Clustering Analysis for Malicious Network Traffic

Jie Wang*, Lili Yang*, Jie Wu† and Jemal H. Abawajy‡
*School of Information Science and Engineering, Central South University, Changsha, China 410083
Email: jwang, liliyang@csu.edu.cn
†Department of Computer and Information Sciences, College of Science and Technology, Temple University, Philadelphia, USA
Email: jiewu@temple.edu
‡School of Information Technology, Deakin University, Burwood 3125, Australia
Email: jemal@deakin.edu.au

Abstract—**With the volume and variety of network attacks increasing, efficient approaches to detect and stop network attacks before they damage the system or steal data is paramount to users and network administrators. Although many different detection mechanisms have been proposed, exiting detection methods generally tend to successfully detect attacks only after the attacks have finished and caused damage to the system. As recent attacks employ polymorphism technology and complicated attack techniques, it has become even more difficult for these approaches to detect attacks in a timely manner. In this paper, we propose an efficient network attack detection algorithm called seed expanding (SE) that detects attacks before they damage the system. SE employs the Two-Seed-Expanding network traffic clustering scheme, which clusters attack traffic into different attack phases. First we pre-process the networks traffic, including constructing the network flow, changing continuous-valued attributes into nominal attributes by adopting the discretization method, and further turning into binary features. Then based on these features, SE computes a weight for each flow and iteratively selects seeds to expand until all flows are divided into clusters. To investigate the effectiveness of the proposed approach, we undertook extensive experimental analyses. The results of the experiment show that the pre-procession greatly improves clustering performance, and the Two-Seed-Expanding Algorithm is better than K-Means and other kinds of Seed-Expanding in attack-flow clustering. These cluster results can be further used in attack detection.**

Index Terms—**malicious network traffic, attack detection, attack phase, network flow clustering.**

I. Introduction

Most malicious network traffic is generated by different types of network attacks. Attackers deploy malware to compromise a computer system by exploiting security vulnerabilities. In recent years, there has been a significant research focus on detecting network anomaly traffic and identifying attack types [1], [2]. However, with the development of advanced attack methodologies, attack scenarios have become more and more complicated. There are many features of an attack, such as imperceptibility, polymorphism and so on, which makes network attacks very difficult to detect. Nonetheless, detecting attacks earlier is very important in preventing the attacks from damaging the system.

In order to detect attacks earlier, it is important to identify what phase the attack is in as early as possible. To detect the attack phase, the first mission is to use some methods to distinguish different attack phases. Clustering analysis is a good technology for classifying network traffic into different attack phases, although it cannot mark the current attack phase. Results from clustering can further be used to analyze the features of the attack phase.

Traditional clustering algorithm schemes include hierarchical clustering, centroid-based clustering, distribution-based clustering, density-based clustering, and so on [3]. Hierarchical clustering decomposes a given dataset until some conditions are satisfied. Centroid-based clustering first selects a representative object for each cluster, and the rest of the objects are allocated to the nearest cluster. The typical centroid-based clustering algorithm is K-Means. Distribution-based clustering utilizes the distribution information of a dataset to find the cluster. Density-based clustering finds the high density areas that are separated by low density areas. However, because we cannot control the similarity level of data points in the clustering process, these cluster algorithms are not appropriate for detecting malicious network flows. In other words, the network administrator cannot control the similarity level in the process of the clustering attack phase according to network policy. For instance, the networks of banks and governments acquire a higher level of security, and personal networks can allow more traffics inpour with a lower security level.

Motivated by this, we design an efficient clustering algorithm which divides one attack into multiple phases, where every attack phase is considered as one type of attack behavior, including discovering a vulnerable host, getting root access privileges or gaining access to a vulnerable host, and gathering sensitive information from the Internet. Specially, although one attack may have one or multiple steps, we consider it as a multi-step attack with default steps. The clustering results will be used to further detect malicious network traffic. The main contributions of this paper are summarized as follows.

