abnormality degree remains low in early normal stage, and after the attack is launched, the abnormality degree increases and exceeds the threshold considerably. These results validate the hypothesis adopted in our method.

We also assess the detection timeliness of our method. We record the time-stamp of the last packet belonging to the *RM* in which the DDoS is first signaled and use it as the alert-time. Since we have recorded the start-time of every mode of DDoS attack, the detection duration can be calculated by subtracting the start-time from the alert-time. We repeated the test with different attack launching times. The average detection durations are shown in Fig. 8. Compared with the ART-durations and ECN-durations of attacks in Table 1, the detection durations were short, which demonstrates that the DDoS attacks could be detected before they take effect.

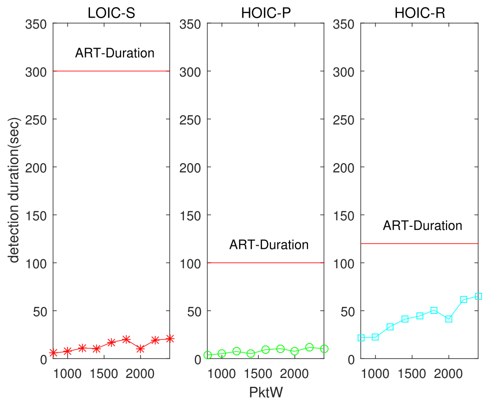


FIGURE 8. Timeliness in detecting DDoS attacks.

TABLE 3. Performance in identifying malicious hosts.

Θ	Precision			Recall		
	LOIC-S	HOIC-P	HOIC-R	LOIC-S	HOIC-P	HOIC-R
0.2	1.00	1.00	0.69	1.00	1.00	1.00
0.4	1.00	1.00	0.92	1.00	1.00	1.00
0.6	1.00	1.00	1.00	1.00	1.00	1.00
0.8	1.00	1.00	1.00	1.00	0.82	0.27

2) VALIDATION OF IDENTIFYING MALICIOUS HOSTS

When a DDoS attack is detected, our system starts tracking the hosts to identify the attack nodes. Let k be 1.5 and *Pktw* be 1400. Let Θ, which is defined in Section III, take values ranging from 0.2 to 0.8. The results are shown in Table 3. Increasing Θ improves the precision but lowers the recall. In our datasets, the optimal results were reached when Θ was near 0.6: both precision and recall reached 100%. However, network conditions are complex and attacks are diverse, so it is difficult to fix the optimal point. When under attack, we would prefer to decrease Θ from 0.8 to 0.2 gradually according to the congestion degree rather than fix the value in advance.

3) ABNORMITY DEGREE OF FLASH CROWDS

Because of the strong elusiveness of AL-DDoS attacks, distinguishing flash crowds from AL-DDoS attacks is a key

$$TPR = \frac{\sharp\left\{j|RM_j \text{ is signaled as abnormal } \& \ RM_j \in RM_{DDoS}\right\}}{Q}$$
$$FPR = \frac{\sharp\left\{j|RM_j \text{ is signaled as abnormal } \& \ RM_j \in RM_{norm}\right\}}{P} \quad (10)$$

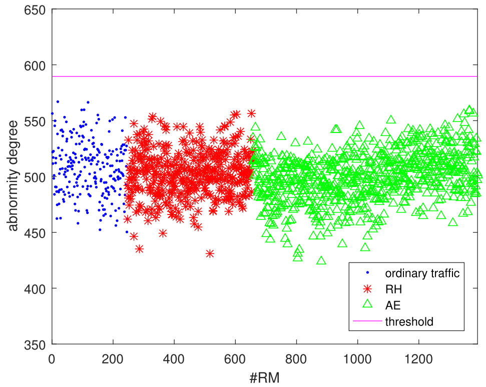


FIGURE 9. Abnormal degree of flash crowds.

requirement for attack detection. We tested our method on the previously mentioned real flash crowd dataset and checked the abnormality degree. This dataset includes some ordinary traffic and two flash-crowd periods, i.e., RC and AE; therefore, we simply replayed the traffic data to the detection system. We chose a *Pktw* of 1400, and set the threshold $\Gamma = 590$ based on a period of ordinary traffic.

