Acta Technica *62* (2017), No. 6B, 971–984 © 2017 Institute of Thermomechanics CAS, v.v.i.

Network traffic prediction based on wavelet decomposition and sparse matrix vector multiplication

Fu Hongwei[1]

Abstract. In order to improve the performance of the software defined network (SDN) suspicious traffic detection algorithm, a k nearest neighbor SDN suspicious traffic detection method based on undirected graph process creation and information aggregation inference is proposed. OpenFlow module is used to create data flow and build intrusion rules. Then, based on the graph node/edge representation of Markov chain, the attack feature representation is performed in undirected graph, the incremental representation of new attack is realized, the computational complexity of undirected graph construction is reduced, and the k nearest neighbor algorithm is used to classify the malicious attack traffic features of undirected graphs, so that the attacks can be detected effectively. Finally, the performance of the proposed algorithm is verified by the constructed SDN test platform.

Key words. Network traffic monitoring, Undirected graph, Information aggregation, Inference system, Sparse matrix, Wavelet.

1. Introduction

In the network security, the autonomic network agents can learn the network topology and invasion data features, to realize the network rules updating, and to realize the effective detection of the intrusion data under the minimum human input. In programmable networks, such as software defined networks, it may take longer time to build network agents, thus realizing the complete self-management of network [1, 2].

Still, professional scholars believe that SDNs is very vulnerable, and can be easily attacked [3]. When the logical behavior of forwarding behavior is centralized and distributed in the controller, a single fault point and attack point are constructed. Whether it exploits vulnerabilities in the controller or communication links between

[1]Basic Public College, University, Jinhua Polytechnic, Jinhua, Zhejiang, China, 321000

switches and controllers, it may cause some attack behavior, such as denial of service (DoS) and host location hijacking attacks, etc., and it needs to pay great attention to SDNs security [4]. However, there are some challenges to SDNs security research; first, it is necessary to create more complex security measures, not just relying on the method of discarding or forwarding traffic. Specifically, generation of statistical and signature matching techniques needs to combine data flow rules. Second, existing programs for creating secure applications depend on the architecture of the controller that manages the network. Third, because of the dynamic nature of SDNs traffic processing, it is difficult to detect different types of attacks by using anomaly detection technology, which can lead to high false alarm rate.

In order to solve these problems, a new intrusion detection algorithm is proposed in this paper, which can effectively recognize SDNs networks. The proposed algorithm utilizes the OpenFlow model, and is based on the graph node / edge representation model of Markov chain and the k nearest neighbor query algorithm. The main innovation points of this paper:

(1) The feasibility of using existing attack features to identify SDNs attacks is demonstrated.

(2) A flexible approach has been created, able to use label-containing events to build attack prediction models at different levels to determine the SDNs attack data set.

(3) It is proved that this method can effectively realize the attack recognition of SDNs by querying the pre-created graph at runtime. Experimental results show that the proposed method is superior to the traditional classifier and the existing SDNS attack classification technique.

(4) An incremental creation technique for graph models is proposed to associate with existing nodes by adding new types of nodes.

2. System model

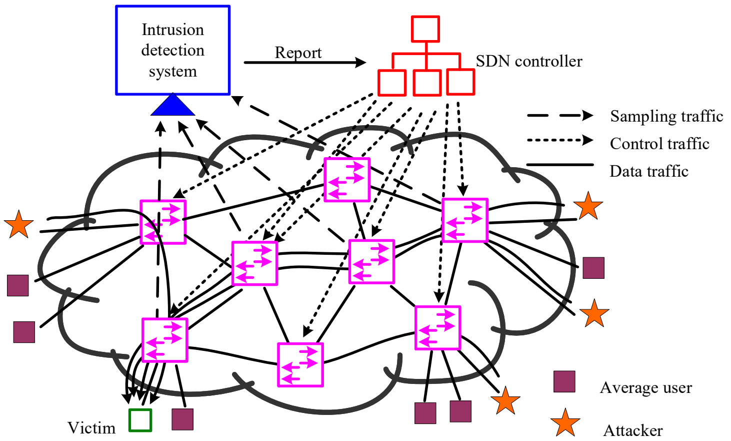

Fig. 1. Software defined network

Considering the SDNs as shown in Fig. 1, a SDN-based network architecture consists of a SDN controller and a OF enable switch. OF is a communication network protocol that allows the SDN controller to access the forwarding tables of the switches. On the control plane, the SDN controller is directly connected to the OF enable switch, which allows control and collection of forwarding table traffic on each switch. The SDN controller implements the switching operations on all packets by the enable switch [5, 6].

