

# Using decision tree classifier to detect Trojan Horse based on memory data

Mosleh M. Abualhaj<sup>1</sup>, Sumaya N. Al-Khatib<sup>2</sup>

<sup>1</sup>Department of Networks and Cybersecurity, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan

<sup>2</sup>Department of Computer Science, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan

---

## Article Info

### Article history:

Received Oct 6, 2023

Revised Dec 9, 2023

Accepted Jan 5, 2024

---

### Keywords:

Decision tree

Machine learning

Malware

Trojan Horse

Obfuscated-MalMem2022

---

## ABSTRACT

Trojan Horse is a major threat that has grown with the spread of the digital world. Data gathered through the study of memory can provide valuable insights into the Trojan Horse's behavior patterns. Because of this, memory analysis techniques are one of the topics that should be investigated in Trojan Horse detection. This study proposes the use of memory data in Trojan Horse detection. Trojan Horse detection used a decision tree (DT) classifier with memory data. Experiments were performed on the Trojan Horse samples from the CIC-MalMem-2022 dataset. The binary classification was made using DT, gradient boosted tree, Naive Bayes (NB), linear vector support machine, K-nearest neighbors (KNN), and machine learning (ML) classifiers. The comparison of the various classification methods was performed utilizing the accuracy, recall, precision, and F1-score metrics. As a result, the most successful Trojan Horse detection was gained with the DT classifier, which achieved accuracy of 99.96% using memory data. The NB classifier showed the lowest achievement in Trojan Horse detection using memory data, which achieved accuracy of 98.41%. In addition, numerous of the classifiers utilized have attained very high results. Based on the achieved results, the data from memory analysis is very valuable in detecting Trojan Horse.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



---

## Corresponding Author:

Mosleh M. Abualhaj

Department of Networks and Cybersecurity, Faculty of Information Technology

Al-Ahliyya Amman University

Amman, 19328, Jordan

Email: m.abualhaj@ammanu.edu.jo

---

## 1. INTRODUCTION

Cyberthreats encompass nefarious and potentially perilous actions, plans, or occurrences that exploit vulnerabilities in digital technologies, networks, and systems. Cybercriminals, hackers, or other malicious entities orchestrate and execute these threats to compromise the confidentiality, integrity, availability, or overall security of digital assets, data, or business operations [1]. Data breaches, phishing attacks, identity theft, and malware infections exemplify the wide array of actions falling under the umbrella of cyber threats [2], [3]. Given the swiftly evolving landscape of technology and the digital realm, proactive cybersecurity measures are imperative to curbing risks and shielding against potential malware infections. Contemporary forms of malware include ransomware, keyloggers, rootkits, and Trojan Horses [4]–[6].

Trojan Horse refers to a computer application downloaded and installed on a computer, masquerading as harmless but carrying harmful intent. Unanticipated modifications to computer settings and unusual activities, particularly during periods of inactivity, indicate the presence of Trojan Horse. These deceptive programs are often concealed within seemingly innocuous email attachments or free downloads.

Upon opening such attachments or initiating downloads, the embedded malware is introduced to the user's device, enabling the malicious code to execute the attacker's intended actions [7]–[9]. Incidents involving Trojan Horse attacks have emerged as significant cybersecurity risks, inflicting financial losses, and operational disruptions on individuals and businesses. In 2022 alone, a staggering 5.5 billion malware attacks were documented, with 2.75 billion attributed to Trojan Horse attacks [6]. To safeguard against these threats, a comprehensive approach combining preventive measures and proactive security protocols is essential to prevent the infiltration of these malicious programs into systems [10], [11]. Given Trojans' ability to deceptively mimic benign software, maintaining vigilance, and adopting necessary precautions are paramount. Augmenting defenses against Trojan Horse threats can be accomplished by implementing advanced strategies like sandboxing, honeypots, and machine learning (ML) [12]–[14].

Harnessing the capabilities of artificial intelligence and data analysis, ML emerges as a potent tool to counteract Trojan Horse attacks. This involves detecting and preventing such threats, facilitated by models designed to recognize and thwart these insidious infiltrations. ML driven models can identify discernible patterns and behaviors linked to Trojan Horses, laying the groundwork for the development of proactive security strategies. The utilization of ML in categorizing Trojan Horse necessitates the training of models to autonomously distinguish between benign and malicious software. By learning from meticulously labeled datasets, ML classifiers discern distinctive patterns and attributes characteristic of Trojan Horse behavior. Renowned ML classifiers encompass random forest (RF), logistic regression (LR), support vector machine (SVM), Naive Bayes (NB), K-nearest neighbors (KNN), and decision tree (DT) [15]–[20]. Within the purview of this article, the DT classifier takes the forefront in identifying Trojan Horse occurrences.

