The ISC Int'l Journal of Information Security
ISeCure

July 2022, Volume 14, Number 2 (pp. 181–192)
http://www.isecure-journal.org

Ransomware Detection Based on PE Header Using Convolutional Neural Networks **

Farnoush Manavi [1,*] and Ali Hamzeh [1]
[1] *Department of Computer Engineering and IT, Shiraz University, Shiraz, Iran.*

ARTICLE INFO.

Article history:
Received: December 20, 2020
Revised: July 14, 2021
Accepted: July 17, 2021
Published Online: August 28, 2021

Keywords:
Convolution Neural Network, Ransomware, Ransomware Detection

Type: Research Article

doi: 10.22042/ISECURE.2021.262846.595

dor: 20.1001.1.20082045.2022.14.2.6.8

ABSTRACT

With the spread of information technology in human life, data protection is a critical task. On the other hand, malicious programs are developed, which can manipulate sensitive and critical data and restrict access to this data. Ransomware is an example of such a malicious program that encrypts data, restricts users' access to the system or their data, and then request a ransom payment. Many types of research have been proposed for ransomware detection. Most of these methods attempt to identify ransomware by relying on program behavior during execution. The main weakness of these methods is that it is not explicit how long the program should be monitored to show its real behavior. Therefore, sometimes, these researches cannot detect ransomware early. In this paper, a new method for ransomware detection is proposed that does not need executing the program and uses the PE header of the executable file. To extract effective features from the PE header file, an image is constructed based on PE header. Then, according to the advantages of convolutional neural networks (CNN) in extracting features from images and classifying them, CNN is used. The proposed method achieves high detection rates. Our results indicate the usefulness and practicality of our method for ransomware detection.

1 Introduction

In recent years, the number of cybercrimes has grown dramatically [1]. One of the most serious threats to computer data and systems is ransomware [2], which has increased with the proliferation of cryptocurrencies such as bitcoin [3–5]. Ransomware is a malicious program that encrypts data, restricts users' access to the system or their data, and then requests a ransom payment [6, 7]. Ransomware is a threat to businesses, governments, and financial institutions worldwide [8, 9]. This malicious software usually uses a strong cryptography algorithm such as *advanced encryption standard* (AES) or *Rivest-Shamir-Adleman* (RSA) to encrypt data [10–12]. It is complicated to break these encryption algorithms and find a decryption key. Therefore, before ransomware is installed on the system, it must be detected [13]. There are several approaches to detect ransomware. In general, three approaches can be used to extract the features and then a detection model is trained [14]. These approaches are described below.

Approaches based on dynamic features. This method requires program execution in a virtual environment to extract the features. In this method, features are extracted according to the program activ-

[*] Corresponding author.
[**]This article is an extended/revised version of an ISCISC'17 paper.
Email addresses: F.manavi@cse.shirazu.ac.ir, Ali@cse.shirazu.ac.ir

ISSN: 2008-2045 © 2020 ISC. All rights reserved.

ities at run time. For example, for each application, dynamic linked libraries (DLLs), file system activities, registry activities, and hardware events are considered [15–19]. In a virtual environment, some malicious program does not show their real behavior after running [20]; thus, they bypass the detection system in this way. Therefore, it is not clear how long each program should be monitored to observe its real behavior.

Approaches based on static features. This category does not require program execution to extract features. In this category, each program's file is decomposed at the assembly level and features are extracted based on the obtained Opcodes [21–24]. In some cases, the features are extracted from the file bytes or the fields on file header [25, 26]. The disadvantage of these methods is that if the program file is packed, extracting popular features such as Opcode will not be efficient, and smart feature extraction alternatives should be sought.

Approaches based on hybrid features. In this category, features are extracted using static and dynamic approaches. In other words, the advantages of the two previous methods are used together and a set of features will be obtained [24, 27, 28]. In some cases, due to the use of static and dynamic features together, this approach will be very time-consuming. After extracting the features, a model should be used for classification. Machine learning classifiers such as Random Forest, Adaboost, K-Nearest Neighbor, and Support Vector Machine are the most common methods [21, 25–30]. Recently, neural networks are used to classify the extracted features [31–34].

This article uses a static method that does not require a program execution to extract features. Each application file consists of two parts, header and data. Ransomware and benign programs are developed for various targets, which results in variations in their structure and header. The program header contains useful data about each program. Therefore, in the proposed method, relying on this subject, the portable executable (PE) file header is used to extract the features. The difference between the proposed method and other static methods is the use of raw header bytes. Like static methods which are based on Opcodes, it does not require file preprocessing or domain knowledge about the fields that make up the header. If the program files are packed, the proposed method can still detect ransomware with high accuracy, while the previously presented static methods lose their accuracy against packed files.

The remainder of this paper is organized as follows. In Section 2, related works on ransomware detection are discussed. In Section 3, the proposed method for converting PE header to an image, and learning convolutional neural network is explained. Section 4 describes the evaluation metrics, dataset, and obtained results. Section 5 shows more experiments and discusses the results. Finally, in Section 6, the conclusion is presented.

2 Related Work

In this section, some research on ransomware detection will be discussed.

Homayoun *et al.* [11] proposed a dynamic approach for ransomware detection. Due to the program's activities on files, they extracted a sequence of logs and used them to obtain features. Finally, they classified obtained features using *long short-term memory* (LSTM) and *convolutional neural network* (CNN). Their method is a dynamic solution, and it is not clear how long each program should be monitored to extract features.

