Journal of Positive School Psychology
2022, Vol. 6, No. 3, 8538–8547 

# Polymorphic Malware Detection based on Supervised Machine Learning

Nur Syuhada Selamat[1] ; Fakariah Hani Mohd Ali[2]

[1, 2] *Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, 40450 Selangor, Malaysia*
[1]*nursyuhadaselamat89@yahoo.co,my,* [2]*fakariah@tmsk.uitm.edu.my*

**Abstract**

Currently, the size of malicious software grows faster each year and poses a thoughtful global security threat. The number of malware developed is increasing as computers became interconnected, at an alarming rate in the 1990s. This scenario caused a rising number of malware. It also caused many protections are developed to against the malware. Malware authors have created them to be more challenging to be evaded from anti- virus scanner. Extracting the behaviour of polymorphic malware is one of the major issues that affect the detection result. The main objectives in this work are to extract the best feature selection and to increase detection of polymorphic malware. This study used machine learning to improve malware detection accuracy. This research used four types of machine algorithm which are K-Nearest Neighbours, Decision Tree, Logistic Regression, and Random Forest. As with most studies , careful attention was paid to false positive and false negative rates which reduced their overall detection accuracy and effectiveness. The result showed that the Random Forest algorithm is the best detection accuracy compares to others classifier with 99 % on a relatively small dataset. The implementation of a feature selection technique plays an important role in machine learning algorithms to increase the performance of polymorphic malware detection

**Keywords:** malware, machine learning , polymorphic

## I. INTRODUCTION

Computer networks are highly essential nowadays because of the numerous advantages they provide (2018). The number internet users is also on a rise (2012). This scenario has given an opportunity of cyber attack. Malicious software, or in short called malware refers to any software designed to cause damage to a single computer, server, or computer network(2020). In both malware and antivirus strategies, where many anti viruses attempt to alleviate, massive growth has occurred (2017). In different ways, this malware has infected various components (2015). User data and information systems on personal computers and mobile devices are at risk from this malicious software. Since its inception in the 1960s, malware has grown into the most important threat to computer systems in the last three decades. Researchers (2015) proposed a method based on behavioral analysis on machine learning that focused on classification and clustering of malware. In their experiment, they used two types of classifiers which are K-Means and Logic Model tree algorithms. The result showed that 82% aimed to corrupt in the computer system or network resources while 18% of analyzed malware were embedded with networking capabilities to connect the outer world. By using both static and dynamic features of malware and machine learning technique (2019) these methods provide the efficient automated classification of malwares . In their experiment , the static features were extracted from the binary code while in dynamic analysis was done by using the tool cuckoo sandbox that focused on system called sequences. The authors made a comparison by

using static, dynamic and integrated method by using two classifiers which were random forest(RF) and SVM. The accuracy detection showed for integrated method in RF 97.68% while 98.71% using SVM algorithm.

In this paper, researchers present polymorphic malware detection based on feature selection approach. Then, these features continued combine with four machine learning algorithm which are K-Nearest Neighbors (K-NN), Decision Tree (DT) and Support VectorMachine (SVM) for malware detection by using portable executable (PE) information as a features extraction. The PE structure contains of a PE file header and a section table followed by the section's data (2014). The results showed that DT is the best machine learning technique to detect malware with 99% detection accuracy.

## II. AN OVERVIEW OF POLYMORPHIC MALWARE

Polymorphic malware is a malware programmed that looks different each time which it is replicated but keeping the original code intact. This type of malware contains encrypted malware with the decryption routine module. The polymorphic generator used to mutate the code while keeping the original algorithm intact.

This type of malware is less likely to be detected by an antivirus application. The most commonly used techniques for writing polymorphic malware are encryption, decryption and data appending. These techniques use code obfuscations to evade antivirus scanners. Thus, an effective method has to be used to detect unknown malware with obfuscation; the machine learning method is now the most effective approach, particularly for abnormal malware (2018).

Obfuscation techniques are the techniques used by malware writers to make a source code difficult to read, understand, reverse engineering, and conceal the malware malicious intent. Virus writers use different code obfuscation techniques. This polymorphic malware is generated using obfuscation techniques(2015).

