MDFRCNN: Malware Detection using Faster Region Proposals Convolution Neural Network

Mahendra Deore[1]*, Uday Kulkarni[2]

[1] Department of Computer Engineering, MKSSS's Cummins College of Engineering for Women, Pune-411052 (India)
[2] Department of Computer Science & Engineering, SGGS Institute of Engineering and Technology, Nanded-431606 (India)

Received 31 August 2020 | Accepted 14 June 2021 | Early Access 30 September 2021

Abstract

Technological advancement of smart devices has opened up a new trend: Internet of Everything (IoE), where all devices are connected to the web. Large scale networking benefits the community by increasing connectivity and giving control of physical devices. On the other hand, there exists an increased '*Threat*' of an '*Attack*'. Attackers are targeting these devices, as it may provide an easier '*backdoor entry to the users' network*'. **MALicious softWARE** (MalWare) is a major threat to user security. Fast and accurate detection of malware attacks are the sine qua non of IoE, where large scale networking is involved. The paper proposes use of a visualization technique where the disassembled malware code is converted into gray images, as well as use of Image Similarity based Statistical Parameters (ISSP) such as Normalized Cross correlation (NCC), Average difference (AD), Maximum difference (MaxD), Singular Structural Similarity Index Module (SSIM), Laplacian Mean Square Error (LMSE), MSE and PSNR. A vector consisting of gray image with statistical parameters is trained using a Faster Region proposals Convolution Neural Network (F-RCNN) classifier. The experiment results are promising as the proposed method includes ISSP with F-RCNN training. Overall training time of learning the semantics of higher-level malicious behaviors is less. Identification of malware (testing phase) is also performed in less time. The fusion of image and statistical parameter enhances system performance with greater accuracy. The benchmark database from Microsoft Malware Classification challenge has been used to analyze system performance, which is available on the Kaggle website. An overall average classification accuracy of 98.12% is achieved by the proposed method.

Keywords

Malware, CNN, Faster RCNN (F-RCNN), Classification, Malware Static, Dynamic Analysis.

DOI: 10.9781/ijimai.2021.09.005

I. Introduction

MALWARE is a major menace to Internet security today. There are various distinctive sorts of cyber assaults in the current day digital world. A few of these are very renowned like, phishing sites, botnets, denial of service (DoS), malware assaults and so on. Lately malware attacks are being increasingly propagated because of the huge development of internet and web-based products like IoE. A report from Symantec in 2019 announced a new malware technique i.e., FormJacking (FJ). Cyber attackers inject malware code in web page forms (specifically, payment page forms handled by Payment Processors) to steal sensitive information about the payment cards, names, addresses, phone numbers, etc. These types of attacks are called 'Supply Chain Attacks' (SCA), and are written in JAVA. According to the Symantec report, an average of approximately 4800 sites have been compromised by the FJ code and there is a 78% hike in SCA. '***Jacking***' is popular amongst cyber attackers. There are varieties of jackings viz. cyber, crypto, form, page, Brand, I, Wi, page, thread, mouse, paste, Data, side, Bio, Juice, etc.

* Corresponding author.
E-mail address: mdeore83@gmail.com

Big data Analytics can be defined as the process of lookup, processing, storing enormous data so as to separate important information out of it. With growth in big data analytics, security and protection concerns are additionally amplified.

Big data servers are effectively open to a more extensive population base; consequently, they increase the possibility of malware attacks. To shield users from the hazards of malware, security companies offer a diverse set of antivirus tools. Usually, these tools follow signature-based methodologies. Signature based recognition is inclined to a few difficulties. For example, there has to be a database with patterns of known sets of threats. Also, frequent refresh is required for these signatures in the repositories which requires the intervention of experienced staff in the signature creation process. Thus, antivirus organizations are not able to define, analyze and develop effective signature patterns.

Due to the escalating growth of online transactions, the level and number of cyber-crimes are increasing. In the present situation, where malware assault is massively expanding, it is very troublesome for pattern matching scanners to recognize new variations of existing malicious programs. Therefore, there is a high demand to formulate other strategies to recognize malware.

Please cite this article in press as:
M.Deore, U. Kulkarni. MDFRCNN: Malware Detection using Faster Region Proposals Convolution Neural Network, International Journal of Interactive Multimedia and Artificial Intelligence, (2021), http://dx.doi.org/10.9781/ijimai.2021.09.005

This requirement asks for a Malware Detection System (MDS) which can detect malware accurately and act fast enough to quarantine the same. MDS is traditionally feature vector based in which crucial characteristics of the malware are extracted and used to identify the same in real time systems. Behavioral based malware detection can broadly be divided into three categories namely, static, dynamic and hybrid. According to Gandotra [1] in the Static Analysis (SA), malware software is analyzed without being executed. SA typically extracts features like operational code (OPCODE) frequency distribution, control flow graph, syntactic library call, byte-sequence n-grams, string signature etc., after unpacking executable in advance. SA protects the Operating System (OS) from malicious damage but it is vulnerable to code obfuscation techniques. In Dynamic Analysis (DA) malware is executed by making use of a controlled environment viz. sandbox, emulator, simulator, virtual machine and then it is analyzed by monitoring tools like Capture Bat, Process monitor (.pmon), poison IVY, etc. Sandbox generates a detailed and extensive report which requires human interpretation and analysis. The analysis process can be automated to a greater extent but with an addition of extensive computational complexities. Thus, it is time consuming [3]. However, both static and dynamic analyses have some limitations and it is difficult to use either static or dynamic analysis for malware detection. The next approach is the combination of both which is called the hybrid approach. It analyzes the signature of malware in the first phase and combines it with behavioral specifications for a complete analysis.

Malware database is huge. Such systems are largely dependent on Machine Learning (ML) algorithms. To figure out malware patterns in the code successfully, effective solution is - Visual Analytic Technique (VAT), where malware patterns are presented as an image. VAT provides summarized pictures of attacks. These images can be trained using ML for analyzing malware patterns. Malware Detection Developer (MDD) focuses on image patterns due to two reasons. The first reason is, even though the malware developers work in the direction of hiding the code, at the same time, while coming up with variants of the malware, they use the same old code. Therefore, the deviation δ between two images of a single malware family is very small. MDD can take advantage of this mind set and use the similarity mining machine learning method to identify the family of malware. The second reason is an image classification technique which is more mature and faster [4] [5].

This paper presents literature survey in Section III and the Key extract is malware can be packed using different packing methods and with different resolutions. Therefore, for VTA (image) based analysis, research options are still open for providing an improved solution to classify malware. Keeping this as a base, this paper proposes extracting ISSP features for all the malware families taking into consideration all the variants in the binary file. Finally, ISSP features and malware images are trained using fast and robust F-RCNN classifier.

The rest of the paper is organized as follows: related work is presented in Section II. The proposed work model is presented in Section IV. Feature vector formation using perceptual features and the mathematical model of the F-RCNN classifier is described in sections V and VI. Computation of the statistical parameters is described in Section VII. Description of the database is given in Section VIII. Experimental setup and related results are presented in Section IX. Section X discusses the performance analysis of the proposed method. Section XI concludes the paper.

II. Related Work

This section focuses on varieties of features and classification techniques explored by researchers. Related work can be bifurcated in two categories i.e. the work based on image representation and methods other than image representation. As paper proposes image representation of malware, work related to this method is explained first.

A. Image Based Methods

Image is a 2D representation. Key points of 2D representations are as follows

1. Data dimension does not affect processing once similarity space is formed.
2. Equally important clusters are formed.
3. Similar clusters are displayed adjacent for clear visualization [27] [28][29].

Malware analysis using Visualization Technique (VT) was proposed by Yoo [16]. He classified images using Self-Organizing Map. VT is mostly used for document and image analysis where files are huge and data is massive. Therefore, it has wide applications in computer security, as malware attacks are in the thousands at a time. Shiravi [35] and N. Diakopoulos [34] identified Brute Force attack on Secure Shell (SSH) by representing details of Internet Protocol (IP) address, UserIDs and various anomalies using varieties of colors. VT was used to display large network packets and helped security analysts to find similarities by checking minute details using a zoom option. S.Foresti [39] and M. Wagner [40] demonstrated usage of VT to represent information like time ('when'), IP address ('Where'), Data ('what') and estimated distances to other hosts.

Quist [17] proposed the use of an Ether hypervisor framework to track and visually represent execution of malware programs. The dynamic analysis framework was named as VERA. Trinius [18] introduced a new concept i.e., Malware Instruction SeT (MIST) for monitoring malwares. They used a CW Sandbox to collect information regarding API calls and performed action. They visually represented distance matrices of features for five malwares.

To improve malware detection, different sections of the binary executable are now represented as gray scale images. These images provide detailed structure of malware, to the extent that they show even small changes in the code, without altering the remaining code structure. Gray scale texture helps in identifying similar patterns of the binary code [41]. L. Nataraj [42] proposed GIST descriptors to classify obfuscated malware.

The Function Length Frequency (FLF) algorithm proposed by Tian [9] was used to detect Trojan after surveying varieties of techniques. Zolkipli [10] suggested the use of Variable Length Instruction Sequences (VLIS) with machine learning algorithms. Shankarapani [11] proposed two models, namely, Malware Examiner using Disassembled Code (MEDiC) and Static Analyzer for Vicious Executables (SAVE). Results were promising as the model had better detection even if malware is obfuscated. Nataraj [19] presented malware binaries as gray scale images. Using a KNN classifier they achieved good average accuracy along with increased speed of malware detection. On a similar line Kancherla [20] used byte plot (image of executables) and achieved 95% accuracy using the SVM classifier.

Kong and Yan [12] used hex dump n-gram, disassembly code, PE header and selected features using the L1 regularized method. They applied different classifiers viz. NB, SVM, K means Neural Network (KNN), decision tree and analyzed the performance of all the features. They concluded that the PE header feature is more prominent in malware detection. Santos et al., 2013b [13] tried to figure out the relevance of each OPCODE and calculated the frequency of OPCODE sequences. They verified the performance using the same four classifiers used by previous researchers. They stated empirically that the model can detect unknown malware as well.

Similarity is calculated based on the distance between each and every pair of points. Minimum distance represents maximum similarity [30] [31]. Projection and semantic orientation are 2D VTs, normally used to check similarity patterns [26] [32] [33].Frequency domain-based feature extraction i.e., Wavelet transforms, was proposed by Gu [14]. Li and Li [15] proposed static features viz. API, classes, functions and packages detecting malware for android APKs. They used different layers of 'Characteristic Tree' containing information of API calls. The method can classify unknown APKs. Han [21] converted a Windows PE binary file into a gray scale image. After that, using an Entropy Graph Generator (EGG), they calculated the entropy of each and every line of an image. Malware detections were done based on similarities of the original binary file.

Arefkhaniet [22] introduced a Local Sensitive Hashing technique specific to image processing, to classify similar input (malware) with high probability. Wu [23] converted disassembled binary executable into opcode sequences into an image. They used PCA to reduce the dimensionality and KNN classifier. Rezaei [24] proposed a similarity measurement algorithm for malware detection which compares the opcode strings of malicious files to improve the detection rate and speed. Venkatraman [36], Zhang [37] and Wylie Shanks [38] explored usage of VT for analyzing malware attack chronology and demonstrated successful system connections with the help of colors.

B. Other Methods

Santos et al., 2013b [13] introduced an OPCODE Executable trace Malware (OPEM) framework. It is a hybrid of statistically obtained frequency of occurrence of OPCODE and dynamically obtained executable traces. Performance was evaluated using Baysine, Decision Tree, SVM and KNN classifiers. Kolosnjaji [25] proposed feature fusion of headers of PE files and convolution of n-grams of an instruction. They achieved a 93% recall rate and accuracy using SVM and ANN (Feed forward) classifier. Ripper Cohen [6] proposed the Repeated Incremental Pruning to Produce Error Reduction (RIPPERk) algorithm that supersedes the learning algorithm, Incremental Reduced Error Pruning (IREP). In RIPPERk, k represents the number of multiple optimization iterations. Schultz [7] made use of three static features (byte sequence, strings and Portable Executable (PE)) and for the first-time used the data mining concept. Kolter [8] proposed a combination of n-gram (instead of non-overlapping byte sequence) features with classifiers viz. Naive-Bayes (NB), Support Vector Machine (SVM), Decision Tree and their boosted versions. They found that the boosted decision tree provides better results.

The following section illustrates an extensive and organized literature summary of innovative techniques proposed by researchers and challenges from state of the art.

III. Literature Survey – Formulating Problem Statement

A research problem should represent the core subject matter and it should be a discovery of new knowledge. This objective not only asks for rigorous literature survey, but also demands interpretation of the surveyed information to achieve a proper research path. Graphical presentation is given by the author which makes it more suitable to extract the required information.

Fig. 1 provides information about three basic analysis techniques like SA, DA and hybrid, explored by researchers. The SA technique is still preferred by most of the researchers [50]. The hybrid approach is not still popular amongst researchers.

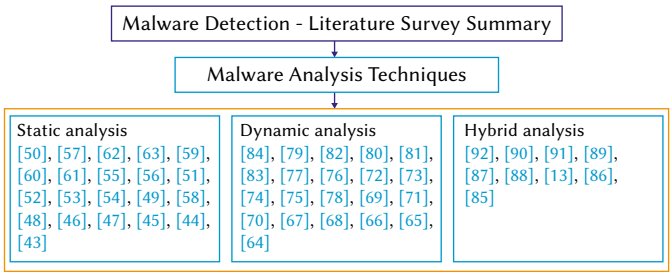


Fig. 1. Survey depicts the type of analysis technique explored by researchers.

Feature extractions and classification techniques are the two pillars of MDS. Fig. 2 is devoted to different feature vectors used by researchers. API calls, system calls, n-gram and OPCODE are still features that are mostly used in MDS. Researchers typically have two paths: the first one is to optimize the feature vector and get a significant limited feature set, and the second one is the selection of a prominent limited set of features manually. Work implemented by researchers in either way is unique itself and uses varieties of byte and hex related features.

