Behaviour-aware Malware Classification: Dynamic Feature Selection

1st Phai Vu Dinh
Le Quy Don Techincal University
Department of Information Security
Hanoi, VietNam
dinhphai88@gmail.com

2nd Nathan Shone
Liverpool John Moores University
Department of Computer Science
Liverpool, UK
n.shone@ljmu.ac.uk

3rd Phan Huy Dung
Le Quy Don Techincal University
Department of Information Security
Hanoi, VietNam
phanhuydungcao@gmail.com

4th Qi Shi
Liverpool John Moores University
Department of Computer Science
Liverpool, UK
q.shi@ljmu.ac.uk

5th Nguyen Viet Hung
Le Quy Don Techincal University
Department of Information Security
Hanoi, VietNam
hungnv@lqdtu.edu.vn

6th Tran Nguyen Ngoc
Le Quy Don Techincal University
Department of Information Security
Hanoi, VietNam
ngoctn@lqdtu.edu.vn

***Abstract*—Despite the continued advancements in security research, malware persists as being a major threat in this digital age. Malware detection is a primary defence strategy for most networks but the identification of malware strains is becoming increasingly difficult. Reliable identification is based upon characteristic features being detectable within an object. However, the limitations and expense of current malware feature extraction methods is significantly hindering this process. In this paper, we present a new method for identifying malware based on behavioural feature extraction. Our proposed method has been evaluated using seven classification methods whilst analysing 2,068 malware samples from eight different families. The results achieved thus far have demonstrated promising improvements over existing approaches.**

***Index Terms*—Malware Behaviours, Malware Classification, Feature Extraction, Machine Learning**

I. Introduction

Despite advancements in cyber security technology, the use of malware as an attack mechanism shows no sign of relenting. This is demonstrated by the 10.5 billion malware-based attacks that occurred globally in 2018. This figure is up by 22% from 2017 and 34% from 2016 [1].

Nearly all modern networks utilise some form of anti-virus protection as a primary defence mechanism. However, the challenge of reliably identifying malware is becoming increasingly more difficult. This is raising many concerns, as in 2019 there will be an estimated 908.25 million unique malware samples generated (an increase of 1830.4% in just under a decade) [2]. No equilibrium has ever been reached - highlighting the fact that the state-of-art defence cannot match the pace of malware development.

Most anti-virus software heavily relies upon signature matching but threats must be known and analysed before such signatures can be created. This approach is unsustainable given the estimated 2.5 million malware samples expected to be generated each day. Additionally, these mechanisms are unable to handle 0-day threats or advanced evasion techniques, such as those encountered in oligomorphic, polymorphic or metamorphic malware. When including such samples, detection accuracy rates range from between 25% to 50% [3]

Taking this into consideration, it is unsurprising that static malware analysis (analysis malware source/assembly code) is highly ineffective in comparison to dynamic malware analysis (run-time behavioural observations). Automated malware behaviour analysis sandbox environments such as Cuckoo [4] can drastically improve the comprehensiveness of the analysis results returned.

Recently, many behaviour-based malware detection solutions have been proposed using shallow learning techniques such as k-Nearest Neighbour (kNN), Support Vector Machine (SVM) and decision trees. As opposed to deep learning, shallow learning approaches offer a much faster and more resource-efficient decision process when it comes to malware detection. One downside to such approaches is the high false positive rates, which can usually be attributed to either malware complexity or sub-standard feature selection.

Poor feature selection for behavioural malware tends to be as a result of narrowly-focused feature sets, which fails to give an accurate holistic view of the behaviour. Alternatively, feature selection can be too inflexible, which fails to account for the dynamic nature of malware behaviour.

In this paper, we propose a novel method for improving behaviour-based malware detection, through enhancing the feature selection process. The method is split into two key stages. *Firstly*, suspicious files undergo a holistic behavioural analysis in a Cuckoo sandbox environment. The subsequent findings are processed and the optimum features are extracted and used to create training and test datasets. *Secondly*, the extracted feature datasets are then used to train and test a

The authors would like to thank World Bank and the Ministry of Science and Technology for their support provided through the FIRST Foreign Talents STI Grant.

