Author(s): Zolotukhin, Mikhail; Hämäläinen, Timo

Title: Support Vector Machine Integrated with game-theoretic approach and genetic algorithm for the detection and classification of malware

Year: 2013

Version:

Please cite the original version:

Zolotukhin, M., & Hämäläinen, T. (2013). Support Vector Machine Integrated with game-theoretic approach and genetic algorithm for the detection and classification of malware. In IEEE Globecom 2013 Conference Proceedings : Big Security 2013, First International Workshop on Security and Privacy in Big Data (pp. 211-216). IEEE. IEEE Global Telecommunications Conference. https://doi.org/10.1109/GLOCOMW.2013.6824988

Support Vector Machine Integrated with Game-Theoretic Approach and Genetic Algorithm for the Detection and Classification of Malware

Mikhail Zolotukhin, Timo Hämäläinen
Department of Mathematical Information Technology
University of Jyväskylä, Jyväskylä, Finland,
email: {mikhail.m.zolotukhin, timo.t.hamalainen}@jyu.fi

***Abstract*—In the modern world, a rapid growth of malicious software production has become one of the most significant threats to the network security. Unfortunately, widespread signature-based anti-malware strategies can not help to detect malware unseen previously nor deal with code obfuscation techniques employed by malware designers. In our study, the problem of malware detection and classification is solved by applying a data-mining-based approach that relies on supervised machine-learning. Executable files are presented in the form of byte and opcode sequences and n-gram models are employed to extract essential features from these sequences. Feature vectors obtained are classified with the help of support vector classifiers integrated with a genetic algorithm used to select the most essential features, and a game-theory approach is applied to combine the classifiers together. The proposed algorithm, ZSGSVM, is tested by using a set of byte and opcode sequences obtained from a set containing executable files of benign software and malware. As a result, almost all malicious files are detected while the number of false alarms remains very low.**

I. Introduction

Malicious software, or malware, remains a significant threat to the Internet and today's computing community [1]. The recent growth in high-speed internet connections and internet network services has led to an increase in the creation of new malicious code, mainly for the theft of personal information and recruitment of computers to botnets [2]. Moreover, malware designers apply sophisticated techniques to hide the presence of their creations in a computer system, making the problem of malware detection even more difficult [3].

A dramatic increase in malware production has resulted in the development of new tools and strategies to detect malicious software. Despite this, signature-based approach of malware detection remains the most widespread commercial anti-malware solution. As a rule, software based on such approach searches for a specific signature inside a file analyzed. The signature can contain a specific sequence of bytes or a portion of a machine language instruction. Unfortunately, a malware signature cannot be extracted until an instance of this malware has damaged several computers or networks. Thus, the signature-based approach cannot detect previously unseen malicious software. Furthermore, this approach cannot cope with code obfuscation techniques such as garbage insertion, code reordering and variable renaming, which are employed by malware designers to hide the actual behavior of their malicious creations [4], [5], [6].

Data-mining-based approach can be used to deal with the problem caused by code obfuscation. This approach involves the analysis of a dataset that includes several characteristic features extracted from malicious samples and benign software to build a classification tool that is able to detect undocumented malware [13]. Data mining approaches rely on machine-learning algorithms that can be classified into three different types: supervised learning [7], unsupervised learning [8] and semi-supervised learning [9].

The extraction of features to build a model for malware detection is usually carried out by analyzing byte sequences of executable binaries. Study [10] proposes a method to analyze binary content of files by using n-gram analysis and efficient statistical modeling techniques in order to determine the validity of file type in network traffic flows or on a local disk. In [11], a byte-frequency based detection model to deal with the problem of malware variants detection is proposed. In addition, recent studies have investigated the ability of operational codes (opcodes) to detect malicious software [12]. An opcode is the portion of a machine language instruction that specifies the operation to be performed. In studies [13] and [14], detection of malicious code is based on previously seen examples and carried out with the help of opcode n-gram representation and several well-known classifiers.

After a malicious software or a file already infected by that software has been detected, the anti-malware system performs a specific action depending on the malware characteristics. A proper determination of the malware type allows detecting the emergence of new threats and assesses the risk in quarantine and cleanup. There are several researches that are devoted to automated classification and analysis of malicious software. Paper [2] presents an effective algorithm, which uses a diversity of static feature selection methods to identify and classify malware families and distinguish malware from goodware. Study [15] proposes a classification method based on function level similarity comparison, which is founded on the observation that most malware variants are generated with metamorphic engines or malware generating tools and that those originated from the same program share most of their components.

