

Detecting Unknown Malicious Executables Using Portable Executable Headers

Tzu-Yen Wang

Dept. of Computer Sci.
National Chiao Tung Univ.
Hsinchu, Taiwan
tzu-yen.wang@hotmail.com

Chin-Hsiung Wu*

Dept. of Info. Tech. & Comm.
Shih Chien Univ. Kaohsiung Campus,
Kaohsiung, Taiwan
chwu@mail.kh.usc.edu.tw

Chu-Cheng Hsieh

Dept. of Computer Sci.
Univ. of California,
Los Angeles, USA
chucheng@ucla.edu

Abstract—Even though numerous kinds of anti-virus software packages have been used for many years, previously unseen malware is still a serious threat to computer and information system. By analyzing portable executable header entries of executables, a malware detection model which consists of four stages: attribute extraction, attribute binarization, attribute elimination, and feature selection and classifier training was carried out in this study. First, we collected header entries from all executables in our dataset and viewed each entry as a potential attribute. Second, information gain and gain ratio were used to binarize numerical and nominal attributes. Next, useless and redundant attributes were eliminated in the third stage. Finally, by using support vector machine which is a classification algorithm of conspicuous generalization ability, feature selection was simultaneously performed with classifier training to reduce the number of attributes and retain the performance of classifier in a cost-effective. We evaluated our model by 1,908 benign programs and 7,863 malicious files (virus, email worm, trojan and backdoor) and estimated its generalization ability by cross validation. The experiment results showed that our model had promising performance for detecting virus and email worm.

Keywords- *previously unseen malware; information gain; gain ratio; feature selection; support vector machine*

I. INTRODUCTION

Malware, which refers to codes performing malicious activities such as crashing computer systems, jamming network flows and stealing personal information, etc., has been a notorious threat to information security for nearly 30 years. Commercial anti-virus packages, which have been widely used for malware detection, seemed to be reliable to fight against malicious codes. In general, these commercial programs are based on signatures generated by both static analysis and dynamic analysis. Static analysis usually constructs signatures by segmenting the binary code sequences from known malware while dynamic analysis executes the malware in a safe environment and observes its own behaviors. By matching malware to those predefined patterns, anti-virus software could accurately detect known malicious codes without any false alarm. However, according to the report from AV-Comparatives.org¹ in May

2008, the proactive detection abilities of sixteen anti-virus software packages were examined by detecting new malicious code without updating their virus definitions. The examination results were utterly unsatisfying that none of them had proactive detection rate higher than 60%. Therefore, the main problem of current signature-based method is that it is ineffective against zero-day attacks launched by previously unseen malicious files or even polymorphic ones due to the lack of virus definitions of them. Every minute people wait for an update of virus definitions is another minute that their computer is vulnerable to any vicious damage. Hence it is no doubt that malware is remaining a noticeable security challenge and the needs of continuing improvements in unknown malware detection methods are crucial.

The goal of this research is to propose a novel model to detect unknown malware in Portable Executable (PE) [13] format since it is the most widely distributed type of malicious codes. To fulfill the purpose of preventing malware from crashing users' computers, we adopted static analysis method to exploit the information of PE headers entries. With the assistance of Support Vector Machine (SVM) [20], which is a classification algorithm of conspicuous generalization ability, these attributes could be used to construct an effective and reliable model to distinguish malicious executables from benign programs.

The rest of this paper is divided into five sections. The first section is devoted to some related studies in the area of unknown malware detection. After which the second section sketches the content of PE header entries. This is followed by the methodology and procedures of our malware detection model. The experiment results for detecting unknown malware are illustrated in the next section. Finally, some concluding remarks and future works are made in the final section.

II. RELATIVE WORKS

Owing to the lack of proactive detection ability of nowadays commercial virus scanners, considerable concern has arisen over the field of unknown malware detection. To detect previously unseen malware in advance, we only focused on the use of static analysis methods and data mining techniques in this study and gave some literature review which is related to our work in this section.

Schultz and Zadok [15] presented some kinds of features such as Dynamic Linked Libraries (DLLs), Application

*Corresponding author. This work was partially supported by the National Science Council under the contract no. NSC-97-2221-E-158-003.

