Resource-Efficient Malware Detection Using Feature-Influence-Driven Ensemble Learning

Mrs.T. Ramya
Cyber Security
Chalapathi
mailintoramya@gmail.com

Donthireddy Madhusudhan Reddy
Cyber Security
Chalapathi
chopparapuravi2003@gmail.com

S.Venkata krishna
Cyber Security
Chalapathi
venkatktishna143@gmail.com

Chopparapu Ravi
Cyber Security
Chalapathi
mreddy90385@gmail.com

Shaik Jaheer
Cyber Security
Chalapathi
jaheershaik1491@gmail.com

Vishnumolakala Chaitanya Prasad
Cyber Security
Chalapathi
chaitanyavishnumolakala75@gmail.com

***Abstract*— The rapid proliferation of malware presents serious challenges to digital security, especially for resource constrained devices where efficiency and memory usage are critical. This study proposes an efficient machine learning based malware detection framework using the Kaggle Malware Detection dataset. Preprocessing includes missing value handling, categorical label encoding, numeric feature standardization, and five fold cross validation. Feature selection is performed using the Extra Trees Classifier based on Gini impurity, followed by data balancing through under sampling and over sampling techniques. A wide range of algorithms is evaluated, including Logistic Regression, Support Vector Machine, K Nearest Neighbors, Gaussian and Bernoulli Naive Bayes, Decision Tree, Random Forest, XGBoost, Gradient Boosting, LightGBM, AdaBoost, CatBoost, Histogram based Gradient Boosting, and Extra Trees. Ensemble strategies such as Voting and Stacking classifiers are also explored. Experimental results show Random Forest achieves 98.97% accuracy without feature selection, while the balanced Stacking Classifier attains 99.90% accuracy. Explainable AI techniques, LIME and SHAP, provide feature level insights. A Flask based web application enables secure user interaction, real time preprocessing, prediction visualization, and classification of inputs as legitimate or malware.**

"Keywords— Malware detection, random forest classifier, feature selection, extra tree classifier, resource constrained device, execution time".

I. Introduction

Malware, short for "malicious software," refers to programs designed to compromise, disrupt, or gain unauthorized access to computer systems, networks, and connected devices [1]. With the rapid expansion of the Internet of Things (IoT) and pervasive digital networks, systems face increasingly complex cyber threats [2]. Common malware types—viruses, trojans, spyware, adware, ransomware, and fileless malware—exploit system vulnerabilities and evade conventional defenses [3], causing data breaches, service interruptions, and unauthorized system control [4].

Traditional malware detection methods, including signature-based, behavioral, and heuristic approaches, have been widely applied [5]. Signature-based methods effectively detect known malware but often fail against new or polymorphic variants [6]. Behavioral approaches improve detection of unknown threats but can demand substantial computational resources, limiting their suitability for resource-constrained devices [7]. Heuristic methods offer adaptive capabilities but remain challenged by sophisticated or zero-day attacks [8]. These limitations highlight a critical gap in achieving timely, accurate, and comprehensive malware detection across modern network infrastructures.

This work proposes a comprehensive approach to enhance malware detection and mitigation across heterogeneous systems, including resource-limited IoT environments. It focuses on continuous monitoring, improving visibility into network activities, and enabling timely identification of malicious behavior. By targeting prevalent and impactful malware categories, the approach aims to deliver adaptive protection that strengthens system resilience and reduces breach risks without relying on static or narrowly focused solutions [9].

The significance of this approach lies in its potential to improve cybersecurity in complex digital ecosystems. Effective malware detection can prevent operational disruptions, safeguard sensitive information, and minimize financial and reputational losses [10]. In IoT and edge environments, where devices are highly interconnected and vulnerable, timely and adaptive security measures are essential for maintaining reliability, trust, and service continuity. By addressing the limitations of conventional techniques and offering scalable protection, this approach contributes to the robustness and reliability of modern networked infrastructures.

II. Related Work

Heuristic-based malware detection techniques have been extensively explored to identify both known and unknown threats. Bazrafshan et al. [11] surveyed heuristic detection methods, highlighting their ability to detect zero-day malware by analyzing program behavior rather than relying solely on signatures. While effective for new variants, these techniques often require significant computational resources and may struggle with complex obfuscation. Alzarooni [12] emphasized the importance of adaptability in detecting malware variants, though the study primarily focused on desktop and server environments, leaving challenges in resource-constrained IoT devices largely unaddressed.