978-1-7281-3003-3/15/$31.00

classification model to correctly identify the malware.

The main contribution of our work is the novel approach to extracting the best indicative features from malware behaviour. We have then evaluated an implementation of our proposed method using real-world malware samples. Additionally, we have compared the results against those achieved using the unigram extraction method.

The remainder of this paper is structured as follows. Section II presents a summary of recent related work. Section III details our proposed malware classification methodology. Section IV details the experiments performed and the results obtained and finally the paper is concluded in Section V.

II. Related Work

In recent research, there is a growing trend concerning the application of machine learning within the field of malware detection. The ability of machine learning models enables them to analyse and learn from data. Therefore, making them ideally-placed to evolve alongside malware itself.

Many recent works have focused upon three main shallow learning approaches:

- *Supervised Learning* - Predominantly ground truth-based algorithms have been proposed to classify malware such as Nave Bayes [5], Support Vector Machine (SVM) [6], Random Forest [7], Decision Tree and Gradient Boosting Decision Tree [8]. Although high accuracy rates are achievable, the reliance on pre-labelled training data is infeasible when aiming to detect 0-day or evolving malware.
- *Unsupervised* - Many clustering-based approaches have been proposed to detect malware such as: k-means, Bayesian [9], DBSCAN and x-means.
- *Semi-supervised* - learning methods, a litter experiment is tested. Recently, the paper [10] applied genetic for malware detection and the paper [11] brought the fuzzy Logic Paradigm in these fields that may give some interesting approaches.

The literature has also identified a rise in the application of deep learning for malware detection. As it enables more complex, hidden or unstructured behavioural patterns to be identified. Some proposed techniques for malware detection involve the use of: stacked auto-encoders, Deep Belief Networks (DBN), Long Short-Term Memory (LSTM), Convolution Neural Networks (CNN) and Multi-Task Neural Networks.

One major disadvantage of using deep learning in malware detection is the slow and resource-intensive nature of the process. To overcome these limitations, some researchers are proposing a hybrid approach that combines shallow and deep learning. For example, Yuan's paper [3] outlines the DeepMalware, which uses shallow learning to identify clear cases of malicious or benign software. Whereas, borderline or uncertain software is analysed using a CNN and LSTM-based deep learning model.

An important pre-requisite for a shallow learning-based methodology is feature selection. This process is commonly overshadowed by a focus on classification algorithm selection or model optimisation. Ineffective feature selection can significantly affect the performance of any shallow learning model regardless. It is apparent from the literature that many behaviour-based malware analysis approaches use a non-adaptive or overly-narrow set of features. The consequences of this are that due to the complex and evolving nature of malware behaviour, it can negatively impact on detection accuracy. A more holistic approach is required to better represent malware behaviour.

Frequently, dynamic behavioural features are sequential e.g. op codes or system calls. To maximise feature extraction these are commonly expressed as *n*-grams [12].

Khammas et al. [13] also present a comparison of feature selection approaches for improving malware detection. These methods included chi-squared, CFsSubset, Principle Component Analysis (PCA), information gain and gain ratio. The authors concluded that information gain offered the best results overall.

Similarly, Singh and Hofman [6] also proposed the use of chi-squared to select Android kernel system calls. It was found that using this approach only 37 of 337 calls were suitable malware predictors.

It is apparent from the literature that there is no consensus over feature extraction methods for malware behaviour. Therefore, further work needed in this area to improve the effectiveness of future malware detection.

