Crawler Classification using Ant-based Clustering Scheme

Naomi Kuze*, Shu Ishikura*, Takeshi Yagi†, Daiki Chiba† and Masayuki Murata*
*Graduate School of Information Science and Technology, Osaka University, Japan
Email: {n-kuze, s-ishikura, murata}@ist.osaka-u.ac.jp
†NTT Secure Platform Laboratories, Japan
Email: {yagi.takeshi, chiba.daiki}@lab.ntt.co.jp

***Abstract*—Attacks against websites are increasing rapidly with the expansion of web services. More and more diversified web services make it difficult to prevent such attacks due to many known vulnerabilities in websites. To overcome this problem, it is necessary to collect latest attacks using decoy web honeypots and to implement countermeasures against malicious threats. Web honeypots collect not only malicious accesses by attackers but also benign accesses such as those by web search crawlers. Thus, it is essential to develop a means of identifying malicious accesses automatically from mixed collected data including both malicious and benign accesses. In this study, we have focused on detection of crawlers whose accesses has been increasing rapidly. A related study proposed a crawler detection scheme in which crawlers are identified based on the features of well-known crawlers such as Google crawlers. However, the diversity of crawler accesses has been increasing rapidly, and adapting to that diversity is a challenging task. Therefore, we have adapted AntTree, a bio-inspired clustering scheme that has high scalability and adaptability, for crawler detection. Through our evaluations using data collected in a real network, we show that AntTree can detect crawlers more precisely than a conventional scheme.**

***Index Terms*—Intrusion detection, web-based attacks, ant-based clustering, feature vector**

I. Introduction

Web-based attacks on web servers that provide web services are increasing rapidly as the Internet becomes an increasingly important infrastructure and web services become more widespread. Moreover, cyber-terrorism targeting governments, corporations, and other large organizations is increasing, which is becoming a serious problem. However, it is difficult to detect all vulnerabilities in web servers, which can be targets of web-based attacks, due to the rapid growth in the diversity of web services. In other words, detecting attacks using known vulnerabilities is insufficient for preventing all web-based attacks. Therefore, we must collect and analyze information on web-based attacks in order to detect unknown attacks.

To collect web-based attack information, systems called *web honeypots*, which collect and monitor web attacks targeting web servers, are deployed [1], [2]. There are two types of web honeypots: *low interaction* and *high interaction* [3]. Low interaction honeypots emulate vulnerable OSs and applications, whereas high interaction honeypots accommodate actual OS applications. High interaction web honeypots can actually be under attacks, and are therefore used to collect and analyze a variety of web-based attacks [1], [2]. In this paper, the honeypots referred to are high interaction web honeypots.

Honeypots monitor not only malicious accesses but also normal accesses such as crawler accesses by search engines. Therefore, we need to identify the malicious accesses from a large number of collected accesses. Researchers and engineers usually identify malicious accesses manually in most cases, but this is becoming difficult due to the rapid increase in traffic. Therefore, a method to identify malicious accesses automatically is necessary. A related study proposed a method of detecting web attacks that involves first identifying accesses by crawlers and then assuming the others to be malicious accesses [1]. In this scheme of crawler classification, the authors first identify accesses by well-known crawlers such as Google and then identify similar accesses as accesses by other crawlers. However, as web services become increasingly diverse, accesses by web crawlers that browse web pages for web service indexing are also becoming diverse. This leads to more difficulty in detecting crawlers using only known information. Consequently, adapting to such diverse accesses is a challenging task for crawler classification.

In this study, we adopted a bio-inspired clustering scheme for the crawler classification. Natural organisms behave individually and autonomously using only local information, and as a result, a global pattern or behavior emerges at a macroscopic level. Therefore, such mechanisms are advantageous for classifying a lot of data and for detecting unknown malicious threats. Because of these advantages, bio-inspired clustering schemes have been studied by many researchers [4]. In this paper, we use AntTree [5], [6], an ant-based clustering scheme, for crawler classification because ants are typical social insects, and application of models based on their behavior is a major research theme. AntTree was inspired by the behavior exhibited by ants in which they form chains with each other to construct a tree structure. In our application, a tree structure enables us to easily interpret and analyze individual clusters of nodes (data). Moreover, AntTree, of course, retains high scalability and adaptability, which are inherent characteristics of bio-inspired mechanisms. We show that AntTree can identify crawlers more precisely than a conventional scheme [1] can through an evaluation done using communication logs collected in a real network.