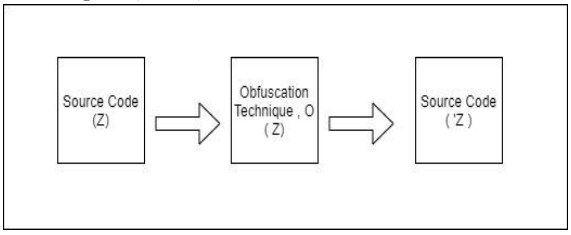


**Figure 1 Code Obfuscation**

Source : data from research (2017)

Based on Figure 1, polymorphic malware used the obfuscation technique to avoid signature-based detection. Given Z as an original source code, malware authors then used this technique and make its original code Z become Z', which contains a malicious code that difficult to reverse engineering.

## III. RESEARCH METHOD

### *I. Data Collection*

Several methods are used to collect, design, and conduct the study in the form of data. In this study, all the sample malware used are listed. This study continued with data collection, which was acquired from three types of samples. All samples were collected about 1025 samples, including 155 known benign files, 770 unknown files, and 100 of known Win32.Sality polymorphic malware. All these files sample were gathered from open source repositories that availble for malware researchers likes VirusShare webpages (www.virusshare.com) , VirusTotal, and VXHeaven.

Some benign software, obtained from the original Windows folder, which is Windows XP and Window 7 OS and Program Files folder, and used commercial software to prove either each executable was indeed benign or clean.

### *II. Software Tools*

**Table 1 - Requirement tools**

| Requirements | Function |
|---|---|
| Window XP and 7 | Platform of Operating System (OS) to detect polymorphic malware. |
| Process Explore | Displays all the processes running on a system and shows them in a tree structure that displays child and parent connections. |
| PE iDentifier (PEiD) | To detect the executable files packed or not. |
| InstallRite | Tool to detect the system configuration changes during malware executing. |
| HexCmp | To compare file in the system before and after malicious file execution. |
| VirusTotal | Scanning tool by multiple antivirus engine. |
| AnacondaPython | Set up Python Environment for Machine Learning. It is a powerful scientific environment written in Python designed by and for scientists, engineers and data analysts. It allows the Scikit- Learn and Pandas library to be used to load datasets, perform data preprocessing, assist in the selection of features, perform data splitting, help train and test datasets and can generate confusion matrix tables. Confusion matrix facilitates the evaluation of techniques designed as it can generate TP, FP, TN and FN values. |

Source : data from research (2017)

Table 1 shows the list requirement tools required in this experiment. The proposed detection approach illustrated by the flowchart in Figure 2. Based on the Figure 2 , first the researchers did dynamic analysis process to detect polymorphic malware file then the process continued to form a new data set which contained unknown files, polymorphic files and benign files. All these data set were extracted the features, in this research the parameters of Random Forest was created which the value number of tree (n_estimators) is equal to 100 (n_estimators=100).

Then this model started to train the data set with the features. If the condition at each node were satisfied, then the variable subset was chosen. Every subset of features was gained based on feature importance value. Then, these features were calculated as the decrease in node impurity weighted by the probability of reaching that node.The measure based on which optional condition choosen is known as impurity.When training a tree, feature importance was calculated as the decrease in node impurity weighted in tree.The higher the value, the more important the features.The measure of impurity based on Gini index which this Gini index measures for attribute selection. Then, the features were sorted from high to low according to their importance.As the classifier learned the appropriate weight for a given features then it removed the feature from the consideration. The “feature importance” attribute of RandomForestClassifier class exposed the importance of each feature. The node probability was calculated by the number of samples that reach the node, divided by the total number of samples.The higher the value the more important the feature.

After generating the best subset of features, these subset features were implemented to all four machine learning algorithm processes. Using embedded methods are a very straightforward approach for selecting good features for machine learning models.

The next process is all four ML techniques were developed using a specific class for example Random Forest. Lastly, all the techniques were evaluated based on detection accuracy, False Positive rate, True Positive rate, True Negative rate, False Negative rate, precision, recall, and f1-score.

