Journal of Engineering and Applied Sciences 13 (15): 6281-6292, 2018
ISSN: 1816-949X

Malware Analysis and Detection Approaches: Drive to Deep Learning

[1]Toqeer Ali, [2,3]Salman Jan, [2]Shahrul Niza Musa and [1]Atiqur Rahman
[1]Islamic University of Madinah, Medina, Saudi Arabia
[2,3]Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Malaysia
[3]University of Peshawar, Peshawar, Khyber Pakhtunkhwa, Pakistan

Abstract: The growing number of malware attacks poses serious threats to private data and to the expensive computing resources. To detect malware and their associated families, anti-malware companies rely on signatures which indeed include regular expressions and strings. The recent malware attacks in the last few years including the resurgence of ransomware have proven that signature-based methods are error-prone and can be easily evaded by intelligent malware programs. This study reviews traditional and state-of-the-art models developed for malware analysis and detection. According to our observation the classification of malware and their behavior facilitates in provision of basic insights for the researchers working in the domain of malware analysis. At the end we present the conception of using Deep Convolutional Generated Adversarial Networks (DCGAN) in the area of malware detection as the DCGANs are the latest approach in deep learning that effectively deals adversarial examples.

Key words: Trusted computing, neural networks, RNN, ESN, CNN, GAN, DCGAN, GPU

INTRODUCTION

Computing infrastructure has transformed into web of interconnected and mutually, dependent hardware and software (Alam *et al*., 2008, 2012). A wide range of services are provided on the internet and as a result individuals, organizations, businesses and the governmental agencies handle their computations online (Cusumano, 2010). The service providers in different areas of interest, provide services online. However, they require strong security mechanisms against the newly build growing attacks on the software (Willems *et al*., 2007; Hasselbring and Reussner, 2006; Moein *et al*., 2014). It has also been observed the growth of metamorphic and polymorphic malware which frequently change appearances to avoid being detected (You and Yim, 2010; Cesare *et al*., 2013; Bodke, 2013, Santamarta, 2006). There is a continuous exponential growth observed in malware in the last two decades as shown in Fig. 1.

As per report of symantec, more than 430 million new malware are reported in the year 2015, i.e., an increase of 36% as compared to the prior year. Furthermore, as available on statistics page of virus total, there are over millions of newly retrieved samples that had to be analyzed (Anonymous, 2017).

The conception of ransomware which is presented back in 1980's have aroused and affected computations all over the world (Cohen *et al*., 2017; Liao *et al*., 2008). The malware typically locks the desktop of target system and makes it inaccessible by overwriting, encryption or by deletion of system's files. The most astounding number of zero-day vulnerabilities was uncovered in 2015. Figure 1 and 2 represents year wise increase in percentage of malware attacks.

The prevention of sensitive data, information and other important contents from malicious softwares running on client computers has now remained top priority of service providers. To ensure protection of data against miscreants, there are many software-based solutions available, e.g., anti-viruses, Intrusion Detection Systems (IDS), etc. (Ismail *et al*., 2014). However, current literature reveals that the existing signature-based anti-viruses and IDS are itself vulnerable to attack. In order to device an effective defensive measure, the antimalware developers rely on a system that automatically analyze a novel variant of an existing malware. Signature-based strategies or heuristic-based detection cannot stay up to date with the security difficulties of the increasing growing malware which are polymorphic in nature. In addition, writing malicious programs has become easier than ever as a result of advances in the development of malware. Malicious programing make use of metamorphic or polymorphic algorithms which produces a completely different variant of a malware sample. These adversarial malware sample causes being avoided in detection through

Corresponding Author: Toqeer Ali, Islamic University of Madinah, Medina, Saudi Arabia

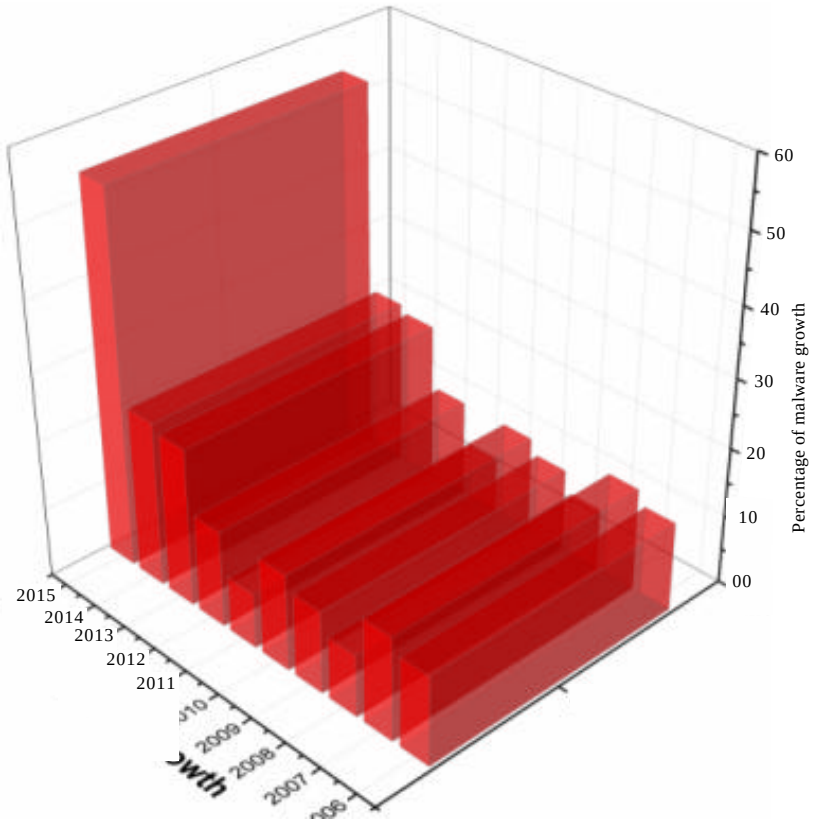

Fig. 1: Malware growth over the years 2006-2015, annual total

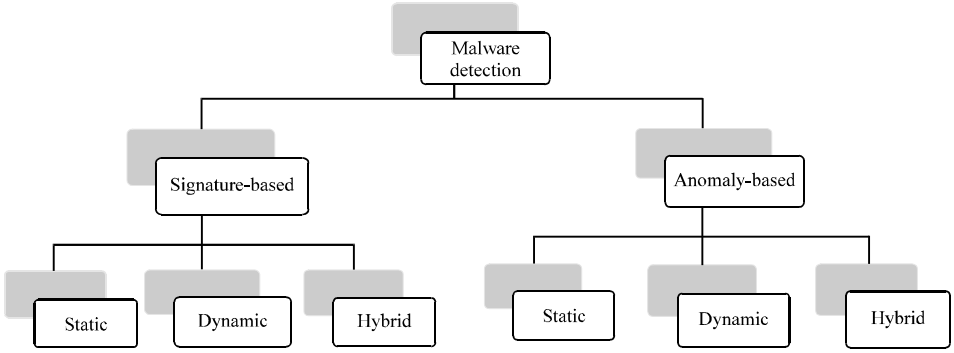

Fig. 2: Malware detection classification techniques (Idika and Mathur, 2007)