The remainder of this paper is as follows. We introduce related work in Section II and explain AntTree in Sec-

tion III. Then, we propose a crawler classification scheme using AntTree in Section IV, and in Section V, we explain our evaluation of our proposed scheme using communication logs collected in a real network. Finally, we conclude the paper and briefly discuss future work in Section VI.

II. Related Work

In this section, we introduce previous studies about web-based attacks and bio-inspired clustering.

A. Web Attack Detection

There are two kinds of methods to detect web attacks: *signature detection*, which uses signatures from known malicious threats, and *anomaly detection*, which detects unknown malicious threats by collecting and analyzing web attacks. The rapid expansion in the number of web applications makes it difficult to detect all vulnerabilities in the applications. Therefore, it is necessary to collect and analyze accesses in order to detect unknown attacks. One method achieves that purpose by capturing traffic on the PCs and servers of users [7], [8], and another method collects web attacks by utilizing honeypots [9]. The former method enables us to detect malicious accesses precisely, but there are some problems in implementing this kind of method. For example, traffic captures invade the user's privacy and reduce the service quality. Therefore, the latter one is more appropriate, and we focus on it in this paper.

B. Bio-inspired Clustering

Swarms of certain animals such as ants, birds, and fish seem to behave intelligently at a macroscopic level, although each component behaves individually and autonomously based only on local information and simple rules. Because of their high scalability, adaptability, robustness, and flexibility, bio-inspired mechanisms are applied in various fields such as industry, chemistry, and economics. Such mechanisms are also applied to clustering schemes. Typical examples include clustering schemes based on particle swarm optimization (PSO) [10] and ant colony optimization (ACO) [11]. Such bio-inspired clustering schemes are suitable for web attack detection because of the following three reasons:

1) for classifying large volumes of data with a low computation cost,
2) for detecting unknown malicious threats,
3) for adapting to dynamically changing traffic (including normal and malicious traffic).

III. AntTree

In this section, we explain AntTree [5], [6], which we adopt to detect accesses by crawlers. AntTree is an ant-based clustering scheme that was inspired by the coordinated behavior exhibited by ants in which they form chains with each other to construct a tree structure. This coordinated behavior is shown in swarms of *Linepithema humile* (Argentine ants) and *Oecophylla longinoda* (weaver ants). *Oecophylla longinoda* ants construct the tree structure in order to make a bridge across an open space, e.g., between leaves or branches, and to build their nest with leaves [12].

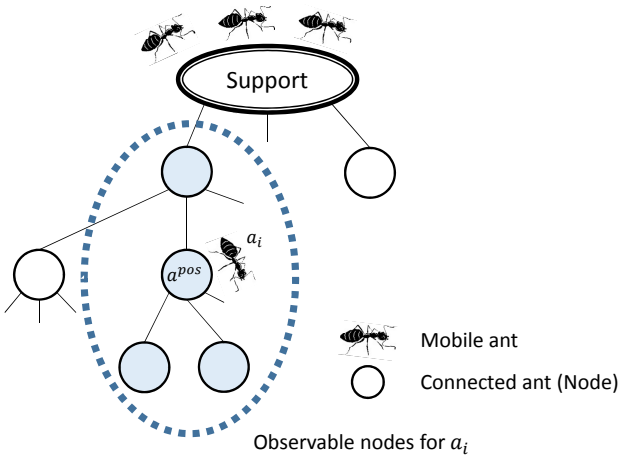

Fig. 1. AntTree

A. Overview

AntTree is a clustering scheme in which data that imitate ants chain with other data according to their similarity to construct a tree structure.