¹ www.av-comparatives.org/

Programming Interfaces (APIs), strings, and byte sequences to portray the characteristics of an executable. Both Ripper and Naïve Bayes algorithms were used to construct classifiers to detect new malware. Similarly, Abou-Assaleh and his colleagues [1] employed common n-grams analysis to generate signatures from their benign and malicious software collections. K-nearest neighbor algorithm was used to classify new instance in their study. As Wang et al. noted in their study [21], they adopted information gain to extract valuable byte sequences and conducted experiments with Naïve Bayes and Decision Tree classifiers.

Kolter et al. [7] utilized a hex-dump utility to convert each executable into a hexadecimal code and also produced n-gram features by combining each four-byte sequence into a single term. Then they used information gain to select top 500 features to build various kinds of learning classifiers. In 2006, their works were further extended [8] to classify malicious codes based on their functions. Moreover, their model was tested with new malware that did not previously obtained in the training set. Similar works on adopting byte sequence patterns as features which were processed by seven different feature selection methods was carried out by Cai. et al. [2]. And the performances of four kinds of classifiers were evaluated in their study.

Based on the hypothesis that any malware should share some common signatures in its different versions, Sung et al. [16] used APIs calling sequences to represent the behaviors of binaries and proposed a vicious executables detection system to detect polymorphic malware and metamorphic malware. Different kinds of measurements were calculated to evaluate the resemblance between known virus and suspicious executable. Masud et al. [10] proposed a hybrid model which combined three kinds of features: binary n-gram, assembly n-gram, and APIs sequence. The practicability of each kind of features was tested by SVM respectively. In our previous works [22, 23], a SVM-based model was also proposed to detect malicious executables which were characterized by the usage of DLLs and APIs sequences.

As mentioned above, byte sequence analysis has been widely investigated since it elaborates on the content of an executable. However, it is neither intuitive for people to comprehend the properties of an executable nor steadfast against the injection of junk code which could easily affect the detection performance. Besides, while a number of studies have investigated the uses of filter-based feature selection method, wrapper-based feature selection method, which cooperates with classifier training process, were neglected in early works. For these reasons, PE header entries which directly exhibit characteristics of an executable were manipulated through static analysis to feature executables in our study. Furthermore, with the aid of wrapper-based feature selection method which associates with SVM, our malware detection model has achieved a promising performance by rather economical number of features.

III. PORTABLE EXECUTABLE FILE

The basic supposition of this study is that there should be some distinctive characteristics between malware and benign programs since they are built in different intention. Thus, it is a good reason to detect vicious binaries by the differences of natural specialties recorded in PE files. As illustrated in Fig. 1, PE file is a data structure that Microsoft has designed to encapsulate the information for Win32-based systems to manage the wrapped executable code. It consists of a number of headers and sections, but here is just a cursory glance at which were used in our study. Materials describe PE file format in details can be found in [11, 13]. First of all, the file header of a PE file reserves some general information such as the target machine on which the PE file can run, the number of sections in the PE file, and so forth. Next, the entries of optional header can be divided in two parts. First part contains general information that is useful for operating system to load and run an executable. Moreover, data directory is the second part in which each entry indicates the address and size of a section. Note that the number of directories is not fixed. Finally, the data directory is followed by section table which is a collection of section headers. The fields of section headers summarize messages of each section.

Although all PE header entries stated above seem trifling, they indeed retain some essential knowledge which depicts the uniqueness of each executable. In next section, a four-stage malware detection model is presented to prove that these header entries can be successfully utilized to classify malware.

IV. MALWARE DETECTION MODEL

As shown in Fig. 2, a malware detection model consists of four stages, which include attribute extraction, attribute binarization, attribute elimination, and feature selection and classifier training, was carried out in this study. A description of our experiment dataset is first given and each stage is described at length in the following subsections.

