Kernel-Based Real-Time DDoS Security System for Software-Defined Networking Postprint

Authors: Liu Min, Teng Hua, He Xianbo

Date: 2019-01-03T10:45:30+00:00

Abstract

To address the issues of low detection accuracy and long latency for DDoS attacks in software-defined networks, a kernel-based real-time security system for DDoS attacks in software-defined networks is proposed. First, packet header information from the software-defined network is extracted each period and organized into matrix form; second, Mahalanobis distance is employed to analyze significant changes in adjacent feature vectors, and two kernel functions are designed to comprehensively evaluate the traffic of attack behaviors; finally, spectral clustering technology and covariance statistical information are utilized to automatically locate attackers. Experiments were conducted based on a real software-defined network, and the results demonstrate that the security system achieves high detection accuracy and ideal processing time.

Full Text

Real-Time DDoS Security System for Software-Defined Networks Based on Kernel Functions

Liu Min, Teng Hua, He Xianbo
(School of Computer Science, China West Normal University, Nanchong, Sichuan 637009, China)

Abstract: To address the problems of low detection accuracy and long latency of DDoS attacks in software-defined networks, this paper proposes a real-time DDoS security system for software-defined networks based on kernel functions. First, packet header information from the software-defined network is extracted periodically and organized in matrix form; second, Mahalanobis distance is employed to analyze significant changes between adjacent feature vectors, and two kernel functions are designed to comprehensively evaluate attack traffic flows; finally, spectral clustering technology and covariance statistical information are used to automatically locate attackers. Experiments conducted on a

real software-defined network demonstrate that the security system achieves high detection accuracy and ideal processing time.

Keywords: software-defined network; network security; denial-of-service attack; kernel function; spectral clustering

0 Introduction

DDoS (Distributed Denial of Service) attacks are currently the most common type of attack on the Internet [1]. DDoS attacks are simple to implement and have low implementation costs, making them a significant threat in various types of networks [2]. Software-Defined Networking (SDN) realizes the concept of separating network control from data forwarding, supporting high openness and programmability. However, security issues limit the large-scale deployment and application of SDN in many scenarios. SDN security mechanisms can be mainly divided into six categories [3]: a) SDN security controllers; b) composable security module libraries; c) DoS/DDoS attack defense systems for controllers; d) legitimacy and consistency verification of flow rules; e) northbound interface security; and f) application security. Among these, DDoS attacks can directly cause network paralysis, posing an extremely serious threat to SDN [4].

Literature [5] utilizes GHSOM technology to design a DDoS attack detection method based on object characteristics. This method combines SDN network and attack features to propose a destination address-based detection heptad, which serves as the detection element for determining whether a target address is under DDoS attack. Literature [6] proposes a modular DDoS attack detection method based on the KNN algorithm in an SDN environment, which selects five key traffic features of SDN networks and employs an optimized KNN algorithm for traffic anomaly detection. Literature [7] proposes a controller DDoS detection system that first classifies port flow events and uses Sequential Probability Ratio Test (SPRT) to detect whether traffic is abnormal. Literature [8] analyzes several types of SDN DDoS security systems and finds that traffic statistical analysis methods generally achieve good detection effectiveness but have low computational efficiency, while SDN packet header feature analysis methods have high computational efficiency but low detection accuracy.

DDoS attacks can cause entire network paralysis, so it is necessary to identify DDoS traffic in the early stages to prevent greater damage [9]. To maintain fast processing speed while ensuring the detection accuracy of the DDoS security system, this paper designs a real-time DDoS security system for software-defined networks based on kernel functions. The emergence of DDoS attacks is often accompanied by dramatic changes in traffic patterns, so this system treats significant changes in message flow features as potential DDoS attacks. Current mainstream DDoS security systems primarily detect DDoS attack traffic but cannot locate attack sources, whereas this system can detect DDoS attack behavior and identify attackers. This system is an unsupervised method that utilizes

observed message traffic types and data volumes without requiring additional information.