Azmoodeh *et al.* [6] proposed a method based on energy consumption footprint. They used *PowerTutor* to monitor and record the device processes' power usage (while running the benign applications and ransomware, separately) for five minutes. Specifically, their proposed method monitors the energy consumption patterns of different processes to classify ransomware from benign applications. They then used Neural Networks, K-Nearest Neighbors, Support Vector Machine and Random Forest to train the detection model. This method also has the weakness of the Homayoun method [11] and the malicious program may be at rest in five minutes of monitoring and show its destructive behavior after this time.

Bae *et al.* [35] proposed a ransomware detection method based on machine learning algorithms. They extracted Application Programming Interface (API) invocation sequences using the Intel PIN tool in a dynamic analysis environment. For feature extraction, they considered N-gram API sequences. Finally, they applied Random Forest, K-Nearest Neighbor, Naïve Bayes, Support Vector Machine on the feature vector and classified each program. Obtaining API sequences needs resources and extends the detection time. Also, some ransomware detects the dynamic analysis environment.

Using N-gram Opcodes, Zhang *et al.* [30] proposed a patch-based CNN and Self-Attention Network for ransomware detection. They used IDA Pro software to extract Opcodes from each application file. They considered the N-gram Opcodes as a patch. Then they fed these patches to the self-attention Network and finally detected the ransomware program. The patch size will be affected on the result and should be specified according to the number of Opcodes obtained for each dataset.

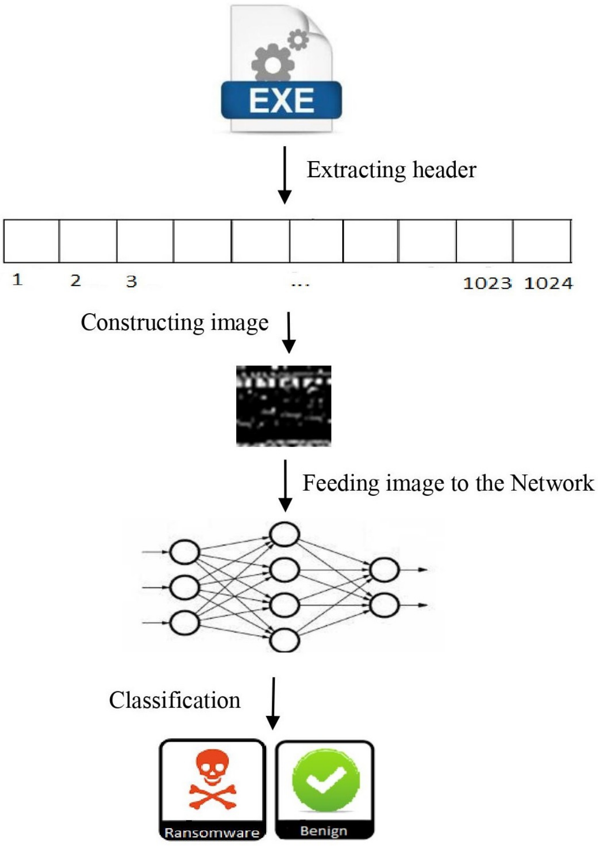

Figure 1. The overall structure of the proposed method

Zhang *et al.* [21] used Opcodes for ransomware detection, such as this approach [30]. They considered the frequency of N-gram Opcode sequences as a feature. Using Term Frequency-Inverse Document Frequency (TF-IDF), they considered important N-grams as final features. Then, they used machine learning algorithms to classify the obtained features.

Baldwin and Dehghantanha [29] proposed a static method for ransomware detection. Using nine feature selection algorithms, they obtained effective Opcodes for ransomware detection. Then, by considering the frequency of these Opcodes and Support Vector Machine, they detected the ransomware.
Using static features in these three methods [21, 29, 30] is an advantage, but it should be noted that when the program file is packed, we will face a major challenge, which requires unpacking the files.

Vidyarthi *et al.* [25] proposed a static method for ransomware detection. Their method is based on the PE header of executable files. For each program using some values from the header fields of the executable files, they extracted features. Finally, they applied these features to train a model based on Naïve Bayes, J48, and Random Forest.

Ashraf *et al.* [24] proposed a hybrid approach for ransomware detection. In static analysis, they extracted features from raw PE headers. In dynamic analysis, they ran each program in the Cuckoo sandbox environment [36] and constructed the binary feature vector from the Registry changes, Application Programming Interface (API) calls, and DLL's. Then, using Principal Component Analysis (PCA), they reduced the number of features. Finally, based on the selected features, they used machine learning techniques and Transfer Learning-based Deep convolutional neural networks (res-net18) [37] to detect Ransomware. This method has a time overhead and in the extraction phase of dynamic features, some important properties such as frequency of API calls and registry keys are ignored.

Previous researches required a different amount of preprocessing, time consumption, and resources to extract features. Ransomware differs from benign files in some respects due to its destructive nature. For example, ransomware header file is different from benign header file [25]. The header of the executable file contains useful information about each program, which can be used to detect the nature of the program. Therefore, in the proposed method, the header of executable files is used to detect ransomware. No special pre-processing is required with this choice, and ransomware detection is done in the shortest possible time.