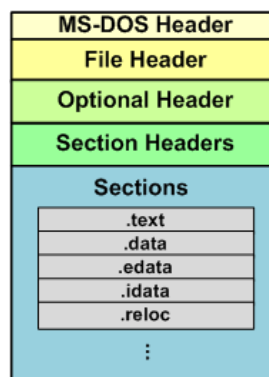


Figure 1. Portable executable file format

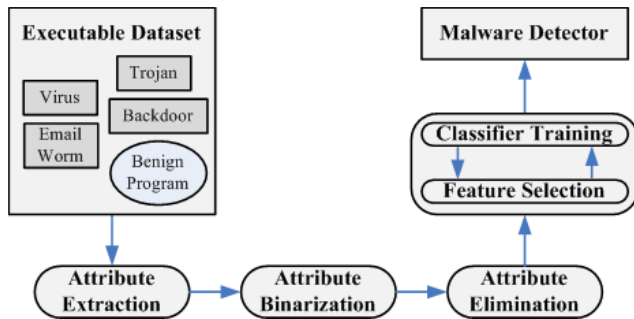


Figure 2. Malware detection model

A. Data collection

Due to the lack of a public dataset to evaluate our model, we must collect both benign and malicious executables by ourselves. Our dataset was made up of 1908 benign programs and 7863 malware (913 viruses, 808 email worms, 2809 trojans and 3333 backdoors). All of them were PE-formatted executable files. Those malicious binaries were acquired from VX Heaven² and each of them was identified by Kaspersky anti-virus software³. Note that we examined only the loader programs for viruses; we did not include infected executables in our study. Besides, the benign programs were obtained from pure installations of Windows 2000 and XP operating system plus several common user applications acquired from the web site PChome⁴. Similarly, these benign programs were also verified by the same anti-virus software to guarantee that they were not composed of any malicious codes.

B. Attribution extraction

To characterize executables, we first utilized a tool called DUMPBIN which is provided by Microsoft to dump the entries of PE headers of each file. Viewing every header entry as a potential attribute, we calculated and visualized the number of appearance of every attribute by category and conducted a total census for further analyzing. An example is given in Table 1 and Fig. 3.

TABLE 1. CENSUS OF HEADER ATTRIBUTES

Entry Name	Entry Values	# of appearance				
		Benign	Virus	Email Worm	Trojan	Backdoor
# of sections	1	1	1	0	26	7
	2	23	12	18	36	19
	3	652	180	428	1186	1377
	4	517	482	153	306	290
	...	...	...	...	...	...
	18	1	0	0	0	0
	1D	1	0	0	0	0
.reloc section	True	521	647	230	1060	1429
	False	1387	266	578	1746	1904

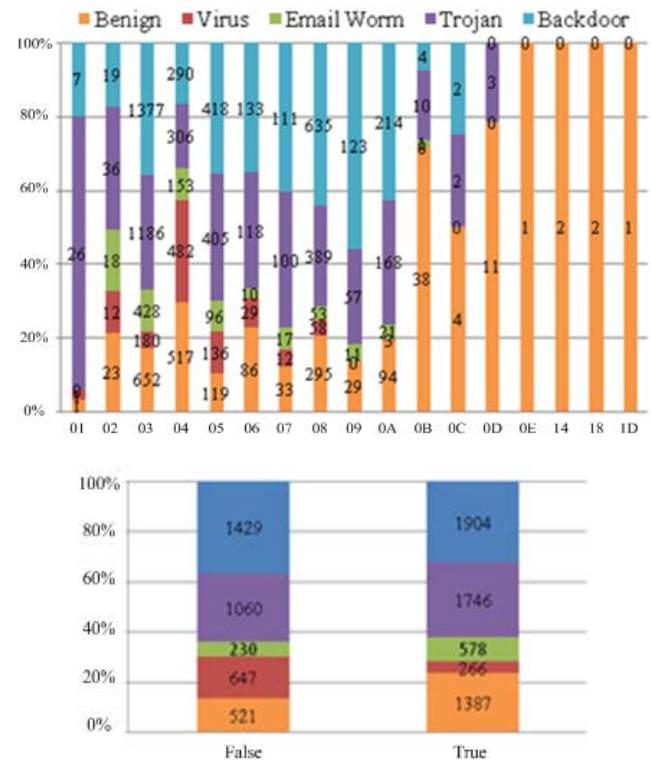


Figure 3. Visualization of header attributes. Top: The number of section. Bottom: Existence of reloc section.

C. Attribute binarization

Although SVM is able to process data of rational attribute values, we converted every numerical or nominal PE header field into a binary attribute for the sake of simplifying the training process. For the case of numerical attribute, after sorting the values of an attribute, we bisected the values so that a split point could be chosen if its resulting two intervals are purely distributed over the classes. Similarly, nominal attributes can also be binarized by repeatedly grouping all values except the one of interest together as a negation of this value.