In this research, we apply the data-mining-based approach for both the detection of malware and its classification. Let us assume that there is a quite big set of properly labeled executable files. This allows us to apply supervised machine-

978-1-4799-2851-4/13/$31.00 ©2013IEEE

learning leaving the analysis of unsupervised malware detection methods for a future work. Files of this set are presented in the form of byte and opcode sequences and n-gram models are employed to extract essential features from these sequences. A classification model is then built with the help of support vector machines, which are well-known binary classifiers. The problem of the classifiers combination is considered as a decision-making task and game theory methods are applied to predict the class or to estimate class probabilities. A genetic algorithm is used to select the most essential features and, therefore, cope with the high dimensionality of the problem.

The rest of the paper is organized as follows. Feature extraction based on applying n-gram models to byte and opcode sequences is considered in Section II. In Section III, we present the classic support vector machine, genetic algorithm and some basics of game theory. Section IV introduces a model which is built with the help of feature vectors extracted and used to detect malware. In Section V, we present several simulation results to evaluate the algorithm proposed and compare it with some analogues. Finally, Section VI draws the conclusions and outlines future work.

II. Feature Extraction

Executable files can be presented in the form of byte or opcode sequences [11], [12]. An opcode is the portion of a machine language instruction that specifies the operation to be performed: arithmetic or data manipulation, logical operation or program control. Opcodes reveal significant statistical differences between malware and legitimate software and even single opcodes are able to serve as the basis for the detection of malicious executables [12]. Opcodes can be used with one or more operands which show upon what data the operation should act. Since the operands strongly depend on CPU architecture and can be used by malware designers to hide malicious code [14], we analyze only the sequence of opcodes without taking into consideration opcode parameters.

An n-gram word model is applied to transform all byte and opcode sequences extracted from executable files of a training set to sequences of n-grams. An n-gram is a sub-sequence of n overlapping items (characters, letters, words, etc) from a given sequence [16]. N-gram sequences are then used to construct n-gram frequency vectors, which express the frequency of appearance of every n-byte and n-opcode. To obtain such vector for opcode n-grams, we find all unique n-opcodes contained in the executables of the training set and build the frequency vector by counting the number of occurrences of each such n-opcode entry in the analyzed sequence. In the same manner, a frequency vector for byte n-grams can be extracted. Thus, each executable file is transformed to two numeric vectors of lengths N_{oc} and N_b equal, respectively, to the number of unique opcode and byte n-grams found in the training set.

III. Mathematical Background

The algorithm proposed to build a classification tool that is able to detect malicious executable files relies on a genetic algorithm (GA) to select the most essential features, support vector machines (SVMs) to classify executable files and the solution of a zero-sum game (ZSG) to combine classifiers together. These mechanisms are explained in more detail in the next subsections.

A. Genetic Algorithm

Genetic algorithms belong to a class of stochastic optimization algorithms in which the principles of organic evolution are used as rules in optimization. They are often applied to optimization problems when specialized techniques are not available or standard methods fail to give satisfactory answers. GAs are also used to automatically determine the relative importance of many different features and to select a good subset of features available to the system [17], [18].

As usual, GA starts with an initial set of feasible solutions (called population) and tends to an optimal solution using processes similar to evolution: crossover and recombination. Crossover is a genetic operator that combines two solutions (parents) to produce a new solution (offspring). The idea behind crossover is that the new solution may be better than any of the parents if it takes the best characteristics from each of the parents. Recombination produces spontaneous random changes in various solutions of the current population and that might improve those solutions. Crossover and recombination contribute new solutions to the population. During each iteration of the algorithm (generation) all members of the current population are evaluated: better solutions have a higher probability to be selected for the new population. The algorithm stops when some stopping criterion is fulfilled: usually, a maximal number of generations is reached or a maximal number of function evaluations is made.

Genetic algorithms where the best individuals survive with the probability of one are usually known as elitist genetic algorithms. Elitism guarantees survival of the best element of the population and therefore guarantees that at least the fitness of the population measured as the fitness of the best individual does not decrease after the next iteration. The elitist genetic algorithm and theoretical estimation for its convergence are considered in study [19].

B. Support Vector Machine

Support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns. Over the last decade, SVMs have been applied for many classification problems because of their flexibility and computational efficiency. As a rule, during the training an SVM takes a set of input points, each of which is marked as belonging to one of two categories, and builds a model representing the input points in such way that the points of different categories are divided by a clear gap that is as wide as possible. Thereafter, a new data point is mapped into the same space and predicted to belong to a category based on which side of the gap it falls on.