#### – Related works

Kulkarni *et al.* [21] have introduced a low-overhead online learning hardware solution for unforeseen attacks on a specialized many-core architecture. The assumption is that memory and processor cores remain secure, with anomalies introduced only through communication exchanges. The training dataset is constructed using the effects of Trojan insertions and hardware feature analysis. AVM, KNN, and the modified balanced winnow algorithms (MBWA) evaluate the effectiveness of detecting unforeseen attacks. To illustrate, a ML model is trained with two types of attacks, and a new attack type is introduced in real-time. The MBWA algorithm demonstrates 5% to 8% higher accuracy in detecting attacks compared to SVM and K-NN. An Attack Insertion module is implemented to test the design with condition-based attacks. The design is fully implemented and routed on the Xilinx Virtex-7 field-programmable gate array (FPGA). Only an additional four cycles are required for the proposed system to detect attacks during operation. Compared to previously published Trojan detection designs, the proposed approach achieves a 56% reduction in area overhead and a 50% decrease in latency.

Worley and Rahman [22] conducted a quantitative assessment comparing the effectiveness of four different supervised ML techniques in classifying integrated circuits based on their ring oscillator network frequencies. Remarkably, when utilizing an SVM classifier, this approach achieved 97.6% accuracy in binary classification, accompanied by an impressively low false positive rate (FPR) of just 7.1%. Additionally, ensemble approaches attained an accuracy of around 88%, demonstrating no instances of false positives. However, despite these encouraging findings, supervised learning methods often need to be more feasible in real-world supply chain contexts. Identifying validated 'golden chips' poses a significant challenge, given the near-impossible task of determining compromised chips at the dataset's assumed scale.

Xuan *et al.* [23] have introduced a hybrid semi-supervised classifier designed to achieve precise detection and classification accuracy for web Trojans while utilizing a limited amount of labeled data. The data utilized in this study is drawn from the web security gateway, primarily due to the greater availability of unlabeled data instead of tagged ones. Their approach entails detecting Web Trojans by combining an autoencoder with a multi-layer feed forward-back propagation (BP) artificial neural network (ANN). Initially, the detection model and the extracted features of the Web Trojan were scrutinized. Subsequently, the robustness of feature extraction was enhanced through unsupervised learning using a stacked denoising autoencoder. Integrating the BP-supervised ANN allowed for fine-tuning the network structure and optimizing the detection model. Compared to the DT and SVM, the proposed approach demonstrated a remarkable accuracy of 91.99%, outperforming the 89.32% of DT and the 91.13% of SVM. Hence, the proposed method unequivocally showcases superior performance against well-established classification techniques.

## 2. METHOD

This section presents the Obfuscated-MalMem2022 dataset that has been used in this paper. Then, the preprocessing operations that have been performed on the used dataset will be discussed. Finally, the DT classifier that will be used to detect the attack will be discussed. Figure 1 shows the operations performed to detect the Trojan Horse attack using the DT classifier.