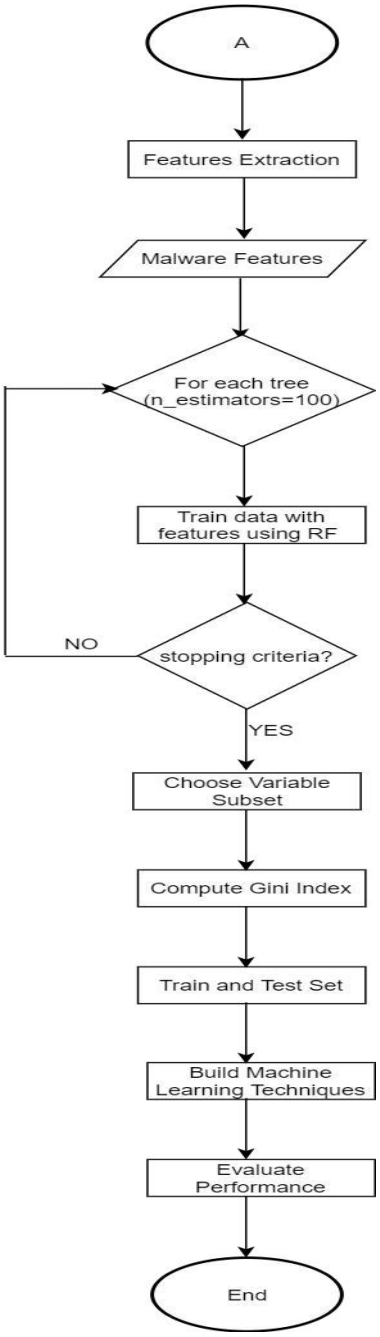


**Figure 2 - Polymorphic malware detection based on feature selection.**

Source : data from research (2019)

## IV. IMPLEMENTATION

As discussed in the previous section, the feature extraction process is very important.

### *I. Implementation of feature extraction*

Figure 3 shows some of data set feature was extracted. This data was extracted from the reports generated based on PE file format. The column shows the features of data set while the row column shows the value of each feature**.**

 

| | Name | TimeDateStamp | PointerToSymbolTa | SizeOfOptionalHeac | Characteristics |
|---|---|---|---|---|---|
| | 19611 unique values | 0 4.29b | 0 4.28b | 144 328 | 2 41.4k |
| 1 | VirusShare_a878ba26000edaac5c98eff4432723b3 | 1236512358 | 0 | 240 | 34 |
| 2 | VirusShare_ef9130570fddc174b312b2047f5f4cf0 | 1365109591 | 0 | 224 | 258 |
| 3 | VirusShare_ef84cdeba22be72a69b198213dada81a | 1438777028 | 0 | 224 | 14 |
| 4 | VirusShare_6bf3608e60ebc16cbcff6ed5467d469e | 1354629311 | 0 | 224 | 783 |
| 5 | VirusShare_2cc94d952b2efb13c7d6bbe0dd59d3fb | 1386631250 | 0 | 224 | 783 |
| 6 | VirusShare_eff7676f69be2b519f3424def92d3590 | 708992537 | 0 | 224 | 33167 |

**Figure 3 - Feature extraction process**

Source : data from research (2019)

Source: data from research (2019)

For the first phase, all the features (26 features) were extracted. Table 2 shows the total of 26 features were extracted from PE file format.

**Table 2 - Feature extraction**

| Feature Extraction |
|---|
| 'TimeDateStamp','SizeOfOptionalHeaders','Characteristics','MajorLinkVersion','MinorLinkVersion','SizeOfCode','BaseOfCode','ImageBase','SectionAlignmnet','FileAlignmnet','CheckSum','SizeImage','Subsysytem','DLL Characteristic','LoadFlags','machine','NumberOfSections','MajorOperatingSystemVersion','MinorOperatingSystemVersion','MajorSubsytemVersion','MInorSubsystemVersion','ecp','emaxalloc','PointerToSymbolTable','NumberOfSymbols','SizeImage' |

