Trojan Detection System Using Machine Learning Approach

M I Jaya[1], M F A Razak*[2], Z Ismail[3], A Firdaus[4]

[1-4] Faculty of Computing, University Malaysia Pahang, 26600 Pekan, Pahang, Malaysia

E-mail: faizalrazak@ump.edu.my[2]

Submitted: 8 March 2022, revised: 18 August 2022, accepted: 24 August 2022

Abstract. Malware attack cases continue to rise in our current day. The Trojan attack, which may be extremely destructive by unlawfully controlling other users' computers in order to steal their data. As a result, Trojan horse detection is essential to identify the Trojan and limit Trojan attacks. In this study, we proposed a Trojan detection system that employed machine learning algorithms to detect Trojan horses within the system. A public dataset of Trojan horses that contain 2001 samples comprises of 1041 Trojan horses and 960 of benign is used to train the machine learning classification. In this paper, the Trojan detection system is trained using four types of classifiers which are Random Forest, J48, Decision Table and Naïve Bayes. WEKA is used for the execution of the classification process and performance analysis. The results indicated that the detection system trained with the Random Forest and Decision Table algorithms obtained the maximum level of accuracy with 100%.

Keywords: Malware; Trojan; Machine Learning; Classification; Artificial Intelligence

1. Introduction

A Trojan horse is a sort of malware that disguises itself as a genuine software and installs onto a computer [1]. This harmful code can be hidden by an attacker in any legitimate software, and it is typically hidden in a seemingly innocuous email attachment or free program download. Trojan horses, unlike worms and viruses, do not self-replicate; instead, they require a genuine user to install the application without recognising the presence of the Trojan [2]. The implanted application functions as a software that can be managed remotely, allowing attackers to work on their victim's computer from a distant location. As a result, once the Trojans are installed on the user's PC, the attackers can steal personal information, passwords, and other confidential data.

Before a Trojan horse to infect a computer, the user must download and install the server-side of the malicious program onto their computer. When a computer is infected, the user is unaware that it is being administered by unauthorised individuals. These devices are used by attackers to propagate malware and establish an infected network. Even worse, the Trojan horse may be used to record keyboard operations in order to gather bank account and password information in the user's machine [3]. The Trojan horse has the ability to infect the host system with harmful infections and can endanger laptops as well as desktop computers. Trojan horses, on the other hand, may be deployed as mobile malware to attack mobile devices

such as smartphones and tablets. Because of this infection, an attacker might redirect traffic on WI-FI-connected devices.

A detection model developed using a machine learning algorithm can detect a Trojan horse. Machine learning is a fast-growing field of computing algorithms that aim to mimic human intelligence by learning from their surroundings. Machine learning also concerned with the creation of systems that can access and analyse data in order to understand how it behaves [4]. Its primary goal is to enable computers to learn on their own, without human interaction, and to adjust their behaviour accordingly [5]. Data collected by the user platform, for example, will be added, processed, and analysed using a machine learning algorithm to provide its insight. Data may be processed in any format, including numerical and textual data. Machine learning also contributes to the automation and quick development of data analysis models. The models that have been constructed are capable of processing and analysing massive amounts of complex data in order to produce reliable findings. These models are accurate and scalable, and they run in less time.

In this paper, we aimed to develop a Trojan detection system utilising machine learning approach. First, a literature review on the available methods for malware detection is conducted. Then, public datasets related to the network traffic and contains Trojan horses' as well as benign example is gathered to train the model. Before the model training phase, preprocessing phase will be conducted to reduce the dataset size and select the effective features that will be utilised by the machine learning classifier. WEKA tool is used during the experiment and its machine learning classification algorithm is utilized to detect the Trojan horses. The machine learning classification algorithms that are used in the experiment is Naïve Bayes, Random Forest, J48 and Decision Table. In addition, we analyse the performance of each classifier in terms of their accuracy in categorizing Trojan horse and benign data. Finally, we determine which machine learning approach is feasible to detect Trojan horse efficiently.