signature-based approaches or during correctly classifying them. There is unanimous agreement among experts in the fields of security that Commercial of the Shelf AntiVirus (COTS AV) are not capable to deal with zero day malware (Kolosnjaji *et al*., 2016; Kolter and Maloof, 2006; Mehdi *et al*., 2010).

The intrusion detection systems have been used to ensure platforms and application's trustworthiness (Rowett and Sikdar, 2005; Scheidell, 2009; Day, 2007; Liao *et al*., 2013) but it is reported that IDS require extensive training as these generate unacceptable false alarms. In the following subsection, we elaborate various malware analysis techniques including static, dynamic and hybrid before we review machine learning approaches (Fig. 3).

Literature review: A number of Machine Learning (ML) algorithms and approaches are suggested in the past which includes decision tree, association rule, Support Vector Machine (SVM), Naive Bayes, clustering and

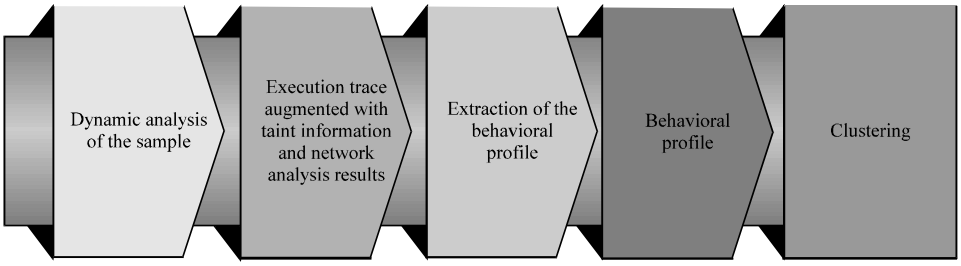

Fig. 3: Overview of the framework (Bayer *et al*., 2009)

random forest, neural networks, Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM) for analyzing malware based on specific criteria and features. The following literature review presents a number of techniques and models used for the purpose of analyzing malware used along with clustering, classification and anomaly based systems. For detection of malware, the very first concept of using data mining approach was provided by Schultz *et al*. (2001). They presented a data mining framework that trains classifiers on benign and malicious dataset to learn underlying patterns in both benign and malicious binaries. The researcher claim that the existing anti-virus approaches are based on heuristics which are generated by hand and unless a signature is created for a malicious code, the anti-viruses are not able to detect the same malicious code. These heuristics based approach is not reliable at times when applied on unseen malicious executables. For the purpose of malware classification, the researcher used static features, i.e., Portable Executable (PE), byte sequences and strings data. Inside the portable executables files, there are DLLs information from which various features are extracted. The information includes list of DLL function calls, DLLs list utilized by binary and various system calls in a DLL. "n" bytes are extracted as per byte sequence approach from executables and similarly text strings from the executables are extracted. A data set of 4266 files was used having 1001 benign and 3265 malicious programs. They further applied rule induction algorithm to find patterns in the DLL data. A learning algorithm Naive Bayes (NB) was used to find patterns in the string data and byte sequences of n-grams was utilized as input data to the algorithm. As per their report, the algorithm produced 97.11% classification accuracy.

Chen *et al*. described the primitive for verification of software known as oblivious hashing. The technique verifies static shape of code and allows the hash computation of the actual execution of a program. The approach is applicable in providing tamper resistance feature in a local protected software code as well as remote code authentication. The program code which is essentially abstract machine instructions of a set $I = \{i_1, i_2, ..., i_N\}$, basically performs various read/write operations on memory locations $M = \{m_1, m_2, ..., m_K\}$. The principal idea is to extract function or program's execution trace and to calculate a hash value for that execution flow. The approach utilizes the hash value to determine the integrity of executables which can be cracked by miscreants to falsify the classification.

For detection and classification of malware executables based on n-grams (Kolter and Maloof, 2006; Kolter and Maloof, 2004) used data mining technique. They obtained 255 million unique n-grams after gathering and encoding 1651 intruded executables and 1971 number of benign executables using n-grams of byte codes as features and training examples. The most relevant n-grams were opted for prediction 4 and the evaluation of framework was carried out using naive bayes, decision trees and support vector machines. Among them, boosted decision generated better results during correctly classification of malware than all and provided AUROC curve of 0.99.

Ye *et al*. (2007) proposed IMDS (Intelligent Malware Detection System). The approach utilizes windows API execution sequences generated by portable executables. The IMDS modules include portable executable parser, object oriented association rule generator and a classifier based on rules for the classification of executables. They compared the efficiency of their malware detection technique with VirusScan, MecAfee and mining techniques including Support Vector Machine (SVM), Decision Tree (J4.8) and Naive Bayes wherein their approach, IMDS outperformed the rest in terms of TP, TN detection rate 97.19% and accuracy 93.07%.

Bayer *et al*. (2009) gave a scalable approach for clustering and identification of samples that exhibit similar behavior. They performed dynamic analysis, using ANUBIS (Anonymous, 2010) to obtain the execution traces of malware programs in a controlled environment which are further generalized into behavioral profiles. Data tainting approach is utilized to track dependences between system calls and to subsequently obtain profiles.

The profiles basically characterize the overall activity performed by programs in an abstract manner, i.e., The system calls their existing dependences over each other, and network activities are generalized in such a manner that consist various operating system objects and the different operations performed over them. These are provided as feed to clustering algorithm, Locality Sensitive Hashing (LSH) which learn various underline pattern within profiles. An overview of their approach is represented through Fig. 3 overiew. According to their results, these algorithm can actually cluster similarly behavior to report behavior as either benign or malicious based on distances with samples. However, their aproach achieves low accuracy rate.