---

1 System Model

1.1 OpenFlow Model for Software-Defined Networks

The controller manages all routing decisions and completes packet forwarding through the flow table of OpenFlow switches. [Figure 1: see original paper] shows the packet header of the OpenFlow version 1.3 protocol [10].

[Figure 1: see original paper] Message header of OpenFlow version 1.3 protocol

[Figure 2: see original paper] shows the processing flow of OpenFlow packets.

[Figure 2: see original paper] Processing flow of OpenFlow message

1.2 DDoS Attack Model for Software-Defined Networks

Control packets are sampled once every period, denoted as $t = i\Delta t$, and samples from this time period are formed into a feature vector. Δt represents an observation period that monitors control packets received by the switch during this period. At the end of Δt, the traffic from IP_{SRC} users is represented as a d-dimensional vector v_r, where d is the number of various traffic types. The elements of vector v are integers, corresponding to the occurrence count of each message type within the i-th time frame ($(i-1)\Delta < t < i\Delta$).

Assume the r-th user sends P_r messages during the observation period, with each message's counter represented as $v_{r,p}$, $p = 1, \ldots, P_r$. Therefore, the total sampled information during the observation period is $v_r = \sum_{p=1}^{P_r} v_{r,p}$, where v_r represents the counter matrix sent by the r-th user, as shown in [Figure 3: see original paper].

[Figure 3: see original paper] Counter matrix of rth user

Let x be the state vector, representing a d-dimensional count vector defined as the number of collaborative activities of $|U|$ users during an observation period. The sum of all user counter vectors at the server side is defined as the server state vector, calculated as $x = \sum_{r=1}^{|U|} v_r$, as shown in [Figure 4: see original paper].

[Figure 4: see original paper] Diagram of server states vector

Assume x_i and $x_j \in \mathbb{R}^d$ represent the server state vectors for the i-th and j-th observation periods, respectively. This paper uses these feature vectors (server state vectors) to monitor traffic changes in software-defined networks. Let M be a $d \times d$ positive definite matrix, $D_M(x_i, x_j)$ be the distance between feature vectors x_i and x_j, where M represents the scale matrix. $f(M|x_n : x_{n-k-1})$ is a

function of M, defined as the record of feature vectors in a window of length k from x_{n-k-1} to x_n.

The above feature vectors do not contain timestamp information. Assume the timestamp sequence for the r-th user and message P_r is $t_{r,1}, \ldots, t_{r,P_r}$. Timestamp information is introduced by augmenting the message indicator vector with an augmented vector. The specific method is:

Let $w_{r,p} = [v_{r,p}^\top, t_{r,p}]^\top$ represent the timestamped message indicator vector, where the relationship between $w_{r,p}$ and vector $v_{r,p}$ is shown in [Figure 5: see original paper].

[Figure 5: see original paper] Time-stamped user message vector

Any user u_r can be characterized as a time series, defined as a matrix $W_r = [w_{r,1}|w_{r,2}|\ldots|w_{r,P_r}]$. Kernel functions are used to calculate the similarity between two users (u_q, u_r), expressed as $K(u_q, u_r)$.

Define $\kappa(w_{r,p}, w_{q,p}) = \exp(-\gamma D_M(v_{r,p}, v_{q,p}) - \rho|t_{r,p} - t_{q,p}|)$ as the similarity between two users in the same time period, where $w_{r,p}$ represents the p-th message of the r-th user and $w_{q,p}$ represents the p-th message of the q-th user. Finally, a $|U| \times |U|$ kernel matrix K for all users in an observation period can be calculated.

---

2 Anomaly Monitoring Based on Adaptive Distance

In a stationary process, message features should have high statistical similarity, while the distance between two feature sets in a non-stationary process is larger. Therefore, by detecting significant change points in message feature distances, sampling time traffic can be analyzed to detect potential DDoS attack behavior.

