Classification of Malware Families Based on N-grams Sequential Pattern Features

Chatchai Liangboonprakong
Department of Computer Science
Suan Sunandha Rajabhat University
Bangkok, Thailand
chatcchai.li@ssru.ac.th

Ohm Sornil
Department of Computer Science
National Institute of Development Administration
Bangkok, Thailand
osornil@as.nida.ac.th

***Abstract*—Malware family identification is a complex process involving extraction of distinctive characteristics from a set of malware samples. Malware authors employ various techniques to prevent the identification of unique characteristics of their programs, such as, encryption and obfuscation. In this paper, we present n-gram based sequential features extracted from content of the files. N-grams are extracted from files; sequential n-gram patterns are determined; pattern statistics are calculated and reduced by the sequential floating forward selection method; and a classifier is used to determine the family of malware. Three classification models: C4.5, multilayer perceptron, and support vector machine are studied. Experimental results on a standard malware test collection show that the proposed method performs well, with the classification accuracy of 96.64%.**

Keywords—Malware Classification, N-Gram, Sequential Pattern, Sequential Floating Forward Selection, C4.5, Multilayer Perceptron, Support Vector Machine.

I. INTRODUCTION

Development of malware poses a major threat to modern information technology. A common method of launching computer attacks is by means of malware such as backdoors, Trojan horses, viruses and worms, which can cause severe damages to security and privacy of computer systems and networks worldwide. Malware has evolved into a powerful instrument for illegal commercial activities, and a significant effort is made by its authors to thwart detection by anti-malware products.

General analysis techniques for detecting malware are commonly classified into dynamic and static approaches. In dynamic analysis (also known as behavioral analysis), detection of malware relies on information that is collected from the operating system at runtime (i.e., during the execution of the program) such as system calls, network access and files [1]. This approach has several disadvantages. First, it is difficult to simulate appropriate conditions for malicious functions of a program, such as the vulnerable applications that the malware will be activated. Secondly, it is not clear what the required period of time is needed to observe the appearance of the malicious activity of a program. In static analysis, information about a program or its expected behaviors employs explicit and implicit observations in its binary code. The main advantage of static analysis is its ability to examine a suspected file without actually executing it and thereby provides rapid classification [2].

Our goal in this research is to introduce a novel set of features for effective classification of malware families. The features are based on n-gram sequential patterns, extracted from disassembling files. Three classification models are explored with the proposed features which consist of C4.5, multilayer perceptron, and support vector machine. Experimental evaluations on a standard malware data collection are performed to evaluate the proposed technique.

II. RELATED WORK

Several techniques have been studied in the past for malware detection. Cohen [23], Chess and White [24] use sandboxing to detect viruses. They show that in general the problem of virus detection is undecidable. Christodorescu and Jha [25] detect malicious code in executable files. Their implementation, called SAFE, handles most common types of obfuscations used by malware writers, such as insertion of NOPs between instructions, to evade detection.

In [33], Christodorescu et al. exploit semantic heuristics to detect obfuscated malware. Although, their approach works well for obfuscated malicious programs, the time taken (over a minute to classify) by their approach makes it impractical for use in commercial antivirus scanners. Kruegel et al [26] use control flow graph information and statistical methods for disassembling obfuscated executables. Bergeron et al. [27] consider critical API calls and security policies to test for presence of malicious code. Their approach does not work for obfuscated malicious executables. Zhang et al. [28] use fuzzy pattern recognition to detect unknown malicious code. The approach does not handle obfuscated program binaries and gives many false positives. Martignoni et al. [30] use real-time program monitoring to detect obfuscation in memory. Their implementation OmniUnpack detects obfuscation for both known and unknown packers. Zhang [31] identifies patterns of system or library functions called by a malware sample to detect their metamorphic versions. Bilar [29] uses statistical structures such as opcode frequency distribution and graph structure fingerprints to detect malicious programs.