Fig. 1 shows the IDS based on SDN network for suspicious traffic checks. For SDN-based IDS traffic monitoring, it can use the mirror method to sample packet at any enable switch, so that SDN can be configured completely. IDS checks all mirrored packets of the switch and, if suspicious packets of network traffic are detected, alerts are generated and fed back to the SDN controller. Based on the switch status and the IDS check result, the SDN controller can reconfigure the network to protect against network attacks.

Supposed that there are f traffic packets and N network enable switches. Make λ_i indicate the malicious attack rate belonging to the i data flow. If the data flow does not contain any attack packets, its malicious attack rate is 0. The transmission rate of data flows can be represented by a transmission rate vector: $s = [s_1, \cdots, s_f]^T$, and the malicious traffic can be expressed as: $\lambda = [\lambda_1, \cdots, \lambda_f]^T$. The transmission unit of malicious rate is packet/sec.

3. Inference-based network suspicious traffic detection

3.1. Attack prediction system framework

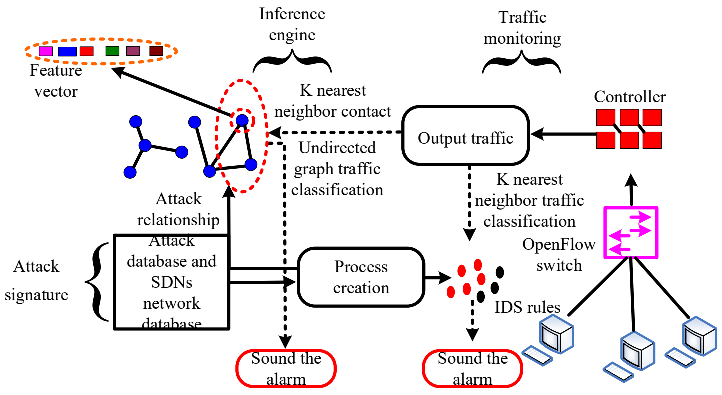

Fig. 2. Attack prediction system framework

Different samples were analyzed by using regular traffic markers. The purpose of the proposed method is defined as: (1) check if there is an attack signature that can be used to analyze the traces produced by SDNs. (2) Study how the traditional suspicious traffic detection system deals with a large amount of traces produced by SDNs. Fig. 2 shows the framework of the attack prediction system presented in this paper[7, 8].

In Fig. 2, the traffic monitoring module uses OpenFlow traffic for customized traffic reception and regular traffic queries based on the requirements of the network or switch. Then, an attack prediction model can be used to analyze and output the data flow records; the attack feature extraction module is used to represent the attack signatures extracted from the network data set. The data stored in the network attack database is mixed and NetFlows is generated by the flow creation mould.

3.2. Flow creation mould based on OpenFlow

Algorithm 1: flow creation based on OpenFlow information aggregation

```
Input: data flow duration, default 30 seconds, feature: extract feature list from packet,
packet, traces of OpenFlow network;
1. procedure createFlow(duration, features, packets)
2. Initialize a list of packets $S = \emptyset$;
3.   for all packet do
4.     if $S = \emptyset$ then packet $\rightarrow S$ else
5.       duration do if the starting time of packet and the duration of $S$ is larger than
duration
6.         The first data flow is output, and the first data flow is read from the $M$;
7.         Remove the first data flow from $S$;
8.         Remove the first data flow from $M$;
9.     endif
10.      Initialize $flag = False$
11.      for packet $\in S$ do
12.        if $feature_{packet} = feature_{startpacket}$ then
13.          $M.get\,(id\ of\ startPacket)\,.add\,(packet)$;
19.          Set $flag = True$
20.        endif
21.      endfor
22.      if $flag = False$ then
23.        packet $\rightarrow S$;
24.        $M.put\,(id\ of\ packet,\ add\ packet\ to\ the\ list)$;
25.      endif
26.    endif
27. endfor
28. if $M \neq \emptyset$ then
29.   Output all data flow in $M$;
30. endif
31. end procedure
```