SVM models can efficiently perform linear and non-linear classification by mapping input vectors into high-dimensional feature spaces. A linear SVM model separates data belonging to different categories by using a hyperplane so that the distance from it to the nearest data point on each side is maximized. In case such a hyperplane does not exist, the algorithm chooses a hyperplane that splits input points as cleanly as possible. For mislabeled points, a penalty function

which measures the degree of misclassification of the data points is introduced. Thus, the model is built to maximize the distance from the separating hyperplane to the nearest data point on each side, taking into account the penalties caused by mislabeled data points. The kernel trick allows the SVM algorithm to become nonlinear to separate points by a hyperplane in a transformed feature space.

A regular SVM model classifies data belonging to two different categories. Let us consider a classification problem where m samples belong to two categories: if sample x_i belongs to the first category then it has label $y(x_i) = 1$, and otherwise $y(x_i) = -1$. In this case, the hyperplane (w, b) for the SVM model can be found after solving the following optimization problem:

$$\begin{aligned} &\min_{w,b,\epsilon_i} \frac{1}{2} w^T w + C \sum_{i=1}^{m} \epsilon_i, \\ &\text{subject to} \begin{cases} y(x_i)(w^T \phi(x_i) + b) \geq 1 - \epsilon_i, \\ \epsilon_i > 0, \quad \forall i = \{1, \ldots, n\}, \end{cases} \end{aligned} \tag{1}$$

where ϵ_i is the slack variable, which measures the degree of misclassification of x_i, C is the penalty parameter and function $\phi(x)$ maps x to a higher dimensional space. Once optimal hyperplane (w, b) is found, a new sample x is classified as follows:

$$y(x) = \begin{cases} 1, \text{ if } s(x) \geq 0, \\ -1, \text{ if } s(x) < 0, \end{cases} \tag{2}$$

where function $s(x)$ is calculated as

$$s(x) = w^T \phi(x) + b. \tag{3}$$

However, in many classification problems feature vectors belong to more than two different classes. One well-known strategy to deal with this case is to build $(n_L)(n_L - 1)/2$ binary classifiers, where n_L is the number of different labels in the training set. Each such classifier is trained on data belonging to two different categories L_i and L_j and returns the corresponding function $s_{ij}(x)$, where $i, j \in \{1, \ldots, n_L\}$ and $i \neq j$. Let us notice, that

$$s_{ij}(x) = -s_{ji}(x). \tag{4}$$

A new sample x goes through all functions $s_{ij}(x)$, and it is defined either belonging to the i-th ($s_{ij}(x) \geq 0$) or the j-th ($s_{ij}(x) < 0$) category. Finally, the category of x is defined as the one which collects the most votes [22].

C. Zero-sum Matrix Game

The normal form of two-person zero-sum game [20] is given by triplet (S^1, S^2, π), where S^1 and S^2 are sets of strategies available for the 1-st and 2-nd player correspondingly and π is a real-valued function $\pi : S^1 \times S^2 \to \mathbb{R}$ which associates the first player payoff (equal to the second player loss) with every pair of strategies. The goal of the first player is to select the strategy maximizing his payoff, whereas the goal of the second player is to select the strategy minimizing his loss.

When S^1 and S^2 are finite sets the function π can be represented as a payoff matrix $\pi = [\pi_{ij}]$, where the value π_{ij} is the first player gain (the second player loss) in the case when the first player selects the i-th strategy and the second player selects the j-th strategy. The point (i^*, j^*) is called the saddle point of the game and $\pi_{i^*j^*}$ is the value of the game if for any strategies i and j of the first and second player correspondingly the inequality $\pi_{ij^*} \leq \pi_{i^*j^*} \leq \pi_{i^*j}$ holds. The matrix game has a saddle point (i^*, j^*) if and only if $\min_j \max_i \pi_{ij} = \max_i \min_j \pi_{ij} = \pi_{i^*j^*}$. In this case, the matrix game has a solution in pure strategies.

In the case a game does not have a saddle point, there are two options: the best guaranteed result solution and solution in mixed strategies. The best guaranteed result solution is defined as $\arg\max_i \min_j \pi_{ij}$ and $\arg\min_j \max_i \pi_{ij}$ for the first and the second players respectively. The set of mixed strategies Q^l of the l-th player ($l \in \{1, 2\}$) may be represented as a set of probability vectors: $Q^l = \{q^l = (q^l_1, \ldots, q^l_m) : q^l_k \geq 0, \sum_{k=1}^{m^l} q_k = 1$, where q^l_k is a probability that the l-th player selects the k-th pure strategy and m_l is the number of such strategies available for the l-th player. For a two-person game with payoff matrix π and sets of mixed strategies Q^1 and Q^2, mixed strategies $q^{1*} \in Q^1$ and $q^{2*} \in Q^2$ are optimal mixed strategies if $(q^1)^T \pi q^{2*} \leq (q^{1*})^T \pi q^{2*} \leq (q^{1*})^T \pi q^2$ for all $q^1 \in Q^1$ and $q^2 \in Q^2$. The point (q^{1*}, q^{2*}) is called the saddle point for mixed strategies and the value $(q^{1*})^T \pi q^{2*}$ is called the value of the game in mixed strategies. It was proved that every matrix game has a solution in mixed strategies [20].

