Postprint: Distributed SOM with K-Means Clustering for Flooding Attack Detection in Software-Defined Networks

Authors: Wang Haitao, Yu Songshen

Date: 2018-10-11T00:00:00+00:00

Abstract

To address the upper-layer performance bottlenecks and overload issues caused by flooding attacks in Software-Defined Networking (SDN), this paper proposes a network traffic attack detection method that combines Distributed Self-Organizing Map (DSOM) with K-means clustering. First, the DSOM controller located at the application layer sends the existing dataset to switches integrated with the DSOM extension package, training DSOM maps separately on each switch; then, the DSOM maps are merged within a predetermined time frame; finally, the DSOM controller sends the merged DSOM map to all OpenFlow switches, utilizing K-means clustering to complete the final classification. Experimental results demonstrate that the DSOM scheme can effectively detect anomalous traffic, resolve bottleneck issues, and offers certain advantages over traditional methods. Furthermore, the proposed method improves the system' s response speed to attack traffic while incurring minimal overhead to the network system.

Full Text

Preamble

Title: Software-Defined Network Flooding Attack Detection Method Based on Distributed SOM and K-Means Clustering

Authors: Wang Haitao[1], Yu Songsen[2]
[1]College of Information & Automation, Guangdong Polytechnic of Science & Trade, Guangzhou 510620, China
[2]School of Software, South China Normal University, Foshan, Guangdong 528225, China

Abstract: To address the performance bottleneck and overload problems in the upper layer caused by flooding attacks in software-defined networks (SDN), this paper proposes a network traffic attack detection method that combines distributed self-organizing maps (DSOM) with K-means clustering. First, the DSOM controller at the application layer sends the existing dataset to switches integrated with the DSOM extension package, where DSOM mappings are trained individually on each switch. Then, within a predetermined time, the DSOM mappings are merged. Finally, the DSOM controller sends the merged DSOM mapping to all OpenFlow switches, and K-means clustering is used to complete the final classification. Experimental results demonstrate that the DSOM scheme can effectively detect abnormal traffic and solve bottleneck problems, offering certain advantages over traditional methods. Additionally, the proposed method improves the system' s response speed to attack traffic while introducing minimal overhead to the network system.

Keywords: software-defined network; flooding attacks; distributed self-organizing map; K-means clustering; OpenFlow

0 Introduction

Software-defined networking (SDN) represents a promising network architecture for the future, where the separation of control and data layers offers significant benefits for network operators in traffic management. The OpenFlow protocol serves as a primary component in SDN, acting as the communication interface between SDN controllers and OpenFlow switches. Using OpenFlow, SDN controllers can configure and update flow tables within OpenFlow switches based on network service instructions such as security and routing policies.

In large-scale networks with numerous OpenFlow switches, the SDN controller naturally becomes a performance bottleneck due to the need for security applications at the upper layer to handle vast amounts of traffic information for resource allocation. Additionally, under high-traffic conditions, the communication channel between the data and control layers may also become a bottleneck, increasing latency and hindering application-layer traffic flow. Consequently, many researchers have introduced multiple controllers for large-scale SDN deployments, such as Google's B4 network. However, when facing flooding attacks or distributed denial-of-service (DDoS) attacks, these issues become critical vulnerabilities in large SDN architectures. Attackers send extremely large volumes of network traffic to victim systems to exhaust resources and degrade the quality of public services. As a result, security applications located in the upper layers of the SDN architecture must process substantial traffic information, and the system may collapse due to resource depletion and performance bottlenecks. Therefore, in large SDN models under high-traffic conditions, the data layer needs to support more advanced functions to reduce the enormous performance pressure on upper-layer security applications.

To address these problems and improve the robustness of large-scale SDN models, this paper proposes a software-defined network system based on clustering algorithms and distributed self-organizing maps. This scheme introduces clustering algorithms and DSOM mechanisms, distributing security modules across OpenFlow switches rather than at the control or application layers. These modules are controlled by a distributed system controller and operate as security applications within the SDN architecture' s application layer. Furthermore, extension modules have been implemented in OpenvSwitch, allowing connection to OpenvSwitch' s default agent for statistics collection, rule modification, and communication with the distributed system controller.

1.1 Software-Defined Networking and OpenFlow Protocol

Software-defined networking provides a promising network architecture for the future, where the control layer residing in the SDN controller is separated from the data layer. Applications running on the application layer can provide complex network services and functionalities. The data layer handles hardware-level operations and focuses on packet processing based on controller configurations. Communication between the control layer (SDN controller) and data layer (OpenFlow switches) is defined by the OpenFlow protocol. Before exchanging messages, OpenFlow switches must establish a secure connection to the SDN controller for authentication. This protocol enables the controller to access flow tables in OpenFlow switches for control, configuration, and statistics collection via secure connections.