2.1 Mahalanobis Distance

Let $M \in \mathcal{S}_+$ be a $d \times d$ symmetric positive semidefinite matrix. The Mahalanobis distance D_M between x_i and x_j is calculated as:

$$D_M(x_i, x_j) = (x_i - x_j)^\top M(x_i - x_j)$$

The inverse of the full-rank sample covariance matrix Σ is a special case of Mahalanobis distance. If the feature set follows a standard Gaussian distribution, then $M = \Sigma = I$. The symmetric positive semidefinite matrix can be decomposed as $M = A^\top A$, where A is an $e \times d$ projection matrix and $e \leq d$. The following relationship can be obtained:

$$D_M(x_i, x_j) = D_E(Ax_i, Ax_j) = \|Ax_i - Ax_j\|^2 = \|a_i - a_j\|^2$$

where $a_i = Ax_i$ is the projection vector and D_E is the Euclidean distance. Equation (2) shows that Mahalanobis distance in feature space is equivalent to Euclidean distance in projection space.

2.2 Network Traffic Model Based on Distance

Network traffic monitoring is achieved by observing the sum of distances in a sliding window, called the moving distance of traffic. The sum of the sliding window is compared with a threshold ϵ to determine whether it is abnormal traffic. The moving distance of a window of size k can be defined as a function of a symmetric positive definite matrix ($M \in \mathcal{S}_{++}$), as shown in Equation (3):

$$f(M|x_n : x_{n-k-1}) = \sum_{j=n-k}^{n-1} D_M(x_j, x_{j-1}) = \sum_{j=n-k}^{n-1} (x_j - x_{j-1})^\top M(x_j - x_{j-1})$$

If the moving distance calculated using Mahalanobis distance exceeds the threshold ϵ, i.e., $f(M_{n-1}|x_n : x_{n-k-1}) > \epsilon_{th}$, an alarm is triggered. The Mahalanobis distance is updated each period, defined as:

$$M_n = \arg\min_{M \in \mathcal{S}_{++}} [f(M|x_n : x_{n-k-1}) + \lambda \mathrm{tr}(M) - \beta \log\det(M)]$$

where the second and third terms are regularization functions based on the LogDet divergence [11]. The LogDet function can estimate the distance between two matrices, where $\mathrm{tr}(\cdot)$ is the matrix trace function.

By calculating the derivative of Equation (4), the optimal Mahalanobis matrix (M^*) can be obtained:

$$M_n^{-1} = \frac{\beta}{\lambda}\left(M_{n-1}^{-1} + \frac{1}{\beta}\sum_{j=n-k}^{n-1}(x_j - x_{j-1})(x_j - x_{j-1})^\top\right)$$

This Mahalanobis matrix is updated repeatedly each period. Algorithm 1 shows the traffic change detection algorithm.

Algorithm 1: Adaptive Moving Distance Traffic Change Detection Algorithm

1. Initialize parameters $k, \lambda, \beta, \alpha$;
2. while (new traffic) {
3. Observe traffic within window (size k), calculate counter vector;
4. if $f(M_{n-1}|x_n : x_{n-k-1}) > \epsilon$ {
5. `Trigger alarm;`
6. }
7. Update M_n according to Equation (6);
8. }

2.3 Threshold for Moving Distance

Based on experimental analysis, the distribution of the sum of moving distances can be approximated as a chi-square distribution. The Mahalanobis distance is obtained from a Gaussian distribution, thus: $\mu = x_n$, $\Sigma = M^{-1}$.

Assume y is the observation set of the current sliding window. If y is a d-dimensional random vector following a Gaussian distribution with mean vector μ and covariance matrix Σ, then $z = (y - x_n)^\top M(y - x_n) = (y - \mu)^\top \Sigma^{-1}(y - \mu)$ can be transformed into a chi-square distribution with d degrees of freedom.

Assume z_i represents k independent and identically distributed random variables, where z_i follows a chi-square distribution $\chi^2_{d_i}$. According to the properties of independent chi-square distribution variables, the sum of random variables follows a chi-square distribution with degrees of freedom $d_1 + d_2 + ... + d_k$.

