Malware Detection Using Genetic Programming

Thi Anh Le
Faculty of IT,
Le Quy Don University
Hanoi, Vietnam
Email:leanh41@gmail.com

Thi Huong Chu
Faculty of IT,
Le Quy Don University
Hanoi, Vietnam
Email:huongktqs@gmail.com

Quang Uy Nguyen
Faculty of IT,
Le Quy Don University
Hanoi, Vietnam
Email: quanguyhn@gmail.com

Xuan Hoai Nguyen
IT Centre,
Hanoi University
Hanoi, Vietnam
Email: nxhoai@gmail.com

***Abstract*—Malware is any software aiming to disrupt computer operation. Malware is also used to gather sensitive information or gain access to private computer systems. This is widely seen as one of the major threats to computer systems nowadays. Traditionally, anti-malware software is based on a signature detection system which keeps updating from the Internet malware database and thus keeping track of known malwares. While this method may be very accurate to detect previously known malwares, it is unable to detect unknown malicious codes. Recently, several machine learning methods have been used for malware detection, achieving remarkable success. In this paper, we propose a method in this strand by using Genetic Programming for detecting malwares. The experiments were conducted with the malwares collected from an updated malware database on the Internet and the results show that Genetic Programming, compared to some other well-known machine learning methods, can produce the best results on both balanced and imbalanced datasets.**

I. Introduction

Genetic Programming (GP) [11, 20] is an evolutionary algorithm aimed to provide solutions to an user-defined task in the form of computer programs. Since its introduction, GP has been applied to many practical problems [20], producing a number of human competitive results [12]. GP has been used as a learning tool for solving some problems in network security [2, 13, 19, 26]. Previous research on the application of GP to network security mainly focused on the intrusion detection problem, in which the intrusion data from KDD99 dataset was used as the benchmark [2, 13, 26]. Recently, GP has also been used to help the detection of phishing websites [19]. However, to the best of our knowledge, there has not been any published work on the use of GP for learning to detect malicious software.

In the field of network security, malware attacks is one of the main threats over the past several decades. According to a report released by an anti-viruses corporation, Kaspersky, there was around 315,000 new malicious files detected every day in 2013 [1]. Malware can obstruct the operation of computer systems, steal sensitive information or even control behavior of user's computers. Malware attacks cause significant financial loss every year. In 2006, malicious softwares lead to a loss of $ 13.3 Billion globally [2]. Therefore, fighting against malware attacks is of great importance for both computer users and enterprises. Typically, anti-malware softwares are often based on a signature definition system. In this, a bit-string of malicious code is stored and used to check if a suspect file is infected by a matching metric. While this signature-based matching method is very accurate to detect already known attacks, it is unable to recognize novel malwares.

In order to compensate for signature-based detection, some methods such as heuristics-based and behavior-based techniques have been proposed to enhance the ability of anti-malware softwares in detecting unknown malwares. Particularly, machine learning methods have been successfully applied to malware detection problems recently [7]. In this paper, we continue in this strand by applying GP to malware detection. We collected a number of malicious codes released recently from VX Heaven website [3]. We used some feature extraction techniques to select important features from the malicious files. Then GP was used to build up the model for distinguishing between infected and normal files. The results show the capability of GP in producing good models for malware dection.

The rest of the paper is organised as follows. In the next section, we briefly review some previous research on detecting malwares using machine learning techniques. In Section III we present our method using GP for solving the malcodes detection problem. It is followed by a section detailing our experimental settings. The experimental results are shown and discussed in Section V. The last section concludes the paper and highlights some potential future work.

II. Related Work

Using machine learning methods for malware detection has received increasing attention recently [1, 10]. Various machine learning techniques have been proposed for this important task. These techniques can be applied in either static or dynamic approaches. While static approach attempts to analyse and distinguish malwares from benign files without executing them, dynamic approach analyses the behavior of a malicious code (interaction with the system) when it is being executed in a controlled environment (virtual machine, simulator, emulator, sandbox etc.). For a recent and comprehensive review on the application of machine learning methods to malware detection, we suggest the readers to see [7]. A few of the related research in the literature are discussed in this section.

