J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr
Copyright ⓒ 2020 KSII

A Feature-Based Malicious Executable Detection Approach Using Transfer Learning

Yue Zhang[1,2] Hyun-HoYang[2*] Ning Gao[1]

ABSTRACT

At present, the existing virus recognition systems usually use signature approach to detect malicious executable files, but these methods often fail to detect new and invisible malware. At the same time, some methods try to use more general features to detect malware, and achieve some success. Moreover, machine learning-based approaches are applied to detect malware, which depend on features extracted from malicious codes. However, the different distribution of features oftraining and testing datasets also impacts the effectiveness of the detection models. And the generation oflabeled datasets need to spend a significant amount time, which degrades the performance of the learning method. In this paper, we use transfer learning to detect new and previously unseen malware. We first extract the features of Portable Executable (PE) files, then combine transfer learning training model with KNN approachto detect the new and unseen malware. We also evaluate the detection performance of a classifier in terms of precision, recall, F1, and so on. The experimental results demonstrate that proposed method with high detection rates andcan be anticipated to carry out as well in the real-world environment.

☞ keyword : Malicious Executable Detection, Transfer Learning, Feature-Based

1. Introduction

With the rapid development and wide application of computer and network, people pay more and more attention to computer virus. A computer virus is defined as a set of program code that a programmer inserts into a set of program code, which is transmissible, insidious, infective, and destructive. Computer virus can destroy computer's function or data, such as compromising the system security, damaging the systems, or obtaining sensitive information without the user's permission. Currently, there are two main virus scanning technologies: one is signature-based detection technique and the other is heuristic classifier for detecting unseen or new viruses [1]. While signature-based scanning is effective against existing executable malware, it is virtually ineffective against invisible or new viruses. According to statistics, between 8 and 10 malicious programs are created every day, most of which cannot be accurately detected until signature methods arrives. However, these signature-based approach protection systems are often vulnerable to attacks. Heuristic scanners try to recompense for this gap by using more general feature from viral code, such as structural or behavioral patterns [1]. Nevertheless, this procedure still requires human involvement and cannot be fully automated, and the final models still failed to get good detection rates and false positive rates for new and unknown viruses.

In this paper, we propose a novel Feature-Based Detecting Approach (FBDA) to detect previously unknown variants of malicious executables by using a feature-based transfer learning approach. The core idea of FBDA method is to find the optimized feature representations from training and testing of executable program dataset. These representations can be obtained by feature extraction via leveraging information gain and principal component analysis (PCA) method [2-3]. Then, KNN and FBDA are used to detect malicious executables. Experimental results show that KNN with FBDA approach is an effective and promising method for detecting malicious executables. The main contributions of this paper are outlined as follows:

(1) We design a novel malicious executable file detection scheme of malicious executable files by using transfer leaning model and feature-based approach.

1 Dept. of Information Science and Technology, Jiujiang University, Jiujiang, 332005, China
2 Dept. of Computer Information & Communication Engineering, Kunsan National University, Kunsan, 54150, South Korea
* Corresponding author: (hhyang@kunsan.ac.kr)
[Received 13 April 2020, Reviewed 13 May 2020(R2 1 August 2020), Accepted 25 August 2020]

(2) We propose a new feature extraction algorithm. The algorithm first employs the information gain method to extract the data features, and then use the PCA method to further optimize the features extraction, which greatly improves the feature extraction effiiency.
(3) The KNN method is selected to detect the malicious executable file through experimental comparison, which can further improve the accuracy and efficiency of malicious executable files detection.
(4) We conduct extensive experiments to verify the detection effiiency of proposed scheme in terms of precision, recall, F1, and so on. Experimental results show the proposed scheme has the best performance in these comparison methods.

The rest of this paper is organized as follows: Section 2 introduces the background. Section 3 describes the feature-based detection method. We demonstrate our experimental settings and results in Section 4. The last section, Section 5, we conclude our works and puts forward the further work.

