A Hybrid Wrapper-Filter Approach for Malware Detection

Mamoun Alazab[a], Shamsul Huda[b], Jemal Abawajy[c], Rafiqul Islam[d], John Yearwood[b],Sitalakshmi Venkatraman[b],Roderic Broadhurst[a]
[a] CEPS at the Australian National University (ANU), Canberra 2601 Australia
Email: mamoun.alazab@anu.edu.au,roderic.broadhurst@anu.edu.au
[b] CIAO, School of SITE, University of Ballarat, VIC 3353, Victoria, Australia
Email: {s.huda, j.yearwood}@ballarat.edu.au
[c] School of Information Technology, Deakin University, Australia
Email: jemal.abawajy@deakin.edu.au [d] School of Computing and Mathematics, Charles Sturt University, Australia
Email: mislam@csu.edu.au

Abstract— **This paper presents an efficient and novel approach for malware detection. The proposed approach uses a hybrid wrapper-filter model for malware feature selection, which combines Maximum Relevance (MR) filter heuristics and Artificial Neural Net Input Gain Measurement Approximation (ANNIGMA) wrapper heuristic for sub-set selection by capitalizing on each classifier's strengths. The novelty of the proposed approach is that it injects the intrinsic characteristics of data obtained by the filter into the wrapper stage and combines this with wrapper's heuristic score. This in turn can reduce the search space and guide the search for the most significant malware features that assist in detection. Extensive cross-validated experimental investigations on actual malware datasets were conducted to evaluate the performance of the proposed model. The model was compared with several existing models including independent wrapper and filter approaches. The results of the model's performance on both obfuscated malware as well as benign datasets showed that the proposed hybrid MR-ANNIGMA model out-performed the independent filter and wrapper approaches by achieving the highest accuracy of 97%. Furthermore, this hybrid model improved execution time by using a more compact set of operation code features, and also reduced the rate of false positives.**

Index Terms— **Malware, opcodes, feature selection, wrapper-filter, neural network, multi-layer perceptron networks**

I. Introduction

Cybercrime represents the fastest growing crime globally. Recent attacks using obfuscated methods of malicious codes (previously unknown malware) have resulted in disruption of services leading towards significant financial losses and legal implications [1]. A review of the history of malware [2], anti-malware reports [3] and attempts at predicting trends in malware [4] show cybercrime continues to evolve in both scale and sophistication while traditional methods of malware detection appear increasingly inadequate. Improved detection of malware should help improve Internet safety and increase confidence in e-commerce.

Malware authors are able to easily deceive detection filters or malware search engines commonly used to identify such risks by enhancing the obfuscation techniques on known malware [5]. These deceptive modifications develop highly sophisticated and frequent variant distribution techniques to produce malware that evade anti-malware scanners. Code obfuscation techniques such as packing, polymorphism and metamorphism are used by cyber-criminals to modify the program code. Such modifications preserve functionally, while reducing vulnerability to many forms of static analysis, and also deter reverse engineering by making the code less readable and thus difficult to understand. Code obfuscation also produces 'offspring' versions that have the same functionality but use different byte sequences that are not recognized by antivirus scanners [6]. Polymorphic malware uses encryption and 'data appending/data pre-pending' methods to change the body/main code of the malware and further change/modify the decryption routines and their keys from one infection to another. Metamorphic methods automatically generate/produce morphed copies of the binary code of an original program without the need for encryption [7].

Code obfuscation techniques have undermined the value of signature-based detection techniques used in traditional Anti-Virus (AV) engines. Existing mitigation/detection strategies suffer from a number of shortcomings that include: (i) high false positive rates that identify benign files as malware [3] [2], (ii) high false negative rate due to the failure to detect obfuscated malware [3] [5], (iii) detection rate are declining [8]. Hence, there is a need to build more accurate models for detecting malware that are less dependent on signature-based detection.

In this paper, we propose an approach that gathers the operation code (OPcode) information of the source code and then using hybrid wrapper-filter techniques to distinguish malicious code from benign code. We develop an automated method to disassemble the binary executable commands in order to calculate the OPcodes frequency on the Win32/API Portable Executable format files (PE) [9] and then compare OPcode distributions within malicious and benign files. We also apply an innovative wrapper-filter model that combines the filter's

doi:10.4304/jnw.9.11.2878-2891

ranking score with the wrapper-heuristic's score for the likelihood of significant malware feature selection by using existing Maximum Relevance (MR) and Artificial Neural Net Input Gain Measurement Approximation (ANNIGMA) techniques. The hybrid wrapper-filter classifier is applied on the OPcode information about/on the source code in order to develop accurate and efficient malware detection systems. Experimental results demonstrate the efficiency of the proposed systems/methods and validate the significance of combining the advantages of both filter and wrapper heuristics not previously reported in the literature. We summarise our results as follows:

1) A fully automated heuristic method was implemented that disassembled the binary executable and computed the OPcode frequency statistics by exhaustively exploring/comparing the OPcode feature patterns for the entire Intel-32 assembly language OPcodes.
2) A proposed hybrid wrapper-filter based feature selection technique was implemented to find a compact and significant set of OPcode features that was accurate and efficient in malware detection. To the best of our knowledge, this is the first work to implement the hybrid method in malware detection.

The rest of the paper is organized as follows. The next section introduces the background literature and limitations of current malware detection techniques. The OPcode frequency statistics and the hybrid of wrapper-filter feature selection algorithm using the combination of Maximum-Relevance (MR) heuristics and Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) is described and the hybrid model (MR-ANNIGMA) discussed in Section 3. Section 4 presents data from our experimental evaluations and analysi on the accuracy and efficiency of our approach. Finally our conclusions are presented in the last section.

II. Related Studies

Since signature-based detection approaches require expert advice and close manual operation, there is a need to capture malware based on anomalies or behavioural patterns and prudently filter them to achieve better accuracy in the detection of unknown malware samples. Research shows that OPcode are useful in the classification of malware as well as in detecting injected malicious executable [10] [2] [11] [12]. Recent studies have used analysis of OPcode for the generation of so-called 'birthmarks' on portable execution files [8] . Statistical analysis of binary file content, including statistical N-gram modelling techniques,have been used and tested as a means of identifying malware binary code in document files [13]. But these methods did not yield sufficient differences to resolve different file types. Other related studies [14], have found that the statistical modelling of hidden malware code have yet to perform detection tasks accurately in the context of 'as it happens' real-time efficiency. For example, metamorphic engine techniques that generate innumerable code obfuscations that produce exponentially large morphed copies of an original program easily outpace such statistical modelling capabilities. These gaps in the literature motivate our efforts to understand malware behavior and we propose an efficient and statistical analysis of OPcode.

Shankarapani et al. [14] have shown that the frequency of Windows API (Windows Application Programming Interface) calls can be used to classify and detect malware with accuracy. The authors performed a static analysis to measure the similarity found among 1593 executable files that contained either malware or benign codes. Two methods have been used based on the frequency of occurrence of each API. First, a similarity analysis method is proposed which computes the mean value for 3 similarity measures (i.e. Cosine or the Jaccard or the extended Jaccard measure). A second method uses a Support Vector Machine (SVM) to classify code as either malware or benign. However, the result of sensitivity or recall measures observed by the Receiver Operating Characteristic (ROC) curve was low and the false positive rate was too high for practical use. Cesare and Xiang [15] also performed similarity analysis using string edit distances based on control flow to identify malware variants, however, their analysis focused only on packed malware and not OPcode. Schultz et al. [16] used supervised machine learning for malicious code detection based on the respective binary code for program headers, strings, and byte sequences. Malan and Smith [17] added a measure of temporal consistency to the OPcode raw frequency in order to calculate the frequency of malware or anomalous code. In another study the authorswebster06, introduced an algebraic specification for the Intel (IA-32) assembly programming language as another approach to the detection of metamorphic computer viruses. In another unigram analysis of binary byte values was applied to generate a code 'fingerprint' for identification and classification purposes [18],. The Portable Executable Analysis Toolkit or 'PEAT' has also been developed to determine malicious behaviour in code for Windows Portable Executable (PE) files and also relied on analysing the structural features of the executable file [9] [19],.