An approach to represent malware content is the use of n-grams which are substrings of a larger string with length n. N-

978-1-4673-6322-8/13/$31.00 ©2013 IEEE

grams exist in constant malware parts although obfuscation techniques are used. These obfuscations are considered part of a file. Representations of malware by using n-gram profiles have been presented in the literature, see for example [32], [3] and [5]. In these studies some promising results towards malware detection are presented. However, malware domain has been evolving due to survivability requirements. Malware has to evade anti-virus scanners to perform its functions. Obfuscation techniques have been developed in order to avoid detection by anti-virus scanners. And these techniques affect n-gram features in the binary form of malware used by previous work. Similar methodologies have been used in source authorship, information retrieval, and natural language processing [34], [35]. Abou-Assaleh et al. [3] contribute to the ongoing research while using common n-gram profiles. The K nearest neighbor algorithm with k=1 is used. A feature set is constituted by using the n-grams and the occurrence frequency. Tests have been done with different values of n (ranging from 1 to 10) and frequencies (ranging from 20 to 5000). The data set used in these experiments comprises 25 malware and 40 benign files. The test results show 98% of success. Using the data in [3], the accuracy slightly drops to around 94%. Kolter et al. [5] use 4-grams as features and select top 500 n-grams through information gain measure. They use instance based learners, TF-IDF, naïve Bayes, support vector machines, and decision trees and also boost the last three learners. The boosted decision tree outperforms all other and gives promising results, such as ROC curve of 0.996. Walenstein et al. [9] explore the use of n-grams and unordered n-perms to disassembly. This has the benefit of using features from specific malware, however, because all n-perms of a file are considered, there is no opportunity to select those features that may distinguish a family in question from other similarly constructed files.

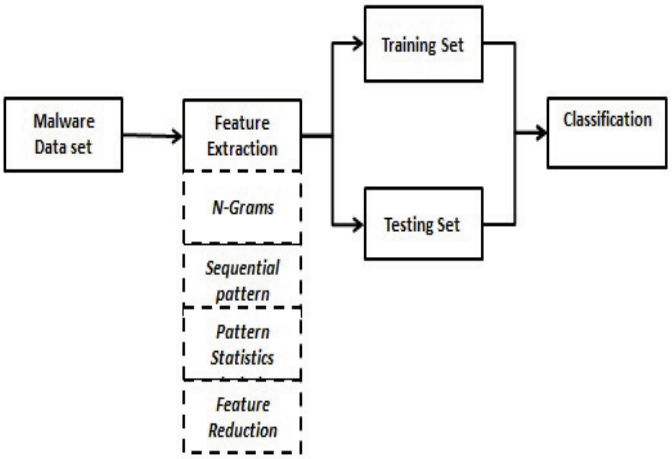

Figure 1. Architecture of the proposed method.

III. METHODOLOGY

In this section, the malware detection process, as shown in Figure 1, is described.

A. FeaturesExtraction

Features are created in 4 major steps: n-gram extraction, sequential pattern extraction, pattern statistics calculation, and feature reduction.

1) N-grams extraction

An n-gram is an n-character slice of a longer string. First, IDA-Pro [11], a tool that disassembler files, is used to extract content of a file into a long string of hexadecimals. The string is then processed into a set of overlapping n-grams. In our study, we explore n-grams of several different lengths. The kfngram tool [18] is employed to generate n-gram slices. In the experiments our tests are run with n=1, n=2, n=3 and n=4. The resulting n-gram-malware matrix is shown in Figure 2.

2) Sequential pattern extraction

There are a very large number of n-grams, but there is no order information. Sequential pattern extraction [19] is employed to find the frequently occurred sequences to describe the data. We implemented the techniques according to [12,13] which can reduce response time to find patterns in sequence data.

After processing a malware into n-grams, malware m can be represented by a set of malware file *F(m)*.

Malware File	Terms (N-Gram)					
mw1	t1	t2	t3	t9	.	.
mw2	t3	t4	t6	t8	.	.
mw3	t3	t4	t6	t7	.	.
mw4	t3	t4	t6	t8	.	.
mw5	t1	t2	t4	t6	.	.
mw6	t1	t2	t4	t9	.	.
mw7	t2	t5	t6	t7	.	.
mw8	t2	t5	t6	t7	.	.
mw9	t3	t4	t5	t8	.	.
…	.	.	.	.	.	.
mw$n$	ti	.	.	.	.	tk

Figure 2. N-gram-malware matrix.

