Postprint of LFA Attack Detection Method Based on MS-KNN Algorithm in SDN

Authors: Sun Wenyue, Wang Changda

Date: 2022-05-10T00:00:00+00:00

Abstract

To address the challenge of detecting a novel DDoS attack—link-flooding attack (LFA)—an LFA detection method based on the MS-KNN (Mean Shift-K-Nearest Neighbor) approach in Software-Defined Networking (SDN) is proposed. First, an SDN experimental platform is constructed to simulate LFA attacks and build an LFA dataset. Subsequently, an improved weighted Euclidean distance Mean Shift (MS) algorithm is employed to classify the LFA dataset. Finally, the K-Nearest Neighbor (KNN) algorithm is utilized to determine whether LFA data is present in the classification results. Experimental results demonstrate that compared with the KNN algorithm alone, MS-KNN not only achieves higher accuracy but also yields a lower false positive rate.

Full Text

Preamble

LFA Attack Detection Method Based on MS-KNN Algorithm in SDN

Sun Wenyue, Wang Changda†

School of Computer & Communication, Jiangsu University, Zhenjiang, Jiangsu 212013, China

Abstract: To address the challenge of detecting a new type of DDoS attack—link-flooding attack (LFA)—this paper proposes an LFA detection method based on the MS-KNN (Mean Shift-K-Nearest Neighbor) approach in SDN. First, an SDN experimental platform was built to simulate LFA and construct an LFA dataset. Then, an improved weighted Euclidean distance Mean Shift (MS) algorithm was used to classify the LFA dataset. Finally, the K-nearest neighbor (KNN) algorithm was employed to determine whether LFA data existed in the classification results. Experimental results demonstrate that compared with

the KNN algorithm alone, MS-KNN achieves higher accuracy and lower false positive rates.

Keywords: link-flooding attack; SDN; mean shift algorithm; K-nearest neighbor algorithm; MS-KNN

0 Introduction

Distributed Denial of Service (DDoS) attacks represent one of the most serious threats to network security. According to the "2021 DDoS Attack Situation Report" released by NSFOCUS and China Telecom [**?**], DDoS attack methods have become increasingly complex and unpredictable, with hybrid DDoS attacks increasing by 80.8% in 2021 compared to 2020. In recent years, a new DDoS attack vector—link-flooding attack (LFA)—has emerged [**?**, **?**]. This attack can effectively sever Internet connectivity for targeted organizations such as university campuses, military bases, or groups of energy distribution stations.

Unlike traditional DDoS attacks, LFA attackers do not directly send massive attack traffic to target servers. Instead, they use a group of compromised machines (or a botnet) to attack specific links or network regions, aiming to disconnect target hosts within that region from external networks. To achieve this, compromised machines send legitimate, low-rate traffic to decoy servers or bots in the target area. When this traffic passes through critical links connecting these servers, the target links become congested and blocked. The most notable characteristic of this attack is its use of legitimate, low-rate traffic to cause significant performance impact, making it particularly difficult to detect and defend against [**?**].

LFA encompasses two attack types: Coremelt [**?**] and Crossfire [**?**]. Coremelt, first proposed by Studer et al. [**?**], defines target links and employs a set of compromised machines that send data to each other to flood the target link. Launching a Coremelt attack involves three steps: (1) selecting core links in the network as target links; (2) identifying which compromised machines can generate traffic traversing the target links; and (3) sending traffic over the target links identified in step 1 to overload them. Crossfire is an upgraded version of Coremelt that targets more complex networks by simultaneously attacking multiple links. This attack uses compromised machines as sources, public servers as destinations, and floods target links through communication between sources and destinations. To ensure attack persistence, adversaries can dynamically change the set of target links—a capability absent in Coremelt. Launching a Crossfire attack involves four steps: (1) constructing a link map; (2) calculating flow density and selecting target links for attack configuration; (3) coordinating compromised machines to distribute attack flows and flood target links; and (4) rolling the attack.

