# Static Detection of Malware and Benign Executable Using Machine Learning Algorithm

Dong-Hee Kim*, Sang-Uk Woo*, Dong-Kyu Lee* and Tai-Myoung Chung†
*Dept of Electrical and Computer Engineering
Sungkyunkwan University, Suwon, Korea
Email: {kkim, suwoo, leedg84}@imtl.skku.ac.kr
†College of Software
Sungkyunkwan University, Suwon, Korea
Email: tmchung@skku.edu

***Abstract*—One of the popular ways of detecting malware is signature based pattern matching. However, the signature of malware should be stored in advance for the pattern matching detection. Moreover, it calculates the similarity of input data using stored signature. Therefore, the storage problem and calculation overheads occur undoubtedly. Also, detection possibility is dropped, when malicious code is modified. So we use machine learning algorithm technique for detecting malicious executable and benign executable. However, previous technique has a limitation on detecting Worms and Trojans. In this paper, distinguished features of Portable Executable header are used. For the machine learning algorithm, Classification And Regression Tree (CART), Support Vector Classification (SVC), and Stochastic Gradient Descent (SGD) are applied for improving to detection rate. The performance of each algorithm firstly evaluated to find the most outperformed algorithm each for classifying benign executable and malicious executable. And then, these algorithms were combined to detect malware more precisely.**

***Keywords–Portable Executable Header; Machine Learning; Malware Detection; Intrusion Detection System.***

## I. INTRODUCTION

Traditionally signature-based static method is mostly used for malware detection. Signature-based method has some drawbacks. Pattern matching method, one of the signature-based static method, should possess all the pattern information of malware samples before the detection. Saving all the pattern informations, may causes the storage management problem and matching overheads. Moreover, detection efficiency of pattern matching method decreases, if pattern is changed by source code modification (e.g., inserting or removing the opcode). Therefore, machine learning-based malware detection methods are being researched [1][2][3][4]. The purpose of using machine learning algorithm is to study the pattern from the learning set and to predict the classes or value from the given data [5]. The acceptable detection rate is described in several previous researches. The various features of benign code and malicious code had been considered from many research paper. Researchers have derived the distinctive characteristics which are from binary code [6][7], opcode [8][9], and Portable Executable header (PE-header) of benign executables and malicious executables [10]. They have evaluated their result using a variety of machine learning algorithms. The advantage of using a machine learning technique is the prediction of unknown class. It can detect not only known malware but also non-recognized malware through the pattern analysis itself. In addition, machine learning algorithm can detect a large amount of malware using relatively small amount of input training sets.

The interested detection method is PE-miner framework [10]. The PE format is a file format for executables, object code, DLLs, Font files, and others used in 32-bit and 64-bit versions of Windows operating systems [11]. In shafiq et al. paper [10], they have analyzed the distinctive characteristics of PE-header between malicious executable and benign one. They categorized malicious executable into 7 types; backdoor + sniffer, Constructor + Virtool, DoS + Nuker, Flooder, Exploit + Hacktool, Work, Trojan and Virus. From the PE-header, 18 different features are founded by Shafiq. However, PE-header features might not convey useful information in a particular scenario. For example, some attribute value could have too much low value or dummy value, and some could be counter. Also, considering the application of the many attributes increases the dimensional spaces in machine learning algorithm. This is the main reason for time delay in fitting process. So, for reducing dimensionality of input feature space, a preprocessor process is removing or combining the PE-header information with other similar features. Redundant Feature Removal (RFR), Principal Component Analysis (PCA), and Haar Wavelet Transform (HWT) mechanisms are used for preprocessing the PE-header feature.