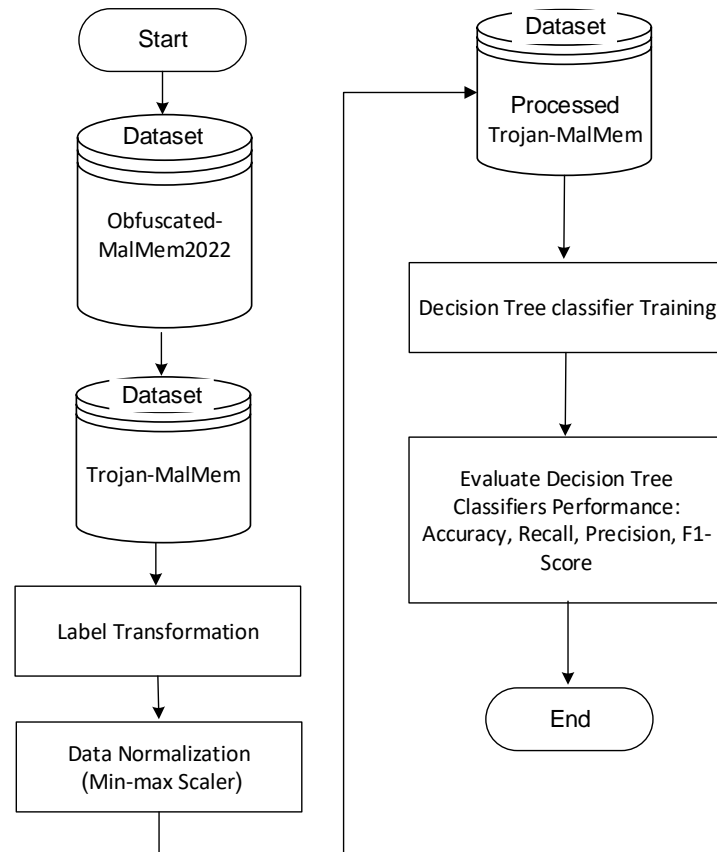


Figure 1. Trojan Horse attack detection

**2.1. Obfuscated-MalMem2022 dataset**

The Obfuscated-MAIMem2022 dataset encompasses three primary malware families: spyware, ransomware, and Trojan Horse. This study, however, focuses primarily on the Trojan Horse category. Consequently, all instances of spyware and ransomware have been excluded, leading to the creation of a refined dataset referred to as Trojan-MalMem. This Trojan-MalMem dataset comprises 9487 entries, categorized across five distinct types of Trojan Horses: Zeus (1950 records), Emotet (1967 records), Refroso (2000 records), Scar (2000 records), and Reconyc (1570 records). The distribution of Trojan Horse types is illustrated in Figure 2. Furthermore, the Trojan-MalMem dataset incorporates an additional 29298 entries of benign data [24].

**Trojan Horse Types Distribution**

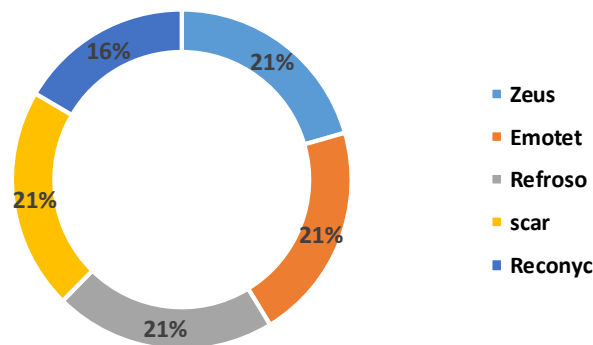


Figure 2. Trojan Horse types distribution

## 2.2. Trojan-MalMem preprocessing

The two primary operations involved in ML data preprocessing encompass data transformation and data normalization. Data transformation involves converting and reformatting data into a suitable format, including transforming text data into numbers using a technique like label encoding. In label encoding, each unique category is assigned a unique integer label. Only the last column in the Trojan-MalMem dataset contains text data [14]. As mentioned above, it contains benign, Zeus, Emotet, Refroso, Scar, and Reconyc values [24]. Therefore, label encoding is used to transform them into 0, 1, 2, 3, 4, and 5, respectively. As for data normalization, it ensures that numerical variables are brought to a similar scale, preventing certain features from dominating the learning process. Min-Max scaling (1) is a data normalization method that scales the data to a specified range, usually between 0 and 1 [14].

$$x_{normalized} = (x - min)/(max - min) \quad (1)$$

Where  $x_{normalized}$  is the normalized value of the original data point  $x$ ,  $x$  is the original data point,  $min$  is the minimum value in the dataset, and  $max$  is the maximum value in the dataset. The Trojan-MalMem dataset contains wide scale of data among features [24]. Therefore,  $min - max$  scaling is used to scale this data between 0 and 1. Tables 1 and 2 show sample of the Trojan-MalMem dataset before and after normalization, respectively. The pre-discussed preprocessing steps, transformation and normalization, collectively serve to enhance the quality of the dataset and facilitate the effectiveness of subsequent ML classifiers.