Let $T = \{t_1, t_2, ..., t_m\}$ be a set of terms (n-grams) which are resulted from n-gram extraction. Given X be a set of terms *(termset)* in malware m, *coverset(X)* denotes the covering set of X for m, which includes all malware file $mw \in F(m)$, where $X \subseteq mw$, i.e., $coverset(X) = \{mw \mid mw \in F(m), X \subseteq mw\}$, the absolute support of X is the number of occurrences of X in $F(m)$: $sup_a(X) = |coverset(X)|$, the relative support of X is the fraction of the malware file that contain the pattern: $sup_a(X) = |coverset(X)| / |F(m)|$. The relationships between frequent patterns and covering sets are shown in Figure 3. A termset X called frequent pattern if its $sup_a \geq min\_sup$, a minimum support.

Frequent Pattern	Covering Set
{t3,t4,t6}	{mw2,mw3,mw4}
{t3,t4,t5}	{mw4,mw9}
{t1,t2,t4}	{mw5,mw6}
{t2,t5,t6,t7}	{mw7,mw8}
{t3,t4,t6,t8}	{mw2,mw4}
{t3,t4}	{mw2,mw3,mw4}
{t4,t6}	{mw2,mw3,mw4}
...	...
{ti,...,tk}	{mwn,...,mwm}

Figure 3. Frequent patterns and covering sets.

A sequential pattern is an ordered list of terms, where its

$$S =< t_1,...,t_r >: (t_i \in T)$$

$$sup_a \geq min\_sup \quad (1)$$

A sequence $s_1 =< x_1,...,x_i >$ is a sub-sequence of another sequence $s_2 =< y_1,...,y_i >$, denoted by $s_1 \subseteq s_2$, if $\exists\ j_1,...,j_y$ such that $1 \leq j_1 < j_2 ... < j_i < j$ and $x_1 = y_{j1}, x_2 = y_{j2},..., x_i = y_{ji}$. Given $s_1 \subseteq s_2$, we usually say s_1 is a sub-pattern of s_2, and s_2 is a super-pattern of s_1. To simplify the explanation, we refer to sequential patterns as patterns.

3) Pattern statistics calculation

After n-gram patterns are generated, they will be organized in the following fashion:

$$\vec{d}_i = \left\langle \left(s_{i_1}, f_{i_1}\right), \left(s_{i_2}, f_{i_2}\right), ..., \left(s_{i_m}, f_{i_m}\right) \right\rangle$$

$$Y = \left\{ \vec{d}_1, \vec{d}_2, ..., \vec{d}_n \right\} \quad (2)$$

where s_j in pair (s_j, f_j) denotes a pattern, and f_j is its frequency in d_i, thus the result of this algorithm is a set of malware vectors. For each vector $\vec{d}_i \in Y$, the pattern significance can be calculated using term frequency-inverse document frequency (TF-IDF) weighting, where a term represents an n-gram sequential pattern, and a document represents a record of malware. A TF-IDF weight can be calculated as follows:

$$Y(w_{ij}) = TF(w_{ij}) * IDF(w_{ij}) \quad (3)$$

As a result, TF-IDF filters out common n-gram sequential patterns by giving low weights to patterns that appear frequently in the data set.

4) Feature reduction

There can be a large number of patterns. Although all of these features constitute the inputs of a classifier, they have different impacts to the classification performance. Some features may not increase the discriminative power of the classification among pattern classes. Vice versa some features may be highly correlated, and some may even be irrelevant for a specific classification. The sequential floating forward selection (SFFS) procedure [14] is thus applied to find a minimum feature set. It consists of applying after each forward step a number of backward steps as long as the resulting subsets are better than the previously evaluated ones at that level. Consequently, there are no backward steps at all if the performance cannot be improved. The SFFS method can be described, as follows:

Input: $Y = \{ \vec{d}_i \mid j = 1,...,n \}$

Output: $X_k = \{x_i \mid j = 1,...,k, xj \in Y\}\ k = 0,1,...,n$

Initialize feature set: $Y_0 = \{\phi\}; m = 0; J(m) = 0$.