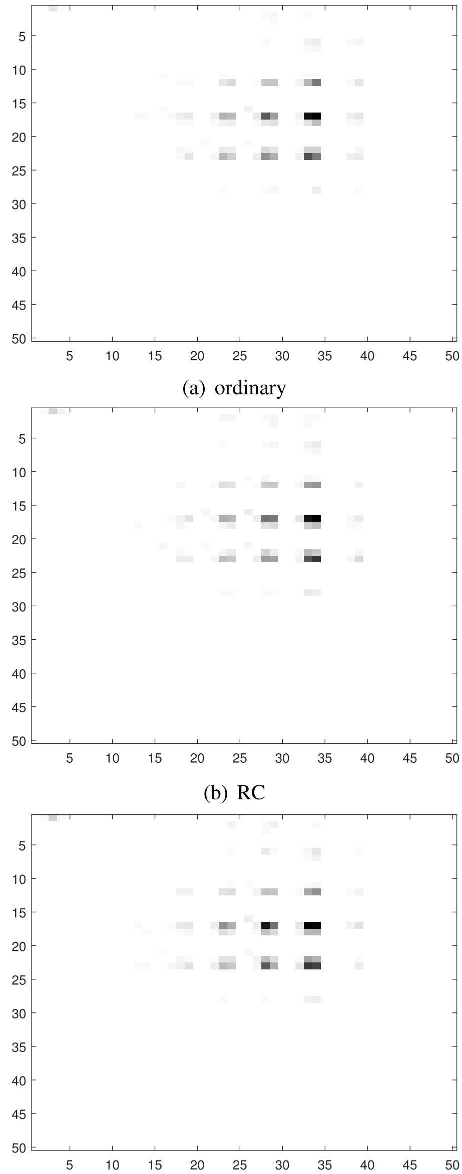
The distribution of the abnormality degree is shown in Fig. 9. Obviously, the abnormality degrees of flash crowd traffic were distributed in the same range as ordinary traffic and did not exceed the threshold. Therefore, a normal sudden traffic spike, i.e., a flash crowd, could not be mistaken for an AL-DDoS attack by our method.

We also selected three *RM*s randomly from the ordinary period and the two flash-crowd periods respectively. Same as before, for visualization purposes, we compacted the 1000×1000 matrices into 50×50 matrices. The gray-scale maps are shown in Fig. 10. Obviously, the distribution of droppoints did not change significantly in the three maps, which can explain why the abnormality degrees in Fig. 9 remain stable.

C. COMPARISON WITH OTHER DDOS DETECTION METHODS

Many methods reported by other researchers could work well in detecting AL-DDoS attacks. However, only a few works have identified malicious hosts and presented the results [19], [20], [27], [48]. In this section, we compare our method with these works in terms of four key aspects: complexity, accuracy, adaptation and data source. The results are shown in Table 4.

TCM-KNN [19], [48] was designed as a lightweight DDoS attack detection scheme for web servers. The reported *TPR* and *FPR* are 99.53% and 1.93%, respectively. E-FCM is employed to boost the real-time detection performance. However, the E-FCM instance selection mechanism for the TCM-KNN algorithm is not amenable to incremental updating, so the model is not adaptive to network dynamics and the training of the model is expensive. Paper [20] extended HsMM (Hidden semi-Markov Model) to describe the browsing behavior of websites. The entropy of the user's HTTP request sequence fitted to the model is used as a criterion to measure the user's normality. This method could achieve an *FPR* as low as 1.5% and a detection rate of approximately 90%. However, the complexity of this algorithm is relatively high, and the data source is the logs of the web server, which will affect the real-time performance. SkyShield [27] exploits the divergence between two sketches to detect anomalies that are caused by numerous requests originating from malicious hosts. A new variant of the Hellinger distance is designed



(a) ordinary

(b) RC

(c) AE

FIGURE 10. Gray-scale map of *RMs* for ordinary and two flash-crowds.

TABLE 4. Comparison with previous works.

	Complexity	Accuracy	Adaptation	Data
TCM-KNN	medium	high	poor	real-time data
HsMM	high	high	fair	web server log
SkyShield	low	medium	good	real-time data
Our Method	low	high	good	real-time data

to measure the divergence to mitigate the impact of network dynamics. However, the sketch is an approximation tool that sacrifices accuracy; thus, SkyShield is not good as the previous methods in terms of accuracy.

Although our proposed method also adopts a highly abstract model to reduce the complexity, a wealth of application-layer information is compacted into the *RM*. Therefore, our method outperforms the aforementioned methods in terms of detection accuracy. The *RM*s are produced online in the detection process, and no special training is required; thus, our model can adapt to network dynamics.