The researcher claim that their framework per form better in terms of precision than the existing approaches which are slow as well as not scalable. Moreover, the presented approach can cluster 75000 samples in span of 3 h approximately. Their technique is trace-dependent, i.e., a malware behavior is triggered once specific condition is met. They have evaluated their framework through state-of-the-art clustering methods (Shafiq *et al*., 2009) proposes method to detect previously unknown malware that automatically extracts distinguishing features from Portable Executables (PE) of windows operating system. The PE is a file format which is standardized by the microsoft windows operating systems for executables, Dynamically Linked Libraries (DLL) and object files. The researcher used a threefold research methodology in which the structural features are selected, reduced by preprocessing and finally data-mining techniques were applied for classification. Their framework comprises of three modules feature extraction module feature selection/preprocessing module and detection module. The researcher also made a comparison with other detection schemes and found their method better than others. They called the framework as PE-miner which is evaluated to have an accuracy of 99% with less than 0.5 false alarm. The framework is built on windows but it can be scaled around different operating systems. They employed different classifier algorithms (Instance Based Learner (IBk), J48, Inductive Rule Learner (RIPPER) and SVM) and carried out evaluation in which J48 outperformed the rest in terms of detection accuracy.

Tian *et al*. (2009) presented malware classification technique based on printable string information inside executable. In their study, they extracted strings from 1367 samples of clean and intruded 5 files and used various classification algorithms including nearest neighbour algorithm, tree based classifiers, statistical algorithms and AdaBoost. Their experiment provided 97% classification accuracy. Since, they have applied

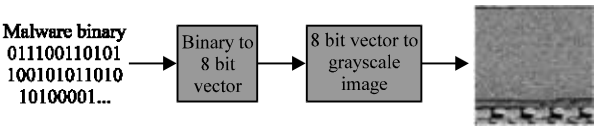

Fig. 4: Malware representation through an image (Nataraj *et al*., 2011)

static technique, these information extracted from these malware in form of printable strings can be manipulated to generate wrong results. In an subsequent study by the same researcher (Tian *et al*., 2010) investigated behavioral features through Application Programming Interface (APIs) calls for detection of malicious files. These features are extracted from executables using trace tool namely HookMe. The framework executes files in virtual environment and collect their logs. They performed and carried out experimental results on malware of size 1368 and cleanware of size 456 files wherein RFk presents the best performance of 97% in terms of accuracy in identifying malware from cleanware. While IB1 is slightly lower in performance. The virtual environment not necessarily obtains all of the API calls as certain newly created malware unpacks themselves and are executed on specific event basis.

Nataraj *et al*. (2011) proposed to represent malware executables as binary strings which may be sequence of zeros and ones to form vectors. The vectors are reshaped to build matrix, so that to represent executables as an images as represented in Fig. 4. They have reported and visualized that various malware families are distint and have different visual characteristics. Moreover, they used texture analysis technique of computer vision to classify families of malware. A K-nearest neighbor technique with Euclidean distance method is used for malware classification. Their results indicate 98% accuracy for classification of malware over an opted database containing 9458 samples having 25 various malware families. Miscreants can adopt countermeasures to manipulate binaries or sections can be added with redundant data to misrepresent executables. In order to analyze dynamic behavior of platform (Ali *et al*., 2010) presents the idea of using the techniques developed for intrusion detection. They used stochastic machine learning techniques to model systems calls sequences generated by a target application on remote machine. Measuring systems calls on a platform and securely reporting the same to challenger for verification purpose is one of the limitations of remote attestation. The researcher have represented systems calls in hyperspace which actually keeps record of sequence of system calls generated by executable. After collecting the hypergrams,

they are measured and its hashes are extended to PCR inside TPM and reported to challenger. Their experimental results depict that the technique is feasible in determining behavior. Their learning models using Naive Bayes, OR, J48, Bayes net was 59.14, 62.19, 78.65, 85.72, respectively. It is however, noticeable that intruders can device mechanism to manipulate the verification procedure by reporting a known good hypergram to the challenger which can lead to higher rate of false positive and compel to restrict the use of the system.

Santos *et al.* (2013) developed a hybrid model to detect malware called OPEM which operates appearance of opcodes. Researcher presented method to extract opcodes that are most relevant and then evaluated frequency of opcode sequences. A labeled data set of 17000 malicious programs with 585 malware families are downloaded from VxHeavens and are feeded to Eset antivirus to confirm the labeling. While 1000 legitimate executables are collected from computers. The optcode equences are extracted for each file in the datasets. Researcher provided empirical validation that the method detects newly created malware too. Used instance selection for resampling new 6 ones and feature selection to decrease number of features which ultimately improves machine learning accuracy. They used several classification algorithms to validate the approach including random forest, J48, K-Nearest Neighbour (KNN), Bayesian Networks (BN), Naive Bayes, Support Vector Machines (SVM). K2 is found to be the fastest Bayesian classifier, requiring 8.93 and 0.10 msec for training and testing, respectively. TAN founded the slowest training time at 488.69 msec requiring 0.15 msec for testing. random forest was better and fast than J48 with 2.68 msec of training time. The SVM yielded 92.92% accuracy for an opcode sequence of length one while 95.90% accuracy is observed for opcode sequence of length two.

Islam *et al.* (2013) carried out a similar study and provided a hybrid malware classification method which integrates features into a single test, i.e., the static features (printable string information, function length frequency) and the dynamic features (i.e., API function names and parameters) are extracted from dataset of 2939 programs which contains 541 benign files. SVM, DT and RF learning models were used as classifiers and found that random forest outperformed the rest.

Kong and Yang (2013) provided an automated framework that extracts structural features from malware programs to create their associate function call graphs. These graphs essentially provides information about the various system calls made in each function to perform I/O read/write operations. The extracted information are encoded as attributes of the function node in the graph. Moreover, they utilized discriminant distance metric learning and pairwise graph matching in order to compare malware programs that actually compares their features and if they are similar they place it in relevant cluster while keeping other clusters at marginal distance. They trained classifier which based on pair-wise malware distances, places new malware in appropriate clusters automatically. However, their framework again seems like to act as signature-based approach as it may not classify correctly those newly born malware which are not defined in the malware clusters/families. The FCG-driven feature extraction and pairwise graph matching, could be used for malware clustering as well.

Asmitha and Vinod (2014) have proposed a classification model for determining intruded executable linkable files based on machine learning for the growing attacks on unix based machines. Their approach extracts system calls dynamically through a system call tracer utility strace. A classification model is built through benign and intruded malware sample system calls sequences and the trained model is test which reported malware classification accuracy of 97%. The researcher have compared their approach accuracy with those of existing ones.

