Journal of Engineering Sciences

Vol 11, Issue 6, JUNE / 2020
ISSN NO: 0377-9254

A PROJECT ON IMAGE-BASED MALWARE CLASSIFICATION USING ENSEMBLE OF CNN ARCHITECTURES (IMCEC)

[1]Vijay Gopal Anantatmakula, [2]Kanchoju Ramalingachary

DEPT OF CSE

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE

Abstract: Unfortunately, both researchers and malware authors have demonstrated that malware scanners are limited and can be easily evaded by simple obfuscation techniques. This paper proposes a novel ensemble convolutional neural networks (CNNs) based architecture for effective detection of both packed and unpacked malware. We have named this method imagebased malware classification using ensemble of CNNs (IMCEC). Our main assumption is that based on their deeper architectures different CNNs provide different semantic representations of the image; therefore, a set of CNN architectures makes it possible to extract features with higher qualities than traditional methods. Experimental results show that IMCEC is particularly suitable for malware detection. It can achieve a high detection accuracy with low false alarm rates using malware raw-input. Result demonstrates more than 99% accuracy for unpacked malware and over 98% accuracy for packed malware. IMCEC is flexible, practical and efficient as it takes only 1.18 second on average to identify new malware sample.

Keyword: Malware; cybersecurity; Detection; Deep Learning; Transfer Learning; Fine-tuning; SVMs; Softmax; Ensemble of CNNs.

I. INTRODUCTION

Malware is an intentionally malicious software developed to cause harm to computer systems [1], [2]. Recently, a significant increase in the number of software used for illegal and malicious goals has been recorded [3], [4]. For example, Kaspersky Lab detected 69,277,289 types of malware (scripts, executable files, etc.) in 2016 [5]. McAfee Labs have reported an increase of 22% in malware attacks in the four years prior to 2017 up to 670 million [6]. The rising trend in malware attacks demands a highly effective approach to detect malwares. Most of the commercial antivirus applications typically use signature-based techniques, which require local signature databases for storing patterns that experts have detected in malicious software [7]. This tactic suffers from significant limitations since malware authors reuse the code to generate new malwares and apply code obfuscation techniques such as packing alter signatures. Because of this, high numbers of malware can remain undetected by detection methods based on signature. Other limitations of these techniques include reverse engineering and considerable domain expertise. In recent years, static and dynamic analysis has been tried for malware detection: static analysis focuses on statistical features (e.g., N-grams, API Calls, Opcode's Sequences, etc.) [8]–[10]; dynamic analysis uses virtual environment (e.g., Sandbox) to analyze the behavior of malicious applications [11], [12]. Various deep and machine learning techniques are currently used to detect and classify malwares [13]–[17]. Most of these techniques rely on feature engineering work or domain knowledge to build a feature database. Since new malware are is constantly created and updated with some specific changes, it becomes increasingly challenging to manually update feature databases against newly generated malware samples. To reduce feature engineering cost and domain expert knowledge, researchers have used visualization approaches to solve malware family classification problems. For example, Conti et al. [18] have suggested a method for the visualization of malware binary into a grayscale image and argued that visual analyses of malware binary help differentiate various regions of data in the image. Kancherla et al. [19] extracted lowlevel features like intensity-based and