In AntTree, one piece of data corresponds to a mobile agent called an *ant*. These ants chain together to construct a tree structure as shown in Fig. 1. At first, all ants are mobile and exist in the root of the tree, which is called the *support*. Then, the ants start to move away from the support one by one. When an ant starts to move away from the support, it moves among ants that are already connected to the tree (nodes), comparing itself with its neighbor nodes. If the current node is the most similar to the ant within its neighbor nodes, the ant stops moving and becomes a descendant node of the current node. When the ant is connected to the tree, the following ant starts to move away from the support. Note that in this paper, *neighbors* refers to the nodes where the ant currently exists as well as its parent/descendant nodes.

In this scheme, the similarity between ants a_i and a_j ($i, j \in [1, \cdots, N]$, where N is the number of ants) is shown by $Sim(a_i, a_j)$ ($\in [0, 1]$). Ant a_i has similarity threshold $T_{Sim}(a_i)$ and dissimilarity threshold $T_{Dissim}(a_i)$. When comparing itself with ant a_j, ant a_i assumes that it is similar to a_j if $Sim(a_i, a_j) \geq T_{Sim}(a_i)$ is satisfied. On the other hand, ant a_i assumes that it is not similar to a_j if $Sim(a_i, a_j) < T_{Dissim}(a_i)$ is satisfied. Thresholds $T_{Sim}(a_i)$ and $T_{Dissim}(a_i)$ are updated while ant a_i moves around the tree, and they finally converge to the values that are proper for exploring similar nodes.

B. Algorithm

In this subsection, we explain the behavior of ants in detail.

At first, the tree only consists of the support, and all ants exist on it. Then, the first ant starts to move and becomes a

descendant node of the support. While the first ant is connected to the support, the next ant starts to move away from the support.

The following ant a_i first compares itself with the descendant nodes of the support. If there are similar nodes to ant a_i, i.e., there are nodes a_j that satisfy $Sim(a_i, a_j) \geq T_{Sim}(a_i)$, within the descendant nodes of the support, a_i moves to the descendant node that is most similar to a_i. In contrast, if all descendant nodes are not similar to ant a_i, i.e., $Sim(a_i, a_j) < T_{Dissim}(a_i)$ is satisfied for all descendant nodes a_j, a_i becomes a new descendant node of the support and stops moving. When ant a_i is connected to the support, the next ant starts to move. Note that the maximum number of descendant nodes of each node (including the support) is limited to l. Therefore, if the support already has l descendant nodes, ant a_i decreases its similarity threshold $T_{Sim}(a_i)$ by using (1), which is explained later, and moves to the descendant node whose similarity to a_i is the highest. Otherwise, ant a_i updates its similarity threshold $T_{Sim}(a_i)$ and dissimilarity threshold $T_{Dissim}(a_i)$ by using (1) and (2) and then moves to the descendant node whose similarity to a_i is highest.

$$T_{Sim}(a_i) \leftarrow T_{Sim}(a_i) \times \alpha_1, \tag{1}$$

$$T_{Dissim}(a_i) \leftarrow T_{Dissim}(a_i) + \alpha_2. \tag{2}$$

In the case where ant a_i compares itself with ant a_j, a_i is likely to be classified in other clusters if similarity threshold $T_{Sim}(a_i)$ is high and dissimilarity threshold $T_{Dissim}(a_i)$ is high. On the other hand, ants are likely to be classified in the same cluster if both $T_{Sim}(a_i)$ and $T_{Dissim}(a_i)$ are low. At first, $T_{Sim}(a_i)$ is initialized to 1 and $T_{Dissim}(a_i)$ to 0. Ants update these thresholds while moving around the tree, and the ants finally obtain the proper values. α_1 and α_2 are parameters that determine the decrease amount of $T_{Sim}(a_i)$ and the increase amount of $T_{Dissim}(a_i)$, respectively, on their updates. With higher α_1 and α_2, ants can find similar or dissimilar nodes faster, which increases the clustering speed but reduces the accuracy of detection.

