International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(3): 694-713
The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085)

Optimizing Decision Tree in Malware Classification System by using Genetic Algorithm

Mohd Najwadi Yusoff and Aman Jantan
School of Computer Science,
Universiti Sains Malaysia,
Penang, Malaysia.
{najwadi,aman}@cs.usm.my

ABSTRACT

Malware classification is a vital component and works together with malware identification to prepare the right and effective malware antidote. Current techniques in malware classification do not give a good classification result while dealing with new as well as unique types of malware. In general, these kinds of malware are highly specialized and very difficult to classify. Therefore, this paper proposed the usage of Genetic Algorithm (GA) as an approach to optimize Decision Tree (DT) in malware classification. GA is chosen because unique types of malware are basically functioning like crossover and permutation operations in GA. New classifier is developed by combining GA with DT that we called as Anti-Malware System (AMS) Classifier. Experimental results obtained from AMS Classifier and DT are compared and visualized in tables and graphs. AMS Classifier shows an accuracy increase from 4.5% to 6.5% from DT Classifier. Outcome from this paper is a new Anti-Malware Classification System (AMCS) consists of AMS Classifier and new malware classes that we named as Class Target Operation (CTO). Malware is classified by using CTO which are mainly based on malware target and its operation behavior.

KEYWORDS

Malware classification, Genetic Algorithm, optimization, unique malware

1 INTRODUCTION

Malware classification is one of the main components in malware detection mechanism. It is used to classify the malware into its designated classes. In malware classification system, the main appliance or engine is named as classifier. According to Gheorghescu, malware classification task by using machine learning techniques is commercially applied in many anti-malware products [1]. As stated by Rieck, malware classification system is necessary and highly important when combating the malware [2]. It is because, malware classification system work together with malware identification process to produce the right and effective malware antidote.

The work by Aycock and Filiol classified malware as Virus, Worms, Trojan Horses, Logical Bombs, Backdoor, Spyware, Exploit and Rootkit [3-4]. According to Filiol, conventional malware classes are mainly based on malware specific objective. For example, malware in worm's classes use the network system to send a copy of itself to other computer to spread but do not attempt to alter the target system. Therefore, by looking at the malware classes and their objectives, the right and effective malware antidote can be produced, thus will greatly help anti-

malware products to act in preventing the malware from affecting the system.

On another issue, Apel reported that the widespread of malware has increased dramatically due to the deployment of avoidance techniques by malware writers [5]. Avoidance techniques are progressively being used by malware writers to avoid detection and analysis of their malicious code by anti-malware products. According to Preda, malware writers used avoidance techniques to change the malware syntax or signature but not their behavior of attacks which has to be maintained [6]. In general, the common avoidance technique used by malware writers is called code obfuscation. There are two types of code obfuscation which are polymorphism and metamorphism technique. By using this technique, Szor showed that many new variants of polymorphic and metamorphic malware can easily be created by encrypting the code, flow transposition, substitution and renaming variable [7]. The other techniques to avoid detection are packing, anti-debugging and anti-virtualization.

The work by Noreen shows that, the design of malware and its intention has become more sophisticated and significantly improved [8]. Generally, current machine learning classifiers used in malware classification system such as Naïve Bayes (NB), Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Decision Tree (DT) does not give a good classification result when it deals with new as well as unique types of malware. As reported by Martignoni, this is due to the highly specialized of the malware and the difficulty to analyze them [9]. New as well as unique types of malware are no longer able to be classified easily to the conventional malware classes. According to Martignoni and Bayer, these variants of malware have numerous attributes and combination syntax but show the same types of behavior [9-10]. Thus, classification of malware has become more complicated and new malware classification system is urgently needed.

This paper proposed the usage of Genetic Algorithm (GA) to optimize the current machine learning classifier which is DT. GA is a heuristic search that simulates the process of natural evolution. According to Mehdi, this heuristic algorithm is regularly used to generate useful solutions in optimization and search problems [11]. Malware created by avoidance techniques and having almost the same functionality as well as showing the same types of behavior can be classified by using GA. It is because these types of malware basically functioning like crossover and permutation operations in GA [12]. New classifier is developed by combining GA with DT that we call as Anti-Malware System (AMS) Classifier. As stated by Edge, GA has an ability to be a learning algorithm and can undergo self-learning [13]. As a result, it will make the new classification system more reliable than the current classification systems.

2 STATE OF THE ART

Malicious code or malware has long been recognized as the major threat to the computing world [10],[14]. According to Mohamad Fadli and Aman Jantan, all malware have their specific objective and target, but the main purpose is to create threats to the data network and computer operation [15]. In general, incoming files are considered as malware if they perform suspicious attack and might harm computer systems. These files can break into

vulnerable systems from many ways such as via internet (HTTP), email, removable media, Peer-2-Peer (P2P) and Instant Messaging (IM). Anti-malware products in host machine will perform scanning process to detect and identify the attack. If unknown kinds of attack are found, the samples will be sent to anti-malware vendors to be analyzed. Once analysis process is completed, vendors will update the virus signature database server and the latest update can be downloaded from host machine side. Classification part will classify the attack sample into the correct malware classes and prediction as well as removal part will remove, clean or quarantine the attacks.

2.1 Growth of Malware Attack

With the development of the underground economy, malware is considered as profitable product as the malware writers use it to spam, steal information, perform web frauds, and many other criminal tasks. According to Martignoni, malware has established during the past decade to become a major industry [9]. New malwares keep on increasing from time to time and this delinquent is seriously concerned by the security group around the world. For example, Panda Security Lab reported that, one third of existing computer malwares are created between January to October 2010. The exponential growth of malware is also reported by many other security groups such as F-Secure, Marcus, Kaspersky, Sophos, Symantec and G-Data [16-21]. Figure 1 shows Malware Evolution from 2003 to 2010 by Panda.

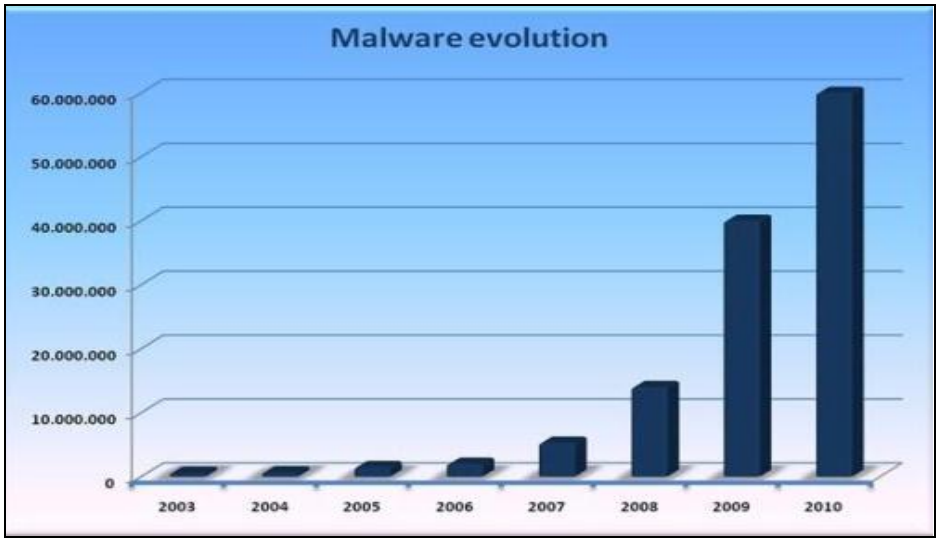

Figure 1. Malware Evolution [22].