2. Related works

There have been various studies on Trojan horse detection approaches, including Classification Approaches [6], Gate-Level Netlists Detection [7], Real-Time Detection [8], Golden Model-Free [9], and Reverse Engineering Improvement [10]. In this paper, Classification Approaches has been chosen as the machine learning approach to identify the Trojan horse within the network. As for that, three algorithms that are related to the classification approach is analysed in this section.

The research in [11] demonstrates that Naïve Bayes Algorithms may be utilised to distinguish between Trojan and benign data samples. For further robustness, the technique employs Levenshtein distance. During the training phase, a database is established to hold all unique API calls discovered in the dataset of Trojan and benign samples. API calls are chosen based on their likelihood of being related with malicious activity. During the testing phase, the classifier calculated the probability score of each tested sample and categorised it as Trojan or benign based on a predefined threshold. The proposed technique achieved higher accuracy and outperformed commercial anti-malware solutions, particularly when dealing with huge malware samples [11]. This classifier has the benefit of being easy to construct and capable of rapidly categorising the dataset to generate the prediction results. It can also handle many prediction classes and large datasets. Furthermore, if the independence of each feature is proven, this classifier is regarded as the best alternative since it performs better. Regrettably, it only works effectively with categorical input variables. If there are numerical attributes, it will use a normal distribution to categorise the number, which may impair the accuracy of its predictions. Another disadvantage of this classifier is that it is very hard to assume that all the attributes are independent of one another.

Trojan detection using decision tree classification is described in [12]. Several techniques, including J48 Decision Tree, Random Tree, and Random Forest, can be used to categorise the dataset as benign or malicious. Thresholding attribute selection can also be coupled with decision tree-based classifiers to improve Trojan detection performance. Several performance measures, such as the percentage of accuracy, the receiver operating characteristics (ROC) curve, and the comparison of F-Measure, may be used to assess

and compare the classifiers' performance [12]. The key advantage of adopting a decision tree-based approach is that it requires less effort for data preparation during pre-processing than other techniques since it does not require scaling and normalisation of data since a messy dataset does not impact the process of decision tree model development. However, as a little change in the input might induce a huge change in the structure of the decision tree, resulting in instability, the computing complexity may increase. This classification may eventually require more time based on number of input.

Research in [13] discussed that malware detection could perform well by employing random projections and neural networks algorithms. Specifically, seven methods can be used to improve the malware detection performance, which are Logistic Regression All Features, Logistic Regression Random Projections, One-Layer Neural Network with and without Pre-training, Two-Layer Neural Network with and without Pre-training, and lastly Three-Layer Neural Network without Pre-training. As for the pre-training, Gaussian-Bernoulli restricted Boltzmann machine (RBM) is implemented. That is because, the input is no longer binary after the random projection stage. They can determine which sort of classifier is best for various malware classification approaches by testing with several types of classifiers. Instead of random projections, principal component analysis (PCA) can be used to reduce the input vector's dimensionality [13]. As a result, massive malware classification schemes can use random projection to reduce the input size, allowing a more complicated supervised classification method to be implemented. Unnecessary increasing hidden layers may cause increase the complexity.

3. Research Method

Figure 1 depicts the general phases in the methodology utilised to construct the Trojan horse detection system, which began with data collecting on the website. The obtained data is subsequently preprocessed, and the selected feature is identified. Finally, a different type of machine learning classification algorithm is used, and the results are evaluated in terms of performance.