When ant a_i arrives at nodes a^{pos} except for the support, a_i first compares itself with a^{pos}. If ant a_i is similar to node a^{pos}, i.e., $Sim(a_i, a^{pos}) \geq T_{Sim}(a_i)$ is satisfied, a_i then compares itself with neighbor nodes of node a^{pos}. If none of the neighbor nodes are similar to ant a_i, i.e., $Sim(a_i, a_j) < T_{Dissim}(a_i)$ is satisfied for all neighbor nodes a_j, a_i becomes a new descendant node of node a^{pos} and stops moving. If node a^{pos} already has l descendant nodes, ant a_i randomly moves to a neighbor node. In contrast, if there are nodes a_j that do not satisfy $Sim(a_i, a_j) < T_{Dissim}(a_i)$, ant a_i updates $T_{Sim}(a_i)$ and $T_{Dissim}(a_i)$ by (1), (2) and then randomly moves to a neighbor node.

If ant a_i arrives at nodes a^{pos} and $Sim(a_i, a^{pos}) \geq T_{Sim}(a_i)$ is not satisfied, a_i randomly moves to a neighbor node.

IV. Crawler Classification using AntTree

In this section, we explain how to classify crawlers using AntTree.

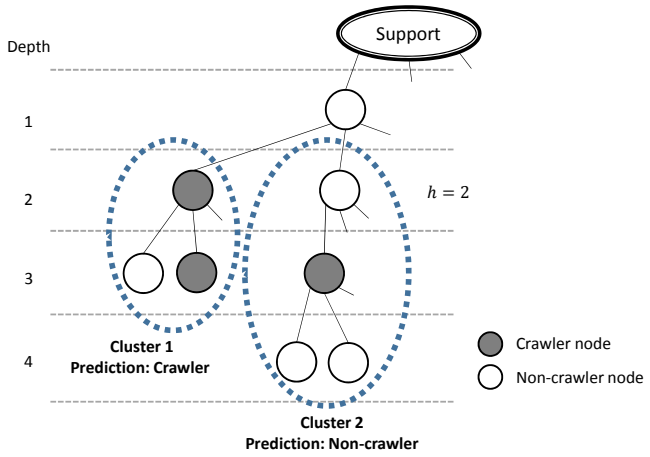

Fig. 2. Cluster interpretation for AntTree ($h = 2$)

A. Similarity

In AntTree, the similarity $Sim(a_i, a_j)$ between ants a_i and a_j is a metric used by moving ants for exploring and constructing a tree structure as explained in Section III. As in [5], we define $Sim(a_i, a_j)$ by (3) using the Euclidean distance $d(a_i, a_j)$ between ants a_i and a_j.

$$Sim(a_i, a_j) = 1 - d(a_i, a_j). \tag{3}$$

The higher similarity $Sim(a_i, a_j)$ is, the more similar ants a_i and a_j are. When an ant (a datum) has M features $\{v_{i_1}, \cdots, v_{i_M}\}$, $d(a_i, a_j)$ is given by

$$d(a_i, a_j) = \sqrt{\frac{1}{M} \sum_{k=1}^{M} (v_{i_k} - v_{j_k})^2}. \tag{4}$$

The feature vector we designed for classifying crawlers is explained in Subsection V-C.

B. Cluster interpretation

We assume that a cluster corresponds to a subtree whose root is an h depth node of the tree constructed by AntTree. Figure 2 shows an example of cluster interpretation of AntTree with $h = 2$.

Each cluster is classified according to which type of data is a majority in the cluster. For example, if the number of crawler nodes is larger than other nodes within a cluster, the cluster is classified to the Crawler cluster, as shown in Fig. 2. If the number of crawler nodes are equal to that of non-crawler nodes in a cluster, the cluster is classified to the type of the root node of the cluster.