Hardware and software are needed to implement the collection and output of data flows in the packet traces. Data flows include specific-function packet sharing, such as generating time, data sources, and target IP[9]. In the algorithm proposed by this paper, ten features of packets are involved: target IP and data source, target port and data source port, bytes and number of packets, ending and starting time, TCP mark and flow time. In algorithm 1, the creation steps of aggregated packet traffic are given.

The procedure is a preprocessing step that executes only when the data set stored in the attack database is not streaming data. Use the duration parameter to adjust the presence time of each flow.

3.3. Suspicious flow detection rule

IDS rules are accessed or updated by a particular machine learning technology or network administrator. Our approach is to analyze OpenFlow-based data flow to determine whether suspicious attacks exist. Here, the flow classification model is used to create this set of rule. These rules are created by using the usual flow stored in the attack database. Suppose $\bar{V} = [f_1 : d_1, \cdots, f_n : d_n]$ to be a set of features extracted from network traces, as shown in Fig. 2. Suppose the flow classification model m to detect $\bar{V}$. If $\bar{V}$ matches with any signature defined by m, then an alarm is issued. Supposed $N = \{n_1, n_2, \cdots, n_p\}$ indicating all the sets that may give out the alarm as determined by m. Assuming that *NS* is a label set of pre-existing suspicious traffic, *NB* is a label set of pre-existing benign flows, and any flow classification model used needs to realize two goals during the running: (1) increase the detection rate of suspicious activity under the condition that the *NS* set is correctly labeled. (2) Reduce the false alarm rate of benign flow when the given *NB* set is correctly labeled.

3.4. Node path inference

Each node n_i is represented by a specific feature (namely the corresponding data flow features). Each tag in the collected data is considered as a node in the graph. Each node n_i is represented by a single feature vector V_{n_i}, as shown in Table 1.

Table 1. Correlation feature vectors between nodes

Feature vector	f_{1_b1}	f_{1_b2}	f_{2_b1}	f_{2_b2}	f_{n_b1}	f_{n_bj}
V_{ni}	0.91	0.66	0.37	0.44	0.54	0.64
V_{nj}	0.82	0.42	0.55	0.41	0.42	0.64

Each vector contains the aggregation weight of the feature source and target IP, port number, number of packet, TCP flag, protocol type, and alert description. The normalized frequency f_{i_bj} of each feature will serve as its weight at the node. Since these features are represented in numerical form, discrete stage division methods with equal widths can be used, and the feature f_i is represented as the form of $b_1, b_2, \cdots, b_j$.

Pearson correlation is used to calculate the correlation between nodes, as shown in formula 1.

$$sim_{n_i,n_j} = \frac{\left(cov\left(\sigma_{V_{n_i}}, \sigma_{V_{n_j}}\right)\right)}{\sigma_{V_{n_i}} \times \sigma_{V_{n_j}}} \frac{e\left(\left(V_{n_i} - \mu_{V_{n_i}}\right)\left(V_{n_j} - \mu_{V_{n_j}}\right)\right)}{\sigma_{V_{n_i}} \times \sigma_{V_{n_j}}} . \tag{1}$$

The resulting weights are normalized and then assigned to the edges of the connected nodes.

In the process of indirect path inference and discovery between nodes, the similarity graph is created based on the length of traversal path. The one with the highest numerical value is chosen as the most feasible path, indicating the possibility of relationship between any two nodes. Fig. 3 shows a legend with four nodes.