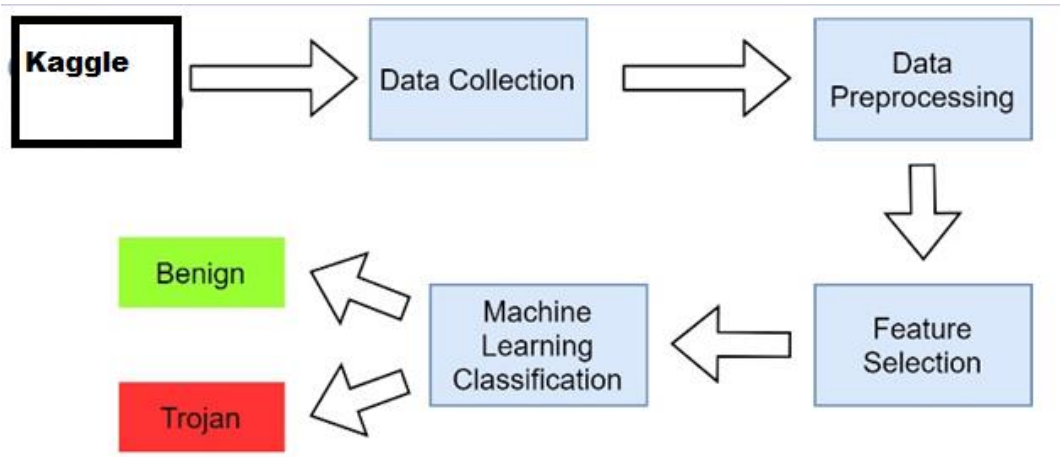

Figure 1. Components of Methodology for Trojan Horse Detection

3.1. Data collection

A public dataset named ‘Trojan Detection’ is gathered from the Kaggle website. It is owned by a user named ‘Cyber Cop’, originated from Canadian Institute for Cybersecurity (CIC), and licensed by GNU Affero General Public License 3.0. It was created in 2021-09-18. The dataset is accessible in a comma-separated values (CSV) file that contains 85 attributes about the records of the network traffic of Trojan Horse and Benign, as well as 177,482 rows data observation.

3.2. Data preprocessing

The Trojan Detection dataset used for analysis consisted of 177,482 samples. These numbers of samples are reduced to 2,001 data samples. Large numbers of samples require high computational memory, before the machine successfully learns how to classify the Trojan and benign. In addition, the 2,001 samples are further examined to achieve nearly balanced data of class Trojan and benign as imbalanced data may cause bias in the classifier performance.

3.3. Feature selection

Feature selection algorithms were used to detect and remove extraneous and redundant attributes from the data, as well as features that contribute less to the accuracy of the prediction model. As a consequence, the total number of features has been lowered from 85 to 35. The performance of machine learning able to degrade when there are many features. By using scoring methods to select relevant features and removed irrelevant features. Those selected features will be trained and tested using machine learning classifier algorithms. Table 1 shows the list of the selected features that is used in this paper.

Table 1. Attributes used in Trojan detection system

Features	Description
Protocol	Internet Protocol Number
Flow Duration	The duration of the packets sent from the source to destination
Total Fwd Packets	Total number of forwarded packets
Total Backward Packets	Total number of backward packets
Total Length of Fwd Packets	Size of forwarded packets in bytes
Total Length of Bwd Packets	Size of backward packets in bytes
Fwd Packet Length Mean	Mean size of forwarded packets in bytes
Bwd Packet Length Mean	Mean size of backward packets in bytes
Flow Bytes/s	The bytes flow in a second
Flow Packets/s	The number of packets transferred within in a second
Flow IAT Mean	The mean of bytes from the flow Information Access Technology
Fwd IAT Total	The total number of forward Information Access Technology in bytes
Fwd IAT Mean	The mean number of forward Information Access Technology in bytes
Bwd IAT Total	The total number of backward Information Access Technology in bytes
Bwd IAT Mean	The mean number of backward Information Access Technology in bytes
Fwd Packets/s	The number of forwarded packets within a second
Bwd Packets/s	The number of backward packets within a second
Packet Length Mean	The mean size of packets in flow
FIN Flag Count	The amount of finish flag
SYN Flag Count	The amount of synchronisation flag
RST Flag Count	The amount of TCP reset flag
PSH Flag Count	The amount of push flag
ACK Flag Count	The amount of acknowledgement flag
URG Flag Count	The amount of urgent flag
CWE Flag Count	The amount of common weakness numeration flag
ECE Flag Count	The amount of ECN-Echo (ECE) flag