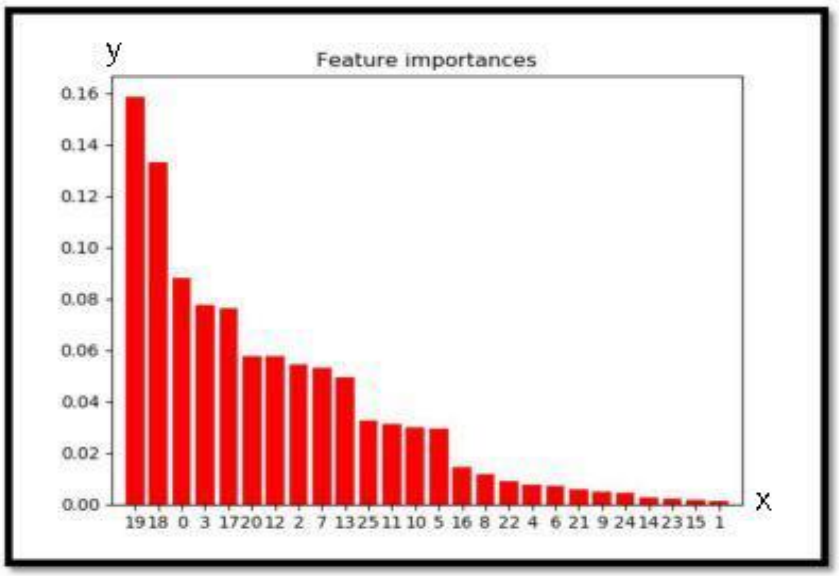


**Figure 4 - Feature important graph**

Source : data research from (2019)

Based on the graph in Figure 4 shows the feature importances used in this research. Label x is a variable and label y is a index value each of variable.The plot shows an index with 19 has the highest significance, and index 1 has the lowest importance feature.

***II. Implementation of algorithm with four machine learning techniques***

In this research, the main part is to focus at detection phase. In this phase, the algorithm was used to determine the file either it is polymorphic malware or clean file . The algorithm combined with four machine learning techniques were able to determine the file either clean file or malware file based on the PE Header value set in this algorithm.

The value of MajorOperatingSystemVersion (2015) is 5 frequently appear in malware files. Same goes to DLLCharacteristic where if DLLCharacteristic are equal to zero then it can be considerd as malware files. Moreover, most types of malware are executable image, while parts of benign software are dynamic-link library (2014) It means that values of characteristics, ImageBase, and Dll characteristics have obvious differences between malware and benign software.

The researchers stated that TimeDateStamp (2017) or known as file creation year is very important and useful to identify malware and benign program. Mostly malware has very suspicious year of creation such as years earlier than 1980 or year beyond the current year while benign program has genuine year of creation.1980–2015.According to Symantec report Win32.Sality polymorphic malware was created in 2003. So, in this study if the TimeDateStamp of files more than 2003 then 1 is assigned as feature value indicating that the sample has suspicious malware.

In this research, based on Figure 5 it involved two phases of important features selection. The first phase is to use the entire feature extracted combined with four machine learning techniques. Then continued with the selection of the important feature after done load the dataset. At this stage of feature selection, there are three basic methods (2019) of feature selection which are Filters methods are a preprocessing step that is independent of a subsequent learning algorithms. The researchers used independent techniques to select features. The set of features were chosen by an evaluation criterion, or a score to assess the degree of relevance of each characteristics to a target variable .

Wrappers are feature selection methods that evaluate a subset of characteristics by the accuracy of a predictive model trained along with them. The evaluation is done using a classifier that estimates the relevance of a given subset of characteristics. This type of method proved to be efficient yet computationally expensive which makes it unpopular.