Specifically, this can be expressed as:

$$Z = \sum_{i=1}^{k} z_i \sim \chi^2_{\sum_{i=1}^{k} d_i}$$

Finally, the threshold for the anomaly traffic detection model is:

$$\epsilon_{th} = \chi^2_{kd,\alpha}$$

where α represents the probability of receiving abnormal traffic. The probability that the Z value reaches the threshold ϵ_{th} under normal traffic conditions is α. The value of α can be set according to application scenarios as $\alpha \in \{0.1, 0.05, 0.02, 0.01\}$.

---

3 Malicious User Identification

If an abnormal traffic is a DDoS attack, it is necessary to identify the set of malicious users to prevent distributed attacks. The historical behavior of each user during the observation period is represented as a time series, as shown in [Figure 3: see original paper]. Similarity functions are used to process time series, classifying users with similar behavior into the same group.

Two attack discrimination methods are proposed: a) calculating the distance between message sequence features based on global time series alignment kernel; and b) extracting user message quantity vectors at the end of the observation period.

3.1 Time Series Alignment Kernel

This paper considers timestamped messages within a k-length window, representing messages in time series format as $(n - k - 1), ..., (n - 1)$. Each user's

sequence contains different numbers of message events occurring at different time points. The goal is to calculate similarity between user messages based on kernel methods, which first requires aligning message queues. Similarity between messages of different lengths is low, while similarity between the same type of messages is high.

Assume the timestamped message sequences for a pair of users (u_q, u_r) are (W_q, W_r), with 3 and 2 message events respectively. [Figure 6: see original paper] shows all possible alignment cases between W_q and W_r, with a total of 5 alignment cases as follows:

Literature [12] proposes a global alignment kernel that uses dynamic programming to calculate the similarity between two sequences. This paper implements the algorithm based on [12], with the only modification being the introduction of Mahalanobis distance.

[Figure 6: see original paper] All aligned cases of W_q and W_r

3.1.1 Global Sequence Alignment Kernel Given two message sequences for users (u_q, u_r):

$$W_q = [w_{q,1}|w_{q,2}|\dots|w_{q,P_q}]$$

$$W_r = [w_{r,1}|w_{r,2}|\dots|w_{r,P_r}]$$

Assume the state space is Ω, and design a two-dimensional array $TP_{q,Pr}$ to store the two-dimensional sequence, where $TP_{q,0} = 0$ $(p_q = 1, \dots, P_q)$, $T_{0,Pr} = 0$ $(p_r = 1, \dots, P_r)$, and $T_{0,0} = 0$. Assume there exists a function to measure the similarity between message events of users u_q and u_r, denoted as $\kappa(w_{q,p_q}, w_{r,p_r})$. Therefore, $TP_{q,Pr}$ can be calculated recursively:

$$T_{p_q,p_r} = \max \begin{cases} T_{p_q-1,p_r-1} + \kappa(w_{q,p_q}, w_{r,p_r}) \\ T_{p_q-1,p_r} + \kappa(w_{q,p_q}, \text{gap}) \\ T_{p_q,p_r-1} + \kappa(\text{gap}, w_{r,p_r}) \end{cases}$$

The final unnormalized similarity result between two users (u_q, u_r) can be obtained as shown in Equation (10):

$$K_{\text{unnormed}}(u_q, u_r) = T_{P_q,P_r}$$

After obtaining the kernel matrix for each pair of users, to solve the problem of inconsistent scales, unit diagonal regularization is performed on the $|U| \times |U|$ kernel matrix, where $|U|$ represents the number of active users in the system. The unit diagonal regularization method is shown in Equation (11):

$$K(u_q, u_r) = \frac{K_{\text{unnormed}}(u_q, u_r)}{\sqrt{K_{\text{unnormed}}(u_q, u_q) \cdot K_{\text{unnormed}}(u_r, u_r)}}$$

The above kernel matrix is called the time series kernel.