To measure the purity of intervals, two entropy-based methods- information gain and gain ratio were considered as discretization strategies. Information gain (IG), which computes the discrimination of an attribute, was proposed by Quinlan [14] in 1986. Let X be a finite set of examples, if there are m classes $\{C_1, C_2, \dots, C_m\}$ in X , the entropy of X is defined in (1), where P_i is the proportion of class i . $H(X)$ measures the randomness of the distribution of examples in X over m possible classes. Suppose that $Y = \{Y^1, Y^2, \dots, Y^p\}$ is a set of attributes and each attribute Y^p has k values $\{V_1^p, V_2^p, \dots, V_k^p\}$, (2) implies the entropy of X conditioned by the values of Y^p . Then IG is given by (3) where $|X_j^p|$ is the number of examples in X having attribute Y^p with value V_j^p . The amount of (3) reflects additional information about X provided by Y^p . The higher the IG is, the purer of the distribution of examples in X over m possible classes we have. We modified (3) into (4) to help us to decide the splitting value V_s^p of Y^p which makes highest IG for a

² vx.netlux.org

³ www.kaspersky.com

⁴ toget.pchome.com.tw

binarized attribute. In the case of numerical (nominal) attribute binarization, V_j^p equals to zero if $V_j^p \leq V_s^p$ ($V_j^p \neq V_s^p$) and it equals to one if $V_j^p > V_s^p$ ($V_j^p = V_s^p$). Gain ratio (GR) consists in dividing IG by (5). We employed both IG and GR to convert numerical and nominal attributes into binary ones. Fig. 4 gives a result of binarization of the section number attribute. According to the high purity of right splitter, we could conclude that there is a strong possibility that executables should be benign while they have more than eight sections.

$$H(X) = -\sum_{i=1}^m P_i \log_2 P_i, P_i = \frac{|X \cap C_i|}{|X|} \quad (1)$$

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

$$IG(Y^p, X) = H(Y^p, X) - H(X) \quad (3)$$

$$H(Y_s^p, X) = \begin{cases} -\frac{|X_{j \leq s}^p|}{|X|} H(X_{j \leq s}^p) - \frac{|X_{j > s}^p|}{|X|} H(X_{j > s}^p), & X: \text{num.} \\ -\frac{|X_{j=s}^p|}{|X|} H(X_{j=s}^p) - \frac{|X_{j \neq s}^p|}{|X|} H(X_{j \neq s}^p), & X: \text{nom.} \end{cases} \quad (4)$$

$$H_v(Y^p) = -\sum_{j=1}^k \frac{|X_j^p|}{|X|} \log_2 \frac{|X_j^p|}{|X|} \quad (5)$$

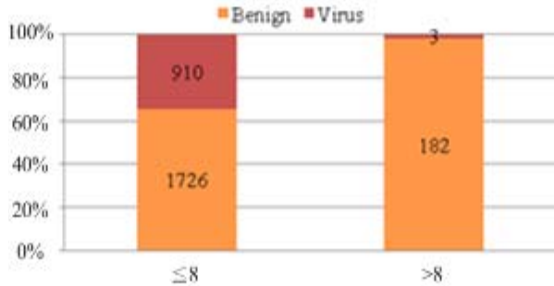


Figure 4. Binarization result of section number attribute

D. Attribute elimination

For the reason that a large number of attributes not only burden on the process of classifier training but also cause the curse of dimensionality [18] that diminishes the generalization property of the resulting classifier, we tried to eliminate two types of attributes: useless and redundant in this stage. We first discarded useless attributes which only belonged to one executable because they were too specific to identify other binaries. On the other hand, attributes that belonged to all binaries were also useless and should be rejected due to their lack of discrimination. Finally, we also eliminated redundant attributes that were completely correlated. The reason is that even these duplicate attributes might carry good classification information when treated separately; there is no added value when they are considered together. After this elimination stage, all executables were converted into Boolean vectors according to the remaining attributes. An example of vectorization is given in Table 2.