2. Background

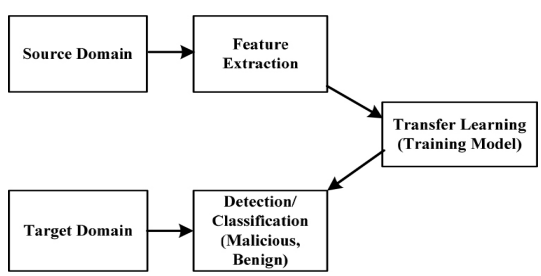
Malicious program identification is not a new research topic, it has a long history. Due to the characteristics of fast spread and strong destruction of computer virus, once infected with the computer virus, it will often cause serious harm to the computer application environment. At the beginning of the emergence of computer viruses, people begin to study the identification methods of viruses, so there is a long history of awaring and studying computer virus identification. But most of the previous methods are basically based on signature.

Signatures are often analyzed manually by experts, and their expertise is used to distinguish them from malicious executables and benign programs. Ultimately, the generatedsignatures consists of many different properties.

Although signature-based detection has a high accuracy rate, it is only effective for small datasets of malicious programs and viruses that have been seen, it has a low accuracy rate for new or unknown viruses. How to seek a new method to automatically generate classifier has become hot topic in the field of anti-virus research. To solve this problem, many researchers applied Artificial Neural Networks (ANNs)for detecting boot sector malicious binaries. Recently, many researchers also have applied machine learning, deep learning, reinforcement learning and transfer learning methods to network attack detection, and achieved good results [4-6]. Using these methods, it not only greatly improves the accuracy of virus detection, but also reduces the detection time. In aword, faster and more accurate virus detection can be obtained in a shorter time, which makes a certain contribution to the field of computer virus detection and makes a big step forward in virus detection technology. However, in the field of computer virus detection, machine learning, deep learning, reinforcement learning and transfer learning are still in its infancy. In [7], Olivier et al proposed a feature selection scheme for computer virus detection, which scans the short sequences of n bytes from the files, then the Intra-Family Support approach and Inter-Family Support approach are used to select and reduce the features. Three data mining methods, LibBFD, GNU Strings and Byte Sequences, are also used to extract the new malicious' features [8]. In [9], a Deep Graph Convolutional Neural Networks (DGCNNs) wasto directly learn from API call sequences and their associated behavioral graphs for behavioral malware detection, which can effectively distinguish malicious and benign software.

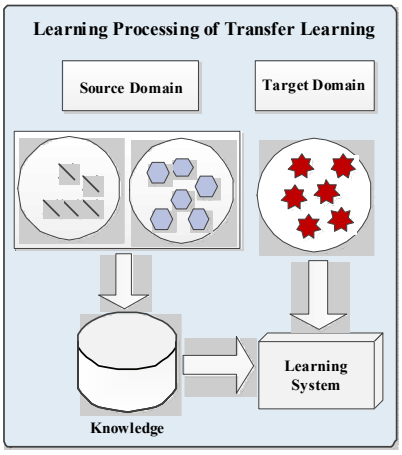
3. The Feature-Based Detection Method

In this paper, we propose a feature-based malicious executables detection approach using transfer learning. Figure 1 depicts the basic framework of the proposed approach.



(Figure 1) The framework of proposed approach.

we have a labeled source domain $D_s = \{x_i, y_i\}_{i=1}^{n}$, where x_i is the feature of the source data, y_i is the labels of the source data, also, we have a unlabeled target domain $D_t = \{x_j\}_{j=n+1}^{n+m}$, where x_j is the feature of the target data, the source domain and the target domain are drawn from different data distributions $P(X_s) \neq P(X_t)$. This is where transfer learning is superior to machine learning. In addition, compared with traditional machine learning methods, transfer learning is more suitable for situations where the sample size of the source domain is small [10]. Our goal is to accurately predict the labels of target domain D_t. The main ideas of the proposed approach are as follows. We first extract the features from the source domain, and then apply the extracted features to the training of the transfer learning model. Ultimately, the training results of the transfer learning model are used to classify the target domain, i.e., detect the malicious executables. The general process of transfer learning is shown in Figure 2.



(Figure 2) A simple illustration of transfer learning process.