III. Methodology

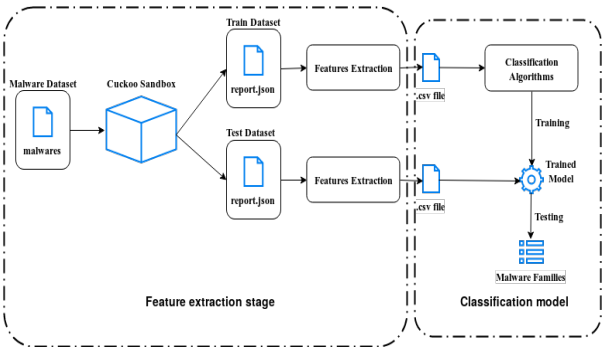

Fig. 1. Overview of Our Proposed Approach

In this section, this paper will present the general methodology for our malware classification based on behaviour analysis approach. As can be seen from Figure 1, this methodology is separated into two stages, namely, feature extraction and model construction. To specify, the former commences with the malicious file being taken into Cuckoo sandbox machine. This is followed by JSON reports which are generated from the Cuckoo software before it is separated into two sub-datasets, training dataset and testing dataset, respectively. At the next step, malware attributes will be extracted from the JSON reports, making it one of the most important contributions in this work.

When it comes to the second stage of the methodology, we will use various classification algorithms for training and

testing e.g. SVM, Random Forest, kNN, and Artificial Neural Network. The final step involved in the evaluation is to compare with other approaches.

In this work, we used the Cuckoo sandbox installed on Windows 7 x64, running in VirtualBox. To enable the malware to show all behaviours, the Windows OS is not updated and all anti-virus software, firewall and UAC are disabled, and Adobe Acrobat Reader is installed. In order to obtain data relating to malware behaviours, the Cuckoo sandbox will record monitoring observations in blocks of 30 minutes.

A. Feature Extraction Method

1) Unigram (1-gram): In principle, this is a fairly robust mechanism that has demonstrated promising results in recent research. The JSON report produced by Cuckoo can be treated as a simple text file. This is followed by using unigrams to extract information from the file. Nevertheless, the high time and memory requirements are the most challenging aspect of this method. For example, with around 30GB worth of JSON files, we will get a dictionary with above 6 million words. Thanks to a method from [13], we may overcome the problem of memory usage. Specifically, (1) in terms of each JSON file, extract its unigrams, (2) concatenate all unigrams to obtain a dictionary, (3) for each word in dictionary, count the number of its appearance in each file, (4) select top 20,000 with highest value, finally, turn each unigram into a vector with 20,000 dimensions [14].

2) Malware Behaviour Analysis: Although unigram is a common approach for extracting information from JSON files, to some extent it does not illustrate the principle of malware behaviours. Each JSON file will likely contain meaningless information such as brackets, captions etc., which feature extraction algorithms may find a challenge. However, the more useful malware behaviour characteristics extracted, the greater the potential classification accuracy will be. In this section, we will outline four malware characteristic categories that we will be extracting from the Cuckoo JSON report: Network, Signatures, Behaviours and Others.

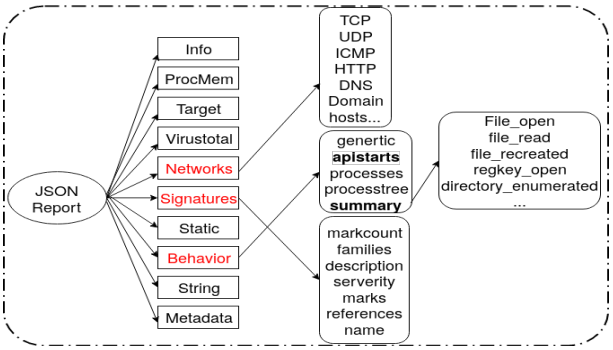

Fig. 2. JSON Structure

Behavioural: Two of the most important reporting features for malware classification are Apistats and summary. Apistats provides information about the API calls utilised by the malware during its execution, whilst summary shows the malware functions that have been deployed.