Last but not least, Embedded method which combine the qualities of filter and wrapper methods. As the Filter methods shown to be faster yet not very efficient while the Wrapper methods are more effective but very computationally expensive especially with big datasets, a solution that combines the advantages of both methods were needed. In this research, the researchers used embedded method to contribute malware classification. Embedded method has been chosen because of it highly accurate, generalized better and interpretable.

```
Start

Read File ()

If DLLCharactrsitic = = 0
If MajorOperatingSystemVersion = = 5
If ImageBase = = 4194304
If TimeDateStamp = = 2003
{
return malware
}
Else {
return benign
}
end If

LoadDataset
Feature Selection : X for attribute and y for target
Split Data into training and testing set : train size = 0.8 , test size = 0.2
Evaluation: TP . FP , TN , FN

End
```

**Figure 5 - Polymorphic malware detection combined with for machine learning**

Source : data research from (2019)

### *III. Application of machine learning methods*

After the features were extracted and selected, then the machine learning methods were applied to the data that obtained. The machine learning methods applied, as discussed previously, are Random Forest , K-Nearest Neighbour, Decision Tree and Logistic Regression.

## V. RESULT AND DISCUSSION

In this works, by using machine learning algorithm the performance of polymorphic malware was evaluated before and after the used of feature selection approach. The accuracy of detection is measured as the percentage of correctly identified instances (2010) :

True Positive (TP) : The cases when the actual class of the data and the predicted is also True.
True Negative (TN) : The cases when the actual class of the data and the predicted is also False.
False Positive (FP) : The cases when the actual class of the data was False and the actual was True.
False Negative (FN) : The cases when the actual class of data was True and predicted was False.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FN+FP} \quad (1)$$

### *I. Result with 26 features*

From the experiment, based on Figure 6 and 7 , Random Forest algorithm shows the highest classification performance accuracy which is 98% and the lowest accuracy is Logistic Regression algorithm. Table 3 shows the summarized of each machine learning techniques with 26 total numbers of features. The dataset 0 refers as unknown malware, 1 is known malware and 2 is polymorphic malware.

```
In [31]: #Import scikit-learn metrics module for accuracy calculation^M
    ...: from sklearn import metrics^M
    ...: # Model Accuracy, how often is the classifier correct?^M
    ...: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
    ...:
Accuracy: 0.975609756097561
```

**Figure 6 - Random Foresct Accuracy**
Source : data reserach from (2019)

```
In [43]: from sklearn.linear_model import LogisticRegression^M
    ...: logreg = LogisticRegression()^M
    ...: logreg.fit(X_train, y_train)^M
    ...: print('Accuracy of Logistic regression classifier on training set: {:.
    ...: 2f}'^M
    ...:       .format(logreg.score(X_train, y_train)))^M
    ...: print('Accuracy of Logistic regression classifier on test set: {:.2f}'
    ...: ^M
    ...:       .format(logreg.score(X_test, y_test)))
    ...:
C:\Users\user\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Ple
ase change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Accuracy of Logistic regression classifier on training set: 0.78
Accuracy of Logistic regression classifier on test set: 0.84
```

Figure 7- Logistic Regression Accuracy
Source : data research from (2019)

In this method, each models used in this study gave different results. Before feature selection was used, the lowest accuracy was Logistic Regression (84%) , followed by Decision Tree and K-Nearest Neighbours (92% and 95% respectively). The highest accuracy was Random forest models which equal to 98% compare to the other models. After the feature selection technique was used, all the classifiers were tested and it showed the improvement on the performance accuracy detection. The highest accuracy of classifier is Random Forest which is 99% of detection followed by Decision Tree 93%, K- Nearest Neighbours 94% and the lowest is Logistic Regression 88%.

**Table 3 - Comparison of four machine learning**

| Technique | Comparison Of Machine Learning Accuracy | | | |
|---|---|---|---|---|
| | *Dataset* | *Precision (%)* | *F1-Score (%)* | *Accuracy (%)* |
| Random forest | 0<br>1<br>2 | 96<br>98<br>100 | 96<br>99<br>40 | 98 |
| Decision Tree | 0<br>1<br>2 | 81<br>97<br>17 | 86<br>95<br>20 | 92 |
| K-NN | 0<br>1<br>2 | 83<br>97<br>100 | 93<br>97<br>25 | 95 |
| Logistic Regression | 0<br>1<br>2 | 48<br>89<br>0 | 44<br>91<br>0 | 84 |

Source : data research from (2019)