IV. Algorithm

The application of n-gram models returns high-dimensional feature vectors even for small values of n. To reduce time and computing resources when classifying those vectors, a dimensionality reduction technique is supposed to be employed. Despite the development of supervised dimensionality reduction methods [21], feature selection based on a genetic algorithm remains one of the most powerful means to escape from high dimensionality in a classification problem [18].

Let us denote the size of feature vectors obtained after applying n-gram models as N, where $N = N_{oc}$ or $N = N_b$. First, we choose the number N_f of the most essential features to be selected. This number should not be high to allow the classifiers work fast, but high enough to classify malware properly. After N_f is chosen, an initial population for GA is formed. Each individual in this population is a binary vector of length N in which one is placed in the i-th position if the i-th feature is selected. Zeros correspond to non-selected features. The initial population is constructed randomly with just the restriction that the number of features selected does not exceed N_f.

Recombination and crossover are used to generate new individuals. To perform recombination, one individual is randomly selected, and half of its values, which are ones, is changed to zeros. Further, taking into account that the total number of units can not exceed N_f, some zero values become units. A simple two-point crossover is performed next. Two individuals $I^1 = (I^1_1, \ldots, I^1_N)$ and $I^2 = (I^2_1, \ldots, I^2_N)$ are chosen randomly and act as parents. Then a number $k : 1 \leq k < N$ is picked randomly, and new individual I is formed as follows: $I = (I^1_1, \ldots, I^1_k, I^2_{k+1}, \ldots, I^2_N)$. If the number of units in the vector obtained is greater than N_f, several values become zeros.

After crossover and recombination have been performed, some individuals which correspond to highest values of fitness function are selected for the next generation. In this study, classification accuracy is chosen as the fitness function. Classification accuracy is calculated by using the k-fold validation approach. In k-fold cross-validation, the training set is randomly partitioned into k equal-size subsets. Of the k subsets, a single subset is retained as the validation data for testing the model and the remaining $k-1$ subsets are used as training data. The cross-validation process is then repeated k times, with each of the k subsets used exactly once as the validation data. Therefore, for features corresponding to the individual I, k classification accuracy values are calculated. The fitness function value $f(I)$ is then calculated as follows:

$$f(I) = \frac{\sum_{v=1}^{k} A_v}{k}, \tag{5}$$

where A_v is malware classification accuracy for the v-th subset of the training set when only features corresponding to the individual I are selected. The rest of the algorithm description is devoted to the malware classification scheme.

Let us assume that the training set consists of benign software executables and different types of malware which belong to n_a categories. After extracting byte or opcode sequences, applying an n-gram model and selecting features by using GA, the set of labeled vectors is obtained, where a label is equal to $a_l \in \{1, \ldots, n_a\}$ for a malware file and zero for a benign software. The aim is to build a model which is trained on the basis of this set and allows to detect malware executable files and define to which malware categories they belong to. For this purpose, we train $n_a(n_a+1)/2$ binary SVMs using the data belonging to two different categories. As described in the previous section, the SVM trained with the data from categories i and j returns the function s_{ij}.

Let us consider a new executable file which is supposed to be classified. After the n-gram model has been applied and the most essential features have been selected, we denote the resulting vector as x. For this vector, the matrix zero-sum game $(S^1, S^2, \pi(x))$ is constructed, the strategies corresponding to different SVM classifiers, i.e. $S^1 = S^2 = \{0, 1, \ldots, n_a\}$. The payoff matrix $\pi(x) = [\pi_{ij}(x)]$, $i, j \in \{0, 1, \ldots, n_a\}$ is calculated as follows:

$$\pi_{ij}(x) = \begin{cases} 1, \text{ if } i = j, \\ \min(1, s_{ij}(x)), \text{ if } i \neq j. \end{cases} \tag{6}$$

