PROCEEDINGS OF SCIENCE

# Research on Intrusion Detection based on Improved AntMiner Algorithm

**Yanping Shen**[1]
*School of CyberSpace Security, Beijing University of Posts and Telecommunications*
*Beijing, 100876, China*
*Department of Disaster Information Engineering, Institute of Disaster Prevention*
*Beijing, 101601, China*
*E-mail:* shenyanping@cidp.edu.cn

**Kangfeng Zheng**
*School of CyberSpace Security, Beijing University of Posts and Telecommunications*
*Beijing, 100876, China*
*E-mail:* kfzheng@bupt.edu.cn

**Chunhua Wu**
*School of CyberSpace Security, Beijing University of Posts and Telecommunications*
*Beijing, 100876, China*
*E-mail:* wuchunhua@bupt.edu.cn

AntMiner is the first swarm intelligence algorithm based on the ant colony optimization algorithm in data mining area. Since AntMiner is easy to fall into the local optimal solution and has long convergence time, we introduce ACSMiner algorithm in this paper. ACSMiner is based on the ant colony system which constructs rules by using the state transition rules, not only updating the pheromone of each ant locally, but also updating the global optimal solution. The KDD99 dataset was used to carry out the experiment. Simulation results demonstrated that compared with the original algorithm, the variant works well in improving the accuracy and reducing false positive rate as well as shortening the training time. It shows that the method can be effectively used in the field of intrusion detection.

*ISCC 2017*
*16-17, December, 2017*
*Guangzhou, China*

[1]This research was supported by the National Key R&D Program of China (2017YFB0802803), the National Natural Science Foundation of China (61602052), the Science and Technology Research and Development Project of Langfang (2017011027) and the Fundamental Research Funds for the Central Universities (2017011027).





PoS(ISCC 2017)003

## 1.Introduction

With the rapid development of network technology, the network security has received much concern. Traditional security technologies, such as firewall, access control, encryption technology, fail to fully protect the network and system security [1]. In such case, Intrusion Detection System (IDS), an irreplaceable part of protecting the system security, has been the research focus of scholars at home and abroad. But there are still some problems such as low accuracy and high false alarm rate to be solved.

There are many methods used in the intrusion detection which can be summarized as follows: artificial neural network, fuzzy system, evolutionary computation, artificial immune system and swarm intelligence [1]. The swarm intelligence method employed in the field of intrusion detection is a new research direction, which can be divided into the following aspects: (1) apply in distributed intrusion detection field [2]; (2) optimize the algorithm framework, including the parameter optimization and feature selection [3-5]; (3) extract the rules, extract classification rules based on the idea of data mining [6-11]. Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) are the most popular swarm intelligence algorithms. ACO has good effect on solving discrete optimization problems, especially on the classification and clustering problems in data mining. PSO has an edge in solving nonlinear combinatorial optimization problems by stimulating the activities of simple natural ecosystems, such as the activities of birds or fish. The improved AntMiner algorithm, which achieves the intrusion detection by extracting rules, was proposed [7-11] . This paper mainly studies the optimization problem of the rule extraction based on the ant colony algorithm. To distinguish different categories of rules, the training samples were separated by category before the extraction. Thus, different kinds of extraction rules can be necessary. Meanwhile, the arccosine function was adopted as a new heuristic function [7]; In the design of the objective function, the dual factors were considered, including classification accuracy and the coverage of the rules [8] [10] [11]; the node state transition rule was introduced into the algorithm, and the quality of rules was used in pheromone updation [9].

Most of the above works only considered local factors, but failed to improve the algorithm framework globally. The pheromone plays a very important role in the process of rules construction. If the pheromone values of different paths vary greatly, the ant will be only moving along paths of high pheromone. That means the ability of the ant for further searching better solution will be weakened, which will cause them to fall into the local optimal solution easily; if the difference of pheromone of different paths is small, it is difficult for the ant to find the optimal solution in a very short period of time, resulting in long convergence time. In view of the above background, this paper proposes a new algorithm, called ACSMiner, which takes into account the pheromone update mechanism and applies it in the field of intrusion detection.

## 2.AntMiner

AntMiner algorithm achieves the classification by extracting rules [12]. The rules are as follows: If $<term_1$ and $term_2$ and…$term_n>$ Then $<class>$. $term_i$ represents the value of an attribute, where , such as: *protocol = tcp*, where *class* denotes the label.