The purpose of this paper is to evaluate the existing PE-miner framework [10] and improving the detection rate by adjusting the attribute of training set and algorithm. We have chosen the PE feature from many other distinctive characteristics because it has an almost fixed size of data structure regardless of program size. If the number of attributes composing the training set is changed depending on data, it will increase the complexity of training process. We expect that the attributes that extracted from previous research could not carry the characteristic of the malware according to the Windows system changes. Also, in previous research [10], Shafiq et al. use insufficient amount of training set and sample file. For their experiment, 1,477 benign sample files and 15,925 malware sample files were used. The most relevant information is stored with the highest coefficients at each order of a transform. The lower order coefficients can be ignored to get only the most relevant information. Decision Tree (J48), Instance Based Learner (IBk), Native Bayes (NB), RIPPER (inductive rule learner), Support Vector Machine using Sequential Minimal Optimization (SMO) algorithms are used for their experiment. The outputs of these algorithms were compared with each other

 ISBN: 978-1-61208-516-6 

and the best performance was evaluated when using the J48 that is achieves more than 99% detection rate with less than 0.5% false alarm rate. However, the most challenging malware categories for detecting are Worms and Trojans. Trojans are inherently designed to appear similar to the benign executables. So, in this paper, Classification And Regression Tree (CART), Support Vector Classification (SVC), and Stochastic Gradient Descent (SGD) are used to classify the worms and trojans. These algorithms are specialized in classification. In addition, the most challenging malware categories for detecting are worms and trojans. Trojans are inherently designed to appear similar to the benign executables [10].

This paper is organized as follows: In Section 2, we denote the source of collected sample and the explanation of training data composition. Section 3 describes methodology of single algorithm based classification process and a simple characteristic about the used algorithm. Section 4 represents the result of algorithm performance. Section 5 suggests the improvement of reducing the error rate. Finally, Section 6 finishes up with a conclusion.

## II. Sample collection and Training set

This section specifies the source of samples and evaluation of PE header features. Also, composition method of training sets is explained. Benign executable files are collected from Windows operating system and Malicious executables are downloaded from internet. PE-header features are extracted using python module. The training set is made in the form of a csv file with system independence.

### A. Sample collection

We collect the 9,773 executable sample files from system 32 folder in Windows 7 and collect 18 files in Ubuntu Linux kernel. The files in system32 folder are extracted immediately right after OS installation with series of updates as long as it is easy to be forged or tampered by malware. Malicious executable sample files are downloaded from the VXheaven website [12]. The total number of malware sample is 271,095 but the 236,707 samples which contain the PE-header are only used to making training sets. The “pefile” which is one of the python module was selected to measure the presence of the PE-header and extracting the PE-header information from the file [13]. It supports various operating system environments like Windows, Linux, and Mac OS. The module extracts a file header data and returns the class instance.

### B. Training data

PE-header of benign code and malicious code are evaluated using 5,000 samples each. The result is shown in Table 1. Comparing with previous research [10][14], the network related dll file is unsuitable for training attribute. The network related dll file is not only frequently used in malware but also used benign executable files since many legitimate software use network resources. As referring to previous study [10], the value of Number of symbols, Major linker version, Initialize data size, Major image version, and Dll character shows distinctive feature between benign and malicious code. The similar result was evaluated from our test. Referring to Table 1, the average of COFF characteristic value shows high gap between benign and malware. The characteristic value in COFF file header

TABLE I. MEAN AVERAGE VALUES OF PE-HEADER FEATURES

| Name of Feature | Benign | Malware |
|---|---|---|
| characteristic in COFF File Header | 7232.26 | 13369.88 |
| # Symbols | 0.21 | 60.5 x $10^6$ |
| Maj Linker Ver | 8.87 | 7.29 |
| Init Data Size | 21.1 x $10^4$ | 61.8 x $10^6$ |
| Maj Img Ver | 107.31 | 31.86 |
| Dll Char | 4274.99 | 545.34 |