According to the statement about the saddle point in a matrix game, the game $(S^1, S^2, \pi(x))$ has a solution in pure strategies (i^*, j^*) if and only if $\min_j \max_i \pi_{ij}(x) = \max_i \min_j \pi_{ij}(x) = \pi_{i^*j^*}(x)$, where $i, j \in \{0, 1, \ldots, n_a\}$. It is obvious, that $\min_j \max_i \pi_{ij}(x) = 1$, because $\pi_{ii}(x) = 1$ and $\pi_{ij}(x) \leq 1$, if $i \neq j$. On the other hand, $\forall i \ \min_j \pi_{ij}(x) \leq 1$ and consequently $\max_i \min_j \pi_{ij}(x) = 1$ only in the case when $\exists i^* : \ \min_j \pi_{i^*j}(x) = 1$ or equivalently $\exists i^* : \ \pi_{i^*j}(x) = 1, \ \forall j \in \{0, 1, \ldots, n_a\}$.

Let us also notice, that the game $(S^1, S^2, \pi(x))$ can have several saddle points, but in this case all of them are placed in the same row. Assume, that there are two saddle points (i_1, j_1) and (i_2, j_2), such that $i_1 \neq i_2$. As we proved earlier, it is required that $\pi_{i_1 j}(x) = 1$ and $\pi_{i_2 j}(x) = 1$, $\forall j \in \{0, 1, \ldots, n_a\}$. In particular, it means that $\pi_{i_1,i_2}(x) = 1$ and $\pi_{i_2,i_1}(x) = 1$, and consequently $s_{i_1 i_2}(x) > 0$ and $s_{i_2 i_1}(x) > 0$, which contradicts the definition of the SVM function $s_{ij}(x)$ in equation (4).

In terms of classification of new vector x, the existence of one or several saddle points means, that one class is winning, because all saddle points are located in the same game matrix row. Thus, if the game $(S^1, S^2, \pi(x))$ has a saddle point (i^*, j^*), we assign to x the label $a_l(x)$ which corresponds to the i^*-th row.

However, usually a matrix game can not be solved in pure strategies, i.e. it does not contain any saddle point. As proposed in the previous section, one variant to solve the game in this case is to find the best guaranteed result solution. For the game $(S^1, S^2, \pi(x))$, this solution can be defined as

$$\begin{aligned} \pi_{i^*\bar{j}}(x) &= \max_{i \in \{0,1,\ldots,n_a\}} \big(\min_{j \in \{0,1,\ldots,n_a\}} (\pi_{ij}(x)) \big), \\ \pi_{\bar{i}j^*}(x) &= \min_{j \in \{0,1,\ldots,n_a\}} \big(\max_{i \in \{0,1,\ldots,n_a\}} (\pi_{ij}(x)) \big), \end{aligned} \tag{7}$$

for the first and the second player correspondingly. This solution is equivalent to the solution that can be found with the help of fuzzy multi-class SVM [23]. Fuzzy SVM classifies the vector x as follows:

$$\begin{aligned} &i^* = argmax_{i \in \{0,1,\ldots,n_a\}} m_i(x), \\ &\text{where } m_i(x) = \min_{j \in \{0,1,\ldots,n_a\}, \ j \neq i} (1, s_{ij}(x)). \end{aligned} \tag{8}$$

As we can see, the label assigned to x by fuzzy SVM is the same as the label assigned by the solution which guarantees the best result for the first player in the game $(S^1, S^2, \pi(x))$, i.e.

$$i^* = argmax_{i \in \{0,1,\ldots,n_a\}} \big(\min_{j \in \{0,1,\ldots,n_a\}} (\pi_{ij}(x)) \big). \tag{9}$$

Nevertheless, in this study, we use another approach to classify vector x when the matrix game $(S^1, S^2, \pi(x))$ does not have any saddle point. This approach is based on the use of mixed strategies of players. Every matrix game has a solution in mixed strategies, For the game $(S^1, S^2, \pi(x))$ optimal mixed strategies $p^* = (p_0^*, p_1^*, \ldots, p_{n_a}^*)$ and $q^* = (q_0^*, q_1^*, \ldots, q_{n_a}^*)$ for the first and for the second player correspondingly can be found by solving the following two linear programming problems:

$$\max \upsilon$$
$$\text{subject to } \begin{cases} \upsilon \leq \sum_{i=0}^{n_a} p_i \pi_{ij}(x), \quad j \in \{0, 1, \ldots, n_a\}, \\ \sum_{i=0}^{n_a} p_i = 1, \\ p_i \geq 0, \quad i \in \{0, 1, \ldots, n_a\} \end{cases} \tag{10}$$

and

$$\min \omega$$
$$\text{subject to } \begin{cases} \omega \geq \sum_{j=0}^{n_a} \pi_{ij}(x) q_j, \quad i \in \{0, 1, \ldots, n_a\}, \\ \sum_{j=0}^{n_a} q_j = 1, \\ q_j \geq 0, \quad j \in \{0, 1, \ldots, n_a\}, \end{cases} \tag{11}$$

where υ and ω are auxiliary variables introduced to get rid of non-linearity of the problems objective functions. Linear programs (10) and (11) can be easily solved, e.g. by standard simplex method [24].