1) We propose an unsupervised malicious network traffic clustering scheme which uses statistical flow-level features to cluster malicious network traffic into different attack phases.
2) We present a clustering algorithm SE which adopts two seed expanding methods to improve clustering performance, and we can control the different security levels in the clustering process.

3) We further expand SE to other kinds of Seed Expanding, and discuss how many chosen seeds are appropriate in Seed Expanding.

The rest of the paper is organized as follows. Section 2 presents an unsupervised malicious network traffic clustering scheme. Section 3 presents a seed-expanding clustering algorithm. The experimental results are shown in section 4. Finally, section 5 concludes the paper.

II. Related Work

Known research about malicious network traffic is concentrated in malware detection, including signature-based and anomaly-based detection methods [4].

Signature-based detection matches network traffic against a set of predefined signatures to determine whether they are malicious. Recently, most signatures are generated from packet payloads. Because of the emergence of metamorphic and polymorphic technologies, it is different to employ these signatures to detect malicious network traffic. Moreover, supervised classification is a related approach to signature-based detection. Some instances based on signatures are used to build a classification model. Therefore, supervised classification approaches show the same limitations as signature-based detection.

For example, Stevanovic et al. [5] used supervised machine learning to detect a flow-based botnet. They made a conclusion that the C4.5 Decision Tree, Random Tree and Random Forest were the most efficient in detecting a botnet. Nogueira et al. [6] also designed a flow based system in which a Neural Network was employed to detect a botnet. Similarly, Hsiao et al. [7] used a flow-based method to detect malicious behavior. They created four sets of attributes: NetFlow variables, Tmeporal Variables, Spatial Variables and a combination of Temporal and Spatial variables. They applied Naive Bayes, Decision Tree and SVM algorithms in the experiments. The results showed that a combination of temporal and spatial attributes worked well. Moreover, Saad et al. [8] implemented an approach based on some classification algorithms to detect malicious botnet behavior by using flow based attributes and host-based attributes. Haddadi et al. [1] proposed a framework on flow-based network traffic to detect a botnet, which employed machine learning algorithms C4.5 and Native Bayes. Tellenbach [9] et al. proposed a comprehensive anomaly detection and classification system based on traffic feature distribution. Coluccia [2] proposed to apply an empirical and estimated feature probability distribution for anomaly detection. Gamer T. [10] introduces a newly developed collaborative attack detection that facilitates collaboration beyond domain boundaries without requiring close trust relationships.

On the other hand, traditional anomaly-based detection profiles the statistical features of normal traffic. Any deviation from the profile will be considered as malicious. However, it faces some risk of generating a high false positive rate, which means a great number of normal network traffic will be wrongly classified as malicious.

In recent years, some unsupervised learning techniques have been designed for malicious network traffic detection and analysis [11], which can be used to identify unknown malware [12]. However, with the development of attack technology, malicious network traffic generated by malware gradually has similar traffic statistical features to that of normal traffic. Therefore, when this kind of features are used by some detection approaches to identify malicious, which will suffer.

III. Malicious Network Traffic Clustering Scheme

This section first introduces the basic concepts in malicious network traffic clustering and the flow-level features we adopt. Then, we describe the feature pre-processing. Finally, we present the unsupervised learning scheme.

A. Basic Concepts and Feature

IP flow is the basic unit in malicious network traffic clustering. Wang et. al. [13] define IP flow as a sequence of IP packets exchanged between a pair of endpoints for the purpose of inter-process communication across the Internet. We also define an IP flow by a 5-tuple: the srcIp, srcPort, dstIp, dstPort, and Protocol.

An IP flow is described by an n-dimension feature vector $X = \{x_1, x_2, \ldots, x_n\}$, where n is the number of features. x_i is a statistical feature extracted from a single IP flow and we do not consider the direction of the IP flows. Andrew M. Moore et. al. [14] proposed 248 sophisticated features in their study, which can be used in flow-based classification and clustering. The study is very useful for researching flow-based classification. However, if a great number of features are used in flow-based classification, it will generate a high computational overhead. Previous studies have shown that simple features have a sufficient discriminating power [12], [13], [15]. Similarly, we also select simple features.