3 Proposed Method

Due to the damage that the ransomware inflicts on the systems, they will cause irreparable damage to users or organizations if they are not identified early. Therefore, feature extraction is a critical step. The extracted features should have low computational complexity and time overhead and ability to identify ransomware programs well. The executable file of each program includes different sections. For example, executable files in the Windows operating system, known as .exe, consist of header and data sections. The header of executable files shows much information about the structure of that file. Therefore, in this paper, the header is used to extract the features. In recent years, several techniques such as Independent Component Analysis, Linear Filtering, Partial Differential Equations, Neural Networks have been presented for feature extraction. Neural Networks are among the most powerful methods available to extract features from images and finally categorize them [38, 39]. Thus, we convert the problem of file classification and ransomware detection into a problem of image classification to use the methods in this area and provide a ransomware detection method with a low computational complexity with high accuracy. Figure 1 shows a diagram of the proposed method.

The main steps of our detection method described

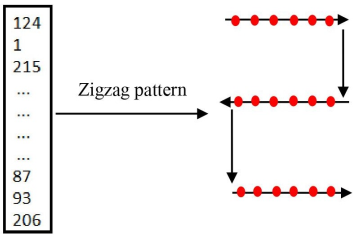

Figure 2. Pattern for converting extracted header to grayscale image

below:

- Extracting the header of executable files from each program
- Constructing a grayscale image using the extracted header
- Training convolutional neural network based on images
- Testing the model on unseen samples

3.1 Extracting the Header of Executable Files

Today, the Windows operating system is widely used. The PE is a format file in the Windows operating system used for executable, DLL, and object code files. Each file in this format consists of two main parts, header and data. In each file, the header shows the structure of that file, which has important information about the nature of the executable file. Therefore, in this paper, using the "pefile" library in the Python programming language, the header of each executable file is extracted and a vector with 1024 components is obtained. Each element in this vector has a value from 0 to 255.

3.2 Constructing a grayscale image using the extracted header

The header obtained from the previous step is a vector with 1024 components, which can be processed with many techniques. PE header includes different parts such as DOS Header, PE Signature, COFF Header, Optional Header, and Section Header, all of which do not have the same value. For example, the PE Signature section contains 4 bytes, indicating the value of "PE / 0 / 0". This section specifies that a file is in PE format, so the value of this field is always constant in ransomware and benign files. In this research, to be able to use raw header information more efficiently, they are converted into an image. In this case, the important and distinctive parts in the header of the benign and ransomware files will show themselves as different objects in the images. Finally, in the next steps, Neural Networks' capabilities are used to extract features from these images and a method with a high detection rate is presented. To construct an image, header bytes can be placed side by side in different techniques. In order to increase the continuity of the bytes, using the zigzag pattern shown in Figure 2, each header is converted into a 32*32 grayscale image. Figure 3 shows eight images constructed in the proposed method.

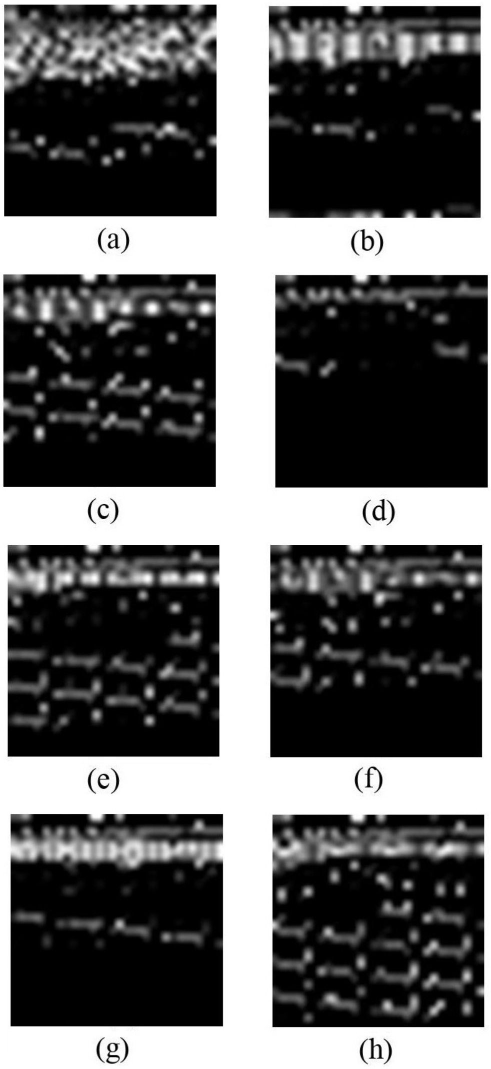

Figure 3. Examples of images generated by the proposed method. Figures a, b, c, d are different family of ransomware, and Figures e, f, g, h are benign samples.

3.3 Training Convolutional Neural Network

Convolutional neural network (CNN) is a type of Deep Neural Network widely used in pattern and image recognition [40]. In this paper, CNN is used to categorize the images obtained from each executable file. Each CNN has important layers under the name convolutional, pooling, and fully connected. The functions of the convolutional layers and pooling are feature

Table 1. Parameters of the CNN

Layer number	Layer (type)	Parameters	Number of parameter
1	BatchNormalization	-	4
2	Convolutional	Kernel˙size= (3, 3), Filter= 64, Activation= 'relu', padding='valid'	640
3	MaxPooling	Pool˙size= (2, 2)	0
4	Dropout	Rate= 0.3	0
5	Convolutional	Kernel˙size= (3, 3), Filter =128, Activation= 'relu', padding='valid'	73856
6	MaxPooling	Pool˙size= (2, 2)	0
7	Flatten	-	0
8	Dropout	Rate= 0.5	0
9	Fully Connected	utput Filter size= 16, Activation= 'relu'	73744
10	Batch Normalization	-	64
11	Fully Connected	Output Filter size= 2, Activation= 'softmax'	37

extraction. Fully connected layer takes the results of convolutional and pooling processes and uses them to label and classify images. As the number of layers in the Network increases, the complexity and possibility of over-fitting increases. Therefore, in this paper, with the least possible number of layers, a model is trained to extract the features and classify the samples.