Vol 11, Issue 6, JUNE /2020
ISSN NO:0377-9254

texture features from malware binary grayscale images and used support vector machine (SVM) as classifier. They obtained 95% accuracy on a dataset comprising 25,000 malware and 12,000 benign samples. The structure of deep learning (DL) methods for malware analysis and classification, however, is very shallow compared to benchmark CNNs for ImageNet containing high number of layers. Furthermore, training deep CNNs with a small labelled dataset is not easy and ImageNet CNN models are trained with ten million annotated images in ImageNet dataset. Some have tried to solve this problem via transfer learning techniques by using pre-trained ImageNet CNN models as feature extractors in small datasets in different domains [3], Based on this, an algorithm for malware classification called image-based malware classification using ensemble of CNNs (IMCEC) has been developed. It uses static features and combines binary visualization and an ensemble of CNNs, which have been previously trained using an extensive set of natural images ($\geq$ 10 million) to classify malware samples into their related families (Fig. 1). Ensemble learning is a machine learning technique in which higher predictive performance is attained by combining results obtained from several classification models, thus resulting in a stronger classifier. Unlike conventional machine learning techniques, which use training data to learn one hypothesis, the method proposed here addresses these challenges employing CNNs on multiclass classification problems using pre-trained CNNs and fine-tuning them for malware images. The presence of several CNNs in our IMCEC makes it possible to extract rich features from visualized malware binaries at a distinctive level. We were able to detect subtle differences among malware families and successfully classify them as distinct. The proposed technique made it possible to obtain more specific generic features learned from natural images for different malware images. This paper describes a novel IMCEC method, which consists of transferring knowledge from different wellknown CNN architectures already trained with ImageNet data and adapting and fine-tuning them to malware images. The features obtained were utilized for training various multiclass classifiers were trained using the transferred features and posterior probabilities were fused to increase the accuracy of the classification of families of unknown malware samples. Finally, based on the Malimg malware benchmark dataset (Table 1), tests for malware family classification were conducted on the IMCEC. The IMCEC classification accuracy reached 99.50%. In addition, IMCEC was able to easily block different obfuscation attacks with an acceptable accuracy of 98.11% for packed samples and 97.59% for salted samples.

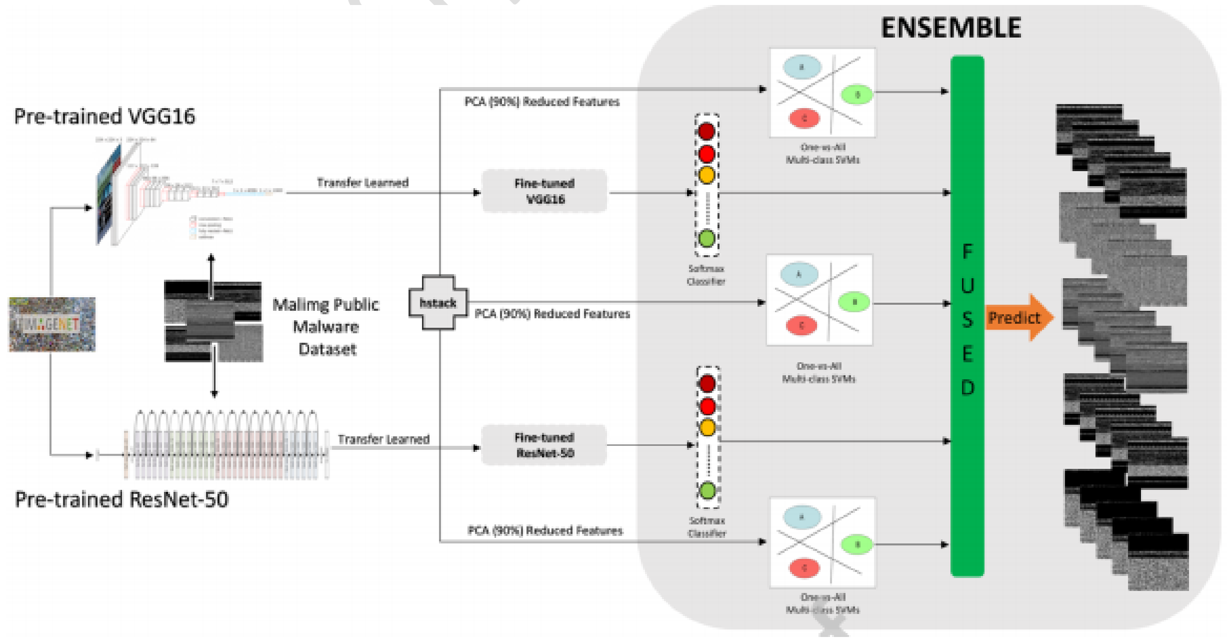

Fig. 1 – Overview of IMCEC

II. Current Methods of Malware Detection

Here, we review the current methods of malware detection, including static, dynamic, and visualization analyses based on deep learning and machine learning methods.