V. Evaluation and Discussion

A. Overview

Here, we discuss the evaluation of crawler classification using AntTree using communication logs collected in a real network. Communication logs of accesses by Google are easy to identify because Google opens the information of its crawlers to the public. Therefore, we first identify the Google

TABLE I CONNECTION LOGS COLLECTED BY HONEYPOTS

Label	Number
Google	8,276,246
Crawler	1,502,254
Non-crawler	11,547,739
Other	710,708
Total	22,036,947

TABLE II NUMBER OF FEATURES

Feature	Number
Request	89
Response	37
Total	126

crawlers and then classify communication logs of accesses by other crawlers. We explain the data set and the feature vector in Subsections V-B and V-C, respectively.

In this evaluation, we compared the crawler classification using AntTree with the conventional scheme proposed in [1]. In the conventional scheme, crawlers are classified in accordance with the features of well-known crawlers. In this evaluation, we used Google as well-known crawlers because Google crawlers are easy to identify using public information. Then, we classified communication logs with other crawlers using the features of Google crawlers. We used RandomForest [13] as the classification algorithm to evaluate the conventional scheme.

Note that we used a program written in C++ for the evaluation of AntTree, and RapidMiner 5 for the evaluation of the conventional scheme.

B. Data Set

We used HTTP communication logs collected in a real network for our evaluation. These logs were collected by 37 honeypots [14] from August 29, 2013 to January 14, 2014. Each log included information of request packets that honeypots receive and the responses of honeypots to these request packets. We attached the following labels to about 220.3 million logs collected by honeypots.

- *Google* (about 8.2 million): Communication logs of accesses by Google crawlers are labeled Google. Google logs are classified in accordance with source IP addresses and UserAgents, which Google opens to the public.
- *Crawler* (about 1.5 million): Communication logs with crawlers other than Google are labeled Crawler. Crawler logs are classified in accordance with source IP addresses and UserAgents that researchers and engineers detect by manually analyzing communication logs. Some examples of the detected crawlers include Baidu, Bing, and Microsoft.
- *Non-crawler* (about 11 million): Communication logs with others are labeled Non-crawler. Non-crawler logs include malicious logs.
- *Other* (about 0.71 millions): Other logs correspond to communication logs that are not categorized to either of the aforementioned labels. Most Other logs lack information for their analysis. Therefore, we did not use Other logs in the evaluation.

We used 3,004,508 communication logs including 1,502,254 Crawler logs and 1,502,254 Non-crawler logs as the test data set for this evaluation. Note that we sampled and used only 1,502,254 Non-crawler logs out of all of the Non-crawler logs. In order to evaluate the performance regardless of the access distribution over time, we sampled Non-crawler logs randomly.

In the crawler detection scheme [1], the authors firstly identify well-known crawlers in accordance with source IP addresses and UserAgent, and then identify other accesses having similar features to well-known crawlers as other crawlers. To quantitatively compare the crawler classification using AntTree with that of the conventional scheme [1], we classified crawlers as follows in the evaluation of the conventional scheme.

1) We used Google as the well-known crawlers and randomly sampled 1,502,254 Google logs. Then, we used 3,004,508 logs including 1,502,254 Google logs and 1,502,254 Non-crawler logs as the learning data set.
2) We produced the classification model using the learning data set with RandomForest. In the evaluation of the conventional scheme, we used the same feature vector as that of AntTree explained in Subsection V-C.
3) We classified the test data set in accordance with the classification model. That is, we classified Crawler logs in accordance with the features of Google logs.

C. Feature Vector

We designed a feature vector and used it for crawler classification. There are two types of features: information on request packets and responses to those request packets.