Signature: The report files generated by Cuckoo will detail identified signs of malicious behaviour. During the analysis process, malware will be matched against current malware signatures. This enables a quicker and more effective identification of known malware. There are several important observations features for each signature, namely severity and 'pe_features'. While the former one shows the degree of danger of the malware, the latter identifies whether the executable's file header structure is unusual, which is a common trait of packed malware.

Network: For malware communicating over the Internet, we will monitor and analyse the network protocols used. We will focus on common communication protocols such as HTTP, IRC, TCP, UDP, ICMP, and SMTP etc. The particular feature will focus on the number of times the malware used each protocol.

Others: Other important information will also be extracted including the runtime duration, assigned category and assigned score for the malware file. Memory usage will be extracted from the procmemory section and file size will be extracted from the Target section. The number of anti-virus software that reported this file will be picked up in Virustotal section and the number of strings is extracted from the String section.

3) Behavioural Malware Extraction (BME): The proposed BME algorithm is shown in Figure 3.

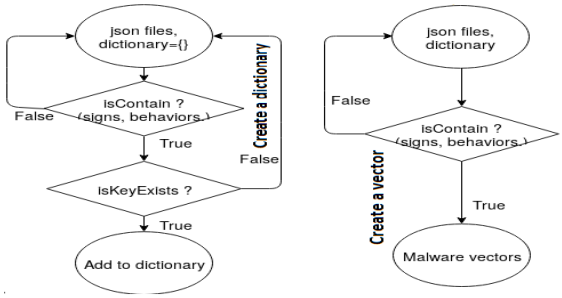

Fig. 3. Behavioural Malware Extraction

The BME commences by generating a Python dictionary to hold the extracted features. The required features will then be extracted from the JSON file by finding a new key-value pair. The key-value pairs will include observations belonging to the four main feature categories (behaviours, signatures, networks and other). If a new JSON key-value pair is identified, it will be appended to the dictionary. The value stored for the dictionary entry will usually be the same value that is stored located in the JSON file. However, for some observations it will be necessary to instead store the number of occurrences of the particular JSON key. After this process is completed, a malware vector will be generated for each JSON file parsed. As the Python dictionary is used, the malware vector can be initialised with the same size as the number of key-value pairs stored in the dictionary.

4) Classification Algorithms: In our evaluation, we will be utilising seven different classifiers, to provide an in-depth comparison of our BME method against the unigram method. We will be using k-Nearest Neighbour (kNN), Nave Bayes

(NB), Support Vector Machines (SVM), Artificial Neural Networks (ANN), Random Forest (RF), Decision Tree (DT) and Logistic Regression (LR).

B. Assessment and Evaluation Criteria

In this section, we show outline the assessment measures and simulation methods used to compare the BME method with unigram approach. *True Positive (TP)* - correctly identified as malware; *False Positive (FP)* - incorrectly identified as malware; *True Negative (TN)* - correctly identified as benign; *False Negative (FN)* - incorrectly identified as benign.

Additionally, the following measures will be used in the evaluation of the classification algorithm:

- **Accuracy = (TP+TN)/(TP+TN+FP+FN)** which indicates the rate of the total number of correct classifications
- **Precision=TP/(TP+FP)** which determines the number of correct classifications penalised by the number of incorrect classifications.
- **Recall=TP/(TP+FN)** which measures the number of correct classifications penalised by the number of missed entries.
- **F-score=2*(Precision*Recall)/(Precision+Recall)** that is used as an effective measurement using the harmonic mean of precision and recall.

In order to provide a comprehensive evaluation, we will also be comparing performance on datasets that have been pre-processed using feature reduction techniques. Firstly, we will use PCA as it is widely utilised to emphasise features and extract strong patterns within a dataset. The second method is a Stacked Auto-Encoder (SAE). An auto-encoder is an unsupervised extraction algorithm, which learns the best parameters required to reconstruct its output as close to its input as possible. In this paper, the data points are extracted from the hidden layer before these are brought into the classifiers [15].