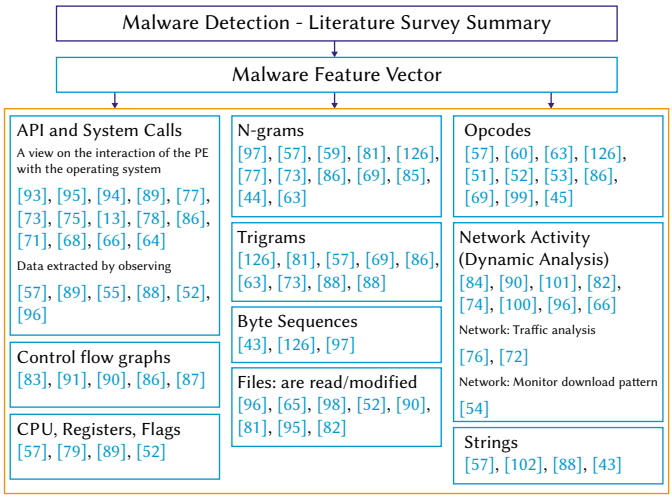


Fig. 2. Survey depicts varieties of feature used by researchers.

A feature vector extracted from malware code should be trained with the help of neural networks. After training, system performance is tested by applying real time malware data. Researchers developed different techniques to extract the feature set. A breakthrough from the survey reveals that researchers using available crawlers, filters etc., end up with a *plethora of* feature vector space which may be redundant. Many times, some features may worsen the accuracy of the system. Thus, feature selection to improve system performance is mandatory but the same should be done without compromising accuracy.

The role of a classifier is important as it defines accuracy and precision. Fig. 3 presents three basic learning process categories i.e., supervised, unsupervised and semi-supervised. Supervised learning is the first choice of researchers where malware annotations are given for training the network.

Fig. 4 presents the wide usage of deep learning techniques like CNN, Deep Neural Network (DNN), Recurrent Neural network (RNN), auto encoders etc. in malware detection. These techniques are well established and provide high performance. CNN and its extensions are preferred for image-based analysis therefore, it is mostly used by researchers. CNN variants viz. Region based CNN (R-CNN), Fast R-CNN, Faster R-CNN are mostly used in image analysis for detecting objects [182].

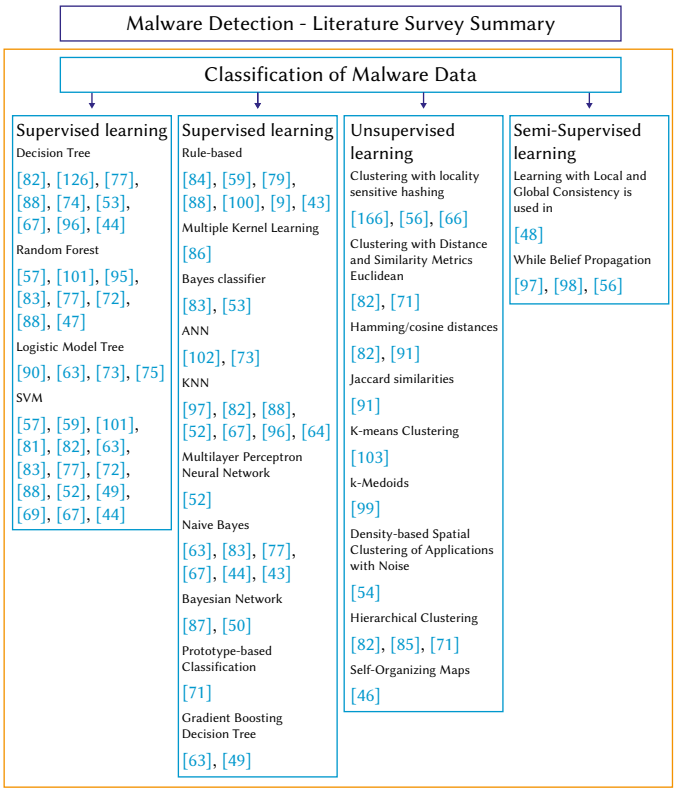


Fig.3. Survey depicts different classifiers explored by researchers.

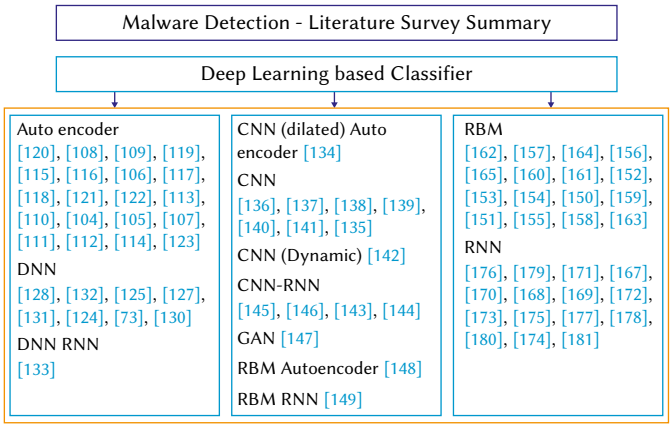


Fig. 4. Literature survey based on Deep Learning Classifier.

These techniques are specific to image analysis and provide less training and testing times, which is the need of a malware detection technique, as it has to run in real time and detect malware as fast as possible. This motivates the author to select FR-CNN. It is a technique with a significantly low computational cost and the same has not been investigated for malware detection. This technique achieves more precision and a faster response.

IV. Proposed Model of MDS

The paper proposes MDS architecture comprising of deep learning network to accurately detect and classify malware families using an image-based technique which is described in Fig. 1. The Benchmark database from Kaggle is used to evaluate the performance of a system. The features vector of malware families is presented as gray scale images. These images will be trained using deep learning and facilitate adaptive learning in real time environment to achieve high accuracy.

The Main contribution of this research work is as follows:

1. *Consideration of prominent static features e.g., string signature, byte-sequence, N-grams, OPCODE*
2. *Represent feature vector as a gray scale image reflecting the malware family behavior*
3. *Arrange feature vector sub parts as varieties of 'Regions' of an image*
4. *NOTE: 'Region' is a generic term. Rectangular regions are considered in this paper, as it is common.*
5. *Minimize the training set*
6. *Compute statistical parameters of the image generated in point-2*
7. *Apply the F-RCNN classifier to a matrix having statistical parameters as well as an image.*

To the best of our knowledge, the above combination i.e., a matrix of static features and an image with F-RCNN classifier has not been evaluated by researchers.

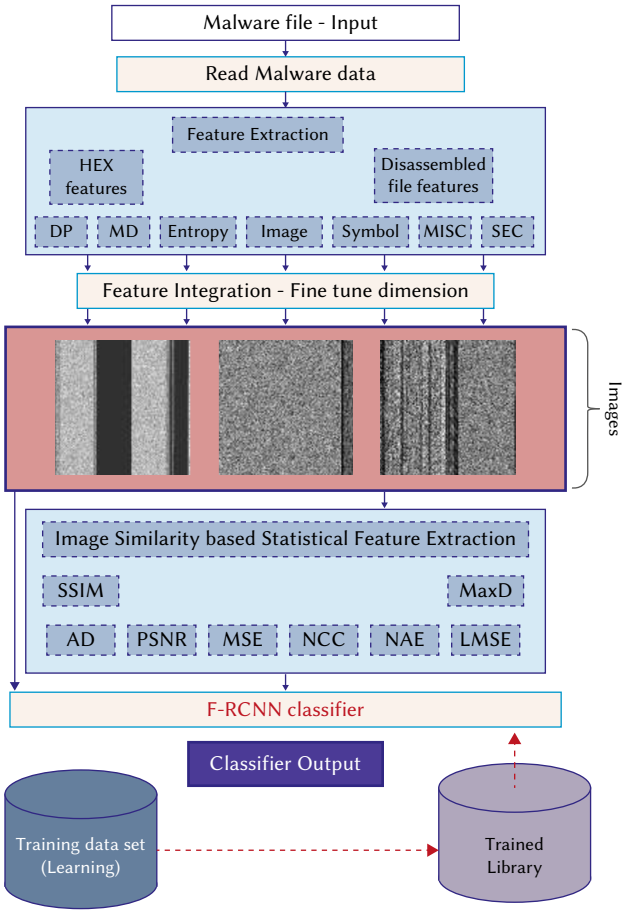


Fig. 5. System Architecture of the proposed MDF-RCNN.

Fig. 5 depicts the system architecture of the proposed MDS. Feature extraction and classification of the malware input file are two major modules of MDS. Each one is described as follows.

V. Feature Extraction Module

A. Features Based on HEX and Disassembled Files

There are basically two major types of features extracted for MDS. These are HEX dump-based features (n-gram, MD1, entropy, image 1(haralick) and Image 2 (lbpfeatures)) and disassembled file features (meta-data, SYM, OPC, MISC, DP and SEC.)

1. Hex Based Features

a) N-gram

An N-gram is a contiguous sequence of n items from a given sequence. The technique is used intensively for characterizing sequences. The sequences may be from a speech or text. Malware samples may be viewed as sequences of HEX value. An N-gram analysis of these HEX values may provide valuable information. The malware sample consists of special symbols viz.??, indicating that the data of that location is not initialized. It also contains byte sequence whose value may range from 0 to 255.

b) Entropy

Entropy (ENT) is basically a measure of randomness or uncertainty or maybe the O amount of disorder. Obfuscation presence can be detected by entropy [183] [184]. In MDS, entropy is calculated based on byte representation. It measures the disarray of the distribution of the bytes in malware code by setting *'order'* and *'randomness factor'*. A sliding window is applied on the malware code and entropy is calculated for each windowed segment. It is represented by:

$$E = e_i, Number\ of\ windowed\ segment\ i = 1, \ldots\ldots, N \quad (1)$$

The Shannon's formula,

$$e_i = -\sum_{k=1}^{m} f(k) * \log_2 f(k) \quad (2)$$

$f(k) =$ *frequency of byte k with window segment.*
$m =$ *number of distinct bytes in the window segment.*

2. Features Extracted From Disassembled Files

Malware executables must first be disassembled to extract features. The PE format is used by Windows OS (WOS). It's basically a data structure encapsulating important information which will be utilized by the WOS loader to manage the wrapped executable code shown in Fig. 6. The PE file contains *Headers* viz. DOS header, Section table, optional header, PE header, and *Sections* viz. code, imports, data. The Dynamic linker uses this information to map the file into memory. The PE information is important as its basic structure manages memory protection based on the code or data region.

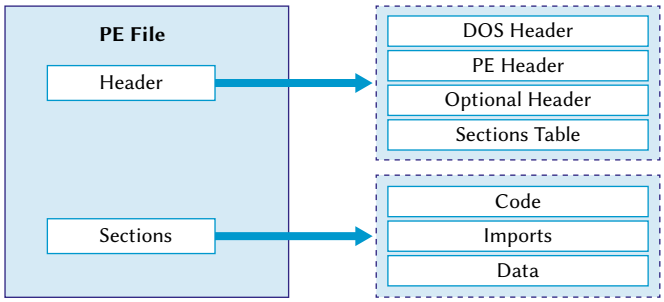


Fig. 6. PE File Structure.

a) Section (SEC)

The '*sections*' consist of code, data and import sections. Further classification includes .data, .idata, .edata, .rdata, .text, .bss, .rsrc, .relocand, .tls. A malware code uses packing techniques where it modifies default sections and may create new sections, to evade Metadata (MD) shown in Fig. 6. An MD program can generate a feature vector by detecting different characteristics of SEC.

b) Metadata

After disassembling, two features viz. the number of lines in the file and file size, are computed and included within the Metadata category (MD2).

c) Symbol:

In malware sample code a set of symbols like [, -, +,], @, ?etc. may be present. Actually, these may correspond to indirect calls or Dynamic Library Loading (DLL), in malware code. In an indirect call, the address of the subroutine is loaded from memory or register. According to [188], the function call depends upon the architecture as well as the optimal decision of a compiler; therefore, such indirect calls may reveal information about data obfuscation. DLL is loaded during runtime by the executable code and executes library functions based on their address. Static analyzers cannot capture such run time events. Therefore, these garbled characters are used specifically by malware developers to evade MD.

d) Operation Code (OPCODE)

OPCODE or mnemonic is a digit and denotes assembly code. The micro-processor executes the OPCODE; therefore, it plays a very important role as it describes the behavioral characteristics of malware as shown in Fig. 7. Machines are x86 based. The instruction set list is complex and large, therefore [189], selected 93 OPCODEs based on frequent use in the malicious application and its commonness. The OPCODE frequency is calculated from the malware code. According to [190], use of Instruction Replacement Technique (IRT) may evade detection. Santos [53] used only OPCODE based features to generate the feature vector and detected malware just by one single class with reasonable amount of accuracy. Thus, it proves that OPCODE based features can contribute more for MD. Researchers also suggested an OPCODE, n-gram based method for MD. The detection is based on the OPCODE frequency feature, calculated by Term Frequency–Inverse Document Frequency (TF-IDF) statistical technique. The OPCODE sequence given vector was used to train the SVM classifier.

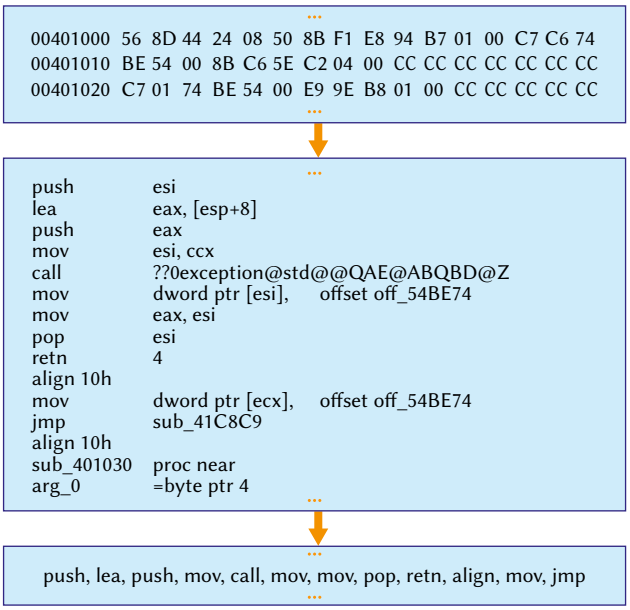


Fig. 7. Disassembly File Structure.

e) Register (REG)

The microprocessor has an internal register set which may be used for a specific task. According to [191], in some situations, registers are renamed to make the MD process more complicated and confusing. This asks for keeping track of REG used and frequency of usage of those registers. This feature is useful and helps in detecting a family of malware.

f) Data Define (DP)

Few malware programs use a packing technique and therefore do not use API calls. Instead they contain a few OPCODEs. Typically, they use data related assembler directives like Define Byte (db), Define Word (dw) and Define Double Word (dd). This feature is key for classifying varieties of malware families.

g) Miscellaneous (MISC)

This feature should be selected manually by identifying keywords from the disassembled code. The Interactive Disassembler (**IDA**) tool may be used for the same. Features extracted will be like the number of imported DLLs, identifying strings viz. **hkey_local_machine**(it specifies access to specific paths of the Windows registry), number of blocks in the PE etc. Thus, it depends upon the experience of the MD software developer engineer.