In the proposed method, *BatchNormalization* is used. BatchNormalization speeds up the learning process. By using it, each layer of the Network is trained a little bit more independently from other ones. BatchNormalization is required for increasing the stability of a Neural Network. In the middle layers, due to the operations applied to the input data, the effect of BatchNormalization is neutralized, so in this paper, it is used after the input layer and before the output layer. In order for the Network to have more generalization and to prevent overfitting, *dropout* is done after several layers and some of the Network weights are deactivated. Finally, using the *Flatten* layer, the resulting features are converted into vectors and used for the final layers of the Network. Table 1 shows the specifications of the proposed Network in this paper. During Network training, *number-of-epochs= 100*, *learning-rate= 0.001*, and *optimizer= 'rmsprop'* are considered. Other Network parameters not mentioned in the article have the default value of *Keras API*.

3.4 Testing the Model on Unseen Samples

Using the trained Network disucussed in Section 3.3, the classifier is evaluated on the test data. To this aim, steps discussed in Section 3.1 and Section 3.2 are repeated, and according to the header of the executable files, a 32*32 image of the program is constructed. Then, the trained Neural Network in the previous step is used to detect the label of that image, and the suspicious program is divided into two categories: ransomware and benign.

4 Dataset and Result

In this section, the evaluation experiments of the proposed method will be examined. For this purpose, the dataset, evaluation metrics, and experimental environment used in this paper are described.

4.1 Dataset 1

This dataset contains 2000 executable files in PE format[1]. The non-malicious data used in this dataset contains 1000 EXE files that have been downloaded from FreewareFiles[2], SnapFiles[3] and PortableApps[4] sites. This dataset is scanned using ESET NODE32 to ensure that they are non-malicious software. The size of these files is from 1 KB to 165 MB. The ransomware files were downloaded from the VirusShare[5] database. In this paper, 1000 ransomware have been randomly selected. These files include different types of ransomware, such as Cerber, Locky, Torrent, and Tesla. The size of the selected files is between 1 KB and 18 MB.

4.2 Dataset 2 (Imbalanced dataset)

In the real world, the number of benign files is much more than the number of ransomware files. Hence, it

[1] In order to access the data, send an email to f.manavi@cse.shirazu.ac.ir

[2] www.freewarefiles.com

[3] www.snapfiles.com

[4] www.portableapps.com

[5] www.virusshare.com

Table 2. The number of samples in the various family in dataset 3

TeslaCrypt	Sage	Wannacry	Locky	Cerber
97	137	213	219	211

is more realistic to evaluate the classifier on an imbalanced dataset. Therefore, this dataset has been collected to evaluate the proposed method and competitor methods. This dataset contains 6000 benign and 1000 ransomware files. Its ransomware files are collected from VirusTotal[6] and VirusShare websites. These files include various types of ransomware such as Wannacry, Cerber, Locky, etc. Its benign files are collected from "System32" and "Program Files" folders of standard Windows and programs available from freewarefiles, snapfiles and portableapps websites.

4.3 Dataset 3

This dataset contains 877 ransomware samples in 5 different families under the name Cerber, Locky, Wannacry, Sage and TeslaCrypt. These files were obtained by crawling on the VirusTotal and VirusShare sites. Table 2 shows the number of files in the various family.

4.4 Evaluation Metrics

Common machine learning evaluation metrics such as Precision, Recall, F-measure, and accuracy are used to evaluate the results [41]. Relations 1-4 show how to obtain these metrics.

- **True positive (TP)**: Indicates the number of samples that are correctly labeled as the benign class.
- **True negative (TN)**: Indicates the number of samples that are correctly labeled as the ransomware class.
- **False positive (FP)**: Indicates the number of samples which are incorrectly labeled to be in the benign class.
- **False negative (FN)**: Indicates the number of samples which are incorrectly labeled as ransomware class.
- **Precision**: Precision for a specific class, is the result of dividing the number of rightly predicted samples of that class by the total number of samples that are predicted as the same class [41]. Relation 1 shows how to obtain Precision.

$$Precision = \frac{TP}{(TP + FP)} \tag{1}$$

- **Recall**: Recall for a specific class, is the result of dividing the number of rightly predicted samples of that class by the total number of class samples [41]. Relation 2 shows how to obtain Recall.

$$Recall = \frac{TP}{(TP + FN)} \tag{2}$$

- **F-measure (F1)**: F-measure is the harmonic mean of Precision and Recall [41]. Relation 3 shows how to obtain F-measure.

$$F - measure = 2 * \frac{(Presision * Recall)}{(Presision + Recall)} \tag{3}$$

- **Accuracy**: Accuracy is the total number of samples that are correctly predicted, divided by the total number of samples [41]. Relation 4 shows how to obtain accuracy.

$$Accuracy = \frac{(TP + FN)}{(TP + FP + TN + FN)} \tag{4}$$

To detect overfitting, cross validation technique has been used to evaluate the results. For this purpose, the dataset is divided into 10 equal parts. Each time one of the 10 parts is selected as a test set and the remaining 9 parts are selected as the training data set to build the model. This will be repeated 10 times and the average result of all the steps are represented as final result [42].

