Regular Article

A Combination of Artificial Neural Network and Artificial Immune System for Virus Detection

Mai Trong Khang[1], Vu Thanh Nguyen[1], Tuan Dinh Le[2]

[1] University of Information Technology, Ho Chi Minh City, Vietnam
[2] Long An University of Economics and Industry, Long An, Vietnam

Correspondence: Mai Trong Khang, khangmt@uit.edu.vn
Communication: received 04 December 2015, revised 10 April 2016, accepted 26 July 2016
Online publication: 22 September 2016, Digital Object Identifier: 10.21553/rev-jec.133
The guest editor coordinating the review of this article and recommending it for publication was Dr. Tran Manh Ha.

Abstract– **In this paper, we propose an Artificial Neural Immune Network (ANIN) for virus detection. ANIN is a combination of Artificial Neural Network (ANN) and Artificial Immune Network (AiNet). In ANIN, each ANN is considered as a detector. A pool of initial detectors then undergoes a mature process, called AiNet, to improve its recognizing ability. Thus, more than one ANN objects can cooperate to detect malicious code. The experimental results show that ANIN can achieve a detection rate of 87.98% on average with an acceptable false positive rate.**

Keywords– **Artificial neural network, artificial immune network, virus detection.**

1 Introduction

Computer viruses are the programs that can inject themselves into benign programs and execute harmful instructions without permission or knowledge of the users. In a typical anti-virus approach, after a computer is infected, the problem is reported and then a solution will be designed and deployed to protect the infected computers [1]. Recently, new algorithms in machine learning and artificial immune system open a new direction for computer virus detection, which is potentially more effective than traditional methods.

Artificial Neural Network (ANN) is one of the most popular algorithms in the field of machine learning which mimics our nervous system. In the ANN model, the basic computing units called neurons are interconnected to create a network that can compute values from inputs [2]. ANN was used to solve human problems in many areas such as computer vision, speech recognition, etc. Despite its strong capability of machine learning and pattern recognition, the training cost of ANN is high. In addition, it is hard to find an appropriate network structure in ANN.

Artificial Immune Network (AiNet) [3] is an Artificial Immune System (AIS) [4, 5] model which is inspired by the idiotypical network theory [6, 7]. The AiNet consists of computing units called antibodies and the connections among them. According to idiotypical network theory, these antibodies not only discriminate among malicious cells (antigens) and benign cells but also recognize and interact with others. AiNet is an unsupervised learning method that was used successfully in many areas including clustering, data visualization and optimizations. However, high computational cost and large number of user-defined parameters are the main disadvantages of the AiNet.

In this paper, we propose a hybrid approach, which combines ANN and AiNet, called Artificial Neural Immune Network (ANIN) for virus detection. Using one ANN is difficult to learn all the virus strings even with a very large number of neurons in the network. In ANIN, each ANN is a detector and a pool of these detectors are used in such a way that they can cooperate to resolve the problem. AiNet is used to train the ANN detectors in both weight and structure. In addition, AiNet can minimize the number of detectors by reducing the overlap between them. Therefore, the approach is more adaptive in applying to large datasets. Our experimental results showed that ANIN can achieve up to 87.98% detection rate with an acceptable false positive rate.

This paper is organized as follows. Section 2 describes the related work. Section 3 introduces our ANIN for virus detection in details. Section 4 presents our experimental results before the paper is concluded in Section 5.

2 Related Work

In [8], the authors use multiple sequence alignment (MSA) techniques from bioinformatics combined with data mining techniques to recognize malware. MSA techniques are used to align variable length computer viral code and then the alignment patterns are obtained with core, invariant regions of the code occupying fixed positions. Data mining techniques such as symbolic rule extraction and ANN then can learn the critical features to determine into which class the aligned patterns fall.

In [9], the authors proposed an Intrusion Detection approach based on the combination of Rough Set methods and Artificial Immune Network. Rough Set method

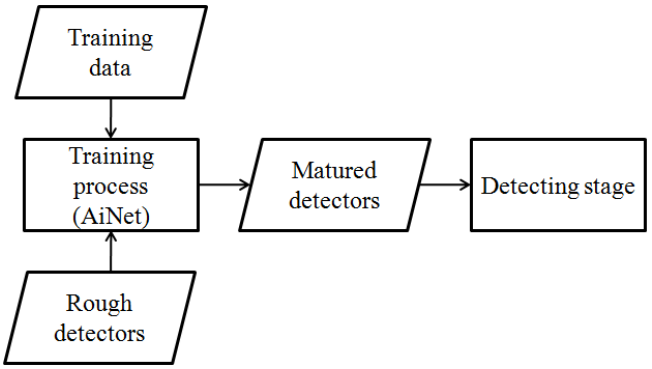

Figure 1. General model of ANIN.