- Request packets:
  Information on HTTP request packets that the honeypots received. Specifically, we use the following information for the crawler classification.
  - Request information: the request URL, the communication method (GET, POST, etc.)
  - Packet header: the UserAgent, the referer, the source/destination port number, the communication protocol (HTTP or HTTPS)
  - Packet body: the body length
- Responses to request packets:
  Information on responses of honeypots to request packets. In detail, we use the following information for the classification.
  - Response type: StatusCodes (200, 404, etc.) defined in RFC 2616
  - Response information: text types (HTML, CSS, etc.) and character encodings (UTF-8, ISO-8859-1, etc.) of content included in response packets if the honeypots send them

All features are normalized and described by real values in $[0, 1]$ for equalizing the weights of features. Feature v_{i_j} of

TABLE III PARAMETER SETTINGS OF ANTTREE

Parameter	Value
l	5
h	3
α_1	0.95
α_2	0.20

TABLE IV ANTTREE

		Prediction		Recall
		Crawler	Non-crawler	
Label	Crawler	1,259,976	242,278	83.87%
	Non-crawler	76,417	1,425,837	94.91%
Precision		94.28%	85.48%	

TABLE V CONVENTIONAL SCHEME [1]

		Prediction		Recall
		Crawler	Non-crawler	
Label	Crawler	1,241,437	260,817	82.64%
	Non-crawler	105,952	1,396,302	92.95%
Precision		92.14%	84.26%	

data i ($i \in [1, \cdots, N]$, $j \in [1, \cdots, M]$) is normalized by

$$\frac{v_{i_j} - \min_{n \in [1,\cdots,N]} v_{n_j}}{\max_{n \in [1,\cdots,N]} v_{n_j} - \min_{n \in [1,\cdots,N]} v_{n_j}}.$$

As an example of features, we explain how to use request URLs included in request packets as features.

1) We use the number of characters of the request URL as a feature.
2) We separate the request URL into path and query parts. The path part gives the location (path) of the requested resource, and the query part specifies parameters. Then, we use the following values as features.
   - The number of levels of the path requested in the path part and the average number of characters of each level of the path.
   - The number of parameters specified in the query part, and the average number of characters of each parameter.
3) We describe the request URL in the form of the regular expression [15] for converting text information to numerical information. We use the types of character strings (string, integer, hex, etc.) and the ratios of strings of each type in the regular expression as features.

The number of features used in this evaluation is indicated in Table II.

D. Results and Discussion

Here, we discuss our evaluation in which we compared the crawler classification performance of AntTree and the conventional scheme. In this evaluation, we used recall and precision as evaluation metrics. The definitions of these metrics are as follows.

- Recall: the fraction of data that are correctly classified within data to which the same label is attached. Given set $\mathcal{L}_A$ of data with label A and set $\mathcal{C}_A$ of data classified to A, recall of A is calculated by

$$\text{Recall} = \frac{|\mathcal{L}_A \bigcap \mathcal{C}_A|}{|\mathcal{L}_A|}.$$

- Precision: the fraction of data that are correctly classified within data classified to the same category. Given $\mathcal{L}_A$ and $\mathcal{C}_A$, the precision of A is calculated by

$$\text{Precision} = \frac{|\mathcal{L}_A \bigcap \mathcal{C}_A|}{|\mathcal{C}_A|}.$$

The results of the crawler classification using AntTree and the conventional scheme are given in Tables IV and V, respectively. In these tables, *label* corresponds to the label attached to each log, as explained in Subsection V-B, and *prediction* corresponds to the result of classification by AntTree or the conventional scheme.

As shown in the Tables, the precision and recall of Crawler and Non-crawler are higher with AntTree than with the conventional scheme. This indicates that Crawler logs can be classified more precisely by using AntTree than the conventional scheme. This is because it is difficult to identify all crawlers according to the features of well-known crawlers (in this evaluation, Google crawlers) due to the diversity of crawler accesses. On the contrary, AntTree is effective for classifying various accesses because similar logs are classified to the same cluster in AntTree whether their features are known or unknown.

Moreover, we found through this evaluation that AntTree can classify data whose features are minor in the entire data set although these minorities of features often make us to overlook them. because each datum explores similar kinds of data using only local information while moving over the tree. Therefore, AntTree can detect new types of accesses from just a few communication logs, which makes AntTree able to adapt to dynamical changes of features of accesses. We will investigate this point thoroughly in the future.

In conclusion, AntTree, a bio-inspired clustering scheme, can classify crawlers more accurately than the conventional scheme can and is suitable for classifying various accesses.