Avdiienko *et al.* (2015) studied malicious and benign applications treatment with the sensitive data and based on the differences of dealing with sensitive data or the data flow pattern in terms of benign and malicious, the applications are classified into benign or malicious. They called their prototype as mudflow an approach for mining, comparing and classifying the data flow of android applications. They trained classifier with data flows from benign applications only and use the same for detection of novel malware. To test prototype, they processed 10.552 malicious applications which were leaking sensitive data, through mudflow and found that 90% of them are malicious with a positive rate of 18.7%. If the prototype is trained through malaware applications, it could learn and generate more acceptable classification results. Their method is prone to obfuscation as the dataflow pattern of malicious applications were compared to benign, intruders may change the data flow patterns accordingly to mislead the system.

Agrawal and Agrawal (2015) reviewed data mining approaches to detect various attacks. They presented a basic model as shown in Fig. 5 to indicate basic processes outlined in malware analysis. The researches have recommended that although, specific algorithms are developed for dealing malware behaviors but it would be more appropriate to use a hybrid version to design a better framework as everyday we come across 7 newly

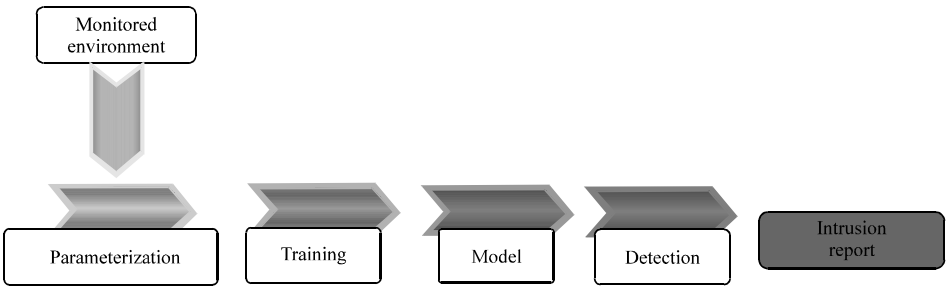

Fig. 5: Reproduced figure of anomaly detection (Agrawal and Agrawal, 2015)

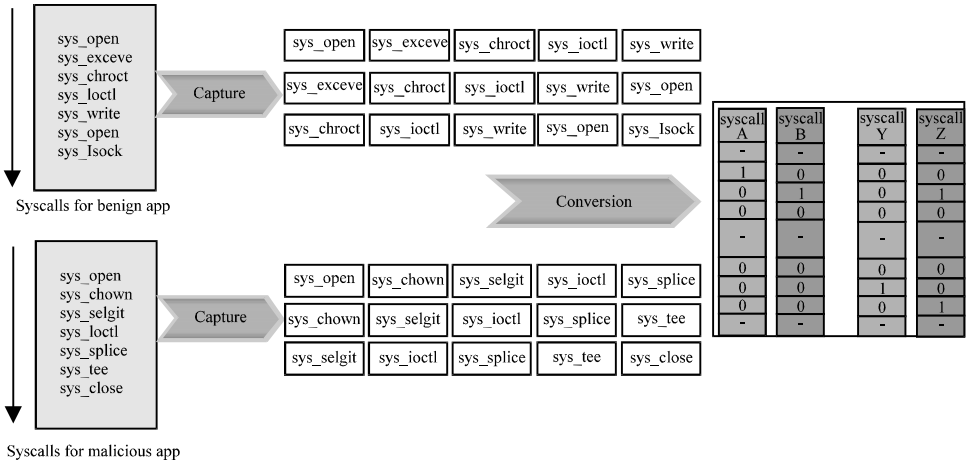

Fig. 6: Capturing system calls from malicious and benign applications and encoding them under one hot encoding scheme (Nauman *et al*., 2016; Christopher Olah, 2015)

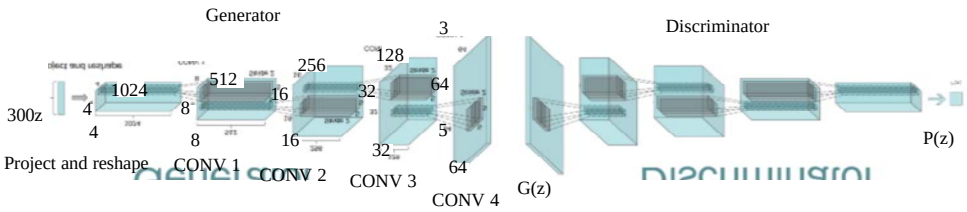

Fig. 7: Layers of generator and discriminator in deep convolutional generated adversarial network (Lipton, 2017)

created malware and using hybrid version we can overcome the deficiency in one algorithm by researcher, e.g., ID3 and C4.5 which are modified form of SVM and decision tree. Fusion is recommended to obtain more accuracy. A lot of research is done on hybrid frameworks (Farid *et al*., 2010; Farid *et al*., 2014; Fu *et al*., 2012).

Mohaisen *et al*. (2015) presents Automated Malware Analysis and Labeling (AMAL) system that is claimed to be more efficient as compared to the existing malware analysis systems. Their framework has two components namely autoMal and malLabel. The autoMal takes into considering the collection of behavioral aspects of malware that require access to memory, registry, network or the file system. These behavioral information are collected in a virtual environment. While MalLabel utilizes those behavioral details to build classifier to classify samples into families that share common characteristics. Their approach achieves 99.5% precision and 99.6% recall for classification of specific families (Fig. 6-9)

Pascanu *et al*. (2015) presented a more better approach for analyzing malware which actually learns the language of malware programs when these are under execution based on time domain features. These features are extracted through the machine learning models recurrent neural networks and echo state networks to learn malware language these techniques are effective in sequence recognition. The unrolled version RNN is presented in Fig. 9. To test the model, a dataset

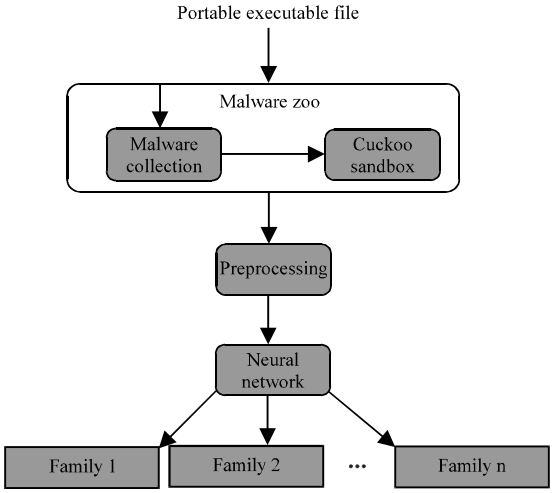

Fig. 8: Malware classification using neural networks (Kolosnjaji *et al*., 2016)