Features	Description
Down/Up Ratio	Packet loss ratio
Average Packet Size	Mean packet size
Avg Fwd Segment Size	Mean forward segment size
Avg Bwd Segment Size	Mean backward segment size
Subflow Fwd Packets	Mean forward packets in a sub flow
Subflow Fwd Bytes	Mean forward bytes in a sub flow
Subflow Bwd Packets	Mean backward packets in a sub flow
Subflow Bwd Bytes	Mean backward bytes in a sub flow
Class	Class of traffic

3.4. Machine Learning Classifier Selection

In this research, the classification process is carried out using the Waikato Environment for Knowledge Analysis (WEKA) tools. It is a free and open-source software distributed under the GNU General Public License that includes many machine learning classification techniques for detecting and classifying datasets with Trojan. The classification techniques used in this work are explained further in the following.

3.4.1. Naïve Bayes

Naïve Bayes algorithm is a machine learning classification technique that is based on the Bayes' Theorem, which has a great assumption of freedom among each attribute or condition [14]. The working principle of the Naïve Bayes algorithm is to create a frequency table from the dataset provided first and create a prediction table by calculating the probability of different outcomes. The probability for each class is then calculated using Bayes' Theorem. Finally, the class with the highest probability will be the forecast outcome.

Equation 1 shows the implementation of Bayes formula in Naïve Bayer Classifier in machine learning.

$$P(\mathrm{A}|\mathrm{B}) = P(\mathrm{A})P(\mathrm{B}|\mathrm{A})\ P(\mathrm{B}) \qquad \textit{Equation (1)}$$

Where:
B: Selected features
A: Target category
P(A|B): probability of condition A to happen given B has happened
P(B|A): probability of condition B to happen given A has happened
P(A): probability of condition A to happen
P(B): probability of condition B to happen

3.4.2. Random Forest

The random forest algorithm generally consists of three main steps as depicted in Figure 2. First, several random vectors are created to apply bootstrap and random attribute decisions [15]. Then, using random vectors, several decision trees is constructed, and bootstrap different observations. The candidate splitting variables for each tree are chosen at random from the entire set of explanatory variables. Splitting continues until each tree reaches its maximum level. Following the estimation of base tree models, the outcome is obtained using the majority voting method. The overall expected result is the one predicted by the ensemble the most.

3.4.3. J48

J48 is an improved version of C4.5 algorithms. It is a useful decision tree technique for dealing with imbalanced data if certain of its properties are suitably adjusted [16]. J48 is made up of three parts: root

node, internal node, and leaf node. The root node contains the test condition for different features, the branch nodes represent all possible results in the test, and the leaf nodes carry the target class.

3.4.4. Decision table

Decision Table algorithm is a representation that consists of two parts: a schema, which is a collection of table-included attributes, and a body, which is a collection of labelled categories from the space represented by the schema's features [17]. A decision table classifier examines the decision table for precise matches using just the characteristics in the schema when given an unlabeled classis, notice that there may be several matching examples in the table. The majority class of the Decision Table is returned if no classification is discovered; otherwise, the majority class of all matched instances is returned.

4. Result and Discussion

Since the dataset collected has been labeled with their class (Trojan, Benign), the supervised machine learning method is used to reduce the misclassification and generate better results. This section discusses and analyse the result of the Trojan detection system using different classification algorithms in terms of its accuracy, false-positive rate (FPR), precision and recall, as well as F-Measure to determine the performance of each classifier. The outcomes of conducted experiments in this paper are shown in Table 2.