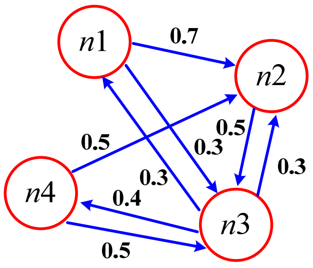

Fig. 3. Graph node/edge representation based on markov chains

The value displayed on the edge of a graph indicates the transition probability from one node to another. The most probable relationship can be determined by following steps:

Step1: Using the sum of products for the weights and the length $|t|$ of all paths of the connection node (n_i, n_j), the fraction l is calculated.

Step2: According to the number of paths $l = N_{n_i - n_j}^{|t|}$ with node (n_i, n_j) length of $|t|$, with different $|t|$ value, the value of fraction l will be different, to select the maximum value as the node link with the most possibility.

When a new node n_i is found, there is no need to regenerate the entire graph. The following steps can be used to simplify:

Step1: Find the similarity between the new node n_i and all other nodes in the existing graph.

Step2: According to similarity values, k nearest neighbor algorithm is used to select the node that is the closest to the new node n_i in the existing graph.

Step3: The average l_{avg} of the fraction l for the k nodes selected by the above k nearest neighbor algorithm is calculated,

Step4: If the k nodes selected belongs to c_i, all nodes belong to the same attack type.

$$l_{(n_i \to n_j)} = l_{avg} + \left(|1 - l_{avg}| \times sim\left(n_i, n_j\right)\right) . \tag{2}$$

If the k nodes selected does not belong to c_i, calculate:

$$l_{(n_i \to n_j)} = l_{avg} - \left(|1 - l_{avg}| \times sim\left(n_i, n_j\right)\right) . \tag{3}$$

3.5. K nearest neighbor traffic classification of undirected graph model

K nearest neighbor classifier [10], uses the similar functions to find *k* approximate nodes, and Fig. 4 shows an instance of each node using feature vectors.

Each feature vector represents a configuration file containing the highest weight feature of the node. The conditional entropy is used to assign the feature weights to create configuration files. Data flows are classified according to the main node types in the nearest neighborhood of the nodes; for example, if $k = 1$, the data flow is classified according to a single node type. Each data flow is classified as a specific type of DoS attack or benign activity. Use undirected graph model for data flow classification, specifically see the step 2 of the algorithm.

Algorithm 2: undirected graph model data flow classification

```
Input: data flow set $FL = \{fl_1, fl_2, \cdots, fl_n\}$, $l\,(n_i, n_j)$,$\forall n_i, n_j \in N$
1. Procedure Flow classification($FL$, $k$, $l$);
2. for i=1:$n$ do
2.     $R = \{\emptyset\}$, $suspiciouscount = 0$, $benigncount = 0$;
3. Find the node $n_i$ that is the most similar to $fl_i$ based on the features.
4. then if $n_i$ is the suspicious node then
       $suspiciouscount = suspiciouscount + 1$;
5. else $benigncount$=$benigncount + 1$;
6. endif
7. for $j = 1 : k$ do
8.   Search $topkl\,(n_i, n_j)$;$R = \{R\} \cup n_j$;
9.      If $n_j$ is the DoS alarm then
10.         $suspiciouscount = suspiciouscount + 1$;
11. else
12.        $benigncount$=$benigncount + 1$;
13.        endif
14.  endfor
15.    $classifyF(R, fl_i)$;
16. endfor
17. end Procedure
Function $classifyF(R, fl_i)$
1. Input: a set R and data flow $fl_i$ with suspicious and / or benign nodes;
2. Returns: the type of each data flow $fl_i$;
3. Check the number of benign and suspicious nodes in R;
4. if $benigncount > suspiciouscount$ then
5.    Return to “benign data flow";
6. elseif $benigncount < suspiciouscount$then
7.    Return to “Dos attack data flow”
8.    $R = (R\backslash k)$;Else, remove$k$ nearest neighbor nodes from R; $R = (R\backslash k)$
9. $classifyF(R, fl_i)$
10. endloop
11.end Function
```