1.2 K-Means Clustering

K-means clustering is a typical heuristic algorithm that partitions n data objects into k clusters, ensuring high similarity among objects within each cluster and low similarity between different clusters. Initially, the algorithm selects k objects from all n objects as initial cluster centers. The remaining objects are assigned to the nearest cluster based on their distance to different cluster centers. The algorithm then updates the centers of each cluster and repeats this process iteratively until an ideal cluster set is obtained.

1.3 Self-Organizing Map Algorithm

The Self-Organizing Map (SOM) is an unsupervised learning scheme in artificial neural networks. This algorithm transforms high-dimensional input space into a low-dimensional representation called an SOM map. SOM classifies or detects new input vectors based on two primary modes: training and mapping. The training process uses input samples to establish and reorganize the map, while the mapping process automatically classifies new input vectors by finding their winning neuron or node in the map.

The following parameters are defined: - m represents the number of iterations or training samples. - The map contains N neurons, each consisting

of a vector W_i composed of n weight values: $W_i = [W_{i1}, W_{i2}, W_{i3}, ..., W_{in}]$, where $1 \leq i \leq S$. - R is the radius of the rectangular map, defined as: $R = \max(\text{MapWidth}, \text{MapHeight})/2$. - λ is a time constant calculated as: $\lambda = N/\log(R)$. - $\sigma(t)$ is the neighborhood radius of the winning neuron (best matching unit) that gradually decreases over time, calculated as: $\sigma(t) = R \times \exp(-t/\lambda)$, for $t = 1, ..., N$.

The main steps of the self-organizing map algorithm are as follows:

a) Initialization: Initialize the n-dimensional weights of neurons with random or fixed values: $w_k = [x_{k1}, x_{k2}, x_{k3}, ..., x_{kn}]$.

b) Mapping: Feed input vectors into the map by calculating the Euclidean distance D_i from the input vector to all neurons w_s: $D_i = \sqrt{\sum_{j=1}^{n}(x_{kj} - W_{ij})^2}$. Select the neuron with the minimum distance D_i as the best matching unit (BMU) for input vector x_k.

c) Update BMU' s neighbor weights: Calculate the BMU' s neighbors according to equation (3). Then adjust the weights of these neighbors using the following equation for the next iteration to bring them closer to the input vector: $W_i(t+1) = W_i(t) + L(t) \times \Omega(t) \times (x_k - W_i(t))$, where $L(t)$ is the learning rate that also decays over time: $L(t) = L_0 \times \exp(-t/\lambda)$, and $\Omega(t)$ is the influence quantity of neighboring neurons' distance from the BMU on their learning: $\Omega(t) = \exp(-D_i^2/(2\sigma^2(t)))$.

d) Loop: Repeat steps (b) and (c) until no more input vectors are fed to the map.

1.4 Flooding Attacks from an SDN Perspective

In traditional network architectures, flooding attacks are generally divided into two categories: bandwidth exhaustion attacks and resource exhaustion attacks. In bandwidth exhaustion attacks, attackers flood the victim' s network with unwanted traffic to exhaust the victim's bandwidth, preventing normal traffic from accessing the victim system, such as ICMP flooding, UDP flooding, or Smurf and Fraggle attacks. In resource exhaustion attacks, attackers send abnormal IP packets or misused network protocol packets to the victim. Consequently, whenever the number of open connections reaches the system threshold, the victim network suffers resource exhaustion and may cease to function.

2 Distributed Network Traffic Attack Detection Method

To handle performance bottleneck issues when large-scale software-defined networks suffer flooding attacks, this paper introduces a distributed SDN system based on clustering algorithms. [Figure 1: see original paper] shows an overview of the proposed system, where all switches operate under the control of an SDN controller. In the application layer, an application called the DSOM controller is placed to control DSOM operations.

2.1 System Overview

In large-scale software-defined networks, to address performance bottlenecks caused by flooding attacks, this paper proposes a system based on clustering algorithms and distributed self-organizing maps. The introduced DSOM system is designed with four main processes:

a) Initialization: The DSOM controller located in the application layer sends the ready-made dataset to switches integrated with the DSOM extension package. On each switch, DSOM mapping is trained using the dataset received from the controller.

b) Merge DSOM: This stage is responsible for merging DSOM mappings within a predetermined time. The DSOM controller collects DSOM mappings from OpenFlow switches and generates a merged SOM map, represented as: $\text{MergedMap} = \sum_{j=1}^{n} \chi_j \times \text{MapDSOM}_j$, where χ_j is the ratio of training input samples for the j-th DSOM, and MapDSOM_j is the mapping of the j-th DSOM.

c) Update: The DSOM controller sends the merged SOM mapping to all OpenFlow switches. The merged SOM mapping replaces the MapDSOM_j mapping to continue the classification process on the switches.

d) Classification: The system calculates results based on their neurons for inputs and sends these results to the DSOM controller for further decision-making.

