A Comparison of the Classification of Disparate Malware Collected in Different Time Periods

[1]Rafiqul Islam, [2]Ronghua Tian, [2]Veelasha Moonsamy, [2]Lynn Batten
[1]School of Computing and Mathematics, Charles Sturt University, Australia.
[2]School of Information Technology, Deakin University, Melbourne, Australia
(mislam@csu.edu.au, tianronghua99@gmail.com, (v.moonsamy, lmbatten)@deakin.edu.au

Abstract- **It has been argued that an anti-virus strategy based on malware collected at a certain date, will not work at a later date because malware evolves rapidly and an anti-virus engine is then faced with a completely new type of executable not as amenable to detection as the first was.**

In this paper, we test this idea by collecting two sets of malware, the first from 2002 to 2007, the second from 2009 to 2010 to determine how well the anti-virus strategy we developed based on the earlier set [18] will do on the later set. This anti-virus strategy integrates dynamic and static features extracted from the executables to classify malware by distinguishing between families. We also perform another test, to investigate the same idea whereby we accumulate all the malware executables in the old and new dataset, separately, and apply a malware versus cleanware classification.

The resulting classification accuracies are very close for both datasets, with a difference of approximately 5.4% for both experiments, the older malware being more accurately classified than the newer malware. This leads us to conjecture that current anti-virus strategies can indeed be modified to deal effectively with new malware.

***Index Terms* - malware, classification, static, dynamic.**

I. Introduction

With the growth of the shadow Internet economy, malware is no longer simply used to damage, break or intrude on computer network systems, but now exists primarily as one of the important means used by criminals to make a profit. Malware has already become a global problem which has affected different parts of the world in different ways. The upsurge in the number of new samples collected by the anti-virus companies over the last few decades [4] clearly justifies the need for continuous deep analysis of malicious files in order to improve the efficacy of anti-virus software. So far, the researchers contributing within the malware field are unable to find an efficient detection approach which is capable of classifying malware and differentiating between malware and cleanware over a long period of time, as explained below.

The work in [15, 21, 13, 2, 1, 19, 14, 9] supports the argument that an anti-virus strategy which has been successful in a given time period will not work at a much later date; this, they argue, is due to changes in malware design which evolves with time and eventually becomes unrecognizable from the original form. Their work indicates that current techniques failed to find a distinctive pattern of malicious software which can be used to identify future malware with the level of accuracy required.

Despite the strong support in the literature for the idea that current detection methods will not easily detect future malware, in this paper, we demonstrate that it is possible to develop a malware detection strategy which maintains stable performance over an extended time period. We do so by considering two sets of malware, one collected during the period 2002 to 2007 (881 samples) and the other during the period 2009 to 2010 (1517 samples). All samples come from CA Technologies VET Zoo (www.ca.com) and all have been pre-classified as members of particular families. The key contribution of this paper is the provision of strong evidence that anti-virus techniques which work well on malware developed at a certain time may continue to be effective on malware developed at a much later time.

We do two different experiments in this paper to test our idea;

i) One for malware family classification to observe the results of two different time periods and
ii) The other for malware versus cleanware classification using a cumulative approach to observe the consistent performance of malware over time within the two different time periods.

The rest of the paper is organized as follows: Section 2 includes a summary of the related work on comparison of malware classification over a time period. In Section 3 we elaborate on the data preparation while Section 4 describes the classification process. Section 5 provides an analysis of the results and lastly, in Section 6 we discuss these results and present some ideas for future work.

II. Related Work

In this section, we review the literature on malware detection and classification which has emphasized a comparison of results over a time period.

More and more malware writers use various obfuscation technologies such as packing, encrypting or polymorphisms, to transform a malicious program into variants so that the latter cannot be detected by anti-virus detection engines. In practice, there is no system that can achieve 100% classification accuracy. Some researchers have achieved quite good results by testing different approaches on various data sets. We discuss some of these here.

Bailey et al. [1] describe malware behaviour in terms of system state changes (taken from event logs). They compare over 3000 pre-classified samples with over 3000 unclassified samples using a clustering technique to measure closeness. Testing about 8,000 malware samples collected over the period 2004 to 2007, they achieve an overall classification accuracy of almost 92%. In their study they use three different evaluation techniques: completeness, conciseness, and consistency. The authors mention that one limitation of their methodology is the failure to "detect fine-grained characteristics of the observed behaviours".

Zheng and Fang [20] propose a novel infrastructure for malware detection that can be implemented in a cloud system thus relieving a local end system of strenuous processing of suspicious files. They compare detection rates on malware sets which differ in age by up to 3 months, with diminishing levels of accuracy.

The authors of [6] use a combination of static and dynamic analysis to achieve a high level of malware accuracy (97%) over an eight year time period, demonstrating that including 'older' malware in the set for feature selection can assist in identifying 'new' malware.

The work of Roundy and Miller [12] showed that by applying a hybrid of dynamic and static analysis, the probability of correctly detecting malicious programs can be significantly increased. They incorporate static and dynamic methods to complement each other to make the detection and classification effective and robust to changes in malware evolution. The combined algorithm provides "analysis-guided instrumentation on obfuscated, packed and self-modifying binaries" [12, p.335]. The authors tested their algorithm on 200 malware sample and found that 33% of the malicious code analysed by the combined methods were not part of the dynamic execution trace reports and would not have been identified by dynamic analysis alone.