The algorithm requires three inputs, namely the data flow set ***FL*** that needs to

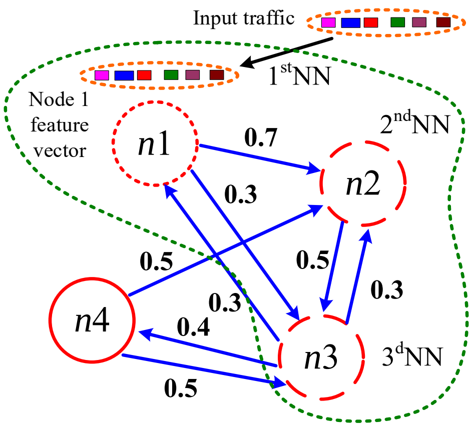

Fig. 4. Approximate node discovery

be processed, the nearest neighbor number k, and the fractional value l. Use data flow features to select k nearest neighbor nodes in graphs. Each node in the graph is represented as a feature vector, used to discover the similarity features of the incoming data flows (line 3). Compared with the k nearest neighbor classifier, once the nearest node n_i of the input flow fl_i is found, the k nearest neighbor node n_i is queried by using the fractional value l(line 8).

4. Experimental analysis

4.1. Experiment platform

In order to verify the effectiveness of the algorithm, a SDN test platform is constructed, and the algorithm performance test is realized on the test platform. This experiment table is composed by HP Z420 workstations, equipped with Intel Xeon e5-1620 processor, the operating environment is Ubuntu 14.04.2 LTS, as a SDN controller, the 6 Odroid-xu3 embedded motherboard acts as the SDN switch, with SDN switch, and the DELL OptiPlex desktop computer acts as IDS, and the topology is shown in figure 5. The details of the test platform are as follows:

(1)SDN controller: there are multiple realization methods of SDN controllers, such as FloodLight, ONOS, and OpenDaylight (ODL). Among them, we chose the ODL as the SDN controller of the test platform. ODL is an open-source project written by Java. It supports the OpenFlow protocol and provides the northbridge API (REST API) and the southbridge API (OSGi) of programming interfaces.

(2)SDN switches and traffic sources: the SDN and SDN enable switches are constructed. The Odroid-xu3 embedded board is mounted with Open vSwitch (OVS) software as the SDN switch. Since each board is equipped with only one single Ethernet port and a USB hub, here USB is added to the Ethernet adapter to obtain four additional Ethernet ports. One is connected to the SDN controller, the other is used for mirroring traffic sampling, and the others are used to construct data network topologies.

(3)IDS configuration: Snort and Suricata are the most popular open-source suspicious traffic detection systems. In the test, the malicious traffic generated by VMs is detected by using Snort. Assume that the attacker uses the port number 10000 to attack port scanning.

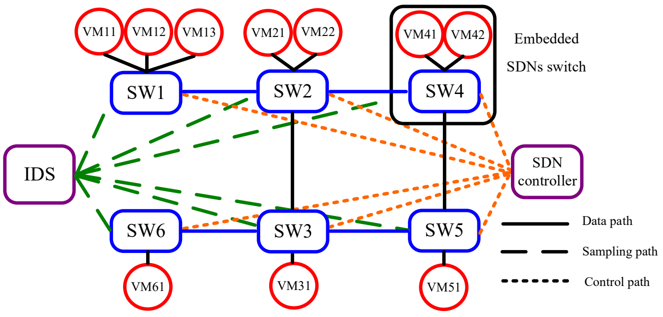

Fig. 5. SDN test platform topologies

4.2. Result analysis

There are two kinds of database used in the experiment: (1) CAIDA data set containing DOS attacks; this data set contains anonymous traces of the network. The attacks contained mainly aim at consuming computing resources to prevent users from accessing the server. This dataset does not have a tag for each packet, so Snort IDS analysis is performed on the data set to determine packet type. (2) Data set containing suspicious network connections; the data set contains a network connection packet of data sources and target IP, data sources and target ports, protocols and grouping type features.