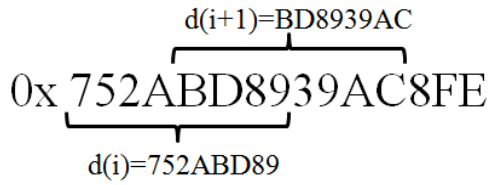

Figure 2. String extraction mechanism.

is applied to reduce the dimension of the dataset by only getting the most significant features. Then, AiNet is used to cluster the attacks in the reduced dataset. The results show a significant improvement in detection rate when using only critical features.

In [10], the most relevant features are extracted from Portable Executable (PE) structure of executable files by Fisher Score and then is learned by an artificial neural network. Although their approach can identify unknown virus patterns, they use only one deployed artificial neural network as learning model which is not efficient in both training cost and performance for large data.

In [11], ANN is integrated with Clonal Selection Algorithm (CSA) to create a new virus detection approach. The CSA is used to train a pool of immature detector to adapt with the problem-space. After the training, mature detectors can cooperate to detect viral code. However, the coverage of detector has not been examined and many irrelevant detectors are obtained, which cause low detection rate.

In [12], the authors combined Negative Selection Algorithm with Artificial Immune Network. Negative Selection Algorithm is used to create the first generation of detectors and AiNet is used to improve detectors' coverage and enhancing the ability of unknown virus detection. The limitation of their approach is that they use strings for detecting strings.

3 The Proposed Approach

This section describes our proposed Artificial Neural Immune Network (ANIN) for virus detection in detail.

ANIN consists of three main components. The first component, called training data, which consists of a set of viral files and a set of benign files and is used for training and detecting stage. The second component is the population of detectors, called antibodies. The detectors are the ANN objects used for recognizing foreign and benign particles. And the third component is AiNet, which can be considered as a training process for generating and maturing detectors. The general model of ANIN is described in Figure 1.

The first part of this section describes the techniques for generating and refining the training data from the collected files. In the second part, the structures of detectors as well as relevant issues are described. The third and the fourth parts describe the training algorithm and the mechanism for using the obtained detectors, respectively.

3.1 Training Data Generation

In our approach, viral files are collected from an online source [13] and benign files are normal programs. After they are collected, the strings with length L are extracted from files. The overlap between two continuous strings is $L/2$. The extraction mechanism is described as shown in Figure 2.

Because computer viruses need a host program, a viral file contains not only malicious codes but also benign codes. The extracted strings from the viral files construct a viral string set. The benign string set contains extracted strings from normal executable files. These strings undergo the Negative Selection Algorithm to remove benign strings from the viral string set. The training data generation algorithm is described in Algorithm 1.

To determine if two strings are considered as *matched*, the Hamming distance or r contiguous distance can be either used. Hamming [5] distance is calculated as

Algorithm 1: Training Data Generation Algorithm

Notation:
Extract(f): return strings extracted from file f.
Match(x, y): return true if string x matches string y.
Initialization:
$S_{VS} \leftarrow \emptyset$
$S_{BS} \leftarrow \emptyset$
$S_{VF} \leftarrow$ viral files
$S_{BF} \leftarrow$ benign files

for each $f \in S_{VF}$ **do**
  $S_{VS} \leftarrow S_{VS} \cup$ Extract(f)
end for
for each $f \in S_{BF}$ **do**
  $S_{BS} \leftarrow S_{BS} \cup$ Extract(f)
end for
for each string $x \in S_{VS}$ **do**
  for each string $y \in S_{BS}$ **do**
    if Match(x, y) **then**
      x is excluded from S_{VS}
    end if
  end for
end for