Another approach. known as the Static Analysis for Vicious Executables (SAVE) [20] was not based on API calls alone to attempt to detect polymorphic and metamorphic malwares. They defined a suspect malware 'signature' as an API sequence of calls and applied a reverse engineering process from decompressed16 binaries, which are then passed through a portable executable (PE) file parser. In another study [21], an intelligent/machine learning? Malware Detection System was designed to detect polymorphic malware and unknown malware based on the analysis of Windows API execution sequences also extracted from binaries. Yet another approach based on Function Length Frequency and Printable String Information was discussed by Islam and colleagues [22]. Features extracted from the executable files are used to classify malware by distinguishing between broad families of malware found in a database of 2398 known malware

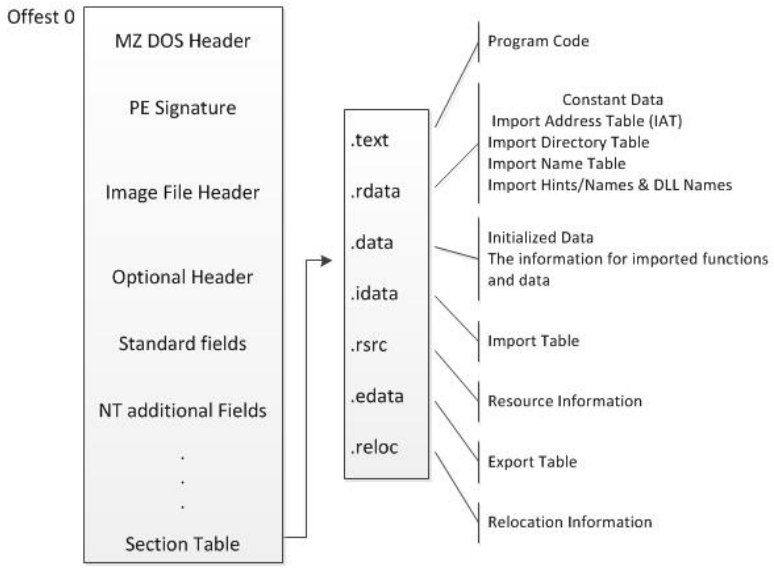

Figure 1. The Portable Executable file format

files.

Rabeck and colleagues proposed a host-based technique that uses static analysis based on monitoring and validating Win32 API calls for detecting malicious code in binary executables [6],. A recent study [14] on the performance of kernel methods in the context of robustness and generalization capabilities of malware classification reported that analysis based on the Win API function call also provided sufficient accuracy to classify malware. Bilar [23] performed statistical analysis based on the frequency distributions of OPcode on 67 known malware code and 20 benign code. Results showed that malware OPcode distributions differ statistically significantly from the OPcode distributions benign software. Shabtai et al [8] used OPcode n-gram patterns as the select features to detect unknown malware on 30,000 executable file and detected 96

Significant research work has also been reported in the literature on feature selection methods [24]–[27]. These can be grouped broadly into three main categories based on the evaluation criteria used: 1) the filter model, 2) the wrapper model, and 3) hybrid models. The filter models are based on the intrinsic characteristics of the data and do not involve the application of an induction/inductive? algorithm. Different filter models have been advanced often involving relevance measures or distance measures as their evaluation criteria [28]. The heuristics of filter models [27] estimate the discrimination capabilities of the subsets of the features which is also dependent on wrapper evaluation for accurate performance measure. In contrast, the wrapper models face huge computational overheads/costs due to the repeated execution of the induction algorithm and iterative subset generation process. Hsu et al. [29], use a wrapper heuristic approach based on the Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA). In short previous research indicated there is a need to identify the advantages and disadvantages of both filter and wrapper approaches. citehuda10, Huda et al. show that a hybrid of wrapper-filter heuristic significantly improves detection performances over independent wrapper and filter approaches in different data mining applications. However, we have not found studies that applied, hybrid models in the context of malware detection.

The above studies summarize current detection approaches however not many perform sufficient statistical analysis to determine if the anomaly detected was in reality malicious [16], [2]. In this paper, we employ static analysis to inspect the program code with the goal of determining OPcode frequency statistics in a controlled environment. Extracting features from the obfuscated executable file for reverse obfuscation is labor intensive and requires deep understanding of kernel and assembly programming. Therefore, this paper develops a fully automated system to extract OPcode features useful for finding the 'fingerprint' of executable programs. In addition, a novel Wrapper-Filter feature selection model is also proposed. Standard filter approaches can extract knowledge of the intrinsic characteristics from real data but do not use any performance criteria based on the likelihood of predictive accuracy. Hence, there is no guarantee that a selected feature subset would do better in the crucial classification/prediction tasks essential in automated detection. The use of predictive-accuracy based evaluation criterion in the wrapper approach would ensure better performance from the selected feature subset, but repeated execution/iteration of the induction algorithm in the search process incur a high computational cost. The proposed hybrid approach uses the advantages offered by both filter and wrapper approaches and this study demonstrates those advantages in terms of both speed and the accuracy of malware detection.

A. Executable files and OPcodes

The Microsoft Win32 Portable Executable (PE) [9] file formats such as (.EXE and .DLL) is the standard executable format for all versions of the Windows operating system on all supported processors. As shown in Fig. (1), PE file has different sections and headers. PE files start with the Microsoft DOS header which is identified by 'MZ' which is the initials of Mark Zbikowski (one of the developers of DOS). The second section is the PE Signature field, which when viewed as ASCII text is PE\0\0. Third is the IMAGE_FILE_HEADER containing the most basic information about the file. Fourth, is IMAGE_OPTIONAL_HEADER that contains the structure of additional information provided by the PE creators, beyond the basic information found in the IMAGE_FILE_HEADER. Last is the section table that has code sections (.text), and data sections (.data). The .text section is the default section for code and the .data section stores writable global variables and also contains the file's Original Entry Point (OEP) which refers to the execution entry point (where the file execution begins) of a portable executable file. Finally, the .rdata section contains read-only data.

B. Wrapper and Filter Feature Selection

In this paper, we adopt two different feature selection approaches, namely Maximum Relevance (MR) filter approach and Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) wrapper heuristic approach, and develop a hybrid model within a multilayer perceptron neural network (MLP-NN) framework for malware detection system.

C. Maximum Relevance (MR) Filter

Relevant features can provide more information about the class variable than irrelevant features [28]. Therefore mutual information based maximum relevance [28] is a suitable heuristic for selecting the most informative APIs. If S is a set of APIs F_i where $\{F_i|F_i \in S : i = 1, 2, 3...\}$ and Malware class variable is c, the maximum relevance can be defined as (1). Here c denotes class values of a particular sample.

$$Maximum\ Relevance(S, c) = \frac{1}{|S|} \sum_{F_i \in S} I(F_i; c) \quad (1)$$

$I(F_i; c)$ is the mutual information between F_i and class c which is defined as

$$I(F_i; c) = H(F_i) - H(F_i|c) \quad (2)$$

$H(F_i)$ is the entropy of F_i with the probability density function p. If F_i takes discrete values from set of values $V = \{v_1, v_2, v_3....v_l\}$, then,

$$H(F_i) = -\sum_{v_l \in V} p(v_l) log(p(v_l)) \quad (3)$$

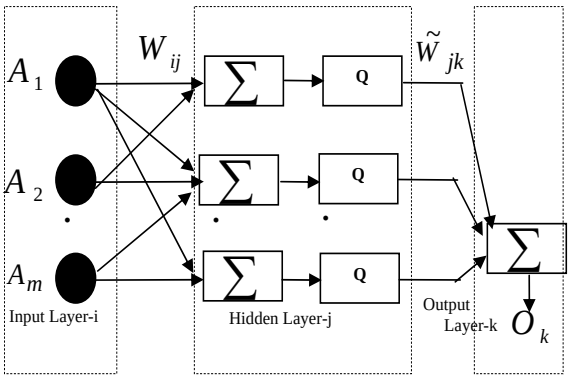

Figure 2. A single hidden layer Multi layer Perceptron (MLP) neural network in wrapper approach algorithm.

Let $H(F_i|c)$ be the conditional entropy between F_i and c then,

$$H(F_i|c) = -\sum_{v_l \in V} \sum_{c_m \in C} p(v_l, c_m) log(p(c_m|v_l)) \quad (4)$$

where class variable c takes the discrete values from the set $C = \{c_1, c_2, c_3...c_m\}$ In general, filter models are computationally cheap due to their evaluation criteria. However, feature subsets selected by filter may result in poor prediction accuracies, since they are independent from the induction algorithm. In contrast, wrapper models use a predetermined induction algorithm and use predictive accuracy as the evaluation criteria for the feature selection. Wrapper models face huge computational overhead due to the use of the induction algorithm's performance criteria as their evaluation criteria. In [30] a hybrid of genetic algorithm and filter heuristic was proposed, where GA framework forms the subset generation process, while the filter heuristic improves local search. GA-based approaches face huge computational overheads due to the evaluation of the induction algorithm embedded in the GA fitness function. Some wrapper approaches [29] use heuristics generated from wrapper knowledge over wrapper iteration. A popular wrapper heuristics is Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA).