Perform preprocessing steps on the traffic generated by each data set. By changing the flow parameters, different types of attacks and benign activities are selected for the testing process. The number and type of nodes used in the experiment are shown in Table 2.

Experiment 1: (parameter k performance influence) the algorithm in this paper uses k nearest neighbor algorithm, and parameter$k$ has a great impact on the algorithm. The standard k nearest neighbor algorithm is chosen to compare the influence of different values of parameter k ($k = 1 \sim 6$) on detection performance. Comparison indexes select the three indicators of detection precision, recall rate and F-score, and the experimental results are shown in Fig. 6.

According to the results of Fig. 6, the parameter k will have a more pronounced impact on the algorithm, but the best result is not obtained when its value is the largest or smallest, but when k=3 or so; it can be seen that for both data sets, the best result is obtained when k=3, which shows that k=3 has certain universality. At the same time, we can see that the performance indexes of the proposed algorithm are better than the selected standard k nearest neighbor algorithm under the condition of k with various parameters.

Table 2. Experimental test data

Type	Data set 1			Data set 2		
	Training sample number	Test sample number	Node number n	Training sample number	Test sample number	Node number n
TCP/SYN Flood	6213	3217	1	8315	1215	1
UDP Flood	–	2513	–	5321	–	4
ICMP Flood	6231	3589	2	7256	2613	2
Total	12444	9319	–	20892	3828	–
TCP/SYN Benign traffic	2713	1102	1	4511	3652	1
UDP Benign traffic	–	–	–	6623	–	4
ICMP Benign traffic	5113	4102	2	7214	2213	2
Total	7826	5204	–	18348	5865	–
Suspicious and benign	20267	14516	–	39241	9687	–
suspicious Proportion of suspicious flows	0.62	0.63	–	0.45	0.62	–
benign Proportion of benign flows	0.37	0.34	–	0.52	0.38	–

Experiment 2: (parameter k calculation efficiency influence) the standard k nearest neighbor algorithm is chosen to compare the influence of different values of parameter k ($k = 1 \sim 6$) on computational efficiency. Compare the running time of the algorithm selected by the index, and the experimental results are shown in Fig. 7.

According to the results shown in Fig. 7, the parameter k has obvious influence on the computational efficiency of the algorithm, and shows obviously monotone increasing feature; the larger the value of parameter k, the longer the computation time of the algorithm; therefore, in combination with the results shown in Fig. 6, it is appropriate to select the value of the parameter to be k=3. At the same time, we can see that the computational efficiency of the proposed algorithm is much better than that of the standard k nearest neighbor algorithm.

Experiment 3: (algorithm comprehensive performance comparison) in order to fully verify the effectiveness of the algorithm, the comparison algorithms select SVM algorithm, SOM algorithm and SVM & PCA algorithm. Comparison indexes select three indicators of detection precision (P), recall rate (R) and F-score (F), the data