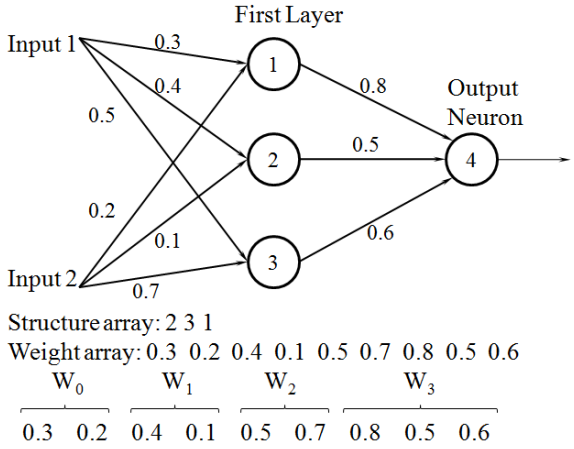

Figure 3. General structure of an ANN object.

follows:

$$h(x,y) = \sum_{n=1}^{N} \overline{X_n \oplus Y_n}, \tag{1}$$

where N is the length of strings, X_n and Y_n are the n^{th} bit of string x and the n^{th} bit of string y. $X_n \oplus Y_n$denotes the XOR logic operation.

In order to reduce the computation cost, the clustering algorithm, called DBSCAN, clusters malicious strings in viral string set [14]. The obtained clusters are then called antigens. In each antigen, the average distance from each element to the others is calculated and the one with the smallest value is kept as a delegate.

3.2 Detectors

In ANIN, detectors are the multilayer feed-forward neural networks with the bipolar sigmoid activation function and are used to discriminate between viral and benign patterns.

To obtain a simple representation scheme for training process, these ANN objects are encoded into two arrays. A weight array is an array of real numbers, which are the weights of connections between neurons in the network. The structure array with integer elements is used to describe the network structure.

The structure array is described as $S_0S_1S_2...S_k$ where $k(k \leq 3)$ is the number of layers in network and $S_j(0 \leq j \leq k)$ denotes the number of neurons in layer j (S_0 is the input layer and S_k is the output layer).

Each ANN's neuron has a set of weights W: $w_0w_1...w_m$ where w_m denotes the weight of a connection from neurons in the previous layer to this neuron. Therefore, the weight array can be constructed by: $W_iW_{i+1}...W_n$ where n is the number of neurons and $w_i(0 \leq i \leq n)$ is the neuron i^{th} in the network. An ANN object with structure array: 2 3 1 and weight array: 0.3 0.2 0.4 0.1 0.5 0.7 0.8 0.5 0.6 is described in Figure 3.

In addition, to determine how well a detector works, the fitness function is used to calculate its antigenic affinity value f as follows:

$$f = \frac{2}{\text{error} + \text{fpr}}, \tag{2}$$

where fpr is the false positive rate, and the error is calculated as

$$\text{error} = \frac{1}{\text{n}} \sum v_i, \tag{3}$$

where v_i denotes the absolute difference between computed output and string's labelled value in antigen and n is the number of strings in antigen.

As mentioned earlier, each detector in the network can interact with both the foreign particles and other objects. To determine the interaction among an ANN object and the others, we introduce a binary array whose elements are the recorded recognition values of the ANN object with respect to the learning data. Hamming Distance is then used to calculate how well an ANN object interacts with the others from these arrays.

3.3 Training Algorithm

After the detectors are created, they undergo a mature process to improve their affinity. The process consists of 2 main parts: the first part is the interaction between detectors and antigens, and the second part is the interaction among detectors themselves. The general form of AiNet algorithm is described as in Algorithm 2.

In the first part of the algorithm (inside the loop for), the affinity values of detectors with presented antigen are calculated and these detectors are cloned proportionally to their affinity. Metadynamics step removes clones whose affinity is less than a predefined threshold. The interaction and suppression operation in this part are called the clonal interaction and the clonal suppression, and are used for determining the interaction between clonal detectors and removing clones based on predefined threshold, respectively.

In the second part, the similarity between each pair of network antibodies is calculated by the procedure Interaction (Smemory). In the Network Suppression step, the procedure Suppression (Smemory) will eliminate antibodies whose interacting affinity is above the predefined threshold. New random antibodies are generated.

In order to stop the training process, various forms of stopping criterion have been used such as a number of iterations, a predefined affinity value or a predefined threshold is used to examine the difference between two continuous generations of detectors. In the ANIN, the absolute difference between two average fitness values of consecutive generations is calculated and compared to a predefined value to check the stop iterations. Instead of using a fixed threshold, a mechanism is introduced to define the value manually during the learning process. In each iteration, training statistics are recorded to help observe the learning and stop the process at any time.