IoT-specific malware detection has gained increasing attention. Ngo et al. [13] reviewed IoT malware detection methods using static features, providing a taxonomy for resource-limited devices but lacking focus on dynamic and real-time detection crucial for evolving threats. Sasi et al. [14] extended this by categorizing IoT attacks and examining detection mechanisms, highlighting limitations in scalability and adaptability. These studies underscore the need for

solutions tailored to heterogeneous IoT ecosystems with constrained resources.

Machine learning approaches have shown promise for malware detection. Azeem et al. [15] compared multiple techniques, demonstrating improved detection rates on complex datasets, though handling obfuscated or memory-resident malware remained challenging. Public datasets such as Meraz'18 [16] support benchmarking but often lack real-time or evolving malware samples. Hossain and Islam [17] addressed obfuscated malware in memory dumps, showing promising detection capabilities while noting the trade-off between accuracy and computational efficiency.

Dynamic and behavioral analyses have been applied to further enhance malware recognition. Bhatia et al. [18] used forensic tools for dynamic analysis, revealing malware behavior and attack patterns, though these methods can be computationally intensive and may not scale efficiently for large IoT networks. Mohd et al. [19] and Kumar et al. [20] focused on hybrid and supervised machine learning-based intrusion detection systems, achieving higher detection accuracy across network environments. However, challenges remain in detecting adaptive, zero-day malware and integrating lightweight monitoring suitable for IoT devices.

III. Materials And Methods

The proposed system develops an efficient malware detection framework tailored for resource-constrained devices using a comprehensive set of [21] machine learning algorithms, including Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes (BNB), Decision Tree (DT), Random Forest (RF), Extreme Gradient Boosting (XGB), Gradient Boosting (GB), LightGBM (LGBM), AdaBoost (AB), CatBoost (CB), Histogram-based Gradient Boosting (HGB), and Extra Trees (ET). Feature selection is performed via Extra Trees Classifier to reduce dimensionality, while [22] data balancing methods, including under-sampling and over-sampling, address class imbalances. A five-fold cross-validation ensures robust evaluation. Ensemble learning techniques, such as Voting Classifier and Stacking Classifier, alongside Explainable AI methods like LIME and SHAP, provide interpretability. The system leverages a real-time, user-friendly interface built with Flask and utilizes malware datasets for comprehensive evaluation across diverse environments.

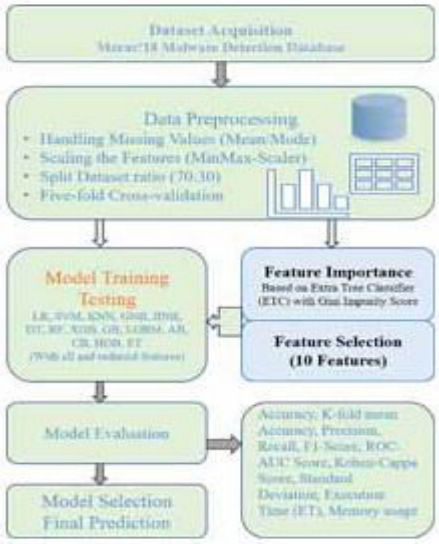


Fig. 1. System Architecture

Fig. 1 illustrates a comprehensive machine learning pipeline for malware detection. The process begins with data pre-processing and feature selection using an Extra Trees Classifier, followed by data balancing. After splitting into train and test sets, multiple algorithms—including ensemble methods like XGBoost and LightGBM—build the models. Finally, the best-performing model is evaluated, interpreted via XAI (LIME/SHAP), and deployed through a Flask web application.

A) Dataset Collection

The dataset comprises 216,352 executable files with 58 features, including header information, section characteristics, imports, exports, resources, and version details. [23] Features such as SizeOfCode, SectionsMeanEntropy, and ResourcesMeanSize capture structural and behavioral attributes, while the legitimate column labels files as benign or malicious. The dataset includes both numeric and categorical data types, with minimal missing values. It provides a comprehensive representation of executable properties, enabling robust training and evaluation of malware detection models for resource-constrained environments.