Step 1: Find the best feature and update Y_m.

$$x^+ = \arg\max[J(Y_m - x)]; x \notin Y_m$$

$$Y_m = Y_m + x^+; m = m + 1$$

Step 2 : Find the worst feature

$$x^- = \arg\max[J(Y_m - x)]; x \notin Y_m$$

If $J(Y_m - x^-) > J(Y_m)$ then $Y_{m+1} = Y_m - x; m = m + 1$.

Go to step 1

Else go to step 2 (4)

B. Classification

The classification model accepts the feature vector and returns the family of the malware. Three learning algorithms are studied in this research, which consist of C4.5, multilayer perceptron, and support vector machine. They are available in KNIME [15]. The malware is randomly split into two partitions: 80% for training and 20% for testing.

1) Decision Tree (C4.5)

The C4.5 decision tree [20] is a powerful and popular tool for classification and prediction. The algorithm uses gain ratio as the impurity measure for split calculation which can be calculated as:

$$\text{Information Gain} = I(parent) - \Sigma^k_{j=1} N(v_j) / N * I(v_j)$$

$$\text{Split info} = -\Sigma^k_{i=1} P(v_i) \log 2\ P(v_i)$$

$$\text{Gain Ratio} = \text{information gain / split info} \quad (5)$$

A decision tree has three main components: nodes, arcs and leaves. Each node is labeled with feature attribute which is most informative among the attributes not yet considered in the path from the root. Each arcs out of a node is labeled with a feature value for the node, and each leaf is labeled with a category or class. A decision tree can then be used to classify a data point by starting at the root of the tree and moving through

it until a leaf node is reached. The leaf node would then provide the classification of the data point.

2) Artificial Neural Network (ANN)

We have used RProp MLP implementation in this study [21]. The RProp algorithm is a learning algorithm for multilayer feedforward networks. The following pseudo-code fragment shows the kernel of the RPROP adaptation and learning process.

$$
\begin{aligned}
&For\ all\ weights\ and\ biases\ \{\\
&\quad If\ (\partial E / \partial w_{ij}(t\text{-}1) * \partial E / \partial w_{ij}(t) > 0)\ then\ \{\\
&\qquad \Delta_{ij}(t) = minimum\ (\Delta_{ij}(t-1) * n^{+}, \Delta_{max})\\
&\qquad \Delta w_{ij}(t) = -sign\ (\partial E / \partial w_{ij}(t)) * \Delta_{ij}(t)\\
&\qquad w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)\\
&\quad \}\\
&\quad Else\ if\ (\partial E / \partial w_{ij}(t\text{-}1) * \partial E / \partial w_{ij}(t) < 0)\ then\ \{\\
&\qquad \Delta_{ij}(t) = maximum\ (\Delta_{ij}(t-1) * n^{-}, \Delta_{min})\\
&\qquad \Delta w_{ij}(t+1) = w_{ij}(t) - \Delta w_{ij}(t-1)\\
&\qquad \partial w_{ij}(t) = 0\\
&\quad \}\\
&\quad Else\ If\ (\partial E / \partial w_{ij}(t\text{-}1) * \partial w_{ij}(t) = 0)\ then\ \{\\
&\qquad \Delta w_{ij}(t) = -sign\ (\partial E / \partial w_{ij}(t)) * \Delta_{ij}(t)\\
&\qquad w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)\\
&\quad \}\\
&\}
\end{aligned}
\tag{6}
$$

The minimum (maximum) operator is supposed to deliver the minimum (maximum) of two numbers; the sign operator returns +1 if the argument is positive, -1 if the argument is negative, and 0 otherwise. To overcome the inherent disadvantages of the pure gradient-descent, RPROP performs a local adaptation of the weight-updates according to the behavior of the error function. In substantial difference to other adaptive techniques, the effect of the RPROP adaptation process is not blurred by the unforeseeable influence of the size of the derivative but only dependent on the temporal behavior of its sign. This leads to an efficient and transparent adaptation process.