V. DISCUSSION

A few limitations are worth discussing. First, the performance of our method may be influenced by network conditions. Our method was designed to run on an edge router near the web server. If the method is deployed on a backbone router, it should be capable of coping with the high bandwidth by setting a larger *PktW*, but the detection time would be slightly delayed.

Second, sophisticated attackers may increase AL-DDoS traffic slowly, and our method may not be able to notice the change. To overcome this vulnerability, we can select more than one *base matrix* over a longer horizon, thus the abnormality degree could be accumulated sufficiently to detect the slowly increasing attack. Furthermore, low rate DoS attacks against applications (LoRDAS) [49] may cause problems because the special attributes of LoRDAS prevent our method from detecting such attacks during the manifestation phase, unless the attack traffic is sufficient to disturb the rhythm matrix.

Finally, two thresholds, i.e., Γ and Θ, must be set based on historical data of the target website. Alternatively, we could adopt a cluster algorithm to differentiate abnormal data. The clustering scheme could avoid the hassle of setting experimental values but would result in higher computational complexity. In practice, we need a balance between complexity and convenience.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new model, i.e., the rhythm matrix, to characterize access behavior. Based on this model, the abnormal change points of the *RM* are utilized to detect AL-DDoS attacks and to identify malicious hosts. We simulated three modes of DDoS attacks by using the state-of-the-art AL-DDoS attack tools LOIC and HOIC on our campus network. The evaluation demonstrated that our approach performs well in terms of both accuracy and timeliness. At last count, the *TPR* is over 99% and the *FPR* is no more than 1%. The precision and recall of identifying malicious hosts increased to nearly 100% with the optimization of the parameters. All the simulation attacks were detected far earlier than the time at which they took effect. Furthermore, two datasets modified from the 1998 FIFA World Cup dataset illustrated that our method possesses excellent performance in the scenario of flash crowds. In our model, although the traffic increased substantially under flash crowds, the abnormality degree did not increase, whereas the abnormality degree did increase under AL-DDoS attack.

We have tried to reduce the complexity of AL-DDoS detection method and proposed the rhythm matrix as a model with strong identification power. Our future work will combine the rhythm matrix with other online detection methods.

REFERENCES

[1] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tutr.*, vol. 15, no. 4, pp. 2046–2069, 4th,Quart., 2013.

[2] W. Zhou, W. Jia, S. Wen, Y. Xiang, and W. Zhou, "Detection and defense of application-layer DDoS attacks in backbone Web traffic," *Future Gener. Comput. Syst.*, vol. 38, pp. 36–46, Sep. 2014.

[3] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Comput. Surv.*, vol. 39, no. 1, p. 3, 2007.

[4] K. Kalkan, G. Gür, and F. Alagöz, "Filtering-based defense mechanisms against DDoS attacks: A survey," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2761–2773, Dec. 2017.

[5] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments," *IEEE Access*, vol. 7, pp. 80813–80828, 2019.

[6] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, vol. 7, pp. 51691–51713, 2019.

[7] *12th Worldwide Infrastructure Security Report*, Arbor Network, Westford, MA, USA, 2017.

[8] L. Lu, D. Liu, and H.-J. Zhang, "Automatic mood detection and tracking of music audio signals," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 1, pp. 5–18, Jan. 2006.

[9] C. Krumhansl, "Rhythm and pitch in music cognition," *Psychol. Bull.*, vol. 126, no. 1, pp. 159–179, Jan. 2000.

[10] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: A survey," *IEEE Signal Process. Mag.*, vol. 23, no. 2, pp. 133–141, Mar. 2006.

[11] A. Srivastava, B. B. Gupta, A. Tyagi, A. Sharma, and A. Mishra, "A recent survey on DDoS attacks and defense mechanisms," in *Proc. Int. Conf. Parallel Distrib. Comput. Technol. Appl.* Berlin, Germany: Springer, 2011, pp. 570–580.

[12] B. B. Gupta and O. P. Badve, "Taxonomy of dos and ddos attacks and desirable defense mechanism in a cloud computing environment," *Neural Computing Appl.*, vol. 28, no. 12, pp. 3655–3682, 2017.

[13] A. Bhardwaj and S. Goundar, "Comparing single tier and three tier infrastructure designs against DDoS attacks," *Int. J. Cloud Appl. Comput.*, vol. 7, no. 3, pp. 59–75, Jul. 2017.

[14] S. Hosseini and M. Azizi, "The hybrid technique for DDoS detection with supervised learning algorithms," *Comput. Netw.*, vol. 158, pp. 35–45, Jul. 2019.