The vector x is classified based on optimal mixed strategies p^* and q^* as follows:

- If
$$\sum_{\substack{j=0 \\ j:s_{ij}(x)\geq 0}}^{n_a} p_0^* q_j^* + \sum_{\substack{i=1 \\ i:s_{ij}(x)\geq 0}}^{n_a} p_i^* q_0^* \geq \sum_{\substack{i,j=1 \\ i,j:s_{ij}(x)\geq 0}}^{n_a} p_i^* q_j^*, \tag{12}$$
then the executable file corresponding to the vector x is classified as a benign software.

- If the inequality (12) is not fulfilled, then the executable file corresponding to the vector x is classified as a malware, and the type of this malware $a_l(x)$ is defined as follows:
$$a_l(x) = \begin{cases} i^*, \text{ if } s_{i^*j^*}(x) \geq 0, \\ j^*, \text{ if } s_{i^*j^*}(x) < 0, \end{cases} \tag{13}$$
where
$$\begin{aligned} i^* &= argmax_{i\in\{1,\ldots,n_a\}}(p_i^*), \\ j^* &= argmax_{j\in\{1,\ldots,n_a\}}(q_j^*). \end{aligned} \tag{14}$$

If the game $(S^1, S^2, \pi(x))$ has several saddle points in mixed strategies, then we select one of them randomly. Let us also notice that the scheme described above is applied in the case when the game $(S^1, S^2, \pi(x))$ can not be solved in pure strategies.

V. Numerical Results

We tested the algorithm proposed using opcode and byte sequences extracted from real executable files, some of which are infected with malware. Each malware belongs to one of twenty different categories. The set of files is divided into the training set (600 entries) and the testing set (489 entries). We assume that the testing set does not contain malware types not presented in the training set. The extraction of features from opcode and byte sequences is carried out by employing 1-gram and 2-gram models. For 2-gram models, GA is used to reduce the dimensionality to less than 1000. For GA, 500 generations are used, the size of each population being equal to 100. For binary SVM classifiers, linear and Gaussian kernels are used, and optimal classifiers parameters are defined with the help of the k-fold validation technique.

To evaluate the performance of the proposed technique, the following characteristics are calculated in our test:

- True positive rate: the ratio of the number of correctly detected malware to the total number of malware in the testing set;
- False positive rate: the ratio of the number of normal files classified as malware to the total number of normal files in the testing set;
- Detection accuracy: the ratio of the total number of normal files detected as normal and malware detected as malware to the total number of files in the testing set;
- Classification accuracy: the ratio of the total number of normal files detected as normal and malware of a category classified as malware of this category to the total number of files in the testing set;

The dependence between false positive and true positive rates for different n-gram models applied to opcode and byte sequences is shown in Figure 1. Detection and classification accuracies for different Gaussian kernel parameter values used in SVMs are presented in Figure 2. As one can see, 2-gram models are much more accurate. In addition, the proposed algorithm applied to byte sequences shows better results in terms of the true positive rate, while in case of 1-gram model opcode sequences allow to obtain fewer false alarms.

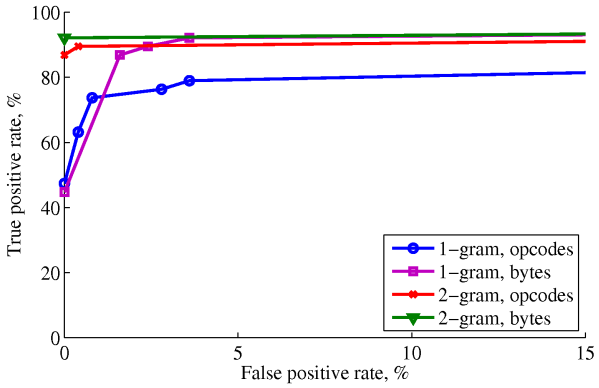
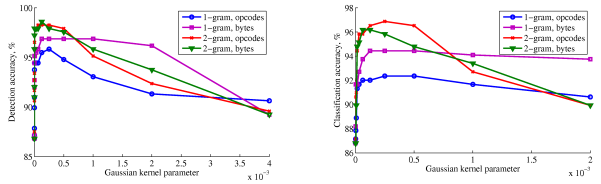


Fig. 1. Dependence between false positive rate and true positive rate for different n-gram models applied to opcode and byte sequences



(a) Detection accuracy (b) Classification accuracy

Fig. 2. Dependence of the accuracy on Gaussian kernel parameter for different n-gram models applied to opcode and byte sequences.