3.1.2 Kernel Function Each user in a window can be characterized as a timestamped message sequence, where user message sequences include different lengths and different message types. Assume two timestamped vectors are $w_{r,p} = [v_{r,p}^\top, t_{r,p}]^\top$ and $w_{q,p} = [v_{q,p}^\top, t_{q,p}]^\top$. The kernel function between the two vectors is defined as:

$$\kappa(w_{r,p}, w_{q,p}) = \exp(-\gamma D_M(v_{r,p}, v_{q,p}) - \rho|t_{r,p} - t_{q,p}|)$$

where M is the Mahalanobis matrix from Equation (6). The coefficients γ and ρ are the weights for message distance and time distance, respectively. This paper assumes they are equal: $\gamma = \rho = 1$.

3.2 User Distance Kernel

The similarity kernel matrix between users is calculated based on Mahalanobis distance. If the Mahalanobis distance is closer to 0, the similarity between two users is higher; conversely, the similarity is lower. The Mahalanobis distance kernel can be considered a special case of the Gaussian kernel. Based on user message count vectors $v_q, v_r \in \mathbb{R}^d$, the similarity between two users u_q and u_r is compared:

$$K(u_q, u_r) = \exp(-(v_q - v_r)^\top M(v_q - v_r))$$

The kernel calculated by Equation (13) is referred to as the distance kernel. If $v_q = v_r$, then $K(u_q, u_r) = 1$. This feature vector does not contain message timestamp information and only evaluates user message features within the window.

3.3 Spectral Clustering Algorithm

Equations (11) and (13) calculate the kernel matrix K representing similarity between users. Matrix K can be represented as a fully connected weighted adjacency graph, where vertices represent users and edges represent similarity. The adjacency matrix should be divisible into two subgraphs: malicious users and legitimate users. The Laplacian spectral clustering algorithm is used to classify the adjacency graph, which can group similar nodes together while ensuring dissimilar nodes remain far apart [13].

Define the sum of elements in the q-th row of the kernel matrix as the degree of the q-th active user. Then the degree of the q-th active user in a given window is defined as:

$$d_q = \sum_{r=1}^{|U|} K(u_q, u_r)$$

The degree matrix D is a diagonal matrix with diagonal elements being the "degrees": $d_{g1}, d_{g2}, \ldots, d_{g|U|}$. The Laplacian matrix L is evaluated as defined in Equation (15):

$$L = D - K$$

where K is the $|U| \times |U|$ kernel matrix, and $K(u_q, u_r)$ is calculated using Equation (11) or Equation (13).

Algorithm 2: Laplacian Spectral Clustering Algorithm

Input: Kernel matrix $K \in \mathbb{R}^{|U|\times|U|}$.
Output: Evaluation results of matrices D and L.

1. Compute the first two eigenvectors Ψ_1 and Ψ_2 of the generalized eigenproblem $L\Psi = \Lambda D\Psi$, where Λ represents the diagonal matrix of eigenvalues $\lambda_1, \ldots, \lambda_{|U|}$.
2. Form matrix $\Psi \in \mathbb{R}^{|U|\times 2}$ from eigenvectors Ψ_1 and Ψ_2. Use each row of Ψ as a new feature vector in the mapping space.

3.4 Automatic Identification of Malicious User Classes

Most malicious users exhibit repetitive and correlated behavior, while legitimate users generally exhibit non-repetitive and diverse behavior. Section 4.3 classifies legitimate and malicious users, and the next task is to identify the attack type of malicious users.

Because malicious user message sequence vectors have high repetitiveness and low diversity, the covariance matrix of intra-class user message sequence vectors is calculated. If the covariance matrix is smaller, the likelihood of that class being malicious users is higher. Algorithm 3 shows the malicious user selection algorithm.

Algorithm 3: Malicious User Class Selection Algorithm

Input: Classified vectors C.
Output: Types of classes C_1 and C_2.