3.1 Feature Pre-Extraction

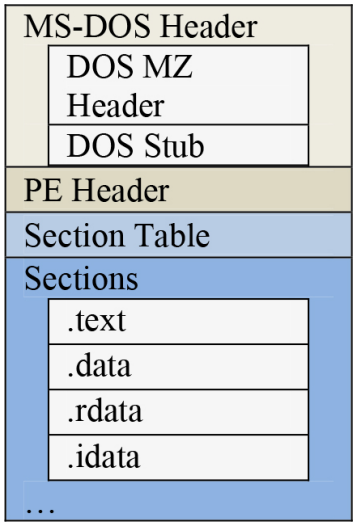
In this paper, we run executables in a virtual environment and extracted features x from the PE file. PE is an acronym of portable and executable file, this file format that comes with Win32. It can be recognized and used by any WIN32 PE loader, even if Windows System is running on a non-Intel CPU. Some of the features of PE are inherited from Unix Coff. The PE file format is shown in Figure 3. The PE file contains many fields, which are briefly described in this paper, but more details can be found in reference [11].

MS DOS header: which contains both the DOS MZ header and the DOS Stub. Once the program executes under DOS, the DOS recognizes that this is a valid execution body and runs the DOS stub immediately after the MZ header. DOS Stub is an actual valid EXE that simply displays the error message on the operating systems that do not support the PE file format.

PE header: PE Header is the short of PE-related structure IMAGE_NT_HEADERS, which contains many important domains used by PE loaders. When the execution body is executed on the operating system that supports the PE file structure, the PE loader finds the starting offset of the PE header from the DOS MZ header. This will skip the DOS stub and directly locate the actual PE file header.

Section Table: which contains the attributes, file offsets, virtual address offsets, and so on ofthe corresponding section.

Sections: the actual data related to each section. The most important are .text (which contains code instructions), .data (containing initialized global and static variables), .rdata (containing constants and other directories such as debug), and .idata (containing import information used in the file).



(Figure 3) The PE file format.

Malicious users can change PE files into malicious executable by rewriting, adding, importing other files or shell and other operations. We can preliminarily judge whether it is a malicious executable by checking the features of PE file, such as compilation time, import functions, whether it contains debug information, the number of exported functions, whether it contains resource files, whether it contains semaphore, whether it enables redirection, whether it enables TLS callback functions and others. Due to the large number of these attributes, the feature selection (information gain) method is used to select the most relevant features.

Information Gain: Information Gain (IG) method is the most common and popular feature selection approach in machine learning. Let X be a finite set of samples, if there are m classes $\{C_1, C_2, ..., Cm\}$ in X, the entropy of X is defined as Equation (1), where Pi is the proportion of class i. H(X) measures the distribution randomness of samples in X over m possible classes. Suppose that $Y = \{Y^1, Y^2, \ldots, Y^P\}$ is the set of attributes and each attribute Y^P has k values $\{V_1^P, V_2^P, ..., V_k^P\}$, Equation (2) represents the entropy of X under Y^P. Then IG value can be computed by Equation (3), where $|X_j^P|$ is the number of samples in X in which the attribute Y^P value in X is V_j^P. The value of Equation (3) reflects additional information about X provided by Y^P. The higher the IG value, the purer the distribution of samples in X over m possible classes.

$$H(X) = -\sum_{i=1}^{m} P(C_i)\log_2 P(C_i) \quad (1)$$

$$H\left(Y^P, X\right) = -\sum_{j=1}^{k} \frac{\left|X_j^P\right|}{|X|} \times H\left(X_j^P\right) \quad (2)$$

$$IG\left(Y^P, X\right) = H(X) - H\left(Y^P, X\right) \quad (3)$$

According to the value of IG, Table 1 lists the pre-extracted main features (lines 1-9) and optional features (lines 10-12). Since the IG value of the optional feature is very small, in order to improve the recognition efficiency, it is ignored in this paper. If the recognition speed is not high requirement, you can selcect to use the these optional features. Since these features are included in each executable, the executables can be defined as a array $X = (x_1, x_2, \ldots, x_n)$, in which the gradient of a vector describes an executable.

(Table 1) The main extracted features.