3) Support Vector Machine (SVM))

SVM represent a supervised learning technique suitable for solving classification problems with high dimensional feature space. In this study, we use the LIBSVM implementation [22] with polynomial kernels function to train SVM. Although the basic technique is conceived for binary classification, several methods for single and multi-class problems have been developed. Being a supervised method, it relies on two phases: during the training phase, the algorithm acquires knowledge about the classes by examining the training set that describes them. During the testing phase, a classification mechanism examines the test set and associates its members to the classes that are available. The target of the algorithm is the estimation of boundaries between the classes. Given training vectors $x_i \in R^n$, $i = 1, 2, ..., l$ in two classes, and a vector $y \in R^l$ such that each $y_i \in \{+1,-1\}$, an SVM for non-separable data considers the following optimization problem [22]:

$$
\begin{aligned}
&min\ \tfrac{1}{2} * w^T w + C\Sigma^{l}_{i=1}\ \alpha_i y_i (w^T K(s_i, x) + b),\\
&subject\ to\ \alpha_i \geq 0, i = 1, 2, ..., l
\end{aligned}
\tag{7}
$$

In the objective function, w is a perpendicular to the hyperplane that separates the positive and negative points, C is a parameter that is used to cost the α_i, $K(s_i, x)$ is a non-linear kernel that maps the input data to another (possibly infinite dimensional) Euclidean space, and s_i are the points called the support vectors that maximize the separation between the positive and negative. Default settings are chosen for all other parameters.

IV. Data Set

VX Heavens Virus Collection [10] database is used as the dataset for evaluations. In our study, we classify malware into 10 families. Backdoors contains one family with 2,014 files. Trojan horses consist of 4 families which are Trojan.BAT (834 files), Trojan.DOS (858 files), Trojan-Downloader.win32 (624 files), and Trojan.win32 (818 files). Viruses contains 3 families: Virus.MSWord (712 files), Virus.DOS (984 files), and Virus.BAT (590 files). Worms consists of 2 families: Worm.win32 (1,907 files) and Email-worm.win32 (2,854 files). All together, there are 10 malware families with a total of 12,199 files.

TABLE I. Feature Charateristics

N	Feature Characteristics		
	N-Grams	***Sequential Patterns***	***Final Features***
1	256	2,169	1,356
2	35,491	22,313	6,714
3	104,213	55,023	10,482
4	218,469	87,285	14,908

V. Result

The dataset is divided into two subsets 80% for training and 20% for testing. Three different classification models are explored.

TABLE II. MALWARE CLASSIFICATION PERFORMANCE

N-GRAMS	Accuracy		
	DT (C4.5)	*ANN (RProp MLP)*	*SVM*
1-gram	71.83%	72.29%	**75.44%**
2-gram	82.10%	82.85%	**83.91%**
3-gram	88.74%	84.82%	**92.96%**
4-gram	91.25%	88.31%	**96.64%**

Table 1 shows the numbers of n-grams, sequential patterns, and final features while varying the values of *n* from 1 to 4. We can see that a higher value of *n* yields more n-grams; more n-gram yields more sequential patterns, thus a larger set of final features after feature reduction. The final feature sets consist of: 1,356 patterns for 1-gram, 6,714 patterns for 2-gram, 10,482 patterns for 3-gram, and 14,908 patterns for 4-gram. These features are then used in the model.

For classification process, we have ten malware families or ten classes to be determined. C4.5, ANN, and SVM are tested with each n-gram size. The classification results are shown in Table 2.

We can observe that SVM is the best classification in every size of n-gram; the receiver operating characteristics (ROC) curve (Figure 4) also shows the higher relative performance of SVM over the other two classifiers. However, the relative performance between C4.5 and ANN is unclear. With $n = 1$ and 2, ANN tends to give higher accuracy while the opposite is observed with $n = 3$ and 4. The larger n-gram yields the higher accuracy in general to capture unique characteristics of different families in the presence of various obfuscations.

The experimental results for classification of malware families used in experiments show that the proposed features have the ability to achieve high classification accuracy.