4.5 Experimental Setup

The proposed method is programmed using Python 3.7, which runs on a Windows 10 system with Intel (R) Core (TM) i5-2400 CPU specifications @ 3.10GHz with 32GB RAM.

4.6 Experimental Results

4.6.1 Comparison of the Proposed Method with Ransomware Detection Methods

To evaluate the performance of the proposed method, the performance results will be compared with these approaches [21, 24, 25, 29]. In this paper, a static method for ransomware detection was proposed, so to evaluate its performance, it is compared with static methods such as [21, 25, 29]. These two methods [21, 29] extracted features according to Opcodes from the executable files and then used machine learning methods for the classification task. In Vidyarthi method [25], as in the proposed method, the header is employed for the detection model. The difference is that in this method, according to the fields that make up the header of the executable files, several features are extracted. Then, these features are classified using Random Forest. To compare the proposed method with a dynamic method, Ashraf method [24] has been selected. In this method, two types of static and dynamic features are extracted from the programs. The combination of these features is used as input in the

[6] www.virustotal.com

Table 3. Compare the proposed method with other methods on dataset 1

Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Metric/Approach
93.40	**93.33**	**93.34**	**93.33**	**Proposed method**
92.78	92.75	92.74	92.75	**Zhang *et al.*** [21]
92.08	91.93	91.92	91.93	**Ashraf *et al.*** [24]
90.67	90.43	90.40	90.43	**Vidyarthi *et al.*** [25]
90.21	90.17	90.12	90.17	**Baldwin and Dehghantanha** [29]

Table 4. Compare the proposed method with other methods on dataset 2

Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Metric/Approach
94.99	**95.11**	**95.00**	**95.11**	**Proposed method**
94.22	94.38	94.26	94.38	**Zhang *et al.*** [21]
93.95	94.13	94.01	94.13	**Ashraf *et al.*** [24]
93.86	94.08	93.90	94.08	**Vidyarthi *et al.*** [25]
92.10	92.37	91.58	92.37	**Baldwin and Dehghantanha** [29]

Table 5. Compare executing time of proposed method with other methods on dataset 1

Test time (s)	Training time (s)	Time/Approach
63.44	**571.32**	**Proposed method**
373.47	3798.85	**Zhang *et al.*** [21]
24300.23	218000.54	**Ashraf *et al.*** [24]
65.60	590.65	**Vidyarthi *et al.*** [25]
366.42	3322.67	**Baldwin and Dehghantanha** [29]

Table 6. Compare executing time of proposed method with other methods on dataset 2

Test time (s)	Training time (s)	Time/Approach
190.02	**1894.78**	**Proposed method**
1403.77	13336.18	**Zhang *et al.*** [21]
84950.58	761023.949	**Ashraf *et al.*** [24]
216.06	2389.20	**Vidyarthi *et al.*** [25]
1699.01	15498.37	**Baldwin and Dehghantanha** [29]

resnet18 [37] and finally, the ransomware is detected. Table 3 and Table 4 show the average accuracy, F-measure, Recall and Precision of ransomware and benign instances. Table 5 and Table 6 show the average execution time of model training and test in 10-fold cross validation.

Our proposed method has a higher detection rate than the mentioned methods and performs model training and testing of suspicious samples in less time. One of the reasons for the better results of the proposed method lies in the fact that it uses the PE header. Each benign and ransomware file has a different header depending on its performance, so header information can effectively detect ransomware's malicious nature. The executable file header consists of different sections such as DOS Header, PE Signature, COFF Header, Optional Header, and Section Header, all of which do not have the same value. In the proposed method, to better process header information and extract more effective features from it, header bytes are converted into images. In this case, the important and distinctive parts in the header of the benign and ransomware files will show themselves as different objects in the images. Therefore, in the proposed method, 1024 bytes of the header will be placed next to each other using the zigzag pattern. The proposed method uses a Neural Network to extract the features in the images. Convolutional neural network has the ability to extract local and global features in images and use them for the categorization phase. Therefore, the proposed method has been able to identify ransomware programs with a higher detection rate.

4.6.2 Comparison of the Proposed Method with Malware Detection Methods

Given that ransomware is a type of malware, in the following, we will compare our proposed method with several methods that have been presented for the classification of malware. Since the proposed method statically extracts features from program files, we chose three methods [43–45] that statically detect malicious programs.

Gibert *et al.* [43] use executable file bytes to detect malware. They construct a grayscale image from the whole byte of the file. Then, they convert this image to a fixed size of 256*256 pixel using the downsampling technique. Finally, they classify these images using CNN.

Le *et al.* [44], using the downsampling technique, extract a vector with 10,000 components from the file's whole bytes. Then, they feed this vector to LSTM and CNN for classification.

Kumar *et al.* [45] extract two types of features (Raw features and Derived features) based on PE header fields. Then, they apply the obtained features to machine learning classifiers such as K-Nearest Neighbor, Naïve Bayes, and Random Forrest.

Table 7 shows the average accuracy, F-measure, Recall and Precision on ransomware and benign instances of the proposed method and these methods. Table 8 shows the average execution time of model training and test in 10-fold cross validation.