Feature selection and transformation play an important role in machine learning. In this paper, our goal is to cluster malicious flows into different phases, which will be beneficial for further detecting and analyzing malwares. Towards this end, we use the layer-3 and layer-4 network traffic features such as the sizes of the packets (see Table1). This is motivated by the previous works [13] and [4]. In [13], it was suggested that ports and sizes of TCP and UDP can be applied to classify online traffic. In [4], it was observed that threats often exhibit specific behavior in terms of their layer-3/layer-4 statistical flow-level features and these features retain their properties even when the payload is encrypted. The flow statistic features we used are illustrated in Table I.

B. Feature Preprocessing

We first preprocess our dataset. A continuous-valued attribute is discretized by partitioning its range into multiple intervals. A threshold value T is a continuous-value attribute. A is a cut point if $A \leq T$ is assigned to the left branch while $A > T$ is assigned to the right branch. As our approach is

TABLE I: Flow-level features

Name	Feature Discription
pkts	total number of packets
pkt-noPayload	total number of packets without payload
bytes	total number of bytes transferred
pay-bytes	total number bytes from all payloads
duration	flow duration
maxsz	maximum packet size
minsz	minimum packet size
avgsz	average packet size
stdsz	standard deviation of packet size
maxpy	maximum payload size
minpy	minimum payload size
avgpy	average payload size
stdpy	standard deviation of payload size
Flag	flags (acks, fins, resets, pushs, etc)

based on the Entropy-based discretization method [17], the following definitions are derived from there as well.

Definition 1: Let T partition the set *S* of examples into the subsets S_1 and S_2. Let there be *k* classes $C_1, \ldots, C_k$ and let $P(C_i, S)$ be the proportion of examples in *S* that have class C_i. The class entropy of a subset *S* is defined as:

$$Ent(S) = -\sum_{i=1}^{k} P(C_i, S) log(P(C_i, S))$$

Definition 2: For an example set *(S)*, an attribute *(A)*, and a cut value *(T)*: Let $S_1 \subset S$ be the subset of examples in *S* with $A - values \leq T$ and $S_2 = S - S_1$. The class information entropy of the partition induced by *T*, $E(A, T; S)$ is defined as

$$E(A, T; S) = \frac{S_1}{S} Ent(S_1) + \frac{S_2}{S} Ent(S_2) \tag{1}$$

Definition 3: MDLPC Criterion: The partition induced by a cut point *(T)* for a set *(S)* of *N* examples is accepted if

$$Gain(A, T; S) > \frac{log_2(N-1)}{N} + \frac{\Delta(A, T; S)}{N}$$

where

$$\Delta(A, T; S) = log_2(3^k - 2) - [kEnt(S) - k_1 Ent(S_1) - k_2 End(S_2)] \tag{2}$$

$Gain(A, T; S)$ is the information gain of a cut point, and is defined as:

$$\begin{aligned} Gain(A, T; S) &= Ent(S) - E(A, T; S) \\ &= Ent(S) - \frac{S_1}{N} Ent(S_1) + \frac{S_2}{N} Ent(S_2) \end{aligned} \tag{3}$$

The number $N = |S|$ in Eq. (3), and the parameters *k*, k_1, and k_2 are constant as discussed in [17]. Specifically, for each feature we divide the range of observed values into a series of intervals based on the accepted cut point, and then, map each interval to a discrete nominal symbol.

A nominal attribute can take on two or more states. For example, pkts in Table I becomes a nominal attribute by discretizing, including five states: $(1.5-2.5]$,$(2.5-3.5]$,$(3.5-25.5]$,$(25.5-28]$,$(28-inf)$. Nominal attributes can be encoded using asymmetric binary attributes by creating a new binary attribute for each of the *M* states. For an object with a given state value, the binary attribute representing that state is set to 1, while the remaining binary attributes are set to 0.