1. Calculate the covariance matrix of projected message vectors within classes C_1 and C_2;
2. if (covariance matrix == 0) {
3. The class is malicious users;
4. } else {
5. Calculate eigenvalues of the covariance matrix;
6. The class with the highest eigenvalue is malicious users.
7. }

Algorithm 4: Overall Pseudocode of DDoS Intelligent Security System

1. Extract OpenFlow packet header information;

2. Establish vector model and matrix model according to Section 2.2;
3. if (time series alignment kernel) {
4. Set weight parameters γ and ρ;
5. Calculate kernel matrix K, i.e., $K_{q,r} = K(u_q, u_r)$;
6. Perform diagonal regularization on K_{unnormed} to obtain K;
7. }
8. if (user distance kernel) {
9. Calculate kernel matrix K, i.e., $K_{q,r} = K(u_q, u_r)$;
10. }
11. Use Laplacian spectral clustering algorithm (Algorithm 2) to classify K and obtain result C;
12. Use Algorithm 3 to detect malicious users in C.

---

4 Experiments

The experimental environment is a PC with an Intel Core i7-4770 processor at 3.4 GHz and 8 GB of memory. The operating system is Linux Ubuntu 14.04, the switch system is Mininet Version 2.2.26, and OpenFlow version 1.3. An actual software-defined network was built in the laboratory, as shown in [Figure 7: see original paper]. The network consists of three controllers, each containing a POX control layer, 64 hosts, and 8 OpenFlow switches. Each OpenFlow switch connects 8 hosts to form a subnet. POX is a fast, lightweight SDN control layer that supports various platforms, so the POX control layer was adopted in the experiments.

The simulation network was built using the MiniEdit tool, with Open vSwitch (OVS) as the network switch, which has implemented various mainstream network interfaces and protocols. The Scapy tool was used to generate actual flooding attacks, with legitimate traffic containing random data flows. Network traffic was generated using Python programming, with the "randrange" function used to generate random source IP addresses. The target IP address range for legitimate traffic was 10.0.0.1~10.0.0.64. Two Python scripts were run, one responsible for generating attack traffic and the other for generating legitimate traffic.

[Figure 7: see original paper] Software defined network constructed in the experiment

4.1 Experimental Setup

4.1.1 Performance Evaluation Metrics The performance of the DDoS security system is evaluated using three metrics: precision, recall, and F-score, defined as:

- Precision = Correctly detected attacks / All detected attacks
- Recall = Correctly detected attacks / Total number of samples

- F-score = Precision × Recall × 2 / (Precision + Recall)

4.1.2 Test Cases To test the security of this system against DDoS attacks, two experimental test cases are considered. The first test case is a single-target attack experiment: three different traffic ratio DDoS attacks are run on one host, with attack traffic ratios of 10%, 20%, and 30%. The second test case is a subnet-target attack experiment: three different ratio DDoS attacks are run in one subnet, with attack traffic ratios of 10%, 20%, and 30%.

4.1.3 Simulation Parameter Settings shows the parameter settings in the experiments. Because there are 64 hosts in the experiments, the window size of this system is set to 80. Based on preliminary experimental results, $k = 5$, $\lambda = 1$, $\beta = 4$, $c = 1$ achieve good detection performance in this experimental environment.

Parameters of test cases

Parameter	Value
Legitimate traffic interval	0.025 seconds
Attack traffic interval	0.025 seconds

Because the sampling period of this system has a certain impact on detection accuracy, this paper considers three sampling periods: 1 s, 3 s, and 5 s. This algorithm is compared horizontally with two similar DDoS detection systems, KNNDD [6] and VNDD [7], to comprehensively evaluate the performance of this system. Each group of experiments is run independently 30 times, with the results of the 30 experiments used as the final statistical results.

4.2 Experimental Results

4.2.1 Single-Target Attack Results [Figure 8: see original paper] shows the results of the single-target attack experiment for the three security systems. The figure shows that the precision, recall, and F-score of this system with a 1 s sampling period are lower than those of the KNNDD and VNDD systems, while the performance of this system with 3 s and 5 s sampling periods has obvious advantages.