According to ESET, more than half of number of the malware samples nowadays are classified as unique and created by assimilating existing malware with avoidance technique. Figure 2 is shows the total number of unique malware samples reported by ESET from 2005 to 2010.

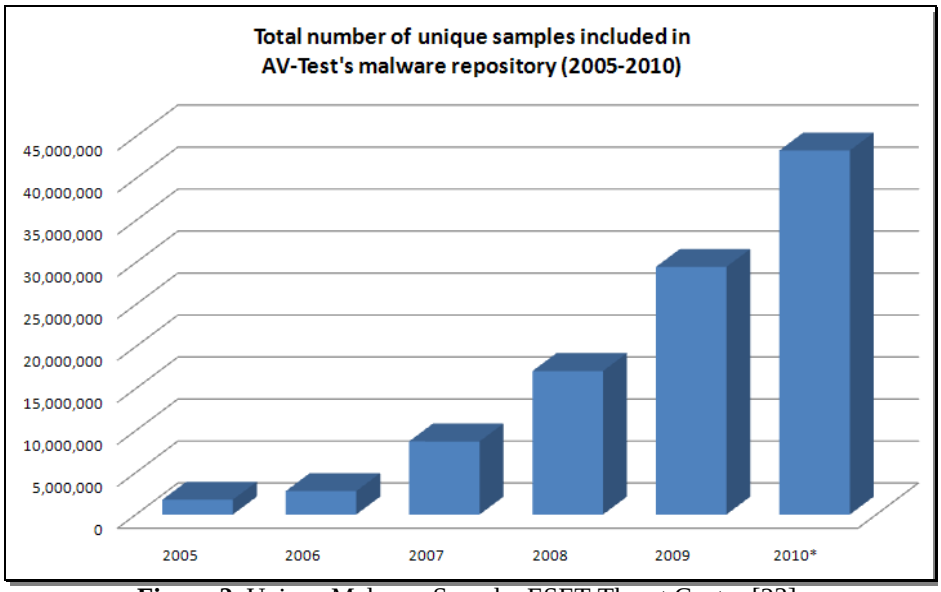

Figure 2. Unique Malware Sample: ESET Threat Center [23].

Both Figure 1 and Figure 2 show the increasing of malware sample and attack. Malware highly utilized the communication tools to spread itself. For examples, worms are being sent out through email and IM, Trojan horses attacked from infected web sites and viruses is downloaded from P2P connections to the user systems. According to Filiol, malware will pursue to abuse existing vulnerabilities on the systems to make their entry silent and easy [24].

In addition, malware continue to remain unnoticed, either by actively hiding or simply not making its presence on a system recognized to the user. The increase of malware causes billions of losses to the computer operation worldwide by breaking down the computer, congest the network, fully utilize the processing power and many more bad impacts. According to Langweg, the security control needs to be implemented in order to protect all code and data against modification, replacement or sub-versioned by malware activities [25].

2.2 Malware Avoidance Techniques

At the present time, malware creation becomes more sophisticated and expressively improved. According to Apel, almost all modern malware has been implemented with a variety of avoidance techniques in order to avoid detection and analysis by anti-malware product [5]. The avoidance techniques used practically by malware writers are code obfuscation, packing, anti-debugging and anti-virtualization. As stated by Noreen, the main purpose of these techniques is to avoid detection and to make the analysis process more complicated [8]. Code obfuscation technique can be divided into two types which are polymorphism and metamorphism. According to Preda, code obfuscation can change the malware syntax or signature but not their behavior of attacks which has to be maintained [6].

Code obfuscation consists of polymorphic and metamorphic technique. A polymorphic technique can change the malware binary representation as part of the replication

process. According to Vinod, this technique consists of encrypted malicious code along with the decryption module [26]. In addition, it has a polymorphic engine to decrypt and generate new mutants for the code before running it. When the polymorphic malware infect the computer systems, it will encrypt itself by using the new encryption key and the new code is generated. The work by Christodorescu shows that, polymorphism has been widely used to escape Syntax-based on specific bit sequences by generating random encodings [27].

As for metamorphic technique, malware will transform the representation of programs by changing the code into different ways when it replicates but it still performs the same behaviors [28]. According to Szor, this technique can include control flow transposition, substitution of equivalent instructions and variable renaming [7]. Metamorphic malware can reproduce itself into different ways to rapidly create new malware variants and never look like the old malware. Research by Preda shows that, most malware detectors nowadays are easily crushed because they use pattern matching and not resilient to modification of variations [6].

Malware writers use packing technique to compress the Win32 portable execution file (PE file) and the actual file will be unpacked when it is needed to be executed. According to Han, the main purpose of this technique is to protect the commercial software code from crack [29]. The work by Alazab shows that, a packed program contain within a program that is used for decompressing the compressed data during execution in the objective of makes the task of static analysis become more difficult [30]. Packers will compress and encrypt the program. However, the program is transparently decompressed and decrypted at runtime when the program is loaded into memory. Some malware even packed its code several times in order to make it harder to be unpacked and used up so many resources until the computer hang or crash.

Debugger is a useful tool for reverse engineering of the malware code. Debuggers normally step through each instruction in a program code before it is executed. According to Desfossez, debuggers perform their monitoring by either inserting breakpoints in the program or by tracing the execution using a special system call [31]. Research by Liu and Chen shows that anti-debugging is an active technique for malware writers to embed code which is aimed to check process list for the debugger process [32]. This technique will change functionalities of program when it interpreted by a debugger and make that program to discontinue its malicious intent or jump to end it.

Virtual environment is a place to do analysis and extract the features of the malware. Dynamic analysis commonly use virtual environment to place a malware sample and observe its behaviors. According to Lau and Svajcer, malware writers used anti-virtualization to create malware code that has a capability to detect virtualization environment to avoid the malware from being analyzed [33]. By using this technique, malware can be detected whether they are running in a virtual environment before the execution. According to Daewon, when the virtual environment is detected, malware will act like a genuine program or refuse to execute inside the virtual environment [34].

3 MALWARE CLASSIFICATIONS

Classifying malware into possible types was started somewhere in 1990s where the concept of malicious software clearly described for all unwanted code. In 1991, Computer Anti-virus Researchers Organization (CARO) decided that the fundamental principle behind the malware naming scheme should be grouped into families, which is based to the similarity of its programming code [35]. According to the naming scheme, only the family name and the variant name of a piece of malware are compulsory. It is known as CARO Malware Naming Scheme.

Normal practice by anti-malware products is creating a malware naming for each malware samples that have analyzed. The malware naming is very subjective and depends on particular vendors to come out with their own name. They also have different naming class. Different naming class can cause the malware characterization is different between each anti-malware products because the unique malware are created with avoidance technique and their characteristics belong to several malware classes [3]. The problem is due to the different naming schemes and can lead to a very serious confusion. In order to overcome this problem, this paper is proposed to standardize malware class and classify the malware based on malware specific target and its operation behavior.