D. ANNIGMA wrapper heuristic

Nonlinear Gain Analysis (NLGA) is an approach of feature subset selection and is also known as Artificial Neural Net Input Gain Measurement Approximation (ANNIGMA) ranks features [29]. Neural networks are suitable for training large amount of data and it is an unsupervised learning, the variables that are higher weight is more important. ANNIGMA is a weight analysis based wrapper heuristic that ranks features by relevance based on the weight associated with feature in a Neural Network based wrapper approach. Features that are irrelevant or redundant will produce more error than relevant features. Therefore, during training, weights of noisy features are controlled in such a way that they contribute to the output as least as possible. ANNIGMA is based on the above

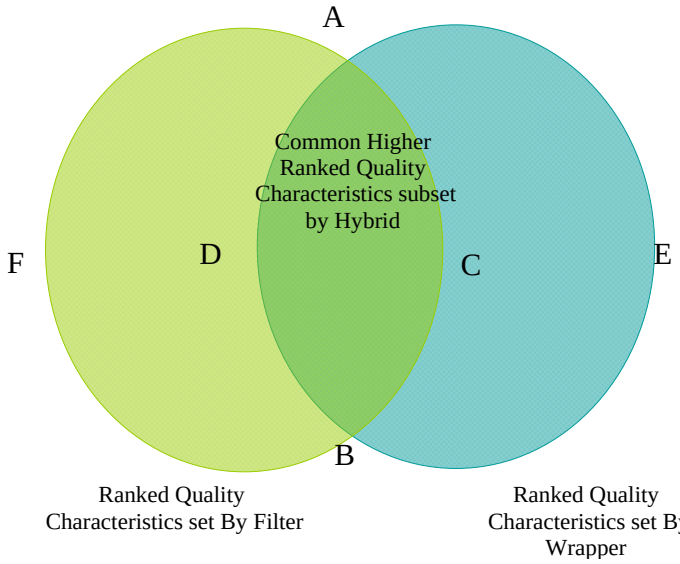

Figure 3. Conceptual feature subset selection by hybrid model.

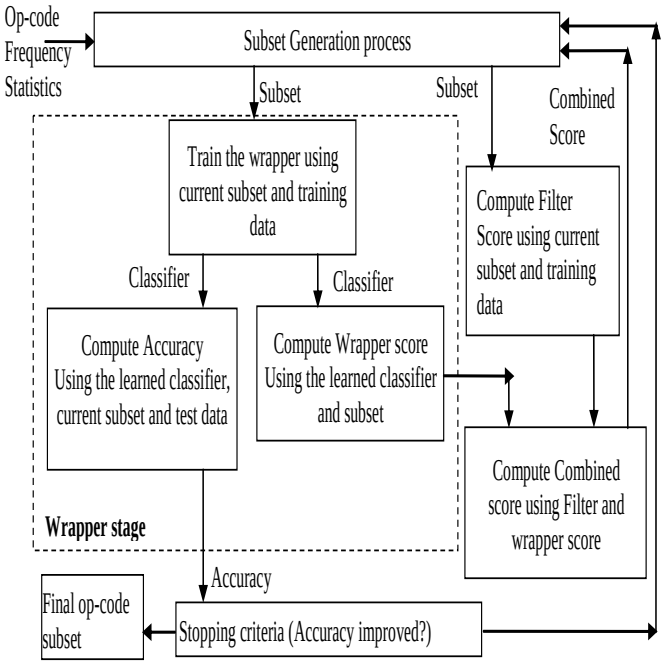

Figure 4. Schema of our hybrid Wrapper-Filter classifier.

strategy of the training algorithm. Fig.(3) demonstrates the architecture of the neural network. The NLGA consists of training process and the calculation of the ANNIGMA score is described as follows. As shown in (2) for a two layer Neural Network if i, j, k are the input, hidden and output layer and Q is a logistic activation function (5) of the first layer and second layer has a linear function, then output of the network 'O_k' is as (3). Here 'A_i' are the input feature, 'W' are the weight between network layers.

$$Q(x) = (1/(1 + exp(-x))) \tag{5}$$

$$O(k) = \sum_j Q(\sum_i A_i \times W_{ij}) \times W_{kj} \tag{6}$$

Then local gain is defined as:

$$LG_{ik} = \frac{\Delta O_k}{\Delta A_i} = \sum_j |W_{ij} \times W_{jk}| \tag{7}$$

Then ANNIGMA score is the local gain normalized on a unity scale as equation (8) [29]:

$$ANNIGMA_{ik} = \frac{LG_{ik}}{max(LG_k)} \tag{8}$$

III. Proposed Hybrid wrapper-filter Model

Our earlier research [31] shows that a hybrid of wrapper-filter heuristic significantly improves the performances in different data mining applications. However, to the best of our knowledge, hybrid wrapper-filter heuristics have not been attempted to classify malware. Proposed methodology does not require any knowledge of the binary signatures.

Proposed hybrid approach introduces the maximum relevance filter heuristic in the wrapper stage and combines MR with ANNIGMA wrapper heuristic to take advantage of both filter and wrapper approaches. The aim is to find a more compact set of significant OPcodes, much faster than either wrapper or filter approaches when applied separately to classify malware. The idea behind this approach can be explained by the Venn-diagram as shown in Fig.(3). If the two feature subsets ACBF and ADBE are separately ordered/ ranked according to their score, then common higher ranked feature subset (ACBD) is the strongly recommended most significant feature subset by the both feature selection algorithms. If the scores of both algorithms are normalized on the same scale and combined, then feature subsets with higher combined scores provide the common higher ranked feature subset from both algorithms. Our premise is that a Backward Elimination (BE) search strategy based on the combined score along with the wrapper evaluation criteria would be able to find the most significant features. Performance of the combined score may be affected due to performance of the incorporated filter for a particular wrapper approach in the hybrid. However, different filter approaches can be combined to find a suitable hybrid for a particular wrapper heuristic and vice-versa. In this paper, we have combined mutual information based Filter-Maximum Relevance (MR) with Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) based wrapper. We have focused on a Neural network based wrapper and different filter heuristics. We will use other wrapper approaches in a future work. The proposed hybrid approach avoid the computational overhead of generating subsets and takes advantage of both filter and wrapper heuristics. The different sub-components in the hybrid approach have been shown in Fig.(4) and main steps of the algorithm are described in the following sub-sections.

Algorithm 1:Procedure Hybrid Wrapper filter approach for Malware detection

Input: $D(F_1, F_2, \ldots F_n)$ // Training data with m features
Output: S_{BEST} //an optimal subset of features
Begin
1. Let S = whole set of m features $F_1, F_2, \ldots F_m$
2. S_0 = Initial set of features which records all generated subsets with accuracy
// Apply a Backward Elimination (BE) search strategy
3.. for $N = 1$ to $m - 1$
4. Current set of features $S_{support} = S$
5. Compute Filter score by 11
6. for $fold = 1$ to n
7. Train the network with $S_{support}$
8. Compute ANNGMA of all features
9. Compute Accuracy
10. End for
11. Compute average accuracy of all folds for $S_{support}$
12. Compute average ANNIGMA of $S_{support}$ by 11
13. Compute combined score for every feature in $S_{support}$
14. Rank the features in $S_{support}$ using the combined score in descending order
15. $S_0 = S_0 \cup S_{current}$
16. Update the current feature set $S_{current}$
17. End for
18. S_{BEST} = Find the subset from S_0 with the highest accuracy
19. Return S_{BEST}
End

$$ANNIGMA(F_i)_{average}) = \frac{1}{n}(ANNIGMA(F_i)_1 + ANNIGMA(F_i)_2 + ... + ANNIGMA(F_i)_n) \quad (9)$$

$$Relevance(F_i) = \frac{I(F_i; c)}{maximum_{(F_i \in S)} I(F_i; c))} \quad (10)$$

$$Combined\ Score : MR_ANNIGMA(F_i) = \frac{I(F_i; c)}{maximum_{(F_i \in S)} I(F_i; c))} + ANNIGMA(F_i)_{average}) \quad (11)$$

A. Combined Model

We use the Artificial Neural Network as the induction framework in the wrapper for the computing the combined score of our hybrid MR- ANNIGMA model. The overall schema of our hybrid Wrapper-Filter approach is given in Fig.(4). An n-fold cross-validation approach has been used in the model to train the wrapper. In each fold we compute the ANNIGMA score for every feature. Then after training of all folds, the ANNIGMA score is averaged as given by equation (9) While computing the combined score in the proposed ANNIGMA, the relevance of a feature in the current subset is computed from the individual score which is scaled to the maximum individual relevance of the subset. Thus, relevance of a feature in a subset within the hybrid approach is defined as given in equation (10): The combined score of filter's heuristic and wrapper's heuristic in the proposed MR-ANNIGMA is computed as in equation (11):

The algorithm (Algorithm 1) details of the hybrid approach are provided in the sections (III-B) and (III-C).