### II. Result with 10 features

```
In [141]: from sklearn.ensemble import RandomForestClassifier
    ...:
    ...: #Create a Gaussian Classifier
    ...: clf=RandomForestClassifier(n_estimators=200)
    ...:
    ...: #Train the model using the training sets y_pred=clf.predict(X_test)
    ...: clf.fit(X_train,y_train)
    ...:
    ...: y_pred=clf.predict(X_test)
    ...:
C:\Users\user\Anaconda3\Scripts\ipython:7: DataConversionWarning: A column-vecto
r y was passed when a 1d array was expected. Please change the shape of y to (n_
samples,), for example using ravel().

In [142]: #Import scikit-learn metrics module for accuracy calculation
    ...: from sklearn import metrics
    ...: # Model Accuracy, how often is the classifier correct?
    ...: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
    ...:
Accuracy: 0.9853658536585366
```

**Figure 8 - Random Forest Accuracy**
Source : data research from (2019)

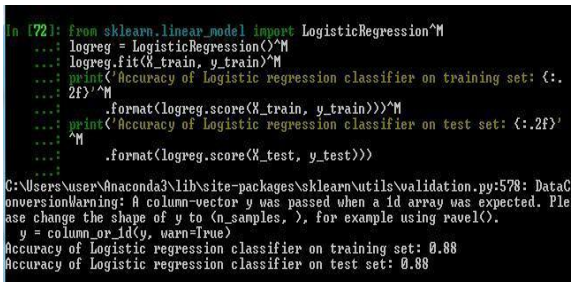


**Figure 9- Logistic Regression Accuracy**

Source : data research from (2019)

Based on Figure 8, the accuracy of Random Forest classifier is 99% after the implementation with 10 features importances while Logistic Regression has achieved 88% of accuracy based on Figure 9.

**Table 5 - New accuracy of classifier after used feature selection method**

| **Technique** | **Comparison Of Machine Learning Accuracy** | | | |
|---|---|---|---|---|
| | ***Dataset*** | ***Precision(%)*** | ***F1-Score (%)*** | ***Accuracy (%)*** |
| Random forest | 0<br>1<br>2 | 97<br>100<br>80 | 96<br>100<br>84 | 99 |
| Decision Tree | 0<br>1<br>2 | 95<br>95<br>20 | 92<br>96<br>0 | 93 |
| K-NN | 0<br>1<br>2 | 95<br>93<br>0 | 91<br>96<br>0 | 94 |
| Logistic Regression | 0<br>1<br>2 | 82<br>89<br>0 | 75<br>92<br>0 | 88 |

Source : data research from (2019)

**Table 6 - Comparison accuracy of total features**

| **Feature Selection** | **Comparison Of Machine Learning Accuracy (%)** | | | |
|---|---|---|---|---|
| | ***Random Forest*** | ***Decision Tree*** | ***Logistic Regression*** | ***K-Nearest Neighbour*** |
| Before Feature Selection | 98 | 92 | 84 | 95 |
| Feature Importances | 99 | 93 | 88 | 94 |

Source : data research from (2019)

From the Table 5 and Table 6 the accuracy between four machine learning algorithms shows that by using 10 attributes of features extraction, the accuracy of all classifiers shows the improvement accuracy compared to use with 26 attributes to detect polymorphic malware.

In this research it is important to implement feature selection methods to improve the accuracy of the dataset. The accuracy of the classifier not only depends on the classification algorithm but also on the feature selection method.

## VI. CONCLUSION

In this research is determined to discriminate polymorphic malware feature based on feature selection. This phase is very important stage to determine the best features because not all the extracted features gave benefit in machine learning classifiers. From this research, the feature selection gathered based on PE32 header format information it also demonstrates the best accuracy range between 90% to 100% detection. It also shows that this identification could be performed efficiently and quickly.

Furthermore, this feature selection approach can also increase the detection rate accuracy of malware. Moreover, in this research, three different datasets were used to classify malware types; using supervised machine learning to identify unknown files , benign files and polymorphic malware. Even the datasets used were imbalanced but the result showed each of the models performed to classify malware files based on accuracy detection. This experiment showed that Random Forest models seems to provide the most accurate classification for this data compared to other models. From the experimental results, the Random Forest showed the highest overall accuracy detection which it obtained the highest scores on the percentage of correctly classified instances at level 99% while the Decision Tree classifier, K-NN, logistic regression, the classification accuracy was lower.