	ID	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion
0	1	b69acb3bb133974e48229627663f96d4	332	224	8450	8.0	0
1	2	1cbee4b3725629bd0aa6ac2ff500925f	332	224	258	9.0	0
2	3	b7027cf0cd31c820928950cbfe7e91ef	332	224	8450	8.0	0
3	4	156a0bb069f94d1e7c25083188051f2a4	332	224	8450	10.0	0
4	5	c72bf851fed5542abba904b1f3944cd5	332	224	8226	48.0	0

5 rows × 58 columns

Fig. 2. Malware Detection Dataset

B) Pre-Processing

The preprocessing pipeline ensures dataset quality and consistency through cleaning, missing value imputation, scaling, feature selection, and class balancing, preparing data for robust and efficient malware detection modeling.

Data Pre-Processing: Pre-processing involves cleaning and preparing the dataset for modeling. Irrelevant columns such as ID, md5, Machine, and Unnamed: 57 are removed. Missing values are imputed using the median strategy to ensure completeness. Duplicate entries are identified and eliminated, and the dataset index is reset. Numerical features are scaled using MinMaxScaler to normalize values between 0 and 1, facilitating efficient learning and improving convergence for machine learning models.

Feature Selection: Feature selection identifies and retains the most informative attributes to enhance model performance. Techniques such as Extra Trees Classifier are used to evaluate feature importance, reducing dimensionality and computational overhead. By selecting relevant features, the system eliminates redundant or less significant attributes, improving model generalization, interpretability, and training efficiency. This ensures accurate malware detection while maintaining suitability for resource-constrained environments and optimizing predictive capability across diverse executable samples.

Data Balancing: Data balancing addresses class imbalances between malicious and benign samples to prevent biased model learning. Methods such as under-sampling and over-sampling are applied to ensure equitable representation of both classes. Balanced datasets improve model generalization, reduce false negatives or false positives, and

enhance robustness. By mitigating skewed distributions, this step ensures that machine learning models can reliably detect malware patterns, particularly in minority-class instances, while maintaining computational efficiency for constrained devices.

C) Training and Testing

The dataset is split 70:30 for training and testing with stratified class representation. Models are trained on preprocessed, balanced features, and evaluated using accuracy, precision, recall, F1-score, ROC-AUC, Cohen-Kappa, execution time, and memory usage. Cross-validation ensures stability and prevents overfitting, providing robust malware detection assessment.

D) Algorithms

Logistic Regression (LR): Processes preprocessed features to model the probability of malware as malicious or benign, providing a simple, interpretable, and computationally efficient classification method suitable for resource-limited devices while maintaining reasonable detection accuracy and low false positives [24].

$$\hat{y}_i = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \quad (1)$$

Support Vector Machine (SVM): Separates malicious and benign samples by learning optimal decision boundaries in high-dimensional feature space, [25] handling non-linear relationships, and providing accurate and robust classification for devices with limited memory and computational resources.

$$minimize\ \frac{1}{2}\ ||W||^2 + C\sum_{i=1}^{n} \xi_i \quad (2)$$

K-Nearest Neighbors (KNN): Classifies samples based on similarity to nearest neighbors in feature space, [26] offering flexible, intuitive detection of malware patterns with minimal preprocessing and effective performance across diverse feature sets.

$$distance(x, X_i) = \sqrt{\sum_{j=1}^{d} \left(x_j - X_{i_j}\right)^2} \quad (3)$$

Gaussian Naive Bayes (GNB): Estimates the likelihood of malware or benign status using feature statistics, [27] providing a fast, lightweight, and interpretable probabilistic classifier suitable for environments with constrained computational resources.

Bernoulli Naive Bayes (BNB): Analyzes binary feature patterns to classify samples as malicious or benign, offering rapid prediction and low memory usage, [28] ideal for devices requiring lightweight yet effective detection mechanisms.

Decision Tree (DT): Builds hierarchical decisions based on feature values to classify malware, offering an interpretable and [29] computationally efficient approach capable of capturing complex relationships while being suitable for resource-constrained devices.

$$I\ (i) = 1 - \sum_{i=1}^{k} p_i^2 \quad (4)$$

Random Forest (RF): Aggregates multiple decision trees to classify samples, improving accuracy and reducing overfitting, providing a robust, [30] high-performance method suitable for environments with limited resources.

$$Gini = 1 - \sum_{i=1}^{C} (Pi)^2 \quad (5)$$

Extreme Gradient Boosting (XGB): Builds sequential trees to refine predictions of malware versus benign samples, improving accuracy, reducing overfitting, and handling feature interactions efficiently for resource-limited environments.