3.1 Conventional Malware Classes

Malware are basically classified based on malware specific objective. As a result, there are diversities and different naming classes among the anti-malware vendors and researchers. According to Aycock, malware can be classed into ten (10) classes [3]. However, Apel stated that malware are consisted of seventeen (17) classes, whereas seven more classes are added to the previous list [5]. Recent study by Filiol stated that malware classes are divided into two groups which are "Self-Reproducing" and "Simple" groups [4].

Self-Reproducing group consists of Virus and Worms. According to Aycock, virus is a computer program that able to perform secret action without owner's permission [3]. It was firstly demonstrated to public in November 1983 by Cohen [36]. At that time, the number of virus increased with an ability to cause damage and the ability to avoid detection. According to Jussi, virus concepts become common in the late 1980 where it spreads through file and diskettes [36]. Virus can increase their probabilities of spreading to other computers by infecting files on a network or a file system accessed by another computer. Virus also can be spread as attachments in the e-mail note, in the downloaded file, on a diskette or CD.

Worms is quite similar to a virus by design and many researchers considered it as a sub-class of virus. Worms is a self-reproducing malware that run independently and travel across network connections without any user intervention. According to Mohamad Fadli Zolkipli and Aman Jantan, Worms use network connectivity to find an attack vulnerable system from nodes to nodes [12]. In general, the goal of worm is to infect as many computer systems that connected to the network. The difference between worms and viruses is, worms normally causes at least some harm to the network by consuming bandwidth and also it has the capability

to travel without any human action, whereas viruses normally corrupt or modify files on a targeted computer.

Simple group consist of Trojan horse and Logical Bombs. According to Filiol, Trojan horse is benign software that appears to perform a necessary function for the user prior to run or install but instead simplifies unauthorized access to the user's computer system [4]. Trojan horse designed to embed secret malicious task into other application or system. This software is normally made for servers and client modules whereby attacker can control and access the whole resources of infected systems [4]. By default Trojan horse will appear to be useful software but it will actually do damage once installed or ran on the computer. Trojan horse is also known for creating a "backdoor" on the computer that gives malicious users access to the system. Unlike viruses and worms, Trojans do not reproduce by infecting other files nor do they self-replicate [36].

Logical bomb is a simple type of malware which waits for significant event such as date, action and particular data to be activated and launch its criminal activity. Many logical bombs attack their host systems on specific dates, such as Friday the 13th or April Fool's Day. According to Filiol, to be considered as a logic bomb, the payload should be unwanted and unknown to the user of the software [4]. Some logical bombs can be detected and abolished before trying to execute through a periodic scan of all computer files, including compressed files, with an up-to-date anti-malware products.

Unique and blended malware is a malware created by malware writers by using avoidance technique. According to Martignoni, these types of malware combine two or more attribute from its predecessor and produce a new class of malware [9]. New class will be created each time when the new combination is detected and the list of malware class will continue to expand. Nevertheless, the unique malware still performs similar behavior even classified in different classes. Therefore, this paper try to propose new malware classes to overcome this issue, hence optimize the classification process. Figure 3 shows additional unique malware classes from the group defined by [4].

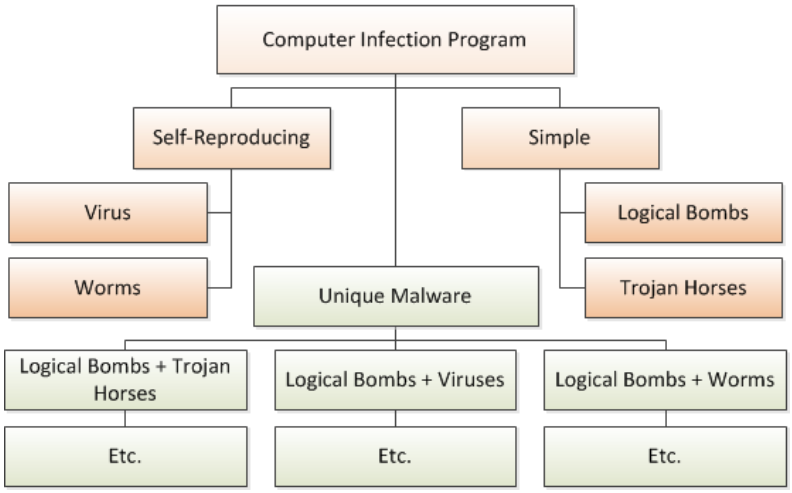

Figure 3. Malware with Additional Unique Class.

3.2 Machine Learning Classifier

Another component in malware classification system can be referred as machine learning classifier. Machine learning is an algorithm that allows computers to evolve behaviors based on empirical data such as from sensor data or databases. A key focus of machine learning research is to automatically learn to identify the complex patterns and make smart decisions based on data. The learner must generalize from the given examples to be able to produce a useful output in new cases. According to Bishop, the core objective of a learner is to generalize from its experience [37].
Machine leaning classifier is the main engine in the malware classification system. It is used to classify the malwares into its designated classes. This paper tries to enhance the current machine learning classifier by using GA. Table 1 shows current Machine learning classifier in malware classification.

Table 1. Machine learning classifier in malware classification

Classifier	Speed	Accuracy	Strength	Weakness
Naïve Bayes [38-39]	Very Fast	High	Fast, easier to maintain and consistence result	Sensitive to the correlated attributes
Support Vector Machine (SVM) [2][40]	Fast	High	Regression and density estimation results. Better performance in text classification, pattern segment and spam classification	Expensive and problem lies on the prohibitive training time.
Decision Tree [39][41]	Very Fast	High	Easy to understand, easy to generate rules and reduce problem complexity	Mistake on higher level will cause all wrong result in sub tree
K-Nearest Neighbor [42]	Slow	Moderate	Useful when the dependent variable takes more than two values and effective if the training data is large	Very computationally intensive. $O(n^2)$

This paper selects Decision Tree (DT) to be combined with GA in order to produce better classifier engine in Anti-Malware Classification System (AMCS). This Anti-Malware System (AMS) Classifier will used new proposed malware Class Target Operation (CTO) which is mainly based on malware specific target and its operation behavior. Malware target and operation is referring to the Windows file system which is in the tree format, thus making DT as the most suitable classifier to be enhanced and used in AMCS. New AMS Classifier is expected to give better classification result than current machine learning classifier.

4 OPTIMIZATION TECHNIQUES

An optimization process is a process that systematically comes up with solutions to optimize some specified set of parameters without sacrificing some constraint. According to Bonnans, the most common goals of optimization process are minimizing cost and maximizing throughput as well as efficiency [43]. In general, optimization is a concept that encompasses all kinds of order-generating processes other than emergence. Optimization is also about choosing or selecting outcomes and defined as better.
The problems created by current trends of malware attacks make classification

process much more complicated and does not provide good classification result. Therefore, there is a need to enhance malware classifiers in order to optimize its performance. This paper proposed the usage of GA under Evolutionary Algorithm (EA) to enhance selected classifier which is DT. According to Shafiq, EA has been received significant attention and does relatively well when tested on various benchmarks especially in classification problem of detecting malicious executable [44].