B. Search Strategies

We apply a Backward Elimination (BE) search strategy in MR-ANNIGMA to generate a subset of OPcode features. Initially, the search process starts with a complete OPcode set. Subset generation in BE is guided by the wrapper-filter hybrid heuristic score. The combined score computation continues, and when the number of OPcode in BE process is significantly reduced compared to the total OPcode, the filter score component is weighted less than the wrapper score as given in equation (12): where $1 \leq u, v \geq 0$.

C. Training process in MR ANNIGMA

We adopt a single hidden layered Multi-Layer Perceptron (MLP) Network in the wrapper stage of our MR-ANNIGMA model. An n-fold cross validation approach has been applied in the training of the network. The

$$Combined\ Score : MR\_ANNIGMA(F_i) = u * \frac{I(F_i;c)}{maximum_{(F_i \in S)} I(F_i;c))} + v * ANNIGMA(F_i)_{average} \quad (12)$$

TABLE I.
DATA SET DESCRIPTION

Type	Qty	Max. Size (KB)	Min. Size (KB)	Avg. Size (KB)
Benign	15,480	109,850	0.8	32,039
Virus	17,509	546	1.9	142
Worm	10,406	13,688	1.6	680
Rootkit	270	570	2.8	380
Backdoor	6,689	1,299	2.4	685
Constructor	1,039	77,662	0.9	1,193
Exploit	1,207	22,746	0.5	375
Flooder	905	16,709	1.0	1,397
Trojan	13,201	17,810	0.7	1,819

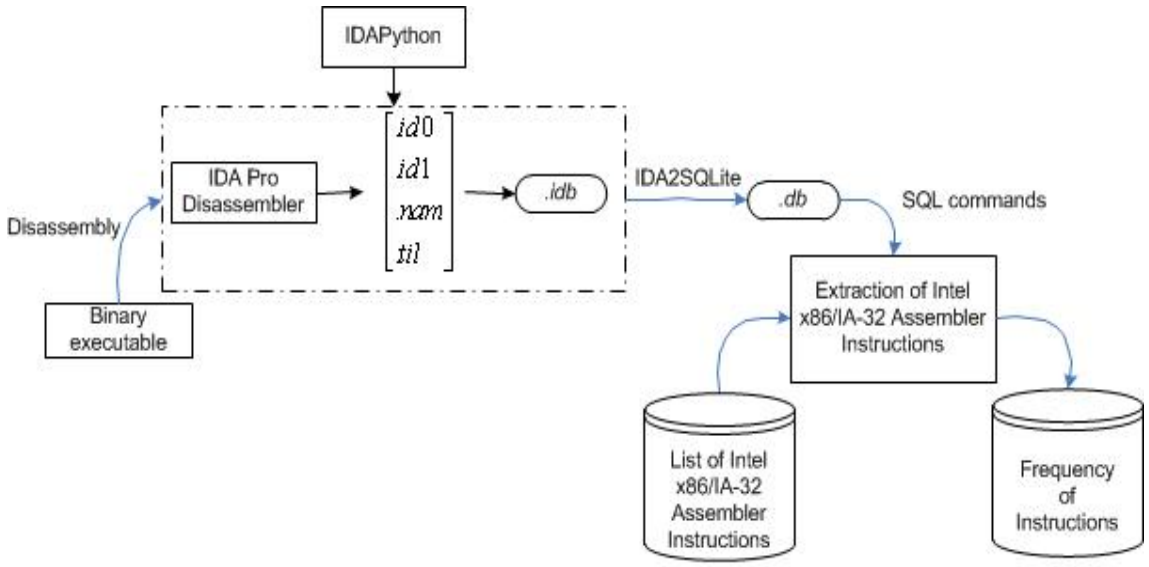

Figure 5. Automated the inspection of the op-code frequency statistics

evaluation criterion of OPcode subset is based on the average prediction accuracy over n-fold of the wrapper (MLP network). In Algorithm-1, steps-1 to 5 compute filter score of current feature subset. Step-6 to 11 compute the average accuracy over n-folds and compute the wrapper score for the current subset of OPcodes. Step-12 to 14 computes the hybrid scores and the OPcodes are ranked based on their combined score. Step-15 to 16 would then generate a new subset based on the OPcode ranking and would keep a record of evaluated OPcode subsets with their accuracy. In every iteration, the BE processes in MR-ANNIGMA would update the MR and ANNIGMA and the combined score. This score guides the subset generation. The BE continues until a single OPcode is remaining in the current subset. The subset with the highest accuracies or close to the highest accuracies with fewer OPcode set is then chosen as the final OPcode subset for the malware detector.

IV. EXPERIMENTS AND RESULTS

A. Data Set

We have gathered 66,703 executable files in total consisting of 51,223 recent Malware datasets and the remaining being benign datasets as shown in table-I. Such large malware datasets with obfuscated and unknown malware used in this research study have been collected from honeynet project, VX heavens [32] and other sources. The 15,480 benign datasets include: Application software such as Databases, Educational software, Mathematical software, Image editing, Spreadsheet, Word processing, Decision making software, Internet Browser, Email software and Programming language software. Both (Malware, Benign) have been uniquely named according to their MD5 hash value.

B. Disassemble code executable

We have automated the inspection of the op-code frequency statistics and have given a preliminary assessment of its frequencies which is used for detection and

differentiation among different malicious and benign files. The steps have been presented in the Fig.5. IDA Pro Dissasember [33] has been selected to view and analyse the PE files. Our approach has been tested directly on the PE format files to compare OPcode distributions within malicious and benign files.

In the first step, we develop fully automated method for disassemble the binary executable for OPcode frequency statistics from malicious and benign binaries executable. Fig.5 shows the system architecture of such an automated process. Python programming language have been used to implement this step and to automate the process of disassembly of all the binaries. Collected samples were pre-processed for anomaly testing. In order to translate a program into an equivalent high-level-language based on the binary content, we have used the most reliable disassembly tool used for static analysis, namely, Interactive Disassembler Pro (IDA Pro) [33] since it can disassemble all types of non-executable and executable files (such as ELF, EXE, PE, etc.). Also, we have selected IDA Pro as a component of the automation process because it automatically recognizes instruction names of the OPcodes for various compilers and can be further extended with our Python programs. IDA Pro loads the selected file into memory to analyse the relevant program portion to create an IDA database whose components are stored in four files including 1) .id0 that contains the content of a B-tree-style database, 2) .id1 that contains flags describing each program byte, 3) .nam that contains index information related to program locations, and 4) .til that is used to store information concerning local type definitions to a given database. IDA Pro generates the IDA database files into a single IDB file (.idb) by disassembling and analysing the binary of the file. IDAPython [34] is used to integrate the Python language which allows scripts to run in IDA Pro to automate the process. We also used the C library SQLite [21] that implements a self-contained SQL database engine with our python program to entitle us to convert the binary executable to a database. Therefore, we developed IDA Pro in SQLite name 'IDA2SQLite' plug-in to store the initial analysis results with the extension (.db). Our developed plugin generates eight tables of information, each table contain information about the executable namely Blocks, Functions, Instructions, Names, Maps, Stacks, Segments, and TargetBinaries. Each of these tables contains different information about the binary content. For the analysis of the features, we have run the SQL commands through our Python program to compute the machine OPcodes frequency statistics.