TABLE 2. REPRESENT EXECUTABLES BY VECTORS
(E:EXECUTABLE; A:ATTRIBUTE)

	A ₁	A ₂	A ₃	A ₄	...	A _n	Vector
E ₁	T	F	T	T	...	F	1011...0
E ₂	T	T	F	F	...	F	1100...0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
E _m	F	F	T	F	...	T	0010...1

E. Support vector machine and feature selection

Support vector machine (SVM) [4] is a learning algorithm based on substantial theory [20] and well-known for its high generalization property in several kinds of application such as text mining [19], biosequence analysis [5] and so forth. Giving a training dataset $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m) \in R^n \times \{\pm 1\}$, SVM tries to find a separating hyperplane $w^T x + b = 0$ with maximum margin to linearly separate two classes data. Parameter w is the normal vector of the hyperplane and b is the functional distance from the hyperplane to the origin. If the data are linearly inseparable, a positive weight parameter C and nonnegative slack variables ξ are both introduced to control the trade-off between the maximization of generalization ability and minimization of the classification error at the same time. As was mentioned above, SVM with linear kernel leads to the following optimization problem:

$$\begin{aligned} \min_{(w,b) \in R^{n+1}} \quad & \frac{1}{2} w^T w + C |\xi| \\ \text{s. t.} \quad & y_i (w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \dots, m \end{aligned} \quad (6)$$

We can convert (6) into (7) by introducing the nonnegative Lagrange multipliers α_i . The data that are associated with positive α_i are support vectors. Then, by solving (7) and excluding the data which are not support vector ($\alpha_i = 0$), we can obtain w and b to construct a decision function (8) for the classification problem where S is the set of support vector indices.

$$\begin{aligned} \max \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s. t.} \quad & \sum_{i=1}^m y_i \alpha_i = 0, 0 \leq \alpha_i \leq C \end{aligned} \quad (7)$$

$$D(x) = w^T x + b = \sum_{i \in S} \alpha_i y_i x_i^T x + b \quad (8)$$

Mapping the data from the original space to a high-dimensional feature space by a nonlinear function ϕ would also be helpful to solve the linearly inseparable problem. This mapping process can be done with a kernel function K that reduces the effort of data mapping by ϕ and dot product in the high-dimensional feature space. By using the kernel function, (7) can be changed into (9) and the decision function in the feature space is given in (10).

$$\max \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

s. t. $\sum_{i=1}^m y_i \alpha_i = 0, 0 \leq \alpha_i \leq C$ (9)

$$D(x) = w^T x + b = \sum_{i \in S} \alpha_i y_i K(x_i, x) + b \quad (10)$$

In this study, we directly used the RBF kernel due to the linear kernel is a special case of it [6] and the sigmoid kernel also behaves like RBF kernel for certain parameters [9]. (11) denotes the RBF function where γ is the parameter of the RBF kernel function.

$$K(x, y) = \exp(-\gamma \|x - y\|_2^2) \quad (11)$$

Since the deletion of a zero or little element of w does not affect the optimal hyperplane, the attribute corresponding to the element can also be deleted. Therefore, we first ranked all attributes by estimating their values described in (12) where $x_i^{(k)}$ is the i^{th} vector in S with its k^{th} element set to zero.

$$\Delta^{(k)} \|w\|^2 = \sum_{i,j \in S} \alpha_i \alpha_j y_i y_j \{K(x_i, x_j) - K(x_i^{(k)}, x_j^{(k)})\} \quad (12)$$

Finally, suppose that P is the performance of a RBF-SVM classifier and Y is the set of attributes which has been process by the elimination stage as stated above. Then the algorithm of SVM-based feature selection in feature space is represented below:

- Train a RBF-SVM classifier using all attributes in Y . Rank all attributes according to their $\Delta^{(k)} \|w\|^2$ in order.
- Delete one or a number of attributes with lower rank from Y and denote remaining attribute set as Y' .
- Use Y' to train a RBF-SVM classifier and get the P' of it. If $P' \geq P$, then set $P=P'$ and $Y=Y'$ and go to Step 2. Otherwise, stop this procedure if P' is significantly degraded and define Y' as the set of selected features.