2.2 System Workflow

[Figure 2: see original paper] illustrates the DSOM switch layer. At the switch layer, several additional modules have been added to OpenFlow switches, called DSOM extension modules: flow collector, feature extractor, training database, DSOM mapping, fast policy enforcement, and DSOM switch agent. The DSOM switch agent acts as the core of the local switch-layer DSOM system, controlling the training and updating of the DSOM mapping process and sending classification results to the fast policy enforcement module and the DSOM controller.

During the initialization step, the training dataset sent from the DSOM controller reaches the DSOM switch agent via the OpenFlow channel of the OpenFlow switch. It is then forwarded to the training database and becomes the training input for the DSOM mapping. When the training process is completed, the DSOM switch agent sends start commands to the DSOM mapping, feature extractor, and flow collector to activate these modules. Similar to the working principle of this scheme, the flow collector periodically sends statistics messages to the switch data layer to obtain individual traffic information during the feature extraction phase, which focuses on feature extraction for each user passing through the switch. Additionally, the training database is updated by the output of the feature extractor, and the training DSOM mapping process is invoked again within the time period set by the DSOM switch agent. This

training enables the DSOM mapping to adapt to incoming traffic and improves the classification accuracy of each local DSOM switch.

To understand how many features are used in the input of the DSOM mapping, let us examine the feature extractor in greater detail. These features can be diverse for each type of network traffic; however, as previously described, this work considers common flooding attacks from an SDN perspective and divides them into two main types: bandwidth exhaustion attacks and resource exhaustion attacks. Therefore, this paper focuses on two types of flooding attacks, and the method involves observing each user within a predetermined time. The DSOM mapping performs the same operations as a single SOM. It determines which users are normal and which are abnormal. To make the final decision, the K-means clustering algorithm is applied to the system to partition the patterns into k clusters. After attack users are clearly identified, the DSOM mapping sends user information to the DSOM switch agent, which then transmits this data via the OpenFlow channel to the fast policy enforcement module. The fast policy enforcement module is placed on switches to enable rapid response to attack traffic by setting direct rules in the flow table, while the policy checking module located in the DSOM controller is responsible for validating rules and can override them if necessary.

3 Experiments

3.1 Dataset and Attack Simulator

3.1.1 Training and Testing Datasets A comprehensive set of SOM training and testing samples was analyzed and constructed from three datasets: CAIDA, NSL-KDD, and DARPA. Brief descriptions of these datasets are summarized as follows:

The CAIDA dataset enables mixed research on various traffic types such as Web, FTP, and Ping. As shown in , TCP and ICMP protocols typically constitute the highest proportion of network traffic. For the NSL-KDD dataset, only DoS attacks were studied. Therefore, only DoS samples were considered for training and testing the proposed system, as shown in .

3.1.2 Attack Simulator Testing was conducted using BoNeSi, which can simulate botnet traffic and generate TCP, ICMP, or UDP flooding attacks to target networks or specified IP addresses through defined botnets.

3.2 System Setup in SDN

The network topology used for experimental testing is shown in [Figure 3: see original paper], consisting of one controller, four OpenvSwitches integrated with DSOM modules, one web server, and three hosts (h1, h2, h3, h4) as traffic generators. Detailed parameters for the DSOM mapping are provided in . Additionally, since most user accesses are normal and legitimate, the number of

cluster centers was set to 4 to form a set of normal users.

3.3 Performance Testing

3.3.1 Performance Bottleneck Testing To evaluate the effectiveness of the proposed system compared to a single SOM in terms of performance bottlenecks, a system was established as shown in [Figure 3: see original paper], and two test instances were conducted: single SOM and DSOM system. In the first test scenario, a SOM module was implemented in the controller, which contained four functions: flow collector, feature extractor, SOM mapping, and policy enforcement. In the proposed system, only the DSOM controller module ran as an application in the SDN controller, while other modules were implemented in the switches. In both instances, the same dataset with a total of 12,000 samples was used to train the SOM mapping, with 4,000 patterns randomly extracted from each dataset. The BoNeSi tool was then used to generate two traffic levels (50 Mbps and 100 Mbps) to attack the web server, and CPU utilization of the SDN controller was measured in both cases.