Table 1. Sample of the Trojan-MalMem dataset before normalization

Data samples	Output
42, 16, 10.73809524, 0, 209.2142857, 1621, 38.5952381, 8787, 209.2142857	0
40, 16, 9.525, 0, 204.175, 1504, 37.6, 8167, 204.175	0
42, 16, 10.02380952, 0, 206.2619048, 1610, 38.33333333, 8663, 206.2619048	0
44, 17, 9.590909091, 0, 200.7954545, 1674, 38.04545455, 8835, 200.7954545	0
45, 17, 10.55555556, 0, 202.8444444, 1694, 38.5, 9129, 212.3023256	1
47, 19, 11.53191489, 0, 242.2340426, 2074, 44.12765957, 11385, 242.2340426	1
40, 14, 14.725, 0, 288.225, 1932, 48.3, 11529, 288.225	1

Table 2. Sample of the Trojan-MalMem dataset after normalization

Data samples	Output
0.091743119, 0.125, 0.602767848, 0, 0.232847424, 0.307725139, 0.682017433, 0.168570919, 0.076448356	0
0.082568807, 0.125, 0.522309316, 0, 0.226113578, 0.257789159, 0.660305073, 0.140899759, 0.068341683	0
0.091743119, 0.125, 0.555392853, 0, 0.228902246, 0.303030303, 0.676303654, 0.163036687, 0.071698876	0
0.100917431, 0.140625, 0.526680736, 0, 0.221597593, 0.330345711, 0.670023219, 0.170713202, 0.062905026	0
0.105504587, 0.140625, 0.590660905, 0, 0.224335597, 0.338881776, 0.679939695, 0.183834687, 0.081416069	1
0.114678899, 0.171875, 0.65541793, 0, 0.276970733, 0.501067008, 0.802714106, 0.284522003, 0.129567064	1
0.082568807, 0.09375, 0.867199276, 0, 0.338427067, 0.440460948, 0.893738914, 0.290948853, 0.203552475	1

## 2.3. Trojan Horse malware detection

The DT classifier is used for Trojan Horse detection. DT is a popular supervised learning classifier used for both classification and regression tasks. It is a tree-like structure where each internal node represents a decision based on a feature, each branch represents an outcome of that decision, and each leaf node represents a class label or a predicted value [14], [25]. Figure 3 clarifies the DT classifier.

In classification tasks, the DT classifier splits the data based on the features to create hierarchical partitions that classify the data into different classes. The classifier selects the best feature to split the data at each internal node using various criteria like Gini impurity or entropy, aiming to maximize the purity of each partition. The Gini impurity and entropy for a given node is calculated using (3) and (2), respectively [26], [27]:

$$Gini\ impurity = 1 - \sum p_i^2 \quad (2)$$

$$Entropy = - \sum (p_i * \log_2(p_i)) \quad (3)$$

Where  $p_i$  is the proportion of data points belonging to class  $i$  in the node.

The construction and utilization of a DT to detect Trojan Horse involve a series of pivotal procedures. Initially, the technique identifies the optimal feature for splitting the Trojan-MalMem dataset into subsets at the root node of the DT, guided by factors such as Gini impurity or entropy. This chosen feature value lay the foundation for creating these subsets of Trojan-MalMem dataset at internal nodes.

Subsequently, the recursive splitting process extends the data partitioning to each subset or child node, progressively crafting the hierarchical structure of the tree. The formulation of leaf nodes finalizes the recursive sequence, encapsulating the ultimate predictions (Trojan Horse or benign). By selectively removing branches or nodes, pruning can be applied to counteract overfitting, enhancing generalization. In the end, the DT is used for prediction; unseen data navigate the tree, choosing branches based on feature values until it reaches a leaf node, enabling the final prediction of Trojan Horse or benign [25]–[28]. These interdependent operations collectively enable the creation of an interpretable decision-making framework, facilitating accurate predictions on new Trojan Horse.