C. Unsupervised Malicious Network Traffic Clustering

The process of malicious network traffic clustering from identifying flows includes flow identification, feature extraction, feature preprocessing, unsupervised learning and clustering. The traffic is in the form of IP packets. The flows are first constructed by aggregating traffic based on the 5-tuple flow identifiers. Then, features can be extracted from the flows. We preprocess the features by using the discretization method, and the features are transformed into nominal attributes. Then these nominal attributes are encoded using asymmetric binary attributes. Finally, these binary attributes can be handled by clustering algorithms, which generate different partitions on the input data. The result we need is to construct pure clusters where the input may include a little noise. That is to say, our goal is for the flows in each cluster to be from the same attack phase.

In previous studies [12], [16], the core algorithm was designed to classify malicious traffic from normal traffic, however, some malicious traffic is the same as normal traffic when they are considered individually. Especially, like DDOS, it has different attack phases in the entire attack process: scanning, installing Trojan horse malware, launching DDoS, and so on. Therefore, it is necessary to cluster malicious network traffic into different phases. Based on this, it will be easier to classify malicious traffic.

IV. Clustering Algorithm Based on Seeding-Expanding

In this section, we describe the process of similarity computing between different flows, and present a novel clustering algorithm based on seed-expanding.

A. Similarity Computing

For asymmetric binary attributes, the two states are not equally important. Given two asymmetric binary attributes, the agreement of two 1s (a positive match) is then considered more significant than that of two 0s (a negative match). Therefore, such binary attributes are often considered "monary" (having one state). The number of negative matches is considered unimportant and is thus ignored. The asymmetric binary similarity between the objects *i* and *j* can be computed as

$$sim(i, j) = \frac{q}{q + r + s}. \tag{4}$$

In Eq. (4), *q* is the number of positive matches that equal 1 for both objects *i* and *j*, *r* is the number of attributes that equal 1 for object *i* but equal 0 for object *j*, and *s* is the number of attributes that equal 0 for object *i* but equal 1 for object *j*. The coefficient $sim(i, j)$ of Eq. (4) is called the Jaccard coefficient [3] .

B. The Process of Clustering Algorithm

Given a set D, $D = \{d_1, d_2, .., d_n\}$, we propose that the SE algorithm divide set D into some clusters. The SE algorithm contains the following steps.

(1) Weight Computing. The first step is to compute the weight for each data point where a data point means a flow. Firstly, for each data point, the weight is computed, where the weight of one data point is the sum of the similarity between this data point and all other data points. It is denoted as:

$$Weight_{d_i} = \sum_{j=1}^{n} sim(d_i, d_j). \tag{5}$$

(2) Seed Selection. Firstly, all data points are sorted based on their *Weights* by decreasing and constructing a candidate queue $Q = \{q_1, q_2, \ldots, q_n\}$. Next, we discuss how to choose two seeds. When selecting seeds, we must make sure the two seeds belong to different clusters with high probabilities. Based on Eq. (5), the *Weights* of the data points in the same cluster are similar. Therefore, we compute the different value of the weights between adjacent data points of the candidate queue, then choose the adjacent two data points with highest difference value as seeds. The first seed, s_1, is selected as follows:

$$s_1 = \arg\max_{q_i} |Weight_{q_i} - Weight_{q_{i+1}}|$$

The second seed, s_2, is next to s_1 in the queue Q.

(3) Seed Expanding. We expand one cluster based on every seed, individually. One data point q in the candidate queue is added to a cluster only if it has the maximal similarity value to seed s. The data point q is as follows:

$$q = \arg\max_{q_j}(sim(q_j, s))$$

Then data point *d* is deleted from *Q*. When similarity values between seed point *s* and the rest of the data points in the candidate queue are less than threshold *r*, which is denoted as:

$$\max_{q \in Q}(sim(q, s)) < r$$

the expanding process is stopped. The selection of two new seeds and the expanding cluster are run iteratively until the queue becomes null.

(4) Noise removal. As a dataset may include a noise flow that does not belong to the attack flow, clusters with sizes smaller than 3 are considered noise data. The algorithm is illustrated in Figure 1.