### 2.1 Rules construction

Whether a node is added to the list of rules is determined by the probability equation (1) [12], and the node $term_{ij}$ represents the $j_{th}$ value of the $i_{th}$ attribute. Where $P_{ij}$ denotes selected

probability of $term_{ij}$; represents the heuristic factor of $term_{ij}$, represents the pheromone of $term_{ij}$ after $t_{th}$ iteration; $a$ is the number of attributes; when attribute $A_i$ is not selected =1, or =0; is the number of all values of $A_i$.

$$P_{ij} = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{i=1}^{a} x_i \cdot \sum_{j=1}^{b_i} (\tau_{ij}(t) \cdot \eta_{ij})}$$

(2.1)

### 2.2 Heuristic Function

The Shannon entropy is used to measure the quality of the node in AntMiner. The greater the entropy, the distribution of samples is more uniform and the smaller the probability that the ant will add this node to the path, as shown in equation (2) [12]. $H$ represents the entropy; $W$ represents the labels; $k$ represents the number of categories; represents the priori probability of when the label is .

$$H(W|A_i=V_{ij}) = -\sum_{w=1}^{k} P(w|A_i=V_{ij}) \cdot \log_2 P(w|A_i=V_{ij})$$

(2.2)

The entropy can be directly employed in equation (1) after normalization, as , so the heuristic function is defined based on the information theory [12]

$$\eta_{ij} = \frac{\log_2 k - H(W|A_i=V_{ij})}{\sum_{i=1}^{a} x_i \cdot \sum_{j=1}^{b_i} (\log_2 k - H(W|A_i=V_{ij}))}$$

(2.3)

### 2.3 Objective Function

The quality of rules is determined by the objective function $Q$ which can be defined as [12]

$$Q = \frac{TP}{TP+FN} \cdot \frac{TN}{FP+TN}$$

(2.4)

where $TP$ represents the number of attack instances correctly classified; $FP$ represents the number of normal instances judged to be attack; $FN$ indicates the number of attack samples to be judged as normal samples; $TN$ represents the number of normal samples correctly classified.

### 2.4 Pheromone Updating

1) Pheromone Initialization

The initial pheromone of different nodes is equal which is inversely proportional to the sum of the pheromone of each attribute, defined as [12]

$$\tau_{ij}(t=0)=\frac{1}{\sum_{i=1}^{a} b_i} \tag{2.5}$$

where the meaning of parameters is the same as equation (1).

2) Update Strategy of Pheromone

The pheromone is updated after rules are constructed, and the update strategy is [12]

$$\tau_{ij}(t+1)=\tau_{ij}(t)+\tau_{ij}(t)\cdot Q, \forall i,j \in R \tag{2.6}$$

where $Q$ represents the quality of the rules as shown in equation (4).

## 3.The Improved Algorithm Framework ——ACSMiner

### 3.1 State Transition Rule

The ant chooses the next node to the rule queue according to equation (7) [13].

$$s=\begin{cases}\arg\max_{z\in allowed_k}\{[\tau_{ij}(t)]\cdot[\eta_{ij}]\}, & if\ q\le q_0\\ S, & else\end{cases} \tag{3.1}$$

where $q$ is a random number that is uniformly distributed in [0,1], $q_0$ is a given parameter (), $S$ is a random variable according to the probability distribution given by equation (1), $allowed_k$ represents the next set of nodes to be selected. When $q<q0$, the ant tends to choose the node of high pheromone and high heuristic factor namely smaller entropy; or the ant explores new paths in a probabilistic manner.

### 3.2 Local Update Strategy

Each ant updates the pheromone in the process of constructing the rule. The local update rule is defined as

$$\tau_{ij}(t+1)=\varepsilon\cdot\tau_{ij}(t)+\rho\cdot\Delta\tau, \forall i,j\in R \tag{3.2}$$

where represents the local renewal factor ; is a given parameter， ; represents the heuristic factor.

### 3.3 Global Update Strategy

The global update strategy is shown in equation (9). To accelerate the convergence speed of the algorithm, the global update factor $w$ is used.

$$\tau_{ij}(t+1)=\tau_{ij}(t)+\mathrm{w}\cdot\tau_{ij}(t)\cdot Q, \forall i,j\in R \tag{3.3}$$

where the meaning of the parameters is the same as section 2.