V. EXPERIMENT RESULTS

In this study, LIBSVM [3] was utilized for the implementation for SVM and the algorithm of feature selection was coded by ourselves. Some terms used for evaluated our malware detection model are listed in the following:

- True Positives (TP): the number of malware correctly classified as malicious.
- True Negatives (TN): the number of benign programs correctly classified as benign.
- False Positives (FP): the number of benign programs falsely classified as malicious.
- False Negative (FN): the number of malware falsely classified as benign.
- Overall Accuracy: evaluation of the performance of a system. Higher TP and lower FP are necessary to be considered at the same time.
- Receiver Operating Characteristic (ROC) [17] curves and its area under curve (AUC) were both produced to

demonstrate the relationship between detection rate and false alarm rate.

- True Positive Rate (Detection Rate) is defined as $TP/(TP + FN)$
- False Positive Rate (False Alarm) is defined as $FP/(TN + FP)$
- Overall Accuracy Rate (OA) is defined as $(TP + TN)/(TP + TN + FP + FN)$

The detection accuracy over previously unseen data was calculated by five-fold cross-validation (CV). In other words, the dataset were randomly divided into five disjoint partitions with an equal number of sizes in each partition. One of them was chosen as the testing set and the remaining four partitions were merged to be the training set. This procedure was repeated five times by choosing different partition as the testing set. The average of the five OA results was set as the generalization ability of our malware detection model.

The results of experiment are described as follows: first of all, we evaluated the performance of our model without performing feature selection. As mentioned above, we used both IG and GR to binarize numerical and nominal attributes and got different amounts of attributes when using different binarization methods as shown in Table 3. Furthermore, it reveals that the number of attributes could be economical after removing useless and redundant attributes. We accomplished virus and email worm detection with 98.19% and 93.96% OA respectively. However, the results of 84.11% and 89.54% OA indicated that there is room for improvement of our model in detecting trojan and backdoor. Next, for each kind of malware, we then chose the attribute set which achieved better performance in previous experiment for the SVM-based feature selection. Table 4 presents the final performances of our malware detection model. The outcome reveals the fact that even a very few number of PE header features were adopted, the performance of our model were still retained and slightly improved. We only used 143 and 133 features to obtain 98.23% and 94.07% OA for virus and email worm detection respectively. Similarly, the performances of trojan and backdoor detections were not promising enough. Finally, Fig. 5 shows ROC curves for detecting the four kinds of malware.

VI. CONCLUSIONS AND FUTURE WORKS

In conclusion, there are three contributions of this study. First, we proposed a model which has high accuracy in unknown virus and email worm detection respectively. Second, we can guarantee that PE header entries is a proper kind of resource for virus and email worm detection. Finally, it is noticeable that both attribute elimination and SVM-based feature selection are useful to economize the cost of SVM classifier training without any decline of performance.

In future works, since we neglected the possible effect of executable packing [12], we should surmount this problem to investigate more sophisticated information in PE file to improve the detection ability of trojan and backdoor. Moreover, it is difficult to compare our experiment results with other studies owing to the lack of public datasets. To

overcome this problem, we will implement other models, which were published in former studies, and test them on our own dataset. Last but not least, it is important to practically evaluate the generalization ability of our model by some new malicious codes, though we have got significant performance in cross validation. We will accomplish it by training our classifier with a chronologically partitioned dataset.