The study in [18] investigates malicious attacks on several websites by creating web honeypots and collecting 366 website-based malware executables over a period of five months. In their study, they carry out two investigations: (i) collect and analyse malware samples using 6 different anti-virus programs, and (ii) conduct the same experiment four months later using the updated versions of the 6 programs to determine their efficacy. The authors categorize the 366 malware samples under four types of malware: strings generator (110 samples), information investigator (191 samples), downloader (2 samples) and bot generator (63 samples). In the first investigation, the group of anti-virus programs detected only 3% and 13% of the first and fourth types of malware respectively. The second and third types of malware went undetected. In the second investigation, the detection rates improved to the extent that one anti-virus program successfully detected 74% of the malware from the dataset, but the detection percentages for the rest of the anti-virus software were under 50%. This work demonstrates that, with training on older malware, some anti-virus software can improve detection rates significantly.

The research conducted by Rosyid et al. [11] is focused on detecting malicious attack patterns in botnets. They use a set of sequential attacks which was recorded by a honeypot during the year 2009. The log files include the sequences of malware occurring in a particular time slot, where the duration of one slot is 20 minutes and the average number of slots in one day is 72. After extracting the malware sequences, they then apply the PrefixSpan algorithm to discover subsequence patterns. The authors extend their work by identifying attack patterns based on IP address and timestamp; this allows the authors to track the IP addresses and to determine the source and distribution pattern of the malware. Finally, the strength of a sequential attack pattern is represented by a confidence value. The highest confidence values for 2-pattern and 3-pattern are on average less than 50% and 57.93% respectively. The authors argue, therefore, that the signature of a single malware file is not enough to detect the complex variants of the attacks by botnets.

Another group of researchers [8], build their malware detection and classification framework based on comparisons of extracted strings using static analysis. The authors suggest a malware detection and classification system using the method of comparing different character strings, which in turn is used to identify and determine whether two instances are variants. The authors present a three-step methodology of extraction, refinement and comparison. In the extraction stage, all the printable strings existing in binary files are collected. The strings are then passed on to the refinement phase where the misleading characters that might affect the classification process are eliminated. The authors choose a set of 100 malware files to determine the type of characters to be discarded and apply the results to a set of 10,000 previously identified malware samples which were collected over several months. In the comparison phase, a modified Jaro-Wrinkler Distance [3] formula is used to measure the number of similar character strings between two different string groups and then, the Edit distance [3] formula is used to determine the similarity between two different character strings. The authors conclude that it is necessary to connect various static analysis methods in parallel, and in some cases, consider the need for the interlocking of a dynamic analysis method to create a more accurate, generalized system.

In [10], the authors study the resulting arms race between detection and evasion from the point of view of Google's Safe Browsing infrastructure, an operational web-malware detection system that serves hundreds of millions of users. Their study focuses on the four most prevalent detection techniques: Virtual Machine honeypots, Browser Emulation honeypots, Classification based on Domain Reputation, and Anti-Virus Engines. The analysis is based on the data collected over a four year period. They propose and verify the hypothesis that malware authors continue to pursue delivery mechanisms that can confuse malware detection systems. Their results show that none of the four detection techniques mentioned above are effective in isolation, but that adopting a multi-pronged approach can improve detection rates.

III. Data Preparation

In this section we provide a detailed explanation of the experiment. *Section A* describes the datasets that are used; *Section B* introduces our feature extraction method.

A. Datasets

The date used for each malware sample was the date assigned to the malware as it was received by CA Technologies VET Zoo. It is of interest to note that within the total of 2398 malware files, no family is represented in both Old and New Datasets. This was not done intentionally but is a true representation of the data in the Zoo.

In our experiment, we use the term 'Old Dataset' to refer to the malicious files collected between 2002 and 2007 and the term 'New Dataset' for those collected between 2009 and 2010. Under this section, we explain the data set preparation process for our two different tests: (i) Family by Family and (ii) Cumulative approach.

Family by Family (F/F) Malware Classification Test

Tables I and II list the families in the Old and New Datasets respectively, along with the numbers of files per family. The date used was the date assigned to the malware as it was received by CA Technologies VET Zoo.

Cumulative Cleanware versus Malware Test

For the second test, we accumulate the malware executables within the old and new datasets separately and carry out a malware versus cleanware classification.

The number of clean files used in our experiment is 541 and all are Win32 based executables.

Figure I shows the number of malware collected by month for the years 2002-2007. Figure II depicts the cumulative graph of the malware executables collected each month for the period 2002-2007. To generate the first malware group MG_1, we take the earliest-dated 10% of the malware executables which comprises 90 executables and includes the period October 2002 to March 2004. The second data group MG_2 contains the executables from October 2002 up to April 2004 inclusive; the process is repeated until all the executables in the dataset are incorporated into the malware groups. We end up with 43 groups, $MG_{1....}$ MG_{43} altogether, as shown in Figure II.

The Figures III and IV are similar to the first two figures, except that they show the malware collection trend and cumulated graph for the period 2009-2010. The generation of the malware groups for the executables in the new dataset is identical to that of the old dataset, as explained above and in this case we obtain 19 malware groups - $MG_{1...}$ $MG_{19,}$ as shown in Figure IV.

TABLE I. Malware files in Old Dataset.

Type	*Family*	*Number of files*
Virus	*Emerleox*	*75*
	Looked	*66*
	Agobot	*283*
Trojan	*Clagger*	*44*
	Alureon	*41*
	Bambo	*44*
	Boxed	*178*
	Robknot	*78*
	Robzips	*72*
	TOTAL	***881***

TABLE II. Malware files in New Dataset.

Type	*Family*	*Number of files*
Worm	*Frethog*	*174*
	SillyAutorun	*87*
Trojan	*Adclicker*	*65*
	Gamepass	*179*
	Banker	*47*
	SillyDI	*439*
	Vundo	*80*
	Bancos	*446*
	TOTAL	***1517***

B. Feature Extraction

In this section, we describe the type of features to be used in the classification process. For the first test, we extract the static and dynamic features for each malware executable within the old and new datasets. The two types of features are then combined to generate the integrated feature vectors, as explained below.