[15] J. David and C. Thomas, "Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic," *Comput. Secur.*, vol. 82, pp. 284–295, May 2019.

[16] M. Kamarudin, C. Maple, and T. Watson, "Hybrid feature selection technique for intrusion detection system," *Int. J. High Perform. Comput. Netw.*, vol. 13, no. 2, pp. 232–240, 2019.

[17] M. E. Ahmed, S. Ullah, and H. Kim, "Statistical application fingerprinting for DDoS attack mitigation," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 6, pp. 1471–1484, Jun. 2019.

[18] X. Jing, Z. Yan, X. Jiang, and W. Pedrycz, "Network traffic fusion and analysis against DDoS flooding attacks with a novel reversible sketch," *Inf. Fusion*, vol. 51, pp. 100–113, Nov. 2019.

[19] Y. Li, T.-B. Lu, L. Guo, Z.-H. Tian, and Q.-W. Nie, "Towards lightweight and efficient DDOS attacks detection for Web server," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 1139–1140.

[20] Y. Xie and S. Z. Yu, "A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 54–65, Feb. 2009.

[21] P. Negi, A. Mishra, and B. B. Gupta, "Enhanced CBF packet filtering method to detect DDoS attack in cloud computing environment," 2013, *arXiv:1304.7073*. [Online]. Available: https://arxiv.org/abs/1304.7073

[22] M. Chhabra, B. Gupta, and A. Almomani, "A novel solution to handle DDOS attack in MANET," *J. Inf. Secur.*, vol. 4, no. 3, pp. 165–179, 2013.

[23] H. Zhang, A. Taha, R. Trapero, J. Luna, and N. Suri, "SENTRY: A novel approach for mitigating application layer DDoS threats," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 465–472.

[24] K. Singh, P. Singh, and K. Kumar, "Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges," *Comput. Secur.*, vol. 65, pp. 344–372, Mar. 2017.

[25] 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.

[26] V. Matta, M. Di Mauro, and M. Longo, "DDoS attacks with randomized traffic innovation: Botnet identification challenges and strategies," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1844–1859, Aug. 2017.

[27] C. Wang, T. T. N. Miu, X. Luo, and J. Wang, "SkyShield: A sketch-based defense system against application layer DDoS attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 559–573, Mar. 2018.

[28] K. J. Singh, K. Thongam, and T. De, "Detection and differentiation of application layer DDoS attack from flash events using fuzzy-GA computation," *IET Inf. Secur.*, vol. 12, no. 6, pp. 502–512, Nov. 2018.

[29] B. Li, M. Gao, L. Ma, Y. Liang, and G. Chen, "Web application-layer DDoS attack detection based on generalized Jaccard similarity and information entropy," in *Proc. Int. Conf. Artificial Intelligence and Security*. Cham, Switzerland: Springer, 2019, pp. 576–585.

[30] A. Praseed and P. S. Thilagam, "DDoS attacks at the application layer: Challenges and research perspectives for safeguarding Web applications," *IEEE Commun. Surveys Tutr.*, vol. 21, no. 1, pp. 661–685, 1st,Quart,. 2019.

[31] M. Li, "Change trend of averaged Hurst parameter of traffic under DDOS flood attacks," *Comput. Secur.*, vol. 25, no. 3, pp. 213–220, May 2006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167404805001963

[32] S. N. Shiaeles, V. Katos, A. S. Karakos, and B. K. Papadopoulos, "Real time DDoS detection using fuzzy estimators," *Comput. Secur.*, vol. 31, no. 6, pp. 782–790, Sep. 2012.

[33] L. Zhou, M. Liao, C. Yuan, and H. Zhang, "Low-rate DDoS attack detection using expectation of packet size," *Secur. Commun. Netw.*, Oct. 2017, Art. no. 3691629.

[34] D. S. Yeung, S. Jin, and X. Wang, "Covariance-matrix modeling and detecting various flooding attacks," *IEEE Trans. Syst., Man, A, Syst. Hum.*, vol. 37, no. 2, pp. 157–169, Mar. 2007.

[35] Y. Xie and S. Z. Yu, "Monitoring the application-layer DDoS attacks for popular Websites," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 15–25, Feb. 2009.

[36] S. M. Lee, D. S. Kim, J. H. Lee, and J. S. Park, "Detection of DDoS attacks using optimized traffic matrix," *Comput. Math. Appl.*, vol. 63, no. 2, pp. 501–510, Jan. 2012.