[Figure 8: see original paper] Result of single target attack of three security systems

4.2.2 Subnet-Target Attack Results [Figure 9: see original paper] shows the results of the subnet-target attack experiment for the three security systems. The figure shows that the precision, recall, and F-score of this system with a 1 s sampling period are lower than those of the KNNDD and VNDD systems. The performance of this system with a 3 s sampling period is close to that of the

KNNDD algorithm, while the performance of this system with a 5 s sampling period shows obvious advantages.

[Figure 9: see original paper] Result of sub-network target attack of three security systems

4.2.3 Computational Efficiency of Security System According to the processing flow of Algorithm 4, this system mainly consists of three modules: distance kernel calculation, time series alignment kernel calculation, and spectral clustering processing. The average processing time of the three modules was statistically measured in the experiments, as shown in [Figure 10: see original paper]. The figure shows that under three different sampling periods, the processing time of this system is relatively stable, with a total time of about 3.5 s, meeting the requirements of real-time detection.

[Figure 10: see original paper] Average processing time of three models

---

5 Conclusion

The emergence of DDoS attacks is often accompanied by dramatic changes in traffic patterns. This system treats significant changes in message flow features as potential DDoS attacks. This system can detect DDoS attack behavior and identify attackers. It is an unsupervised method that utilizes observed message traffic types and data volumes without requiring additional information. Experiments based on a real software-defined network show that the security system achieves high detection accuracy and ideal processing time. Future work will consider testing in actual large-scale software-defined networks.

References

[1] Saied A, Overill R E, Radzik T. Detection of known and unknown DDoS attacks using artificial neural networks [J]. Neurocomputing, 2016, 172(C): 385-393.
[2] Kolias C, Kambourakis G, Stavrou A, et al. DDoS in the IoT: mirai and other botnets [J]. Computer, 2017, 50(7): 80-84.
[3] Wang Mengmeng, Liu Jianwei, Chen Jie, et al. Software defined networking: security model, threats and mechanism [J]. Journal of Software, 2016, 27(4): 969-992.
[4] Wang Xiulei, Chen Ming, Xing Changyou, et al. Software defined security networking mechanism against DDoS attacks [J]. Journal of Software, 2016, 27(12): 3104-3119.
[5] Yao Linyuan, Dong Ping, Zhang Hongke. Distributed denial of service attack detection based on object character in software defined network [J]. Journal of Electronics & Information Technology, 2017, 39(2): 381-388.

[6] Xiao Fu, Ma Junqing, Huang Xunsong, et al. DDoS attack detection based on KNN in software defined networks [J]. Journal of Nanjing University of Posts and Telecommunications: Natural Science, 2015, 35(1): 84-88.
[7] Dong P, Du X, Zhang H, et al. A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows [C]//Proc of IEEE International Conference on Communications. 2016: 1-6.
[8] Kalkan K, Gur G, Alagoz F. Defense mechanisms against DDoS attacks in SDN environment [J]. IEEE Communications Magazine, 2017, 55(9).
[9] Yang Jungang, Wang Xintong, Liu Guqing. DDoS attack detection method based on network traffic and IP entropy [J]. Application Research of Computers, 2016, 33(4): 1145-1149.
[10] Šulák V, Helebrandt P, Kotuliak I. Performance analysis of OpenFlow forwarders based on routing granularity in OpenFlow 1.0 and 1.3 [C]//Proc of Open Innovations Association. 2017: 236-241.
[11] Davis J V, Kulis B, Jain P, et al. Information-theoretic metric learning [C]//Proc of International Conference on Machine Learning. 2007.
[12] Cuturi M, Vert J, Birkenes O, et al. A kernel for time series based on global alignments [C]//Proc of IEEE International Conference on Acoustics, Speech and Signal Processing. 2006: II-413-II-416.
[13] Luxburg U V. A tutorial on spectral clustering [J]. Statistics & Computing, 2007, 17(4): 395-416.

Note: Figure translations are in progress. See original paper for figures.