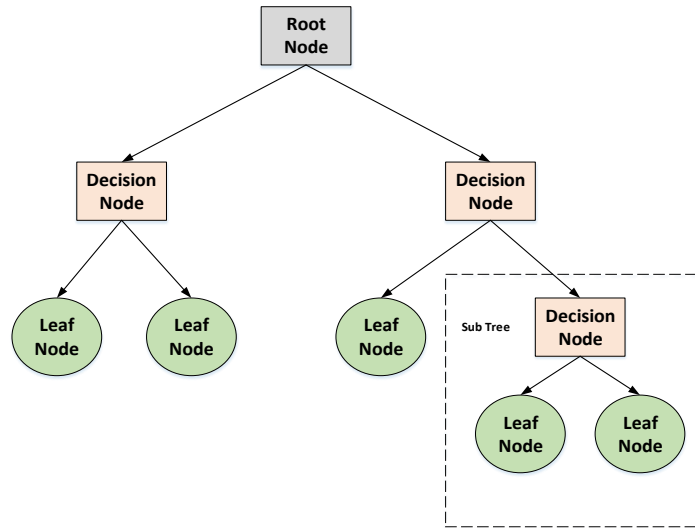


Figure 3. DT classifier

### 3. RESULT AND DISCUSSION

The proposed model was tested on a PC with the following specification: CPU Intel 13 Gen Core i9-13900F 24-Cores up to 5.6 GHz, memory 32 GB, RGB 3200 MHz DDR4 memory graphic card GeForce RTX 4070, 1TB M.2 SSD up to 3500 MB/s, and Ubuntu 20.04.4 LTS O.S. The model is trained and evaluated 5 times using K-fold cross validation method. Evaluating the proposed model involves utilizing various metrics to gauge its performance and effectiveness in making predictions. These metrics provide quantitative insights into how well the model is performing on the given dataset. True positive (TP), true negative (TN), false positive (FP), and false negative (FN) of the confusion matrix was used derive four metrics to evaluate the proposed model. These four metrics are accuracy (4), recall (5), precision (6), and F1-score (7) [14], [25].

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

$$Recall = \frac{TP}{(TP+FN)} \quad (5)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (6)$$

$$F1 - score = 2 \times \frac{Pre \times Rec}{Pre+Rec} \quad (7)$$

Figures 4 to 7 show the accuracy, recall, precision, and F1-score, respectively, of detecting the Trojan Horse when using DT against other common classifiers. DT has achieved the highest value of 99.96% with all metrics among all other classifiers. Conversely, the NB classifier registers the lowest scores for accuracy (98.41%), precision (97.02%), and F1-score (98.42%) among all the classifiers under consideration. Additionally, the linear support vector classifier (SVC) exhibits the lowest recall rate (99.57%) when compared to the other classifiers.