3.4 Detection Mechanism

In our approach, dangerous level is used to determine how dangerous a file is. A file with a large value of dangerous level is considered as dangerous. The dangerous level of a file is calculated based on one of

Algorithm 2: General Form of AiNet Algorithm

Notation:
Smemory: set of memory detectors.
Sclonal: set of clonal detectors.
X, Y: either Smemory or Sclonal, depending on the input of the algorithm's procedure.
AffinityComputation(X, a): compute affinities of all elements in X with respect to antigen a.
ClonalSetection(X): select a number of elements with highest affini-ty in X and reproduce them proportionally to their affinity.
Metadynamics(X): remove elements in X whose affinity with current antigen is less than a predefined threshold.
Interaction(X): determine the network interaction among elements in X.
Suppression(X): Remove elements whose affinity with each other is more than a predefined threshold.
Update(X, Y): Incorporate elements in X with all elements in Y.
DiversityHandling(X): Introduce a set of new randomly generated element into the X.

while not stopping criterion met **do**
  for each antigen a **do**
    AffinityComputation(Smemory,a)
    Sclonal ← ClonalSetection(Smemory)
    Metadynamics(Sclonal)
    Interaction(Sclonal)
    Suppression(Sclonal)
    Update(Sclonal, Smemory)
  end for
  Interaction(Smemory)
  Suppression(Smemory)
  DiversityHandling(Smemory)
end while

all the strings extracted from the file. The dangerous level of a string is computed by averaging the value outputted by all the detectors as follows:

$$DL(s) = \frac{\sum_{i=1}^{|DS|} \text{compute}(d_i, s)}{|DS|}, \tag{4}$$

where DS is the detector set, s is the presented string and $DL(s)$ is its dangerous level, d_i is the i^{th} detector in DS and

$$\text{compute}(d_i, s) = \begin{cases} 1, & \text{if } s \text{ is dangerous to } d_i, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The dangerous level of a file is determined in the following as the rate between viral strings and total strings and is used to detect that file:

$$DL(F) = \frac{\sum DL(s)}{|F|}, \tag{6}$$

where $|F|$ is the number of strings in file F. These dangerous level concepts are introduced in [15].

The dangerous levels of files in training data are calculated and are used to construct new training data, whose element has two field $<$ dangerous level, label $>$. An ANN object is then deployed as a learning model to study these data and is used to detect files in the testing set.

Table I
Training Data

Data	Training files		Testing files	
	Viral Files	Benign Files	Viral Files	Benign Files
Dataset 1	80	20	10	10
Dataset 2	200	50	25	25
Dataset 3	400	100	50	50
Dataset 4	800	200	100	100
Dataset 5	1600	400	200	200
Dataset 6	2000	500	250	250

4 Evaluation

A virus detection program was implemented in C# to evaluate our approach. The program allows the users to define parameters manually as well as observe the learning process and the experimental results.

4.1 Evaluation Metrics

In order to evaluate ANIN, the following metrics are used [12, 15]:

- **Detection rate and false positive rate**: The first experiment was carried out to study the detection rate and the false positive rate of our model as well as the effect of the number of files in the dataset on the performance of the model.
- **The correlation between Network suppression threshold and the performance of the model**: In the Network Suppression step of the training algorithm, any pair of detectors in the network, whose affinity is greater than the predefined threshold, are excluded as well as new detectors are generated into the pool. Network Suppression Threshold (NST) has a significant impact on the total coverage of detectors, which are important to the ANIN performance. In our experiment, we investigate how NST affects the detection rate as well as the false positive rate of ANIN.

In order to evaluate our approach, we also compare our approach with the approach in [11].

4.2 Training Data

In ANIN, the experimental data are divided into six smaller datasets. In each dataset, the viral files and benign files are randomly selected from the collected data. The training data are shown in Table I.

4.3 Experimental Results

The results of the first experiment are shown in Table II.