4.1 Genetic Algorithm

GA is a heuristic search that simulators the process of natural evolution. The standard GA can be seen as the combination of bit-string representation, with bit-exchange crossover and bit-flip mutation, roulette-wheel selection plus generational replacement. GA is belongs to the larger class of Evolutionary Algorithm (EA). GA include the survival of the fittest idea into a search algorithm which provides a method of searching which does not need to explore every possible solution in the feasible region to obtain a good result.
GA also commonly used on a learning approach to solve computational research problem. According to Mehdi, this heuristic algorithm is regularly used to generate useful solutions in optimization and search problems [11]. In a GA, a population of strings which encode candidate solutions to an optimization problem evolves toward better answers. By tradition, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible [45]. A simple presentation of GA is as follows;

```
generate initial population,
G(0);
evaluate G(0);
t:=0;
repeat
      t:=t+1;
      generate G(t) using G(t-
1);
      evaluate G(t);
until find a solution
```

GA technique is implemented in this paper in order to solve and classify the unique malware that was created by an avoidance technique. A framework is proposed by combining GA with the current machine learning classifier. New and unique malware can be detected and classify through this technique. It is because, unique malware work similar like crossover and permutation operation in GA [12]. An avoidance techniques that normally used by malware writers can be detected and classified using this new malware classification system because it not only filter the content but also train and learn by itself, so it can predict the current and upcoming trend of malware attacks.

4.2 Design Techniques

The first stage in GA is to decide on a genetic representation of a candidate solution to the problem. Chromosomes are represented by the malware, and it stores the representation of a malware class into four attributes based on malware target and its operation behaviors. The collection of chromosomes divided into training and testing data set. This collection also called as malware corpus.

```
// Malware target, specific
target for malware attack
vector<list<MalwareClass*>>
_slots;
```

```
// Malware classes for
chromosome
hash_map<MalwareClass*,int>
_Dataclasses;
hash_map<MalwareClass*,int>
_Appsclasses;
hash_map<MalwareClass*,int>
_Sysclasses;
hash_map<MalwareClass*,int>
_Dosclasses;
```

In addition, the chromosome should store its fitness value and the parameters which are used by genetic operations. The fitness value is described by;

```
// Fitness value of chromosome
float _fitness;
// Flags of class requirements
satisfaction
vector<bool> _criteria;
```

Chromosome also should be created according to the number of chromosomes and number of genes. Initially, each gene of chromosomes was filled by random group number between 1 and 4. Chromosome parameters are shows as below;

```
// Number of crossover points of
parent's malware
int _numberOfCrossoverPoints;
// Number of classes that is
moved randomly by single
mutation operation
int _mutationSize;
// Probability that crossover
will occure
int _crossoverProbability;
// Probability that mutation
will occure
int _mutationProbability;
```

After initialization of chromosome, the fitness of each chromosome must be evaluated. The following formula was used for evaluating the fitness of each chromosome.

$$f(x) = \sum_{n=1}^{\infty} (class\_score \div maximum\_score) * W_i$$

First part of this formula *(class_score ÷ maximum_score)* denotes the number of score obtain *(class_score)* divided to total number of maximum class which is 4 *(maximum_score)*. Score is obtained by checking the classification result. If malware in higher class is classified in lower class, we increase it score. Else we decrease the score. W_i denotes the weight of group of selected individuals. After selection process, the average weight of each individuals (W_i) was calculated by averaging the weight of each malware in each gene *(malware_class*4)* as below.

$$W_i = AVG(malware\_class * 4)$$

The fitness values are represented by single-precision floating point numbers (float) in the range 0 to 1. Once chromosome initialization phase was done, the genetic algorithm operations such as crossover and mutation must be applied on initialized chromosomes in order to create new population with better fitness value.

A crossover operation combines data in the hash maps of two parents, and then it creates a vector of slots according to the content of the new hash map. A crossover 'splits' hash maps of both parents in parts of random size. The number of parts is defined by the number of crossover points (plus one) in the chromosome's parameters. Then, it alternately copies parts from parents to the new chromosome, and forms a new vector of slots.

```
// Performes crossover operation
using to chromosomes
// and returns pointer to
offspring
```

```
Schedule*Crossover(const
Schedule& parent2) const;
```

A mutation operation is very simple. It just takes a class randomly and moves it to another randomly chosen slot. The number of classes which are going to be moved in a single operation is defined by the mutation size in the chromosome's parameters.

```
// Performs mutation on
chromosome
void Mutation();
```

5 PROPOSED FRAMEWORK

Framework is an abstraction in providing generic functionality and can be selectively overridden or specialized by user code, thus providing specific system functionality. Our proposed framework consists of three steps which are starting from input, followed by classification process and finishes with classification result as an experiment output. Figure 4 shows our proposed framework.

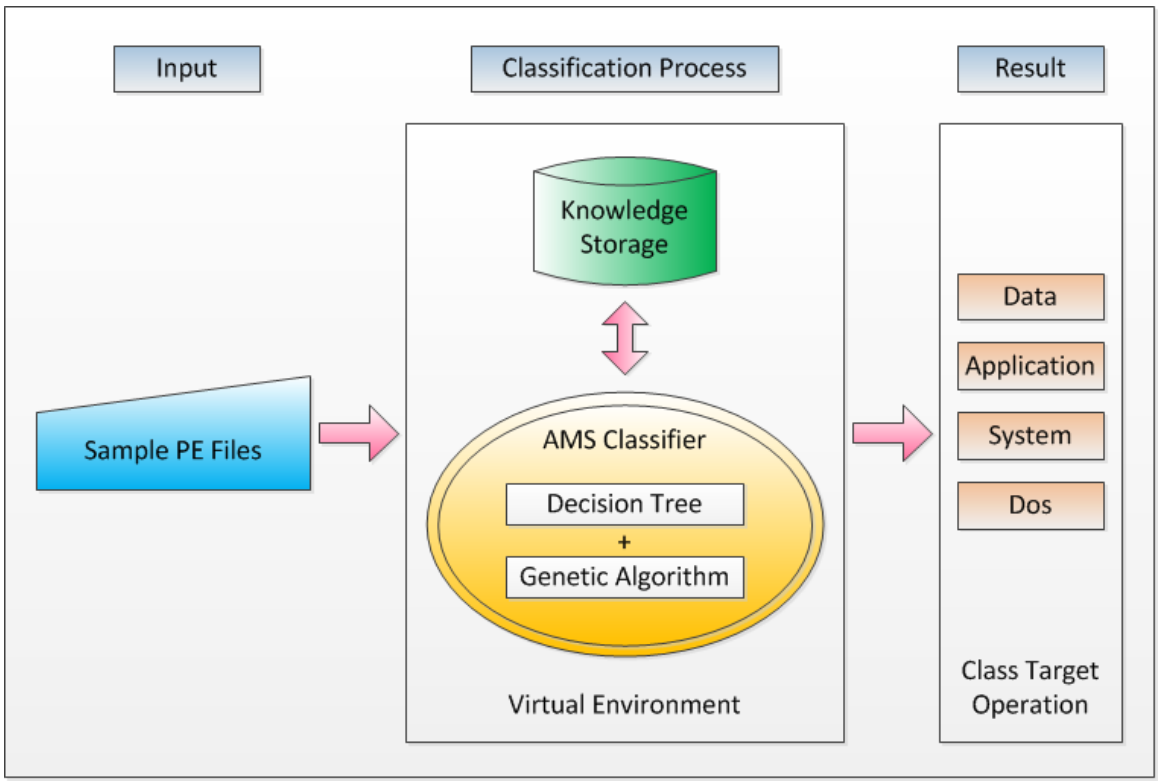

Figure 4. Framework for Optimizing DT in Malware Classification System by using GA [46]

5.1 Sample PE Files

Sample Portable Execution (PE) files for this experiment are divided into two categories which are malware and benign types. Malware behavior analysis is conducted to identify the behavior and characteristic of sample PE files. We conducted an analysis based on malware behavior analysis proposed by Mohamad Fadli Zolkipli and Aman Jantan [12].