Since LFA attack traffic easily hides within normal network traffic, an effective

method is needed to identify LFA attacks within massive datasets.

Software-Defined Networking (SDN) is an emerging network paradigm characterized by granularity, flexibility, and elasticity, offering new approaches for defending against network attacks. SDN typically has three fundamental characteristics [**?**]: (a) clear separation of control and data planes, with forwarding decisions made in the control plane; (b) abstraction of network logic from hardware to software implementation; and (c) use of a controller or network operating system [**?**] to implement device forwarding decisions.

SDN provides a mainstream network management architecture that breaks free from hardware limitations by decoupling the control and data planes in networks. The control plane can manage network devices through unified interface protocols (such as OpenFlow [**?**], P4 [**?**]) and define network policies by planning forwarding rules; the data plane processes and forwards packets according to these defined rules. Consequently, compared with traditional network architectures, SDN' s key difference lies in its ability to flexibly define network device forwarding capabilities through flow table operations. Its forwarding decisions are flow-based rather than destination-based, defined by matching criteria from packet header field values and a set of actions. In SDN, a flow is a sequence of packets between sender and receiver devices. This flow-based abstraction unifies the behavior of various network devices such as switches, routers, firewalls, and middleboxes [**?**].

SDN offers several advantages [**?**], including network-wide visibility, logically centralized control, software-based traffic analysis, and dynamic updating of forwarding rules, all of which enable more effective attack detection. Therefore, the SDN architecture provides a solid foundation for LFA detection.

Existing research has proposed several LFA detection methods. RL-Shield [**?**] mitigates LFA by monitoring source IP behavior using Dirichlet distributions and Bayesian statistics, and employs hop-by-hop techniques to connect related node pairs, achieving detection through frequent path changes. However, this method is affected by network topology and has poor portability in real networks. BALANCE [**?**] proposes a new mechanism based on hybrid SDN that enables controllers to collect data from all links in the network for congestion detection by strategically placing nodes. However, in large networks, collecting statistics from all links incurs substantial computational overhead. LFADefender [**?**] is a novel LFA defense system that uses SDN-based target link selection algorithms. Through the global view provided by SDN controllers, it dynamically tracks flow paths in the network and employs probes to send large numbers of real-time probe packets to detect link congestion. However, this system has slow response times, and attackers may change target links faster than the defense mechanism can take effect. The LFA detection method based on damage probability [**?**] considers shortest paths between all node pairs in the network, requiring enormous computational cycles in large networks. LinkScope [**?**] uses hop-by-hop and end-to-end network measurement methods to detect LFA attacks, but requires deploying many additional probing points across the

network, resulting in high resource overhead in large networks. Moreover, its probing progress depends on previous day' s network traffic, introducing additional traffic into the network. Woodpecker [**?**] uses incremental SDN deployment to mitigate LFA, but its congestion detection module must be installed on all SDN nodes, and not all nodes have programmable capabilities or sufficient memory for module installation. Similarly, Software-Defined HoneyNet [**?**] leverages SDN' s global view advantages to infer potential honeypot nodes connected to the honeynet, increasing attacker costs. However, this scheme does not consider attribute importance for accurately identifying bottlenecks in real-world infrastructure and lacks additional graph metrics for intelligent honeypot node selection.

Although SDN architecture innovations in network security offer clear benefits and are considered suitable for today' s high-bandwidth, dynamic network environments [**?**, **?**, **?**], SDN alone is insufficient for LFA detection due to LFA' s unique characteristics.

K-Nearest Neighbor (KNN) [**?**] is a lazy learning algorithm. Its basic principle involves calculating distances or similarities between a sample to be classified and known training samples, identifying the k nearest neighbors, and determining the sample's class based on these neighbors' classes. KNN demonstrates high accuracy and precision for network data classification. However, for large-scale data classification, KNN suffers from two major drawbacks: (1) it must calculate distances from each unclassified sample to all known samples to find its k nearest neighbors, incurring massive overhead for large network datasets; and (2) when determining test sample classes, the algorithm only considers nearest neighbors, and when facing large-scale network data, indistinct features between data can bias classification results.