As the results of Table 8 illustrates, the proposed method, with the proper use of raw bytes of header, specifies the label of a program in less time than the methods mentioned. In Gibert [43] and Le [44] method, all bytes of the program are reviewed. Then, using the downsampling technique, features are extracted from all bytes of the program, which increases the training and testing time of the model. Kumar method [45] uses only header bytes to extract the feature and according to them, it extracts two types of features, which makes the training and testing time of a sample less. In this method, to extract features from header bytes, domain knowledge about the executable file header structure is required. In contrast, in the proposed method, it is not necessary to specify the different header sections.

5 Discussion

In this section, three experiments are discussed to examine the proposed method further.

5.1 Testing the proposed Network on whole bytes of the file

In the proposed method, 1024 bytes of the header are converted into a 32*23 pixel image and then CNN is used to extract features and classify samples. This time, to determine the header's effect in the proposed Network, the whole bytes of the file are used to create the images, and to train, these images will be fed to the Network designed in Section 3.3. The number of bytes of each program file will be different from other programs. Therefore, the downsampling technique [43] is used to create images of the same size. Each program file will have an average size of a few megabytes, and the use of downsampling technique to create a 32*32 pixel image of each program file will cause a large amount of data to be lost. Therefore, for a fairer comparison with whole bytes of the file, a 320*320 pixel image is be created; hence other parts of the executable file and header are effective in image construction. Table 9, Table 10, Table 11 and Table 12 show the result of executing the proposed Network using the images obtained from the whole file and header bytes.

As the results show, the use of header bytes alone will show higher accuracy in ransomware detection. Table 9 and Table 10 show the Network's train time in two modes of constructing images from the whole file and header bytes. As a result, using the header alone, will greatly reduce the training time of the model. Henceforth, selecting a header for ransomware detection will be a wise choice, and the accuracy and time consumption of detection will change optimally with this selection.

5.2 Testing the Proposed Method for the Ransomware Family Categorization

In this section, the proposed method will be tested to identity different families of ransomware. The third dataset will be used for this purpose. As mentioned in section 4.3, this dataset includes five important families of ransomware. In the last layer of the proposed Network, five neurons will be used to identify these five families instead of two neurons. Table 13 shows the results of executing the proposed method for ransomware categorization on the third dataset.

As the results in Table 13 indicate, the proposed method for ransomware categorization is also very accurate. These results show that converting headers to 32*32 pixel images is a wise choice that will summarize the nature of each file in these images. Finally, by using the designed network, the nature of each program can be well detected.

Table 7. Compare the proposed method with malware detection methods on dataset 1

Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Metric/Approach
93.40	**93.33**	**93.34**	**93.33**	**Proposed method**
90.01	89.95	89.94	89.95	**Gibert *et al.*** [43]
91.10	90.95	90.94	90.95	**Le *et al.*** [44]
92.92	92.89	92.88	92.89	**Kumar *et al.*** [45]

Table 8. Compare executing time of proposed method with malware detection methods on dataset 1

Test time (s)	Training time (s)	Time/Approach
63.44	**571.32**	**Proposed method**
8771.51	94188.73	**Gibert *et al.*** [43]
4210.56	44712.36	**Le *et al.*** [44]
93.73	2745.98	**Kumar *et al.*** [45]

Table 9. Executing time of the proposed Network in two modes of constructing images from whole file and header bytes on dataset 1

Test time (s)	Training time (s)	Time/Approach
1608.51	79791.92	**Proposed network whole file**
63.44	**571.32**	**Proposed network with header bytes**

Table 10. Executing time of the proposed Network in two modes of constructing images from whole file and header bytes on dataset 2

Test time (s)	Training time (s)	Time/Approach
3807.32	250175.78	**Proposed network whole file**
190.02	**1894.78**	**Proposed network with header bytes**

Table 11. The result of proposed Network with whole file and header bytes on dataset 1

Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Metric/Approach
90.69	90.50	90.48	90.50	**Proposed network with whole file**
93.40	**93.33**	**93.34**	**93.33**	**Proposed network with header bytes**

Table 12. The result of proposed Network with whole file and header bytes on dataset 2

Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Metric/Approach
92.64	92.85	92.17	92.85	**Proposed network with whole file**
94.99	**95.11**	**95.00**	**95.11**	**Proposed network with header bytes**

Table 13. The result of proposed Network for ransomware categorization

Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Metric/Approach
94.59	94.41	94.39	94.41	**Proposed method for ransomware categorization**

Table 14. Compare the proposed method with other methods on packed dataset

Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Metric/Approach
83.35	**82.93**	**82.88**	**82.93**	**Proposed method**
63.99	62.24	61.02	62.24	**Zhang *et al.*** [21]
80.50	80.27	80.23	80.27	**Vidyarthi *et al.*** [25]
61.23	60.71	60.23	60.71	**Baldwin and Dehghantanha** [29]

5.3 Testing the Proposed Method on the Packed Dataset

If the executable file of the program is packed, in most cases, the static methods of ransomware detection will have problems and their accuracy will be reduced. In this section, the effect of packing the program file on the proposed method and other static mentioned methods will be examined. For this purpose, the training data of the first dataset is fed to the Network same as before; only the test data is packed using the UPX tool. These packed test samples are then used to evaluate the performance of the methods. The results of this experiment are presented in Table 14. In this experiment, only static ransomware detection methods have been investigated.