The first step in malware behavior analysis is to collect the sample PE files. All the samples are collected from college network, internet and some suspicious execution file in windows operating system itself. This paper used HoneyClient [47] and Amun [48] as

malware collector tools. The samples also downloaded from VX Heaven [49] in order to maximize the varieties of sample PE files.

The second step is to extract the sample PE files in our collection. This sample is extracted to identify features reflecting behavior patterns such as modified file attribute, distributed global memory, load register and many more suspicious behavior by malware analyzer tools. For security purposes, this process is conducted in the virtual environment machine. This experiment used CWSandbox [50] and Anubis [51] as a malware analyzer.

The work by Wagener used up until 104 malware samples for malware analysis. 104 malware samples are used in order to obtain and identified malware behavior pattern [52]. Some of that malware sample shows the same behavior of attack and Wagener divided the similarity of malware into two categories which are lowest average similarity and highest average similarity. For this paper, we observed and monitored malware sample three times (3x) higher than malware sample used by Wagener. The reason we used higher number of malware sample during analysis is to obtain and observed more malware behavior of attack. This paper observed and monitored about 300 malware samples and divided the similarity into 20 suspicious behaviors.

In the third step, human-based behavior analysis is used to customize the result generated by both malware analyzers by using human expertise to analyze the sample PE files. It is important to have custom analysis because there might have different result on different malware analyzer tools. If the suspicious behaviors are found, the sample PE files is group under malware, else is group under benign types as shown in Figure 5.

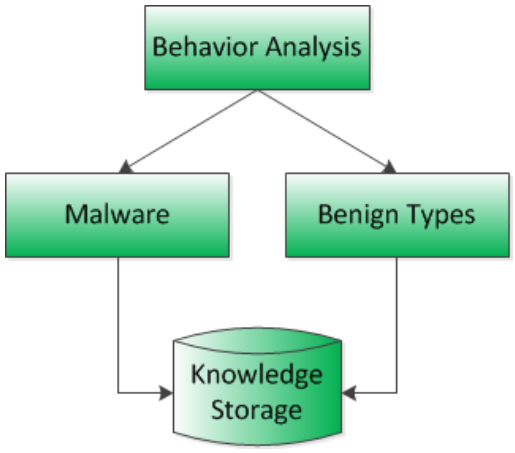

Figure 5. Malware and Benign Type.

The last step in the malware behavior analysis is a statistical report. This report is produced based on the result in the previous step and stored in the knowledge storage. The report is created in the profile format and we called it as "Malware Profile". This malware profile is used by classifiers to classify malware in our new Anti-Malware Classification System (AMCS).

5.2 Classification Process

Classification process is the main mechanism in our AMCS. This process consists of three components which are Virtual Environment, Knowledge Storage and AMS Classifier. Figure 6 shows the classification process in AMCS and its components.

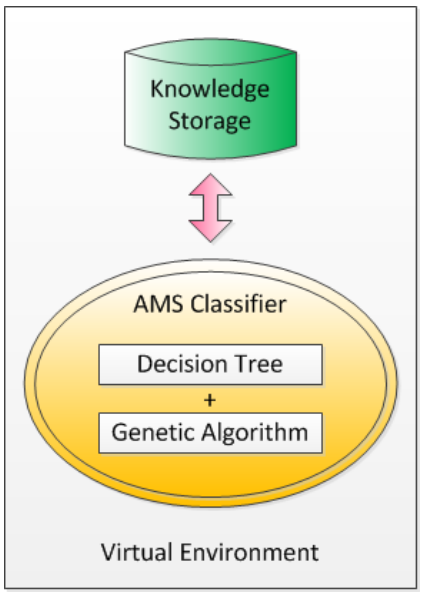

Figure 6. Classification Process.

Virtual environment is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third-parties, suppliers and untrusted users. Virtual environment machine run is isolated, so the benefit of a virtual environment machine is that it cannot directly modify the "real" operating system running on our machine. The virtual machine is assigned its own hard disk space, and that's what it treats as its virtual "entire hard disk". In this proposed framework, it has been decided to use Windows XP Mode in Windows 7 Platform as the testing platform. The purpose of this virtual environment is to create secure platform in order to minimize damage and attack to the actual machine when the malware sample is executed. Since virtual environment is assigned with its own hard disk space, if got infected, it can simply be removed from the test machine.

Knowledge storage is database storage to store all the malware profile after finishing the analysis and features are extracted. Malware profile in the storage consists of malware sample in MD5 format, malware size, malware specific target and CTO. The knowledge storage is designed to be re-writable by the system. Certain unique malware has a relationship and link with the other malware when it is executed. According to Desfossez, this malware relationship is unable to analyze because during analysis process, malware is executed one by one [31]. At first, the information in the knowledge storage is not sufficient enough to be used in the classification system. For this reason, this paper used GA to conduct a training phase together with the DT. The system will update the database during the training phase.

Anti-Malware System (AMS) Classifier is the main engine in this new malware classification system. We have selected the DT as our machine learning classifier and combine it with GA. In this new proposed system, we are providing the classifier training phase and GA is used to become a learning algorithm. During the classifier training phase, 800 sample PE files are used as training data set consists of 200 benign and 600 malware samples. The training data set is different from testing sample PE files. We must use different sample PE files because we want to let the classifier to learn and update the new malware into malware profile.

One of the main goals conducting this training phase is to detect and classify the unique malware that has a relationship when it is executed. The other goal is to find unique malware that perform the same behavior but providing different syntax representation. As mention earlier, the malware profile in the knowledge storage is designed to be re-writable. Classifier will keep on updating the malware profile during this training phase. The classification result

will become more accurate during the training towards threshold values. The training phase also shows the ability of GA in helping DT to optimize the classification task. Figure 7 shows the classifier training phase in AMCS.

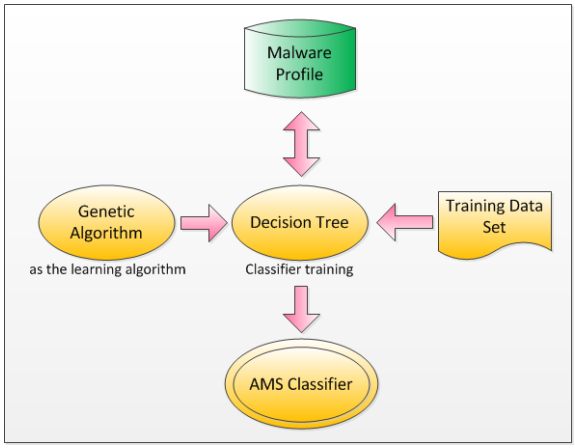

Figure 7. Classifier Training Phase.

5.3 Class Target Operation (CTO)

Malware behavior analysis has discovered 20 suspicious malware behaviors. This paper observed the executed malware by focusing it to the specific target and operation behavior in windows environment systems. Figure 8 shows the malware operation behavior chart percentage based on the result in malware behavior analysis.