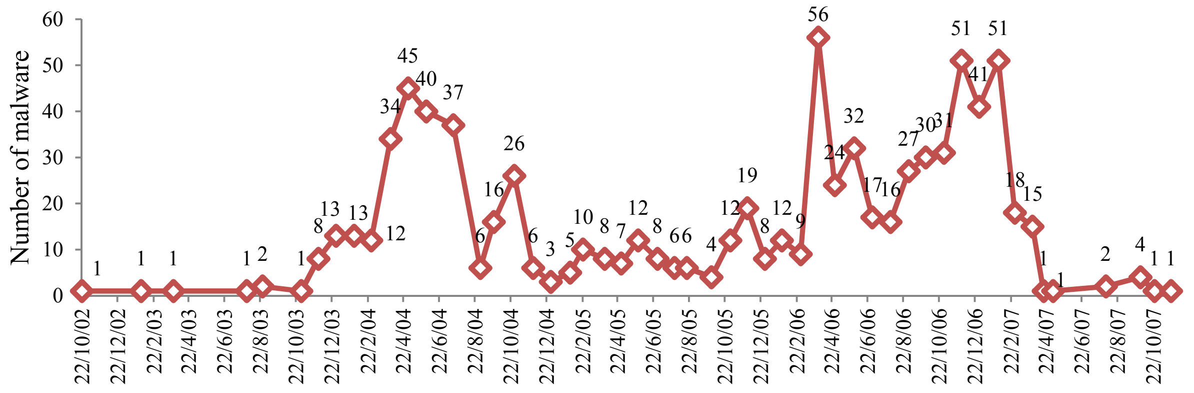

Figure I. Collection of malware executables from 2002 to 2007.

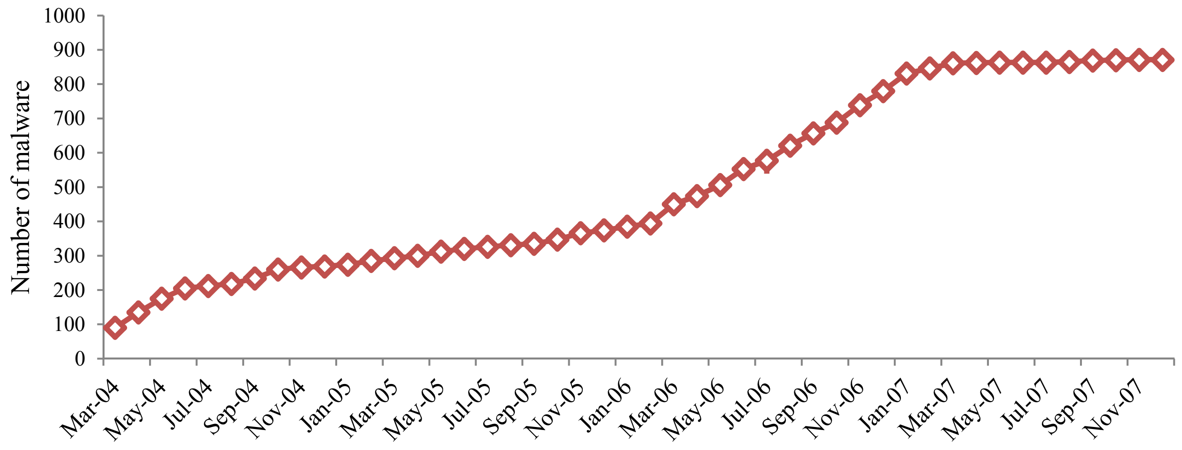

Figure II. Number of malware cumulated by month for the period 2002 to 2007.

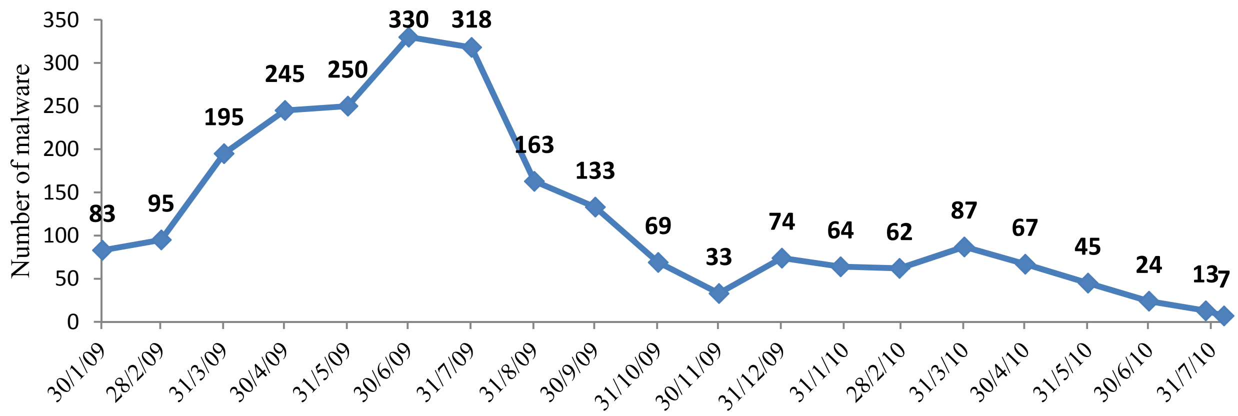

Figure III. Collection of malware executables from 2009 to 2010.