represents summary of image that calculated in sum of characteristic field value [15]. For the average value of Characteristic in benign executable is 7232.26 and for malicious is 13369. Comparing to average value of Number of Symbols, malicious sample shows 29 x $10^7$ greater than benign. The greater the value, meaning the more system options are used. Benign file has the value of 8,000 around and some of them are 100 under. But in malicious sample, most of the them shows 10,000 and only few samples are 100 under. However, the average value of Number of Symbols tend to represent distinguished feature in previous system (e.g, Windows XP), but it does not show the clear differences between the benign and malicious sample because, most of benign and malicious executables have value of 0, but few of malicious file has extremely large value to increase the average value [15]. Moreover, Major Linker Version value does not show great gap but the value maintains constant value in both benign and malicious. It expects that both benign and malware use similar version of linker. Matter of fact, this field was a very distinctive feature in previous research result [10]. But now, it is featureless value that only increases the dimensional spaces. So, we decided to get rid off a Number of Symbols field and Major Linker Version field from the training sets. Other fields, Initialized Data Size, Major Image Version, and Dll Characteristic, are still showing their own feature. Malicious Initialize Data Size value is 292 times greater than the benign executable. Major Image Version of benign executable is approximately three times greater than malicious. Also, Dll Characteristic value of benign program is about 4 times greater than malware. Finally, we have made training sets with 4 attributes which are Charateristic in COFF File Header, Init Data Size, Maj Img Ver, and Dll Characteristic.

Training data including attribute and target value that represents the benign or malicious executable were created and saved as a CSV file type. We prepared the 10 sets of training data with different amount of samples. We divided the samples into 10 blocks. One block for benign sample contains 950 files and for malware sample contains 23,000 files. And the n sets composed with n blocks of benign sample and n blocks of malware. For example, composing third set, 3 blocks of benign samples and 3 blocks of malware samples are needed. Thus, 2,850 benign files and 69,000 malware files are used for composing the training data. To get precise result, test is proceed 10 times with different combination of training data.

## III. Algorithm Performance Evaluation

Two experiments were performed. First experiment is to find the best algorithm for each benign and malware. From this experiment, we have found that some algorithms are

 ISBN: 978-1-61208-516-6 

outperformed for predicting the benign files and some are outperformed in malware. Therefore in second experiment, we combined the two best algorithm to evaluate the prediction performance.

### A. Methodology

The methodology of the first experiment, single machine learning classification method is divided into three parts as in Fig 1. First, in training process (tiny dash line), the machine learning algorithms (CART, SVC, and SGD) are trained using the training data which was explained in the previous section. To check the detection efficiency depending on the amount of sample that used for training, training data is prepared with 10 different sets as mentioned in previous section. Each algorithm generates classifier when training data is assigned. In Second, input file filtering process (dash line) is conducted. The "pefile" module checks the existence of PE-header or the architectural maintenance from the input executables. It has a purpose to maintain service availability. If wrong PE format file is conducted to classifiers, it ceases the input file and call the next file. The total number of input file is 246,497. Input file for benign is 9,790 and malicious is 236,707. Input file contains not only trained samples but also unrecognized samples. Finally, in classify process (dotted line), machine learning classifier classifies the input files. Classification results are written to a csv file with the original target value. But in wild, the classifier can predict the result right away without reporting them. The experimental environment for classification of files is as follow: CPU with i5-3.90 GHz and 16GB ram and the operating system is Ubuntu desktop.

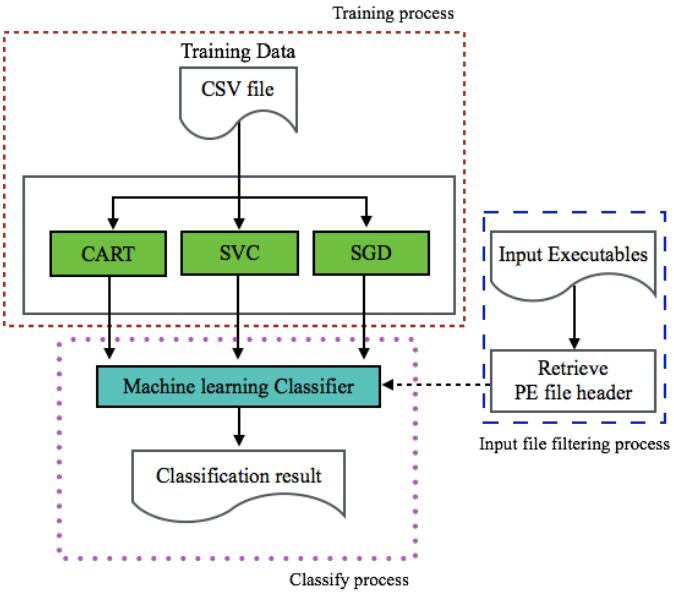


Figure 1. Single algorithm based classification methodology

### B. Algorithm Explanation

In this section, a brief description of each algorithm and the options that we applied to this experiment is described. In this research, the scikit-learn Python module is used for the classification of data. Scikit-learn is one of the most widely used machine learning module in Python [16].