After extracting these features gray scale image is prepared with the help of feature coefficients. Refer Fig. 8 for the same. Typically, these images of a single family should show similarity as the changes in the malware code is not drastic. Consider this point as a base author motivated to compute image similarity based statistical parameters. The next section describes the same.

B. Features - Image Similarity Based Statistical Parameters

This feature set focuses on similarity between two images. A similarity parameter matrix will be computed based on malware images from the '*x*' family compared with itself as well as the remaining families. Reference image (R_I) and Input image (I_I) are two input images. Suppose R_I is from the '*x*' family then I_I will be the remaining images from the '*x*' family and images from the other families. R_I will be constant throughout the process of computing the similarity parameters. As the number of images per family are in the thousands their mean value will be calculated.

The NCC method is used for template matching which is a process used for finding incidences of a pattern or object within an image. Eq. (1), is used to calculate NCC.

$$NCC(R_I, I_I) = C_{R_I I_I}(\widehat{R_I}, \widehat{I_I}) = \sum_{[m,n]\in R} \widehat{R_I}(m,n)\widehat{I_I}(m,n) \quad (3)$$
$$Where,\ \widehat{R_I} = \frac{R_I - \overline{R_I}}{\sqrt{\sum(R_I - \overline{R_I})^2}},\ \widehat{I_I} = \frac{(I_I - \overline{I_I})}{\sqrt{\sum(I_I - \overline{I_I})^2}}$$

AD provides the average of change concerning the input image and the reference image. AD can be expressed as follows:

$$AD(R_I, I_I) = \frac{1}{mn}\sum_{m=1}^{M}\sum_{n=1}^{N}[R_I(m,n) - I_I(m.n)] \quad (4)$$

MaxD provides the maximum of the error signal (i.e., the difference between the processed and reference image). MD is defined as follows:

$$MaxD(R_I, I_I) = \max\{[R_I(m,n) - I_I(m.n)]\} \quad (5)$$

SSIM is based on three factors i.e., luminance, contrast, and structure to better suit the workings of the human visual system. It is a perceptual metric that quantifies image quality degradation. This parameter is selected as the malware developer makes changes in the old code and comes up with the modified code. The modified code can be thought of as the '*Noise*' element in an image. SSIM is defined as follows:

$$SSIM(R_I, I_I) = \left[l(R_I, I_I)^{\alpha} .\ c(R_I, I_I)^{\beta} .\ s(R_I, I_I)^{\gamma}\right] \quad (6)$$

where l = luminance, c = contrast, s = structure

The Laplacian error map shows spatial error distribution across an image. The overall image quality is given by LMSE as follows:

$$LMSE(R_I, I_I) = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}[L(R_I(m,n)) - L(I_I(m.n))]^2}{\sum_{m=1}^{M}\sum_{n=1}^{N}[L(R_I(m,n))]^2} \quad (7)$$

where L((m,n)) is the Laplacian operator

NAE measures the numerical variance between the R_I and I_I. Moreover, the results that are near to zero means that the image has a high similarity to the original one and the results near the value one indicate that the image has a very poor quality. NAE is calculated as follows:

$$NAE(R_I, I_I) = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}|R_I(m,n) - I_I(m.n)|}{\sum_{m=1}^{M}\sum_{n=1}^{N}[R_I(m,n)]} \quad (8)$$

MSE and PSNR are used to compare the quality of the image compression. MSE represents the cumulative squared error between the R_I and I_I, whereas the PSNR represents a measure of the peak error. The lower the value of the MSE, the lower the error.

$$MSE(R_I, I_I) = \frac{1}{mn}\sum_{m=1}^{M}\sum_{n=1}^{N}[L(R_I(m,n)) - L(I_I(m.n))]^2$$
$$PSNR = 10 * \log_{10}\frac{255^2}{MSE} \quad (9)$$

After computing ISSP one more feature vector is produced which will be used to train classifier.

VI. Classifier – F-RCNN

This section describes the reason for selecting the classifier i.e., F-RCNN. In computer vision (CV) object or region detection is a major task. Ross [182] proposed a selective search method to extract *N* number of limited regions from an image. These regions are referred to as *Region Proposals*. Associated training network is referred as *Region Proposal Network* (RPN). The algorithm overrides the problem of selecting a huge number of regions. Region based CNN (R-CNN) is thus a fusion of the *Region Proposals* algorithm with CNN. As the first step, this algorithm selects some proposed regions from the image, puts Bounding Boxes (BB) and labels their categories. The Deep learning algorithm (CNN) extracts varieties of features using forward computation from these proposed regions and then trains the network to classify the categories and BB. Following section describes methodology to represent feature coefficients as an image.

A. Image Representation

Malware samples are represented as an image, where each byte of the malware code corresponds to one pixel of gray scale image [185]. The author proposes use of all the features extracted viz. n-gram, MD1, MD2 etc., to construct an image as shown in Fig. 8. The image is formed in such a way that there are ***'regions'*** of feature vectors. ***Each feature vector (N-gram, MD1, entropy, MD2 etc.) may be viewed as a region of an image. Thus, fundamentally the R-CNN algorithm is more suitable for MDS. R-CNN*** will be more effective and efficient for classifying malware.

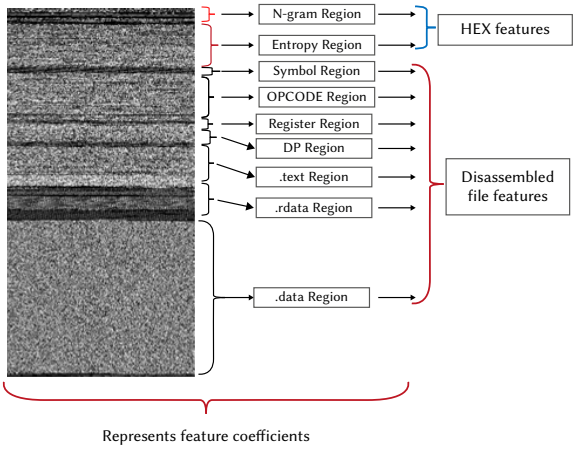


Fig. 8. Gray scale image of malicious code with feature coefficient Regions.

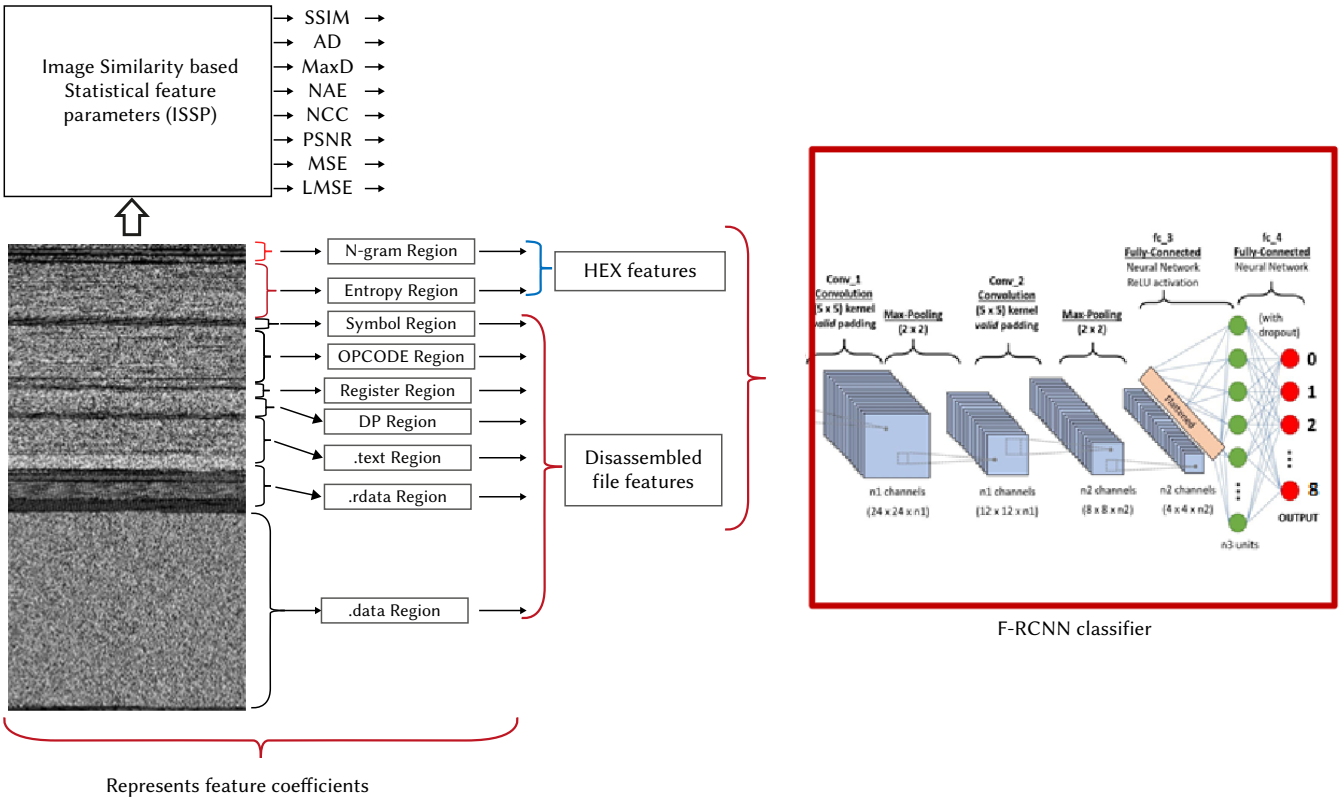


Fig. 9. F-RCNN Architecture for MDS.

Fig. 11 represents a sample of nine malware categories. The image has very fine and typical texture patterns and the same may be used to visualize the malware family. It has been observed that malware of the same family has similar signatures or fingerprints in some area of an image [187]. This information may be used as knowledge to identify the malware family. Zhang [186] represented information gains and the probability of the OPCODE to construct an image. M. Ahmadi [57] used Local Binary Pattern features and Haralick features to construct an image.

As R-CNN extracts features from each block, features from the same block will be repeatedly extracted, leading to a greater number of repetitive computations as shown in Fig. 9. **Therefore, the author proposes F-R-CNN, which is an improved version of R-CNN where CNN performs forward computation on the whole image**.

The entire image with a set of proposed regions (*K* object) with similarity based statistical parameters is input for an F-RCNN network. The network performs several convolution operations *(conv)* and maximum pooling layers on the entire input feature vector, and produces *conv* feature map. Next step it performs is to generate a feature map, by extracting a fixed length feature vector generated from the Region of Interest (RoI) pooling layer operation on the proposed objects. Each and every generated feature vector is input to a sequence of fully connected (f_c) layers. f_c layer provides two outputs. The first output is Softmax Probability Estimation (SPE) over *K* object classes plus a catch-all "background" class. The second output is 4 real valued numbers for every *K* object class. One of the *K* object classes has a set of four values which provides updated bounding box positions, calculated to reduce overlaps shown in Fig. 10.

Faster R-CNN consists of two major modules. The first module i.e., Region Proposal Network (RPN) is a deep fully convolutional network which proposes regions. The second module uses F-RCNN detector which uses the regions proposed by first module and learns about region positions from the same module due to attention-based CNN. The first module is presented in Table I. Table II presents the algorithm for generating the unified network for the overall system.

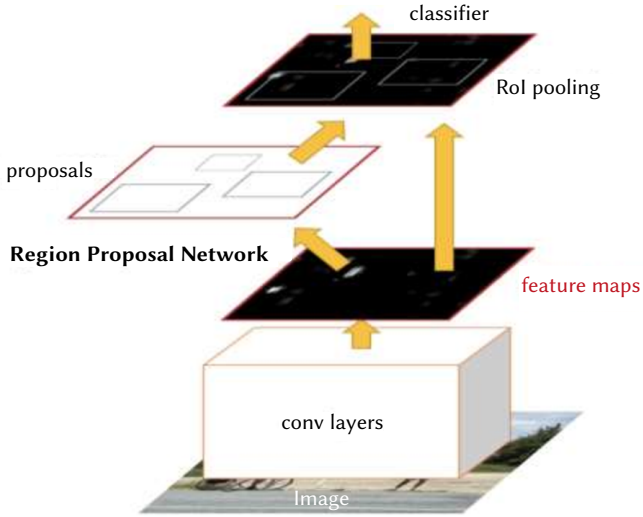


Fig. 10. F-RCNN Architecture.

VII. Statistical Parameters

To compute different statistical parameters, initially all the malware families are segregated in different folders. Nine folders are created as there are nine malware families. Malware files are processed and gray images are produced. These images are used to compute parameters. Table III presents the algorithmic steps.

TABLE I. Region Proposal Generation Algorithm

Region Proposal Network

Input: Image of any size

Output: 1. Set of rectangular object proposals
2. Each object with abjectness score

Convolutional model used: Zeiler and Fergus(ZF) (five sharable convolutional layers)

1 Generate convolutional feature map.

2 $n * n$ Sliding spatial window generated for small network. Sliding window is mapped to feature map of low dimension ($256 d for ZF$).