In Figure 1, the input D is the set of data points $\{d_1, d_2, ...d_n\}$. Threshold *r* is a pre-value. We can control the similarity level in the clustering process by adjusting the number of thresholds. When the similarity between the seed and the data points of a candidate set is less than the threshold, the process of expanding the current cluster will be stopped.

```
Algorithm Two-Seed-Expanding
Input: $D = \{d_1, d_2, \ldots, d_n\}$, $r$;
Output: $CLUSTER = \{cluster_1, cluster_2, ...\}$;
Begin
    Weight Computing: Calculating weight for each data point in D, where
        $Weight_i = \sum_{j=1}^{n} sim(d_i, d_j)$;
    Sorting all data points and constructing candidate queue $Q = \{q_1, q_2, \ldots, q_n\}$ by decreasing;
    do
        Seed selection: Selecting two data points $s_1, s_2$ as seeds from $Q$, where
            $(s_1, s_2) = \arg\max_{(q_i, q_{i+1})} |Weight_{q_i} - Weight_{q_{i+1}}|$;
        Seed expanding: Generating a new cluster, $cluster_k$ based on seed $s_1$ by adding new data point $q$ from $Q$, where
                $q = \arg\max_{q_j} sim(q_j, s_1)$,
                $cluster_k \leftarrow q$;
                Deleting $q$ from queue $Q$; until $max(sim(q_j, s_1)) < r$;
        Generating a new cluster $cluster_{k+1}$ based on seed $s_2$ by adding new data point $q$ from $Q$, where
                $q = \arg\max_{q_j} sim(q_j, s_2)$,
                $cluster_{k+1} \leftarrow q$;
                Deleting $q$ from queue $Q$; until $max(sim(q_j, s_2)) < r$;
    Until $Q = \Phi$;
    Noise removal;
    return CLUSTER;
End
```

Fig. 1: The Two-Seed Expanding Algorithm

V. Evaluation

A. Data

SE is evaluated by adopting a DDoS sample that comes from the classical intrusion detection dataset Darpa2000 scenario 1 [18]. This attack scenario includes multiple networks and audit sessions. Specially, the process of attack in the Darpa2000 dataset is divided into five stages: 1) IP-sweep of the AFB from a remote site; 2) Probe of the live IPs to look for the sadmind daemon running on Solaris hosts; 3) Breakings via the sadmind vulnerability, both successful and unsuccessful on those hosts; 4) Installation of the trojan mstream DDoS software on three hosts at the AFB; 5) Launching the DDoS.

B. Result and Discussion

There are three common indexes to evaluate the clustering results (purity, RI, F-Measure) [19]. We run a series of clustering experiments on test datasets using a range of thresholds, *r*, from 0 to 1. The results are illustrated in Figure 2, Figure 3 and Figure 4. In these figures, the horizontal axis stands for a threshold, *r*, adopted by the SE algorithm. With the increase of *r*, cluster purity, the Rand Index and F-measures will increase. Especially, when *r* is larger than 0.5, they will obtain better results. In addition, in Fig. 2, there are three curves, which denote the clustering results obtained by using the discretization method and asymmetric binary attributes, only using the discretization method and using directly continuous-valued attributes. From these figures, it can be seen that feature discretization and the use of asymmetric binary attributes both improve flow clustering performance.

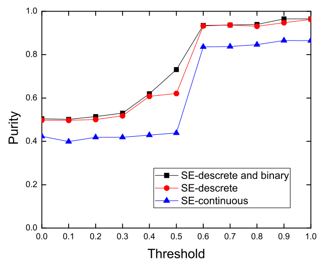

Fig. 2: Purity (SE)

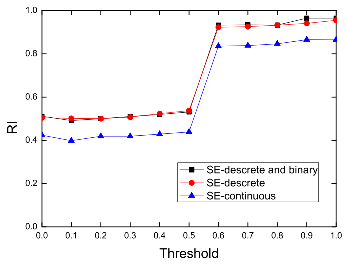

Fig. 3: Rand Index (SE)