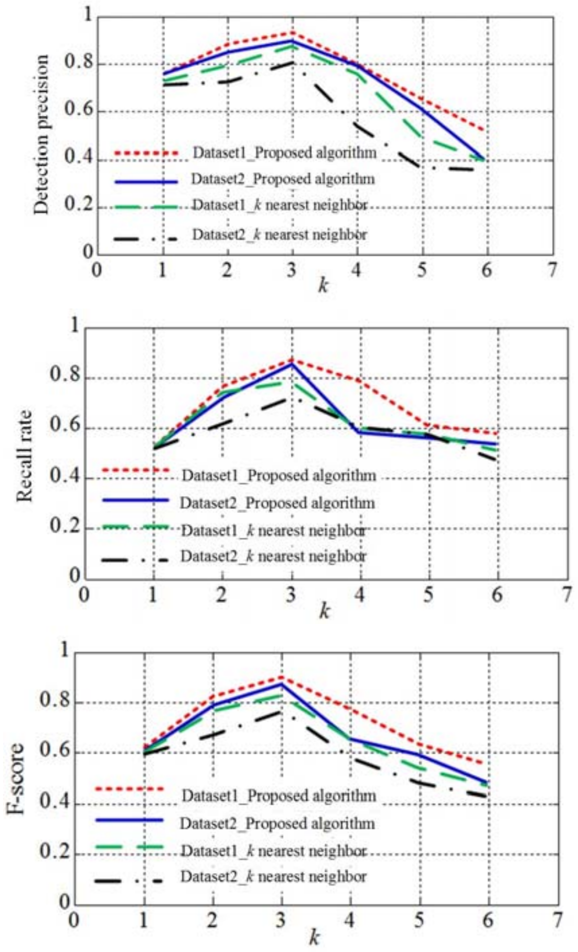

Fig. 6. Detection performance influence by parameter k

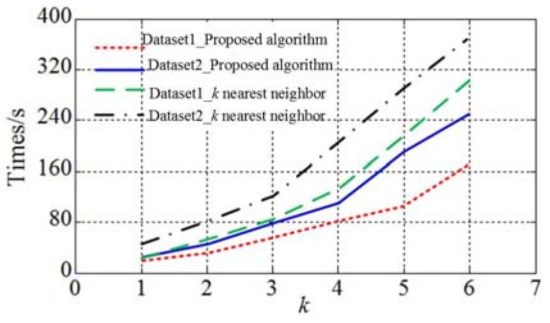

Fig. 7. Computational efficiency influence by parameter k

set selects the above two data sets, and the experimental results are shown in Table 3.

Table 3. Comprehensive performance comparison of algorithms

Comparison algorithm	Data set 1			Data set 2		
	P	R	F	P	R	F
Proposed algorithm	0.91	0.86	0.88	0.87	0.83	0.86
SOM	0.72	0.68	0.69	0.76	0.75	0.76
SVM&PCA	0.77	0.73	0.75	0.82	0.76	0.78
SVM	0.78	0.65	0.73	0.76	0.75	0.78

According to the experimental results of Table 3, in the index of comparison algorithm, we can see that the proposed algorithm is better than the selected SVM algorithm, SOM algorithm and SVM & PCA algorithm, the detection performance of SVM & PCA algorithm is better than that of SVM algorithm and SOM algorithm, while the SVM algorithm and the SOM algorithm have little difference in detection performance, and they have their own advantages; the above results verify the strong detection ability for the malicious attack of the proposed algorithm.

5. Conclusion

In this paper, a method of k nearest neighbor SDN suspicious traffic detection based on undirected graph process creation and information aggregation inference is proposed, using the OpenFlow module for data flow creation, implementing the intrusion rules construction, presenting the attack features by undirected graph method, realizing the incremental representation of new attacks, reducing the computational complexity of undirected graph construction, and using the k nearest neighbor algorithm to classify the malicious attack traffic features of undirected graphs. In the future, this work will be extended to consider other types of SDNs for attacks, and the embedded development of OpenFlow applications for predictive models is also the evolutionary research direction.

Acknowledgement

Zhejiang Natural Science Foundation (LY18A010018), Zhejiang Provincial Department of education research project (Y201718244).

References

[1] Catalyurek U, Aykanat C: (1999) *Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication*[J]. Parallel & Distributed Systems IEEE Transactions on, 10(7):673-693.

[2] Akbudak K, Kayaaslan E, Aykanat C: (2012) *Analyzing and enhancing OSKI*

for sparse matrix-vector multiplication[J]. Computer Science.

[3] Çatalyürek, Ümit V, Aykanat C: (1996) *Decomposing Irregularly Sparse Matrices for Parallel Matrix-Vector Multiplication*[C]// International Workshop on Parallel Algorithms for Irregularly Structured Problems. Springer-Verlag, 1996:75-86.

[4] Mansour A, Götze J: (2012) *Sparse Matrix-Vector Multiplication Based on Network-on-Chip: On Data Mapping*[C]// Fifth International Symposium on Parallel Architectures, Algorithms and Programming. IEEE Computer Society, 2012:41-44.