Gradient Boosting (GB): Sequentially combines weak learners to correct errors, capturing complex feature relationships and enhancing classification accuracy, while maintaining computational efficiency suitable for constrained devices.

$$\hat{y} = f\left(W^L f\left(W^{L-1} \cdots f(W^1 X + b^1) + b^{(L-1)}\right) + b^L\right) \quad (6)$$

LightGBM (LGBM): Optimizes tree-based learning for large feature sets, reducing memory usage and accelerating training, providing fast, scalable, and accurate detection for devices with limited resources.

AdaBoost (AB): Combines multiple weak classifiers to emphasize challenging samples, improving overall detection accuracy and providing a robust, computationally efficient method suitable for resource-limited environments.

$$H(x) = sign\left(\sum_{t=1}^{T} \propto_t h_t(x)\right) \quad (7)$$

CatBoost (CB): Processes numeric and categorical features efficiently, applying ordered boosting and symmetric tree construction for accurate malware classification with minimal memory and execution time.

$$F(x) = F_0(x) + \sum_{m=1}^{M} \sum_{i=1}^{N} f_m(x_i) \quad (8)$$

Histogram-based Gradient Boosting (HGB): Discretizes features into bins for faster training and sequential tree building, balancing predictive accuracy with computational efficiency for constrained environments.

Extra Trees (ET): Builds multiple randomized decision trees and aggregates predictions, enhancing robustness, accuracy, and efficiency while handling high-dimensional feature sets in memory-limited environments.

Voting Classifier: Aggregates predictions from multiple models via majority voting, improving classification robustness and accuracy while balancing computational demands for constrained environments.

$$\hat{y}\ = argmax_c \left(\sum_{i=1}^{n} II\ (\hat{y}_i = c)\right) \quad (9)$$

Stacking Classifier: Combines base model outputs with a meta-classifier to enhance accuracy and reliability by learning relationships among predictions, supporting robust malware identification in limited-resource environments.

$$\hat{y} = g(Y_{base}) = g\big(f_1(x), f_2(x), \dots, f_m(x)\big) \quad (10)$$

Stacking Classifier Balanced Data: Uses balanced datasets with base models and a meta-classifier to reduce bias, improving detection of majority and minority classes while maintaining computational efficiency in constrained environments.

E) Integration of XAI and Flask Framework

The system employs Explainable Artificial Intelligence (XAI) techniques, including LIME and SHAP, to provide transparent, interpretable insights into malware detection predictions. LIME offers local explanations by highlighting feature contributions for individual instances, while SHAP delivers global interpretability, quantifying feature importance across multiple samples and visualizing their influence on model outputs, enhancing trust and reliability.

A Flask-based interface enables interactive deployment of the model and XAI functionalities. Users can input feature values, obtain real-time predictions, and visualize explanations from LIME and SHAP. This integration ensures practical usability, allowing resource-constrained devices to achieve interpretable, efficient, and actionable malware detection.

IV. Experimental Results

Accuracy: The accuracy of a test is its ability to differentiate the patient and healthy cases correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases. Mathematically, this can be stated as:

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} (11)$$

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} (12)$$

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} (13)$$

F1-Score: F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

$$F1\ Score = 2 * \frac{Recall\ \text{X}\ Precision}{Recall + Precision} * 100 (14)$$

AUC-ROC Curve: The AUC-ROC Curve is a performance measurement for classification problems at various threshold settings. ROC plots the True Positive Rate against the False Positive Rate. AUC quantifies the overall ability of the model to distinguish between classes, where a higher AUC indicates better model performance.

$$AUC = \sum_{i=1}^{n-1} (FPR_{i+1} - FPR_i) \cdot \frac{TPR_{i+1} + TPR_i}{2} \quad (15)$$

Cohen Kappa: Cohen's Kappa (κ) is a statistical measure used to quantify the level of agreement between two raters (or judges, observers, etc.) who each classify items into categories. It's especially useful in situations where decisions are subjective and the categories are nominal (i.e., they do not have a natural order).