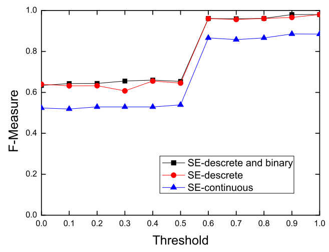

Fig. 4: F-measure (SE)

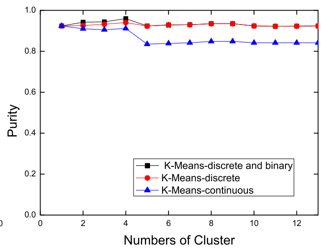

Fig. 5: Purity (K-Means)

Fig. 5 illustrates the cluster purity of K-Means. As K-Means needs to set the number of clusters, we give purity values with different clustering numbers. From Fig. 5, it can be shown that the clustering performance has obviously improved because of the feature discretization and asymmetric binary attributes. Fig. 6 and Fig. 7 separately show the Rand Index and F-measures of K-Means clustering, which have similar results with Fig. 5.

Moreover, we compare the SE algorithm with K-Means, where we choose threshold $r = 0.9$. The results are shown in Table II. According to Table II, the SE algorithm can achieve a higher clustering performance than that of K-Means with any input cluster number. In fact, it is hard for K-Means to choose the most suitable clustering number.

C. Algorithm Analysis

In the section, we discuss some questions about the algorithm and experiments. Firstly, we present the clustering process without using discretion or asymmetric binary attributes. Next, we give the reason why we choose two seeds for each iteration rather than one seed, three seeds, and so on. Furthermore, we compare the clustering results with different seed numbers in the process of iteration.

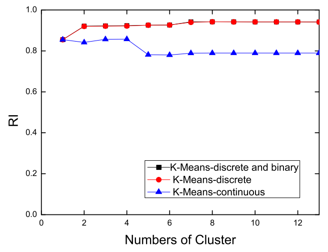

Fig. 6: Rand (K-Means)

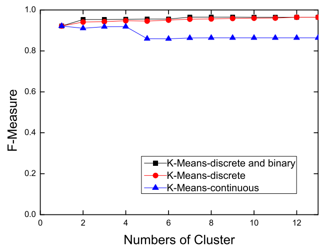

Fig. 7: F-measure (K-Means)

TABLE II: The result comparing between SE and K-Means

	input cluster number	Purity	Rand Index	F-measure
SE-discrete		0.9552	0.9651	0.980306
K-Means-discrete	1	0.923751	0.854832	0.921735
K-Means-discrete	2	0.925504	0.920917	0.953059
K-Means-discrete	3	0.903593	0.92204	0.953408
K-Means-discrete	4	0.909728	0.923537	0.954257
K-Means-discrete	5	0.923751	0.925695	0.955486
K-Means-discrete	6	0.928133	0.926251	0.955806
K-Means-discrete	7	0.928133	0.94195	0.964883
K-Means-discrete	8	0.935145	0.942144	0.964992
K-Means-discrete	9	0.935145	0.941913	0.964848
K-Means-discrete	10	0.923751	0.94149	0.964584
K-Means-discrete	11	0.921998	0.941469	0.96457
K-Means-discrete	12	0.921998	0.941404	0.96453
K-Means-discrete	13	0.923751	0.94145	0.964556