*1) Classification And Regression Tree:* CART is one of the decision tree algorithm. A decision tree is a rooted tree with internal nodes corresponding to attributes and leaf nodes corresponding to class labels. CART is similar to C4.5, but it not only supports discrete target value but also numerical target value and does not compute rule sets. CART constructs binary trees using the feature and threshold that yields the largest information gain at each node [16]. Fig. 2 is the partial example of our CART model. The CART algorithm is structured as a sequence of questions where in the next question is determined depending on the answers. Algorithm is designed to keep continue questioning until the end of the node. The end of the node is the prediction result of the target value. When training data comes, the algorithm starts with tree growing process. The basic idea of tree growing is to choose a split among all the possible splits at each node so that the resulting child nodes are the purest. The next step is splitting criteria and measuring impurity. If the impurity measurement occurs, the splitting criterion corresponds to a decrease in impurity. The tree is not continuously growing either by customer options or algorithm design itself. If a node becomes pure or node has the identical value, it stops growing. For our CART model, we use Gini impurity criterion for growing tree. Limitation of maximum feature, depth, and the number of leaf nodes are not set. Therefore, the tree used all the training data attributes and grows until the stopping rule initiated.

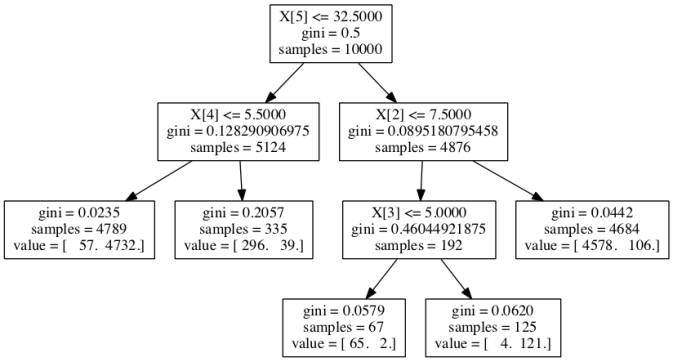


Figure 2. CART algorithm sample

*2) Support Vector Classification:* SVM is supervised learning models that analyze data used for classification and regression analysis. SVC (Support Vector Classification) is one of the SVM method for specializes in classification and is effective in high dimensional spaces. Calculating the best fitted decision function is important. If the subset of training point in decision functions are well-defined, then memory is efficient. SVC has various kernel functions and it is important to select suitable kernel functions for improving the pattern recognition ratio [17]. Customized kernel can be designed depending on its purpose. Thus in case insufficient kernels exist, then user create his own kernel. For our SVC model, we select rbf (Radial-Basis Function) kernel mode. Rbf kernel handles the set weights for finding a curve fitting problem. Rbf kernel has the advantages when the weights are in higher dimensional space than the original data. Training is equivalent to finding a surface in high dimensional space that provides the best fit to training data. We set degree value as 3 and gamma for 0.167. Gamma value calculated with formula that 1/number of

 ISBN: 978-1-61208-516-6 

features.

*3) Stochastic Gradient Descent:* SGD algorithm is a stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions. SGD has been researched in the past, but recently it has been proven that SGD shows high classification ratio when $10^5$ training samples and $10^5$ features are trained [16]. Therefore, this algorithm is often used to classify the natural languages and recognition of characters. SGD has plenty of parameters (loss regularization, alpha, shuffle, verbose etc.) to elaborately control the decision point. In this paper, we select the loss regularization for perceptron which is a source of neural network. Perceptron is a basic processing element. It has inputs that may come from the environment or may be driven by other perceptrons [18]. Perceptron is a type of linear classifier. It predicts based on a linear predictor function combining a set of weights with the feature vector. Curved model is already adopted in SVC, thus we tried to use linear model of decision point. The alpha value is set to 0.0001 and regularization set to l2 as a normal.