In a nutshell, in this research features selection approach are important to improve the detection accuracy of polymorphic malware.

## ACKNOWLEDGMENT

The author hereby acknowledges the financial support from Faculty Computer and Mathematical Sciences, University Technology MARA (UiTM) Shah Alam.

## BIBLIOGRAPHY

1. DHAMMI, A., & SINGH, M. (2015). Behavior analysis of malware using machine learning. 2015 Eighth International Conference on Contemporary Computing (IC3). Published. https://doi.org/10.1109/ic3.2015.7346730
2. YUSOF, M. A., ALI, F. H., & DARUS, M. Y. (2018). Detection and defense algorithms of different types of DDoS attacks. International Journal of Engineering and Technology, 9(5), 410-444. doi:10.7763/ijet.2017.v9.1008
3. BELAOUED, M., & MAZOUZI, S. (2016). A chi-square-based decision for real-time malware detection using PE-file features. Journal of Information Processing Systems, 12(4), 644–660. http://jips-k.org/digital-library/22736
4. BIONDI, F., ENESCU, M. A., GIVEN-WILSON, T., LEGAY, A., NOUREDDINE, L., & VERMA, V. (2019). Effective, efficient, and robust packing detection and classification. Computers & Security, 85, 436–451. https://doi.org/10.1016/j.cose.2019.05.007
5. BAI, J., WANG, J., & ZOU, G. (2014). A malware detection scheme based on mining format information. The Scientific World Journal, 2014, 1–11. https://doi.org/10.1155/2014/260905
6. JALIL, K. A., KAMARUDIN, M. H., & MASREK, M. N. (2010). Comparison of machine learning algorithms performance in detecting network intrusion. 2010 International Conference on Networking and Information Technology. doi:10.1109/icnit.2010.5508526
7. KUMAR, B. J., NAVEEN, H., KUMAR, B. P., SHARMA, S. S., & VILLEGAS, J. (2017). Logistic regression for polymorphic malware detection using ANOVA F-test. 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). Published. https://doi.org/10.1109/iciiecs.2017.8275880
8. SHIJO, P., & SALIM, A. (2015). Integrated static and dynamic analysis for malware detection. Procedia Computer Science, 46, 804–811. https://doi.org/10.1016/j.procs.2015.02.149
9. SHAMSUDDIN, M. R., ALI, F. H., & ABIDIN, M. S. (2020). Transforming malware behavioural dataset for deep denoising autoencoders. IOP Conference Series: Materials Science and Engineering, 769, 012071. doi:10.1088/1757-899x/769/1/012071
10. HUSSEIN, S. M., ALI, F. H. M., & KASIRAN, Z. (2012). Evaluation effectiveness of hybrid IDS using Snort with Naïve Bayes to detect attacks. 2012 Second International Conference on Digital Information and Communication Technology and It's Applications (DICTAP). Published. https://doi.org/10.1109/dictap.2012.6215386
11. SARI, M.S.A.M & MAAROF, M. A. (2017). Classification of malware family using decision tree algorithm.UTM Computing Proceedings Innovations in Computing Technology and Applications Volume 2 , Year: 2017 , ISBN: 978-967-0194-95-0 https://doi:10.18411/a-2017-023.2017
12. SINGLA, S., GANDOTRA, E., BANSAL, D., & SOFAT, S. (2015). Detecting and classifying morphed

malwares: A survey. International Journal of Computer Applications, 122(10), 28–33. https://doi.org/10.5120/21738-4937

13. TADIST, K., NAJAH, S., NIKOLOV, N. S., MRABTI, F., & ZAHI, A. (2019). Feature selection methods and genomic big data: A systematic review. Journal of Big Data, 6(1). https://doi.org/10.1186/s40537-019-0241-0
14. TAJODDIN, A., & JALILI, S. (2018). HM3alD: Polymorphic malware detection using program behavior-aware Hidden Markov Model. Applied Sciences, 8(7), 1044. https://doi.org/10.3390/app8071044.