Perhaps, Schultz et al. [15] were the first authors who advocated the static approach of using machine learning for malware detection. They extracted features using program header and using Naive Bayes [8] to learn the detection system. Their results showed that machine learning method was more

[1] http://www.kaspersky.com/about/news/virus/2013/number-of-the-year
[2] http://www.computereconomics.com/article.cfm?id=1225
[3] http://vxheavens.com/

978–1–4799-5431-5/14/$31.00

accurate than the signature-based algorithm. Abou-Assaleh et al. used a common n-gram method to select important features from malicious files [1]. After that, they use K-nearest neighbor algorithm [6] to classify between malicious and benign programs. This work was then extended by using multi-class classification techniques to classify malcodes into different families [10]. Their results showed that some types of malwares such as *mass mailer, backdoor* and *virus* can well be recognized.

After that, Moskovitch et al. [17] studied the impact of imbalanced data when applying machine learning to malcodes detection. They tested different datasets of various rate between malicious and benign files. Their experimental results showed that machine learning techniques such as neural networks, decision tree and support vector machines can produce good results even when the datasets are imbalanced. Santos et al. [25] pointed out that supervised learning often requires a significant amount of labeled executables for both classes (malicious as well as benign datasets) that are not always available. They proposed a semi-supervised learning approach for detecting unknown malwares. It is designed to build a machine learning classifier using a small labeled and a large unlabeled instances. Their method helped to reduce the number of required labeled instances while maintaining high precision.

Recently, some researchers used dynamic techniques to improve the accuracy and effectiveness of machine learning algorithms. Zolkipli et al. [16] presented an approach for malware behavior analysis. They collected malwares from the Internet by using HoneyClients. Then, the features of these malwares are identified by executing every sample on both CWSandbox [29] and on a virtual machine platform. The results generated by both of these analyzers are used to classify malwares into families of Worms and Trojans.

Rieck et al. [23] proposed a framework for automatic analysis of malware behavior using machine learning. A large number of malware samples was collected and their behaviors were monitored using a sandbox environment. By embedding the observed behaviors in a vector space, they could apply learning algorithms on this space. Clustering was used to identify the novel classes of malware with similar behavior. Assigning unknown malware to these discovered classes was done by classification. The results showed that this approach was capable of processing the behavior of thousands of malware binaries on daily basis.

Tian et al. [28] extracted API call sequences from executables while these are running in a virtual environment through an automated tool. These API call sequences are used as feature vector for machine learning methods available in the WEKA library. They used a dataset of 1368 malwares and 456 benign files to demonstrate their work. Their results showed that the algorithms in Weka can discriminate malware files from clean files with an accuracy of about 95%.

Although dynamic approach has an advantage over static approach in which the number of extracted features in dynamic approach is often smaller. The downside is that the malwares must be executed before they can be detected. In order to achieve this, a sandbox or a virtual environment are often used to execute malcodes. However, this may slow down the detection process. Moreover, the accuracy of dynamic approach is not always as high as statistic approach [3]. Subsequently, researches have adapted a hybrid technique which incorporates both static and dynamic features simultaneously for better malware detection and classification [9, 24]. In this paper, we only used static features in the experiments. Extracting dynamic and hybrid features are left for future research.

III. Methods

This section presents the methods used in this paper. First, the way to extract the features from malicious files is presented. After that, the GP system for malware detection is described.

A. Features Extraction

The first step for using GP to tackle the malware detection problem is features extraction. This is a very important step as it may strongly affect the effectiveness of the learning algorithm (GP). The extracted features must contain information that helps to distinguish malicious and legitimate files. In this paper, several methods based on n-gram are used for features extraction. They are detailed as follows.