As the results show, the two methods [21, 29] that use the program's Opcodes, are less accurate than the packed test files. Since the program executable file is packed, when the program Opcodes are extracted, they will not be the main Opcodes, and these Opcodes are to take the program out of the pack format and run the program. Therefore, static methods based on Opcodes are not resistant to packed files and lose their accuracy. To improve the accuracy of these methods, one must first look for solutions to unpacked the file, which in some cases is not simple.

Vidyarthi [25] method, which extracted features from some header fields, is still resistant to packed files and shows that extracting attributes from the header is an accurate choice. However, this method's disadvantage is that it extracts features only from some header fields and some important fields are ignored. In the proposed method, the first 1024 bytes of the file that make up the executable file header are required to extract the features. If the program files are also packed, accessing these 1024 bytes will not be difficult. The results in Table 14 show that if the program files are also packed, the proposed method will still detect ransomware with proper accuracy.

6 Conclusion

In this paper, a static method based on feature extraction from the PE header was proposed. Ransomware is structurally different from benign files due to their destructive nature. This difference is evident in the header. The program header contains useful data about each program that can be used to detect ransomware. The raw header bytes are converted into an image in the proposed method to show the header's important parts in benign and ransomware files as different objects in their images. Using these images, a convolutional neural network is trained, and the ransomware files are identified. In this paper, the proposed method is fairly compared with other methods, and the result shows the efficiency and accuracy of the proposed method. The advantage of the proposed method is that it does not need to run the program to extract the feature, and each file, regardless of its size, is summarized in the form of a 32*32 pixel image.

References

[1] Nabie Y Conteh and Paul J Schmick. Cybersecurity risks, vulnerabilities, and countermeasures to prevent social engineering attacks. In *Ethical Hacking Techniques and Countermeasures for Cybercrime Prevention*, pages 19–31. IGI Global, 2021.

[2] Fakhroddin Noorbehbahani, Farzaneh Rasouli, and Mohammad Saberi. Analysis of machine learning techniques for ransomware detection. In *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, pages 128–133. IEEE, 2019.

[3] Kim-Kwang Raymond Choo. Cryptocurrency and virtual currency: Corruption and money laundering/terrorism financing risks? In *Handbook of digital currency*, pages 283–307. Elsevier, 2015.

[4] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoit Dupont. Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, 5(1):tyz003, 2019.

[5] Abbas Yazdinejad, Hamed HaddadPajouh, Ali Dehghantanha, Reza M Parizi, Gautam Srivastava, and Mu-Yen Chen. Cryptocurrency malware hunting: A deep recurrent neural network approach. *Applied Soft Computing*, 96:106630, 2020.

[6] Amin Azmoodeh, Ali Dehghantanha, Mauro Conti, and Kim-Kwang Raymond Choo. Detecting crypto-ransomware in iot networks based

on energy consumption footprint. *Journal of Ambient Intelligence and Humanized Computing*, 9(4):1141–1152, 2018.

[7] Mamoona Humayun, NZ Jhanjhi, Ahmed Alsayat, and Vasaki Ponnusamy. Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, 22(1):105–117, 2021.

[8] Wira Zanoramy A Zakaria, Mohd Faizal Abdollah, Othman Mohd, and Aswami Fadillah Mohd Ariffin. The rise of ransomware. In *Proceedings of the 2017 International Conference on Software and e-Business*, pages 66–70, 2017.

[9] Pierre-Luc Pomerleau and David L Lowery. The evolution of the threats to canadian financial institutions, the actual state of public and private partnerships in canada. In *Countering Cyber Threats to Financial Institutions*, pages 47–85. Springer, 2020.

[10] K Savage, P Coogan, and H Lau. The evolution of ransomware, symantec security response. *Symantec Corporation, Mountain View, CA*, 2015.

[11] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, and Raouf Khayami. Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. *IEEE transactions on emerging topics in computing*, 8(2):341–351, 2017.

[12] Bander Ali Saleh Al-Rimy, Mohd Aizaini Maarof, Mamoun Alazab, Syed Zainudeen Mohd Shaid, Fuad A Ghaleb, Abdulmohsen Almalawi, Abdullah Marish Ali, and Tawfik Al-Hadhrami. Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection. *Future Generation Computer Systems*, 115:641–658, 2021.

[13] Ala Bahrani and Amir Jalaly Bidgly. Ransomware detection using process mining and classification algorithms. In *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, pages 73–77. IEEE, 2019.

[14] Laxmi B Bhagwat and Balaji M Patil. Detection of ransomware attack: A review. In *Proceeding of International Conference on Computational Science and Applications*, pages 15–22. Springer, 2020.

[15] Amir Afianian, Salman Niksefat, Babak Sadeghiyan, and David Baptiste. Malware dynamic analysis evasion techniques: A survey. *ACM Computing Surveys (CSUR)*, 52(6):1–28, 2019.

[16] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*, 2016.

[17] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)*, 44(2):1–42, 2008.

[18] Mahboobe Ghiasi, Ashkan Sami, and Zahra Salehi. Dyvsor: dynamic malware detection based on extracting patterns from value sets of registers. *The ISC International Journal of Information Security*, 5(1):71–82, 2013.

[19] Ibrahim Bello, Haruna Chiroma, Usman A Abdullahi, Abdulsalam Ya'u Gital, Fatsuma Jauro, Abdullah Khan, Julius O Okesola, and M Abdulhamid Shafi'i. Detecting ransomware attacks using intelligent algorithms: recent development and next direction from deep learning and big data perspectives. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–19, 2020.