3 **Anchors**
- a. Each sliding window predicts multiple region proposals, simultaneously.
- b. k = Maximum region proposals for each location (k anchor boxes)
- c. Feature map size = $W * H$
- d. In the sliding window the question and an anchor, both are centered. Aspect ratio and scale is associated. Aspect ratio = 3 and scale = 3, provides $k = 9$ anchors [204].
- e. 'cls' layer is a two class Softmax layer. It estimates the probability of the object or non-object for each region proposal. Logistic regression will be used to produce of $2 * k$ scores.
- f. Regression ***(reg)*** layer has coordinates of k boxes. It will generate $4 * k$ outputs.
- g. The total number of anchors will be $W * H * k$
- h. Anchors are ***translation-invariant which reduces the model size***. As per the case of Fully Convolutional Network (FCN) [204]

4 **Multi-Scale Anchors**
- a. Compute multi-scale anchors by selecting multiple scale sliding window.
- b. Apply multi-scale sliding window to image and feature map of single scale.
- c. Generated structure may be viewed as 'pyramid of filters'

5 **Loss Function**
- a. Apply binary label to each and every bounding box.
- b. ***If*** (*Intersection over Union* (*IoU*) ∩ *Ground Truth*(*GT*) *box*) ≥ 0.7) ***then*** assign positive label
- c. If $(Intersection\ over\ Union\ (IoU) \cap GTbox) \leq 0.3$) ***then*** assign negative label ***else*** discard anchor
- d. Multiple anchors may be marked as positive by a single GT box.
- e. Compute Loss function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda * \frac{1}{N_{reg}}\sum_i P_i^* * L_{reg}(t_i, t_i^*) \quad (10)$$

where, i = anchor index in mini batch
P_i = Predicted probability of anchor being an object
P_i^* = Binary GTlabel
t_i = four parameteroised co-ordinates of Bounding Box (BB)
t_i^* = four parameteroised co-ordinates of GT Box of positive anchor
L_{cls} = log loss over object versus non-object classes
L_{reg} = regression Loss calculated only when $P_i^* = 1$ cls layer has $\{P_i\}$ and reg layer has $\{t_i\}$

- f. Compute Robust Loss

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*) \quad (11)$$

- g. Normalize L_{cls} and L_{reg}

N_{cls} = Normalisation of L_{res} by mini batchsize
N_{reg} = Normalisation of L_{reg} by the number of anchor locations

- h. *Apply weight factor λ*
- i. BB regression from an anchor box to GT box

$$t_x = \frac{(x-x_a)}{w_a}, t_y = \frac{(y-y_a)}{h_a}, t_w = \log\frac{(w)}{w_a}, t_h = \log\frac{(h)}{h_a},$$
$$t_x^* = \frac{(x^*-x_a)}{w_a}, t_y^* = \frac{(y^*-y_a)}{h_a}, t_w^* = \log\frac{(w^*)}{w_a}, t_h^* = \log\frac{(h^*)}{h_a} \quad (12)$$

where,
x, y = centre co-ordinates of box
w and h = width and height of box
x, y, w and hare variables for predicted box
x_a, y_a, w_a and h_a are variablesfor anchor box
x^*, y^*, w^* and h^* are variables for GTbox

6 **RoI pooling layer algorithm**
1. Input RoI window of size $h * w$ with $r\,(row)$ *and* $c\,(column)$ information.
2. Divide ROI window into sub windows of size $H * W$

$$window\ size_{approximate} = \frac{h}{H} * \frac{w}{W}$$

3. Generate output grid cell by maximum pooling values from each sub window
4. Apply independent pooling to each feature map channel, on a similar line to the standard maximum pooling.

7 **Optimization of Loss function**
1. From a single image mini-batch of many +/- anchors are identified.
2. Sample size of 256 anchors from an image is randomly selected to compute loss function of a mini batch.
3. Equal number of positive and negative anchors are selected i.e., 128 each, if sample size is 256.
4. In case of a low number of either of the anchors (≤ 128), then the mini batch will be padded in such a way to get equal numbers of both the anchors.
5. All new layers are randomly initialized by computing weights from zero mean Gaussian distribution with 0.01 standard deviation.
6. Shared convolutional layers are initialized by standard practice i.e. pre-training model a model for ImageNet classification.

8 **Training RPN**
RPN is trained by back propagation (BP) and stochastic gradient descent (SGD) using 'image centric' sampling strategy.

TABLE II. Approach to Unified Network of RPN and F-RCNN

Problem: Independently trained F-RCNN and RPN networks will modify their convolutional layer differently. The algorithm is required so that both can '*share*' the convolutional layer

Generating Unified system Network

Input: RPN generated regions
Output: Trained model of overall system

1 Train RPN network as per algorithm described in Table I

2
- a. Initialized the ImageNet-pre-trained model
- b. Input region proposals generated by step -1
- c. Train detection network by Fast R-CNN using.

Note: convolutional layer is yet not shared by both the training model

3 *Shared* convolutional layers are finalized.

4 Layers unique to RPN will be fine-tuned.

5 RPN training network is initialized by detector network and the system is trained.

5 Fine tune layers from ***Shared*** convolutional layers unique to F-RCNN.

6 F-RCNN network is trained.

VIII. Database Description

This section describes the publicly available datasets for malware. System performance is analyzed on the benchmark database from Kaggle (https://www.kaggle.com/c/malware-classification/data).It was a Microsoft malware classification challenge. The database contained known malware files representing a mix of 9 different families viz. Gatak, Obfuscator.ACY, Kelihos_ver1, Tracur, Simda, Vundo, Kelihos_ver3, Lollipop and Ramnit. The file structure of the database is shown in Table IV and Table V.

TABLE III. STATISTICAL PARAMETER COMPUTATION

Input Folder structure is as follows
Main folder – contains sub folders equal to number of malware families ($i = 9$ for this case)
- Sub-folders (9 malware families)
- Each sub-folder has different number of images j

R_I = *Reference image*

I_I = *Input image*

i = *Number of malware families* in main directory (folder)

j = *Number of malware variants (images) of specific malware family in a subfolder*

// Initialize empty array

Parameter Array = $\{\emptyset\}$

for $(\beta = 0 ; \beta < i ; \beta ++)$

// Load reference image – first image of malware family

$R_I = \beta[0]$

***for*// select malware families one by one**

$(k = 0 ; k < i ; k ++)$

// Get number of images present of a specific malware family

$j = size\ (k_{sub\text{-}folder})$

for $(local_{cnt} = 0; local_{cnt} < j ; local_{cnt} ++)$

// Load Input image from malware family

$I_i = (k)[local_{cnt}]$

// calculate SSIM

$SSIM(R_i, I_i) = [l(R_i, I_i)^{\alpha} . c(R_i, I_i)^{\beta} . s(R_i, I_i)^{\gamma}]$

where l = *luminance*, c = *contrast*, s = *structure*

// Calculate MSE

$$MSE(R_I, I_I) = \frac{1}{mn} \sum_{m=1}^{M} \sum_{n=1}^{N} [L(R_I(m,n)) - L(I_I(m.n))]^2$$

// calculate PSNR

$$PSNR = 10 * \log_{10} \frac{255^2}{MSE}$$

// calculate Normalized Cross-Correlation (NK)

$$NCC(R_I, I_I) = C_{R_I I_I}(\widehat{R_I}, \widehat{I_I}) = \sum_{[m,n] \in R} \widehat{R_I}(m,n)\widehat{I_I}(m,n)$$

$$\widehat{R_I} = \frac{R_I - \overline{R_I}}{\sqrt{\sum (R_I - \overline{R_I})^2}} \ , \ \widehat{I_I} = \frac{(I_I - \overline{I_I})}{\sqrt{\sum (I_I - \overline{I_I})^2}}$$

// calculate Normalized Absolute-error (NAE)

$$NAE(R_I, I_I) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} |R_I(m,n) - I_I(m.n)|}{\sum_{m=1}^{M} \sum_{n=1}^{N} [R_I(m,n)]}$$

// calculate Maximum difference

$MD(R_I, I_I) = \max\{[R_I(m,n) - I_I(m,n)]\}$

// calculate Laplacian Mean Square Error (LMSE)

$$LMSE(R_I, I_I) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} [L(R_I(m,n)) - L(I_I(m.n))]^2}{\sum_{m=1}^{M} \sum_{n=1}^{N} [L(R_I(m,n))]^2}$$

where $L(m, n)$ *is Laplacian operator*

// Store all the values in an array

end

// Take average of an array an obtain single value

Parameter_array(k)=[mean(SSIM); mean(MSE); mean(PSNR); mean(NCC); mean(NAE); mean(MaxD); mean(LMSE)]

end

end

TABLE IV. KAGGLE DATASET BASIC INFORMATION

Header	Description
ID	Twenty-character hash value for unique identification of file
Class	Integer representing family of malware
RAW data	HEX representation of the file's binary content
Metadata manifest	Log of various metadata information e.g. Function calls, Strings etc. extracted from the binary using IDA disassembler tool.
Size	0.5 Tera byte uncompressed

TABLE V. DATASET DESCRIPTION

Malware Family	Malware category	Sample Size
Gatak	Backdoor	1013
Obfuscator. ACY	obfuscated malware	1228
Kelihos_ver1	Backdoor	398
Tracur	Trojan Downloader	751
Simda	Backdoor	42
Vundo	Trojan	475
Kelihos_ver3	Backdoor	2942
Lollipop	Adware	2478
RAmnit	Worm	1541

IX. EXPERIMENTAL RESULTS AND DISCUSSION

The gray scale images of the feature vector for the malware family listed in Table V are shown in Fig. 11. It can be clearly observed that the image for each family is unique in itself. Identification becomes simpler. Feature vector 'Regions' are also clearly visible.

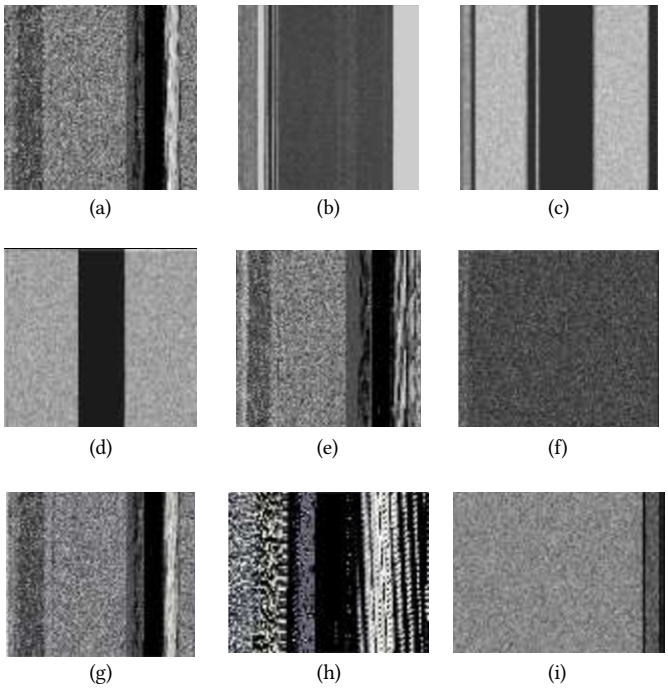
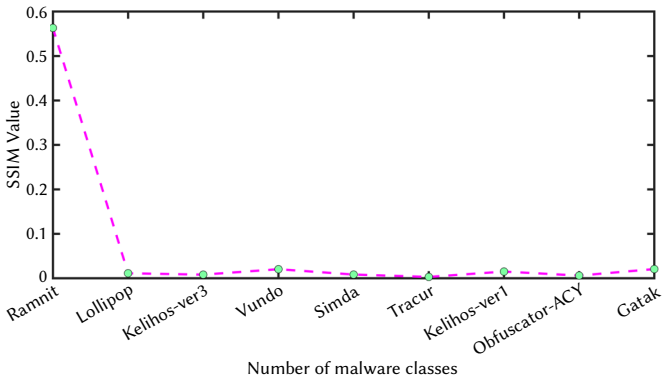


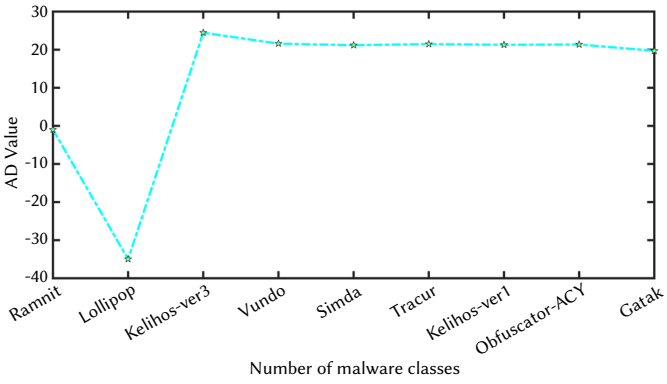
Fig. 11. Malware images of different malware families.
(a) Ramnit (b) Lollipop (c) Kelihos_ver3 (d) Vundo (e) Simda (f) Tracur (g) Kelihos_ver1 (h) Obfuscator. ACY (i) Gatak

As seen, initially one image from the first malware family is taken as a '*reference image*'. The remaining images from the first malware family and all the images of all eight malware families are used as '*Input image*'. Reference image and Input images are used to compute all the parameters. All the parameters per '*Input image*' are stored in the respective arrays (e.g.SSIM_array, MSE_array and so on). After iterating through all the images of one family, the mean value of an array is calculated. Thus, per family there is a single mean value. The mean value matrix is plotted. The same is depicted in Fig. 12.