Feature	Description
Time Date Stamp	The times tamp, date stamp indicates the time/date the file is created.
Number of Rva and Sizes	The length of the data directory array that follows.
Image Directory Entry Import	Import directory. Identifies the function information that program imports from other modules.
Size of image	The image size on disk.
Address of Entry Point	Memory address of the first byte of the section relative to the image base.
Section Min Entropy	Calculate the min entropy of the section.
Virtual Size	The size of the section when loaded into memory.
Characteristics	Properties of this file.
Malware	Indicates whether the file is malicious or not.
Size of Code	The size of all the code sections.
Number of Relocations	The number of relocations in the relocation table.
Forwarder Chain	Relates to forwarding.

3.2 Feature Optimization

To improve the data processing speed and the timeliness of malicious program detection, we use Principal Component Analysis (PCA) approach to reduce the dimensionality of the pre-extracted multidimensional feature groups and reduce the number of features. The features dimensionality reduction method based on PCA is described in detail as follows. In the samples, the mean of vector is calculated as:

$$m = \frac{1}{n}\sum_{i=1}^{n} X_i \quad (4)$$

where n is the total number of samples in the data set, $X_i=(x_{i1}, x_{i2}, \ldots, x_{in})$ is the sample i. The deviation of the mean is defined as:

$$\overline{\phi_i}=X_i-m \tag{5}$$

The sample covariance matrix of the data set is defined as:

$$\begin{aligned} C &= \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i-m\right)\left(X_i-m\right)^T \\ &= \frac{1}{n-1}\sum_{i=1}^{n}\overline{\phi_i}\overline{\phi_i}^T \\ &= \frac{1}{n-1}\Phi\Phi^T \end{aligned} \tag{6}$$

where

$$\Phi=\left[\phi_1,\phi_2,\cdots,\phi_n\right].$$

There are usually two methods for dimensionality reduction of PCA: eigenvalue decomposition and Singular Value Decomposition (SVD). The SVD method is more effective when the data set contains a large number of samples. In this paper, we employ SVD to reduce the dimension of PCA.

The eigenvalues and eigenvectors of the sample covariancematrix can be calculated by SVD method, which is expressed by λ and ω respectively. Then K eigenvectors with the largest eigenvalues are selected, the value of K can be determined by the inequality (7)

$$1-\frac{\sum_{i=1}^{k}S_{ii}}{\sum_{i=1}^{n}S_{ii}}\le\alpha \tag{7}$$

where S_{ii} is a matrix generated by SVD, α is the error of the effect of the change in the subspace on the total change in the original space. The value of α can be set freely according to actual needs. In this paper, we set it as 0.01. The matrix U can be calculated, which size is $N*K$. The principal component data is represented by K-dimensional subspace as follow:

$$Z^{(i)}=U^T-X^{(i)}=U^T(X_i-m)=U^T\overline{\phi}_i \tag{8}$$

where $i\in\{1,2,\ldots n\}$ Algorithm 1 gives the detailed process of feature extraction.

Algorithm 1: The feature extraction algorithm.

Input : Sample set D and features set A;
Output : New sample set D';
1. Compute the entropy of X by Equation (1);
2. Compute the condition entropy of X by Equation (2);
3. Calculate information gain IG,
   $IG(X, A)=H(X)-H(Y^P, X)$;
4. $X=IG(X, A)$;
5. Calculate the covariance matrix XX^T;
6. Perform the eigenvalue decomposition of XX^T;
7. Select n maxinum eigenvalues $(W_1, W_2, \cdots, W_n)$, and combine n eigenvalues into W;
8. **for** $i=1$ *to* m
9. Convert a new sample $Z^{(i)}=W^T X^{(i)}$;
10. **endfor**
11. **Return** $D'=(Z^{(i)}, Z^{(i)}, \cdots Z^{(i)})$.