Table 2. Performance of Classifiers

Classifier	Accuracy (%)	FPR	Precision	Recall	F-Measure
Naïve Bayes	88.21	0.121	0.884	0.882	0.882
Random Forest	100	0.000	1.000	1.000	1.000
J48	99.95	0.000	1.000	1.000	1.000
Decision Table	100	0.000	1.000	1.000	1.000

Table 2 shows that Random Forest and Decision Table is the best classifier to use for detecting Trojan horses. Both classifiers have 100% accuracy in detecting Trojan horses. J48 algorithms also have a great result of the accuracy of 99.95%. As compared to the others, the Naïve Bayes algorithm has low accuracy which is 88.2%, due to the high number of the input variables are not categorical data type. Numeric input variables may affect the accuracy of the result. It also shows that feature selection plays an important role in determining the efficiency of Trojan horse detection based on the high precision rate shown by the classifiers. A high precision rate represents that the classifiers can produce more relevant results and producing accurate results. Graph in Figure 3 shows the summary of the accuracy of different classifiers.

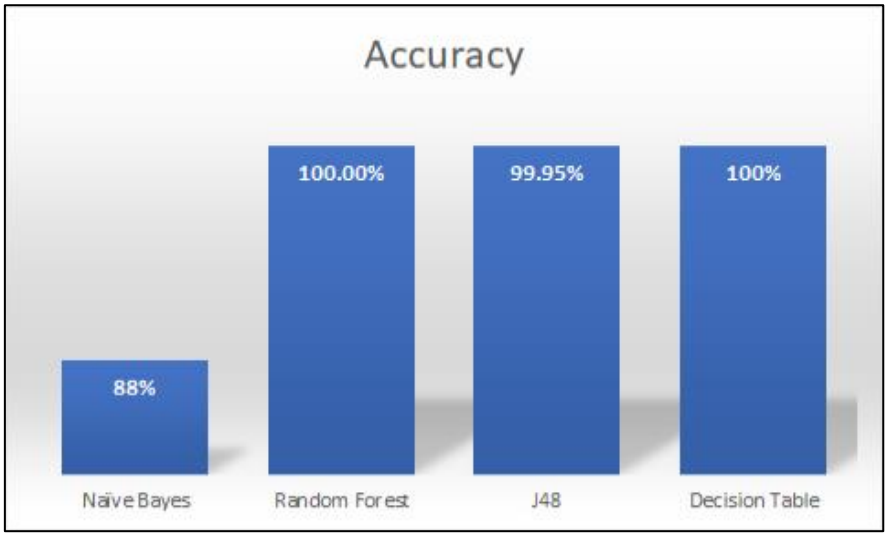

Figure 3. Accuracy of Supervised Classifiers

The confusion matrix is a method for evaluating a classification algorithm's performance. Table 3 depicts the possible classifier prediction of Trojan, and benign data example. Table 3 also summarised that Random Forest and Decision Table techniques achieved the best outcomes in predicting Trojan horse with 1,041 accurate predictions. Meanwhile, the outcomes for prediction of benign is 960 accurate predictions from the Random Forest, J48, and Decision Table techniques respectively. Based on the result, we are able to determine that Random Forest and Decision Table techniques are the most accurate classifier for Trojan horse identification.

Table 3. Confusion Matrix of Classifiers

Classifier	Class	Prediction	
		Trojan	Benign
Naïve Bayes	Actual Trojan	962	79
	Actual Benign	157	803
Random Forest	Actual Trojan	1,041	0
	Actual Benign	0	960
J48	Actual Trojan	1,040	1
	Actual Benign	0	960
Decision Table	Actual Trojan	1,041	0
	Actual Benign	0	960

To further examine the capability of detecting Trojan horses using machine learning approaches, the ROC curve graphs are visualized for each classifier. ROC curve is a graph that depicts a classification model's performance overall categorization levels [18]. This graph depicts two parameters: The rate of True Positives (TPR) on the y-axis and False Positives Rate (FPR) on the x-axis. Figure 4 pictured the ROC curve of Naïve Bayes classifiers in detecting the Trojan horses.