A. Static Analysis:

In author performed a static analysis and developed the first data mining technique for detecting malicious codes using three different static features of malware binaries: string sequences, byte sequences and portable executable (PE) heads. They later employed a rule-based method called Ripper for DLL and naive Bayesian method as learning algorithm to obtain useful features of byte sequence. They achieved a level of classification accuracy up to 97.11% after feeding malicious codes as input data. The author tested Opcodes with different n-grams sizes and classifiers. They claimed that the performance of 2-gram Opcode method was better than that of byte n-grams methods. The techniques proposed in were required domain expert knowledge and feature engineering tools, however they failed to achieve the satisfactory level of accuracy. In developed a technique to distinguish benign ware and malware based on neural network technology. They extracted and used four types of features to calculate DLL import, entropy histogram from binary data and metadata of execution files. These features were transformed into 256 dimensions of vector. They found that TRP result was 95.2% while FPR was 0.1%, however, their approach was required a feature engineering tools. The developed technique does not give enough information about the verity of features of the malware and benign samples, as well as, their achieved accuracy was not enough. Using n-grams technique extracted bytes from windows binaries and trained several classifiers based on one-vs-rest multi-class classification approach. They then combined the predictions of individual classifiers and obtained a true- and false-positive rates of 0.98 and 0.05, respectively, however, their technique did not consider the overhead time.

B. Dynamic Analysis

Dynamic analyses generally include behavior-based analysis techniques and API call monitoring .In developed an automatic malware binary cluster method based on malware behavior. The author recorded malware samples using two types of security tools such as Amun and Honey- then used two virtual platforms to execute the collected malwares and analyzed their behaviors. In used hidden Markov model for malware classification relying on system calls as observed symbols. The author in analyzed network flow activity to develop a malware identification system. They applied sequence alignment and flow feature clustering for the comparison of character sequences to extract their similarities.

The main limitations of dynamic analysis, especially the sandbox-based solutions, are that some malware can detect and change behavior when running in virtual environments. Therefore, dynamic analysis might not always uncover malicious behavior.

Dynamic analysis is useful for malware binary transformation detection, while the use of the virtual environment is very time intensive. Hence, dynamic analysis does not full feel the need of most of the practice application.

C. Visualization Analysis

Some visualization also offers tools for the detection and classification of malwares. The author in were able to detect and visualize virus by employing self-organizing map. Using image processing technique, Nataraj et al. visualized malware binaries into grayscale. Through a machine learning approach such as GIST they extracted features from malware grayscale images and used k-nearest neighbor (KNN) as a classifier. Accuracy reached 97.18% on a dataset containing 9,458 malware samples related to 25 different malware families. In subsequent work, they compared their approach based on image processing technique with dynamic analysis and achieved almost the same accuracy using dynamic malware analysis. They claimed that their modified technique could address both unpacked and packed malware samples.

Vol 11, Issue 6, JUNE / 2020
ISSN NO:0377-9254

It assumed that a set of texture features could enough to present entire content of malware images.

In visualized malware and benign binaries into grayscale images. They trained a model with DL techniques and achieved 95.66% test accuracy for a dataset with 10,000 benign and 2,000 malware samples. The proposed system does not give enough information on the structure and characteristics of the malware, as well as, did not consider the overhead time.

In focused on detecting malware in IoT environments. They created one-channel grey-scale images from executable binaries then employed light-weight DL techniques to classify them into their related families. They achieved classification accuracy of 94.0% for DDoS malware and goodware, and 81.8% for two leading malware families and goodware. However, their proposed network structure was very shallow, and the number of samples were limited from 2 malware family.

Fine-tuning is a modified transfer learning technique consisting of updating pretrained CNN weights by backpropagation. Fine-tuning is helpful for adapting pretrained CNNs to different datasets. When applied to medical imaging data, fine-tuning was as effective as training a CNN from scratch but it was more robust in relation to the size of the data. Fine-tuning has been applied to different classification tasks including image retrieval gender and age classification, plants classification text classification [58], object detection re-identification and fine art classification .