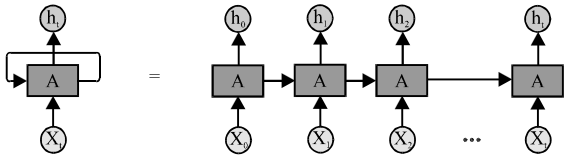

Fig. 9: Unpacked recurrent neural networks (Christopher Olah, 2015)

containing 500000 of malware and benign applications was divided into 29750 for training, 52500 for validating and 150000 for testing. The RNN and ESN with logistic regression and maxpooling provided a classification accuracy of 98%. However, the RNN have an indefinite period of hidden states which are multiplied with weights to produce some other hidden states. On the last hidden state when the gradient is computed and back propagated cause vanishing gradient issue which effects learning as weights (parameters to compute minimum cost of models) are not updated which we need for determining the minimum cost. To overcome these learning issues, Long Short Term Memory (LSTM) is used.

Joshua *et al*. (2015) utilized deep neural networks for malware analysis keeping in view that the earlier machine learning based malware detection methods produce low false positive rates. The approach is scalable to real world examples and research even with commodity hardware. Their evaluation results depict 95% accuracy in detecting malware with false positive rate of 0.1% over a dataset containing 400000 software binaries. Neural networks have incremental learning embedded feature which allows it to train the learning model in batches and upon receiving training examples, the model can be retrained. The network may not perform very well upon receiving perturbed data which is one of the flaws of the neural network.

Kolosnjaji *et al*. (2016) attempted to model the malware system call sequences for the purpose of malware classification as elaborated in Fig. 8. In order to collect best features for malware classification, they formed a neural network based on CNN and RNN. The mostly advanced paradigm neural networks are used to classify malware samples into families into predefined malware.

Moreover, they model system calls using convolutional networks for counting the presence of system calls in a behavioral trace while the recurrent neural networks are utilized as state-full model for learning or memorizing the appearances of system call with relation to previous system calls. Both, the networks are used in a hierarchical fashion for obtaining enhanced detection capabilities. As per evaluation results presented in the paper, model achieves an average accuracy, recall and precision above 90% for malware classification.

In a later study of Kolosnjaji *et al*. (2017) implemented convolutional with feedforward NN that encompasses structure or hidden layers for extraction of useful information. The headers of portable executable files are used as means for extraction of information. Their hybrid neural network is combination of layers of convolution and feedforward, utilizes metadata of PE, imported functions and series of opcodes. These are provided as input to hybrid NN which learn the underline patterns within these executables and after training, applying back-propagation algorithms. They are able to separate malicious programs from those of benign and classify malware into 13 predefined classes. Their approach has yet to determine technique which can further extract most important features for malware classification into their respective families. Moreover, their architecture feeds the convolutional layers with one-hot encoding assembly instructions they reported 93% recall and precision.

MATERIALS AND METHODS

Static, dynamic and hybrid malware analysis: The malware analysis is roughly classified as static, dynamic and hybrid malware analysis (Sharif *et al*., 2008; Moser *et al*., 2007;Schmidt *et al*., 2009; Kim *et al*., 2017; Islam *et al*., 2013). The static analysis does not execute the software for analysis purpose. One of the static analysis tools include PEInfo (Anonymous, 2017) which can extract information or properties from malware code, e.g., histogram, entropy, sectionlength, these information can be used to characterize malware samples. Among the static malware analysis approaches, filters are developed

in order to extract malicious characteristics for unix operating system. Miscreants can manipulate or obfuscate the malicious code from which such information extraction becomes difficult (Linn and Debray, 2003; Moser *et al*., 2007). On the other hand, we have behavior analysis mechanisms which considers to record traces of an activity during execution of malware sample in a controlled virtual environment. It is not guaranteed that a malware will execute unexpected in controlled environment as certain malware require a particular condition to occur or become unpack itself. A dynamic approach is complement to that of static technique as it is less prone to obfuscation (Moser *et al*., 2007). Both types of analysis help us understand risks and intension of the attacker. The approach that utilizes the strengths of both static and dynamic approaches is referred to as a hybrid. The malware detection classification techniques are summarized in Fig. 2.

Machine learning based malware analysis: Machine learning approaches are previously proven effective for discovering various under lying features that are hidden in large scale datasets. A number of domains including natural language processing, computer vision are utilizing machine learning tools like the neural networks which offer superior accuracy in classifying various patterns through its multiple layers which makes it possible to learn deeply. In the subsequent study, we present the growing concept of malware analysis using machine learning techniques.

RESULTS AND DISCUSSION

For this research, various employed techniques are throughly reviewed and presented in preceding study while a summary of machine learning based techniques are elaborated in beneath Table 1. The basic idea of malware analysis and the available malware detection approaches are archived in JCR, IEEE, scopus, github and google scholar. The strings including remote attestation, trusted computing, malware analysis, machine learning based malware analysis, neural networks and its variants in pattern recognition, DCGAN implementation were used to explore the indexes.

Table 1: Details of experimental results of various employed malware detection approaches