First, all malicious files are converted to the hexa format. Then 4-gram is used as terms in each file [4]. Here, an n-gram is a contiguous sequence of n items from a given sequence of text. After that, three methods are used to extract feature vectors for each malcode. The first method for feature extraction is called term frequency (TF). Formula 1 shows the definition of a normalized TF, in which the term frequency is divided by frequency of the most frequent term in the document to achieve a value in the range of [0-1].

$$TF = \frac{\textit{term frequency}}{Max(\textit{term frequency in malcode})} \tag{1}$$

The second method is an extension of term frequency. This method is called term frequency inverse document frequency (TFIDF), which combines the frequency of a term in the document (TF) and its frequency in the documents collection, denoted by Document Frequency (DF). Formally, TFIDF is defined as in formula 2.

$$TFIDF = TF * Log(\frac{N}{DF}) \tag{2}$$

where N is the number of documents in the entire file collection and DF is the number of files in which the term appears.

The last method for feature extraction is a statical measure: Fisher Score (FS). The Fisher score ranking technique calculates the difference, described in terms of mean and standard deviation, between the positive and negative examples relative to a certain feature. Formula 3 defines the Fisher score, in which FS_i is the rank of feature i. $\mu_{i,p}$ and $\mu_{i,n}$ are the mean of term frequency of a feature in positive and negative samples. Similarly, $\delta_{i,p}$ and $\delta_{i,n}$ are the standard deviation of this term.

$$FS_i = \frac{\mu_{i,p} - \mu_{i,n}}{\delta_{i,p} + \delta_{i,n}} \tag{3}$$

[4] 4-gram has been shown for good performance of machine learning algorithms in detection malcodes [17]

The bigger the FS_i, the bigger the difference between the values of positive and negative examples relative to feature i. Thus, this feature is more important for separating the positive and negative examples. This technique is described in details in [14]. Based on each feature selection measure we selected the top 50, 100 and 200 features to build the dataset for learning systems.

B. System Description

The evolutionary learning process of GP for solving the problem of malware detection is divided into two stages: training and testing. The objective of training stage is to evolve the model (the classifier) that can determine a file as either infected or legitimate based on its feature values. In the testing stage, the learnt model is used to make predictions on the unseen data. The accuracy of these prediction is used as an indicator for the quality (effectiveness) of the model.

In the training stage, a set of training data (both malicious and benign) with their labels (either as malcode or normal) are provided. The feature extraction process is called to convert every files to a feature vector. This vector is then used as the input for an individual in GP and the output of the individual is a real value. If this real value is greater than zero, this file is tagged as a malcode, otherwise it is considered as benign.

The next step in the training process is to measure the fitness of an individual in GP. In this paper, we use a simple way to measure the fitness of individuals where the fitness is the percentage of files in the training set that are correctly classified. This is defined as in the following equation.

$$Fitness = \frac{TP + TN}{\textit{Total number of files}} \quad (4)$$

where TP (true positive) and TN (true negative) are the number of malcodes and benign files that are correctly distinguished. Although this fitness may not be a good indicator if the data is very much imbalanced, it is intuitive to identify the overall quality of a model.

IV. Experimental Settings

This section outlines the settings used in our experiments. First, we present the way that data was collected for training and testing the systems. After that GP configurations for the experiments are described.

A. Data Collection

We created two datasets of malicious and benign executables for the Windows operating system, as its popularity makes it the most commonly attacked. We acquired the malicious files from the VX Heaven website [5]. This site contains a massive, continuously updated collection of all types of malwares. The dataset of 800 malicious files were collected from this database. In order to guarantee that these are malcodes, we used the Kaspersky anti-virus to test these files. Next, the normal files were collected from our computer systems at the network security Laboratory, Le Quy Don University. Totally, 1600 benign files were obtained and the Kaspersky anti-virus program was also used to verify that these files do not contain any malicious code.

[5]http://vxheavens.com/

TABLE I. Run and Evolutionary Parameter Values.