We compared the performance of the algorithm proposed with well-known classifying techniques: Artificial Neural Network (ANN), Data Tree (DT), K-Nearest Neighbors (KNN), Semi-supervised Density-Based Spatial Clustering of Applications with Noise (SSDBSCAN) [25], major voting multi-class SVM (MVSVM) [22] and fuzzy multi-class SVM (FSVM) [23]. In order to extract features 2-gram model is applied to opcode and byte sequences. To escape from the high dimensionality of the problem, several dimensionality reduction techniques were applied : Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA) plus Neighborhood Components Analysis (NCA) and Large Margin Nearest Neighbor metric learning (LMNN) and RELIEF [21]. Comparison results based on the analysis of opcode and byte sequences are listed, correspondingly, in Tables I and II, where malware detection and classification accuracies are shown. All optimal classifiers parameters are found with the help of k-fold validation during the training stage, and accuracy is calculated when applying those classifiers to the testing set. As one can notice, SVM

integrated with ZSG (ZSGSVM) and GA outperforms all other techniques in terms of the classification accuracy.

TABLE I. MALWARE DETECTION AND CLASSIFICATION ACCURACY (IN BRACKETS) OF THE ALGORITHM PROPOSED COMPARED TO ANALOGUES (IN PERCENT) BASED ON THE OPCODE SEQUENCES ANALYSIS.

Algorithm	LDA	PCA+NCA	PCA+LMNN	RELIEF	GA
ANN	89.58 (85.76)	92.01 (89.93)	86.00 (82.25)	94.44 (87.15)	88.89 (85.42)
DT	93.06 (91.67)	89.24 (88.54)	81.94 (78.13)	96.88 (93.07)	93.40 (89.24)
KNN	94.79 (92.01)	94.79 (92.71)	93.06 (90.63)	97.22 (95.14)	94.10 (92.36)
SSDBSCAN	92.36 (92.01)	95.83 (92.36)	94.10 (90.63)	91.32 (90.63)	92.71 (92.01)
MVSVM	93.06 (87.15)	94.79 (93.06)	94.44 (93.06)	96.88 (94.44)	**97.57** (95.83)
FSVM	93.06 (87.15)	94.79 (93.06)	94.44 (93.06)	96.88 (94.44)	**97.57** (95.83)
ZSGSVM	93.06 (85.76)	94.79 (93.06)	94.44 (93.06)	96.88 (95.14)	**97.57** (**96.18**)

TABLE II. MALWARE DETECTION AND CLASSIFICATION ACCURACY (IN BRACKETS) OF THE ALGORITHM PROPOSED COMPARED TO ANALOGUES (IN PERCENT) BASED ON THE BYTE SEQUENCES ANALYSIS.

Algorithm	LDA	PCA+NCA	PCA+LMNN	RELIEF	GA
ANN	92.71 (86.46)	89.58 (85.76)	85.50 (83.50)	88.19 (84.38)	88.89 (87.50)
DT	95.14 (91.67)	91.32 (90.28)	86.46 (83.68)	96.18 (94.10)	94.44 (91.32)
KNN	95.49 (94.79)	95.14 (93.75)	95.83 (94.79)	96.53 (95.14)	96.53 (94.44)
SSDBSCAN	93.40 (89.93)	94.10 (91.67)	93.75 (91.67)	92.36 (91.32)	93.75 (91.67)
MVSVM	96.86 (95.14)	**98.26** (94.79)	89.93 (88.89)	95.49 (94.79)	**98.26** (95.83)
FSVM	96.86 (95.14)	**98.26** (94.79)	89.93 (88.89)	95.49 (94.79)	**98.26** (95.83)
ZSGSVM	96.86 (95.14)	**98.26** (95.49)	89.93 (89.24)	95.49 (94.79)	**98.26** (**96.18**)

VI. CONCLUSION

In this research, we detect and classify malware with the help of a supervised machine-learning approach. Files of the training set are presented in the form of byte and opcode sequences and n-gram models are employed to extract data from these sequences. A genetic algorithm is used to select the most essential features and, therefore, to cope with the high dimensionality of the problem. A classification model based on binary support vector machines is built using feature vectors obtained. Then binary SVM classifiers are combined by using the game theory approach. Numerical examples carried out show that the algorithm proposed produces good results in terms of malware detection and classification accuracy.