VI. CONCLUSION AND FUTURE WORK

Due to the rapid growth in the diversity of web services, it is becoming increasingly difficult to classify a large number of communication logs using only known features that are detected manually. In this study, we adopted AntTree, an ant-based clustering scheme, to achieve crawler classification that can adapt to a diverse range of web services. Our evaluation results indicated that AntTree can classify crawlers more accurately than the conventional scheme.

As a future task, we will evaluate AntTree by considering the changes in communication features. Meanwhile, statistical information of communication logs, e.g., the intervals and the distribution of packet arrivals, would be an important cue for detecting crawlers and web-based attacks. Therefore,

we will use such statistical features for the classification of communication logs.

ACKNOWLEDGEMENTS

This research was supported by a Grant-in-Aid for JSPS Fellows of the Japan Society for the Promotion of Science (JSPS) in Japan.

REFERENCES

[1] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi, "Heat-seeking honeypots: Design and experience," in *Proceedings of the 20th International Conference on World Wide Web*, Mar. 2011, pp. 207–216.

[2] D. Canali and D. Balzarotti, "Behind the scenes of online attacks: an analysis of exploitation behaviors on the web," in *Proceedings of 20th Annual Network & Distributed System Security Symposium (NDSS 2013)*, Feb. 2013.

[3] N. Provos and T. Holz, *Virtual honeypots: from botnet tracking to intrusion detection*. Addison-Wesley Professional, Jul. 2007.

[4] C. Grosan, A. Abraham, and M. Chis, *Swarm Intelligence in Data Mining*. Springer Berlin Heidelberg, 2006, vol. 34.

[5] H. Azzag, N. Monmarche, M. Slimane, and G. Venturini, "AntTree: a new model for clustering with artificial ants," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2003)*, vol. 4, Dec. 2003, pp. 2642–2647.

[6] H. Azzag, G. Venturini, A. Oliver, and C. Guinot, "A hierarchical ant based clustering algorithm and its use in three real-world applications," *European Journal of Operational Research*, vol. 179, no. 3, pp. 906–922, Jun. 2007.

[7] C. Kruegel, G. Vigna, and W. Robertson, "A multi-model approach to the detection of web-based attacks," *Computer Networks*, vol. 48, no. 5, pp. 717–738, Feb. 2005.

[8] T. H. Kim, K. Kim, J. Kim, and S. J. Hong, "Profile-based web application security system with positive model selection," in *Proceedings of the 2nd Joint Workshop on Information Security*, Aug. 2007.

[9] T. Yagi, N. Tanimoto, and T. Hariu, "Design of provider-provisioned website protection scheme against malware distribution," *IEICE transactions on communications*, vol. 93, no. 5, pp. 1122–1130, May 2010.

[10] S. Alam, G. Dobbie, Y. S. Koh, P. Riddle, and S. U. Rehman, "Research on particle swarm optimization based clustering: A systematic review of literature and techniques," *Swarm and Evolutionary Computation*, vol. 17, pp. 1–13, Aug. 2014.

[11] O. M. Jafar and R. Sivakumar, "Ant-based clustering algorithms: A brief survey," *International journal of computer theory and engineering*, vol. 2, no. 5, pp. 1793–8201, Oct. 2010.

[12] A. Lioni, C. Sauwens, G. Theraulaz, and J.-L. Deneubourg, "Chain formation in oecophylla longinoda," *Journal of Insect Behavior*, vol. 14, no. 5, pp. 679–696, Sep. 2001.

[13] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, Dec. 2002.

[14] T. Yagi, N. Tanimoto, and T. Hariu, "Intelligent high-interaction web honeypots based on url conversion scheme," *IEICE transactions on communications*, vol. 94, no. 5, pp. 1339–1347, May 2011.

[15] T. Nelms, R. Perdisci, and M. Ahamad, "ExecScent: Mining for new C&C domains in live networks with adaptive control protocol templates." in *Proceedings of 22nd USENIX Security Symposium*, Aug. 2013, pp. 589–604.