REFERENCES

- [1] T. Abou-Assaleh, N. Cercone, V. Keselj and R. Sweidan, "N-gram-based Detection of New Malicious Code," In Proceedings of the 28th Annual International Computer Software and Applications Conference, IEEE Computer Society, 2004, pp. 41-42.
- [2] D.M. Cai, M. Gokhale and J. Theiler, "Comparison of Feature Selection and Classification Algorithms in Identifying Malicious Executables," Comput. Stat. Data Anal., vol. 51, no. 6, 2007, pp. 3156-3172.
- [3] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," 2001; Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [4] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.
- [5] S. Hua and Z. Sun, "A Novel Method of Protein Secondary Structure Prediction with High Segment Overlap Measure: Support Vector Machine Approach," Journal of Molecular Biology, vol. 308, no. 2, 2001, pp. 397-407.
- [6] S.S. Keerthi and L. Chih-Jen, "Asymptotic behaviors of support vector machines with Gaussian kernel," Neural Comput., vol. 15, no. 7, 2003, pp. 1667-1689.
- [7] J.Z. Kolter and M.A. Maloof, "Learning to Detect Malicious Executables in the Wild," In Proceedings of the Tenth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, ACM, 2004.
- [8] J.Z. Kolter and A.M. Marcus, "Learning to Detect and Classify Malicious Executables in the Wild," J. Mach. Learn. Res., vol. 7, 2006, pp. 2721-2744.
- [9] H.-T. Lin and C.-J. Lin, A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods, National Taiwan University, 2003.
- [10] M.M. Masud, L. Khan and B. Thuraisingham, "A Hybrid Model to Detect Malicious Executables," In Proceedings of the IEEE International Conference on Communications (ICC), Computer and Communications Network Security Symposium, 2007, pp. 1443-1448.
- [11] Microsoft, Microsoft Portable Executable and Common Object File Format Specification, 2008.
- [12] R. Perdisci, A. Lanzi and W. Lee, "Classification of Packed Executables for Accurate Computer Virus Detection," Pattern Recognition Letters, vol. 29, no. 14, 2008, pp. 1941-1946.
- [13] M. Pietrek, "Peering Inside the PE: A Tour of the Win32 Portable Executable File Format," Microsoft Systems Journal, vol. 9, no. 3, 1994, pp. 15-34.
- [14] J.R. Quinlan, "Induction of Decision Trees," Machine Learning, vol. 1, no. 1, 1986, pp. 81-106.
- [15] M.G. Schultz, E. Eskin, E. Zadok and S.J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," In Proceedings of the 2001 IEEE Symposium on Security and Privacy, IEEE Computer Society, 2001, pp. 38-49.
- [16] A.H. Sung, J. Xu, P. Chavez and S. Mukkamala, "Static Analyzer of Vicious Executables (SAVE)," In Proceedings of the 20th Annual Computer Security Applications Conference, IEEE Computer Society, 2004, pp. 326-334.
- [17] J. Swets and R. Pickett, Evaluation of diagnostic systems: Methods from signal detection theory, Academic Press, 1982.
- [18] S. Theodoridis and K. Koutroumbas, Pattern Recognition, Academic Press, 2006.
- [19] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," J. Mach. Learn. Res., vol. 2, 2002, pp. 45-66.
- [20] V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.
- [21] J.-H. Wang, P.S. Deng, Y.-S. Fan, L.-J. Jaw and Y.-C. Liu, "Virus Detection Using Data Mining Techniques," In Proceedings of the IEEE 37th Annual International Carnahan Conference on Security Technology, IEEE Computer Society, 2003, pp. 71-76.
- [22] T.-Y. Wang, et al., "A Surveillance Spyware Detection System Based on Data Mining Methods," In Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Computer Society, 2006, pp. 3236-3241.
- [23] T.-Y. Wang, C.-H. Wu and C.-C. Hsieh, "A Virus Prevention Model Based on Static Analysis and Data Mining Methods," In Proceedings of the IEEE 8th International Conference on Computer and Information Technology Workshops, IEEE Computer Society, 2008, pp. 288-293.

TABLE 3. PERFORMANCES WITHOUT SVM-BASED FEATURE SELECTION.

	Virus		Email Worm	
	IG	GR	IG	GR
Initial Attribute #	2675	2675	3207	3207
Useless Attribute #	1689	1695	2006	2031
Redundant Attribute #	618	620	779	778
Remaining Attribute #	368	360	422	398
OA	98.09%	98.19%	93.63%	93.96%
C	32.0	32.0	128.0	8.0
γ	0.5 ⁹	0.5 ⁹	0.5 ¹¹	0.5 ⁷

	Trojan		Backdoor	
	IG	GR	IG	GR
Initial Attribute #	4614	4614	4446	4446
Useless Attribute #	1585	1597	2725	2762
Redundant Attribute #	2769	2771	1069	1068
Remaining Attribute #	260	246	652	616
OA	82.67%	84.11%	89.54%	89.45%
C	32.0	2.0	32.0	32.0
γ	0.5 ⁵	0.5 ⁹	0.5 ⁶	0.5 ⁵

TABLE 4. PERFORMANCES WITH SVM-BASED FEATURE SELECTION

	Virus	Email Worm	Trojan	Backdoor
Eliminated Attribute #	217	265	122	407
Final Feature #	143	133	124	245
OA	98.23%	94.07%	84.20%	89.93%
C	2048.0	8.0	2.0	32.0
γ	0.5 ¹¹	0.5 ⁷	0.5 ⁵	0.5 ⁷
AUC	0.9886	0.9752	0.8932	0.9425