Parameter	Value
Population size	500
Generations	50
Selection	Tournament
Tournament size	7
Crossover probability	0.9
Mutation probability	0.1
Initial Max depth	6
Max depth	17
Max depth of mutation tree	5
Non-terminals	+, -, *, / (protected version), sin, cos, exp, iff, log (protected version)
Terminals	X1, X2,..., XN,
Raw fitness	percentage of correct classification
Trials per treatment	100 independent runs for each value

After that, two experiments were conducted. The first experiment was set up with the balanced data. In this experiment, 750 (out of 800) malicious files and 800 (out of 1600) benign files were used. The second experiment aims to investigate the performance of GP when the dataset is imbalanced. In this experiment, 800 infected files and 1600 legitimate files were used. Each dataset was then divided into two parts (one for training and one for testing) of the equal size.

B. GP Parameters Settings

To tackle a problem with GP, several elements need to be clarified beforehand. These elements often depend on the problem and the experience of practitioners. The first and important element is the fitness function. As aforementioned, in this paper we use the percentage of correct classifications as the fitness measurement for each individual in the population.

Other factors that strongly affect the performance of GP are the set of non-terminals and terminals. The terminal set includes N variables (X_1, X_2,..., X_N) representing N features extracted from the files. The non-terminal set includes 5 functions (+, -, *, /, iff). Here, we used the protected versions of division (/), meaning that if the denominator is zero, the returned value is 1. Other evolutionary parameters are presented in Table I. These are typical values that are often used by GP researchers and GP practitioners [11].

V. Results and Discussion

To determine quality of the models produced by GP, at the end of each run, we selected the best-of-the-run individual (the individual with the best fitness on the training set in the entire run). This model is then tested on the testing set and the output on the testing set is considered as the prediction error of the model.

We compare the results of GP with the results produced by a number of other machine learning techniques such as Support Vector Machines, Bayesian Networks, Artificial Neural Networks and Decision Trees. They are detailed as follows.

Support Vector Machines: Support Vector Machines (SVM) are a relatively new learning method used for binary classification [4]. The basic idea is to find a hyperplane which

separates the d-dimensional data perfectly into its two classes. However, since example data is often not linearly separable, SVM introduces the notion of a kernel induced feature space which casts the data into a higher dimensional space where the data is separable. Overall, SVM is intuitive, theoretically well-founded, and have shown to be practically successful in solving a large number of problems. In this paper, we will use SVM with the kernel function as the Gaussian function to tackle phishing detection problem.

Artificial Neural Network: An Artificial Neural Networks (ANN) is an information processing paradigm inspired by the way that the biological nervous systems processes information [5]. The important element of this paradigm is the structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working together in order to approximate a specific function. Similar to human beings, ANNs learn by example. ANNs have been successfully applied to many real-world applications such as data classification or pattern recognition. There are several different kinds of neural network of these RBFNetwork (Radial Basis Function Network) and Multilayer Perceptron [30] are perhaps the most popular, therefore they were used in the experiments in this paper.

Bayesian networks: Bayesian networks (BNs) consist of nodes and arcs that present for random variables and connections between them, respectively [8]. BNs can often be seen as probabilistic graphical models used to reason under uncertainty. When constructing the network, two main components namely estimator and searching algorithm need to be identified. Estimator is a function used for evaluating a given network, and searching algorithm is once that searching through the space of possible networks. In our experiments, we used SimpleEstimator algorithm for estimator, and K2 for the searching algorithm [30].

Decision trees: Decision tree learners are a well-established family of learning algorithms for classification [21]. In contrast to other black-box learning methods, decision trees represent rules. This allows the solutions obtained by decision trees can easily be understood and analysed. There have been a number of algorithms developed for constructing decision trees for a problem. Among them, C4.5 is the most popular algorithm [22]. In this paper, we use an extension of C4.5 called J48 for constructing the decision tree for the problem. J48 has been implemented in Weka [30].