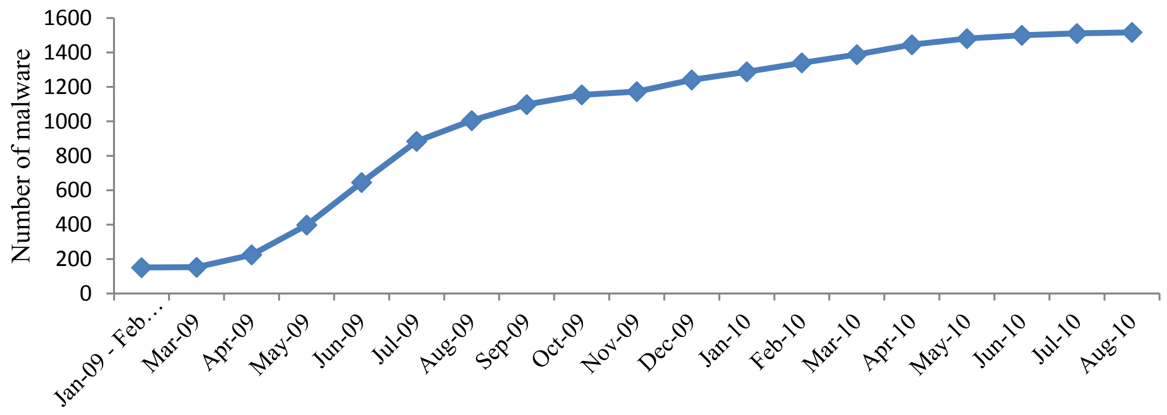

Figure IV. Number of malware cumulated by month for the period 2009 to 2010.

For the second test, the malware executables within the old and new dataset are split into groups. To generate the groups of malware, we start with the earliest malware and add month by month across the years until all data is in the final group. We take the earliest-dated 10% of the executables to form the first group. Ultimately, we end up with 43 groups in the old dataset and 19 groups in the new dataset.

Static Features

We unpack all malware using a command line anti-virus engine provided by CA Technologies. The software allows us to unpack the executables in batch mode which considerably reduces the unpacking time. We then consider two static features which are: (a) Function Length Frequency (FLF) and (b) Printable String Information (PSI).

In extracting the FLF features, we follow the methodology described in [16] using IDAPro to define the functions.

The lengths of each function (as a bitstring) are computed and these lengths are divided into 50 'bins' based on the methodology of [16]. The number 50 was chosen as an arbitrary value and for future work, we plan to test other values to compare results. Each bin is correlated with the total number of function lengths (with repeats) it contains and this number is called the *function length frequency* of any function which has a length in the bin.

As an example, we consider a malicious file, *F*, which includes 12 functions which have the following lengths (represented in bytes and in increasing order of size): 4, 5, 5, 12, 15, 15, 18, 19, 23, 45, 60 and 90. For the purpose of illustration, let us create 10 exponentially spaced bins based on the function length ranges. The distribution of frequencies across the bins is depicted in Table III.

TABLE III.
FLF BIN DISTRIBUTION.

Length of functions per bin	*FLF Vectors*
1-2	*0*
3-8	*3*
9-21	*5*
22-59	*2*
60-166	*2*
167-464	*0*
465-1291	*0*
1292-3593	*0*
3594-9999	*0*
>=10000	*0*

Finally, we select all the entries from the last column of the above table to form the following FLF vector for the file *F*: (0, 3, 5, 2, 2, 0, 0, 0, 0, 0). In our actual experiment, we fix the bin size to 50 based on a global list of all function length frequencies.

The second type of static feature, PSI, is extracted from the disassembled malicious executables. For each dataset, the printable strings are collected and combined to build a global string list. The example below explains the steps used in generating the PSI vector for a particular malicious executable.

Let us consider the following global string list consisting of 7 ***distinct*** strings: {"GetProcAddress", "RegQueryValueExW", "CreateFileW", "OpenFile", "FindFirstFileA", "FindNextFileA", "CopyMemory"}, where the order of strings within the list is fixed. Assume that the list of printable strings extracted from a particular executable file *F,* including repeats, is: {"GetProcAddress", "RegQueryValueExW", "CreateFileW", "RegQueryValueExW"}. We then track the presence of the strings in *F* against the global list using a '1' to indicate that a string is present (at least once) in the global string list and a '0' to denote the absence of the string. Table IV presents the corresponding information where we also include the total number of strings (with repeats) in the file in the first row. Hence, the corresponding PSI vector for *F* is (4,1,1,1,0,0,0,0).

TABLE IV.
PSI DATA FOR FILE *F*.

Number of strings	*4*
"GetProcAddress"	*1*
"RegQueryValueExW"	*1*
"CreateFileW"	*1*
"OpenFile"	*0*
"FindFirstFileA"	*0*
"FindNextFileA"	*0*
"CopyMemory"	*0*

Dynamic Features

The dynamic features are obtained from runtime behaviour of packed malware executables. We execute both datasets in a controlled virtual machine (VM) environment and record the behaviours in log files. To generate the log files, the executables were run for 30 seconds and then stopped. Below is an excerpt of a log file for executable *F*, where the API calls and the parameters are shown in italics:

2010/09/02 11:24, *RegQueryValueExW, Compositing*
2010/09/02 11:24, *RegOpenKeyExW, 0x54, Control Panel\Desktop*
2010/09/02 11:24, *RegQueryValueExW, LameButtonText*
2010/09/02 11:24, *LoadLibraryW, .\UxTheme.dll*
2010/09/02 11:24, *LoadLibraryExW, .\UxTheme.dll*

Let us assume that the global API list of distinct features is as follows: {"RegOpenKeyEx", "RegQueryValueExW", "Compositing", "RegOpenKeyExW", "0x54", "Control Panel\Desktop", "LameButtonText", "LoadLibraryW", ".\UxTheme.dll", "LoadLibraryExW", "MessageBoxW"}. We then compare the API features from the log file of *F* with the global list and count the frequencies of each item to generate the dynamic vector.