Clustering algorithms are unsupervised learning methods primarily used for dimensionality reduction. Since LFA attack traffic easily hides within normal network traffic, aggregating similar data to reduce the amount requiring analysis is essential for effectively discovering LFA attacks in large datasets.

Mean Shift (MS) [**?**] is an unsupervised clustering algorithm that does not require pre-specifying the number of cluster centers. Network data can be divided into one or more clusters, and LFA presence can be identified by analyzing cluster characteristics. However, the MS algorithm does not consider the varying contribution degrees of different data attributes to classification, leading to unsatisfactory clustering results. Therefore, setting appropriate weights for specific data types significantly impacts clustering effectiveness.

To address KNN' s limitations for large-scale network data and MS' s poor classification performance, this paper proposes the MS-KNN algorithm. Experimental results demonstrate that MS-KNN not only overcomes poor classification performance across different data types but also substantially reduces computational time overhead. The main contributions are: (a) proposing a weighted Euclidean distance-based MS clustering algorithm for LFA network traffic clas-

sification, which addresses MS' s poor performance on LFA traffic; (b) proposing an LFA detection method based on MS-KNN that combines improved MS and KNN algorithms, using MS output as KNN input and employing grid search and cross-validation to find optimal parameters for best detection results; and (c) conducting experiments in a real SDN environment, with results confirming MS-KNN' s effectiveness and showing it achieves higher True Positive Rate (TPR), Positive Predictive Value (PPV), Accuracy (ACC), and lower False Positive Rate (FPR) compared with traditional KNN.

1.1 MS Algorithm Principle

The MS algorithm [**?**] utilizes probability density gradients to find local optima. By defining kernel functions, samples at different distances from the offset point contribute differently to the mean shift vector. The most typical kernel functions are Gaussian and Epanechnikov kernels [**?**]. By defining weight coefficients, different sample points have varying importance, expanding MS' s application range.

Given data points distributed in a d-dimensional Euclidean space R^d, with multivariate kernel $K(x)$, symmetric positive definite bandwidth matrix H, the kernel density estimate at point x is [**?**]:

$$\hat{f}_K(x) = \frac{1}{n}\sum_{i=1}^{n} K_H(x - x_i) \tag{1}$$

where the kernel function $K(x)$ [**?**] satisfies

$$K(x) = c_{k,d}k(\|x\|^2) \tag{2}$$

where $c_{k,d}$ is a normalization constant ensuring the estimate integrates to 1. The gradient in Eq. (1) is

$$\nabla\hat{f}_K(x) = \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(x_i - x)g\left(\left\|\frac{x_i - x}{h}\right\|^2\right) \tag{3}$$

The mean shift vector is derived from Eq. (3) as

$$m_h(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x_i - x}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x_i - x}{h}\right\|^2\right)} - x \tag{4}$$

The mean shift vector always points toward the direction of maximum density increase. The mean shift process is formed through iterative steps: (a) compute

the mean shift vector $m_h(x)$, and (b) translate to a new center point $y_{i+1} = y_i + m_h(x)$.

The MS algorithm' s use of Euclidean distance treats differences between data attributes equally, which fails to meet practical requirements. Therefore, based on attribute importance, different weights are assigned to optimize Euclidean distance to weighted Euclidean distance, improving clustering performance. The weighted Euclidean distance between two points in d-dimensional Euclidean space can be defined as

$$D_w(x_i, x_j) = \sqrt{\sum_{k=1}^{d} w_k (x_{ik} - x_{jk})^2} \tag{5}$$

where w_k are weight coefficients for each attribute.