V. Results and discussion

In this section, we describe performance results of the proposed hybrid Wrapper-Filter model, $MR\_ANNIGMA$, and analysis of the results, including comparison with MR and ANNIGMA classifiers that we tested individually. The proposed MR-ANNIGMA based signature-free approach has been tested on a set of real and recent malware samples for detecting more unknown malwares. We have two types of datasets, namely malwares, and Executable benign files or 'good wares'.

In our dataset for malware and benign files, the aggregate malware dataset yielded a total of $48,629,512$ OPcodes and the aggregate benign dataset yielded a total of $405,942$ OPcodes. The experiment was run on a total of 590 different OPcodes collected from Intel, but for the analysis part the OPcodes that have been found in our sample binaries was only just considered which are in total of 80 OPcodes. In Fig.(6), analysis show that the top 13 listings for both malware and benign are identical (ADD/ CALL/ CMP/ JMP/ JNZ/ JZ/ LEA/ MOV/ POP/ PUSH/ RETN/ TEST/ XOR) Many of the new OPcodes were not used at all in all our samples such as: Move Data from String to String (MOVS/ MOVSB/ MOVSW/ MOVSD/ MOVSQ), Compare String Operand (CMPS/ CMPSB/ CMPSW/ CMPSD/ CMPSQ), Load Machine Status Word (LMSW), Load String (LODS/ LODSB/ LODSW/ LODSD/ LODSQ) , Repeat String Operation Prefix (REP/ REPE/ REPZ/ REPNE/ REPNZ) , Scan String (SCAS/ SCASB/ SCASW/ SCASD). Fig.(6) shows the top 12 listings of OPcodes for both malware and benign executable and their frequencies of use. From Fig.(6) it is evident that the percentages of using the most popular set of OPcodes in both malware and clean binaries are different. This shows that obfuscations are targeted on using less popular OPcodes which differ statistically and significantly to a great extent. For example, the dead-code insertion NOPs (No Operation Performed) which inserts operation that do nothing exhibits a very high frequency of use in malware as compared to benign, NOP used in Malware was 98.8% compared to NOP used in benign 1.2%. Similar to the OPcode (JMP) and (JZ) which used to shuffle the binary content, in malware (84%) compare to benign (16%). This is clear evidence that the malware authors are using obfuscation methods to transform malcode into a new code without affecting the original functionality or purpose. Thereby making it very difficult to reserve engineer and decipher the signature successfully. This phenomenon is effectively utilised in our model to filter the most significant OPcodes.

We evaluate the performance of the individual classifiers incluidng MR and ANNIGMA, as well as our proposed hybrid Wrapper Filter MR-ANNIGMA classifier using commonly adopted metrics: Accuracy, area under ROC curve and the OPcode feature subset achieved through the training process. Experimental results of selected OPcodes have been illustrated in Figures-(7) to (11) and Tables-II to Table-IV. Table-II and Table-III provides the score for filter approach (MR), wrapper approach (ANNIGMA), and our new hybrid Wrapper-Filter model (MR- ANNIGMA). The backward elimination (BE) process starts with all the OPcodes and accuracies of different iterations of the BE process for all three algorithms are calculated. Table 4 provides the accuracies for selected OPcodes. We observe that on an average, ANNIGMA

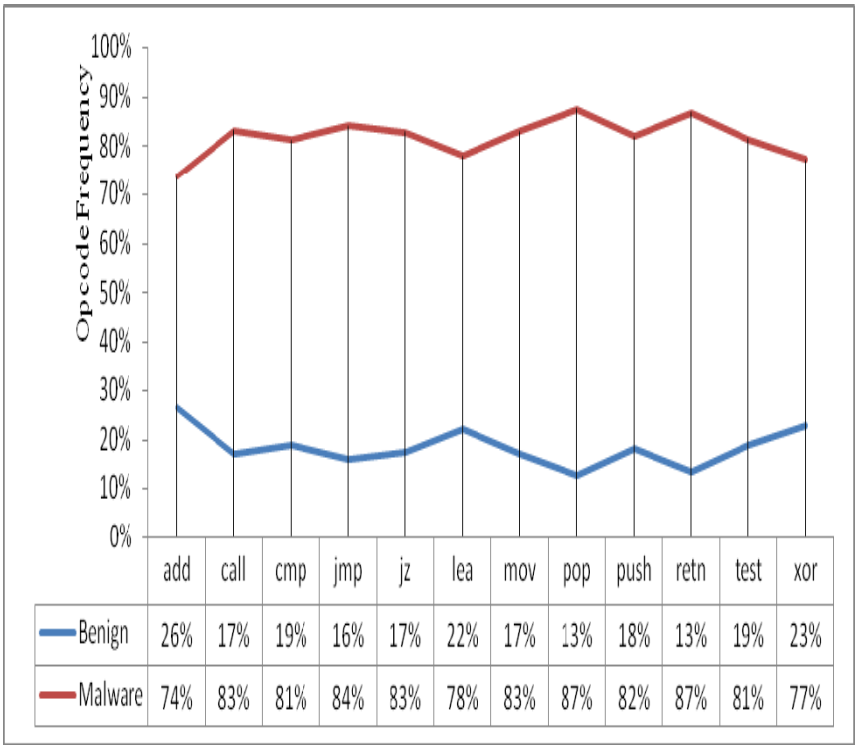

Figure 6. OPcode frequency statistics

TABLE II.
SCORE OF OPCODE. SL NO: SERIAL NUMBER; ANNIG.=ANNIGMA

Sl No	OP code	ANNIG.	MR	Sl No	OP Code	ANNIG.	MR	Sl No	OP code	ANNIG.	MR .
1	AAA	0.174914	0.010719	28	INT	0.045247	0.042276	55	PUSHF	0.038037	0.037304
2	AAD	0.313344	0.009664	29	INTO	0.291025	0.006226	56	RCL	0.035991	0.520724
3	AAM	0.023984	0.021478	30	IRET	0.105444	0.014621	57	RCR	0.009614	0.090842
4	AAS	0.050952	0.015322	31	JA	0.015265	0.691031	58	RETF	0.097164	0.185096
5	ADC	0.011652	0.473899	32	JB	0.00551	0.813973	59	RETN	0.033747	0.943082
6	ADD	0.006098	0.854097	33	JBE	0.005516	0.702876	60	ROL	0.023514	0.429159
7	AND	0.004548	0.696157	34	JMP	0.012144	0.953505	61	ROR	0.019423	0.525929
8	ARPL	0.144947	0.0073	35	JNB	0.019942	0.732964	62	SAHF	0.053646	0.259657
9	CALL	0.121646	0.986869	36	JNZ	0.036437	0.962211	63	SAL	0.055511	0.006703
10	CBW	0.244401	0.017806	37	JZ	0.007424	0.982198	64	SAR	0.007614	0.491719
11	CLC	0.030442	0.024569	38	LAR	0.042301	0.002855	65	SBB	0.013093	0.254259
12	CLD	0.00904	0.444688	39	LEA	0.078249	0.962276	66	SGDT	0.002902	0.000407
13	CLI	0.176006	0.026297	40	LGDT	0.012764	0.00133	67	SHL	0.063011	0.466893
14	CLTS	0.006739	0.000407	41	LIDT	0.005273	0.000814	68	SHR	0.025327	0.624252
15	CMC	0.076959	0.117061	42	LOOP	0.061564	0.227182	69	SIDT	0.01152	0.00163
16	CMP	0.015737	0.978182	43	LSL	0.100197	0.003674	70	STC	0.031549	0.040135
17	CWD	0.195856	0.018228	44	LTR	0.007535	0.000407	71	STD	0.005277	0.410024
18	DAA	0.205789	0.01508	45	MOV	0.009863	1	72	STI	0.051603	0.024678
19	DAC	0.092419	0.00812	46	MUL	0.011767	0.109671	73	STR	0.009569	0.000407
20	DEC	0.006268	0.837359	47	NEG	0.012699	0.585016	74	SUB	0.003248	0.774897
21	DIV	0.007255	0.553247	48	NOP	0.057607	0.110791	75	TEST	0.006226	0.927154
22	FLDCW	0.005791	0.302102	49	NOT	0.003147	0.307726	76	VERR	0.015194	0.00163
23	HLT	0.790823	0.00516	50	OR	0.012643	0.681364	77	WAIT	0.370241	0.295436
24	IDIV	0.031867	0.525841	51	OUT	0.129565	0.019885	78	XCHG	0.005536	0.374692
25	IMUL	0.02052	0.364124	52	POP	0.017314	0.988841	79	XLAT	0.054843	0.017447
26	IN	0.305143	0.020618	53	POPF	0.027232	0.029645	80	XOR	0.023298	0.824443
27	INC	0.009938	0.886538	54	PUSH	0.02489	0.999034				