In this case, the dynamic vector for *F* is (0,2,1,1,1,1,1,1,2,1,0), drawn from the frequency column of Table V.

TABLE V.
DYNAMIC FEATURE DATA FOR F.

Dynamic Features in global list	***Frequency***
"RegOpenKeyEx"	*0*
"RegQueryValueExW"	*2*
"Compositing"	*1*
"RegOpenKeyExW"	*1*
"0x54"	*1*
"Control Panel\Desktop"	*1*
"LameButtonText"	*1*
"LoadLibraryW"	*1*
"".\UxTheme.dll"	*2*
"LoadLibraryExW"	*1*
"MessageBoxW"	*0*

Integrated Features

The integrated feature vector is a combination of the FLF, PSI and dynamic vectors. The motivation behind combining the different types of features described in the previous subsections is to prevent a malware writer from bypassing anti-virus technologies based on a single component of a malware file.

Hence, we collect the three vectors, FLF, PSI and Dynamic, and merge them into a single vector for each malicious executable, as shown in Figure V.

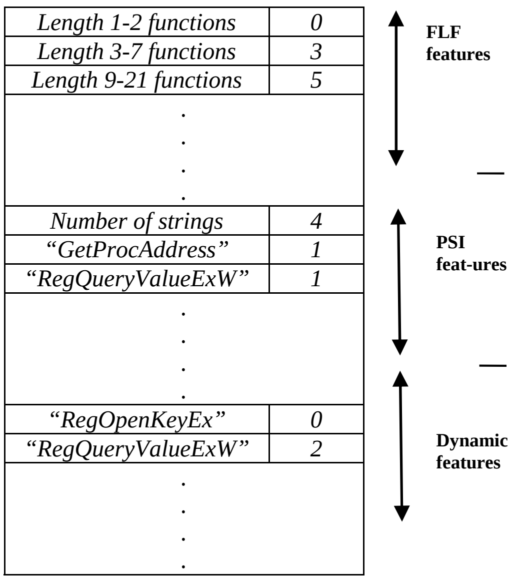

Figure V. Example of an integrated vector.

IV. Classification

For the classification process, we use four base classifiers from WEKA [5] representing a broad spectrum of classifier types: Sequential Minimal Optimization (SMO), Instance-Based (IB1), Decision Table (DT) and Random Forest (RF), and apply the statistical method known as 10-fold cross-validation [7] to classify the data.

In the cross-validation phase, we select files from one particular malware family and choose the same number of files at random from other families, using a random function. The files are then divided into 10 groups, where one group is used as a testing set and each of the remaining nine groups as a training set. The same classification procedure is applied to the F/F and cumulative tests.

A step-by-step breakdown of our methodology is given below:

1. Extract and generate the vectors for the FLF, PSI and dynamic features from all files in the Old Dataset, as described in Section 3.
2. Build the integrated vector from the 3 feature vectors in Step 1 and use these to generate the WEKA (arff) files.
3. Select a malware family, *M*, and from the remaining families, randomly choose a set of size |*M* |of malware files.
4. For the malware files chosen in Step 3, consider the set of corresponding arff files; break this set into 10 groups of equal size using the procedure, 'Making 10 groups of equal size', discussed below.
5. Select one of the constructed groups as a testing set and the union of the remaining nine as a training set.
6. Call WEKA libraries to train the classifiers using the training set.
7. Evaluate the classifiers using the testing set.
8. Repeat steps 3 to 7 for the remaining families. Then go to Step 9.
9. The first time, return to Step 1 and repeat for the New Dataset. The second time, exit.

Making 10 groups of equal size

In Step 4 from the above described methodology, we split the set of arff files, *A*, into 10 groups of equal size as follows:

- If $10 \mid |A|$, then each group has size $\frac{|A|}{10}$.
- If $10 \nmid |A|$, then we first generate 9 groups using the following equation,
  $$|A| = 9 * B + r, \text{ where } 0 \leq r < 9,$$
  whereby we take 9 groups of size *B* and place the remaining r arff files in a 10th group along with (*B*-*r*) randomly chosen arff files from *A*.

We also apply the meta classifier, AdaboostM1, to each of the base classifiers and rerun the tests. The meta classifier enhances the capabilities of the base classifiers by operating on the output of those classifiers. The classification accuracies produced by AdaboostM1 represent the correctness of each file (also referred to as an instance) classified by each of the four base classifiers, as described in [18]. In all cases, the boosted base classifiers perform better than the base classifier and therefore we present only the meta-classifier results in the next section.

V. Analysis of Results

In this section, we describe the empirical results using each of the four classifiers mentioned in Section 4. In our first test we use malware family classification and in the second test we use a cumulative approach to distinguish malware from cleanware. However in both of these tests we use an integrated feature generation approach. The following sub-sections present our experimental results.

A. Experimental results – F/F classification

The classification results for the Old and New Datasets are presented in Table VI and in Figure VI.

TABLE VI.
Weighted Average Accuracy (with Adaboost).