In this paper, MS' s new center point is given by Eqs. (3) and (5):

$$y_{i+1} = \frac{\sum_{i=1}^{n} x_i w_i g\left(\left\|\frac{x_i - y_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} w_i g\left(\left\|\frac{x_i - y_i}{h}\right\|^2\right)} \tag{6}$$

where h is kernel bandwidth, g is the kernel function, and w_k is the weight coefficient for the k-th attribute.

As shown in Eq. (5), weight coefficients are crucial for calculating the next center point and significantly impact clustering performance. Due to LFA' s unique characteristics, traditional MS algorithms show clear deficiencies. Therefore, this paper optimizes the MS algorithm for LFA detection by replacing traditional Euclidean distance with weighted Euclidean distance and using delay rate as the weight coefficient.

When LFA attacks occur in a network, the load on attacked links increases significantly, causing distinct differences in time variation between normal packets and LFA attack packets. Normal packets exhibit low and dispersed delay distributions, while LFA packets show high and continuous delay distributions.

Within a given time period T, the packet delay rate on the main link is defined as

$$D_T = \frac{1}{n} \sum_{i=1}^{n} \frac{|t_i - t_{i-1}|}{T} \tag{7}$$

where n is the number of packets. Therefore, within time period T, a higher packet delay rate in the network indicates greater LFA likelihood.

1.2 KNN Algorithm Principle

The KNN algorithm [**?**] involves the following basic steps: (a) calculate distances between samples; (b) sort unknown and training samples by increasing distance; (c) select the k points with smallest distances; (d) determine category frequency among the top k points; and (e) assign the most frequent category to the unknown sample.

KNN implementation depends on the "distance" between unknown and training samples. This paper uses Euclidean distance as defined in Eq. (5).

1.3 LFA Detection Method Based on MS-KNN Algorithm

Leveraging the advantageous properties of both MS and KNN algorithms, this paper combines them to create the MS-KNN method for effective LFA attack detection. The main steps are illustrated in [Figure 1: see original paper]:

(a) Preprocess the dataset, including data cleaning and normalization; (b) Initialize parameters; (c) Coarsen the data to avoid using very close sample points as initial centroids, obtaining candidate starting points; (d) Calculate Euclidean distances and Gaussian kernels from mean points to each sample point; (e) Compute weights; (f) Perform one independent mean shift iteration to calculate the next drift point coordinates; (g) Classify data into the nearest cluster based on nearest neighbors, obtaining k clusters; (h) Use grid search [**?**] and cross-validation [**?**] to find optimal k values, optimal weights, and optimal implementation methods for KNN on each cluster; (i) Analyze each cluster using the optimal KNN algorithm to obtain final results.

The time and space complexity of MS-KNN compared with MS and KNN are shown in , where n is the number of samples, T is the number of iterations, and k is the feature dimension of a single sample.

2.1 Experimental Environment Construction

The experimental setup used three Dell tower servers equipped with Intel Core i7-10700 2.90GHz 8-core processors and 16GB memory as compromised machines. Five computers with Intel Core i5-8400 2.80GHz 6-core processors and 16GB memory served as normal users. To ensure sufficient bandwidth between devices, three EdgeCore AS4610-54P gigabit Ethernet switches supporting OpenFlow protocol were selected as forwarding devices. For the controller, the RYU controller was deployed on a server with 2 Intel Xeon Gold 6248R 3.00GHz 48-core processors and 128GB memory. This server functioned as the entire network' s control plane, controlling bots to generate attack traffic and implementing network-wide data collection. After data collection, the control plane mixed traffic features from legitimate end hosts to generate the final dataset for experimental analysis. All experiments utilized 40 processor cores

allocated by the controller for parallel computation of results, including dataset clustering, grid search, cross-validation, and final data analysis.

A small network was designed based on literature [**?**], as shown in [Figure 2: see original paper]. The network topology includes compromised machines, SDN switches, decoy machines, a critical region, and the SDN control plane.