Different from previous research work, this work has advanced further from a previous work that proposed an ensemble-based deep learning framework for malware detection. We proposed a novel and unified method for multi-class malware classification. Our method utilized ensemble of fine-tuned CNNs previously trained with ImageNet dataset (≥10 million) and used various multiclass classifiers to classify the families of unknown malware samples. Our experimental results proved that IMCEC was highly efficient in the classification of malware families, even in small sample sizes, and was capable of blocking different obfuscation attacks. Our experimental considered both overall accuracy and run- time overhead.

III. METHODOLOGY

Overview of the methods used

Our IMCEC method is depicted in Fig. 1. We used CNN architectures previously trained (initialized) using natural images. Each CNN was then applied in one of the following two ways: 1) as classifiers for the generation of soft max probabilities based on fine-tuned, or 2) as image feature extractors with independent feature vectors concatenated and applied for training multiclass SVMs. Posterior probabilities from soft max classifiers and SVM ensembles were then combined to obtain the families of unknown malware samples.

We employed the following CNN models, each having various capabilities:

Fig. 2 shows VGG16 network architecture. This well-known CNN has a standard neural network architecture of connected layers and contains 16 layers needing to be trained, 5 convolutional layer blocks and 3 fully-connected layers. The filter size used in the convolutional layers was 3 x 3 kernel with 1 padding and 1 stride to guarantee the same spatial dimension for each activation map as the previous layer. To accelerate training, rectified linear unit

(ReLU) nonlinearity was applied to each convolutional layer, and a max-pooling step was used at each block end for decreasing spatial dimensions. A 2 x 2 kernel filter with no padding and 2 strides was used in max-pooling layers to guarantee that the spatial dimensions of activation map were halved compared to previous layer. Then, two fully-connected layers containing 4096 ReLU activated units were employed before the last 1000 fully-connected soft max layer.

Residual networks (ResNet-50) are deep convolutional networks and the fundamental idea behind these networks is to skip convolutional layer blocks with shortcut connections. The basic blocks, called bottleneck blocks, obey two design rules: (i) for similar sizes of output feature maps, the same filter numbers were applied to layers, and (ii) if

Vol 11, Issue 6, JUNE /2020
ISSN NO:0377-9254

feature map sizes were halved, filter number was doubled. Down-sampling was directly conducted by convolutional layers with 2 strides and batch normalization was carried out immediately after each convolution and before the activation of ReLU. When output and input had similar dimensions, identity shortcut was applied. By the increase of dimensions, projection shortcut was applied for dimension matching via 1 × 1 convolutions. In both cases, when shortcuts passed through feature maps with two different sizes, they were performed with 2 strides. Networks ended with 1,000 fully-connected layers with soft max activation. The total weighted layer number was 50, containing 23,534,592 trainable parameters. The original ResNet-50 architecture is shown in Fig. 3.

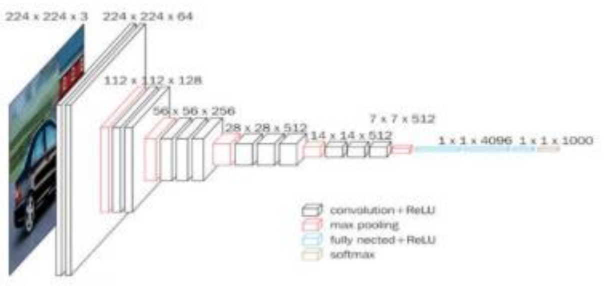

Fig. 2 – VGG16 Network Architecture

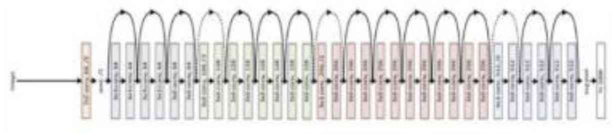

Fig. 3 – ResNet-50 Network Architecture

We selected these models because they are well understood and have high performance in different scenarios of malware image classification [3].