3.3 Classification

The K-Nearest Neighbor (KNN) is a very classical classification method of supervised learning. In this paper, the training set is represented as (X_i, Y_i), $i\in\{1,2,\ldots n\}$ where X_i is the input sample, and Y_i is the corresponding category. The samples can be divided into two categories, 0 and 1, where 0 represents the benign program, and 1 represents the malicious executable. When a new sample X_j comes, K samples closest to the sample are first selected in X_i, and then the voting mechanism is used to determine the category of the new sample.Finnally the category with the most votes among the K closest new samples are regarded as the category of the new sample X_j. In this paper, the Euclidean distance is used to select K samples closest to X_j. The Euclidean distance can be represented as:

$$D(x_i, x_j) = \sqrt{\sum_{d=1}^{n} | x_i^{(d)} - x_j^{(d)} |^2} \quad (9)$$

4. Performance Evaluation

In this section, we will compare of the proposed method with other methods, which include AdaBoost+FBDA, Decision Tree+FBDA, K- Neighbor+FBDA, Logistic Regression+FBDA, Random Forests+FBDA on the real PE file data set from the aspects of precision, recall, F1 and so on.

4.1 Datasets Description

In this paper, we employ the pe-files-malwares dataset, which contains benign and malicious PE Files. The PE section headers are extracted from the 'pe_sections' elements, and the malwares files are downloaded from virusshare.com, and the benign are downloaded from the Operation System (OS) of a Windows Server 2018. The datasets includes "dataset_malwares" and "dataset_test" two parts, where the "dataset_malwares" is used for training, and the "dataset_test" is used totest. The training set contains a total of 19,612 files, including benign and malicious files, and the testing set contains 18 files.

4.2 Experimental setting

The experiment arecarried out on a Dell computer with 2.2GHz IntelCore i7-3770 CPU and 16G RAM.

The detection of malicious executable can be regarded as a binary classification problem to distinguish between malicious executablesand normal executables. To evaluate the detection capability of the proposed method for new and unseen malicious executables and reduce the evaluation error, we use the *k*-fold cross-validation scheme. In this scheme, test dataset containing *N* samples is divided into *K* groups, each group containing *N/K* samples. These groups are labeled as *G1, G2, G3..., Gk*. Algorithm 2 shows the procedure of this *k*-fold cross-validation scheme.

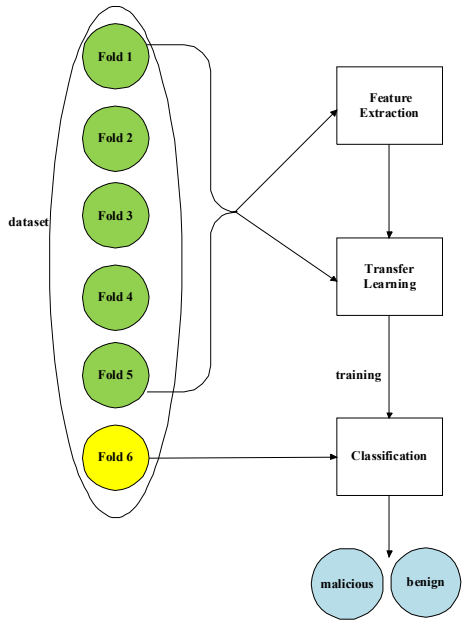
Algorithm 2: The k-fold algorithm.

Input : Sample set *Gi*;

Output : Class;

1. Extracting feature for each $Gi(i = 1 \text{ to } k)$;
2. **For** $j = 1$ to k
3. Training the classifier on train set $Gtrain = \{Gi \mid i \neq j\}$;
4. Validating the classifier on test set $Gtest = Gj$;
5. **Endfor**
6. **Return** Class.

The overview of our system using the 6-fold cross-validation scheme is given in Figure 4.



(Figure 4) The flow chart of 6-fold cross-validation.

4.3 Evaluation

In transfer learning, the feature spaces between source and target domain are different. Hence, we can estimate our experiment results over test data set, which is independent of the training data set by using 6-fold cross validation scheme. To evaluate proposed method, we focus on the following parameters:

1. True Positives (TP): the number of malicious executable that has been correctly classified.
2. True Negatives (TN): the number of benign programs that has been correctly classified.
3. False Positives (FP): the number of benign programs that has been misclassified.
4. False Negatives (FN): the number of malicious executables classified as benign programs.