TABLE III.
SCORE OF OPCODE. SL NO: SERIAL NUMBER; ANNIG.=ANNIGMA

Sl No	OP code	MR ANNIG.	Sl No	OP code	MR ANNIG.	Sl No	OP code	MR ANNIG.	Sl No	OP code	MR ANNIG.
9	CALL	1.108515	31	JA	0.706296	62	SAHF	0.313303	57	RCR	0.100456
39	LEA	1.040525	7	AND	0.700705	49	NOT	0.310873	28	INT	0.087523
54	PUSH	1.023924	50	OR	0.694006	22	FLDCW	0.307893	72	STI	0.076282
45	MOV	1.009863	77	WAIT	0.665678	29	INTO	0.297251	55	PUSHF	0.075342
52	POP	1.006156	68	SHR	0.649579	42	LOOP	0.288747	79	XLAT	0.07229
36	JNZ	0.998648	47	NEG	0.597715	58	RETF	0.28226	70	STC	0.071684
16	CMP	0.993919	21	DIV	0.560502	65	SBB	0.267351	4	AAS	0.066275
37	JZ	0.989621	24	IDIV	0.557708	10	CBW	0.262207	63	SAL	0.062214
59	RETN	0.976829	56	RCL	0.556715	18	DAA	0.220869	53	POPF	0.056877
34	JMP	0.965649	61	ROR	0.545352	17	CWD	0.214083	11	CLC	0.055011
75	TEST	0.93338	67	SHL	0.529904	13	CLI	0.202303	3	AAM	0.045463
27	INC	0.896476	64	SAR	0.499333	15	CMC	0.19402	38	LAR	0.045157
6	ADD	0.860196	5	ADC	0.485551	1	AAA	0.185634	76	VERR	0.016824
80	XOR	0.847741	12	CLD	0.453728	48	NOP	0.168398	40	LGDT	0.014093
20	DEC	0.843627	60	ROL	0.452674	8	ARPL	0.152247	69	SIDT	0.013149
32	JB	0.819483	71	STD	0.415301	51	OUT	0.149451	73	STR	0.009975
23	HLT	0.795983	25	IMUL	0.384644	46	MUL	0.121438	44	LTR	0.007942
74	SUB	0.778144	78	XCHG	0.380228	30	IRET	0.120065	14	CLTS	0.007146
35	JNB	0.752906	26	IN	0.325761	43	LSL	0.103872	41	LIDT	0.006087
33	JBE	0.708391	2	AAD	0.323008	19	DAC	0.100539	66	SGDT	0.003309

TABLE IV.
ACCURACIES FOR ANNIGMA, MR AND MR-ANNIGMA IN DIFFERENT BE ITERATIONS. SIZE IS THE SUBSET SIZE IN PARTICULAR ITERATION. ANNIG.=ANNIGMA

size	ANNIG.	MR	MR ANNIG.	size	ANNIG.	MR ANNIG.	MR
80	96.694	96.884	96.816	19	96.572	97.013	97.346
37	96.483	97.549	97.536	18	96.361	96.307	96.999
36	96.483	97.196	97.366	17	96.47	96.253	96.979
35	96.477	97.447	97.325	16	96.463	96.456	96.775
34	96.531	97.305	97.284	15	96.49	96.599	97.054
33	96.382	97.42	97.407	14	96.463	96.497	96.585
32	96.497	97.515	97.475	13	95.669	96.382	96.497
31	96.538	96.986	96.925	12	95.771	96.463	96.511
30	96.463	97.013	96.897	11	95.866	97.06	96.042
29	96.443	97.149	97.196	10	95.655	96.843	95.757
28	96.429	97.169	97.135	9	95.662	96.802	95.995
27	96.327	97.23	97.223	8	95.73	96.857	95.594
26	96.531	97.115	97.481	7	95.791	96.66	95.743
25	96.219	96.938	97.23	6	95.567	96.762	95.648
24	96.361	96.877	97.183	5	95.642	95.94	95.221
23	96.151	96.986	97.108	4	89.669	90.117	91.112
22	96.28	97.047	97.115	3	89.031	88.997	89.214
21	96.388	97.386	97.194	2	79.092	77.037	78.139
20	96.395	96.85	97.237	1	76.555	77.569	75.793

achieves accuracies of about 96.4%, MR achieves accuracies of 96% and MR-ANNIGMA achieves about 97%. Hence, in terms of accuracy, all the three feature selection methods have achieved quite high performance levels when compared to the recent achievements of only about 85% accuracy [35]. This demonstrates that the proposed feature selection method has the capabilities to detect obfuscation patterns in malware. In particular, our MR-ANNIGMA has the benefit of achieving highest accuracy.

The next performance measure is how fast the learning algorithm is able to discard unnecessary OPcode to arrive at a minimal set of relevant OPcode for classification of malware. In the BE iteration process, the OPcodes are sorted according to their score, and in second iteration, the OPcode with lowest score is discarded and then the subset is evaluated. We have provided here intermediate score computation for a number of BE iterations in Fig.(7) to (11) from a total of $m-1$ iterations as an example. When the total OPcodes=28 is reached in Fig.(7), the BE process re-computes all OPcode scores (Fig.(8)) resulting in ANNIGMA achieves the lowest score for OPcode 6, MR achieves lowest score for OPcode 19, and our hybrid Wrapper-Filter achieves the lowest score for OPcode 74. Therefore, in this iteration, OPcode 74 is eliminated and MR-ANNIGMA achieves an accuracy of 97.203%.

In the next cycle, MR-ANNIGMA eliminates OPcode 58 due to its lowest combined score as shown in Fig.(8) and the accuracy of hybrid increases to 97.434%. In Fig.(9) when total OPcode is 17, MR-ANNIGMA eliminates OPcode 22 and as shown in Fig.(10), OPcode 90 is eliminated when a total of 16 OPcodes is reached (due their lowest combined score). Here, MR-ANNIGMA achieves an accuracy of 97.434%. In the next cycle, BE process eliminates OPcode-97 for lowest combined score