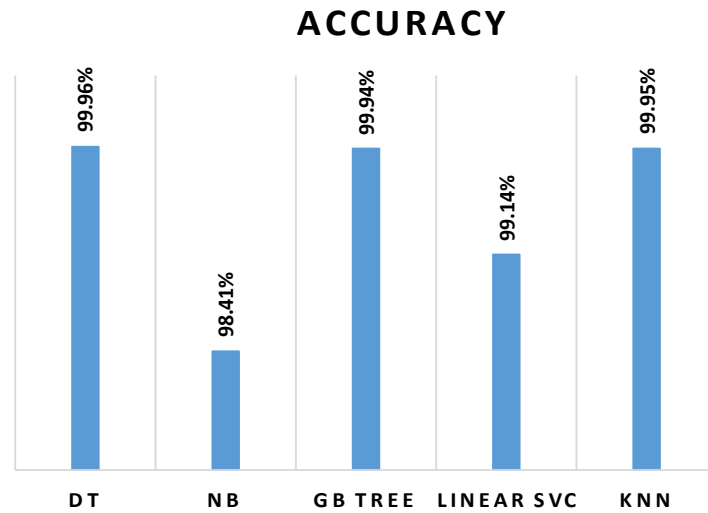


Figure 4. Accuracy of detecting the Trojan Horse

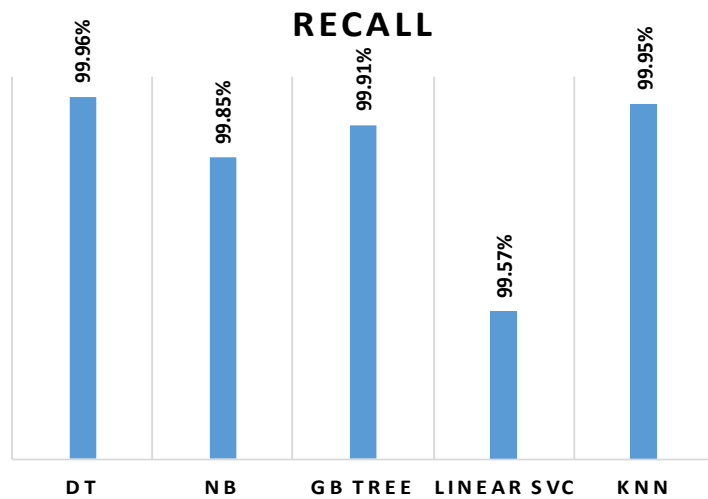


Figure 5. Recall of detecting the Trojan Horse

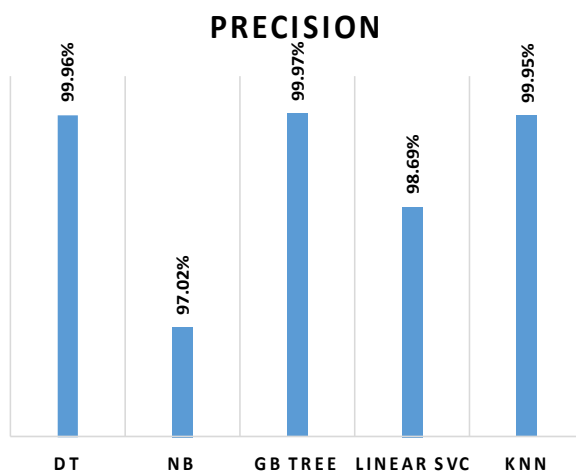


Figure 6. Precision of detecting the Trojan Horse

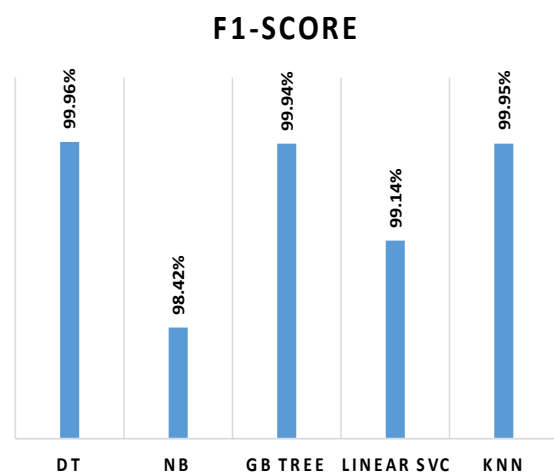


Figure 7. F1-score of detecting the Trojan Horse

#### 4. CONCLUSION

As networking and internet technologies continue to advance, Trojan Horse developers have rapidly adapted their malicious code, often exploiting vulnerabilities in operating systems. Despite the existence of various techniques for detecting Trojan Horses through operating system memory analysis, newly developed Trojans continue to evade these methods. In response to this challenge, we propose a Trojan Horse detection model that utilizes a DT classifier based on data extracted from system memory. To assess the effectiveness of our model, we evaluated it using the Trojan-MalMem dataset. The performance metrics demonstrated that all classifiers achieved high accuracy in Trojan Horse classification. Among these classifiers, DT excelled with an impressive accuracy, recall, precision, and F1-score of 99.96%. These results underscore the significant contribution of memory analysis data to achieving a high success rate in Trojan Horse detection. It's worth noting that the parameters used in this study and the results obtained are specific to the Trojan-MalMem dataset. Different datasets with varying features or classes may yield different results, which is a limitation to consider. However, we firmly believe that employing the ML approach holds promise for successful Trojan Horse detection. This study lays the foundation for further classification research using ML in-memory analysis and Trojan Horse detection. Future studies could explore different hyperparameters, and we intend to expand into multiclass classification, considering the six distinct class labels within the Trojan-MalMem dataset: Benign, Zeus, Emotet, Refroso, Scar, and Reconyc.