3.3.2 SOM Performance Testing For performance evaluation of the SOM mapping in the proposed solution, the following setup was implemented to assess the DSOM system and demonstrate that the proposed method achieves the same performance as traditional SOM mechanisms:

Samples (including normal and attack patterns) were extracted and randomly selected from pcap files of the three datasets for testing. The SOM mapping initialization and training process were performed on separate datasets (CAIDA, NSL-KDD, and DARPA). The testing procedure was then executed using 5-fold cross-validation from the training dataset and traffic from the BoNeSi DDoS simulator. Multiple test runs were conducted with different numbers of neurons in the SOM mapping: 400, 900, and 1600.

4 Results and Discussion

4.1 Processing Performance Bottleneck

[Figure 4: see original paper] shows the CPU utilization of the SDN controller, revealing significant differences between test instances. The single SOM mechanism consistently consumes more CPU resources from the SDN controller, while the DSOM system maintains stable CPU consumption. In the 50 Mbps traffic generation instance, CPU utilization reached 40% and 60% just 12 seconds and 33 seconds after attack initiation, respectively. Meanwhile, the DSOM solution occupied only about 23% of CPU resources throughout the entire test period. The same difference is observed in the 100 Mbps scenario: the single SOM scheme achieved 60% CPU utilization after only 10 seconds and fluctuated around this value until the test ended. In contrast, the DSOM solution maintained consistently low CPU occupancy.

These results demonstrate that the DSOM system is an excellent solution that completely outperforms the single SOM scheme. In the single SOM case, the controller must frequently send messages to switches to obtain traffic information and forward it to subsequent modules for further processing. Simultaneously processing traffic information from all edge switches represents a heavy workload for the modules. The DSOM system delegates traffic processing tasks to edge switches to reduce workload, processing only a small amount of information during policy checking. Additionally, each edge switch only processes traffic entering its ports from external networks. Therefore, compared to the single SOM scheme, the pressure on DSOM switch agents is not significant.

4.2 SOM Performance Evaluation

Before evaluating the performance of the DSOM method, this paper defines the following metrics: TP represents the probability that attack users are classified as attack users, TN represents the probability that normal users are considered normal, FP represents the probability that abnormal users are considered normal, and FN represents the probability that normal users are identified as attack users.

4.2.1 Detection Rate To evaluate the efficiency of the proposed DSOM system, a key criterion considered is the detection rate, calculated as: $DR = \frac{TP}{TP+FN}$.

[Figure 5: see original paper] shows the detection rates of SOM mappings across four different datasets under three scenarios. The results indicate only slight differences among the three cases across all four datasets with the same number of neurons. This is because the K-means clustering algorithm is used to distinguish between normal and abnormal users, and the detection rate is evaluated using k-fold cross-validation techniques on the datasets themselves. However, even when using attack tools to generate traffic, the trained SOM can easily detect attack users due to the large number of training samples. These experiments also demonstrate that if the SOM mapping has more neurons, it means higher detection performance, and there is not much difference between the proposed DSOM and single SOM systems.

4.2.2 Accuracy Accuracy measures how accurately the SOM mapping makes decisions and is calculated as: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$.

[Figure 6: see original paper] shows the accuracy of SOM mappings across four different datasets under three scenarios with different numbers of neurons. All three experiments achieved high accuracy. With 400 neurons, the lowest accuracy was approximately 96.9%, while with 900 and 1600 neurons, accuracy exceeded 97%. The reason is that when the number of learning samples for the SOM mapping increases, the accuracy of decisions made about users also becomes more accurate. Moreover, when the SOM mapping reaches a threshold where it can make completely accurate decisions, the number of neurons has

little impact on accuracy, which explains why there are only minor differences among these tests in terms of precision.

4.2.3 System Overhead System overhead was evaluated by measuring processing time and classification time for both systems. For the DSOM system, assuming the time required to transmit data through packet connections is only a few milliseconds, the processing time is mainly calculated through the sum of two processes: the first is training the DSOM mapping on OpenvSwitch, which occupies the most time, and the second is the merging time on the DSOM controller. Meanwhile, the processing time for single SOM is only its training time. The DSOM and single SOM mappings were trained with input sets as shown in .

The results in demonstrate that if a larger number of neurons is used in the SOM mapping, the DSOM system' s processing time is faster than that of single SOM. This is because OpenvSwitch' s DSOM uses a smaller dataset for training, while single SOM always requires a larger input dataset to train its mapping. Regarding classification time, the three scenarios are roughly equal and increase proportionally with the number of neurons. This can be explained by using the same configuration for the controller and OpenvSwitch; therefore, there is not much difference in CPU processing time among the three scenarios.