## IV. ALGORITHM PERFORMANCE RESULT

In this section, the algorithm performance is evaluated in two cases. One is false-negative and the other is false-positive. Fig. 3 represents the false-negative rate of each algorithm. False-negative refers to the error when a benign application is classified as malicious. 23,950 samples (950 for benign and 23,000 for malware) trained CART classifier shows about 2.58% error. The false-negative rate continually decreases as number of trained samples increases. When 215,550 sample which is 90% of total sample was trained, it showed 0.2% of error rate which is the lowest. In this case, CART classifier incorrectly predicted 20 files from overall 9,790. This classifier outperformed 13 times in prediction comparing to 23,950 sample trained classifier. On the other hand, SVC algorithm performs 40.36% false-negative rate when 23,950 samples are adapted. The error rate of SVC also keeps decreasing as training sample are increasing. But still it shows high error rate compare to CART algorithm. For SGD, it shows 80% of error value, but it drops most significantly among the three algorithms. Nevertheless, SVC and SGD show high error rate comparing to CART algorithm. CART algorithm is outperformed approximately 14 times than SVC and is 60 times more efficient than SGD algorithm.

The false-positive rate of each algorithm is shown in Fig. 4. False-positive is when the malicious is predicted as a benign. CART error rate is decreasing steadily by increasing the number of training sample. The highest error rate is shown to be 0.086% when the trained sample is 23,950, and the lowest error rate is 0.0034% when the 215,550 training samples used. Just 8 files were misclassified among the 236,707 malware samples. The false-positive rate of 215,550 sample trained CART classifier is improved about 25 times comparing to the 23,950 sample trained CART classifier. On the other hand, even from the beginning, the SVC algorithm shows error rate of 0.0097%. Only 23 files were misclassified among 236,707 malware samples. As the number of training samples increased, only 2 files were misclassified from the overall malware samples. For SGD algorithm, the lowest error rate is 0.8154%. The value seems to be acceptable enough, but compared to other algorithm, this value is 1,020 times higher than SVC algorithm.