#### REFERENCES

- [1] J. Lei *et al.*, "A Reinforcement Learning Approach for Defending Against Multiscenario Load Redistribution Attacks," *IEEE Transactions on Smart Grid*, vol. 13, no. 5, pp. 3711–3722, Sep. 2022, doi: 10.1109/TSG.2022.3175470.
- [2] T. Alves, R. Das, and T. Morris, "Embedding Encryption and Machine Learning Intrusion Prevention Systems on Programmable Logic Controllers," *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 99–102, Sep. 2018, doi: 10.1109/les.2018.2823906.
- [3] M. M. Abualhaj, A. Abu-Shareha, Q. Shambour, A. Alsaaidah, S. Al-Khatib, and M. Anbar, "Customized K-nearest neighbors' algorithm for malware detection," *International Journal of Data and Network Science*, vol. 8, no. 1, pp. 431-438, 2024, doi: 10.5267/j.ijdns.2023.9.012.
- [4] W. Peng, F. Li, X. Zou, and J. Wu, "Behavioral Malware Detection in Delay Tolerant Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 53–63, Jan. 2014, doi: 10.1109/tpds.2013.27.
- [5] S. Sen, E. Aydogan, and A. I. Aysan, "Coevolution of Mobile Malware and Anti-Malware," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2563–2574, Oct. 2018, doi: 10.1109/tifs.2018.2824250.
- [6] SonicWall, Cyber Threat Report, *Cyber Threat Intelligence for Navigating the Unknowns of Tomorrow*, 2022.
- [7] R. Mukherjee and R. S. Chakraborty, "Novel Hardware Trojan Attack on Activation Parameters of FPGA-Based DNN Accelerators," in *IEEE Embedded Systems Letters*, vol. 14, no. 3, pp. 131-134, Sept. 2022, doi: 10.1109/LES.2022.3159541.
- [8] C. Bai, Q. Han, G. Mezzour, F. Pierazzi, and V. S. Subrahmanian, "\$\{DBank\}\$DBank: Predictive Behavioral Analysis of Recent Android Banking Trojans," in *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1378-1393, May-Jun. 2021, doi: 10.1109/TDSC.2019.2909902.
- [9] J. Cruz, C. Posada, N. V. R. Masna, P. Chakraborty, P. Gaikwad, and S. Bhunia, "A Framework for Automated Exploration of Trojan Attack Space in FPGA Netlists," in *IEEE Transactions on Computers*, vol. 72, no. 10, pp. 2740-2751, Oct. 2023, doi: 10.1109/TC.2023.3266592.
- [10] S. S. Tirumala, M. R. Valluri, and G. Babu, "A survey on cybersecurity awareness concerns, practices and conceptual measures," *2019 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, 2019, pp. 1-6, doi: 10.1109/ICCCI.2019.8821951.
- [11] M. Belaoued, A. Derhab, S. Mazouzi, and F. A. Khan, "MACoMal: A Multi-Agent Based Collaborative Mechanism for Anti-Malware Assistance," *IEEE Access*, vol. 8, pp. 14329–14343, 2020, doi: 10.1109/access.2020.2966321.
- [12] Saurabh, "Advance Malware Analysis Using Static and Dynamic Methodology," *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*, Bhopal, India, 2018, pp. 1-5, doi: 10.1109/ICACAT.2018.8933769.
- [13] J. V. D. Assen, A. H. Celdrán, A. Zermin, R. Mogenicato, G. Bovet, and B. Stiller, "SecBox: A Lightweight Container-based Sandbox for Dynamic Malware Analysis," *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, Miami, FL, USA, 2023, pp. 1-3, doi: 10.1109/NOMS56928.2023.10154293.
- [14] M. M. Abualhaj, A. A. Abu-Shareha, M. O. Hiari, Y. Alrabanah, M. Al-Zyoued, and M. A. Alsharaiah, "A Paradigm for DoS Attack Disclosure using Machine Learning Techniques," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 3, pp. 192-200, Jan. 2022, doi: 10.14569/ijacsa.2022.0130325.
- [15] M. Kolhar, F. Al-Turjman, A. Alameen, and M. M. Abualhaj, "A Three Layered Decentralized IoT Biometric Architecture for City Lockdown During COVID-19 Outbreak," *IEEE Access*, vol. 8, pp. 163608–163617, 2020, doi: 10.1109/access.2020.3021983.
- [16] H. Al-Mimi, N. A. Hamad, M. M. Abualhaj, S. N. Al-Khatib, and M. O. Hiari, "Improved Intrusion Detection System to Alleviate Attacks on DNS Service," *Journal of Computer Science*, vol. 19, no. 12, pp. 1549–1560, Dec. 2023, doi: 10.3844/jcsp.2023.1549.1560.
- [17] R. Elnaggar, K. Basu, K. Chakraborty, and R. Karri, "Runtime Malware Detection Using Embedded Trace Buffers," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 1, pp. 35-48, Jan. 2022, doi: 10.1109/TCAD.2021.3052856.
- [18] S. M. Jureček and R. Lórencz, "Application of Distance Metric Learning to Automated Malware Detection," in *IEEE Access*, vol. 9, pp. 96151-96165, 2021, doi: 10.1109/ACCESS.2021.3094064.
- [19] V. K. Singh and M. Govindarasu, "A Cyber-Physical Anomaly Detection for Wide-Area Protection Using Machine Learning," in *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3514-3526, Jul. 2021, doi: 10.1109/TSG.2021.3066316.
- [20] H. Haddadpajouh, A. Mohtadi, A. Dehghantanaha, H. Karimipour, X. Lin, and K. -K. R. Choo, "A Multikernel and Metaheuristic Feature Selection Approach for IoT Malware Threat Hunting in the Edge Layer," in *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4540-4547, Mar. 2021, doi: 10.1109/JIOT.2020.3026660.
- [21] A. Kulkarni, Y. Pino, and T. Mohsenin, "Adaptive real-time Trojan detection framework through machine learning," *2016 IEEE*