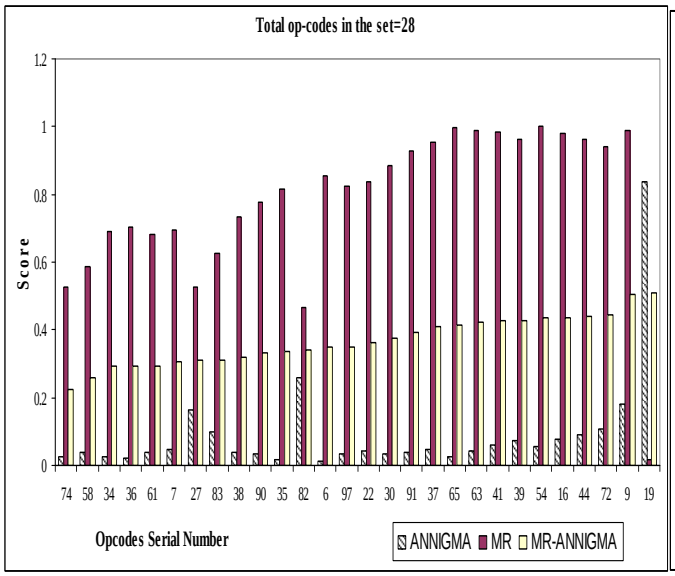

Figure 7. Score of Opcodes when total copcode is 28

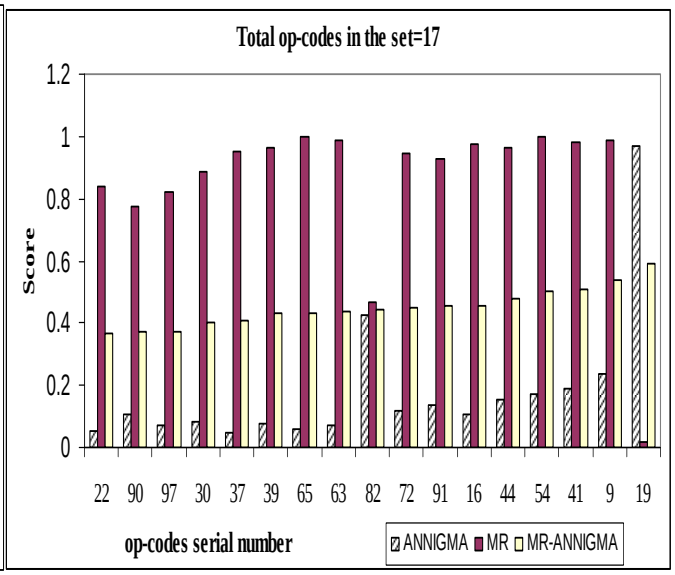

Figure 9. Score of Opcodes when total copcode is 17

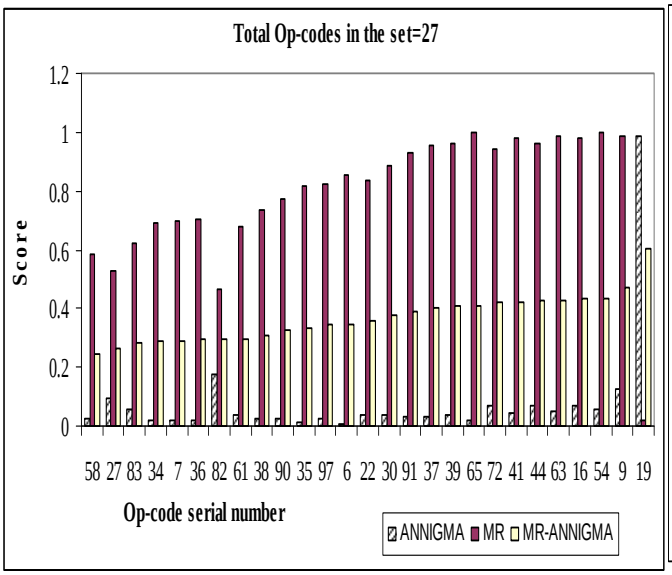

Figure 8. Score of Opcodes when total copcode is 27

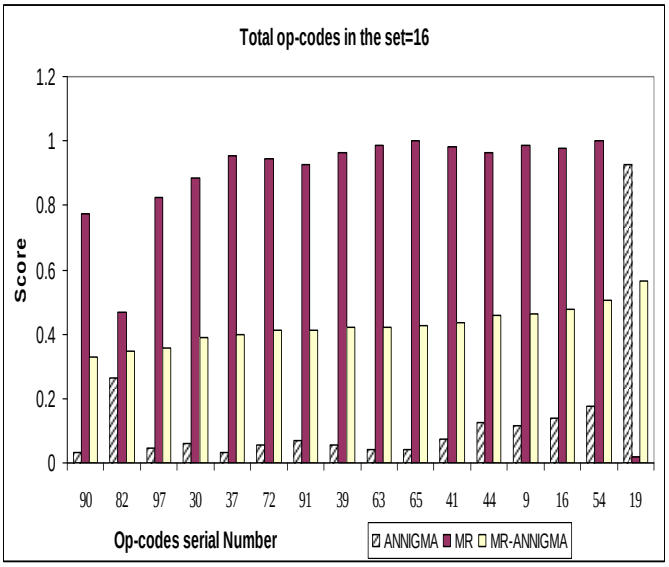

Figure 10. Score of Opcodes when total copcode is 16

as shown in Fig.(11) and arrives at a smallest set of 15 significant OPcodes. MR-ANNIGMA achieves (97.529%) accuracies as given in Table-IV with smallest set OPcodes (15 only) which is the best accuracy achieved through the iterations. Therefore, OPcode set-15 has been considered as the final and most significant OPcode set for MR-ANNIGMA.

The final OPcode sets from all three algorithms (ANNIGMA, MR and MR-ANNIGMA) have been used in a 10-fold cross validation set. The class discriminative performance of the most significant OPcode sets from all three algorithms has been tested by varying the NN's threshold values of the output node in the cross-validation. Then average sensitivity and specificity over 10-fold have been used to produce the receiver operating characteristics (ROC) curve for each algorithm which have been presented in Fig.(12). Among all three algorithms (MR, ANNIGMA and MR-ANNIGMA), the ROC curve of MR-ANNIGMA has achieved the highest sensitivity with the highest specificity. Through all the three performance measures used to evaluate our proposed hybrid MR-ANNIGMA model, we have clearly demonstrated its efficacy for malware detection.

VI. Conclusions and Future Work

With obfuscation techniques such as packer, polymorphism and metamorphism, rapidly evolving more recent malware code have been able to evade available detection methods. There remains the hope that obfuscation patterns could be extracted as features and feature selection methods could be applied to classify an executable file as either malware or benign. In this paper, we have examined

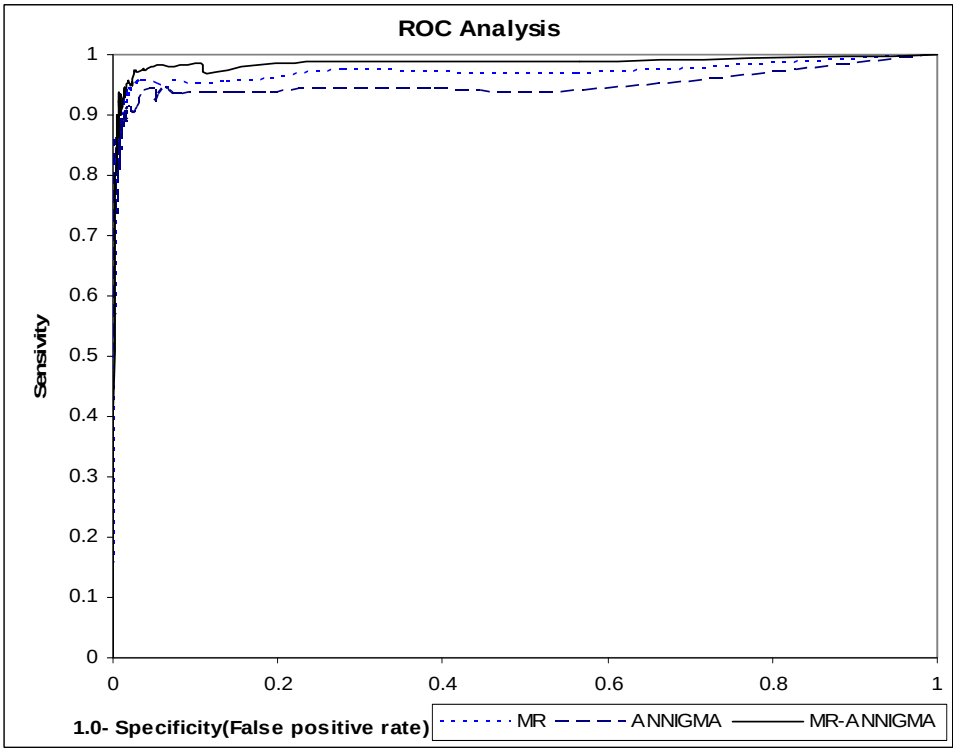

Figure 12. Receiver Operating Characteristics analysis

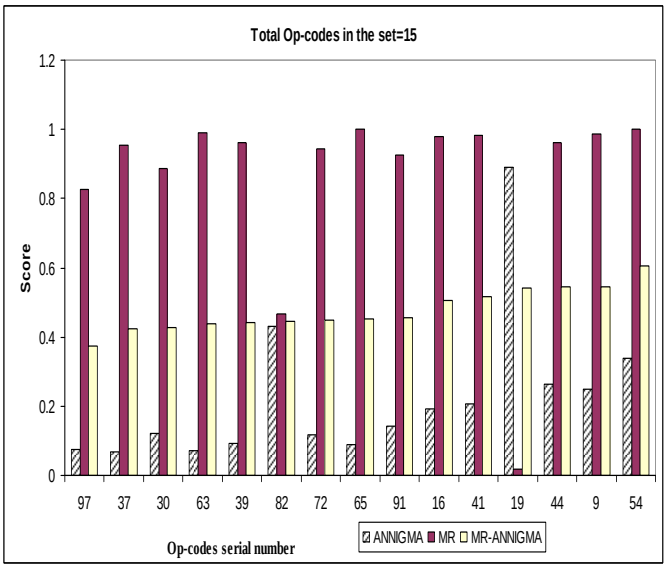

Figure 11. Score of Opcodes when total copcode is 15

two different feature selection methods: the filter approach (MR) and wrapper approach (ANNIGMA) and then proposed a hybrid wrapper-filter approach. This approach exploits the strengths of each of these two approaches and selects the OPcode features of an executable file that are obfuscated to accurately and more efficiently identify malware.