As shown in Table II, ANIN can achieve up to 92% detection rate (Dataset 2). The average detection rate

Table II
The Correlation between the Number of Files in Dataset and the Performance of the Models

Data	Detection rate (%)		False Positive Rate (%)	
	ANIN	Method in [11]	ANIN	Method in [11]
Dataset 1	90	100	0	0
Dataset 2	92	96	4	8
Dataset 3	88	84	8	6
Dataset 4	87	90	13	14
Dataset 5	86.5	83.5	12.5	11.5
Dataset 6	84.4	80.4	14.8	16.4
Average	87.98	88.98	8.72	9.32

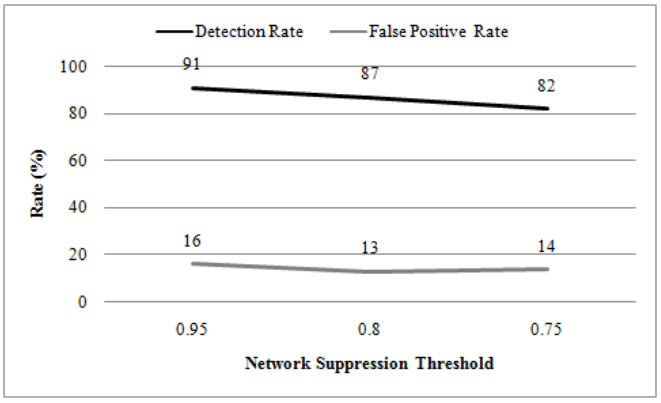

Figure 4. The correlation between Network Suppression Threshold and Detection Rate as well as False Positive Rate on Dataset 4.

and false positive rate achieved for the 6 datasets are 87.98% and 8.72% respectively. Therefore, with appropriate predefined parameters (the NST is 0.8 and the stopping value is 0.05), the detection rate is reasonably high with the acceptable false positive rate (below 15%). We also observe that there is a relationship between the size of the dataset and the performance of the model. A large dataset may result in worse performance (lower detection rate and higher false positive rate). This is due to the fact that a large dataset creates a larger problem-space for detectors to cover and thus, much effort is required to train the detectors. In this case, the results may not be good if the number of detectors is not large enough or the learning process fails to converge.

Figure 4 shows the correlation between Network Suppression Threshold (NST) and the detection rate as well as false positive rate on the Dataset 4. Our results showed that the detection rate could achieve up to 91% with 0.95 network suppression threshold. We also observe that the larger the network suppression threshold is, the higher the detection rate is. With the large value of NST, the training process forces the detectors to cover new area instead of occupied areas. In such situation, with the same detectors, the total coverage is larger, which results in higher detection rate. However, as we can see in Figure 4, too large or too small NST value may cause high false positive rate.

Table II compares the detection rate and false positive rate between our approach and the approach in [11]. As shown in Table II, the average detection rate of the method in [11] is higher than ANIN average detection

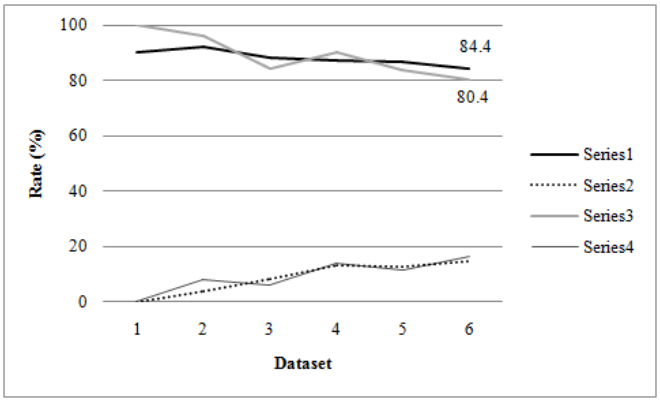

Figure 5. The comparison of experimental results between ANIN and the method in [11] (Series 1: ANIN detection rate, Series 2: ANIN ANIN false positive rate, Series 3, Series 4: detection rate and false positive rate of the method in [11]).

rate. However, the detection rate of the method in [11] is lower than ANIN detection rate when the testing set is larger (see Figure 5). The detection rate of the method in [11] is 100% with Dataset 1 whereas ANIN detection rate is 90%. When applying to Datasets 5 and 6, the detection rate of the method in [11] are 83.5% and 80.4% whereas ANIN has higher detection rate (86.5% and 84.4% respectively). There is not much difference in false positive rate between the two methods. With the same amount of detectors in the training process, ANIN tries to reduce the overlap between them. Therefore, the total coverage is better.