Boosted Classifiers	*Old Dataset*	*New Dataset*
SMO	*98.9%*	*83.9%*
IB1	*99.2%*	*90.7%*
DT	*99.2%*	*92.4%*
RF	*99.8%*	*94.4%*

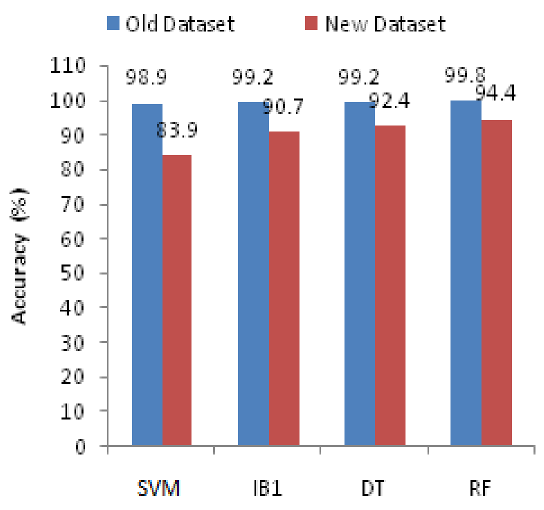

Figure VI. Weighted Average Accuracy.

Overall, RF outperformed the other classifiers with 99.8% classification accuracy for the Old Dataset and 94.4% for the New Dataset. These numbers also have the smallest difference (5.4%) across classifiers. We believe that the high accuracies obtained from RF can be attributed to the fact that this classifier runs capably on large datasets which incorporate diverse features, hence confirming the effectiveness of using integrated features.

On the other hand, the classifier SMO shows worst performance for the new dataset which is almost a 15% drop in accuracy compared to old dataset. The reason for this drop in accuracy between old and new datasets is not clear and remains for further investigation. However, in the classification process, while SMO builds its training model, it determines optimal tangential hyperplanes which can separate the data points (support vectors) into categories. Then SMO attempts to classify these support vectors into groups by determining on which side of a hyperplane a point of data lies. It could be that the time-consuming process of building the optimal hyperplanes with minimum support vectors, reduces the classification accuracy.

In addition, Figure VI indicates that the classification accuracy of the old malware dataset shows significant improvement compared to that of the new malware dataset for all classifiers. It was observed that our old experimental dataset consists of approximately 51% of malicious files from Trojan families and 49% from virus families while the new experimental dataset consists of 83% of malicious files from Trojan families and 17% from worm families but contains no files from virus families. Therefore, the dissimilarities of malware types between the old and new families could have had an impact on the drop in classification accuracy.

B. *Experimental results – malware versus cleanware using cumulative approach.*

Figure VII shows the malware classification results using the old data set (2002-2007) and Figure VIII those of the new dataset (2009-2010). Although, overall, the accuracy of the old dataset is better than that of the new dataset, there are clear differences in accuracy within each set over the respective time periods. In Figure VII, there is a noticeable drop in accuracy (approx 3-4%) between July 2006 and August 2006. Referring to Figure II, this time period coincides with a significant jump in the number of malware samples acquired and may explain the drop in accuracy at this point. This needs further investigation. We note that SMO and RF provide more consistent accuracy than IB1 and DT.

Figure VIII shows the empirical results of our new dataset collected over the time period 2009 to 2010. All four classifiers give consistent performance with comparable accuracy over the time period but, as in Figure VII, SMO and RF give slightly better results across the period than do DT and IB1. Figure IV indicates a sharp rise in the number of samples accumulated between April and September 2009, while the general accuracy as shown in Figure VIII does not appear to be affected by this.

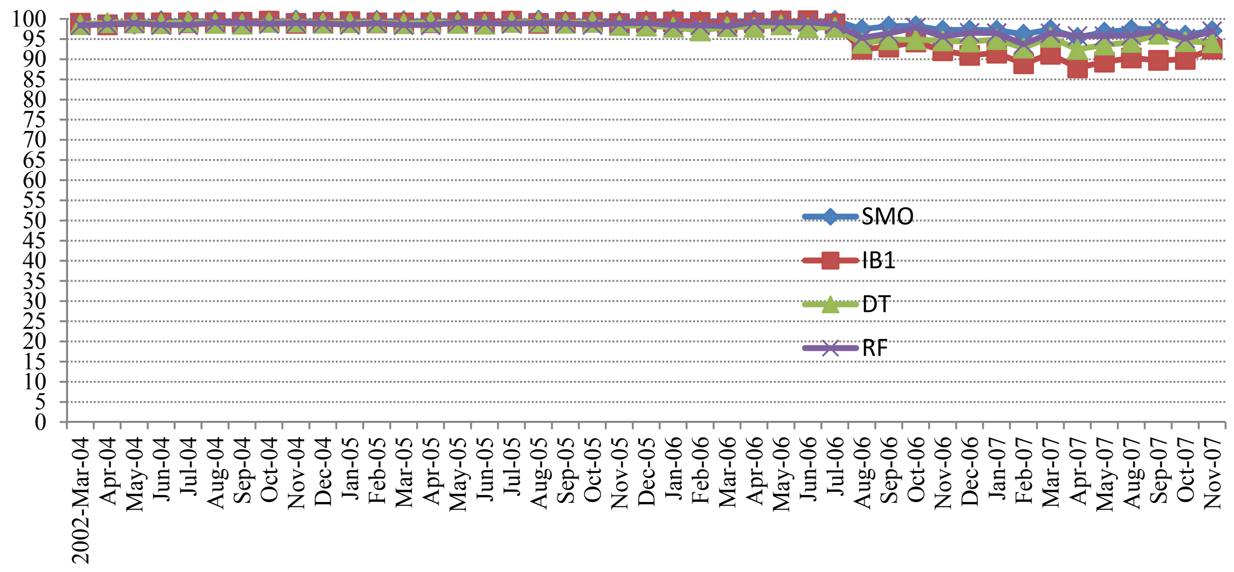

Figure VII. Experimental results based on cumulative approach – old dataset.