2.2 Dataset

Traffic from legitimate end hosts: CIC-IDS2017 [**?**] is a reliable dataset constructed by the Canadian Institute for Cybersecurity for testing and validation, as detailed in . This dataset contains benign and recent common attacks, similar to real-world data (PCAP). It includes network traffic analysis results using CICFlowMeter, with flows labeled based on timestamps, source and destination IPs, ports, protocols, and attack types (CSV files). Since it does not contain LFA flows, only legitimate flows were used, extracting feature sets from Monday' s legitimate end hosts and labeling them as positive.

Traffic from compromised machines launching LFA: As no public LFA dataset currently exists, this paper simulated LFA based on literature [**?**, **?**] to construct the dataset. The network state with LFA traffic over 200 seconds is shown in [Figure 3: see original paper]. Between 20-80 seconds, the network suffered LFA attacks with increased and significantly fluctuating throughput. From 80-135 seconds, LFA attack traffic decreased, leaving the network in a relatively safe state with reduced and stable throughput. From 135-185 seconds, LFA attack traffic increased again, causing throughput to rise with large fluctuations. After 185 seconds, the network slowly recovered to a safe state.

2.3 Experimental Results and Analysis

For preliminary evaluation of the dataset using different KNN implementations and weighting schemes, the dataset was split into training and test sets at a 70%/30% ratio. Performance was compared using four metrics: TPR, FPR, PPV, and ACC, as shown in [Figure 4: see original paper].

Figure 4: see original paper shows results under uniform weights using three KNN implementations. The experiments indicate that Figure 4: see original paper and Figure 4: see original paper outperform Figure 4: see original paper. Specifically, Figure 4: see original paper demonstrates that with uniform weights and ball tree implementation [**?**] at $k = 10$, TPR, PPV, and ACC reach 98.92%, 96.98%, and 96.75% respectively, while FPR drops to 10.99%. Figure 4: see original paper shows that with uniform weights and brute force implementation at $k = 3$, TPR, PPV, and ACC reach 79.66%, 96.89%, and 94.98% respectively, with FPR decreasing to 0.72%. While Figure 4: see original paper has slightly lower TPR, PPV, and ACC than Figure 4: see original paper, it achieves a much lower FPR.

Figure 4: see original paper presents results under inverse distance weights using three KNN implementations. All three show high TPR, PPV, and ACC. Figure 4: see original paper shows that with inverse distance weights and kd-tree implementation [**?**] at $k = 19$, TPR, PPV, and ACC reach 98.57%, 97.23%, and 96.69% respectively, with FPR at 10.01%. Figure 4: see original paper demonstrates that with inverse distance weights and ball tree at $k = 15$, TPR, PPV, and ACC reach 89.92%, 89.79%, and 95.55% respectively, with FPR dropping to 2.87%. Figure 4: see original paper shows that with inverse distance weights and ball tree at $k = 49$, TPR, PPV, and ACC reach 97.06%, 97.19%, and 95.51% respectively, with FPR at 10.01%. Compared with Figure 4: see original paper and Figure 4: see original paper, Figure 4: see original paper has slightly lower TPR, PPV, and ACC but achieves lower FPR.

compares the time consumption of different KNN implementations for preliminary dataset evaluation. The kd-tree implementation requires the least time, brute force requires the most, and ball tree falls in between.

The data shows that using KNN alone for dataset analysis, while effective for LFA detection, results in high FPR and long detection times. To reduce FPR and detection time, the MS-KNN method first clusters the dataset to partition it into multiple clusters, reducing data volume and detection time. Grid search and cross-validation then select optimal parameters for each cluster, which are used for final analysis to achieve optimal results and lower FPR. To reduce dataset impact on results, source/destination address features were removed, and experiments were conducted using protocol, packet length, and other features.