### 3.4 Algorithm Flow

The algorithm begins with the initialization of the pheromone. In the process of constructing rules, the ant selects nodes using the state transition rule, so that they can construct the rule according to the prior knowledge while exploring a new path; In addition, the pheromone of selected nodes will be reduced by using the local update strategy which can enhance the ability of the ant to search for better solutions; After the rule construction, rules will be pruned which makes rules more robust; After iterations, the global optimal ant will be selected and the additional pheromone will be added to it to increase the difference of the pheromone between the optimal path and the remaining paths, so the ant can search for the optimal solution in a shorter time.

There are two kinds of optimal ants in ACSMiner, one is the best ant in one iteration. The original algorithm only updates the pheromone of this kind of ant; the other is the global best ant, namely the best ant selected from all iterations. In the original algorithm, the rules are produced by this kind of ant. In ACSMiner, the global update strategy is introduced and the pheromone of the global best ant will be updated to provide additional pheromone to the optimal ant, so the ant can find the optimal solution earlier. The pseudo code of ACSMiner is as follows:

**Input**: , where; Search stagnation parameter: *NoConverg*; Number of iterations: *NoIter*; Number of ants: *NoAnts*.

**Output**: *rule sets*.

Initialization: $i$=0; $j$=0; *iter*=0

While the number of training >threshold $M$

Initialize the pheromone of paths;

While *iter* < *NoIter* and $j$ < *NoConverg*

While $i$ < *NoAnts*

Applies equation (7) to select the node;

Apply equation (8) for the local update;

Prune rules after the rule construction;

$i$++; End

Apply equation (9) to update the pheromone of the best ant in one iteration;

$j$++; *iter*++; End

Apply formula(9) to update pheromone of the global best ant；

Delete training samples that meet the rule; End

where $x_i$ represents the features of the $i_{th}$ sample, $n$ represents the feature dimension; $t_i$ represents the label of $i_{th}$ sample, the label dimension is 1. $i$ is the serial number of the ant; $j$ indicates whether the algorithm searches for stagnation; *iter* refers to the number of iterations; $M$ is defined as the threshold of a certain number of samples.

## 4.The Intrusion Detection Model based on ACSMiner

### 4.1 The Algorithm Model

The proposed model based on the ACSMiner can be divided into two steps：

1) *Training phase*: In this phase, the training dataset is used as the input of the ACSMiner algorithm, and the output is the rule set shown as follows: If <$term_1$ and $term_2$ and…$term_n$> Then<*class n*>. The algorithm takes the class of the most instance that meet the rules as the final

value of the class. Finally, instances which satisfy the rules will be removed from the training dataset until the number of the remaining instances is less than the threshold.

2) *Testing phase*: the testing phase mainly do the rule matching using the rule set generated by Step 1, and finally output the evaluation index.

### 4.2 Evaluation Criteria of Performance

The author evaluates the model by using standard evaluation parameters which can be obtained by the confusion matrix. Evaluation parameters include Detection rate (*DR*), False Positive Rate (*FPR*) and *Acc* [11].

## 5. Experimental Results

### 5.1 Data Used and Experimental Parameters

In this paper, we use the KDD99 dataset which can be divided into four categories, namely *Probe*, *DoS*, *U2R* and *R2L*. First of all, discretization and feature selection were performed. In this paper, the supervised-attribute-Discretize algorithm in Weka has been used to discretize the continuous attributes. 14 attributes were selected as the new features [11]. Then, we removed duplicated records, and randomly extract 24524 samples which were divided into two parts *T1* (11705) and *T2* (12819) used as training and testing dataset, respectively.

The selection of parameters, including the number of iterations, the number of ants and the stagnation coefficient, must ensure the convergence of the algorithm. The stagnation coefficient indicates the maximum allowed number that the quality of rule extracted this time is the same as that of rule extracted next time. The number of iterations is 100, the number of ants is about 5, and it is reasonable that the stagnation coefficient is 10; The minimum number of covered rules directly affects the number of extracted rules. According to the discrete degree of the dataset and the number of instances, its value is set to be 3. The allowed maximum number of remaining instances can control when the algorithm will determinate, with its value set to be 10. In this paper, the results are the average of ten experimental executions based on ten randomly extracted data.