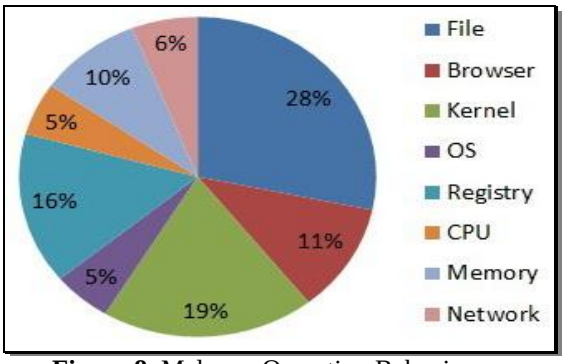

Figure 8. Malware Operation Behavior.

From the above chart, we classify the malware specific operation into 4 main classes which are Data, Application, System and Dos. Malware that are attacked File is group under Data class while malware that attacked Browser is group under Application class. Malware that are attacked Kernel, Operating System and Registry is group under System class. Lastly, malware that are attacked CPU, Memory and Network is group under Dos class. Table 2 shows more detail about new malware CTO.

Table 2. Machine learning classifier in malware classification

Class Target Operation (CTO)	Rank	Attacked examples	Affected examples
Data	1	Malware attack office and adobe file	.doc, .ppt, .xls and .pdf file
Application	2	Malware attack application such as office application, audio application and video application	Microsoft Office, Winamp and Windows media Player
System	3	Malware attack the entire Operating System	Windows XP, Windows 7 and Windows Server 2008
Dos	4	Malware attack physical hardware and entire machine	CPU usage, memory and network access

These four malware classes are related with each other in rank 1 to 4 starting with Data class and end with Dos class. These classes are actually inspired from basic fundamental of computer physical architecture. According to Ivy, physical architecture of host-based computer

consists of application, operating system and hardware [53].

In general, all these three layers are related with each other. Based on malware behavior analysis that we conducted earlier, we group the malware behavior into four classes. Data and Application classes are mainly grounded by application layer, System class is grounded by operating system layer and Dos class is grounded by hardware layer. The main reason that leads to the proposal of new malware CTO based on its operation is to reduce the process flow and cost in malware detection mechanism.

If the classifier is classified malware sample into Data class, the antidote is prepared based on Data operation only and if malware sample is classified into Application class, the antidote is prepared based on Data and Application operation. For malware in Data class, the detection mechanism does not need to look and prepare the antidote for other classes because that particular malware only attack file without attempt to attack the application and operating system. However, if malware attack Microsoft Office application which is under Application class, usually it will also affect the office application file under Data class, but not necessary to attack that particular operating system, which is under System class. The prediction system will check as well as estimate either that particular malware is harming System or not. If malware is harming System, the antidote is also prepared to System operation.

There are different focuses with System and Dos classes. When malware sample is classified into System class, the prediction system will estimate and check Application and Dos classes in order to prepare the antidote. The antidote will only be prepared when confirm by prediction system that the attack is affecting Application as well as Dos classes; else antidote is only for System operation. Same goes when malware sample is classified into Dos class. The prediction system will estimate and check System class and when confirm the attack also affecting System class, antidote will be prepared for both. Our new malware CTO relationship can be summarized as

$$\text{Data} \subseteq \text{Application} \leftrightarrow \text{System} \leftrightarrow \text{Dos}$$

However, not all malware will attack based on this relationship and rank class but it normally do. Duplicating a file is considered under Data class but it also consumes and bottlenecks the memory usage, so for that cases, it is classified into Dos class. The antidote is prepared for Data and Dos class only. All the decision in preparing antidote is made by prediction system.

6 EXPERIMENT and RESULTS

Similar work by Zhang and Wang used about 632 and 914 malware sample [54-55]. For this paper, we used more sample data which are 1000 sample portable execution files (PE file) consist of 250 benign programs and 750 malware samples. We follow ratio suggested by Wang for training and testing data set which are 80 % from sample PE files is used for training phase and the remaining 20% is for testing purposes. Training phase is used up until 800 sample PE files consist of 200 benign programs and 600 malware samples and the remaining 200 samples is for testing. Table 3 shows the summary of sample PE files for this experiment.

Table 3. Sample Data

	Sample PE Files	Training (80%)	Testing (20%)
Benign	250	200	50
Malware	750	600	150
Total	1000	800	200

The control value in this experiment is called as "Threshold". Threshold is the total or the value of training data that has been trained. This paper trained the training sample first and the control value is set from 0.6 to 0.9 (60 % to 90% trained samples). There are 200 sample PE files for testing purposes and the same samples is used for each Threshold values. In order to review the classifier, accuracies are calculated by using formula suggested in Edge as shown below [13].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The experiment is conducted by using DT Classifier first. 200 sample PE files have been tested by adjusting Threshold value from 0.6 to 0.9. Table 4 shows the relationship between Threshold value and the accuracy.

Table 4. Classification Result for DT Classifier

Threshold	TP	TN	FP	FN	Accuracies
0.6	134	45	16	5	89.5%
0.7	134	45	16	5	89.5%
0.8	136	45	14	5	90.5%
0.9	137	45	13	5	91.0%

After finishing the first experiment using DT, we proceed with the next experiment using our new AMS Classifier. The same Threshold value is used again. Table 5 shows the result of our second experiment.

Table 5. Classification Result for AMS Classifier

Threshold	TP	TN	FP	FN	Accuracies
0.6	141	47	9	3	94.0 %
0.7	142	49	8	1	95.5 %
0.8	144	50	6	0	97.0 %
0.9	144	50	6	0	97.0 %

The next step is to compare the first and second result of our experiments. Figure 9 shows the comparison between DT Classifier versus AMS Classifier.

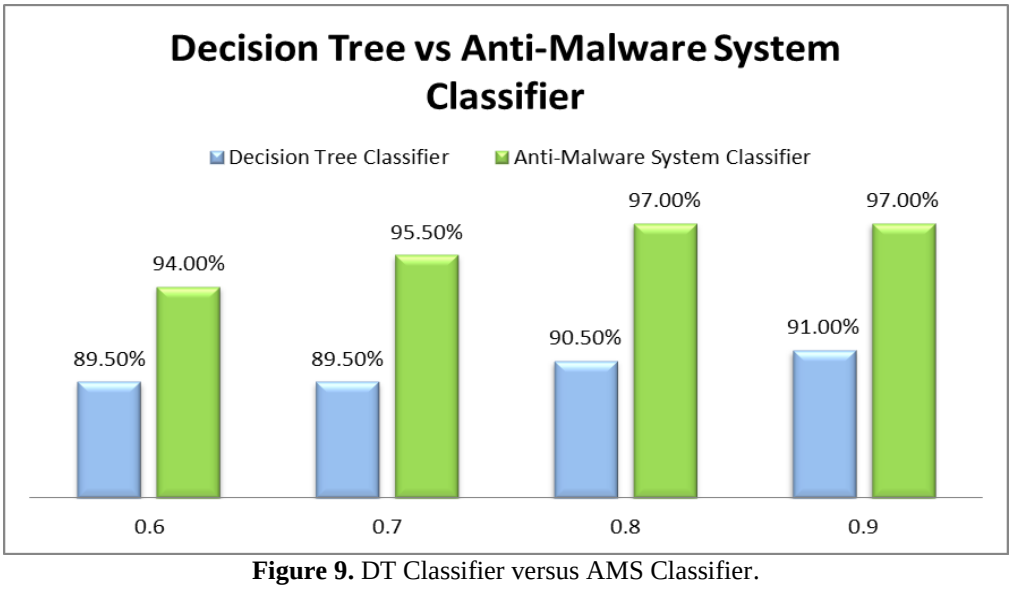

Figure 9. DT Classifier versus AMS Classifier.