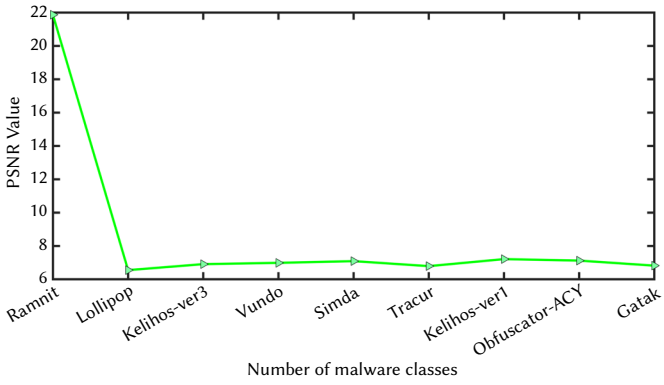
Fig. 12(a) shows the SSIM value. It is 0.56 for the Ramint malware. For the remaining families the value ranges between 0 to 0.02. Thus, there is high structural similarity with self-family, but with other families less SSIM value reflects very little similarity. On a similar line, the NAE parameter for the same family is 0.4 and for other families it is more than 0.7. Refer Fig. 12 (b) – (h). It depicts plots for PSNR, MD, MSE, LMSE, NK. It has been observed that, there is clear bifurcation in



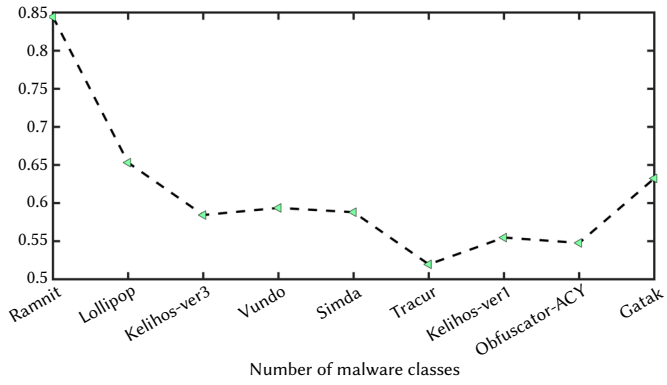
(a)



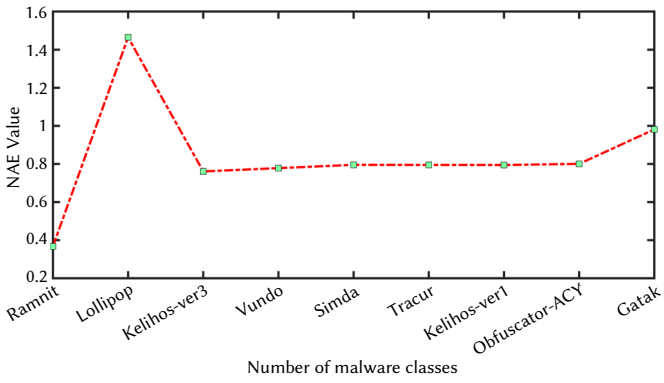
(b)



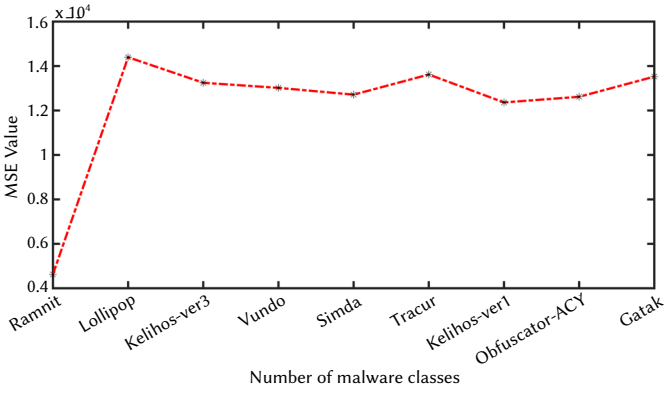
(c)



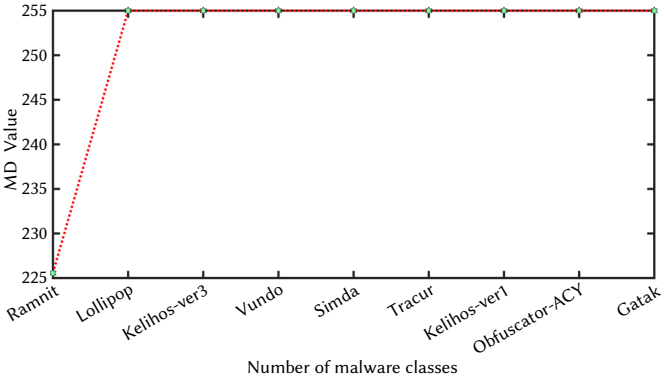
(d)



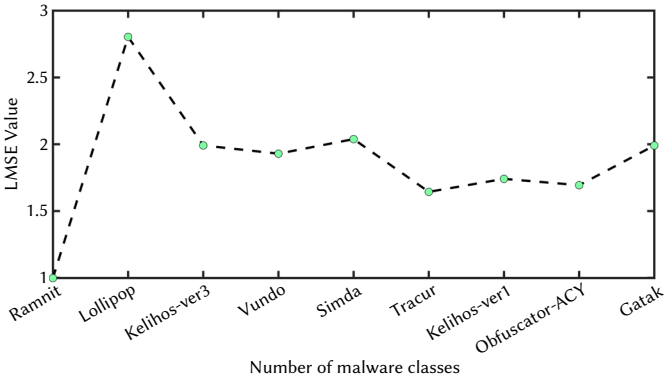
(e)



(f)



(g)



(h)

Fig. 12. Parameter plots - (a)SSIM (b) AD (c) PSNR (d) NK (NCC) (e) NAE (f) MSE (g) MD (h) LMSE.

the statistical parameter values to the same family class and a different family class. But in case of AD for the same class of family the value is approximately -1, but for the remaining families the value is either positive or negative with appropriate value difference. One can define a threshold range (0.95 to 1.1) for the AD. All these parameters can be used to train the F-RCNN classifier.

In Fig. 13 When the SC values were plotted for the same scenario then the results were not so promising. Range or proper threshold was difficult; therefore, this parameter was not taken into consideration by the author.

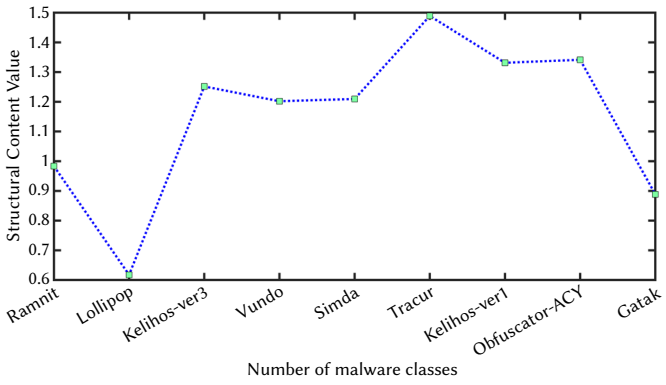


Fig. 13. SC plot.

Generated images are used to train the F-RCNN network. Annotations about regions are marked. In the experiment, the malware image set is randomly divided into a training set (60%) and a testing set (40%). This ratio of 60:40 has been selected to check the robustness of the model. The *training* data is less as compared to standard training data, i.e., 70:30. A trained library or network is created after the training process. After the training model is completed, the error and loss function of the model is used for judgment and evaluation. While training the system 30,000 epochs are selected. But while plotting the graph it is represented in percentage of the total value. Fig. 14 and Fig. 15 represent the error and loss plots in the training process. As the training iteration increases, the total error as well as loss value decreases and gradually stabilizes.

When the iterations reach 100%, the total loss value becomes flat and achieves the possible minimum value. The result shows that the training model based on F-RCNN with ISSP fusion with gray scale image is successful.

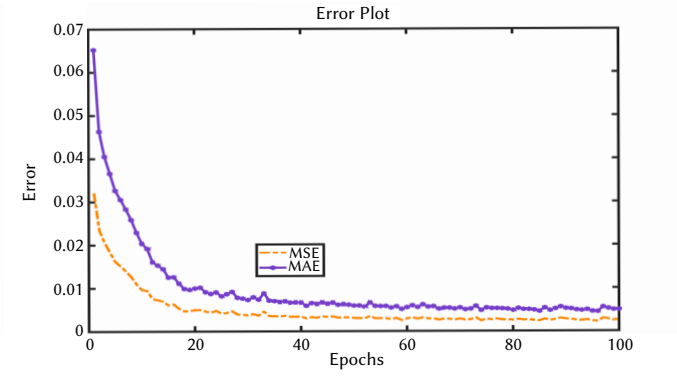


Fig. 14. MSE and MAE plot.

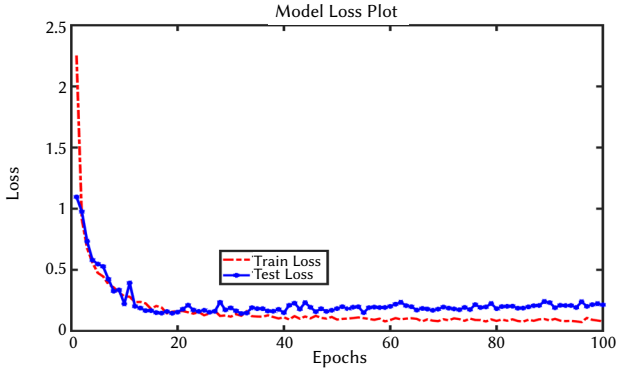


Fig. 15. Loss plot.

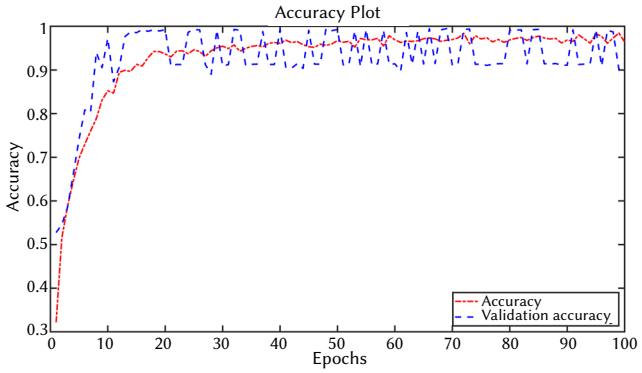


Fig. 16. Training model accuracy plot.

The network keeps a history of the trained data with error, loss and accuracy achieved while training the model. Fig. 16 represents the overall accuracy of the model with validation accuracy which approximates to 98.12%. In the testing phase, the remaining 30% of the malware files will be used. For each file a feature vector will be presented as a gray scale image will be generated and ISSP will be computed. The total matrix will be input to the trained network. The output generated from the trained network will be analyzed with the help of the statistical method where different parameters like True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) will be computed.

X. Performance Analysis

A. Performance Metrics

This section compares results obtained from the proposed work and state-of-the art methods. The proposed method opted the Kaggle benchmark dataset therefore results are compared with those research techniques that opted for the same dataset. Similarly, a comparison of the proposed algorithm for various performance metrics is stated in Table VI and Table VII respectively. Graphical plots for all comparisons are illustrated in Fig. 17.

Accuracy is the major performance parameter for the MD system, which specifies how accurately malwares are classified. Accuracy is calculated based on the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} X\ 100$$

Table VI depicts the confusion matrix of the proposed scheme for the MDS using Kaggle database.

TABLE VI. Confusion Matrix

Malware	Malware Detection %								
	Ramnit	Lollipop	Kelihos_ver3	Vundo	Simda	Tracur	Kelihos_ver1	Obfuscator.ACY	Gatak
RAmnit	98.57	0.19	0.19	0	0	0.06	0.45	0.45	0.06
Lollipop	0.36	98.82	0	0.40	0	0	0	0.40	0
Kelihos_ver3	0.03	0	99.66	0	0	0	0	0.30	0
Vundo	0	0	0	96.42	0.21	0.84	0	2.52	0
Simda	0	0	0	0	95.12	2.43	2.43	2.43	0
Tracur	0	0	0	0	0	99.73	0.13	0.13	0
Kelihos_ver1	0	0	0	0	0	0	100	0	0
Obfuscator.ACY	0.97	0.40	0.40	0.40	0.65	1.30	3.50	91.36	0.97
Gatak	0	0	0	0	0	0.09	0	0.19	99.70

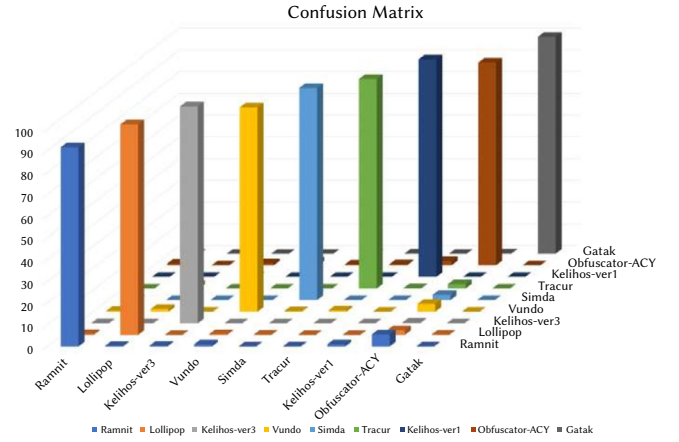


Fig. 17. Confusion Matrix Plot.

As the FRCNN classifier is not used by other researchers, the author compared the results with learning algorithms presented by researchers, as shown in Table VII.

TABLE VII. Comparative Performance of the Proposed System

Author/ Year	Dataset Used	Classifier	Accuracy (%)
Rao et al., 2017 [192]	NSL-KDD	IPDS-KNN	99.6
Shapoorifard et al., 2017 [193]	NSL-KDD	KFN-KNN	99
Vishwakarma et al., 2017 [194]	KDD cup 99	ACO-KNN	94.7
Dada et al., 2017 [195]	KDD cup 99	MIX-KNN	98.55
Ingre et al., 2017 [196]	NSL-KDD	CFS-DT	90.3
Malik et al., 2017 [197]	KDD cup 99	MULTI-DTs	91.94
Moon et al., 2017 [198]	Netflow	DT	84.7
Zhao et al., 2017 [199]	KDD cup 99	DBN-PNN	99.14
Tan et al., 2017 [200]	NETFLOW	DBN	97.6
Le et al., 2017 [201]	KDD cup 99	LSTM	97.54
Agarap et al., 2017 [202]	NETFLOW	GRU	84.15
Saxe et al., 2017 [203]	NETFLOW	CNN	92
Ding et al., 2016 [165]	Netflow	DBN	96.1
Nadeem et al., 2016 [120]	KDD cup 99	DBN	99.18
Alom et al., 2016 [159]	NSL-KDD	DBN	97.5
Krishnan et al., 2016 [180]	KDD cup 99	RNN	77.55
Kim et al., 2016 [173]	KDD cup 99	LSTM	96.93
MDFRCNN	**Kaggle dataset**	**F-RCNN**	**98.12**

XI. Conclusion

The paper proposes a state-of-the-art technique at feature extraction as well at classification level. The paper analyses different features viz. n-gram, MD1, MD2, entropy, OPCODE, Register, symbols, data define and sections of malware file for generating the feature vector. The feature vector is converted to a gray level image for visual analysis, where typical behavioral patterns can be observed for a particular malware family. Gray-scale image conversion widely opens up the scope for using state of the art image processing techniques, which have been more mature and proven.

Feature vectors have been presented in an image as different '*Regions*' which allows the use of Region Proposed Network (RPN). Exhaustive work done in the region-based analysis in an image, motivated the author to opt for the proposed methodology.

Malware codes are normally 75% to 80% identical. The image constructed from this code after extracting features should show similarity. Considering this point the author is motivated to introduce different image similarity based statistical parameters (ISSP) such as NCC, AD, MD, SSIM, LMSE, MMSE and PSNR as a feature set to improve system performance. The feature plot shown in Fig. 14, concludes that the features are distinctive. Thus, fusion of gray scale image with similarity parameters is used to train the classifier.