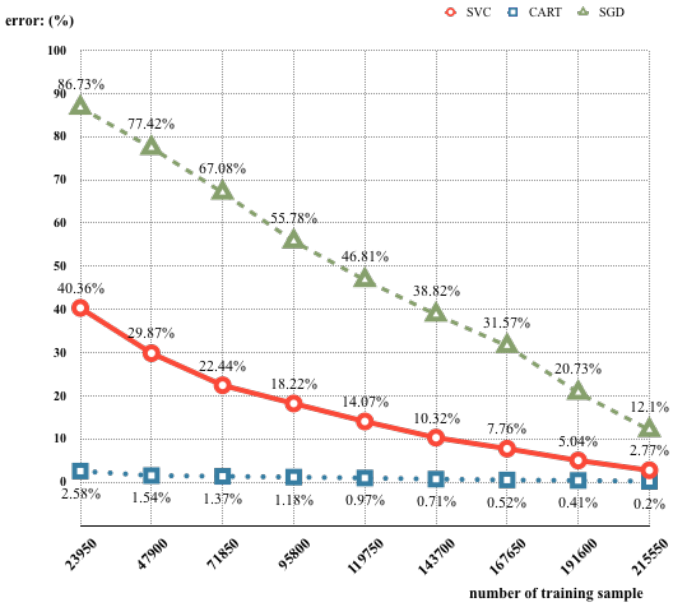


Figure 3. False-negative rate

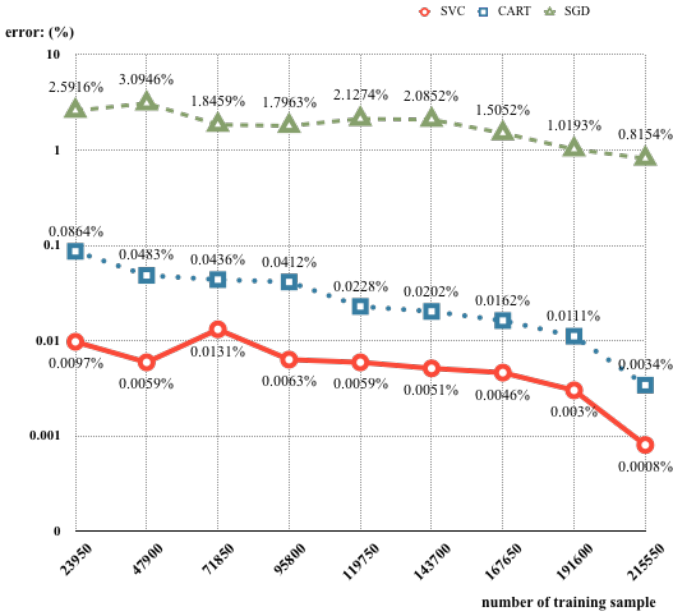


Figure 4. False-positive rate

Both false-negative and false-positive rate of CART shows prediction accuracy over the 99%. Especially when 90% of samples are trained, the false-negative prediction accuracy is 99.8% and the false-positive prediction accuracy is 99.99%. Result of SVC false-negative rate is notable. It presents 97.23% of prediction accuracy. However, CART is more appropriate for predicting the benign executable. Nevertheless SVC algorithm is more efficient when detecting the malicious executable. The accuracy of SVC for predicting the malicious executable represents 99.9992%. CART error rate is 0.0034%.

 ISBN: 978-1-61208-516-6

This seems to be little difference in error capacity, but if even a single malicious code passed into system harms all. Therefore, the malware detector should lessen the error rate. Also, SVC can show efficient prediction accuracy even though small amount of sample are trained.

From this experiment, the number of samples are the same, but the test results are done repeatedly by applying a different training data 10 times to machine learning algorithms. We have noticed that both CART and SGD case, types of trained sample and the number of training data both are affected. However, in the case of the SVC, the result has a constant value, regardless of the type of data but it is influenced by number of training data. Because CART considered all the training sample data to make best result of information gain. But, SVC algorithm defines the hypothesis space according to kernel function. So, the sample distribution that scattered in hypothesis space does not change significantly.

## V. Improvement of Detection Efficiency

This section proposes the improved methodology combining the two algorithms. When using the combination of CART algorithm which is excellent for detecting benign executable and SVC algorithm which well detects the malicious executable, we expect to determine the unknown executable better. For the last part of this section, the combined algorithm efficiency is evaluated.

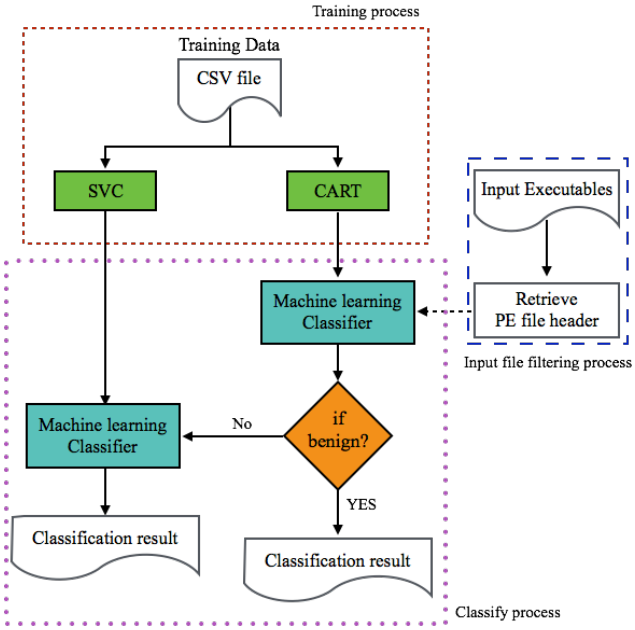


Figure 5. Two algorithm combined classification methodology

### A. Methdology

The input executable files are always in unknown state whether it is benign or malicious. The combination of the two algorithms are adapted for detecting the unknown state of file. The classifier assume the predicted result of CART is trustworthy only for benign case. If CART returns the prediction result that pointing malicious, then it should toss to SVC for re-inspection. As in Fig. 5, procedure is also divided into 3 part as mention in Section 3. First of all in training process, CART and SVC make a classifier using same training data. Secondly, input file filtering process exceed. They filter the non-proper PE-header or PE-header non-existence files. Finally, in the classify process, CART algorithm predicts whether the input executables are benign or malicioys. If CART classifies the input executable as benign, it believes the result and pass them. But if, CART predicts the input file as malicious executable, it sent to SVC algorithm for re-inspection. It takes time to check one file again. But time requirement of inspection took 0.01 seconds. It is not a big loss as it guarantees the security.