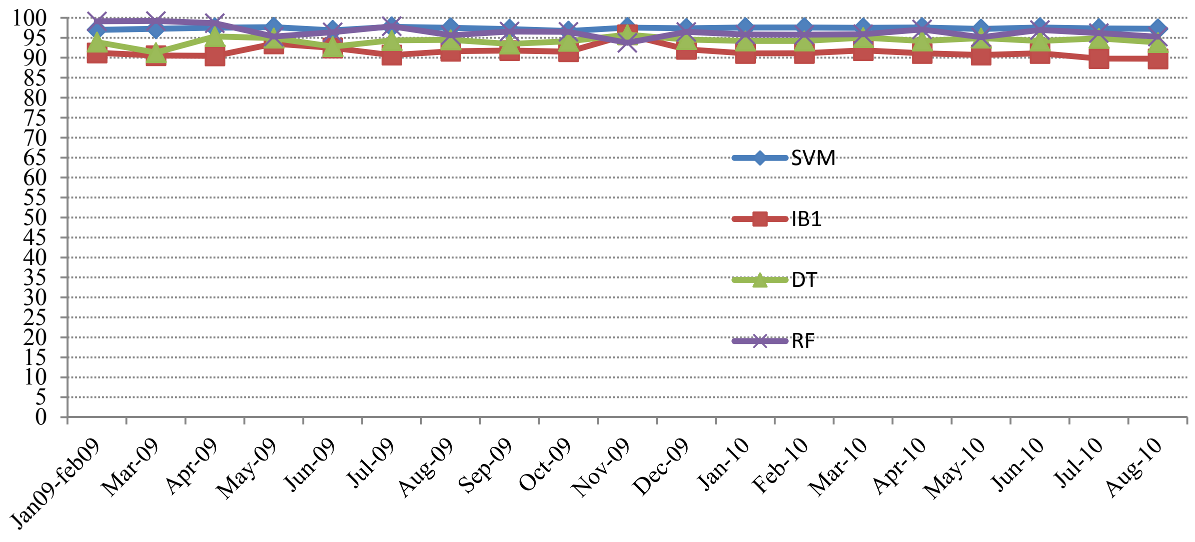

Figure VIII. Experimental results based on cumulative approach – new dataset.

The following tables summarize the average results of our cumulative tests using our old and new datasets. Table VII shows the average results of false positive (FP), false negative (FN) and accuracy (ACC) of the four classifiers on the old dataset. SMO gives the best average performance (classification accuracy 98.68% with lowest FP and FN) for all parameters compared to the others and IB1 performs worst in this test.

TABLE VII.
AVERAGE SUMMARY RESULTS (FOR OLD DATASET)

Classifier	*FP*	*FN*	*ACC*
SMO	*0.00339*	*0.02195*	*98.68114*
IB1	*0.005415*	*0.064578*	*96.47692*
DT	*0.011669*	*0.051574*	*97.27089*
RF	*0.012961*	*0.029558*	*97.88233*

Table VIII shows the average results of the new dataset. It is clear that the average performance of the old dataset is better compared to the new dataset for all parameters. As for the old dataset, SMO shows significantly better performance here compared to the other classifiers and IB1 is the worst.

TABLE VIII.
AVERAGE SUMMARY RESULTS (FOR NEW DATASET)

Classifier	*FP*	*FN*	*ACC*
SMO	*0.009418*	*0.044494*	*97.33234*
IB1	*0.017546*	*0.154653*	*91.47096*
DT	*0.035676*	*0.099289*	*94.22557*
RF	*0.006605*	*0.063789*	*96.48839*

VII. DISCUSSION AND FUTURE WORK

While our malware classification strategy worked very well on the old malware set, the results were much more moderate on the new malware set. This weaker result was almost certainly due to the difference in malware in the samples. Some malware families in the New Dataset require the user's input along with an Internet connection in order to execute some of the in-built functions, and so these functions would not have been extracted into our classification test using our method. In contrast, the boosted RF test gave reasonable results and this indicates that older classification techniques should not be abandoned en masse but that they could be adapted to cope with malware as it evolves. One such adaptation might be to include both old and new malware in the same test; another might be to combine the features for the datasets in other ways as, for example, in [12].

On the other hand, the classification results of our second test, using a cumulative approach to distinguish malware from cleanware, show more consistent performance on the new dataset compared to the old dataset. However, the accuracy on the old dataset is better than on the new dataset. Therefore, it is obvious from our results that it is possible to develop a malware classification technique which can defend against future malware.

Moreover, we have demonstrated in this paper that it is possible to develop an anti-malware technique which can maintain consistent performance with more advanced, future malware. The main approach we have used was to combine all feature types, derived from FLF, PSI and dynamic API calls and API parameters, into a single vector thus allowing the classifier algorithm to identify complex patterns which span multiple feature types. Our empirical study indicates that our strategy performed well on the new malware data set with a 5.4% drop in accuracy (both for the family classification approach and cumulative approach). Therefore it is expected that our proposed method can deal with malware generated in 2012 and beyond. However, it is difficult to predict whether the detection rate will maintain the same performance or not. In our future work we will investigate the capabilities of our system on more challenging datasets.

ACKNOWLEDGEMENT

The authors wish to thank Dr. Steve Versteeg of CA Technologies for his support and advice on the work in this paper.

REFERENCES

[1] Bailey, M., Oberheide, J., Andersen, J., Mao, Z., Jahanian, F. and Nazario, J. (2007) Automated Classification and Analysis of Internet Malware, *Chapter in Recent Advances in Intrusion Detection*, *LNCS 4637*, 178-197, 2007.

[2] Barford, P., Yagneswaran, V. (2007) An inside look at botnets. *Advances in Information Security*, Springer, Heidelberg, 171-191.