7 DISCUSSIONS and CONCLUSION

The purpose of this experiment is not only to test workability of proposed approach but also to prove the proposed approach fairs better than existing approaches. Two experiments have been conducted in order to prove our approach. The first experiment is done using DT Classifier and the second experiment is done using AMS Classifier which is our proposed approach. Both experiments are tested using 200 sample PE files with four chosen threshold value for both experiments. The four threshold values are 0.6, 0.7, 0.8 and 0.9. The sample data is tested according to ascending value of the threshold. The output gained from both experiments is a classification accuracy rate. The first experiment yields out the classification accuracy rate using DT classifier while the second experiment yields out the classification accuracy rate using AMS Classifier. Obtained output from both experiments is then compared to see the classification accuracy rate difference. All obtained results are visualized in tables and graphs.

Results from the first experiment using DT Classifier are shown in Table 4. The results show that for threshold 0.6 and 0.7 obtained the same result. Both of these threshold obtained 134 TP, 45 TN, 16 FP and 5 FN. Threshold value of 0.8 obtained 136 TP, 45 TN, 14 FP and 5 FN while threshold value 0.9 obtained 137 TP, 45 TN, 13 FP and 5 FN. Based on Table 4, the accuracy rate for the first two thresholds stays at 89.5%. The accuracy increases to 90.50% for the third threshold and 91.00% for the last threshold.

Results from the second experiment using AMS Classifier are shown in Table 5. The results show that for threshold 0.6 obtained 141 TP, 47 TN, 9 FP and 3 FN. Threshold value of 0.7 obtained 142 TP, 49 TN, 8 FP and 1 FN. Threshold value of 0.8 obtained 144 TP, 50 TN, 6 FP and 0 FN while threshold value 0.9 obtained 144 TP, 50 TN, 6 FP and 0 FN. Based on Table 5, there is an increase of 1.5% in accuracy rate for the first two thresholds from 94.0% to 95.5%. The accuracy rate for the final two threshold values stays at 97.0%. Classification result of DT Classifier shows that there is no different for FP result in each different Threshold setting value. On the other hand, our AMS Classifier shows a decreasing result for TN and FN result over the Threshold values. It means that AMS Classifier is able to classify the sample PE files that has been failed to classify by the DT Classifier. This also proves that DT classifier is not resilient and does not perform well to an avoidance technique.

Results from the first experiment and the second experiment is then compared and graphically displayed in Figure 9. The comparison result between proposed approach and existing approach shows that there is an increase of accuracy rate for all threshold values in the favor of proposed approach. Threshold value of 0.6 shows an increase of 4.5% in accuracy rate while threshold value of 0.7 shows an increase of 6.0% in accuracy rate. As for the final two threshold values, the increased rate is 6.5% and 6.0%. As a conclusion, both experiment results shows that accuracy rate increases as the value of threshold increases and our proposed approach which is AMS Classifier yields a better accuracy rate compared to DT Classifier.

Acknowledgments. The author would like to thanks to the members in the Security Research Group, Universiti Sains Malaysia for their helpful discussion and suggestion. This work is supported by Short-term Grant No.304/PKOMP/639021.

8 REFERENCES

1. Gheorghescu, M.: An Automated Virus Classification System (2005).
2. Rieck, K., Holz, T., Willems, C., Düssel, P., Laskov, P.: Learning and Classification of Malware Behavior. In: Detection of Intrusions and Malware, and Vulnerability Assessment. Vol. 5137, pp. 108--125. Berlin, Heidelberg: Springer (2008).
3. Aycock, J.: Computer Viruses and Malware. Springer (2006).
4. Filiol, E.: Viruses and Malware. In: Handbook of Information and Communication Security. pp. 747--769. Springer (2010).
5. Apel, M., Bockermann, C., Meier, M.: Measuring Similarity of Malware Behaviour. In: The 5th LCN Workshop on Security in Communication Network. pp. 891--898. Zurich, Germany: IEEE (2009).
6. Preda, M., Christodorescu, M., Jha, S., Debray, S.: A Semantics-Based Approach to Malware Detection. In: Transactions on Programming Languages and Systems, Vol 30, No 5, pp 25--54 (2008).
7. Szor, P.: The Art of Computer Virus Research and Defense. In: Addison-Wesley Professional (2005).
8. Noreen, S., Murtaza, S., Shafiq, M., Farooq, M.: Evolvable Malware. In: 11th Annual Conference on Genetic and Evolutionary Computation. pp. 1569–1576. Montreal, Quebec, Canada: ACM (2009).
9. Martignoni, L., Paleari, R., Bruschi, D.: A Framework for Behavior-Based Malware Analysis in the Cloud. In: 5th International Conference on Information Systems Security. Vol. 5905, pp. 178-192. Berlin, Heidelberg: Springer-Verlag (2009).
10. Bayer, U., Habibi, I., Balzarotti, D., Kirda, E., Kruegel, C.: A View on Current Malware Behaviors. In: Usenix Workshop on Large-scale Exploits and Emergent Threats. Berkeley, CA, USA: USENIX Association (2009).
11. Mehdi, S., Tanwani, A., Farooq, M.: IMAD: In-Execution Malware Analysis and Detection. In: 11th Annual Conference on Genetic and Evolutionary Computation. New York, USA: ACM (2009).
12. Mohamad Fadli Zolkipli, Aman Jantan.: Malware Behavior Analysis: Learning and Understanding Current Malware Threats. In: Second International Conference on Network Applications, Protocols and Services. pp. 218--221. Kedah, Malaysia: IEEE (2010).
13. Edge, K., Lamont, G., Raines, R.: A Retrovirus Inspired Algorithm for Virus Detection and Optimization. In: 8th Annual Conference on Genetic and Evolutionary Computation. New York, USA: ACM 2006).
14. Zhao, H., Xu, M., Zheng, N., Yao, J., Ho, Q.: Malicious Executables Classification Based on Behavioral Factor Analysis. In: International Conference on e-Education, e-Business, e-Management, and e-Learning. pp. 502-506. Sanya: IEEE (2010).
15. Mohamad Fadli Zolkipli, Aman Jantan.: A Framework for Malware Detection Using Combination Technique and Signature Generation. In: Second International Conference on Computer Research and Development. pp. 196-199. Kuala Lumpur: IEEE (2010).
16. F-Secure IT Threats Security Summary (2008), http://www.f-secure.com/en_EMEALabs/news-info/threat-summaries/
17. Marcus, D. Malware Is Their Business…and Business Is Good! (2009), http://www.avertlabs.com/research/blog/index.php/2009/07/22/malware-is-their-businessand-business-is-good/
18. Kaspersky Security Bulletin. Malware Evolution (2010), http://www.securelist.com/en/analysis/204792161/Kaspersky_Security_Bulletin_Malware_Evolution_2010
19. Sophos – Security Threats Report: Mid-Year (2010), http://www.sophos.com/sophos/docs/eng/papers/sophossecurity-threat-report-midyear-2010-wpna.pdf
20. Symantec. Cyber War — Much Ado About Nothing or the Real Deal? (2010),