| Index | Extracted rules |
|---|---|
| 1 | IF src_bytes = '(52426.5-54614.5)' THEN 'back' |
| 2 | IF src_bytes = '(-inf-0.5)' AND dst_bytes = '(-inf-0.5)' AND serror_rate = '(-inf-0.005)' AND dst_host_count = '(254.5-inf)' THEN 'neptune' |
| 3 | IF flag='S0' AND dst_host_diff_srv_rate='(-inf-0.005)' THEN 'neptune' |
| 4 | IF flag = 'S0' AND serror_rate = '(0.995-inf)' THEN 'neptune' |
| 5 | IF flag = 'S0' AND count='(-inf-1.5)' AND dst_host_diff_srv_rate = '(-inf-0.005)' THEN 'land' |
| 6 | IF protocol = 'tcp' AND dst_bytes = '(-inf-0.5)' AND serror_rate = '(-inf-0.005)' AND srv_diff_host_rate = '(-inf-0.005)' THEN 'portsweep' |
| 7 | IF dst_bytes = '(-inf-0.5)' AND dst_host_count = '(254.5-inf)' THEN 'satan' |
| 8 | IF count= '(-inf-1.5)' AND serror_rate = '(-inf-0.005)' THEN 'normal' |
| … … | |
| 2 | Default rule: normal |
| 4 | |

**Table 1:** a set of rules extracted by ACSMiner

### 5.2 Experimental Results and Analysis

Table 1 lists 24 rules extracted from dataset *T1* by ACSMiner. The last rule <Default rule: normal> means that if an instance does not satisfy the above rules, the instance is considered to belong to *normal* class. Rule 1 <IF *src_bytes* = '(52426.5-54614.5)' THEN '*back*'>, is used to determine the *DoS* attack—"*back*". The src_bytes field represents the number of bytes transmitted from the source host to the destination host. When the "*back*" attack occurs, the source host sends a large number of request packets to the target host until the destination host resource is depleted and can not provide services. This rule is a good reflection of the characteristics of the back attacks. Rule 7 is used to determine the Satan attack which is a kind of *Probe* attack. It scans the hosts which have vulnerabilities and have poor capabilities of defence in the network to steal relevant information. *Dst_host_count* refers to the number of connections that have the same target host as the current connection in the first 100 connections. Rule 7 shows that the target host is likely to be malicious scanning when data transferred between the target host and the source host is small, but the number of connections that have the same target host as the current connection is high. This rule also reflects the characteristics of such an attack.

| | Algorithm | |
|---|---|---|
| | **AntMiner** | **ACSMiner** |
| **Rule** | 23.5 | 24 |
| **term** | 35 | 39 |

**Table 2**: number of extracted rules

| **Algorithm** | **Types** | | | | |
|---|---|---|---|---|---|
| | **Normal** | **Probe** | **DoS** | **U2R** | **R2L** |
| **AntMiner** | 95.25 | 92.58 | 95.58 | 16.67 | 2.19 |
| **ACSMiner** | 99.64 | 91.39 | 95.46 | 12.5 | 2.19 |

**Table 3**: *Acc* on all types

Table 2 shows the number of extracted rules by ACSMiner is almost the same as that of the original algorithm. But the number of term in the rule set is 11.43 percentage points higher than the original algorithm. Since the Normal, *DoS* and *Probe* samples account for a relatively large proportion on dataset *T1*, so the three types of rules are more.

The comparison of the *Acc* is shown in Table 3. A large number of *Norma*l class samples are wrongly classified as *Probe* attacks in AntMiner, and ACSMiner can better identify *Normal* samples. This is mainly because the number of rules and the term extracted by AntMinerare less, which can not well represent the *Normal* samples. The *Acc* of *DoS* and *R2L* is almost the same as the original algorithm, and the *Acc* of *U2R* and *Probe* is slightly lower than the original algorithm. The *Acc* of *U2R* and *R2L*, as shown in Table 3, is lower than that of the other three type, mainly because the samples of these two types of attacks are significantly less than the other two types.