[3] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P. and Fienberg, S., (2003) Adaptive name matching in information integration, *Intelligent Systems, IEEE 18(5)*, 16 - 23.

[4] Fossi, M., Mack, T., Mazurek, D., Egan, G., Adams, T., McKinney, D., Haley, K., Blackbird, J., Wood, P., Johnson, E. and Low, M. (2011), 'Symantec Internet security threat report: Trends for 2010', Volume 16, 20 pages.

[5] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. (2009); The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, Volume 11, Issue 1.

[6] Islam, R., Tian, R., Moonsamy, V., Batten, L., Versteeg, S.: A cumulative timeline approach to malware detection. Submitted.

[7] Kohavi, R. (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. *In IJCAI*, 1137–1145

[8] Lee, J.; Im, C. and Jeong, H. (2011), A study of malware detection and classification by comparing extracted strings, in *'Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*', ACM, New York, NY, USA, 75:1-75:4.

[9] Nair, V., Jain, H., Golecha, Y., Gaur, M. and Laxmi, V. (2010), MEDUSA: Metamorphic malware dynamic analysis using signature from API, in *'Proceedings of the 3rd international conference on Security of information and networks*', ACM, 263-269.

[10] Rajab, M., Ballard, L., Jagpal, N., Mavrommatis, P., Nojiri, D., Provos, N. and Schmidt, L. (2011) Trends in Circumventing Web-Malware Detection. *Google Technical Report*.

[11] Rosayid, N., OhruiI, M., Kikuchi, H., Sooraksa, P. and Terada, M. (2010) A Discovery of Sequential Attack Patterns of Malware in Botnets, *Information Processing Society of Japan*, SMC 2010, 2564-2570.

[12] Roundy, K. and Miller, B. (2010), Hybrid Analysis and Control of Malware, in 'Recent Advances in Intrusion Detection', *Springer Berlin - Heidelberg*, 317-338.

[13] Sukwong, O., Kim, H. and Hoe, J. (2010) An Empirical Study of Commercial Antivirus Software Effectiveness, *Computer 44 (3),* 63-70.

[14] Tang, H., Zhu, B. and Ren, K. (2009), A New Approach to Malware Detection, in Jong Park; Hsiao-Hwa Chen; Mohammed Atiquzzaman; Changhoon Lee; Tai-hoon Kim & Sang-Soo Yeo, ed., 'Advances in Information Security and Assurance', *Springer Berlin / Heidelberg*, 229-238.

[15] Tian, R., Batten, L., and Versteeg, S. (2008) Function length as a tool for malware classification. *In Proceedings of the 3rd International Conference on Malicious and Unwanted Software: MALWARE* 2008, 69–76.

[16] Tian, R., Islam, R., Batten, L., and Versteeg, S. (2010) Differentiating malware from cleanware using behavioural analysis. *In Proceedings of the 5th International Conference on Malicious and Unwanted Software: MALWARE 2010*, 23-30.

[17] Witten, I., Frank, E., Trigg, L., Hall, M., Holmes, G. and Cunningham, S. (1999) Weka: Practical machine learning tools and techniques with Java implementations, in *Computer Science Working Papers*, 10289/1040, University of Waikato, 192-196.

[18] Yagi, T., Tanimoto, N., Hariu, T. and Itoh, M. (2010) Investigation and analysis of malware on websites, in *'Web Systems Evolution' (WSE), 2010*, IEEE , 73 -81.

[19] You, I. and Yim, K. (2010) Malware Obfuscation Techniques: A Brief Survey, in 'Broadband, Wireless Computing, Communication and Applications' *(BWCCA),* 297 -300.

[20] Zheng, X. and Fang, Y. (2010), An AIS-based cloud security model, in *'Intelligent Control and Information Processing (ICICIP)*, 153 -158

[21] Cyveillance, accessed on 19th May 2011, http://www.cyveillance.com/web/news/press_rel/2010/2010-08-04.asp

Dr. Rafiqul Islam is working as *Research Academic* with School of Information Technology, Deakin University, Melbourne, Australia. He obtained his Ph.D. from School of Engineering and Information Technology, Deakin University, Melbourne, Australia. He has published more than 50 peer reviewed research papers. He is Fellow member of AATT and Member of IEEE. His research interest includes IT security, Machine learning, Cluster classification etc. He is involved in different International conferences such as IEEE ICIS, IEEE NSS, IEEE ISPAN, ATIS, IEEE SNPD, IEEE ICCIT, IEEE SERA, IEEE SSNE, ATIS etc.

Ronghua Tian is currently a PHD candidate at Deakin University in Australia. She obtained her Master of Engineering in Computer Organization and Architecture at Chongqing University in China. She obtained her Bachelor of Engineering in Computer Software at Changchun University of Science and Technology in China. Her current research topic is Malware Analysis and Classification and her research interests spanned various topics including agent-related technology, P2P-based Distributed Storage Technology, Real Time Database System. She has worked as research assistant at Deakin University in Australia. She has worked as Systems Designer and Applications and Analyst Programmer in China. She also worked as LAN Administrator in China.

Veelasha Moonsamy completed her Bachelor (Hons) of Information Technology with a major in IT Security at Deakin University in 2011. Her Honours thesis focused on the use of feature reduction methods to speed up malware classification. Her research interests include malicious software, machine learning algorithms and security protocols.

Professor Lynn Batten holds the Deakin Research Chair in Mathematics and is Director of Information Security Research at Deakin University. She is a Fellow of the Australian Computer Society, a Graduate of the Australian institute of Company Directors and a Senior Member of the IEEE. Her research interests cover a broad set of areas in information security from cryptography to malicious software and digital forensics.