Although ZSGSVM shows good results, feature selection with GA takes long time. We are planning to continue our research with supervised malware detection and design a feature selection technique that would be comparable to GA in terms of accuracy but which would work faster. In addition, we are going to employ anomaly detection approach to detect malicious software executables unseen previously.

REFERENCES

[1] House of Commons - Science and Technology Committee. Malware and Cyber Crime. 12th Report of Session 2010-12. TSO (The Stationery Office), 2012.

[2] R. Islam, R. Tian, L. Batten, S. Versteeg. Classification of Malware Based on String and Function Feature Selection. Proc. of Cybercrime and Trustworthy Computing Workshop (CTC), pp. 9-17, 2010.

[3] N. Kuzurin, A. Shokurov, N. Varnovsky, V. Zakharov. On the Concept of Software Obfuscation in Computer Security, Lecture notes in computer science, 2007.

[4] D. Bruschi, L. Martignoni, M. Monga, Detecting self-mutating malware using control-flow graph matching, Lecture Notes in Computer Science, 2006.

[5] M. Chouchane, A. Lakhotia. Using engine signature to detect metamorphic malware. Proc. of the 2006 ACM workshop on Recurring malcode, ACM New York, NY, USA, pp. 73-78, 2006.

[6] Q. Zhang, D. Reeves. Metaaware: Identifying metamorphic malware. Proc. of the 2007 Annual Computer Security Applications Conference (ACSAC), pp. 411-420, 2007.

[7] S. Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. Proc. of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, pp. 3-24, 2007.

[8] S. Kotsiantis, P. Pintelas, Recent advances in clustering: A brief survey, WSEAS Transactions on Information Science and Applications 1, pp. 73-81, 2004.

[9] O. Chapelle, B. Sch lkopf, A. Zien, Semi-supervised learning, MIT Press, 2006.

[10] W. Li, K. Wang, S. Stolfo, B. Herzog. Fileprints: Identifying file types by n-gram analysis. Proc. of the IEEE Workshop on Information Assurance and Security, 2005.

[11] S. Yu, S. Zhou, L. Liu, R. Yang, J. Luo. Malware variants identification based on byte frequency. Proc. of Networks Security Wireless Communications and Trusted Computing (NSWCTC), vol. 2, pp. 32-35, 2010.

[12] D. Bilar, Opcodes as predictor for malware. International Journal of Electronic Security and Digital Forensics, pp. 156-168, 2007.

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

[14] R. Moskovitch, C. Feher, N. Tzachar, E. Berger, M. Gitelman, S. Dolev and Y. Elovici. Unknown Malcode Detection Using OPCODE Representation. Proc. of the 1-st European Conference on Intelligence and Security Informatics (EuroISI '08), 2008.

[15] Y. Zhong, H. Yamaki, H. Takakura. A Malware Classification Method Based on Similarity of Function Structure. In Proc. of the 12th International Symposium on Applications and the Internet, 2012.

[16] T. Hirsimaki, J. Pylkkonen, M. Kurimo. Importance of High-Order N-Gram Models in Morph-Based Speech Recognition. Audio, Speech, and Language Processing, IEEE Tran. Vol. 17, Is. 4. pp. 724-732. 2009.

[17] W. Punch, E. Goodman. Further research on feature selection and classification using genetic algorithms. Proc. of the Fifth International Conference on Genetic Algorithms (1993), pp. 557-564, 1993.

[18] T. Shon, Y. Kim, C. Lee, J. Moon. A Machine Learning framework for network anomaly detection using SVM and GA. Proc. from the Sixth Annual IEEE SMC, 2005.

[19] R. Sharapov, A. Lapshin. Convergence of genetic algorithms. Pattern Recognition and Image Analysis. Vol. 16, No. 3, pp. 392-397, 2006.

[20] G. Romp. Game Theory: Introduction and Applications. Oxford University Press, 1997.

[21] K. Kira, L. Rendell. The feature selection problem: traditional methods and new algorithm. Proc. of Conference on Artificial Intelligence, 1992.

[22] J. Friedman. Another approach to polychotomous classification. Dept. Statistics, Stanford Univ., Tech. Rep, 1996.

[23] S. Abe and T. Inoue. Fuzzy support vector machines for multiclass problems. Proc. of European Symposium on Artificial Neural Networks, pp. 113-118, 2002.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. Sec. 29.3: The simplex algorithm, pp. 790804.

[25] L. Lelis and J. Sander. Semi-supervised Density-Based Clustering. Proc. of the 2009 Ninth IEEE International Conference on Data Mining (ICDM '09), 2009.