Optimal parameters for each cluster were obtained through grid search with different cross-validation methods. Results show that for two-fold, five-fold, and ten-fold cross-validation, the optimal KNN implementation is consistently ball tree with uniform weights, except for the first cluster under ten-fold cross-validation which uses inverse distance weights. The time consumed per cluster under different cross-validation methods is shown in [Figure 5: see original paper]. Therefore, this paper uses optimal parameters with uniform weights and ball tree implementation.

[Figure 6: see original paper] and [Figure 7: see original paper] show the final detection performance of MS-KNN on the dataset. [Figure 6: see original paper] presents the four evaluation metrics for each cluster: TPR, PPV, and ACC all exceed 99%, while FPR drops below 1%. Specifically, TPR, PPV, and ACC reach maximum values of 99.99%, 99.95%, and 99.98% respectively, with minimum FPR at 0.05%. [Figure 7: see original paper] shows the detection time per cluster using MS-KNN, which is significantly reduced compared with KNN processing time from . The comprehensive data demonstrates that compared with traditional KNN, MS-KNN achieves higher TPR, PPV, and ACC, lower FPR, and substantially reduced time overhead for LFA detection.

3 Conclusion

Link-flooding attacks on critical links cause severe consequences including link congestion and disconnection of target network regions. This paper proposes the MS-KNN method, which transforms traditional MS algorithm' s Euclidean distance into weighted Euclidean distance, using packet delay rate as the weight coefficient to optimize clustering performance and improve clustering effectiveness. Additionally, grid search and cross-validation select optimal KNN parameters for analyzing the LFA dataset. Experimental results demonstrate that MS-KNN achieves high TPR, PPV, and ACC, along with low FPR and detection time for LFA dataset detection.

While the proposed method demonstrates good effectiveness for LFA detection, several limitations remain. Future work will focus on constructing more complex network environments and studying comprehensive network traffic information features in more realistic physical settings. Additionally, research will address real-time dynamic network traffic detection to further validate the practicality of LFA detection methods.

References

[1] NSFOCUS. 2021 DDoS Attack Situation Report [R/OL]. (2022-02-09). https://www.nsfocus.com.cn/html/2022/92_{0209}/173.html.

[2] Studer A, Perrig A. The coremelt attack [C]// European Symposium on Research in Computer Security. Berlin, Heidelberg: Springer, 2009: 37-52.

[3] Kang M S, Lee S B, Gligor V D. The crossfire attack [C]// IEEE symposium on security and privacy. [S.l.]: IEEE, 2013: 127-141.

[4] Kang M S, Gligor V D, Sekar V. SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks [C]// [S.l.]: NDSS. 2016, 1: 53-55.

[5] Csikor L, Szalay M, Rétvári G, et al. Transition to SDN is HARMLESS: Hybrid architecture for migrating legacy ethernet switches to SDN [J]. IEEE/ACM Transactions On Networking, 2020, 28(1): 275-288.

[6] Chica J C C, Imbachi J C, Vega J F B. Security in SDN: A comprehensive survey [J]. Journal of Network and Computer Applications, 2020, 159: 102595.

[7] Isyaku B, Mohd Zahid M S, Bte Kamat M, et al. Software defined networking flow table management of openflow switches performance and security challenges: A survey [J]. Future Internet, 2020, 12(9): 147.

[8] 夏计强, 崔鹏帅, 李子勇, 等. 基于 P4 的 SDN 控制-数据平面流规则一致性校验 [J/OL]. 计算机应用研究: 1-6 [2022-03-30]. DOI: 10.19734/j.issn.1001-3695.2021.12.0694. (Xia Jiqiang, Cui Pengshuai, Li Ziyong, et al. P4-based rules consistency verification for SDN control-data plane. [J/OL]. Application Research of Computers: 1-6 [2022-03-30]. DOI: 10.19734/j.issn.1001-3695.2021.12.0694.)

[9] Singh J, Behal S. Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions [J]. Computer Science Review, 2020, 37: 100279.