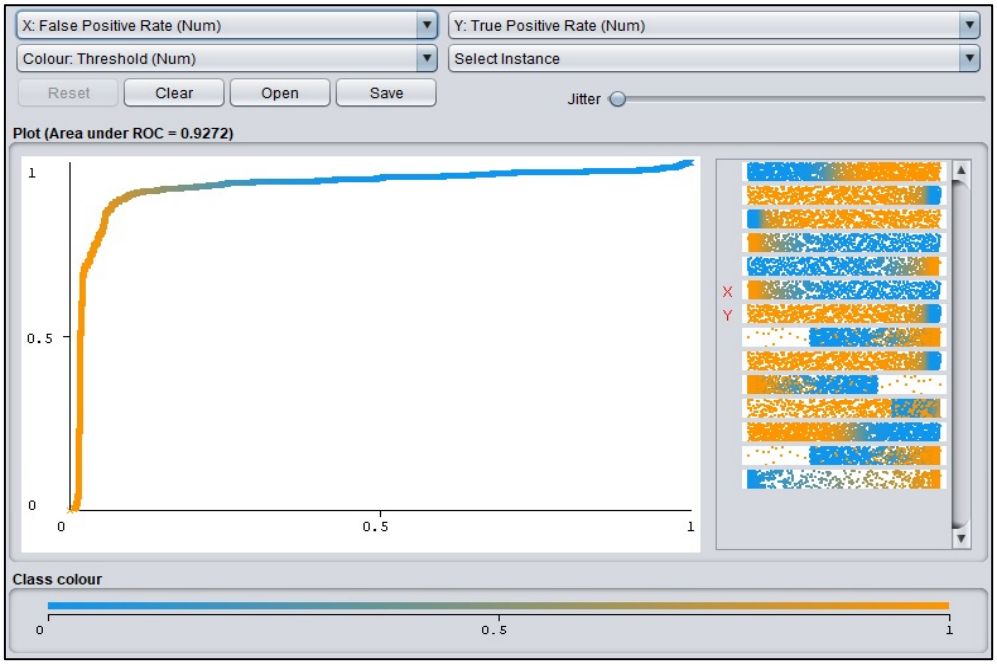

Figure 4. ROC Curve

Since the ROC curve seems equivalent by observing the graph for every classifier, it is hard to compare the accuracy of each classifier. To facilitate this, detection accuracy is measured based on the value of the area under the curve (AUC) [18]. The AUC findings could determine if the detection method was excellent or bad. An area value of 1 indicates a flawless forecast, whereas 0.5 indicates a poor prediction.

Table 4 showed that Random Forest and Decision Table algorithms generate the best result of AUC values, with the perfect value of 1.0000. Other algorithms such as Naïve Bayes and J48 algorithms also provide a decent AUC value indicates that they are not a bad choice in classifying Trojan horse and benign network traffic flow.

Table 4. AUC Value and Indicator of Classifiers

Classifier	Area Under the Curve (AUC)	Indicator
Naïve Bayes	0.9272	Good perfection
Random Forest	1.0000	Perfect Prediction
J48	0.9995	Perfect Prediction
Decision Table	1.0000	Perfect Prediction

5. Conclusion

In summary, this paper has successfully proved that machine learning technique can be used to detect Trojan horse. All the machine learning algorithms used in this research have generated good accuracy in classifying Trojan horse and benign malware. Among those classifiers, Random Forest and Decision Table algorithms provided the best result which is 100.00% perfect prediction of Trojan horse and benign samples, followed by J48 algorithm with 99.95% of accuracy and Naïve Bayes algorithm with 92.72% of accuracy. Therefore,

our proposed method can be implemented in the computer system to help users to identify Trojan horse in the network.

6. Acknowledgement
Great appreciation to the Universiti Malaysia Pahang (UMP) under project ID: RDU210321.

7. References

[1] L. Gang and Y. Wen, "Research on Clue Mining in Criminal Cases of Smart Phone Trojan Horse under the Background of Information Security," *Journal of Robotics*, vol. 2022, pp. 1–11, Feb. 2022, doi: 10.1155/2022/7568110.

[2] F. Zareen and R. Karam, "A Framework for Detecting Hardware Trojans in RTL Using Artificial Immune Systems," *Behavioral Synthesis for Hardware Security*, pp. 265–287, 2022, doi: 10.1007/978-3-030-78841-4_12.