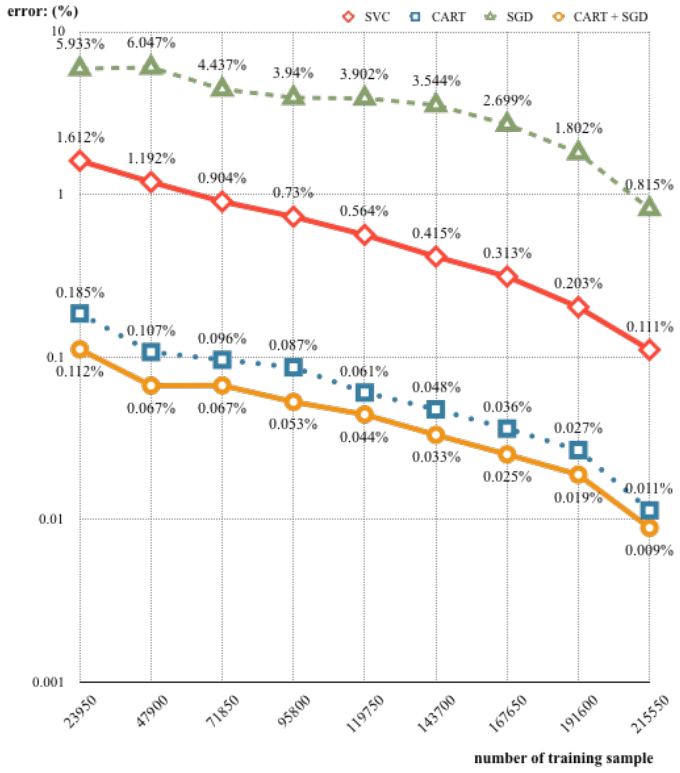


Figure 6. Total error of three algorithms and combined algorithm

### B. Experimental result and Discussion

The misclassification error rate is represented in Fig. 6. For the first, SGD algorithm shows about 6% of error in the beginning, but when 215,550 samples are trained, it represents 99% of prediction accuracy. SGD perceptron algorithm shows high classification ratio, if more than $10^5$ training samples and $10^5$ features are trained. However in this experiment, only 4 features are applied when making a training set, and because of limitation of samples, the training is insufficiently conducted. So, it displays relatively high error than others, but if enough samples are trained, it will perform better.

SVC algorithm begins with 1.6% error because the performance of classifying a benign code dropped significantly. However, after learning the 71,850 sample data, the detection rate represents value of 99%, and eventually only 0.111% are misclassified. In particular, the SVC is specialized in detecting malicious code. The improvement of capability of classifying the benign code can exhibit better performance than CART.

CART algorithm has high detection accuracy in both benign and malware, so it has an accuracy rate of 99% or more

 ISBN: 978-1-61208-516-6 

from the beginning. It shows a 0.011% error when 215,550 of training samples are used. CART algorithm has an advantage in single uses from restricted condition as malware detection performance is degraded than the SVC algorithm. However, if the configuration of a robust system is desired, it is possible to reduce the false positives through the combination of CART and SVC. From Fig. 6, The combined algorithm presents the error of 1.6 times better performance than 0.112% of the initial value of CART. By continuing the training the algorithm, it sharply reduces the error rate. This error rate of 0.0009% (about 12 times better than the first time) is shown when 215,550 sample are trained. Only 22 samples are fault detected from the total 246,497 samples.

## VI. Conclusion

We have analyzed current characteristics of PE-header. The result shows that the Characteristic in COFF header has a prominent features and the network related dll does not face distinguished characteristics between benign program and malware program. Also, Number of symbols and Major Linker Version are featureless for current Windows system.