IV. Experiments and Results

A. Dataset

In our experiments, we will be using real-world malware samples obtained from virusign.com. These evaluations will be undertaken using samples from eight malware families: Dridex, Kelihos, Locky, Ramnit, Sality, Simda, Vawtrak and Zeus. The breakdown of the number of samples in each malware family is shown in Table I.

TABLE I
Malware Dataset

Dridex	Kelihos	Locky	Ramnit	Sality	Simda	Vawtrak	Zeus	Total
173	271	135	421	180	469	60	359	2068

Using a script, each malware sample file is submitted to the Cuckoo sandbox. The algorithm presented in Figure 3 will then return a set of features. In this example, the malware is represented by a vector with 846 elements in total, with differing numbers of features representing the aforementioned categories: 316 Behaviours features, 496 Signature features, 18 Network features and 16 Other features.

TABLE II
BME vs 1-gram with rate (70-30)

	AC		Precision		Recall		F-score	
	BME	1gram	BME	1gram	BME	1gram	BME	1gram
DT	**97.42**	93.11	**96.23**	91.21	**96.91**	89.77	**96.56**	90.32
KNN	**94.53**	87.95	**93.53**	87.88	**93.95**	80.72	**93.68**	81.23
LR	**95.81**	93.66	**94.00**	92.90	**95.02**	91.92	**94.43**	92.38
ANN	**95.81**	87.05	**94.45**	88.67	**95.23**	83.55	**94.77**	85.21
NB	**92.11**	68.66	**91.38**	72.46	**90.32**	61.48	**90.53**	58.43
RF	**95.97**	94.70	**96.10**	95.09	**94.83**	92.88	**95.38**	93.86
SVM	**90.34**	36.36	**92.69**	13.84	**87.13**	24.51	**88.94**	16.53

TABLE III
BME vs 1-gram with rate (60-40)

	AC		Precision		Recall		F-score	
	BME	1gram	BME	1gram	BME	1gram	BME	1gram
DT	**95.17**	94.34	**93.88**	91.18	**94.44**	90.89	**94.10**	90.95
KNN	**93.48**	91.08	**92.31**	88.84	**92.98**	90.73	**92.54**	89.51
LR	**95.65**	95.30	**94.39**	93.91	**95.01**	94.70	**94.60**	94.21
ANN	**95.65**	91.33	**94.43**	90.74	**94.99**	90.48	**94.58**	90.53
NB	**91.43**	68.92	**90.95**	76.03	**89.27**	65.57	**89.77**	60.81
RF	96.38	**96.39**	**95.89**	95.70	**95.52**	95.41	**95.68**	95.54
SVM	**85.63**	37.23	**79.02**	14.08	**74.44**	24.30	**75.63**	16.60

B. Results

Whilst comparing our BME method against the unigram method, we have used seven classifiers and four different assessment measurements (as outlined in Section III-C. The results obtained are presented in Tables II and III.

TABLE IV
BME vs 1-gram utilized PCA

	AC		Precision		Recall		F-score	
	BME	1gram	BME	1gram	BME	1gram	BME	1gram
DT	**91.95**	89.57	**89.10**	85.47	**90.83**	86.99	**89.89**	86.10
KNN	**94.69**	91.17	**94.05**	88.84	**94.34**	88.19	**94.10**	88.41
LR	**93.88**	85.07	**93.06**	87.80	**92.90**	76.39	**92.82**	77.47
ANN	**93.88**	80.26	**93.23**	63.00	**92.90**	61.42	**92.92**	60.62
NB	**72.14**	45.43	**52.04**	26.64	**49.06**	28.70	**45.56**	23.62
RF	**95.33**	94.38	**94.05**	92.62	**94.68**	92.17	**94.35**	92.38
SVM	**85.67**	42.05	**80.27**	26.99	**72.82**	26.59	**74.94**	20.79