[10] 贾锟, 王君楠, 刘峰. SDN 环境下的 DDoS 检测与缓解机制 [J]. 信息安全学报, 2021, 6(01): 17-31. DOI: 10.19363/J.cnki.cn10-1380/tn.2021.01.02. (Jia Kun, Wang Junnan, Liu Feng. DDoS detection and mitigation Framework in SDN. [J]. Journal of Cyber Security, 2021, 6(01): 17-31. DOI: 10.19363/J.cnki.cn10-1380/tn.2021.01.02.)

[11] Rezapour A, Tzeng W G. Rl-shield: mitigating target link-flooding attacks using sdn and deep reinforcement learning routing algorithm [J]. IEEE Transactions on Dependable and Secure Computing, 2021.

[12] Ravi N, Shalinie S M, Theres D D J. BALANCE: link flooding attack detection and mitigation via hybrid-SDN [J]. IEEE Transactions on Network and Service Management, 2020, 17(3): 1715-1729.

[13] Wang J, Wen R, Li J, et al. Detecting and mitigating target link-flooding attacks using SDN [J]. IEEE Transactions on Dependable and Secure Computing, 2018, 16(6): 944-956.

[14] Liaskos C, Ioannidis S. Network topology effects on the detectability of crossfire attacks [J]. IEEE Transactions on Information Forensics and Security, 2018, 13(7): 1682-1695.

[15] Xue L, Ma X, Luo X, et al. Linkscope: Toward detecting target link flooding attacks [J]. IEEE Transactions on Information Forensics and Security, 2018, 13(10): 2423-2438.

[16] Wang L, Li Q, Jiang Y, et al. Woodpecker: Detecting and mitigating link-flooding attacks via SDN [J]. Computer Networks, 2018, 147: 1-13.

[17] Kim J, Shin S. Software-defined HoneyNet: Towards mitigating link flooding attacks [C]// The 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). [S.l.]: IEEE, 2017: 99-100.

[18] Deb R, Roy S. A comprehensive survey of vulnerability and information security in SDN [J]. Computer Networks, 2022: 108802.

[19] Hande Y, Muddana A. A survey on intrusion detection system for software defined networks (SDN) [M]// Research Anthology on Artificial Intelligence Applications in Security. IGI Global, 2021: 467-489.

[20] Li W, Meng W, Liu Z, et al. Towards blockchain-based software-defined networking: security challenges and solutions [J]. IEICE Transactions on Information and Systems, 2020, 103(2): 196-203.

[21] Zhang S. Cost-sensitive KNN classification [J]. Neurocomputing, 2020, 391: 234-242.

[22] Zhang Y, Chen Y C. Kernel smoothing, mean shift, and their learning theory with directional data [J]. Journal of Machine Learning Research, 2021, 22(154): 1-92.

[23] Huang K, Fu X, Sidiropoulos N. On convergence of epanechnikov mean shift [C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2018, 32(1).

[24] Silverman B W. Density estimation for statistics and data analysis [M]. Routledge, 2018.

[25] LeCun Y, Bengio Y, Hinton G. Deep learning [J]. nature, 2015, 521(7553): 436-444.

[26] Marcot B G, Hanea A M. What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis? [J]. Computational Statistics, 2021, 36(3): 2009-2031.

[27] Sharafaldin I, Lashkari A H, Ghorbani A A. Toward generating a new intrusion detection dataset and intrusion traffic characterization [J]. ICISSp, 2018, 1: 108-116.

[28] Cui L, Zhang Y, Zhang R, et al. A modified efficient KNN method for antenna optimization and design [J]. IEEE Transactions on Antennas and Propagation, 2020, 68(10): 6858-6866.

[29] Xu Y, Yu Y, Hong H, et al. DDoS detection using a cloud-edge collaboration method based on entropy-measuring SOM and KD-tree in SDN [J]. Security and Communication Networks, 2021, 2021.

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