- International Symposium on Hardware Oriented Security and Trust (HOST)*, McLean, VA, USA, 2016, pp. 120-123, doi: 10.1109/HST.2016.7495568.
- [22] K. Worley and M. T. Rahman, "Supervised Machine Learning Techniques for Trojan Detection with Ring Oscillator Network," *2019 SoutheastCon*, Huntsville, AL, USA, 2019, pp. 1-7, doi: 10.1109/SoutheastCon42311.2019.9020626.
- [23] S. Xuan, D. Man, W. Wang, K. Qin, and W. Yang, "Hybrid Classification of WEB Trojan Exploiting Small Volume of Labeled Data Vectors," *2018 14th International Conference on Computational Intelligence and Security (CIS)*, Hangzhou, China, 2018, pp. 286-290, doi: 10.1109/CIS2018.2018.00070.
- [24] M. Dener, G. Ok, and A. Orman, "Malware Detection Using Memory Analysis Data in Big Data Environment," *Applied Sciences*, vol. 12, no. 17, p. 8604, Aug. 2022, doi: 10.3390/app12178604.
- [25] J. H. Al-Mimi, N. A. Hamad, and M. M. Abualhaj, "A Model for the Disclosure of Probe Attacks Based on the Utilization of Machine Learning Algorithms," *2023 10th International Conference on Electrical and Electronics Engineering (ICEEE)*, Istanbul, Turkiye, 2023, pp. 241-247, doi: 10.1109/ICEEE59925.2023.00051.
- [26] L. Ma, B. Sun, and C. Han, "Training Instance Random Sampling Based Evidential Classification Forest Algorithms," *2018 21st International Conference on Information Fusion (FUSION)*, Cambridge, UK, 2018, pp. 883-888, doi: 10.23919/ICIF.2018.8455427.
- [27] W. B. Zulfikar, Y. A. Gerhana, and A. F. Rahmania, "An Approach to Classify Eligibility Blood Donors Using Decision Tree and Naive Bayes Classifier," *2018 6th International Conference on Cyber and IT Service Management (CITSM)*, Parapat, Indonesia, 2018, pp. 1-5, doi: 10.1109/CITSM.2018.8674353.
- [28] A. MahendraVardhan and S. Sridhar, "Determining False Positive Analysis of Software Vulnerabilities with Predefined Scan Rules using Random Forest Classifier and Decision Tree Technique," *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2022, pp. 622-625, doi: 10.1109/ICAC3N56670.2022.10074458.

## BIOGRAPHIES OF AUTHORS



**Mosleh M. Abualhaj**     is a senior lecturer in Al-Ahliyya Amman University. He received his first degree in Computer Science from Philadelphia University, Jordan, in 2004, master degree in Computer Information System from the Arab Academy for Banking and Financial Sciences, Jordan in 2007, and Ph.D. in Multimedia Networks Protocols from Universiti Sains Malaysia in 2011. His research area of interest includes VoIP, multimedia networking, and congestion control. He can be contacted at email: m.abualhaj@ammanu.edu.jo.



**Sumaya N. Al-Khatib**     is a lecturer in Al-Ahliyya Amman University. She received her B.Sc. degree in Computer Science from Baghdad University, Iraq, in July 1994, and her master degree in Computer Information System from The Arab Academy for Banking and Financial Sciences, Jordan in 2007. Her research area of interest includes text categorization, information retrieval, VoIP, and machine learning. She can be contacted at email: sumayakh@ammanu.edu.jo.