In order to apply these machine learning techniques to solve the malware detection identification, we used their implementations in Weka [30]. All parameters were tuned using the method described in [30]. We compare the best results produced by these methods with the best results obtained by GP in two experimental sets (balance and imbalance) and they are detailed below.

A. Results on Balance Data

The percentage of correct prediction of GP and other machine learning methods on the balance dataset is presented in Table II. In this table (and also in the following table), "GP" is the results produced by genetic programming. "Bayes" presents for the results obtained by Bayesian network. "RBF" and "Percentron" are shorthanded for two types of neural networks: Multilayer Perceptron network and Radial Basis Function Network. Finally, "J48" and "SVM" are the results achieved by decision tree trained by J48 algorithm and support vector machine. The column "Features" presents the number of features (50, 100 and 200 respectively) used in the experiments. The first row presents for different features extraction methods. It should be noted that in all tables, the greater values are the better and the best result in each set of experiments is printed bold faced.

TABLE II. COMPARISON GP WITH OTHER METHODS ON BALANCE DATA

Features	Methods	TF	TFIDF	Fisher Score
50	Bayes	0.663	**0.925**	0.621
	RBF	0.631	0.921	0.556
	Perc	0.657	0.628	0.608
	SVM	0.626	0.868	0.589
	J48	0.48	0.851	0.561
	GP	**0.707**	0.921	**0.875**
100	Bayes	0.649	0.853	0.621
	RBF	0.668	0.905	0.42
	Perc	0.701	0.559	0.541
	SVM	0.623	0.880	0.540
	J48	0.487	0.849	0.561
	GP	**0.741**	**0.935**	**0.839**
200	Bayes	0.625	0.892	0.633
	RBF	0.614	0.933	0.413
	Perc	0.58	0.525	0.559
	SVM	0.576	0.892	0.581
	J48	0.463	0.849	0.461
	GP	**0.731**	**0.937**	**0.844**

It can be seen from this table that the best model produced by GP is often the best among all models obtained by all tested machine learning systems. Overall, the prediction accuracy of GP learnt model is from 70% to 94%. The lowest value was produced on data set of 50 features with term frequency (TF) is used as the features extraction technique and the highest value was obtained when TFIDF was extracted and the number of features was 200. These values of other methods often much smaller than those of GP except when ITIDF features extraction was used. In this case, all other learning systems but Percetron neural network (perc) achieved rather good results. Particularly, Bayes Network was better than GP in one configuration when 50 features of TFIDT was used.

Comparing between different features extraction methods, it can be observed from this table that the best method is TFIDF while the worst is TF. This is understandable since TFIDF contains richer information for distinguishing between two sets of data [27]. Regarding to fisher score, the results show that this features selection is well suitable for GP but for other learning algorithms, it is not as good as TFIDF. In terms of number of used features, the table shows that the differences are negligible. The results confirm that all machine learning techniques performed mostly equally on three set of

TABLE III. COMPARISON GP WITH OTHER METHODS ON IMBALANCE DATA

Features	Methods	TF	TFIDF	Fisher Score
50	Bayes	0.794	**0.851**	0.799
	RBF	0.775	0.667	0.667
	Perc	0.719	0.611	0.665
	SVM	0.713	0.667	0.675
	J48	0.479	0.728	0.619
	GP	**0.811**	0.667	**0.931**
100	Bayes	0.745	**0.835**	0.806
	RBF	0.535	0.747	0.667
	Perc	0.575	0.634	0.641
	SVM	0.725	0.667	0.658
	J48	0.513	0.627	0.635
	GP	**0.803**	0.717	**0.928**
200	Bayes	0.666	0.797	0.806
	RBF	0.402	0.81	0.643
	Perc	0.409	0.699	0.568
	SVM	0.552	0.661	0.558
	J48	0.406	0.679	0.635
	GP	**0.792**	**0.854**	**0.938**

features (50, 100 and 200 features). In general, the results in this subsection evidence for the efficiency of GP when applying to malware detection problem.