In our ensemble, we have applied two different types of classifiers:

Softmax is a generalized logistic function emphasizing the essential values of vectors while blocking those with values lower than maximum. Softmax function can be applied as a nonlinear version of multinomial logistic regression to a D-dimensional feature vector giving a D probability values vector in which the dth element is the probability by which the vector may represent a dth class member . Several CNN architectures employ Softmax function as classification layer.

Multiclass SVMs [68]: Support vector machines (SVM) called support vector networks are generally employed in machine learning and are of great importance in deep learning processes. SVM is a machine learning classifier capable of classifying the data that have been previously obtained, which relies on assigning specific scores to each data. Data classification is among the most critical aspects of deep learning. Generally, the resulting data are N-dimensional vectors and for this reason, it is called vector machine. We trained one-vs-all SVMs with extracted knowledge from trained CNNs.

Transfer Learning

Transfer learning consists of transferring the parameters of a neural network trained with a single dataset and task to another problem with a different dataset and task . Many deep neural networks trained on natural images share an interesting phenomenon in common: on the first layers, they learn general features, that is, features that do not appear to be specific to a particular dataset or task, but are applicable to many datasets and tasks. When the target dataset is significantly smaller than the base dataset, transfer learning can be a powerful tool to enable the training of a large target network without overfitting.

Next, we used VGG16 and ResNet-50 as base models, which were pre-trained for detecting objects on ImageNet dataset. We employed the convolutional layers of VGG16 and ResNet-50 to obtain malware image bottleneck features, which then served as input in training SVM classifiers.

D. CNN Fine-Tuning

CNN architectures previously trained on ImageNet a dataset consisting of 1,000 classes, were customized to our problem by using a fully-connected layer containing 25 classes (25 malware families) instead of a final fully connected layer (intended for 1,000 classes). Initial weights of pretrained CNNs of natural images were used then fine-tuned techniques was applied to optimize via back-propagation.

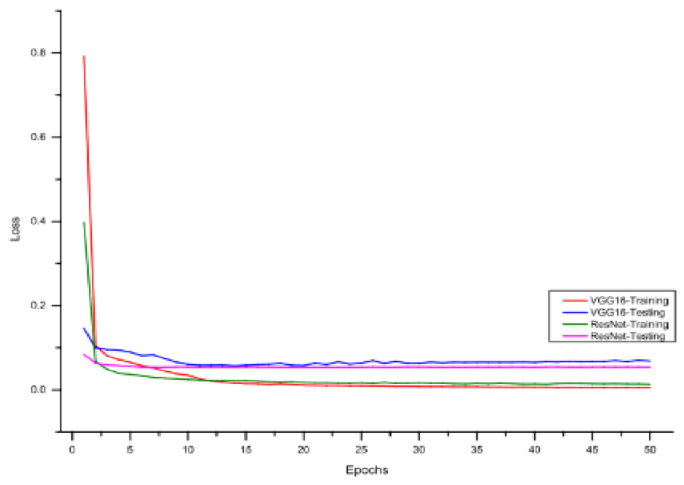

Fig. 4 – Training and testing loss for both fine-tuned CNNs

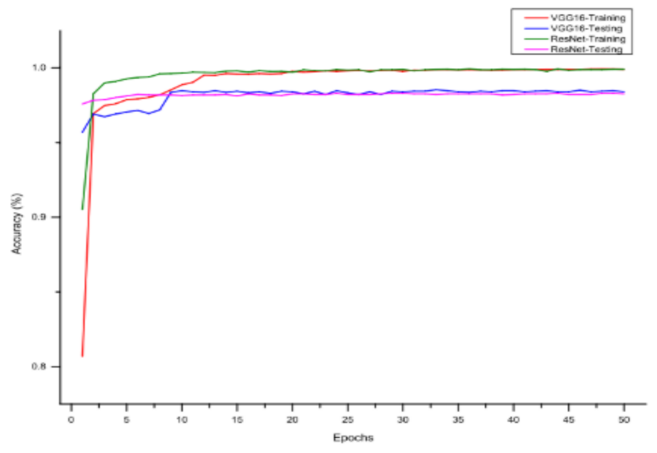

Fig. 5 – Training and testing accuracy for both fine-tuned CNNs

Assuming that X is a training dataset with n malware images, fine-tuning is an iterative procedure optimizing w filter weights and reducing error rate, according to the following:

$$L(w,X)=\frac{1}{n}\sum_{i=1}^{n} l(f(x_i,w),\hat{c_i})$$

(1)

where f (xi, w) is CNN function predicting the class ci of xi by assuming w, ci ˆ is the true class of the I th image, xi is the I th image of X, and l(ci , ci ˆ) is a penalty function for the prediction of ci instead of ci ˆ; l is a logistic loss function.

All models needed memory for storing filter weights w and therefore, batch size b depended on the memory capacity of the training hardware. Batch size was set at b=32 and learning rate was η = 5 x 10-6, which is the uniform learning rate that makes it possible for fine-tuning procedure to efficiently learn filter weights in different CNN models. We obtained this value experimentally through monitoring validation errors during fine-tuning via different learning rates. The learning rate was constantly modified until the optimum value was achieved. Generally, higher learning rates resulted in overfitting; however, lower rates limited error variations across epochs (i.e., slow learning).

IMCEC Design

Our proposed IMCEC design contains the following classifiers.

1) Fine-tuned VGG16 employing a Softmax classifier.

2) Fine-tuned ResNet50 employing a Softmax classifier.

3) A one-vs-all multiclass SVM, which was trained based on the features obtained from pre-trained

VGG16 network. We extracted 4,096 features from the fully connected (FC2) layer of the pre-trained network. We then applied principal component analysis (PCA) technique to decrease the dimensionality of the obtained features for efficient classifier training. Our feature vectors were the components explaining 90% of data variation (dimensionality from 4,096 to 410).

4) A one-vs-all multiclass SVM which was trained based on the features obtained from the pre-trained ResNet50 network. We extracted 2,048 features from the last flatten (avg_pool) layer of the pre-trained network. We then applied PCA technique to decrease the dimensionality of the obtained features for efficient classifier training. We used PCA so that the proposed feature vectors comprised principal components explaining 90% of data variation (dimensionality from 2,048 to 205).

IV. Conclusion and future work

Modern antimalware solutions rely on ML techniques to protect digital assets from malware. While ML-based methods, have demonstrated a good job at detecting new malware, but they also come with substantial development costs. Creating large set of useful features for the ML techniques takes significant amounts of time and expertise from both of malware analysts and data scientists. Deep learning architectures, in particular convolutional neural networks (CNNs), have demonstrated a great job in detecting malware simply by looking at the raw bytes such as Windows Portable Executable (PE) files.

We developed and tested a novel image-based Malware classification using an ensemble of CNN architectures (IMCEC) with optimized features capable of learning rich features. Our IMCEC classified the

majority of malware samples correctly even under obfuscation attacks and overall performed better than existing methods employing similar benchmark. Our experiments have demonstrated great levels of accuracy that are competitive with traditional ML-based solutions, while avoid the manual feature engineering phase. Our IMCEC is flexible, practical and efficient as it takes only 1.18 second on average to identify new malware sample. In the future, we plan to evaluate IMCEC against larger datasets and implementing a prototype of the proposed approach in a real-world environment for evaluation and refinement, future research also would focus on time reduction.

Vol 11, Issue 6, JUNE /2020
ISSN NO:0377-9254

Reference

[1] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," J. Comput. Secur., 2011, doi: 10.3233/JCS-2010-0410.

[2] M. Alazab, "Profiling and classifying the behavior of malicious codes," J. Syst. Softw., 2015, doi: 10.1016/j.jss.2014.10.031.

[3] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, and P. de Geus, "Malicious Software Classification Using VGG16 Deep Neural Network's Bottleneck Features," in Advances in Intelligent Systems and Computing, 2018, doi:10.1007/978-3-319-77028-4_9.