Based on these experimental results, the DSOM system in software-defined networks can be evaluated. First, in terms of processing performance bottlenecks, the DSOM system is an excellent solution that completely outperforms the single SOM scheme. Second, both systems demonstrate excellent performance in detection rate and accuracy for abnormal traffic. In terms of overhead required to prepare the system for the classification process, the proposed DSOM is less than that of using single SOM. Overall, the DSOM system in software-defined networks can effectively solve bottleneck problems under flooding attacks in large networks and offers certain advantages compared to traditional methods.

5 Conclusion

This paper proposes a K-means clustering algorithm and distributed self-organizing map system that analyzes network data using a distributed system rather than a centralized one to address bottleneck problems caused by aggregation in the upper layer due to flooding attacks in large software-defined networks. Experimental results demonstrate that the proposed method completely outperforms traditional methods in processing performance bottlenecks. Moreover, the results not only show that the mechanism achieves the same performance as a single SOM system but also indicate that the DSOM system' s overhead is superior to that of single SOM.

References

[1] Hu Tao, Zhang Jianhui, Mao Ming. Controller load balancing strategy based on migration optimizing in SDN [J]. Application Research of Computers, 2018, 35(2): 559-563.

[2] Qiu Xinyi, Li Jun, Zhou Jianer, et al. New delay measurement method based on accurate timestamp in OpenFlow [J]. Journal on Communications, 2017, 38(11): 178-187.

[3] Wang Haopei, Xu Lei, Gu Guofei. FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks [C]//Proc of International Conference on Dependable Systems and Networks. Washington DC: IEEE Computer Society, 2015: 239-250.

[4] Jain S, Kumar A, Mandal S, et al. B4: experience with a globally-deployed software defined wan [J]. Computer Communication Review, 2013, 43(4): 3-14.

[5] Wang Qilin, Li Xiaopeng, Yu Bin, et al. Defense scheme of flooding attacks in bluetooth low energy based on connection authentication [J]. Application Research of Computers, 2017, 34(2): 499-502.

[6] Zhang Yongzheng, Xiao Jun, Yun Xiaochun, et al. DDoS attacks detection and Control mechanisms [J]. Journal of Software, 2012, 23(8): 2058-2072.

[7] Lara A, Kolasani A, Ramamurthy B. Network innovation using openflow: a survey [J]. IEEE Communications Surveys & Tutorials, 2014, 16(1): 493-512.

[8] Jafarian J H, Al-Shaer E, Duan Qi. Openflow random host mutation: transparent moving target defense using software defined networking [C]//Proc of Workshop on Hot Topics in Software Defined Networks. New York: ACM Press, 2012: 127-132.

[9] Sulaiman S N, Nor Ashidi Mat Isa. Adaptive fuzzy-K-means clustering algorithm for image segmentation [J]. IEEE Trans on Consumer Electronics, 2010, 56(4): 2661-2668.

[10] Zhang Shunlong, Ku Tao, Zhou Hao. Accelerate K-means for multi-center clustering of big datasets [J]. Application Research of Computers, 2016, 33(2): 413-416.

[11] Wu Yi, Yang Qiong, Wu Qingxiang, et al. Networking algorithm based on self-organizing map neural network for VANET [J]. Journal on Communications, 2011, 32(12): 136-145.

[12] Zhang Shuguang, Zhou Xuehai, Yang Feng, et al. Traceback Mechanism Based on Neighbor Information in Wireless Sensor Networks [J]. Journal of Chinese Computer Systems, 2015, 36(3): 483-487.

[13] Wen Yuhao, Wang Han, Chen Zhen, et al. MASC: A bitmap index encoding algorithm for fast data retrieval [C]//Proc of IEEE International Conference on Communications. 2016: 1-6.

[14] KumarShrivas A, Dewangan A K. An Ensemble Model for Classification of Attacks with Feature Selection based on KDD99 and NSL-KDD Data Set [J]. International Journal of Computer Applications, 2014, 99(15): 8-13.

[15] Moustafa N, Slay J. Creating novel features to anomaly network detection using DARPA-2009 data set [C]//Proc of European Conference on Cyber Warfare and Security Eccws. 2015: 23-24.

[16] Ye Jianfeng, Wang Huaming. An automatic face recognition method using AdaBoost detection and SOM [J]. Journal of Harbin Engineering University, 2018, 39(1): 129-134.

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