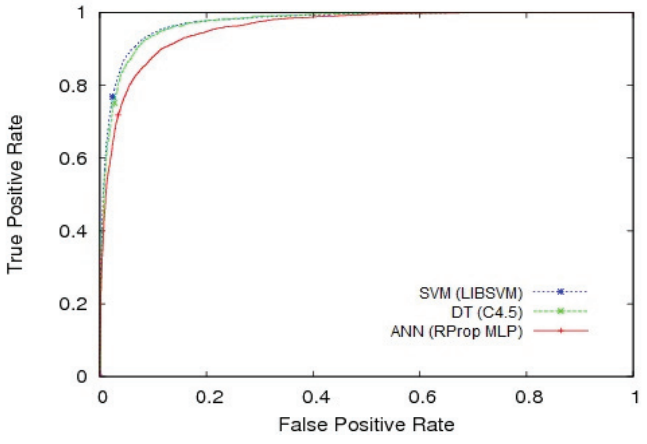

Figure 4. ROC plot for classification.

VI. CONCLUSIONS

Malware family identification is a complex process involving extraction of distinctive characteristics from a set of malware samples while authors employ various obfuscation techniques to prevent the identification of unique characteristics of their programs. In this paper, we propose n-grams sequential pattern features for classifying malware into 10 families. N-grams are created from the binary content of files; n-gram sequential patterns are formed; and patterns are reduced to a minimal set by sequential floating forward selection procedure. Four different sizes of n-grams (n = 1, 2, 3, and 4) are studied; and 3 classification models (C4.5 decision tree, artificial neural network, and support vector machine) are studied. Due to the complexities of malware, the larger n-gram size yields the higher accuracy. The proposed feature achieves 96.64% in accuracy with 4-gram and support vector machine.

REFERENCES

[1] K. Rieck, T. Holz, P. Dussel, and P. Laskov, "Learning and classification of malware behavior" in Conference on Detection of Intrusions and Malware & Vulnerability Assessment Heidelberg, Springer, 2007, pp. 108-125.

[2] A. Shabtai, D. Potashnik, Y. Fledel, R. Moskovitch, and E. Elovici, "Monitoring analysis and filtering system for purifying network traffic of known and unknown malicious content" Security and Communication Networks, 2010.

[3] T. Abou-Assaleh, V. Keselj, and R. Sweidan, "N-gram based detection of new malicious code" Proceeding of the 28th Annual International Computer Software and Applications Conferen c2007e IEEE Computer Society, 2004, pp. 41-42.

[4] M. G. Schultz, E. Eskin, E. Zadok and S.J. Stolfo, "Data Mining Methods for Detection of New Malicious Excutables" Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001, pp. 38-49.

[5] J. Z. Kolter and M. A. Maloof, "Learning to Detect Malicious Excutable in the Wild" The Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA, 2004 pp. 470-478.

[6] J. Z. Kolter and M. A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild" Journal of Machine Learning Research, 2006, pp. 2721-2744.

[7] M. Schmall, "Heuristic Techniques in Anti-Virus Solutions: An Overview", 2002.

[8] M. Baily, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian and J. Nazario, "Automated classification and analysis of internet malware" Lecture Notes in Computer Science, Spinger, pp.178-197.

[9] A. Walenstein, M. Venable, M. Hayes, C. Thompson, and A. Lakhotia, "Exploiting similarity between variants to defeat malware" Proceeding of BlackHat 2007 DC Briefings, 2007.

[10] VX Heavens Virus Collection, VX Heavens website, available at http://vx.netlux.org

[11] IDA-Pro Tool, available at http:// www.hex-rays.com

[12] L. Pipanmaekaporn and Y. Li, "Mining a Data Reasoning Model for Personalized Text Classification" IEEE Intelligent Informatics Bulletin, 2011, pp. 17-24.

[13] N. Zhong, Y. Li and S. T. Wu, "Effective Pattern Discovery for Text Mining" IEEE Transactions on Knowledge and Data Engineering, 2012.

[14] P. Pudil, F. J. Ferri, J. Novovicova and J. Kittler, "Floating search methods for feature selection with nonmonotonic criterion functions" International Conference on Pattern Recognition, 1994, pp. 279-283.