[37] L. Nie, D. Jiang, and Z. Lv, "Modeling network traffic for traffic matrix estimation and anomaly detection based on Bayesian network in cloud computing networks," *Ann. Telecommun.*, vol. 72, nos. 5–6, pp. 297–305, 2017.

[38] X. Luo, X. Di, X. Liu, H. Qi, J. Li, L. Cong, and H. Yang, "Anomaly detection for application layer user browsing behavior based on attributes and features," in *Proc. J. Phys., Conf. Ser.*, 2018, vol. 1069, no. 1, Art. no. 012072.

[39] D. Gavrilis, J. Chatzis, and E. Dermatas, "Flash crowd detection using decoy hyperlinks," in *Proc. IEEE Int. Conf. Netw., Sens. Control*. London, U.K., Apr. 2007, pp. 466–470.

[40] X. Jun, Y. XiaoChun, and Z. YongZheng, "Defend against application-layer distributed denial-of-service attacks based on session suspicion probability model," *Chin. J. Comput.*, vol. 33, no. 9, pp. 1713–1724, 2010.

[41] (2019). *High Orbit Ion Cannon*. [Online]. Available: http://sourceforge.net/projects/highorbitioncannon/

[42] (2019). *Low Orbit Ion Cannon*. [Online]. Available: http://sourceforge.net/projects/loic/

[43] NTOP. (2018). *High-Speed Packet Capture, Filtering And Analysis*. [Online]. Available: http://www.ntop.org/

[44] M. Arlitt and T. Jin. (Aug. 1998). *1998 World Cup Web Site Access Logs*. [Online]. Available: http://www.acm.org/sigcomm/ITA/

[45] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, and F. Tang, "Discriminating DDoS attacks from flash crowds using flow correlation coefficient," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 1073–1080, Jun. 2012.

[46] M. Arlitt and T. Jin, "A workload characterization study of the 1998 World Cup Web site," *IEEE Netw.*, vol. 14, no. 3, pp. 30–37, May 2000.

[47] R. Pries, Z. Magyari, and P. Tran-Gia, "An HTTP Web traffic model based on the top one million visited Web pages," in *Proc. 8th Euro-NF Conf. Next Gener. Internet NGI*, Karlskrona, Sweden, Jun. 2012, pp. 133–139

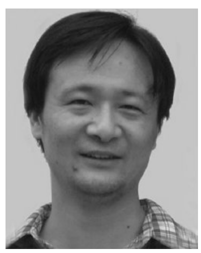
[48] Y. Li, L. Guo, Z. H. Tian, and T. B. Lu, "A lightweight Web server anomaly detection method based on transductive scheme and genetic algorithms," *Comput. Commun.*, vol. 31, no. 17, pp. 4018–4025, Nov. 2008.

[49] G. Maciá-Fernández, J. E. Díaz-Verdejo, P. García-Teodoro, and F. De Toro-Negro, "LoRDAS: A low-rate DoS attack against application servers," in *Proc. Int. Workshop Crit. Inf. Infrastruct. Secur.* Berlin, Germany: Springer, 2007, pp. 197–209.

HUAN LIN received the bachelor's degree in computational mathematics from the Qingdao University of Science and Technology, in 2016, and the master's degree from the School of Software, Shandong University, in 2019. Her research interests include DDoS attack detection and network traffic modeling.

SHOUFENG CAO received the Ph.D. degree in information security. He is currently a Senior Engineer with the National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC). His research interests include the collection and analysis of data that yield new insights about network structure and network performance (e.g., traffic, topology, routing, and energy utilization), malware, and unwanted software.

JIAYAN WU received the B.S. degree in mathematics from the National University of Defense Technology, in 2006, and the M.A. degree in software engineering from Shandong University, in 2016. His research interests include in network and system security, such as network traffic analysis, the Internet malware analysis/detection/defense, intrusion/anomaly detection, and threat intelligence.



ZHENZHONG CAO received the B.Sc. and M.Sc. degrees in computer science from Shandong University, China, in 1996 and 1999, respectively. He is currently a Lecturer with the School of Software, Qufu Normal University. His research interests include binary code analysis and vulnerability discovery.

FENGYU WANG received the master's degree from Shandong University, Jinan, China, in 1999, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2007. He is currently an Associate Professor with the School of Software, Shandong University. His research interests include network traffic modeling and binary code analysis.