TABLE V
BME vs 1-gram utilized SAE

	AC		Precision		Recall		F-score	
	BME	1gram	BME	1gram	BME	1gram	BME	1gram
DT	**91.47**	83.31	**89.03**	78.28	**89.68**	76.09	**89.30**	76.78
KNN	**95.17**	88.28	**94.64**	86.35	**94.56**	83.54	**94.58**	84.41
LR	**95.49**	92.46	**94.20**	91.23	**94.97**	91.19	**94.53**	90.96
ANN	**95.33**	91.65	**93.87**	89.76	**94.83**	91.09	**94.25**	90.14
NB	**92.11**	75.60	**89.55**	67.70	**91.12**	72.35	**89.84**	66.43
RF	**95.01**	92.13	**94.61**	91.32	**94.16**	89.65	**94.34**	90.39
SVM	**94.04**	87.96	**93.74**	88.30	**93.02**	81.98	**93.29**	84.30

Table II shows the experiment with a training:test ratio of 70%:30%, whilst Table III shows the results of a 60%:40% ratio. As can be seen from both tables, BME has produced superior results for all measures, in comparison to the unigram method. The one exception to this is the AC measurement with RF in Table III, where the unigram method is slightly higher (96.38 vs 96.39). Examining the best score (using AC and F-score) shows that BME performed best with the DT classifier (97.42 and 96.56) in Table II, whilst unigram performed best

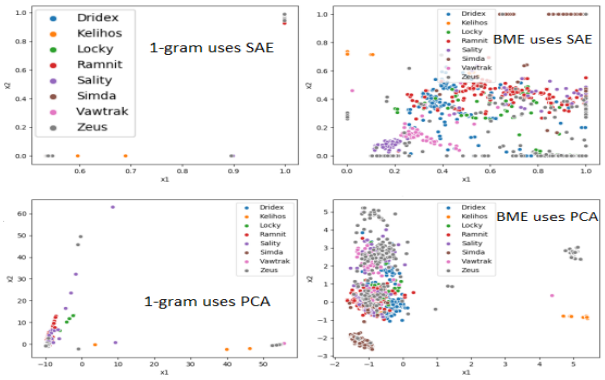

Fig. 4. BME vs 1-gram

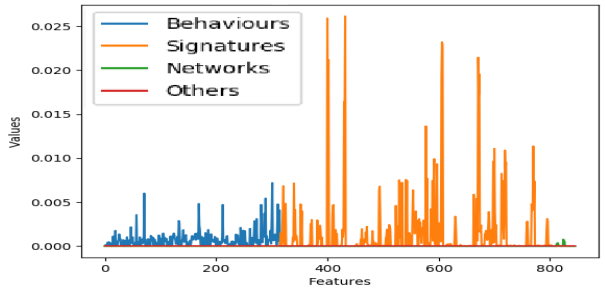

Fig. 5. Features Ranking

with the RF classifier (96.39 and 95.68) in Table III.

Table IV and V compare BME against the unigram method when analysing datasets pre-processed using different feature reduction techniques. Table IV presents a PCA-based comparison, whilst V presents the SAE-based comparison. Each malware vector is reduced in size to 50 dimensions for both of these techniques.

The results show that BME outperforms unigram in all of the accuracy measurements. It is evident that most of the measurement values have decreased overall. More specifically, the AC and F-score of BME for all classifiers show an average decline of 0.48% and 1.24% in the two tables. A more significant average decline can be seen for unigram with 7.13% and 4.83% in the two tables. To some extent, this can be explained by the fact that reduction techniques can bring malware families dimensionally closer together. Therefore, classifiers will face greater difficulties in separating this data.

Figure 4 presents a comparison plot featuring both reduced datasets created using BME and unigram. As can be seen, the unigram data points stacked and mixed together. By contrast, those created using BME are more clearly distinguishable and have much clearer separation and distribution. There is therefore strong evidence to support the claims that BME is better than unigram for extracting malware features.