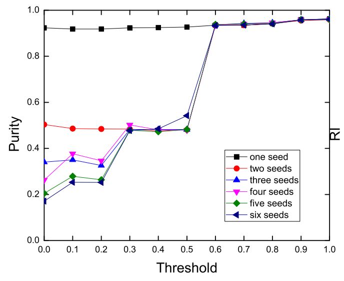

Fig. 8: Purity (SE) with different seed numbers

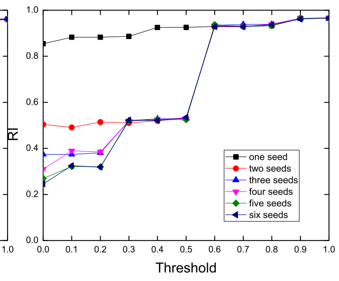

Fig. 9: Rand (SE) with different seed numbers

When the process of clustering does not use discretion, we adopt the RBF(Radial Basis Function) to calculate similarity. The similarity between objects i and j can be computed as

$$sim(i,j) = exp(-\frac{\|i-j\|^2}{2\sigma^2}) \tag{6}$$

In Eq. (6), $\|i-j\|^2$ is the Euclidean distance between objects i and j, and $\sigma^2 = \frac{1}{2}$. When the two objects are very similar, the value of $sim(i,j)$ is close to 1. On the contrary, when the distance between i and j is greater, the value of $sim(i,j)$ is close to 0.

If the process of clustering does not use asymmetric binary attributes, the similarity between objects i and j can be computed based on the ratio of matches:

$$sim(i,j) = \frac{m}{p} \tag{7}$$

where m is the number of matches (i.e., the number of attributes for which i and j are in the same state), and p is the total number of attributes describing the objects.

Moreover, in the SE algorithm, we choose two seeds for each iteration. In the next experiments, we choose one seed, two seeds, three seeds, four seeds, five seeds and six seeds in the process of seed selecting, and the clustering results are illustrated in Fig.8, Fig.9, and Fig.10. Specifically, we choose a data point that has a max weight of as one seed. While choosing three seeds, first, we get four data points from the first largest difference weight and the second largest difference

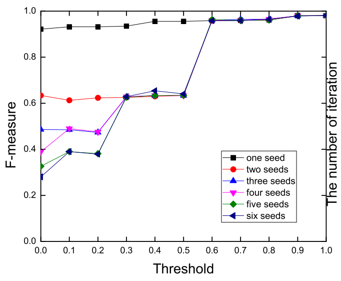

Fig. 10: F-measure (SE) with different seed numbers

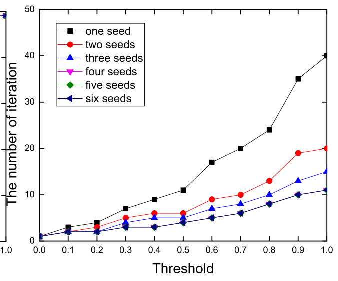

Fig. 11: The number of iterations required with different seed numbers

weight, then we select three data points as seeds with the top three maximum weights. In a similar fashion, we get four seeds, five seeds, and six seeds.

Figure 8 illustrates the purity value with a different number of clustering seeds. It is easy to find that clustering can obtain good results no matter how many seeds are selected for each iteration when threshold r is larger than 0.5. From Figure 9 and Figure 10, a similar conclusion can be drawn.

We compare the number of iterations required with different seed numbers. The results of the comparison are illustrated in Figure 11.

According to Figure 11, when the number of seeds increases, the number of iterations decreases gradually. Specifically, when two seeds are selected, the number of iterations reduces significantly. However, when the number of seeds further increases, the iteration number remains the same. That is to say, the number of iterations does not always decrease with the increase of the number of seeds selected. In addition, when datasets include only two or three kinds of objects, selecting more seeds will cause some small clusters. To the extreme extent, these small clusters may be considered as noise. Therefore, we choose two seeds for each iteration.

At the same time, we compare the running time. The running time of one seed is about 40s when clustering the datasets, while the running time of two seeds is about 30s. Furthermore, the running time of three seeds, four seeds, five seeds and six seeds is around 28s. This is because the seed selection process will take some time. Therefore, it is reasonable to select two seeds for each iteration.

VI. Conclusion

In this paper, we propose a clustering algorithm and a clustering scheme based on Two-Seed-Expanding, which clusters attack flows into different phases. We also discuss the clustering's effectiveness when different seed numbers are used in the Seed-Expanding algorithm. Experimental results show that SE adopts the discretization method and asymmetric binary attributes to process attacks flows, which greatly improves clustering performance. Moreover, Two-Seed-Expanding is better than K-Means and other kinds of Seed-Expanding that use different seed numbers in attack flow clustering. Next, we will further research how to identify attack flows from normal flows based on the clustering results.