$$Kappa(k) = \frac{P_o - P_e}{1 - P_e} \quad (16)$$

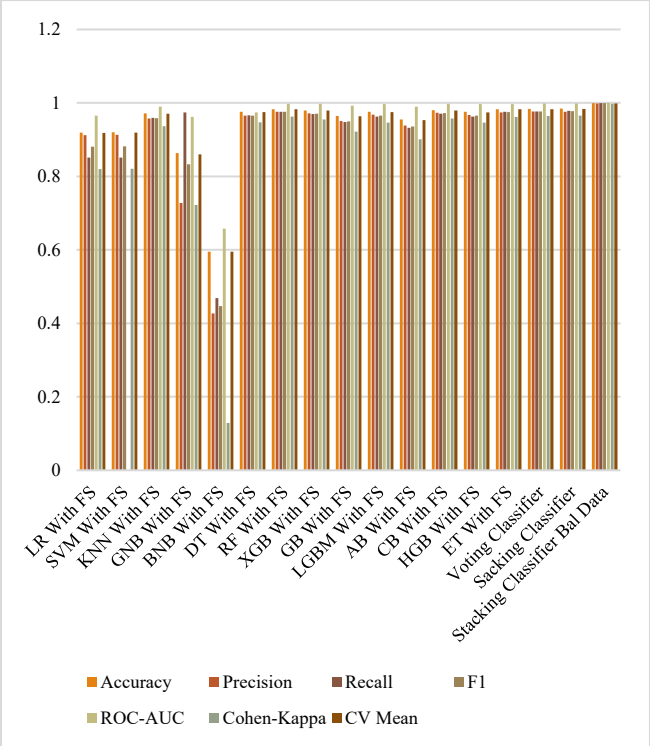
Table.1 Performance Evaluation – With FS

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC	Cohen-Kappa
LR With FS	0.919514	0.912591	0.850868	0.880649	0.965814	0.820046
SVM With FS	0.920007	0.913387	0.851530	0.881375	NaN	0.821145
KNN With FS	0.971235	0.957999	0.959649	0.958823	0.989887	0.936721
GNB With FS	0.863680	0.727554	0.974173	0.832994	0.962115	0.721861
BNB With FS	0.594953	0.426860	0.468809	0.446852	0.658044	0.128453
DT With FS	0.975950	0.965029	0.966094	0.965561	0.974350	0.947085
RF With FS	0.983253	0.975732	0.976292	0.976012	0.998002	0.963148
XGB With FS	0.979524	0.971559	0.969714	0.970636	0.997722	0.954918
GB With FS	0.964687	0.950959	0.947684	0.949319	0.992612	0.922223
LGBM With FS	0.975734	0.967649	0.962651	0.965143	0.996837	0.946532
AB With FS	0.955228	0.938797	0.932497	0.935637	0.990076	0.901313
CB With FS	0.980556	0.973480	0.970730	0.972103	0.997729	0.957181

HGB With FS	0.97562 6	0.96730 7	0.96269 5	0.964995	0.996766	0.946300
ET With FS	0.98266 7	0.97447 5	0.97589 5	0.975185	0.996755	0.961868
Voting Classifier	0.98376 1	0.97673 4	0.97673 4	0.976734	0.998550	0.964262
Sacking Classifier	0.98416 2	0.97613 1	0.97854 4	0.977336	0.998642	0.965164
Stacking Classifier Bal Data	0.99902 6	0.99877 0	0.99928 6	0.999028	0.999989	0.998053

Table 1 evaluates models using accuracy, precision, recall, F1-score, ROC-AUC, Cohen-Kappa, cross-validation mean, execution time, and memory usage. The Stacking Classifier with balanced data uniquely achieves superior performance across all metrics.

Fig. 3. Comparison Graph– With FS



In Fig.3 clearly demonstrates that the Stacking Classifier with balanced data achieves the highest accuracy, precision, recall, and F1-score, outperforming all other models across evaluated performance metrics.

V. Conclusion

The study confirms that machine learning based techniques can deliver highly accurate malware detection while preserving efficiency for resource constrained devices. Comprehensive preprocessing, Extra Trees based feature selection, and effective data balancing enabled robust evaluation across diverse classifiers. Without feature selection, the Random Forest model achieved strong results with 98.97% accuracy, 98.61% precision, 98.43% recall, and a 98.52% F1 score, demonstrating reliable detection using full feature sets. After applying feature selection, the balanced Stacking Classifier significantly improved performance, reaching 99.90% accuracy, 99.88% precision, 99.92% recall, and a 99.90% F1 score, highlighting the benefit of ensemble learning with reduced dimensionality. Explainable AI techniques, including LIME and SHAP, enhanced transparency by clearly identifying influential features and supporting trustworthy decisions. For real world usability, the trained models were deployed using a Flask based web application with secure interaction, real time input preprocessing, and prediction visualization. The system outputs clear classifications as legitimate or malware, enabling efficient, interpretable, and scalable malware detection suitable for constrained computing environments requiring rapid threat response and minimal operational overhead globally.