The development of region-based analysis with CNN as a base classifier offers R-CNN. The next modified versions of the basic R-CNN are Fast RCNN and Faster R-CNN (F-RCNN) techniques which have been proven for less training and testing time as shown in Fig. 5. This type of deep learning technique is more suitable for MDS where not only real time learning can be implemented with less time, but testing or producing output in the form of malware detection is desideratum. The system performance is analyzed using the benchmark database from Kaggle. This dataset is publicly available and results can be compared with the baseline. The database consists of nine malware families listed in Table V with details of malware families, malware categories and the number of sample files.

F-RCNN classifier with image-based visualization of the feature vector and ISSP as an additional feature resulted in better performance for classifying nine classes of malware. The proposed model offered an overall accuracy of 98.12% with improved rate of MD.

References

[1] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," Journal of Information Security, vol. 5, no. 02, pp. 56, 2014.

[2] Sahs, Justin & Khan, Latifur, "A Machine Learning Approach to Android Malware Detection" Proceedings - European Intelligence and Security Informatics Conference, EISIC 2012, pp.141-147, 2012.

[3] M. P. Deore and U.V. Kulkarni, "Malware Detection Techniques and its Classification: A Survey", International Journal of Research in Electronics

AND Computer Engineering (IJRECE), vol.6, no 4, pp.63-71, 2018.

[4] E. Bou-Harb, M. Debbabi and C. Assi, "Cyber Scanning: A Comprehensive Survey," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1496-1519, 2014.

[5] M. P. Deore, U.V. Kulkarni and B.M. Patre, "Malware Classification Using Machine Learning: A Survey", Journal of Advanced Research in Dynamical and Control Systems (JARDCS), vol.10, Issue no.10, pp.181-190, 2018.

[6] Cohen, W. W, "Learning to classify English text with ILP methods", In Advances in Inductive Logic Programming, L. De Raedt, ed. IOS Press, Amsterdam, The Netherlands, pp.124-143, 2002.

[7] M. G. Schultz, E. Eskin, F. Zadok, S. J. Stolfo, "Data mining methods for detection of new malicious executable", Security and Privacy, Proceedings. 2001 IEEE Symposium, pp. 38-49, 2001.

[8] J. Z. Kolter, M. A. Maloof, "Learning to detect and classify malicious executables in the wild", Journal Machine Learning Research. 7, pp. 21-44, 2006.

[9] R. Tian, L. M. Batten, S. C. Versteeg, "Function length as a tool for malware classification", in: Malicious and Unwanted Software, MALWARE 2008. 3rd International Conference on, pp. 69-76, 2008.

[10] Zolkipli Mohamad Fadli, Aman Jantan, "An approach for malware behavior identification and classification", *Computer Research and Development (ICCRD) 2011 3rd International Conference on*, vol. 1, 2011.

[11] Shankarapani, M., Ramamoorthy, S., Movva, R., Mukkamala, S., "Malware detection using assembly and api call sequences". J. Comput. Virol. pp. 1–13, 2010.

[12] D. Kong, G. Yan, "Discriminant malware distance learning on structural information for automated malware classification", in: ACM SIGKDD '13,nKDD '13, ACM, New York, NY, USA, pp. 1357-1365,2013.

[13] Santos I., Devesa J., Brezo F., Nieves J., Bringas P.G, "OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection", International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sesstelligent Systems and Computing. Springer, Berlin, Heidelberg, vol 189 2013b.

[14] B. Gu, Y. Fang, P. Jia, L. Liu, L. Zhang and M. Wang, "A New Static Detection Method of Malicious Document Based on Wavelet Package Analysis," *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, 2015, pp. 333-336,2015.

[15] Q. Li and X. Li, "Android Malware Detection Based on Static Analysis of Characteristic Tree," *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Xi'an, pp. 84-91, 2015.

[16] Yoo, In Seon, "Visualizing windows executable viruses using self-organizing maps", pp. 82-89, 2004.

[17] D. A. Quist and L. M. Liebrock, "Visualizing compiled executables for malware analysis", *6th International Workshop on Visualization for Cyber Security*, Atlantic City, NJ, pp. 27-32, 2009.

[18] P. Trinius, T. Holz, J. Göbel and F. C. Freiling, "Visual analysis of malware behavior using tree maps and thread graphs", *6th International Workshop on Visualization for Cyber Security*, Atlantic City, NJ, pp. 33-38, 2009.

[19] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath B, "Malware Images: Visualization and Automatic Classification", Proceedings of the 8th International Symposium on Visualization for Cyber Security, Article No. 4, 2011.

[20] K. Kancherla and S. Mukkamala, "Image visualization based malware detection", *IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, Singapore, pp. 40-44, *2013.*

[21] S. Han, H. Mao, and W. J. Dally, "Deep compression: Com-pressing deep neural network with pruning, trained quantization and Huffman coding", CoRR, abs/1510.00149, 2, 2015.

[22] M. Arefkhani and M. Soryani, "Malware clustering using image processing hashes", *9th Iranian Conference on Machine Vision and Image Processing (MVIP)*, Tehran, pp. 214-218,*2015.*

[23] Wu Q., Qin Z., Zhang J., Yin H., Yang G., Hu K, "Android Malware Detection Using Local Binary Pattern and Principal Component Analysis", In: Zou B., Li M., Wang H., Song X., Xie W., Lu Z. (eds) Data Science. ICPCSEE 2017. Communications in Computer and Information Science, vol 727. Springer, Singapore, 2017.

[24] S. Rezaei, A. Afraz, F. Rezaei, M. R. Shamani, "Malware detection using opcodes statistical features", in: 2016 8th International Symposium on Telecommunications (IST), pp. 151–155, 2016.

[25] B. Kolosnjaji, G. Eraisha, G. Webster, A. Zarras and C. Eckert, "Empowering convolutional networks for malware classification and analysis," *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, pp. 3838-3845, 2017.

[26] S. Dübel, M. Röhlig, H. Schumann and M. Trapp, "2D and 3D presentation of spatial data: A systematic review," 2014 IEEE VIS International Workshop on 3DVis (3DVis), 2014, pp. 11-18, doi: 10.1109/3DVis.2014.7160094.

[27] N. Cao and W. Cui, "Introduction to Text Visualization", Atlantis Press, Paris, 2016.

[28] D. Keim, "Information visualization and visual data mining", IEEE Transactions on Visualization and Computer Graphics, vol. 8, no. 1, pp. 1–8, 2002.

[29] S. Few, "Information Dashboard Design - The Effective Visual Communication of Data", Sebastopol, CA: O'Reilly, 2006.

[30] J. Jacobs and B. Rudis, "Data-driven security analysis, visualization, and dashboards", in Indianapolis, John Wiley & Sons, 2014.

[31] N. Cao, L. Lu, Y.-R. Lin, F. Wang, and Z. Wen, "Social Helix: visual analysis of sentiment divergence in social media", Journal of Visualization, vol.18, no. 2, pp. 221–235, 2015.

[32] T. Songqing, "Imbalanced Malware Images Classification: a CNN based Approach", arXiv:1708.08042, 2017.

[33] W. B. Balakrishnan, "Security Data Visualisation", SANS Institute Inc, 2014.

[34] N. Diakopoulos, D. Elgesem, A. Salway, A. Zhang, and K. Hofland, "Compare clouds: visualizing text corpora to compare media frames", in Proceedings of IUI Workshop on Visual Text Analytics, 2015.

[35] H. Shiravi, A. Shiravi, and A. A. Ghorbani, "A survey of visualization systems for network security", IEEE Transactions on Visualization and Computer Graphics, vol.18, no.8, pp.1313–1329, 2012.

[36] Venkatraman, Sitalakshmi and Mamoun Alazab, "Use of Data Visualisation for Zero-Day Malware Detection", *Security and Communication Networks*, 1728303:1-1728303:13.,2018.

[37] T.Y.Zhang, X.M.WangLi, Z.Z.Li, F.Guo,Y.Ma, and W.Chen, "Survey of network anomaly visualization", Scince China Information Sciences, vol. 60, no. 12, 2017.

[38] W. Shanks, "Enhancing Intrusion Analysis through Data Visualization", SANS Institute, Inc, 2015.

[39] S.Foresti, J.Agutter, Y.Livnat, S.Moon, and R.Erbacher, "Visual correlation of network alerts", IEEE Computer Graphics and Applications, vol.26, no.2, pp.48–59,2006.

[40] M. Wagner, D. Sacha, A. Rind et al.,"Visual Analytics: Foundations and Experiences in Malware Analysis," in book: Empirical Research for Software Security: Foundations and Experience, L.benOthmane, M. GiljeJaatun, and E. Weippl, Eds., CRC/Taylor and Francis, pp. 139–171, 2017.

[41] K. Han, J. H. Lim, and E. G. Im, "Malware analysis method using visualization of binary files," in Proceedings of the the2013 Research in Adaptive and Convergent Systems, Montreal, Quebec, Canada, pp. 317–321, 2013.

[42] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification", in Proceedings of the 8th International Symposium on Visualization for Cyber Security, (VizSec '11), USA, 2011.

[43] N. Nissim, R. Moskovitch, L. Rokach, Y. Elovici, "Novel active learning methods for enhanced pc malware detection in windows OS", Expert Systems with Applications, vol. 41, no. 13, pp. 5843 – 5857, 2014.

[44] S. M. Tabish, M. Z. Shafiq, M. Farooq, Malware detection using statistical analysis of byte-level file content, in: Proceedings of the ACM SIGKDD Workshop on Cyber Security and Intelligence Informatics, CSI-KDD '09, pp. 23–31, 2009.

[45] W. Wong, M. Stamp, "Hunting for metamorphic engines", Journal in Computer Virology, vol. 2, no. 3, pp. 211-229, 2006.

[46] S. Attaluri, S. McGhee, M. Stamp, "Profile hidden Markova models and metamorphic virus detection", Journal in Computer Virology, pp.151-169, 2009.

[47] M. Siddiqui, M. C. Wang, J. Lee, "Detecting internet worms using data mining techniques", Journal of Systemic, Cybernetics and Informaticsm, pp.48-53, 2009.

[48] I. Santos, J. Nieves, P. G. Bringas, "Semi-supervised Learning for Unknown Malware Detection", International, Symposium on Distributed

Computing and Artificial Intelligence, Springer Berlin Heidelberg Berlin, Heidelberg, pp. 415-422, 2011.

[49] Z. Chen, M. Roussopoulos, Z. Liang, Y. Zhang, Z. Chen, A. Delis, "Malware characteristics and threats on the internet ecosystem", Journal of Systems and Software, pp.1650-1672, 2012.

[50] J. Yonts, "Attributes of malicious files", Tech. rep., The SANS Institute, 2012.

[51] X. Hu, K. G. Shin, S. Bhatkar, K. Gri_n, Mutantx-s, " Scalable malware clustering based on static features", in: USENIX Annual Technical Conference, pp. 187-198, 2013.

[52] D. Kong, G. Yan, "Discriminant malware distance learning on structural information for automated malware classification", in: ACM SIGKDD '13, nKDD '13, ACM, New York, NY, USA, pp. 1357-1365, 2013.

[53] I. Santos, F. Brezo, X. Ugarte-Pedrero, P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection", Information Sciences 231 pp.64-82, 2013.

[54] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, M. Antonakakis, "Measuring and detecting malware downloads in live network traffic", in: Computer Security ESORICS 2013: 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 556-573, 2013.

[55] J. Bai, J. Wang, G. Zou, "A malware detection scheme based on mining format information", The Scientific World Journal, 2014.

[56] A. Tamersoy, K. Roundy, D. H. Chau, "Guilt by association: large scale malware detection by mining file-relation graphs", in: Proceedings of the 20th ACM SIGKDD, ACM, pp. 1524-1533, 2014.

[57] M. Ahmadi, G. Giacinto, D. Ulyanov, S. Semenov, M. Tromov, "Novel feature extraction, selection and fusion for effective malware family classification", arXiv:1511.04317, 2016.

[58] M. Egele, T. Scholte, E. Kirda, C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools", ACM computing surveys (CSUR), 2012.

[59] Z. Feng, S. Xiong, D. Cao, X. Deng, X. Wang, Y. Yang, X. Zhou,Y. Huang, G. Wu, "Hrs: A hybrid framework for malware detection", In Proceedings of the 2015 ACM International Workshop on Security and Privacy Analytics, ACM, pp. 19-26, 2015.

[60] M. Gharacheh, V. Derhami, S. Hashemi, S. M. H. Fard, "Proposing an hmm-based approach to detect metamorphic malware", Fuzzy and Intelligent Systems (CFIS), pp. 1-5, 2015.

[61] P. Khodamoradi, M. Fazlali, F. Mardukhi, M. Nosrati, "Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms", in: Computer Architecture and Digital Systems (CADS), 2015 18th CSI International Symposium on, IEEE, pp.1-6, 2015.

[62] Pai, S., Troia, F.D., Visaggio, C.A.. "Clustering for malware classification", *Journal Computer Virology, Hack Tech*13, pp. 95–107, 2017.

[63] J. Sexton, C. Storlie, B. Anderson, "Subroutine based detection of APT malware", Journal of Computer Virology and Hacking Techniques, pp. 1-9, 2015.

[64] T. Lee, J. J. Mody, "Behavioral classification", In EICAR Conference, pp. 1-17, 2006.

[65] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, J. Nazario, "Automated classification and analysis of internet malware", In Recent advances in intrusion detection, Springer, ,pp. 178-197, 2007.

[66] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda, Scalable, "Behavior-based malware clustering", In NDSS, vol. 9, pp. 8-11, 2009.

[67] I. Firdausi, C. Lim, A. Erwin, A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection", in: ACT '10, IEEE, pp. 201-203, 2010.

[68] Y. Park, D. Reeves, V. Mulukutla, B. Sundaravel, "Fast malware classification by automated behavioral graph matching", in: Workshop on Cyber Security and Information Intelligence Research, ACM, pp. 45, 2010.

[69] B. Anderson, D. Quist, J. Neil, C. Storlie, T. Lane, "Graph-based malware detection using dynamic analysis", Journal in Computer Virology, vol. 7, no. 4, pp. 247-258, 2011.

[70] M. Lindorfer, C. Kolbitsch, P. M. Comparetti, "Detecting environment sensitive malware", in: Recent Advances in Intrusion Detection, Springer, pp. 338-357, 2011.

[71] K. Rieck, P. Trinius, C. Willems, T. Holz, "Automatic analysis of malware behavior using machine learning", Journal of Computer Security, vol. 19, no. 4, pp. 639-668, 2011.