The experimental result was obtained by using more than 270 thousand malicious samples and 9 thousand benign samples. When classifying the benign executable, the use of CART algorithm is worthy. This algorithm represents more than 99 percent of prediction accuracy with 0.2 percent of false-negative rate. SVC is suitable for detecting the malware. It properly predicts malware with 99.99 percent. However, CART is more efficient than SVC according to the total error. Based on the result of our evaluation, we notice that there is specialized algorithm for predicting the malicious executable or benign executable. Therefore the combination of two algorithms were proposed. The result of the proposed method shows the low error rate compared to single use of CART. In addition, the combined mechanism clearly demonstrates the efficiency of classification on malware, including Worm and Trojan. But, the use of two algorithms has a disadvantage for time and resource consuming. Even though it has some drawbacks, the proposed method is needed to provide a stable protection for the system. Now, we are interested in improving the efficiency of a single use of SVC algorithm. This will leave for the future works.

## Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2010-0020210)

## References

[1] C. Sinclair, L. Pierce, and S. Matzner, “An application of machine learning to network intrusion detection,” in Computer Security Applications Conference, 1999.(ACSAC’99) Proceedings. 15th Annual. IEEE, 1999, pp. 371–377.

[2] J. Bergeron et al., “Static detection of malicious code in executable programs,” Int. J. of Req. Eng, vol. 2001, no. 184-189, 2001, p. 79.

[3] C. Smutz and A. Stavrou, “Malicious pdf detection using metadata and structural features,” in Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012, pp. 239–248.

[4] D. Maiorca, G. Giacinto, and I. Corona, “A pattern recognition system for malicious pdf files detection,” in International Workshop on Machine Learning and Data Mining in Pattern Recognition. Springer, 2012, pp. 510–524.

[5] K. P. Murphy, Machine learning: a probabilistic perspective. MIT press, 2012.

[6] J. Z. Kolter and M. A. Maloof, “Learning to detect and classify malicious executables in the wild,” Journal of Machine Learning Research, vol. 7, no. Dec, 2006, pp. 2721–2744.

[7] B. Zhang, J. Yin, J. Hao, D. Zhang, and S. Wang, “Malicious codes detection based on ensemble learning,” in International Conference on Autonomic and Trusted Computing. Springer, 2007, pp. 468–477.

[8] I. Santos, F. Brezo, B. Sanz, C. Laorden, and P. G. Bringas, “Using opcode sequences in single-class learning to detect unknown malware,” IET information security, vol. 5, no. 4, 2011, pp. 220–227.

[9] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, “Opcode sequences as representation of executables for data-mining-based unknown malware detection,” Information Sciences, vol. 231, 2013, pp. 64–82.

[10] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, “Pe-miner: Mining structural information to detect malicious executables in realtime,” in International Workshop on Recent Advances in Intrusion Detection. Springer, 2009, pp. 121–141.

[11] C. Visual and B. Unit, “Microsoft portable executable and common object file format specification,” 1999.

[12] “Vxheaven,” http://vxheaven.org/vl.php, 2016, accessed November 2, 2016.

[13] E. Carrera, “erocarrera/pefile,” https://github.com/erocarrera/pefile, 2016, accessed November 2, 2016.

[14] M. Z. Shafiq, S. Tabish, and M. Farooq, “Pe-probe: leveraging packer detection and structural information to detect malicious portable executables,” in Proceedings of the Virus Bulletin Conference (VB), 2009, pp. 29–33.

[15] “image file header structure (windows),” https://msdn.microsoft.com/en-us/library/windows/desktop/ms680313(v=vs.85).aspx, 2016, accessed November 2, 2016.

[16] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, 2011, pp. 2825–2830.

[17] L.-P. Bi, H. Huang, Z.-Y. Zheng, and H.-T. Song, “New heuristic for determination gaussian kernels parameter,” in 2005 International Conference on Machine Learning and Cybernetics, vol. 7. IEEE, 2005, pp. 4299–4304.

[18] E. Alpaydin, Introduction to machine learning. MIT press, 2014.

 ISBN: 978-1-61208-516-6 