5 Conclusion

In this paper, we propose an Artificial Neural Immune Network (ANIN) for virus detection, a hybrid approach which combines ANN and AiNet. ANIN takes the benefits of both the recognition ability of ANN model and the adaptive ability of AiNet model in training a number of detectors concurrently. The detectors are ANN objects that undergo a mature process to improve both their structure and weights. The result is a pool of matured detectors that can cooperate to detect malicious codes. The experimental results show that ANIN can achieve 87.98% detection rate on average with an acceptable false positive rate. In the future, we will study on how to apply data mining techniques and rough set techniques in ANIN to extract the most critical features in order to improve the training process.

6 Acknowledgment

This research is funded by Vietnam National University, Ho Chi Minh City (VNU-HCM) under grant number C2016-26-05.

References

[1] E. Al Daoud, I. H. Jebril, and B. Zaqaibeh, "Computer virus strategies and detection methods," *Int. J. Open Problems Compt. Math*, vol. 1, no. 2, pp. 12–20, 2008.

[2] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
[3] L. Nunes de Casto and F. J. Von Zuben, "An evolutionary immune network for data clustering," in *Proceedings. Sixth Brazilian Symposium on Neural Networks*. IEEE, 2000, pp. 84–89.
[4] D. Dasgupta and F. Nino, *Immunological computation: theory and applications*. CRC press, 2008.
[5] J. Al-Enezi, M. Abbod, and S. Alsharhan, "Artificial immune systems-models, algorithms and applications," 2010.
[6] N. K. Jerne and J. Cocteau, "Idiotypic networks and other preconceived ideas," *Immunological reviews*, vol. 79, no. 1, pp. 5–24, 1984.
[7] N. K. Jerne, "Towards a network theory of the immune system." in *Annales d'immunologie*, vol. 125, no. 1-2, 1974, pp. 373–389.
[8] Y. Chen, A. Narayanan, S. Pang, and B. Tao, "Multiple sequence alignment and artificial neural networks for malicious software detection," in *Eighth International Conference on Natural Computation (ICNC)*. IEEE, 2012, pp. 261–265.
[9] M. A. Rassam and M. A. Maarof, "Artificial immune network clustering approach for anomaly intrusion detection," *Journal of Advances in Information Technology*, vol. 3, no. 3, pp. 147–154, 2012.
[10] S. Shah, H. Jani, S. Shetty, and K. Bhowmick, "Virus detection using artificial neural networks," *International Journal of Computer Applications*, vol. 84, no. 5, 2013.
[11] V. T. Nguyen, N. P. Anh, M. T. Khang, N. H. Ngan, N. Q. Thai, and N. T. Quoc, "A combination of clonal selection algorithm and artificial neural networks for virus detection," in *Advances in Computer Science and its Applications*. Springer, 2014, pp. 95–100.
[12] V. T. Nguyen, T. T. Nguyen, K. T. Mai, and T. D. Le, "A combination of negative selection algorithm and artificial immune network for virus detection," in *Future Data and Security Engineering*. Springer, 2014, pp. 97–106.
[13] "Computer virus collection," 2015. [Online]. Available: https://vxheaven.org/vl.php
[14] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.
[15] R. Chao and Y. Tan, "A virus detection system based on artificial immune system," in *International Conference on Computational Intelligence and Security (CIS'09)*, vol. 1. IEEE, 2009, pp. 6–10.

Mai Trong Khang received the B.S. degree in Computer Science from University of Information Technology, Ho Chi Minh City, Vietnam in 2013. He is currently pursuing the M.S. degree in Computer Science from University of Information Technology, Ho Chi Minh City, Vietnam. He is a Teaching Assistant and Researcher at Software Engineering, University of Information Technology, Ho Chi Minh City, Vietnam. His research interests include virus detection and computer security.

Vu Thanh Nguyen received the Ph.D degree in Information Technology from Moscow State University and Russian Academy of Sciences. Assoc. Prof. Dr. Nguyen is the Dean of the Faculty of Software Engineering, University of Information Technology, Ho Chi Minh City, Vietnam. His research interests include software technology and knowledge technology.

Tuan Dinh Le received the Ph.D degree in Computer Science and Engineering from the University of New South Wales, Sydney. He is currently the Vice Rector at the Long An University of Economics and Industry, Tan An City, Long An, Vietnam. His research interests include novel application, low power communication, security and compressive sensing in wireless sensor network and Internet of Things (IoT). Dr. Tuan published regularly in top rated sensor network and mobile computing journals.