[15] KNIME Data Mining Tool, available at http:// www.knime.org

[16] 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 Conference on security Technology, 2003, pp.71-76.

[17] P. Kierski, M. Okoniewski, P. Gawrysiak, "Automatic Classification of Executable Code for Computer Virus Detection" International Conference on Intelligent Information Systems, Springer, Poland, 2003, pp. 277-284.

[18] KfNgram, available at http://www.kwicfinder.com/kfNgram

[19] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases", 1994, pp. 478-499.

[20] J. Ross Quinlan, "C4.5: Programs for Machine Learning" Morgan Kaufman, 1993.

[21] H. Riedmiller, and M. Braun, "A direct adaptive method for faster backpropagation learning: theRPROP algorithm" Proceedings of the IEEE International Conference on Neural Networks (ICNN) pp. 586-591.

[22] LIBSVM Tool, available at http://www.csie.ntu.edu.tw/~cjlin/

[23] F. Cohen, "Computer Virus Theory and experiments", Computers and Security 6, 1987, pp. 22–35.

[24] D.M. Chess, and S.R. White, "An undetectable computer virus" In Proceedings of Virus Bulletin Conference, 2000.

[25] M. Christodorescu, and S. Jha, "Static Anlaysis of Executables to Detect Malicious Patterns", In Proceeding of the 12th USENIX Security Symp Security 2003, pp.169–186.

[26] C. Kruegel, W. Robertson, F. Valeur, and G. Vigna, "Static disassembly of obfuscated binaries", In Proceedings of USENIX Security, San Diego, CA, 2007, pp. 255–270.

[27] J. Bergeron, M. Debbabi, J. Desharnais, M.M Erhioui, Y. Lavoie, and N. Tawbi, "Static Detection of Malicious Code in Executable Programs", In Symposium on Requirements Engineering for Information Security, 2001.

[28] B. Zhang, J. Yin, and J. Hao, "Using Fuzzy Pattern Recognition to Detect Unknown Malicious Executables Code", Fuzzy Systems and Knowledge Discovery. LNCS (LNAI), vol. 3613, 2005, pp. 629–634.

[29] D. Bilar, "Statistical Structures: Tolerant Fingerprinting for Classification and Analysis", Las Vegas, NV. Blackhat Briefings USA, 2006.

[30] L. Martignoni, M. Christodorescu, and S. Jha, "OmniUnpack: Fast, Generic, and Safe Unpacking of Malware", In Twenty-Third Annual Computer Security Applications Conference (ACSAC), Miami Beach, 2007.

[31] Q. Zhang, and D.S. Reeves, "MetaAware: Identifying Metamorphic Malware", In: Annual Computer Security Applications Conference, 2007.

[32] I. Santos, Y. K. Penya, J. Devesa, and P.G. Bringas, "n-Grams-Based File Signatures For Malware Detection", The Proceedings of the 11th International Conference on Enterprise Information Systems, Volume AIDSS, 2009, pp. 317-320.

[33] M. Christodorescu, S. Jha, A. Seshia, D. Song, R.E. Bryant, "Semantics-Aware Malware Detection", In Proceedings of the 2005 IEEE Symposium on Security and Privacy, May 08-11, 2005, pp. 32–46.

[34] S. Burrows and S. M. M Tahaghoghi, "Source code authorship attribution using n-grams", In Proceedings of the Twelfth Australasian Document Computing Symposium, A. Spink, A. Turpin, and M. Wu, Eds. RMIT University, Melbourne, Australia, 2007, pp. 32–39.

[35] G. Frantzeskou, E. Stamatatos, S. Gritzalis and S. Katsikas, "Effective identification of source code authors using byte-level information", In Proceedings of the Twenty-Eighth International Conference on Software Engineering, ACM Press, Shanghai, China, 2006, pp. 893–896.

[36] Y. Ye, T. Li, Q. Jiangshan and Y. Wang, "CIMDS: Adapting Postprocessing Techniques of Associative Classification for Malware Detection", IEEE : Applications and Reviews, Vol. 40, No. 3, 2010.