[5] Črná, Dana, Fině V: (2011) *Adaptive wavelet methods - Matrix-vector multiplication*[C]// American Institute of Physics, 2011:147-151.

[6] Črná D, FiněV: (2012) *Adaptive wavelet methods - Matrix-vector multiplication*[J]. Applications of Mathematics in Engineering & Economics, 1410(1):147-151.

[7] Yan X, Wang N, Yin Y, et al.: (1997) *A negabinary encoded matrix - vector multiplication for wavelet transformsMultiplication matrice - vecteur par codage négabinaire pour la transformée en ondelettes*[J]. Journal of Optics, 28(28):66-69.

[8] Yan X, Wang N, Yin Y, et al.: (1997) *A negabinary encoded matrix - vector multiplication for wavelet transformsMultiplication matrice - vecteur par codage négabinaire pour la transformée en ondelettes*[J]. Journal of Optics, 28(28):66-69.

[9] Song W, Deng Z, Wang L, et al.: (2016) *G-IK-SVD: parallel IK-SVD on GPUs for sparse representation of spatial big data*[J]. Journal of Supercomputing, 2016:1-18.

[10] Hackbusch W, Khoromskij B N: (1999) *A sparse $scr H$-matrix arithmetic. II. Application to multi-dimensional problems.*[J]. 64(1):21-47.

[11] Griebel M, Knapek S: (2009) *Optimized general sparse grid approximation spaces for operator equations*[J]. Mathematics of Computation, 78(268):2223-2257.

[12] Harten A, Yad-Shalom I: (1994) *Fast Multiresolution Algorithms for Matrix-Vector Multiplication*[J]. Siam Journal on Numerical Analysis, 31(4):1191-1218.

[13] He Z, Ogawa T, Haseyama M: (2011) *Cross Low-Dimension Pursuit for Sparse Signal Recovery from Incomplete Measurements Based on Permuted Block Diagonal Matrix*[J]. Ieice Transactions on Fundamentals of Electronics Communications & Computer Sciences, 94-A(9):1793-1803.

[14] Chegini N, Stevenson R: (2012) *The adaptive tensor product wavelet scheme: sparse matrices and the application to singularly perturbed problems*[J]. Ima Journal of Numerical Analysis, 32(1):75-104.

[15] Atkinson K, Chien D D: (2000) *A fast matrix–vector multiplication method for solving the radiosity equation*[J]. Advances in Computational Mathematics, 12(2):151-174.

[16] Zheng C Y, Li H: (2013) *Small infrared target detection based on harmonic and sparse matrix decomposition*[J]. Optical Engineering, 52(6):066401.

[17] Knibbe H, Mulder W A, Oosterlee C W, et al.: (2014) *Closing the performance gap between an iterative frequency-domain solver and an explicit time-domain scheme for 3D migration on parallel architectures*[J]. Geophysics, 79(2):S47–S61.

[18] Knibbe H, Mulder W A, Oosterlee C W, et al.: (2014) *Closing the performance gap for 3-D migration with an iterative frequency-domain solver and an explicit time-domain scheme on parallel architectures*[J]. Geophysics, 79:547-561.

[19] Francomano E, Tortorici A, Toscano E, et al.: (2005) *Wavelet-like bases for thin-wire integral equations in electromagnetics*[J]. Journal of Computational & Applied Mathematics, 175(1):77-86.

[20] Ravnik J, Škerget L, ŽuniěZ: (2009) *Comparison between wavelet and fast multipole data sparse approximations for Poisson and kinematics boundary – domain integral equations*[J]. Computer Methods in Applied Mechanics & Engineering, 198(17–20):1473-1485.

[21] Harmany Z T, Marcia R F, Willett R M: (2012) *This is SPIRAL-TAP: Sparse Poisson Intensity Reconstruction ALgorithms–theory and practice*[J]. IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society, 21(3):1084-96.

Received May 7, 2017