http://www.invincea.com/blog/2010/07/cyber-war-much-ado-about-nothing-or-the-real-deal/
21. G-Data - Number of New Computer Viruses at Record High (2010), http://www.gdatasoftware.co.uk/about-g-data/press-centre/news/news-details/article/1760-number-of-new-computer-viruses.html
22. Panda Security Lab. One third of existing computer viruses were created in Jan-Oct 2010 (2010), http://www.channeltimes.com/story/one-third-of-existing-computer-viruses-were-created-upto-october-2010-panda/
23. ESET. The Malware Challenge (2011), http://www.eset.com/us/threat-center
24. Filiol, E., Jacob, G., Liard, M. L.: Evaluation Methodology and Theoretical Model for Antiviral Behavioural Detection Strategies. In: Journal in Computer Virology, Vol. 3, No 1, pp 23--37 (2006).
25. Langweg, H.: Framework for Malware Resistance Metrics. In: 2nd ACM workshop on Quality of protection, pp. 39--44. New York, USA: ACM (2006).
26. Vinod, P., Laxmi, V., Gaur., M.: Survey on Malware Detection Methods. In: 3rd Hackers' Workshop on Computer and Internet Security. Kanpur, India (2009).
27. Christodorescu, M., Jha, S., Maughan, D., Song, D., Wang, C.: Malware Detection. In:Advances in Information Security. Verlag New York, Inc. Secaucus, NJ, USA: Springer (2006).
28. Zhang, Q., Reeves, D.: MetaAware: Identifying Metamorphic Malware. In: Twenty-Third Annual Computer Security Applications Conference. pp. 411--420). Miami Beach, Florida: IEEE (2007).
29. Han, S., Lee, K., Lee, S.: Packed PE File Detection for Malware Forensics. In: 2nd International Conference on Computer Science and its Applications, pp. 1--7. Jeju, Korea: IEEE (2009).
30. Alazab, M., Venkataraman, S., Watters, P.: Towards Understanding Malware Behaviour by the Extraction of API Calls. In: Second Cybercrime and Trustworthy Computing Workshop, pp. 52 - 59. Ballarat, VIC, Australia: IEEE (2010).
31. Desfossez, J., Dieppedale, J., Girard, G.: Stealth Malware Analysis From Kernel Space with Kolumbo. In: Journal in Computer Virology, Vol. 7, No. 1, pp. 83--93 (2011).
32. Liu, L., Chen, S.: Malyzer: Defeating Anti-detection for Application-Level Malware Analysis. In: Applied Cryptography and Network Security - Lecture Notes in Computer Science. Vol. 5536, pp. 201-218. Berlin, Heidelberg: Springer (2009).
33. Lau, B., Svajcer, V.: Measuring Virtual Machine Detection in Malware Using DSD Tracer. In: Journal in Computer Virology, Vol. 6, No. 3, pp. 181--195 (2010).
34. Daewon, K., Ikkyun, K., Jintae, O., Jongsoo, J.: Behavior-Based Tracer to Monitor Malicious Features of Unknown Executable File. In: Fifth International Multi-Conference on Computing in the Global Information Technology, pp. 152 - 156. Valencia: ICCGI (2010).
35. Banković, Z., Stepanović, D., Bojanić, S., Nieto-Taladriz, O.: Improving Network Security Using Genetic Algorithm Approach. In: Computers and Electrical Engineering, Vol. 33, pp. 5--6 (2007).
36. Jussi, P.: Digital Contagions. In: A Media Archaeology of Computer Viruses. New York, USA (2007).
37. Bishop, C.: Pattern Recognition and Machine Learning. Springer-Verlag (2006).
38. Wang, C., Pang, J., Zhao, R., Fu, W., Liu, X.: Malware Detection Based on Suspicious Behavior Identification. In: First International Workshop on Education Technology and Computer Science, pp. 198--202. Wuhan, Hubei: IEEE (2009).
39. Dewan Md Farid, Nouria Harbi, Mohammad Zahidur Rahman.: Combining Naive Bayes and Decision Tree for adaptive Intrusion Detection. Vol. 2, No. 2, pp. 12--25 (2010).
40. Mezghani, D., Boujelbene, S., Ellouze, N.: Evaluation of SVM Kernels and Conventional Machine Learning Algorithms for Speaker Identification. In: International Journal of Hybrid Information Technology, Vol. 3, No. 3 (2010).
41. Komashinskiy, D., Kotenko, I.: Malware Detection by Data Mining Techniques Based on Positionally Dependent Features. In: International Conference on Parallel, Distributed and Network-Based Processing, pp. 617—623. Pisa (2010).
42. Hall, P., Park, B., Samworth, R.: Choice of neighbor order in nearest-neighbor

classification. In: An Official Journal of the Institute of Mathematical Statistics, Vol. 36, No. 5, pp 2135--2152 (2008).
43. Bonnans, J. F., Gilbert, J. C., Lemaréchal, C., Sagastizábal, C. A.: Numerical Optimization: Theoretical and Practical Aspects. Berlin: Springer-Verlag (2006).
44. Shafiq, M. Z., Tabish, S. M., Farooq, M.: On The Appropriateness of Evolutionary Rule Learning Algorithms for Malware Detection. In: 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference. New York, USA: ACM (2009).
45. Cha, S.-H., Tappert, C.: A Genetic Algorithm for Constructing Compact Binary Decision Trees. In: Journal of Pattern Recognition Research, Vol. 4, No. 1 (2009).
46. Mohd Najwadi Yusoff, Aman Jantan.: A Framework for Optimizing Malware Classification by using Genetic Algorithm. In: Communications in Computer and Information Science, Vol. 180, Part 1, pp 58--72, SpringerLink (2011).
47. HoneyClient. Capture-HPC Client Honeypot. (2011), https://projects.honeynet.org/capture-hpc
48. Amun. Python Honeypot (2011), http://amunhoney.sourceforge.net/
49. Schufa. VX Heavens. (2011), http://vx.netlux.org/
50. CWSandbox. Honeyblog (2011), http://honeyblog.org/categories/5-CWSandbox
51. Anubis. Analyzing Unknown Binaries (2011), http://anubis.iceclab.org/
52. Wagener, G., State, R., Dulaunoy, A.: Malware Behaviour Analysis. In: Journal in Computer Virology, Vol 4, No. 4, pp 279--287 (2007).
53. Irv, E.: The Architecture of Computer Hardware and System Software: An Information Technology Approach. John Wiley & Sons (2009).
54. Zhang, B.-y., Yin, J.-p., Hao, J.-b., Zhang, D.-x.,Wang , S.-l. Using Support Vector Machine to Detect Unknown Computer Viruses. In: International Journal of Computational Intelligence Research, Vol. 2, No. 1 (2006).
55. Wang, Z., Cui, D.: A Hybrid Algorithm Based on Genetic Algorithm and Simulated Annealing for Solving Portfolio Problem. In: International Conference on Business Intelligence and Financial Engineering, pp. 106--109. Beijing: IEEE (2009).