The precision is the percentage of actually positive samples in all detected positive samples. It can be defined as:

$$precision = \frac{TP}{TP + FP} \tag{10}$$

The recall rate is how many of the positive samples in total samples are exactly detected as positive samples. The solution formula of recall rate is as follows:

$$recall = \frac{TP}{TP + FN} \tag{11}$$

F1-score comprehensively considers the precision and the recall rate. It defines as:

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{12}$$

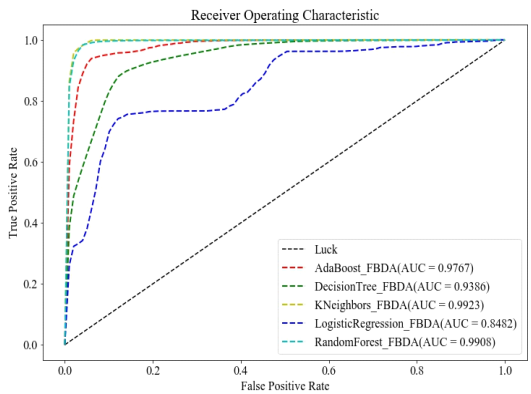
In this paper, five different popular methods are combined with FBDA respectively, and the performance comparison of their classifiers included precision, recall and F1, as shown in Table 2.

From Table 2, it is not difficult to find that the KNN method with FBDA is better than other combination methods. The precision of proposed KNN_FBDA is more than 98%, while the Logistic Regression with FBDA method is only about 80%. The recall rate of the KNN_FBDA method is also the highest, which is greater than 98%, while the Decision Tree_FBDA method is less than 96%. And the F1-score of the KNN_FBDA is also the highest, which is greater than 98%, while Logistic Regression_FBDA is only about 87%.

(Table 2) Classifier Performance.

Classifier Type	Precision	Recall	F1-score
AdaBoost_FBDA	94.63%	96.42%	95.50%
DecisionTree_FBDA	92.72%	95.44%	94.02%
KNN_FBDA	98.54%	98.84%	98.68%
LogisticRegression_FBDA	80.17%	96.52%	87.58%
RandomForest_FBDA	98.17%	98.34%	98.26%

Figure 5 displays the ROC curve of these several comparison methods. The ROC value of KNN _FBDA method is greater than 99%, which is also better than other several methods.



(Figure 5) The ROC comparison of several different methods with FBDA method.

There are two main reasons why the proposed KNN _FBDA approach is better than other comparison methods. The first is: because the tranditional KNN method needs to calculate the distance to all known samples for each text to be classified, it has high computational complexity and high overhead. However, in this paper, the proposed KNN_FBDA approach can reduce the redundant operations by extracting the main features, thus reducing the computational complexity and improving the classification efficiency and making up for the shortcomings of the traditional KNN method.

The second is, because KNN method mainly relies on the finite adjacent samples rather than the class domain discrimination method to determine the categories, its classification performance, such as precision, is better than

other method when the sample sets are divided into more overlapping or overlapping class domains.

5. Conclusions

This paper proposes a new KNN_FBDA approach, whichidentifies malicious executables by using a feature-based transfer learning approach. In the feature extraction process, IG and PCA method is first used for feature dimension reduction, then KNN is used for classification to enhance the recognition accuracy and real-time performance of malicious executable. Our experiments are all carried out on the real malicious PE files dataset. The KNN_FBDA is compared with other several comparison approaches, and it is superior to other combined methods in precision, recall rate and F1-score and so on. In future, we will consider applying transfer learning on other virus detection problems such as network virus, network attack. Moreover, the specific classification of computer viruses will be a research area that we will focus on.

References

[1] Lee Sang-Hun, Kim Won, Do Kyoung-Hwa, Jun Moon-Seog, "WAVScanner: Design and Implement of Web Based Anti-Virus Scanner", Journal of Internet Computing and Services, Vol. 5, No. 3, pp. 11-24, 2004.
https://www.koreascience.or.kr/article/JAKO200414714103092.page

[2] K. K. Vasan, B. Surendiran, "Dimensionality Reduction Using Principal Component Analysis for Network Intrusion Detection", Perspectives on Science, vol. 8, pp. 510–512, 2016.
https://doi.org/10.1016/j.pisc.2016.05.010

[3] Cho San, Mie Su Thwin, "Proposed Effective Feature Extraction and Selection for Malicious Software Classification", Advances in Biometrics, pp.51-71,2019.
https://doi.org/10.1007/978-3-030-30436-2_3

[4] Oh-Ryun Kwon, Kyong-Pil Min, Jun-Chul Chun, "Real-Time Face Recognition Based on Subspace and LVQ Classifier",Journal of Internet Computing and Services, Vol. 8, No. 3, pp. 19-32, Jun. 2007.