Figure 5 shows the use of forests of trees to evaluate the importance of features in the dataset extracted by BME. In summary, the Signature category of features contribute the most in terms of malware classification, followed by the Behaviours category. For instance, 26.41% of Signature features are greater than the average value of ranking features, while Behaviours accounts for 16.77%. All features from the Network and Others category are under the average value.

V. Conclusion & Future Work

In this paper, we have discussed the problems faced by existing malware classification. Our novel BME method for malware extraction based on malware behaviours is proposed. We have presented a thorough analysis of our proposed method using 2,068 malware samples from eight different families. The results of this have clearly demonstrated that our proposed method enhances malware classification significantly in comparison to the main existing unigram approach. Similarly, we also demonstrated importance of the extracted features using the a featuring ranking analysis.Due to the page limit, only a brief analysis is provided in this paper.

In our future work, we would like to further build upon the method proposed in this paper, to implement a fully automated approach for malware analysis.

References

[1] "Number of malware attacks per year 2018 — statista," https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/, accessed: 2019-06-13.

[2] "Malware statistics and trends report — av-test," https://www.av-test.org/en/statistics/malware/, accessed: 2019-06-13.

[3] X. Yuan, "Phd forum: Deep learning-based real-time malware detection with multi-stage analysis," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, May 2017, pp. 1–2.

[4] "Github - cuckoosandbox/cuckoo," https://github.com/cuckoosandbox/cuckoo, accessed: 2019-06-13.

[5] A. Razaque, Z. Xihao, W. Liangjie, M. Almiani, Y. Jararweh, and M. J. Khan, "Nave bayesian and fuzzy c-means algorithm for mobile malware detection precision," in *IoTSMS 2018*, Oct 2018, pp. 239–243.

[6] L. Singh and M. Hofmann, "Dynamic behavior analysis of android applications for malware detection," in *2017 International Conference on ICCT*, Dec 2017, pp. 1–7.

[7] R. Vyas, X. Luo, N. McFarland, and C. Justice, "Investigation of malicious portable executable file detection on the network using supervised learning techniques," in *2017 IFIP/IEEE Symposium on IM*, May 2017, pp. 941–946.

[8] E. M. Rudd, R. Harang, and J. Saxe, "Meade: Towards a malicious email attachment detection engine," in *2018 IEEE International Symposium on HST*, Oct 2018, pp. 1–7.

[9] M. E. Ahmed, S. Nepal, and H. Kim, "Medusa: Malware detection using statistical analysis of system's behavior," in *2018 IEEE 4th International Conference on CIC*, Oct 2018, pp. 272–278.

[10] T. A. Le, T. H. Chu, Q. U. Nguyen, and X. H. Nguyen, "Malware detection using genetic programming," in *the 2014 Seventh IEEE Symposium on CISDA*, Dec 2014, pp. 1–6.

[11] E. Gandotra, D. Bansal, and S. Sofat, "Malware threat assessment using fuzzy logic paradigm," *Cybernetics and Systems*, vol. 48, no. 1, pp. 29–48, 2017.

[12] C. D. Morales-Molina, D. Santamaria-Guerrero, G. Sanchez-Perez, H. Perez-Meana, and A. Hernandez-Suarez, "Methodology for malware classification using a random forest classifier," in *2018 IEEE International Autumn Meeting on ROPEC*, Nov 2018, pp. 1–6.

[13] B. M. Khammas, S. Hasan, R. A. Ahmed, J. S. Bassi, and I. Ismail, "Accuracy improved malware detection method using snort sub-signatures and machine learning techniques," in *2018 10th Computer Science and Electronic Engineering (CEEC)*, Sep. 2018, pp. 107–112.

[14] E. David and N. Netanyahu, "Deepsign: Deep learning for automatic malware signature generation and classification," 07 2015, pp. 1–8.

[15] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE TETCI*, vol. 2, no. 1, pp. 41–50, Feb 2018.