[72] P. M. Comar, L. Liu, S. Saha, P. N. Tan, A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection", in: INFOCOM, 2013 Proceedings IEEE, pp. 2022-2030, 2013.

[73] G. E. Dahl, J. W. Stokes, L. Deng, D. Yu, " Large-scale malware classification using random projections and neural networks", in: Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3422-3426, 2013.

[74] S. Nari, A. A. Ghorbani, "Automated malware classification based on network behavior", in: Computing, Networking and Communications (ICNC), 2013 International Conference on, IEEE, pp. 642-647, 2013.

[75] S. Palahan, D. Babi_c, S. Chaudhuri, D. Kifer, "Extraction of statistically significant malware behaviors", in: Computer Security Applications Conference, ACM, pp. 69-78, 2013.

[76] M. Kruczkowski, E. N. Szynkiewicz, "Support vector machine for malware analysis and classification", in: Web Intelligence (WI) and Intelligent Agent Technologies (IAT), IEEE Computer Society, pp. 415-420, 2014.

[77] D. Uppal, R. Sinha, V. Mehra, V. Jain, "Malware detection and classification based on extraction of api sequences", in: ICACCI, IEEE, pp. 2337-2342, 2014.

[78] A. Elhadi, M.A. Maarof, B. Barry, "Improving the Detection of Malware Behaviour Using Simplified Data Dependent API Call Graph", International Journal of Security and Its Applications, vol. 7, no. 5, pp. 29-42, 2013.

[79] M. Ghiasi, A. Sami, Z. Salehi, "Dynamic VSA: a framework for malware detection based on register contents", Engineering Applications of Artificial Intelligence, pp.111- 122, 2015.

[80] N. Kawaguchi, K. Omote, "Malware function classification using apis in initial behavior", in: Information Security (AsiaJCIS), 2015 10th Asia Joint Conference on, IEEE, pp. 138-144, 2015.

[81] C.-T. Lin, N.-J. Wang, H. Xiao, C. Eckert, "Feature selection and extraction for malware classification", Journal of Information Science and Engineering, vol. 31, no. 3, pp. 965-992, 2015.

[82] A. Mohaisen, O. Alrawi, M. Mohaisen, "Amal: High-fidelity, behavior based automated malware analysis and classification", Computers & security, vol. 52, pp. 251-266, 2015.

[83] T. Wuchner, M. Ochoa, A. Pretschner, "Robust and effective malware detection through quantitative data flow graph metrics," in: Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, pp. 98-118, 2015.

[84] G. Liang, J. Pang, C. Dai, "A behavior-based malware variant classification technique", International Journal of Information and Education Technology, vol. 6, pp.291, 2016.

[85] J. Jang, D. Brumley, S. Venkataraman, "Bitshred: feature hashing malware for scalable triage and semantic analysis", in: Computer and communications security, ACM, pp. 309-320, 2011.

[86] B. Anderson, C. Storlie, T. Lane, "Improving malware classification: bridging the static/dynamic gap", in: Proceedings of the 5th ACM workshop on Security and artificial intelligence, ACM, pp. 3-14, 2012.

[87] M. Eskandari, Z. Khorshidpour, S. Hashemi, "Hdm-analyser: a hybrid analysis approach based on data mining techniques for malware detection", Journal of Computer Virology and Hacking Techniques, vol. 9, pp. 77-93, 2013.

[88] R. Islam, R. Tian, L. M. Batten, S. Versteeg, "Classification of malware based on integrated static and dynamic features", Journal of Network and Computer Applications, pp.646-656, 2013.

[89] M. Egele, M.Woo, P. Chapman, D. Brumley, "Blanket execution: Dynamic similarity testing for program binaries and components", in: USENIX Security 14, USENIX Association, San Diego, CA, pp. 303-317, 2014.

[90] M. Graziano, D. Canali, L. Bilge, A. Lanzi, D. Balzarotti, "Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence", in: USENIX Security '15, pp. 1057-1072, 2015.

[91] M. Polino, A. Scorti, F. Maggi, S. Zanero, "Jackdaw: Towards Automatic Reverse Engineering of Large Datasets of Binaries, in: Detection of Intrusions and Malware", and Vulnerability Assessment, Lecture Notes in Computer Science, Springer International Publishing, pp. 121-143, 2015.

[92] P. Vadrevu, R. Perdisci, "MAXS: Scaling Malware Execution with Sequential Multi-Hypothesis Testing", in: ASIA CCS '16, ASIA CCS '16, ACM, New York, NY, USA, pp. 771-782, 2016.

[93] M. P. Deore and U.V. Kulkarni, "Static Way of Effective Feature Extraction

and Malware Classification", Online International Interdisciplinary Research Journal, International Conference on Recent Multidisciplinary Research (ICRMR-2018), Organized and Hosted by Foundation of Innovative Research at conference center, AIT,Thailand ,vol.8 , no 2, pp.81-93, 2018.

[94] M. Asquith, "Extremely scalable storage and clustering of malware metadata", Journal of Computer Virology and Hacking Techniques, pp. 1-10, 2015.

[95] W. Mao, Z. Cai, D. Towsley, X. Guan, "Probabilistic inference on integrity for access behavior based malware detection", in: International Workshop on Recent Advances in Intrusion Detection, Springer, pp. 155-176, 2015.

[96] F. Ahmed, H. Hameed, M. Z. Shafiq, M. Farooq, "Using spatio-temporal information in api calls with machine learning algorithms for malware detection", in: Proceedings of the 2nd ACM workshop on Security and artificial intelligence, ACM, pp. 55-62, 2009.

[97] E. Raff, C. Nicholas, "An alternative to ncd for large sequences, lempel-zivjaccard distance", in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 1007-1015, 2017.

[98] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, C. Faloutsos, "Polonium: Tera-scale graph mining for malware detection", in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 131-142, 2010.

[99] Y. Ye, T. Li, Y. Chen, Q. Jiang, "Automatic malware categorization using cluster ensemble", in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 95- 104, 2010.

[100] M. Lindorfer, C. Kolbitsch, P. M. Comparetti, "Detecting environment sensitive malware", in: Recent Advances in Intrusion Detection, Springer, pp. 338-357, 2011.

[101] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, T. Dumitras, "The dropper effect: Insights into malware distribution with downloader graph analytics", in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, pp. 1118-1129, 2015.

[102] J. Saxe, K. Berlin, "Deep neural network based malware detection using two dimensional binary program features, in: Malicious and Unwanted 47Software (MALWARE)", 2015 10th International Conference on, IEEE, pp. 11-20, 2015.

[103] K. Huang, Y. Ye, Q. Jiang, Ismcs: an intelligent instruction sequence based malware categorization system, in: Anti-counterfeiting, Security, and Identification in Communication, 2009, IEEE, pp. 509-512, 2009.

[104] Hardy, W. Chen, L.Hou, S. Ye, Y. Li X, "A deep learning framework for intelligent malware detection", In Proceedings of the International Conference Data Mining (ICDM), Barcelona, Spain, pp. 61, 2016.

[105] Wang X., Yiu S.M. "A multi-task learning model for malware classification with useful file access pattern from API call sequence", arXiv, arXiv:1610.05945, 2016.

[106] Javaid, Salman., "Analysis and Detection of Heap-based Malwares Using Introspection in a Virtualized Environment.", University of New Orleans Theses and Dissertations. 1875. 2014.

[107] Ma, T.; Wang, F.; Cheng, J.; Yu, Y.; Chen, X., "A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks.", Sensors, 1701.,2016.

[108] Aminanto, M.E., Kim, K., "Deep Learning-Based Feature Selection for Intrusion Detection System in Transport Layer", Available online:https://pdfs.semanticscholar.org/bf07/e753401b36662eee7b8cd6c65cb8cfe31562.pdf (accessed on 23 February 2019).

[109] Diro, A.A. Chilamkurti, N., "Deep learning: The frontier for distributed attack detection in Fog-to-Things computing", IEEE Communication, pp.169–175, 2018.

[110] Chawla, S., "eep Learning Based Intrusion Detection System for Internet of Things", University of Washington: Seattle, WA, USA, 2017.

[111] Cox, J.A. James, C.D. Aimone, J.B., "A signal processing approach for cyber data classification with deep neural networks", Procedia Comput. Sci., pp.61, 349–354, 2015.

[112] Wang, Z. "The Applications of Deep Learning on Traffic Identification", Black Hat: Washington, DC, USA, 2015.

[113] Lotfollahi, M.; Shirali, R.; Siavoshani, M.J.; Saberian, M. ,"Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning", arXiv:1709.02656, 2017.

[114] Mi, G.; Gao, Y.; Tan, Y., "Apply stacked auto-encoder to spam detection", In Proceedings of the International Conference in Swarm Intelligence, Beijing, China, 26–29 ,pp. 3–15,2015.

[115] Loukas, G., Vuong, T., Heartfield, R., Sakellari, G., Yoon, Y., Gan, D., "Cloud-based cyber-physical intrusion detection for vehicles using Deep Learning", pp. 3491–3508, 2018.

[116] Diro, A.A.; Chilamkurti, N. Leveraging LSTM Networks for Attack Detection in Fog-to-Things Communications", IEEE Commun. Mag. 56, pp. 124–130, 2018.

[117] Shi, C.; Liu, J.; Liu, H.; Chen, Y., "Smart user authentication through actuation of daily activities leveraging WiFi-enabled IoT " , In Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Chennai, India, ACM: New York, NY, USA, pp. 10–14, 2017.

[118] Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Auto encoder-based feature learning for cyber security applications. In Proceedings of the 2017 International Joint Conference Neural Networks (IJCNN) Anchorage, AK, USA, 14–19, pp. 3854–3861,2017.

[119] Abdulhammed, R., Faezipour, M., Abuzneid, A., AbuMallouh, A., "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic", IEEE Sens. Lett., 2018.

[120] Nadeem M., Marshall O., Singh, S., Fang, X., Yuan X., Semi-Supervised Deep Neural Network for Network Intrusion Detection", Available online: https: //digitalcommons .kennesaw . edu/ccerp/2016/Practice/2/ (accessed on 23 February 2019).

[121] Alom, M.Z. Taha, T.M., "Network intrusion detection for cyber security using unsupervised deep learning approaches", In Proceedings of the 2017 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 27–30 ,pp. 63–69, June 2017.

[122] Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: "An ensemble of auto encoders for online network intrusion detection", arXiv:1802.09089., 2018.

[123] David, O.E.; Netanyahu, N.S. "Deep sign: Deep learning for automatic malware signature generation and classification", In Proceedings of the 2015 International Joint Conference Neural Networks (IJCNN), Killarney, Ireland, pp. 1–8., 2015.

[124] Saxe, J.; Berlin, K. "Deep neural network based malware detection using two dimensional binary program features", In Proceedings of the 10th International Conference Malicious and Unwanted Software (MALWARE),Washington, DC, USA, pp. 11–20, 2015.

[125] Mizuno, S.; Hatada, M.; Mori, T.; Goto, S. "Bot Detector: A robust and scalable approach toward detecting malware-infected devices", In Proceedings of the 2017 IEEE International Conference Communications (ICC), Paris, France; pp. 1–7, 2017.

[126] S. Srakaew, W. Piyanuntcharatsr, S. Adulkasem, On the comparison of malware detection methods using data mining with two feature sets", Journal of Security and Its Applications , pp. 293-318, 2015.

[127] Grosse, K.; Papernot, N.; Manoharan, P.; Backes, M.; McDaniel, P., "Adversarial perturbations against deep neural networks for malware classification", arXiv:1606.04435, 2016.

[128] Cordonsky, I.; Rosenberg, I.; Sicard, G.; David, E.O., "Deep Origin: End-to-end deep learning for detection of new malware families", In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil; pp. 1–7,2018.

[129] Huang, W.; Stokes, J.W., "MtNet: A multi-task neural network for dynamic malware classification. In Proceedings of the International Conference Detection of Intrusions and Malware, and Vulnerability Assessment", Donostia-San Sebastián, Spain, pp. 399–418, 2016.

[130] Roy, S.S.; Mallik, A.; Gulati, R.; Obaidat, M.S.; Krishna, P.V. "A Deep Learning Based Artificial Neural Network Approach for Intrusion Detection", In Proceedings of the International Conference Mathematics and Computing, Haldia, India, pp. 44–53, 2017.

[131] Tang, T.A. Mhamdi, L. McLernon, D. Zaidi, S.A.R. Ghogho, M., "Deep learning approach for network intrusion detection in software defined networking", In Proceedings of the 2016 International Conference Wireless Networks and Mobile Communication (WINCOM), Fez, Morocco, pp. 258–263, 2016.

[132] Diro, A.A. Chilamkurti, N. Distributed attack detection scheme using deep learning approach for internet of things", Future Gener. Comput. Syst., 82, 761–768, 2018.

[133] Mi, G.; Gao, Y.; Tan, Y. "Apply stacked auto-encoder to spam detection", In Proceedings of the International Conference in Swarm Intelligence, Beijing, China; pp. 3–15, 2015.

[134] Yu, Y.; Long, J.; Cai, Z., "Network intrusion detection through stacking dilated convolutional auto encoders", Secur. Commun. Netw. , 2017.

[135] Gibert, D., "Convolutional Neural Networks for Malware Classification", Universit at Politècnica de Catalunya: Barcelona, Spain, 2016.

[136] Zeng, F.; Chang, S.;Wan, X., "Classification for DGA-Based Malicious Domain Names with Deep Learning Architectures", Int. J. Intell. Inf. Syst., 6, pp. 67–71, 2017.

[137] Yamanishi, K., "Detecting Drive-By Download Attacks from Proxy Log Information Using Convolutional Neural Network", Osaka University: Osaka, Japan, 2017.

[138] McLaughlin, N. del Rincon, J.M. Kang, B. Yerima, S. Miller, P. Sezer, S. Safaei, Y. Trickel, E. Zhao, Z. Doupe, A., " Deep android malware detection", In Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy, Scottsdale, AZ, USA, pp. 301–308, 2017.

[139] Wang, W. Zhu, M. Zeng, X. Ye, X. Sheng, Y., "Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the IEEE 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam pp. 712–717, 2017.

[140] Wang, W. Zhu, M. Wang, J. Zeng, X. Yang, Z., "End-to-end encrypted traffic classification with one-dimensional convolution neural networks", In Proceedings of the 2017 IEEE International Conference Intelligence and Security Informatics (ISI), Beijing, China, pp. 43–48., 2017.