Acknowledgment

This work was supported by NSF of China under Grant No. 61573379, No. 61202495, and in part by NSF grants CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, and ECCS 1231461.

References

[1] F. Haddadi, J. Morgan, E.G. Filho, A.N. Zincir-Heywood. Botnet behaviour analysis using IP flows: with HTTP filters using classifiers. 2014 28th International Conference on Advanced Information Networking and Applications Workshops, pp. 7-12.
[2] A. Coluccia, A. Dalconzo, F. Ricciato. Distribution-based anomaly detection via generalized likelihood ratio test: A general maximum entropy approach, Computer Networks 2013; 57(17): 3446C3462.
[3] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Tchiques, Third Edition, 2011.
[4] P. M. Comar, L. Liu and S. Saha et.al., Combining Supervised and Unsupervised Learning for Zero-Day Malware Detection, IEEE INFOCOM 2013, pp.2022-2030.
[5] M. Stevanovic, J.M. Pedersen. An efficient flow-based botnet detection using supervised machine learning. In: 2014 International Conference on Computing, Networking and Communications (ICNC), pp. 797-801.
[6] A. Nogueira, P. Salvador, F. Blessa. A botnet detection system based on neural networks. In: 2010 Fifth International Conference on Digital Telecommunications, pp. 57-62.
[7] H. Hsiao, D. Chen, T. Ju Wu. Detecting hiding malicious website network traffic mining approach. 2010 2nd International Conference on Education Technology Computer (ICETC), vol. 5, pp. 276-280.
[8] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, P. Hakimian. Detecting P2P botnets through network behavior analysis and machine learning. Proceedings of 9th Annual Conference on Privacy, Security and Trust.
[9] B. Tellenbach, M. Burkhart, D. Schatzmann, D. Gugelmann, D. Sornette. Accurate network anomaly classification with generalized entropy metrics, Computer Networks 2011; 55(15): 3485-3502.
[10] T. Gamer. Collaborative anomaly-based detection of large-scale internet attacks, Computer Networks 2012, 56(1): 169?85.
[11] J. Mazel, P. Casas, Y. Labit and P. Owezarski. Sub-space clustering, inter-clustering results association and anomaly correlation for unsupervised network anomaly detection. In Proceedings of the 7th International Conference on Network and Service Management, Paris, France, 2011: 1-8.
[12] P. Casas, J. Mazel, and P. Owezarski. Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. Computer Communications, April 2012, 35(7): 772-783.
[13] Y. Wang, Y. Xiang and Jun Zhang et.al., Internet Traffic Classification Using Constrained Clustering, IEEE transactions on parallel and distributed systems, 25(11):2932-2943, 2014.
[14] A.W. Moore, D. Zuev, M.L. Crogan. Discriminators for use in flow-based classification. Technical Report, RR-05-13 [R]. UK: Queen Mary University of London, 2005.
[15] Y. Wang, Y. Xiang, J. Zhang, W.L. Zhou, B.L. Xie. Internet Traffic clustering with side information. Journal of Computer and System Science, 2014, 80: 1021-1036.
[16] L.L Yang, J. Wang, P. Zhong. Combining Supervised and Unsu-pervised Learning for automatic attack signature generation system. Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing. Dalian: Springer Verlag, 2014: 607-618.
[17] U.M. Fayyad and K. B. Irani, Multi-interval discretization of continuous-valued attributes for clasification learning, In Proceedings of the Internationaloint Conference on Uncertainty in Articial Intelligence, 1993.
[18] Lincoln Laboratory. 2000 DARPA Intrusion Detection Scenario Specific Data Sets.www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/2000data.html, 2000.
[19] C.D. Manning, P. Raghavan and H. Schutze, Introduction to Information Retrieval. Cambridge University Press. 2008.