References	Detection method	Employed technique	Remarks
Schultz *et al.* (2001)	Static	Applied rule induction algorithm Ripper, NB	Utilized 3 static features for malware classification, i.e., firstly, Portable Executable (PE), the second, byte sequences and third strings data. A data set having 1001 benign and 3265 malicious is fed in to NB and obtained classification accuracy of 97.11%
Kolter and Maloof (2004, 2006)	Dynamic	DT, NB, boosting and SVM	Encodes 1,971 benign and 1,651 malicious executable using n-grams of bytecodes as features and training examples. The Boosted decision trees outperformed other methods with an area under the ROC curve of 0.996
Ye *et al.* (2007)	Static	IMDS, SVM, NB, J4.8	Tested IMDS on 12214 benign and 17366 malicious samples and obtained the following results: TP:1590, TN:1056, FP: 151, FN: 46, detection rate:97.19%, accuracy: 93.07%
Bayer *et al.* (2009)	Dynamic	Clustering algorithm, Locality Sensitive Hashing (LSH)	Clusters similarly behavior to report behavior based on distances with samples. The presented approach is able to cluster 75000 samples in span of three hours approximately
Shafiq *et al.* (2009)	Static	Instance Based Learner (IBk), J48, inductive rule learner (RIPPER) and SVM	Malware datasets form VX Heavens, Malfease and virology lab are used. Their PEMinor framework is able to achieve 99% detection rate with less than 0.5 false alarm in a smaller computation overhead. J48 outperforms the rest in detection accuracy
Tian *et al.* (2009)	Static	IB1, RF, boosting technique Adaboost	Presented a malware classification technique based on printable strings contained within executable. Achieved classification accuracy of 97% using IB1 and RF
Farid *et al.* (2010, 2014), Fu *et al.* (2012), Agrawal and Agrawal (2015)	Hybrid	ID3 and C4.5 as modified form of SVM and decision tree	Fusion is recommended to deliver acceptable performance and to yield accurate results in classification. Modified version of existing approaches includes ID3, C4.5, GA, SVM.
Tian *et al.* (2010)	Dynamic	SMO, IB1, DT, RF	Executes files in virtual environment and collect logs. Experimental results, on malware of size 1368 and cleanware of size 456 files, RF presents the best performance of 97% accuracy in detection of malware from benign. While IB1 is slightly lower in performance
Nataraj *et al.* (2011)	Static	K-nearest neighbor technique with Euclidean distance method	Malware executable are represented as binary strings which may be sequence of zeros and ones to form vectors. The vectors are reshaped to build matrix, so that, to represent executable as images. Results indicate 98% accuracy for classification of malware over a database with 9458 samples having 25 various malware families

Table 1: Continue

References	Detection method	Employed technique	Remarks
Ali *et al.* (2011)	Dynamic	BayesNet, OR, J48	The approach allows the modeling of an application's behavior through stochastic models of machine learning. System calls generated by applications are represented in a hyperspace which possesses complete history of system calls generated. Thusly benign and malicious hypergrams are compared to classify behaviors with AUC of 85.72 with BayesNet.
Islam *et al.* (2013)	Hybrid	RF, DT, SVM, IB1	Used static and dynamic features it is explored that test results depend on malware age. Random forest outperformed the rest in classifying old family of malware with accuracy of 99.821, while on New Family, RF provided 94.407 accuracy
Santos *et al.* (2013)	Hybrid	RF and J48, K-Nearest Neighbour (KNN), BN, NB, SVM	Presented a model based on the frequency of the appearance of opcode sequences. Used instance selection for resampling new ones and feature selection to decrease number of features which ultimately improves machine learning accuracy.
Kong and Yan (2013)	Static	kNN, SVM, malware distance metric learning method, function call graph	The objective is to automatically classify malware instances into their corresponding families through a framework that learns malware distance metrics based on the structural information of labeled malware dataset 526,179 unique malware variants and reported 98.73 accuracy on SVM and 89.32 on kNN. On two sets of experiments with 30 malicious programs and 15 benign, the framework accuracy was found to be 86.67 and 93.33, respectively
Asmitha and Vinod (2014)	Dynamic	NB, AdaboostM1(J48), IBK-5, RF	The approach extracts system calls dynamically using system call tracer utility known as Strace. While the approach identifies best feature set of benign and malware specimens to build classification model. A classification accuracy of 97% is reported to identify malicious samples with feature set of size 30 as the best feature length
Avdiienko *et al.* (2015)	Static	V-SVM	MUDFLOW through first verification recognizes 86.4% of malware as such with a false positive rate of 18.7%. While 90.1% of malware leaking sensitive with a false positive rate of 18.7% with no training on malware data
Mohaisen *et al.* (2015)	Static	SVM, LR, nearest neighbor, decision trees	AMAL achieves 99.5% precision and 99.6% recall for classification of specific families. While during performing clustering, the system achieved precision and recall of 98% for unsupervised clustering
Kolosnjaji *et al.* (2016)	Static	Convolutional neural network, Recurrent neural network	Modeled the malware system call sequences for malware classification and achieved an average accuracy, recall and precision above 90% for malware classification
Kolosnjaji *et al.* (2017)	Static	Convolutional, feed forward with unit PRelu and max pooling dropout as sublayers, Softmax separate layers	They extracted 48 unique functions from portable executable to make able the Neural Network to learn and uniquely identify behavior of malware and benign programs. The convolution is used to capture the semantics of the instruction usage patterns. Their model classify malware into 13 predefined classes and achieve 93% on precision and recall

Proposed framework: We propose use of the one of the most hottest concepts of deep learning, i.e., generative Models specifically the deep convolutional Generative Adversarial Networks (GANs) (Goodfellow *et al.*, 2014; Lipton, 2017; Burns, 2009) that have recently gain marvelous acceptance in comparing adversarial examples. It has a pair of models, i.e., genrator and discriminator as depicted in Fig. 7. The generator receives some sort of data, learns parts of input representations and generates specific data that is similar to the input. The discriminator verifies whether the input is original or its generated. Polymorphic malware are mostly generated version of existing malware that are modified (Table 1).

Experimental data and results: In this study, we explain the division of datasets into training and testing sets and then how a machine learning framework, the DCGAN make use of them for classification. The malware data is collected from the VX hevens virus collection and the Malfease dataset. The system calls are generated from malicious and benign applications in pristine environment and the datasets are created from the same. The dataset of behaviors (sequence of system calls) after conversion to one-hot encoding scheme shall be classified into training set, and test sets with ratio of 2/3 and 1/3, respectively at random to ensure learning and to avoid memorization and look up Table 1, i.e., if we use whole dataset as training set that will be like look up table and model will not perform well when real data is provided to it. Machine might have saved the results somewhere in memory. The k-fold cross validation concept will also be applied for providing more rigorous sampling and to overcome anomalies within datasets.

CONCLUSION

Security of applications and other organizational resources has always remained a top priority of service providers and consumers. A number of techniques are implemented to protect platforms from malicious executions. These techniques includes the traditional Commercial-Ofthe-Shelf Anti-Viruses (COTA AV) also, called signature based techniques, intrusion detection systems, trusted computing and machine learning based approaches. The approaches adopted so far are useful in certain situations while they are not efficient enough for the current security threats. Table 1 provides a detailed survey of employed malware detection approaches. There is a drastic need for developing sophisticated, efficient and smart malware detection and classification technique with high accuracy, i.e. with maximum TP and minimum FP and to improve right classification. It should be real time deployable and nonsignature based malware detection technique. The current implemented machine learning tools that classify clients behavior into malicious and benign are not effective to implement in security centric organizations as these approaches do report false positives and have undesirable TP and FP. State-of-the-art employed machine learning approaches include the neural networks, RNN and ESN. The study reports a review of selected good quality study covering literature era of 2001-2017. Furthermore, the brief idea is presented that we shall use Deep Convolutional Generated Adversarial Networks (DCGANs) for malware analysis. The DCGANs have provided fine-tuned results in pattern recognition in various domains. Indeed DCGANS are more robust as compared to the present machine learning techniques employed. We believe that our approach shall improve and produce more accurate results than previously provided machine learning techniques.