[141] Shibahara, T. Yamanishi, K. Takata, Y. Chiba, D.Akiyama, M. Yagi, T. Ohsita, Y. Murata, M., "Malicious URL sequence detection using event de-noising convolutional neural network", In Proceedings of the 2017 IEEE International Conference Communications (ICC), Paris, France, pp. 1–7,2017.

[142] Hill, G.D. Bellekens, X.J.A., "Deep learning based cryptographic primitive classification", arXiv2017, arXiv: 1709.08385.

[143] Kolosnjaji, B. Zarras, A.Webster, G. Eckert, C., "Deep learning for classification of malware system call sequences", In Proceedings of the Australasian Joint Conf. on Artificial Intelligence, Hobart, Australia, pp. 137–149,2016.

[144] Tobiyama, S., Yamaguchi, Y. Shimada, H. Ikuse, T., Yagi, T., "Malware detection with deep neural network using process behavior", In Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, USA, Volume 2, pp. 577–5822016.

[145] Mac, H. Tran, D. Tong, V. Nguyen, L.G. Tran, H.A., "DGA Botnet Detection Using Supervised Learning Methods", In Proceedings of the 8th International Symposium on Information and Communication Technology, Nhatrang, Vietnam, pp. 211–218, 2017.

[146] Yu, B. Gray, D.L. Pan, J. de Cock, M. Nascimento, "A.C.A. Inline DGA detection with deep networks", In Proceedings of the 2017 IEEE International Conference Data Mining Workshops (ICDMW), New Orleans, LA, USA, pp. 683–692.

[147] Anderson, H.S. Woodbridge, J. Filar, B., "DeepDGA: Adversarially-tuned domain generation and detection. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, Vienna, Austria, pp. 13–21, 2016.

[148] Li, Y. Ma, R. Jiao, R., "A hybrid malicious code detection method based on deep learning. Methods 2015, 9, 205–216.

[149] Maimó, L.F. Gómez, A.L.P. Clemente, F.J.G. Pérez, M.G., "A self-adaptive deep learning-based system for anomaly detection in 5G networks", IEEE Access, 6, pp. 7700–7712, 2018.

[150] Alrawashdeh, K. Purdy C., "Toward an online anomaly intrusion detection system based on deep learning", In Proceedings of the 15th IEEE International Conference Machine Learning and Applications (ICMLA), Miami, FL, USA, pp. 195–200, 2015.

[151] Yuan, Z. Lu, Y. Wang, Z. Xue, Y., "Droid-sec: Deep learning in android malware detection", ACM SIGCOMM Comput. Commun. Rev. pp. 44, 371–372, 2014.

[152] Weber M., Schmid M., Schatz M., Geyer D., "A toolkit for detecting and analyzing malicious software", In Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, NV, USA, pp. 423–431, 2002.

[153] Hou, S. Saas, A. Ye, Y. Chen, L., "Droiddelver: An android malware detection system using deep belief network based on API call blocks", In Proceedings of the International Conference Web-Age Information Manage, Nanchang, China, pp. 54–66, 2016.

[154] Xu, L. Zhang, D. Jayasena, N. Cavazos, J. "HADM: Hybrid analysis for detection of malware", In Proceedings of the SAI Intelligent Systems Conference, London, UK, pp. 702–724, 2016.

[155] Benchea, R. Gavrilu ̧ t, "D.T. Combining restricted Boltzmann machine and one side perceptron for malware detection", In Proceedings of the International Conference on Conceptual Structures, Iasi, Romania, pp. 93–103,2014.

[156] Zhu, D. Jin, H. Yang, Y. Wu, D. Chen, W., "Deep Flow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data", In Proceedings of the 2017 IEEE Symposium Computers and Communications (ISCC), Herakli on, Greece, pp. 438–443, 2017.

[157] Ye, Y. Chen, L. Hou, S. Hardy, W. Li, X., "DeepAM: A heterogeneous deep learning framework for intelligent malware detection", Knowl. Inf. Syst., pp.265–285, 2018.

[158] Gao, N.; Gao, L.; Gao, Q.; Wang, H., "An intrusion detection model based on deep belief networks", In Proceedings of the 2014 2nd International Conference Advanced Cloud and Big Data (CBD), Huangshan, China, pp. 247–252,2014.

[159] Alom, M.Z. Bontupalli, V. Taha, T.M. Intrusion detection using deep belief networks", In Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, pp. 339–344, 2015.

[160] Dong, B.; Wang, X. "Comparison deep learning method to traditional methods using for network intrusion detection", In Proceedings of the 8th IEEE International Conference Communication Software and Networks (ICCSN), Beijing, China, pp. 581–585, 2016.

[161] Kang, M.J. Kang, J.W., "Intrusion detection system using deep neural network for in-vehicle network security", PLoS ONE, e0155781, 2016.

[162] Nguyen, K.K. Hoang, D.T. Niyato, D., "Wang, P.; Nguyen, P.; Dutkiewicz, E. Cyberattack detection in mobile cloud computing", A deep learning approach. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, pp. 1–6, 2018.

[163] Tzortzis, G.; Likas, A., "Deep Belief Networks for Spam Filtering. in Tools with Artificial Intelligence", In Proceedings of the 2007 19th IEEE International Conference on ICTAI, Patras, Greece, Volume 2, pp. 306–309, 2007.

[164] He, Y.; Mendis, G.J.; Wei, J., "Real-time detection of false data injection attacks in smart grid A deep learning-based intelligent mechanism", IEEE Trans. Smart Grid, pp. 2505–2516, 2017.

[165] Ding, Y.; Chen, S.; Xu, J., "Application of Deep Belief Networks for opcode based malware detection", In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, pp. 3901–3908, 2016.

[166] J. Upchurch, X. Zhou, "Variant: a malware similarity testing framework", in: 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, pp. 31-39, 2015.

[167] Pascanu, R.; Stokes, J.W.; Sanossian, H.; Marinescu, M.; Thomas, "A. Malware classification with recurrent networks", In Proceedings of the 2015 IEEE International Conference Acoustics, Speech and Signal Process, (ICASSP), Brisbane, Australia, pp. 1916–1920,2015.

[168] Shibahara, T.; Yagi, T.; Akiyama, M.; Chiba, D.; Yada, T. "Efficient dynamic malware analysis based on network behavior using deep learning", In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM),Washington, DC, USA, pp. 1–7, 2016.

[169] Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. ,"Predicting domain generation algorithms with long short-term memory networks", arXiv2016, arXiv:1611.00791, 2016.

[170] Lison, P.; Mavroeidis, V., "Automatic Detection of Malware-Generated Domains with Recurrent Neural Models", arXiv2017, arXiv:1709.07102, 2017.

[171] Tran, D.; Mac, H.; Tong, V.; Tran, H.A.; Nguyen, L.G., "A LSTM based framework for handling multiclass imbalance in DGA botnet detection", Neurocomputing, pp 2401–2413, 2018.

[172] Torres, P.; Catania, C.; Garcia, S.; Garino, C.G., "An Analysis of Recurrent Neural Networks for Botnet Detection Behavior", In Proceedings of the 2016 IEEE Biennial Congress of Argentina (ARGENCON), Buenos Aires,

Argentina, pp. 1–6, 2016.
[173] Kim, J. Kim, J.; Thu, H.L.T.; Kim, H. "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection", In Proceedings of the 2016 International Conference Platform Technology and Service (PlatCon), Jeju, Korea, pp. 1–5, 2016.
[174] Kim, J.; Kim, H. "Applying recurrent neural network to intrusion detection with hessian free optimization", In Proceedings of the International Conference on Information Security Applications, Jeju Island, Korea, pp. 357–369,2015.
[175] Kim, G.; Yi, H.; Lee, J.; Paek, Y.; Yoon, S., "LSTM-Based System-Call Language Modeling and Robust Ensemble Method for Designing Host-Based Intrusion Detection Systems", arXiv2016, arXiv:1611.01726, 2016.
[176] Loukas, G. Vuong, T. Heartfield, R.; Sakellari, G. Yoon, Y. Gan, D. "Cloud-based cyber-physical intrusion detection for vehicles using Deep Learning", IEEE Access, vol. 6, pp. 3491–3508, 2018.
[177] Cheng, M. Xu, Q. Lv, J. Liu,W. Li, Q.Wang, J., "MS-LSTM: A multi-scale LSTM model for BGP anomaly detection", In Proceedings of the IEEE 24th International Conference Network Protocols (ICNP), Singapore, pp. 1–6,2016.
[178] Kobojek, P.; Saeed, K., "Application of recurrent neural networks for user verification based on keystroke dynamic", J. Telecommun. Inf. Technol., pp.80–90., 2016.
[179] McDermott, C.D. Majdani, F. Petrovski, A., "Botnet detection in the internet of things using deep learning approaches", In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, pp. 1–8,2018.
[180] Krishnan, R.B.; Raajan, N.R., "An intellectual intrusion detection system model for attacks classification using RNN" Int. J. Pharm. Technol. Pp. 23157–23164, 2016.
[181] Staudemeyer, R.C., "Applying long short-term memory recurrent neural networks to intrusion detection", S. Afr. Comput. J., pp. 136–154, 2015.
[182] R. Girshick., "Fast R-CNN", arXiv:1504.08083, 2015.
[183] D. Baysa, R. Low, and M. Stamp., "Structural entropy and metamorphic malware", Journal of Computer Virology and Hacking Techniques, vol. 9, no. 4, pp. 179–192, 2013.
[184] R. Lyda and J. Hamrock., "Using entropy analysis to find encrypted and packed malware", IEEE Security and Privacy, vol. 5, no. 2, pp. 40–45,2007.
[185] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. "Malware images: Visualization and automatic classification", In Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11, pages 4:1–4:7, New York, NY,USA, ACM, 2011.
[186] Zhang, Jixin & Qin, Zheng & Yin, Hui & Ou, Lu & Hu, Yupeng., "IRMD: Malware Variant Detection Using Opcode ImageRecognition",pp.1175-1180, 2016.
[187] Mahotasfeatures,http://mahotas.readthedocs.org/en/latest/features.html, 2015.
[188] A. Moser, C. Kruegel, and E. Kirda., "Limits of static analysis for malware detection", In Computer Security Applications Conference, 2007. ACSAC 2007.Twenty-Third Annual, pp. 421–430, 2007.
[189] D. Bilar., "Statistical structures: Finger printing malware for classification and analysis", InBlackhat, 2006.
[190] B. Biggio, I. Corona, D. Maiorca, B. Nelson,N. Aˇarndi ̈ Aˇ G, P. Laskov, G. Giacinto, and F. Roli., "Evasion attacks against machine learning at test time", In H. Blockeel, K. Kersting, S. Nijssen, andF. A¡elezn A¡, editors, Machine Learning and Knowledge Discovery in Databases, volume 8190 of Lecture Notes in Computer Science, pages 387–402, Springer Berlin Heidelberg, 2013.
[191] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant. "Semantics-aware malware detection", In Security and Privacy, 2005 IEEE Symposium on, pp. 32–46, 2005.
[192] B. B. Rao and K. Swathi, "Fast kNN classifiers for network intrusion detection system", Indian J. Sci. Technol., vol. 10, no. 14, pp. 1-10, 2017.
[193] H. Shapoorifard and P. Shamsinejad, "Intrusion detection using a novel hybrid method incorporating an improved KNN", Int. J. Comput. Appl., vol. 173, no. 1, pp. 5-9, 2017.
[194] S.Vishwakarma,V. Sharma, and A. Tiwari, "An intrusion detection system using KNN-ACO algorithm", Int. J. Comput. Appl., vol. 171, no. 10, pp. 18-23, 2017.
[195] E. G. Dada, "A hybridized SVM-kNN-pdAPSO approach to intrusion detection system", in Proc. Fac. Seminar Ser., pp. 14-21,2017.
[196] B. Ingre, A. Yadav, and A. K. Soni, ``Decision tree based intrusion detection system for NSL-KDD dataset,'' in Proc. Int. Conf. Inf. Commun. Technol. Intell. Syst., pp. 207-218,2017.
[197] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection", Clust. Comput., vol. 2, no. 3, pp. 1-14, Jul. 2017.
[198] D. Moon, H. Im, I. Kim, and J. H. Park, "DTB-IDS: An intrusion detection system based on decision tree using behavior analysis for preventing APT attacks", J. Supercomput., vol. 73, no. 7, pp. 2881-2895, 2017.
[199] G. Zhao, C. Zhang, and L. Zheng, "Intrusion detection using deep belief network and probabilistic neural network", in Proc. IEEE Int. Conf. Comput. Sci. Eng., vol. 1, pp. 639-642,2017.
[200] Q. Tan, W. Huang, and Q. Li, "An intrusion detection method based on DBN in ad hoc networks", in Proc. Int. Conf. Wireless Commun. SensorNetw., pp.477-485, 2016.
[201] T.T.H. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization", pp. 1-6,2017.
[202] A. F. Agarap., "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data.'' [Online], Available: https://arxiv.org/abs/1709.03082, 2017.
[203] J. Saxe and K. Berlin., "eXpose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys., [Online]. Available: https://arxiv.org/abs/1702.08568, 2017.
[204] Ren, Shaoqing, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, pp.1137-114, 2015.

Mahendra Deore

M. Deore is working as an Asst. Professor in Computer Engineering Department at MKSSS's Cummins College of Engineering for Women, Pune 411051, India. He was awarded his Master of Technology Degree from Bharati Vidyapeeth Deemed University College of Engineering, Dhankawadi, Pune. He is currently pursuing Ph.D. Degree in Computer Science & Engineering from SGGS Institute of Engineering and Technology, Nanded under Swami Ramanand Teertha Marathwada University, Nanded, India. His areas of interest are big data, Security, Computer Networks and Machine learning. He has Fourteen years' experience in teaching.

Uday Kulkarni

U. Kulkarni is working as a Professor, Head in the Department of Computer Science and Engineering at SGGS Institute of Engineering and Technology, Nanded, India. He received doctoral degree from Swami Ramanand Teertha Marathwada University, Nanded, India in 2002. He is a recipient of a national level gold medal in the Computer Engineering Division for his research paper "Fuzzy Hyper sphere Neural Network Classifier" published in the journal of Institution of Engineers in 2004. He has published more than forty research papers in the field of Neural Networks, Fuzzy Logic and hybrid computing systems in the reputed journals and conferences.