B. Results on Imbalance Data

The percentage of correct prediction of GP and other machine learning methods on the imbalance data set is presented in Table III. It can be seen that the results in this table are consistent with the results in II in which GP often produced the best model amongst all tested learning systems. In most experiments, GP is the best algorithm in terms of producing the highest percent of correct prediction. This happens in 7 out of 9 experimental settings. Only in two settings of 50 and 100 features with TFIDF features selection, Bayes network outperforms GP and is the best method. Overall, The best solution obtained by GP in imbalanced data is as good as in the balance case. The correct prediction rate of the best model produced by GP in all experimental settings is about 94% (equal to the balanced case).

However, different from the balanced data where TFIDF is the best feature selection method, in the imbalanced data, Fisher Score is often better than both TF and TFIDF. In imbalance data, while TFIDF is less effective, Fisher Score is much better than TFIDF. This is reflected by the smaller values of the best solutions found by all learning methods with TFIDF and the greater values with Fisher Score. Comparing between different number of extracted features, it can be observed that they are mostly equal. This means that 50 features seem to be enough for all learning methods in solving the malware detection problem. Generally, the results in this section present the good performance of GP compared to other tested learning techniques in solving malware detection problem. These results also showed that TFIDF is good feature selection in balanced data whereas Fisher Score is more suitable for imbalanced case. Moreover, these results seem universal for all number of extracted features (50, 100 and 200).

VI. CONCLUSION

In this paper, we conducted a first comprehensive investigation on the use of Genetic Programming (GP) for solving the problem of detecting malwares. We collected a number of malwares from the malware database on the Internet. We used three different methods to extract the important features from these malicious files. Two data set of balanced and imbalanced were set up. For each dataset, nine experimental settings were implemented and executed. The results produced by GP were compared with the results obtained by four other machine learning techniques (Support Vector Machine, Artificial Neural Networks, Bayesian Networks and Decision Trees). The results showed that GP is capable of producing the prediction models (classifiers) that are more accurate than other machine learning techniques on most of experimental settings. This result inspires us to get GP integrated with signature-based anti-malwares to improve their ability in detecting unknown malwares.

In future, we are planning to extend the work in this paper in a number of ways. First, we would like to enrich data set with more malcodes and benign files to see if the performance of GP and other learning methods are maintained. Second, we want to give GP more computational time (by increasing the population size and number of generations) to see if it can help GP to find better models. Third, we want to use some recent advanced techniques, especially some technique for improving the generalization ability of GP [18] to see if they can further enhance the results. Last but not least, we want to make a more thorough analysis on the obtained models to get better understanding of the factors that affect the prediction accuracy.

ACKNOWLEDGMENT

The work in this paper was funded by The Vietnam National Foundation for Science and Technology Development (NAFOSTED), under grant number 102.01-2014.09. The Network Security Lab of Le Quy Don University provided research facilities for this study. The first author would like to thank Mr. Tuan Anh Pham for his help in this research.

REFERENCES

[1] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. N-gram-based detection of new malicious code. In *COMPSAC*, pages 41–42. IEEE Computer Society, 2004.

[2] J. Blasco, A. Orfila, and A. Ribagorda. Improving network intrusion detection by means of domain-aware genetic programming. In *International Conference on Availability, Reliability, and Security, ARES '10*, pages 327–332, Feb. 2010.

[3] F. I. L. C., E. A., and N. A.S. Analysis of machine learning techniques used in behavior-based malware detection. In *2010 Second International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT)*, pages 201–203. IEEE, 2010.

[4] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines*. Cambridge University Press, Mar. 2000.

[5] S. Das. Elements of artificial neural networks. *IEEE Transactions on Neural Networks*, 9(1):234–235, Jan. 1998.