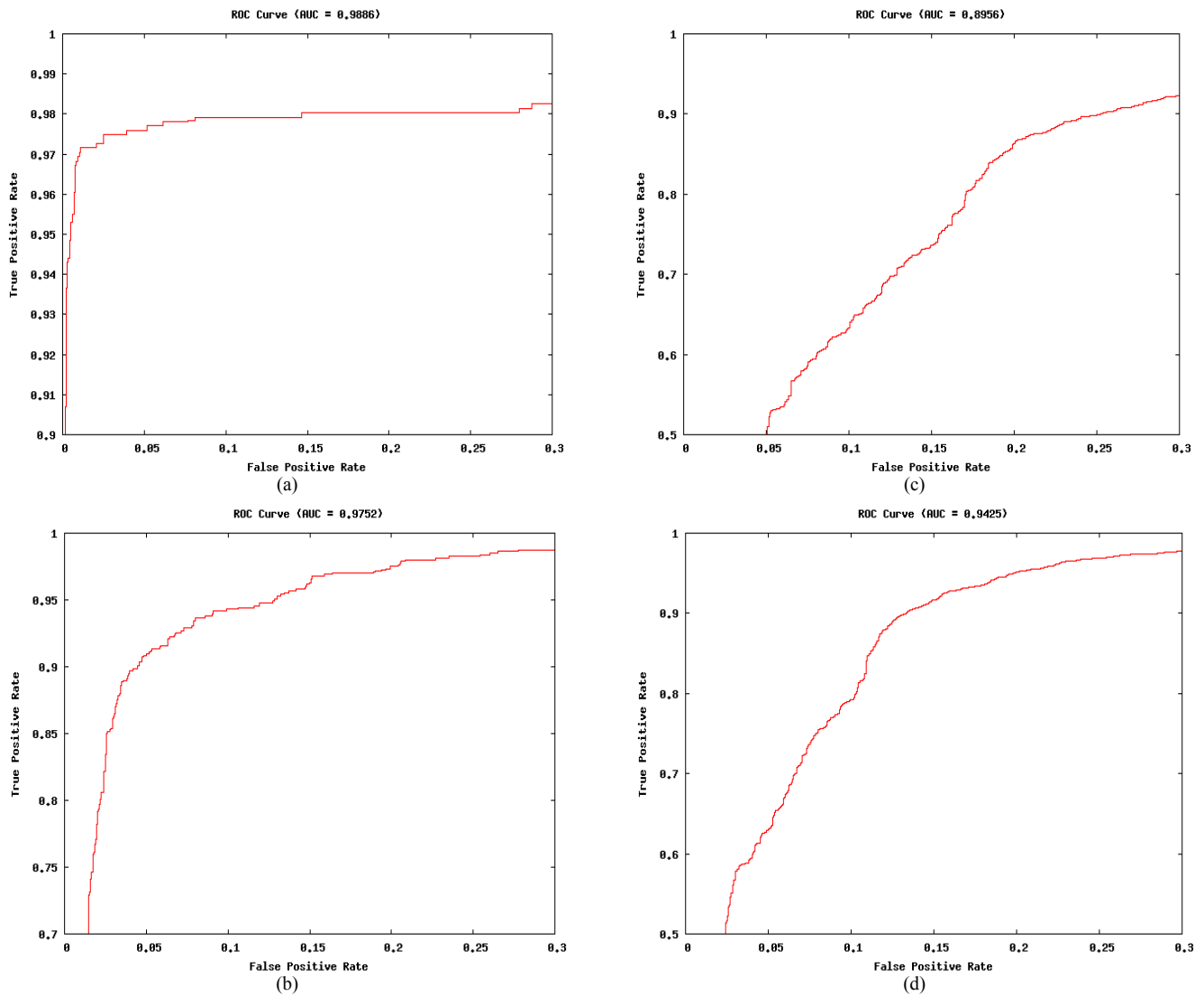


Figure 5. ROC curves for detecting malicious executables. (a): Virus detection. (b) Email Worm detection. (c) Trojan detection. (d) Backdoor detection