[4] F. Farivar, M. S. Haghighi, and S. Member, "Artificial Intelligence for Detection , Estimation , and Compensation of Malicious Attacks in Nonlinear Cyber Physical Systems and Industrial IoT," IEEE Trans. Ind. Informatics, vol. PP, no. c, p. 1, 2019, doi: 10.1109/TII.2019.2956474.

[5] Kaspersky Lab, "What is a Keylogger? | Definition | Kaspersky Lab US," Kaspersky Lab, 2016. .

[6] C. Beek et al., "McAfee Labs Threats Report: June 2017," McAfee Labs Rep., 2017.

[7] B. Jung, S. Il Bae, C. Choi, and E. G. Im, "Packer identification method based on byte sequences," Concurr. Comput. Pract. Exp., doi: 10.1002/cpe.5082.

[8] S. YusirwanS, Y. Prayudi, and I. Riadi, "Implementation of Malware Analysis using Static and Dynamic Analysis Method," Int. J. Comput. Appl., 2015, doi: 10.5120/20557- 2943.

[9] P. V. Shijo and A. Salim, "Integrated static and dynamic analysis for malware detection," in Procedia Computer Science, 2015, doi: 10.1016/j.procs.2015.02.149.

[10] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: Android malware characterization and detection using deep learning," Tsinghua Sci. Technol., 2016, doi: 10.1109/TST.2016.7399288.

[11] M. Lindorfer, M. Neugschwandtner, and C. Platzer, "MARVIN: Efficient and Comprehensive Mobile App Classification through Static and Dynamic Analysis," in Proceedings - International Computer Software and Applications Conference, 2015, doi:10.1109/COMPSAC.2015.103.

[12] A. Damodaran, F. Di Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," J. Comput. Virol. Hacking Tech., 2017, doi: 10.1007/s11416-015-0261-z.

[13] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," Comput. Secur.,vol. 77, pp. 871–885, Aug. 2018, doi:10.1016/j.cose.2018.04.005.

[14] P. Mohamed Shakeel, S. Baskar, V. R. Sarma Dhulipala, S.Mishra, and M. M. Jaber, "Maintaining Security and Privacy in Health Care System Using Learning Based Deep-Q- Networks," J. Med. Syst., 2018, doi: 10.1007/s10916-018-1045-z.

[15] A. Azmoodeh, A. Dehghantanha, M. Conti, and K. K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," J. Ambient Intell. Humaniz. Comput., 2018, doi: 10.1007/s12652-017-0558-5.

[16] R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious Code Detection based on Image Processing Using Deep Learning," pp. 81–85, 2018, doi:10.1145/3194452.3194459.

[17] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," Futur. Gener. Comput. Syst., 2016, doi: 10.1016/j.future.2014.06.001.

[18] G. Conti, E. Dean, M. Sinda, and B. Sangster, "Visual reverse engineering of binary and data files," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008, doi: 10.1007/978-3-540-85933-8_1.

[19] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in

Vol 11, Issue 6, JUNE /2020
ISSN NO:0377-9254

Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security, CICS 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013, 2013, doi: 10.1109/CICYBS.2013.6597204.

[20] E. Cetinic, T. Lipic, and S. Grgic, "Fine-tuning Convolutional Neural Networks for fine art classification," Expert Syst. Appl., vol. 114, pp. 107–118, 2018, doi: 10.1016/j.eswa.2018.07.026.

[21] A. K. Reyes, J. C. Caicedo, and J. E. Camargo, "Fine-tuning deep convolutional networks for plant recognition," in CEUR Workshop Proceedings, 2015.

[22] A. Kaya, A. S. Keceli, C. Catal, H. Y. Yalic, H. Temucin, and B. Tekinerdogan, "Analysis of transfer learning for deep neural network based plant classification models," Comput.Electron. Agric., vol. 158, no. January, pp. 20–29, 2019, doi:10.1016/j.compag.2019.01.041.