[6] B. V. Dasarathy, editor. *Nearest Neighbor Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[7] E. Gandotra, D. Bansal, and S. Sofat. Malware analysis and classification: A survey. *Journal of Information Security*, 3(5):56–64, 2014.

[8] D. Heckerman. Tutorial on learning in bayesian networks. Technical Report MSR-TR-95-06, Microsoft, 1995.

[9] R. Islam, R. Tian, L. M. Batten, and S. Versteeg. Classification of malware based on integrated static and dynamic features. *J. Network and Computer Applications*, 36(2):646–656, 2013.

[10] J. Z. Kolter and M. A. Maloof. Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 6:2721–2744, 2006.

[11] J. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press, MA, 1992.

[12] J. Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):251–284–, 2010.

[13] S. Mabu, C. Chen, N. Lu, K. Shimada, and K. Hirasawa. An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 41(1):130–139, Jan. 2011.

[14] W. Malina. On an extended fisher criterion for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell*, 3(5):611–614, 1981.

[15] E. Z. S. J. S. Matthew G. Schultz, Eleazar Eskin. Data mining methods for detection of new malicious executables. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 178–184. ACM, 2001.

[16] Z. M.F. and J. A. An approach for malware behavior identification and classification. In *2011 3rd International Conference on Computer Research and Development*, pages 191–194. IEEE, 2011.

[17] R. Moskovitch, D. Stopel, C. Feher, N. Nissim, N. Japkowicz, and Y. Elovici. Unknown malcode detection and the imbalance problem. *Journal in Computer Virology*, 5(4):295–308, 2009.

[18] Q. U. Nguyen, T. H. Nguyen, X. H. Nguyen, and M. O'Neill. Improving the generalisation ability of genetic programming with semantic similarity based crossover. In A. I. Esparcia-Alcazar, A. Ekart, S. Silva, S. Dignum, and S. Uyar, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 184–195, Istanbul, 7-9 Apr. 2010. Springer.

[19] T. A. Pham, Q. U. Nguyen, and X. H. Nguyen. Phishing attacks detection using genetic programming. In V.-N. Huynh, T. Denoeux, D. H. Tran, A.-C. Le, and S. B. Pham, editors, *Proceedings of the Fifth International Conference on Knowledge and Systems Engineering, KSE 2013, Volume 2*, volume 245 of *Advances in Intelligent Systems and Computing*, pages 185–195, Hanoi, Vietnam, 17-19 Oct. 2013. Springer.

[20] R. Poli, W. Langdonand, and N. McPhee. *A Field Guide to Genetic Programming*. http://lulu.com, 2008.

[21] Quinlan. Learning decision tree classifiers. *CSURV: Computing Surveys*, 28, 1996.

[22] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[23] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668, 2011.

[24] I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas. OPEM: A static-dynamic approach for machine-learning-based malware detection. In *SOCO Special Sessions*, volume 189, pages 271–280. Springer, 2012.

[25] I. Santos, J. Nieves, and P. G. Bringas. Semi-supervised learning for unknown malware detection. In *International Symposium on Distributed Computing and Artificial Intelligence, DCAI 2011, Salamanca, Spain, 6-8 April 2011*, volume 91 of *Advances in Soft Computing*, pages 415–422. Springer, 2011.

[26] S. Sen and J. A. Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks. In *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, pages 95–102, Zurich, Switzerland, Mar. 16-19 2009. ACM.

[27] P. Soucy and G. Mineau. Beyond TFIDF weighting for text categorization in the vector space model. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1130–1135, Edinburgh, Scotand, Aug. 2005.

[28] R. Tian, R. Islam, L. M. Batten, and S. Versteeg. Differentiating malware from cleanware using behavioural analysis. pages 23–30. IEEE, 2010.

[29] C. Willems, T. Holz, and F. C. Freiling. Toward automated dynamic malware analysis using CWSandbox. *IEEE Security & Privacy*, 5(2):32–39, 2007.

[30] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.