ACKNOWLEDGEMENT

The study is based on PhD research work at MIIT, UniKL. We thank all staff UniKL for extending every possible support.

REFERENCES

Agrawal, S. and J. Agrawal, 2015. Survey on anomaly detection using data mining techniques. Procedia Comput. Sci., 60: 708-713.

Alam, M., T. Ali, S. Khan, S. Khan and M. Ali *et al*., 2012. Analysis of existing remote attestation techniques. Secur. Commun. Netw., 5: 1062-1082.

Alam, M., X. Zhang, M. Nauman, T. Ali and J.P. Seifert, 2008. Model-based behavioral attestation. Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, June 11-13, 2008, ACM, Colorado, USA., ISBN:978-1-60558-129-3, pp: 175-184.

Ali, T., M. Nauman and X. Zhang, 2010. On leveraging stochastic models for remote attestation. Proceedings of the 2nd International Conference on Trusted Systems (INTRUST'10), December 13-15, 2010, Springer, Beijing, China, ISBN:978-3-642-25282-2, pp: 290-301.

Anonymous, 2010. Computer immune systems. University of New Mexico, ýAlbuquerque, New Mexico.

Anonymous, 2017. File statistics. VirusTotal, Dublin, Ireland. https://www.virustotal.com/en/statistics/

Anonymous, 2017. PE infor service. GitHub Inc., San Francisco, California, USA.

Asmitha, K.A. and P. Vinod, 2014. A machine learning approach for linux malware detection. Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), February 7-8, 2014, IEEE, Ghaziabad, India, ISBN:978-1-4799-2900-9, pp: 825-830.

Avdiienko, V., K. Kuznetsov, A. Gorla, A. Zeller and S. Arzt *et al*., 2015. Mining apps for abnormal usage of sensitive data. Proceedings of the 37th International Conference on Software Engineering Vol. 1, May 16-24, 2015, IEEE, Florence, Italy, ISBN:978-1-4799-1934-5, pp: 426-436.

Bayer, U., P.M. Comparetti, C. Hlauschek, C. Kruegel and E. Kirda, 2009. Scalable, behavior-based malware clustering. NDSS., 9: 8-11.

Bodke, A., 2013. Systems and methods for identifying polymorphic malware. US Patent No. US8479291B1, Symantec, California, USA. https://patents.google.com/patent/US8479291B1/en

Burns, J., 2009. Exploratory androidTM surgery. Proceedings of the Black Hat Conference on Technical Security, July 25-30, 2009, iSEC Partners, Inc., San Francisco, California, USA., pp: 1-47.

Cesare, S., Y. Xiang and W. Zhou, 2013. Malwise: An effective and efficient classification system for packed and polymorphic malware. IEEE Trans. Comput., 62: 1193-1206.

Chen Y., R. Venkatesan, M. Cary, R. Pang and S. Sinha *et al*., 2002. Oblivious hashing: A stealthy software integrity verification primitive. Proceedings of the 5th International Workshop on Information Hiding (IH'02), October 7-9, 2002, Springer, Noordwijkerhout, The Netherlands, ISBN:978-3-540-00421-9, pp: 400-414.

Cohen, I.G., S. Hoffman and E.Y. Adashi, 2017. Your money or your patient's life? Ransomware and electronic health records. Ann. Internal Med., 167: 587-588.

Cusumano, M., 2010. Cloud computing and SaaS as new computing platforms. Communi. ACM, 53: 27-29.

Day, C.W., 2007. Intrusion detection system. US Patent No. US7260846B2, SteelCloud Inc., Virginia, USA. https://patents.google.com/patent/US7260846B2/en

Farid, D.M., L. Zhang, C.M. Rahman, M.A. Hossain and R. Strachan, 2014. Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. Expert Syst. Appl., 41: 1937-1946.

Farid, D.M., N. Harbi and M.Z. Rahman, 2010. Combining naive bayes and decision tree for adaptive intrusion detection. Int. J. Network Secur. Appl., 2: 12-25.

Fu, S., J. Liu and H. Pannu, 2012. A hybrid anomaly detection framework in cloud computing using one-class and two-class support vector machines. Proceedings of the 8th International Conference on International Conference on Advanced Data Mining and Applications (ADMA'12), December 15-18, 2012, Springer, Nanjing, China, ISBN:978-3-642-35526-4, pp: 726-738.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu and D. Warde-Farley *et al*., 2014. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems, December 08-13, 2014, ACM, Montreal, Canada, pp: 2672-2680.

Hasselbring, W. and R. Reussner, 2006. Toward trustworthy software systems. Comput., 39: 91-92.

Idika, N. and A.P. Mathur, 2007. A survey of malware detection techniques. Purdue University, Arxan Technologies/21STC.R&T Fund, February 2, 2007. http://www.serc.net/system/files/SERC-TR-286.pdf.

Islam, R., R. Tian, L.M. Batten and S. Versteeg, 2013. Classification of malware based on integrated static and dynamic features. J. Netw. Comput. Appl., 36: 646-656.

Ismail, R., T.A. Syed and S. Musa, 2014. Design and implementation of an efficient framework for behaviour attestation using n-call slides. Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication, January 09-11, 2014, ACM, Siem Reap, Cambodia, ISBN:978-1-4503-2644-5, pp: 36:1-36:8.

Kim, D., A. Majlesi-Kupaei, J. Roy, K. Anand and K. ElWazeer *et al*., 2017. DynODet: Detecting dynamic obfuscation in malware. Proceedings of the 14th International Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA'17), July 6-7, 2017, Springer, Bonn, Germany, ISBN:978-3-319-60875-4, pp: 97-118.

Kolosnjaji, B., A. Zarras, G. Webster and C. Eckert, 2016. Deep learning for classification of malware system call sequences. Proceedings of the 29th Australasian Joint Conference on Artificial Intelligence, December 5-8, 2016, Springer, Hobart, Australia, ISBN:978-3-319-50126-0, pp: 137-149.