[5] Zhao, Juan, Sachin Shetty,Jan Wei Pan, "Feature-based transfer learning for network security", MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM), pp.17-22, 2017.
https://doi.org/10.1109/MILCOM.2017.8170749

[5] Umarani, S, D. Sharmila, "Predicting application layer DDoS attacks using machine learning algorithms", International Journal of Computer and Systems Engineering, Vol. 8, No. 10, pp. 1912-1917, 2015.
https://doi.org/10.5281/zenodo.1099004

[6] Nguyen Hoai-Vu, Yongsun Choi, "Proactive detection of DDoS attacks utilizingk-NN classifier in an anti-DDoS framework", International Journal of Electrical, Computer, and Systems Engineering, Vol. 4, No. 3, pp. 537-542, 2010.
https://pdfs.semanticscholar.org/38fe/3f1f9a7913a561a2878b8498f91b1550ab87.pdf

[7] Henchiri Olivier, Nathalie Japkowicz, "A feature selection and evaluation scheme for computer virus detection", Sixth International Conference on Data Mining (ICDM'06), IEEE, pp. 891-895, 2006.
https://doi.org/10.1109/ICDM.2006.4

[8] Schultz M G, Eskin E, Zadok F, "Data mining methods for detection of new malicious executables", Proceedings 2001 IEEE Symposium on Security and Privacy, S&P 2001, pp. 38-49, 2001.
https://doi.org/10.1109/SECPRI.2001.924286

[9] Oliveira Angelo, Renato José Sassi, "Behavioral Malware Detection Using Deep Graph Convolutional Neural Networks", 2019.
https://scholar.google.com.hk/scholar?hl=zh-CN&as_sdt=0%2C5&q=Behavioral+Malware+Detection+Using+Deep+Graph+Convolutional+Neural+Networks&btnG

[10] Jeongwhan Choi, "Iceberg-Ship Classification in SAR Images Using Convolutional Neural Network with Transfer Learning", Journal of Internet Computing and Services, Vol. 19, No. 4, pp. 35-44, 2018.
http://www.jics.or.kr/digital-library/15357

[11] Matt, Pietrek, "Peering inside the PE: a tour of the Win32 portable executable file format", MSDN Library, 1994.
https://www.cnblogs.com/antoniozhou/archive/2008/10/22/1317274.html

◐ 저 자 소 개 ◑

Yue Zhang

B.S. in Computer Science and Technology, Chongqing University, Chongqing, P.R.China in 2004.
M.S. in Computer Technology, Huazhong University of Science and Technology, Wuhan, P.R.China in 2007.
Lecturer at School of Information Science and Technology, Jiujiang University, Jiujiang, P.R.China in 2004.07~2018.08.
Currently, pursuing Ph.D. in Kunsan National University, Kunsan, Republic of Korea, since 2018.09.
Research Interests: pervasive computing, big data, transfer learning, network forensics.
E-mail: zhangyue@kunsan.ac.kr

Hyun-HoYang

B.S. and MS in Electronic Engineering, Kwangwoon University, Republic of Korea in 1986, 1990 respectively.
Ph. D in Information and Communication Engineering, Gwangju Institute of Science and Technology(GIST), Gwangju, Republic of Korea in 2003.
Currently, professor at School of Information and Communication Engineering, Kunsan National University, since 2005.
Research Interests: Wireless sensor network, pervasive computing, big data, transfer learning.
E-mail: hhyang@kunsan.ac.kr

Ning Gao

B.S. in Computer Science and Technology, Central China Normal University, Wuhan, P.R.China in 2004.
M.S. in Computer Technology, Dalian University of Technology, Dalian, P.R.China in 2010.
Currently, lecturer at School of Information Science and Technology, Jiujiang University, Jiujiang, P.R.China since 2004.07
Research Interests: big data, network forensics.
E-mail: 3090026@jju.edu.cn