[3] A. Attkan and · Virender Ranga, "Cyber-physical security for IoT networks: a comprehensive review on traditional, blockchain and artificial intelligence based key-security," *Complex & Intelligent Systems 2022*, pp. 1–33, Feb. 2022, doi: 10.1007/S40747-022-00667-Z.

[4] R. Sánchez-Salmerón *et al.*, "Machine learning methods applied to triage in emergency services: A systematic review," *International Emergency Nursing*, vol. 60, p. 101109, Jan. 2022, doi: 10.1016/J.IENJ.2021.101109.

[5] I. Tiddi and S. Schlobach, "Knowledge graphs as tools for explainable machine learning: A survey," *Artificial Intelligence*, vol. 302, p. 103627, Jan. 2022, doi: 10.1016/J.ARTINT.2021.103627.

[6] K. Dushyant, G. Muskan, Annu, A. Gupta, and S. Pramanik, "Utilizing Machine Learning and Deep Learning in Cybesecurity: An Innovative Approach," *Cyber Security and Digital Forensics*, pp. 271–293, Feb. 2022, doi: 10.1002/9781119795667.CH12.

[7] H. Salmani, "Gradual-N-Justification (GNJ) to Reduce False-Positive Hardware Trojan Detection in Gate-Level Netlist," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022, doi: 10.1109/TVLSI.2022.3143349.

[8] N. M. Chayal and N. P. Patel, "Review of Machine Learning and Data Mining Methods to Predict Different Cyberattacks," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 52, pp. 43–51, 2021, doi: 10.1007/978-981-15-4474-3_5.

[9] J. Plusquellic and F. Saqib, "Detecting Hardware Trojans Using Delay Analysis," *The Hardware Trojan War: Attacks, Myths, and Defenses*, pp. 219–267, Nov. 2018, doi: 10.1007/978-3-319-68511-3_10.

[10] C. Bao, D. Forte, and A. Srivastava, "On Reverse Engineering-Based Hardware Trojan Detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 49–57, Jan. 2016, doi: 10.1109/TCAD.2015.2488495.

[11] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic API calls for malware detection," *https://doi.org/10.1080/1206212X.2020.1732641*, vol. 44, no. 3, pp. 270–277, 2020, doi: 10.1080/1206212X.2020.1732641.

[12] M. Hossain, S. Rafi, and S. Hossain, "An Optimized Decision Tree based Android Malware Detection Approach using Machine Learning," *ACM International Conference Proceeding Series*, pp. 117–125, Dec. 2020, doi: 10.1145/3428363.3428375.

[13] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 3422–3426, Oct. 2013, doi: 10.1109/ICASSP.2013.6638293.

[14] S. Harini, A. Ravikumar, and N. Keshwani, "Malware Prediction Analysis Using AI Techniques with the Effective Preprocessing and Dimensionality Reduction," pp. 153–169, 2022, doi: 10.1007/978-981-16-7167-8_12.

[15] S. Jin, Z. Guo, D. Liu, and Y. Yang, "A Study on the Application of Distributed System Technology-Guided Machine Learning in Malware Detection," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–12, Feb. 2022, doi: 10.1155/2022/4977898.

[16] U. Urooj, B. A. S. Al-Rimy, A. Zainal, F. A. Ghaleb, and M. A. Rassam, "Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions," *Applied Sciences 2022, Vol. 12, Page 172*, vol. 12, no. 1, p. 172, Dec. 2021, doi: 10.3390/APP12010172.

[17] D. Singh, S. Karpa, and I. Chawla, "'Emerging Trends in Computational Intelligence to Solve Real-World Problems' Android Malware Detection Using Machine Learning," pp. 329–341, 2022, doi: 10.1007/978-981-16-3071-2_28.

[18] X.-W. Wu, Y. Wang, Y. Fang, and P. Jia, "Embedding vector generation based on function call graph for effective malware detection and classification," *Neural Computing and Applications*, pp. 1–14, Feb. 2022, doi: 10.1007/S00521-021-06808-8/TABLES/11.