[20] Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. Detecting environment-sensitive malware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 338–357. Springer, 2011.

[21] Hanqi Zhang, Xi Xiao, Francesco Mercaldo, Shiguang Ni, Fabio Martinelli, and Arun Kumar Sangaiah. Classification of ransomware families with machine learning based on n-gram of opcodes. *Future Generation Computer Systems*, 90:211–221, 2019.

[22] Jeong Kyu Lee, Seo Yeon Moon, and Jong Hyuk Park. Cloudrps: a cloud analysis based enhanced ransomware prevention system. *The Journal of Supercomputing*, 73(7):3065–3084, 2017.

[23] Juan A Herrera Silva, Lorena Isabel Barona López, Ángel Leonardo Valdivieso Caraguay, and Myriam Hernández-Álvarez. A survey on situational awareness of ransomware attacks—detection and prevention parameters. *Remote Sensing*, 11(10):1168, 2019.

[24] Arslan Ashraf, Abdul Aziz, Umme Zahoora, Muttukrishnan Rajarajan, and Asifullah Khan. Ransomware analysis using feature engineering and deep neural networks. *arXiv preprint arXiv:1910.00286*, 2019.

[25] Deepti Vidyarthi, CRS Kumar, Subrata Rakshit, and Shailesh Chansarkar. Static malware analysis to identify ransomware properties. *International Journal of Computer Science Issues (IJCSI)*, 16(3):10–17, 2019.

[26] Ban Mohammed Khammas. Ransomware detection using random forest technique. *ICT Express*, 6(4):325–331, 2020.

[27] Alberto Ferrante, Miroslaw Malek, Fabio Martinelli, Francesco Mercaldo, and Jelena Milosevic.

Extinguishing ransomware-a hybrid approach to android ransomware detection. In *International Symposium on Foundations and Practice of Security*, pages 242–258. Springer, 2017.

[28] Suyeon Yoo, Sungjin Kim, Seungjae Kim, and Brent Byunghoon Kang. Ai-hydra: Advanced hybrid approach using random forest and deep learning for malware classification. *Information Sciences*, 546:420–435, 2021.

[29] James Baldwin and Ali Dehghantanha. Leveraging support vector machine for opcode density based detection of crypto-ransomware. In *Cyber threat intelligence*, pages 107–136. Springer, 2018.

[30] Bin Zhang, Wentao Xiao, Xi Xiao, Arun Kumar Sangaiah, Weizhe Zhang, and Jiajia Zhang. Ransomware classification using patch-based cnn and self-attention network on embedded n-grams of opcodes. *Future Generation Computer Systems*, 110:708–720, 2020.

[31] Hyunji Kim, Jaehoon Park, Hyeokdong Kwon, Kyoungbae Jang, and Hwajeong Seo. Convolutional neural network-based cryptography ransomware detection for low-end embedded processors. *Mathematics*, 9(7):705, 2021.

[32] G Radhakrishnan, K Srinivasan, S Maheswaran, K Mohanasundaram, D Palanikkumar, and Abhay Vidyarthi. A deep-rnn and meta-heuristic feature selection approach for iot malware detection. *Materials Today: Proceedings*, 2021.

[33] Muna Al-Hawawreh and Elena Sitnikova. Leveraging deep learning models for ransomware detection in the industrial internet of things environment. In *2019 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6. IEEE, 2019.

[34] Farnoush Manavi and Ali Hamzeh. Static detection of ransomware using lstm network and pe header. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–5. IEEE, 2021.

[35] Seong Il Bae, Gyu Bin Lee, and Eul Gyu Im. Ransomware detection using machine learning algorithms. *Concurrency and Computation: Practice and Experience*, 32(18):e5422, 2020.

[36] Digit Oktavianto and Iqbal Muhardianto. *Cuckoo malware analysis*. Packt Publishing Ltd, 2013.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020.

[39] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia computer science*, 132:377–384, 2018.

[40] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of convolution neural network advances on the imagenet. *Computer Vision and Image Understanding*, 161:11–19, 2017.

[41] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[42] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[43] Daniel Gibert, Carles Mateu, Jordi Planes, and Ramon Vicens. Using convolutional neural networks for classification of malware represented as images. *Journal of Computer Virology and Hacking Techniques*, 15(1):15–28, 2019.

[44] Quan Le, Oisín Boydell, Brian Mac Namee, and Mark Scanlon. Deep learning at the shallow end: Malware classification for non-domain experts. *Digital Investigation*, 26:S118–S126, 2018.

[45] Ajit Kumar, KS Kuppusamy, and G Aghila. A learning model to detect maliciousness of portable executable using integrated feature set. *Journal of King Saud University-Computer and Information Sciences*, 31(2):252–265, 2019.

Farnoush Manavi is a Ph.D candidate of artificial intelligence at Shiraz University since 2017. She also has a master's degree in information security from Shiraz University and a bachelor's degree in information technology. She is currently working as a lecturer at Technical and Vocational College, Shiraz, Iran. Her research interests are cyber security, machine learning applications in computer security and computer networks.

Ali Hamzeh is currently a full professor at Electrical and Computer Engineering School, Shiraz University, Shiraz, Iran. He received the Ph.D degree in artificial intelligence from Iran University of Science and Technology in 2007. He received his B.S. and M.Sc. degrees in computer engineering and artificial intelligence from Shiraz University in Iran. His research interests include malware detection, machine learning and social networks.