Kolosnjaji, B., G. Eraisha, G. Webster, A. Zarras and C. Eckert, 2017. Empowering convolutional networks for malware classification and analysis. Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN'17), May 14-19, 2017, IEEE, Anchorage, Alaska, ISBN:978-1-5090-6183-9, pp: 3838-3845.

Kolter, J.Z. and M.A. Maloof, 2004. Learning to detect malicious executables in the wild. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Seattle, Washington, August 22-25, 2004, pp: 470-478.

Kolter, J.Z. and M.A. Maloof, 2006. Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res., 7: 2712-2744.

Kong, D. and G. Yan, 2013. Discriminant malware distance learning on structural information for automated malware classification. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 11-14, 2013, ACM, Chicago, Illinois, USA., ISBN:978-1-4503-2174-7, pp: 1357-1365.

Liao, H.J., C.H.R. Lin, Y.C. Lin and K.Y. Tung, 2013. Intrusion detection system: A comprehensive review. J. Network Comput. Applic., 36: 16-24.

Linn, C. and S. Debray, 2003. Obfuscation of executable code to improve resistance to static disassembly. Proceedings of the 10th ACM Conference on Computer and Communications Security, October 27-30, 2003, ACM, Washington D.C., USA., pp: 290-299.

Lipton, Z.C., 2017. Deep convolutional generative adversarial networks. GitHub Inc., San Francisco, California, USA.

Mehdi, B., F. Ahmed, S.A. Khayyam and M. Farooq, 2010. Towards a theory of generalizing system call representation for in-execution malware detection. Proceedings of the 2010 IEEE International Conference on Communications (ICC'10), May 23-27, 2010, IEEE, Cape Town, South Africa, ISBN:978-1-4244-6402-9, pp: 1-5.

Moein, S., F. Gebali and I. Traore, 2014. Analysis of covert hardware attacks. J. Convergence, 5: 26-30.

Mohaisen, A., O. Alrawi and M. Mohaisen, 2015. Amal: High-fidelity, behavior-based automated malware analysis and classification. Comput. Secur., 52: 251-266.

Moser, A., C. Kruegel and E. Kirda, 2007. Exploring multiple execution paths for malware analysis. Proceeding of the IEEE Symposium on Securit and Privacy, May 20-23, Berkeley, CA, pp: 231-245.

Moser, A., C. Kruegel and E. Kirda, 2007. Limits of static analysis for malware detection. Proceedings of the 23rd Annual Conference on Computer Security Applications (ACSAC'07), December 10-14, 2007, IEEE, Miami Beach, Florida, ISBN:978-0-7695-3060-4, pp: 421-430.

Nataraj, L., S. Karthikeyan, G. Jacob and B.S. Manjunath, 2011. Malware images: Visualization and automatic classification. Proceedings of the 8th International Symposium on Visualization for Cyber Security, July 20, 2011, ACM, Pittsburgh, Pennsylvania, USA., ISBN:978-1-4503-0679-9, pp: 1-7.

Nauman, M., N. Azam and J. Yao, 2016. A three-way decision making approach to malware analysis using probabilistic rough sets. Inf. Sci., 374: 193-209.

Pascanu, R., J.W. Stokes, H. Sanossian, M. Marinescu and A. Thomas, 2015. Malware classification with recurrent networks. Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'15), April 19-24, 2015, IEEE, Brisbane, Australia, ISBN:978-1-4673-6997-8, pp: 1916-1920.

Rowett, K. and S. Sikdar, 2005. Intrusion detection system. US Patent No. US20050216770A1, Gigafin Networks Inc., Cupertino, California, USA. https://patents.google.com/patent/US20050216770A1/en

Santos, I., F. Brezo, X. Ugarte-Pedrero and P.G. Bringas, 2013. Opcode sequences as representation of executables for data-mining-based unknown malware detection. Inf. Sci., 231: 64-82.

Scheidell, M., 2009. Intrusion detection system. US Patent No. US7603711B2, SECNAP Network Security, Boca Raton, Florida, USA. https: //patents. google. com/patent/US7603711B2/en

Schmidt, A.D., R. Bye, H.G. Schmidt, J. Clausen and O. Kiraz *et al*., 2009. Static analysis of executables for collaborative malware detection on android. Proceedings of the 2009 IEEE International Conference on Communications (ICC'09), June 14-18, 2009, IEEE, Dresden, Germany, ISBN:978-1-4244-3435-0, pp: 1-5.

Schultz, M., E. Eskin, E. Zadok and S.J. Stolfo, 2001. Data mining methods for detection of new malicious executables. Proceedings of the IEEE Symposium on Security and Privacy, May 14-16, IEEE Computer Society Washington, DC, USA., pp: 38-49.

Shafiq, M.Z., T.S. Momina, F. Mirza and M. Farooq, 2009. PE-Miner: Mining structural information to detect malicious executables in real time. Proceedings of the Recent Advances in Intrusion Detection, September 23-25, 2009, France, Springer, pp: 121-141.

Sharif, M., V. Yegneswaran, H. Saidi, P. Porras and W. Lee, 2008. Eureka: A framework for enabling static malware analysis. Proceedings of the 13th European Symposium on Research in Computer Security, October 6-8, 2008,: Springer, Malaga, Spain, ISBN:978-3-540-88312-8,-pp: 481.

Tian, R., L. Batten, R. Islam and S. Versteeg, 2009. An automated classification system based on the strings of Trojan and virus families. Proceedings of the 4th International Conference on Malicious and Unwanted Software (MALWARE'09), October 13-14, 2009, IEEE, Montreal, Canada, ISBN:978-1-4244-5786-1, pp: 23-30.

Tian, R., R. Islam, L. Batten and S. Versteeg, 2010. Differentiating malware from cleanware using behavioural analysis. Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE), October 19-20, 2010, IEEE, Nancy, France, ISBN:978-1-4244-9353-1, pp: 23-30.

Willems, C., T. Holz and F. Freiling, 2007. Toward automated dynamic malware analysis using cwsandbox. IEEE. Secur. Privacy, 5: 32-39.

Ye, Y., D. Wang, T. Li and D. Ye, 2007. IMDS: Intelligent malware detection system. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 12-15, 2007, ACM, San Jose, California, USA., ISBN:978-1-59593-609-7, pp: 1043-1047.

You, I. and K. Yim, 2010. Malware obfuscation techniques: A brief survey. Proceedings of the 2010 International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA'10), November 4-6, 2010, IEEE, Fukuoka, Japan, ISBN:978-1-4244-8448-5, pp: 297-300.