One of the main contributions of this research was the development of a fully-automated signature-free method to unpack, de-obfuscate and reverse engineer the binary executable files without the need for manual inspection of assembly codes. In addition the method enabled the OPcode features to be identified also without the need for further manual inspection. The novelty of our approach is due to the integration of the knowledge (from the intrinsic characteristics of data) obtained by the filter and wrapper approach and, combined with the wrapper's heuristic score and the filter's ranking score compiled in the wrapper stage of the hybrid. To the best of our knowledge, this approach is new and has not yet been explored in the malware literature. Anther novelty is that it is signature-free and thus able detect the malware variants which evade detection from signature-based approaches. The combined heuristics in the hybrid model takes advantage of the complementary properties of both filter and wrapper heuristics and uses these to efficiently guide the wrapper (sub-routines) to find the optimal and most compact OPcode subsets.

Three important performance measures, namely: accuracy, compact feature sets (in aid of execution speed), and area under ROC (a measure of false and true positives) were employed to compare our model with the individual classifiers, MR and ANNIGMA. Experimental results on real world malware and benign datasets show edthat among three approaches MR, ANNIGMA and MR-ANNIGMA, our proposed hybrid MR-ANNIGMA outperforms the independent wrapper and filter methods and produce accuracies of 97.53%. The MR-ANNIGMA also produced a more compact Opcode subset (e.g. only 15 OPcode features) as well as higher area under the ROC

curve close to 1.0. This demonstrates the significance of the hybridization approach and potential effectiveness in real world malware detection.

Further work could include the examination of other wrapper approaches and rule-generation processes along with investigations of the dynamic features of malware. Another challenge would be to analysis the extent that the OPcode frequency statistic may be skewed by the type of packer used by malware writers. Therefore, an automatic unpacking process for packed executable files would be the target of future research.

ACKNOWLEDGEMENTS

This research is supported by the Australian Research Council grant $DP01096833$ and Australian Criminological Research Council Grant CRG $1312-13$ at Australian National University (ANU).

REFERENCES

[1] S. McCombie, J. Pieprzyk, and P. Watters, "Cybercrime attribution: An eastern european case study,," in *The 7th Australian Digital Forensics Conference*, Perth, Australia, 2009, pp. 41–51.

[2] S. Ghosh and E. Turrini, *Cybercrimes: A Multidisciplinary Analysis*. Springer Verlag, 2010.

[3] S. E. Security, "Symantec internet security threat report: Trends for 2010," Symantec Enterprise Security, Tech. Rep., 2011.

[4] E. Konstantinou and S. Wolthusen, *Metamorphic virus: Analysis and detection*. London: Royal Holloway, 2008.

[5] M. Sharif, V. Yegneswaran, H. Saidi, P. Porras, and L. W, "Eureka: A framework for enabling static malware analysis," in *Computer Security - ESORICS 2008*, S. Jajodia and J. Lopez, Eds. Springer Berlin / Heidelberg, 2008, vol. 5283, pp. 481–500.

[6] J. C. Rabek, R. I. Khazan, S. M. Lewandowski, and R. K. Cunningham, "Detection of injected, dynamically generated, and obfuscated malicious code," in *2003 ACM workshop on rapid malcode*, Washington, DC, USA, 2003.

[7] I. You and K. Kim, "Malware obfuscation techniques: A brief survey," in *International Conference on Broadband, Wireless Computing, Communication and Applications*, 2010, pp. 297–300.

[8] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on opcode patterns," *Security Informatics*, vol. 1, pp. 1–22, 2012.

[9] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns," in *The 12th conference on USENIX Security Symposium*, Washington, DC, USA, 2003, pp. 12–21.

[10] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *Journal of Machine Learning Research*, vol. 7, pp. 2721–2744, 2006.

[11] M. Alazab, P. Watters, S. Venkatraman, and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of api call signatures," in *In Proc. Australasian Data Mining Conference, Ballarat, Australia*, P. S. A. O. K.-L. C. P. CRPIT, Vamplew and P. J. E. Kennedy, Eds., vol. 121, pp. 171 – 18.

[12] M. Alazab, R. Layton, S. Venkatraman, and P. Watters, "Malware detection based on structural and behavioural features of api calls," in *The 1st International Cyber Resilience Conference*, Perth, WA Australia, 2010, pp. 1–10.

[13] C. Wang, J. Pang, R. Zhao, W. Fu, and X. Liu, "Malware detection based on suspicious behavior identification," in *First International Workshop on Education Technology and Computer Science*, Wuhan, Hubei, P. R. China, 2009, pp. 198–202.

[14] M. Shankarapani, K. Kancherla, S. Ramammoorthy, and S. Mukkamala, "Kernel machines for malware classification and similarity analysis," in *2010 International Joint Conference on Neural Networks*, Barcelona, Spain, 2010, pp. 1–6.

[15] S. Cesare and Y. Xiang, "Classification of malware using structured control flow," in *8th Australasian Symposium on Parallel and Distributed Computing*, Brisbane, Australia, 2010, pp. 61–70.

[16] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, USA, 2001, pp. 38–49.

[17] D. J. Malan and M. D. Smith, "Host-based detection of worms through peer-to-peer cooperation," in *Proceedings of the 2005 ACM workshop on Rapid malcode*, Fairfax, Virginai, USA, 2005.

[18] Y. Li and D. Ruppert, "On the asymptotics of penalized splines,," *Biometrik*, vol. 95, no. 2, pp. 415–436, 2008.

[19] M. Weber, D. Schmit, D. Geyer, and M. Schatz, "A toolkit for detecting and analyzing malicious software," in *18th IEEE Annual Computer Security Applications Conference*, Washington, DC, USA, 2002, pp. 423–431.

[20] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala, "Static analyzer of vicious executables (save)," in *20th Annual Computer Security Applications Conference*, Tucson, Arizona. USA, 2004, pp. 326–334.

[21] Y. Ye, T. Li, Q. Jiang, and Y. Wang, "Cimds: Adapting postprocessing techniques of associative classification for malware detection," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, pp. 298–307, 2010.

[22] R. Islam, R. Tian, L. Batten, and S. Versteeg, "Classification of malware based on string and function feature selection," in *Cybercrime and Trustworty Computing Workshop*, Ballarat, Victoria. Australia, 2010, pp. 9–17.

[23] *Intel 64 and IA-32 Architectures Software Developer's Manuals : Basic Architecture, Vol 1*, Intel, http://www.intel.com/content/www/us/en/processors/architectures- software- developer- manuals.html?, 2010.

[24] K. S. Balagani and V. V. Phoha, "On the feature selection criterion based on an approximation of multidimensional mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1342–1343, 2010.

[25] Q. Hu, J. Liu, and D. Yu, "Mixed feature selection based on granulation and approximation," *Journal of Knowledge-Based Systems*, vol. 21, pp. 294–304, 2008.

[26] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, pp. 246–271, 1997.

[27] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Machine Learning: Proceedings of the 11th International Conference*. San Fransisco, California. USA: Morgan Kaufmann Publishers, 1994, pp. 121–129.

[28] H. Wang, D. Bell, and F. Murtagh, "Axiomatic approach to feature subset selection based on relevance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 271–277, 1999.

[29] C.-N. Hsu, H.-J. Huang, and S. Dietrich, "The annigma-wrapper approach to fast feature selection for neural nets," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 32, pp. 207–212, 2002.

[30] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transaction on Pattern Analysis and Machine Intelligence,*, vol. 26, 2004.
[31] S. Huda, J. Yearwood, and A. Strainieri, "Hybrid wrapper-filter approaches for input feature selection using maximum relevance and artificial neural network input gain measurement approximation (annigma)," in *Fourth International Conference on Network and System Security*, Melbourne, Victoria. Australia, 2010, pp. 442–449.
[32] http://vx.netlux.org/. (2011, March) Vx heavens site. http://vx.netlux.org/. [Online]. Available: http://vx.netlux.org/
[33] *IDA. Pro Disassembler and Debugger*, 5th ed., IDA Pro, 2010.
[34] *IDAPython : Python Plugin for Interactive Disassembler Pro*, 1st ed., IDA Python, 2011.
[35] P. Watters and S. McCombie, "A methodology for analyzing the credential marketplace," *Journal of Money Laundering Control*, vol. 14, pp. 32–43, 2011.