Table 4 gives the comparison of AntMiner, ACSMiner, and SMO [14] algorithm in *Acc*, *DR*, *FPR* and training time. Experimental results show that the *Acc* of ACSMiner has increased by 3.16 percentage points and the *FPR* of ACSMiner has reduced by 4.39 percentage points compared with the original algorithm. Experimental results show that the *Acc* and *FPR* are all improved. It shows that the algorithm can be applied to practical problems. Due to the number of *U2R*, *R2L* training samples outnumber the test samples, which results that the *DR* of all

algorithms is low. Table 4 shows that the *DR* of the three algorithms is almost the same and SMO algorithm is slightly higher than other algorithms.

| | Algorithm | | |
|---|---|---|---|
| | **AntMiner** | **ACSMiner** | **SMO** |
| **Acc(%)** | 92.38 | 95.54 | 97.04 |
| **DR (%)** | 82.74 | 82.47 | 84.59 |
| **FPR (%)** | 4.75 | 0.36 | 0.34 |
| **Trining time（s）** | 171 | 127.4 | 14.64 |

**Table 4**: comparison of simulation results

As can be seen from Table 4, the training time of ACSMiner has been reduced by 25.5 percentage points, from 171s to 127.4s. This is mainly because the global update strategy is introduced to ACSMiner. That means the additional pheromone will be added to the optimal solution, so that the ant can quickly move near the optimal solution which can improve the speed of convergence. In addition, the training time of SMO algorithm is far less than AntMiner and ACSMiner algorithm. It shows that the methods based on ant colony algorithm have slow convergence speed compared with SMO.

## 6. Conclusion

Intrusion detection has been a hot research topic at home and abroad. In this paper, we propose ACSMiner algorithm and apply it intrusion detection. Compared with the AntMiner, experimental results show the accuracy, false alarm rate and training time all have been improved. However, the convergence time is still too long compared with other machine learning methods which need further improvement.

## References

[1]S. X. Wu, W. Banzhaf. *The use of computational intelligence in intrusion detection systems: A review.* Applied Soft Computing, 2010, 10(1): 1-35.

[2]L. T. Chang, C. T. Chun, C. H. Chin. *Intrusive behavior analysis based on honey pot tracking and ant algorithm analysis.* Proceeding of the 43rd annual 2009 international carnahan conference on security technology 2009, 248:252.

[3]S. M. H. Bamakan, H. D. Wang, Y. J. Tian, Y. Shi. *An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization.* Neurocomputing, 2016, 199(26): 90–102.

[4]K. A. P. Costa, L. A. M. Pereira, R. Y. M. Nakamura, et al. *A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks.* Information Sciences, 2015, 294(10): 95-108.

[5]W. Y. Feng, Q. L. Zhang, G. Z. Hu. *Mining network data for intrusion detection through combining SVMs with ant colony networks.* Future Generation Computer System, 2014, 37(7): 127-140.

[6]G. Chen, Q. Chen, W. Guo. *A PSO-based approach to rule learning in network intrusion detection.* Fuzzy Information and Engineering. Berlin: Springer, 2007, 666-673.

[7]E. Soroush, M. S. Abadeh, J. Habibi. *A boosting ant-colony optimization algorithm for computer intrusion detection.* Proceedings of the IEEE 20th International Symposium on Frontiers in Networking with Applications. Vienna, Austria: IEEE Press, 2006.

[8]J. B. He, D. Y. Long, C. Chen. *An Improved Ant-based Classifier for Intrusion Detection.* ICNC 2007, Haikou, China: IEEE Press, 2007, 819-823.

[9]D. Agravat, U. Vaishnav, P. B. Swadas, *Modified ant miner for intrusion detection.* 2010 Second International Conference on Machine Learning and Computing, Washington, DC, USA: IEEE Press, 2010, 228-232.

[10] K. A. Mafaz, N. B. Omar, M. A. Zainab, etal. *Intrusion detection and classification using ant colony optimization algorithm.* The 6th Scientific Conference of the College of Computer Sciences & Mathematics, IEEE Press, Iraqi: 2013, 194-209.

[11]J. B. He, *The improvement of ant based classifier and it's application in Intrusion detection.* Guangzhou: Sun Yat-sen University, 2006

[12]R. Parpinelli, H. Lopes, A. Freitas, *Data mining with an ant colony optimization algorithm.* IEEE Transactions on Evolutionary Computation, 2002, 6(4): 321–332.

[13]M. Dorigo, V. Maniezzo, and A. Colorni. *The Ant System: Optimization by colony of cooperating agents*. IEEE Transaction Systems, Man and Cybernetics, 1996, 26(26): 29-41.

[14]W. W. Cohen. *Fast effective rule induction.* ICML-95, Tahoe city, California, USA, IEEE,1995, 115-123