Future research can explore lightweight deep learning models, including CNNs and recurrent architectures, tailored for embedded devices. Incremental and online learning can enable real-time adaptation to emerging malware. Expanding datasets to cover diverse malware families and zero-day attacks will enhance robustness. Optimizing hybrid ensemble strategies and integrating models with edge computing and IoT platforms can ensure rapid, accurate malware detection with minimal computational and memory overhead.

References

[1] Basak, M., Kim, D. W., Han, M. M., & Shin, G. Y. (2024). Attention-Based Malware Detection Model by Visualizing Latent Features Through Dynamic Residual Kernel Network. Sensors, 24(24), 7953.

[2] Gutierrez, R., Villegas-Ch, W., Godoy, L. N., Mera-Navarrete, A., & Luján-Mora, S. (2024). Application of deep learning models for real-time automatic malware detection. IEEE Access, 12, 107742-107756.

[3] Torres, M., Álvarez, R., & Cazorla, M. (2023). A malware detection approach based on feature engineering and behavior analysis. IEEE Access, 11, 105355-105367.

[4] Gorment, N. Z., Selamat, A., Cheng, L. K., & Krejcar, O. (2023). Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions. IEEE Access, 11, 141045-141089.

[5] Euh, S., Lee, H., Kim, D., & Hwang, D. (2020). Comparative analysis of low-dimensional features and tree-based ensembles for malware detection systems. IEEE Access, 8, 76796-76808.

[6] H. Gill, ‘‘Malware: Types, analysis and classifications,’’ Univ. Bradford, U.K., Tech. Rep., 2022.

[7] M. Al-Fawa’Reh, J. Abu-Khalaf, P. Szewczyk, and J. J. Kang, ‘‘MalBoT DRL: Malware botnet detection using deep reinforcement learning in IoT networks,’’ IEEE Internet Things J., vol. 11, no. 6, pp. 9610–9629, Mar. 2024.

[8] I. Firdausi, C. Lim, A. Erwin, and A. S. Nugroho, ‘‘Analysis of machine learning techniques used in behavior-based malware detection,’’ in Proc. 2nd Int. Conf. Adv. Comput., Control, Telecommun. Technol., Dec. 2010, pp. 201–203.

[9] D. Venugopal and G. Hu, ‘‘Efficient signature based malware detection on mobile devices,’’ Mobile Inf. Syst., vol. 4, no. 1, pp. 33–49, Jan. 2008.

[10] Ö. A. Aslan and R. Samet, ‘‘A comprehensive review on malware detection approaches,’’ IEEE Access, vol. 8, pp. 6249–6271, 2020.

[11] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, ‘‘A survey on heuristic malware detection techniques,’’ in Proc. 5th Conf. Inf. Knowl. Technol., May 2013, pp. 113–120.

[12] K. Alzarooni, ‘‘Malware variant detection,’’ Ph.D. thesis, Dept. Comput. Sci., Univ. College London, London, U.K., Mar. 2012.

[13] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, ‘‘A survey of IoT malware and detection methods based on static features,’’ ICT Exp., vol. 6, no. 4, pp. 280–286, Dec. 2020.

[14] T. Sasi, A. H. Lashkari, R. Lu, P. Xiong, and S. Iqbal, ‘‘A comprehensive survey on IoT attacks: Taxonomy, detection

mechanisms and challenges,'' J. Inf. Intell., vol. 2, no. 6, pp. 455–513, Nov. 2024.

[15] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, ''Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches,'' Heliyon, vol. 10, no. 1, Jan. 2024, Art. no. e23574.

[16] [16] (2018). Meraz'18 Dataset. Accessed: Aug. 15, 2024. [Online]. Available: https://www.kaggle.com/competitions/malware detection/dataselect=data.csv

[17] M. A. Hossain and M. S. Islam, ''Enhanced detection of obfuscated malware in memory dumps: A machine learning approach for advanced cybersecurity,'' Cybersecurity, vol. 7, no. 1, p. 16, Jan. 2024.

[18] A. M. Bhatia, I. Kumar, and N. Mohd, ''Dynamic analysis of a malware sample: Recognizing its behavior using forensic application,'' in Proc. 4th IEEE Global Conf. Advancement Technol. (GCAT), Oct. 2023, pp. 1–6. 12664

[19] N. Mohd, A. Singh, and H. S. Bhadauria, ''Intrusion detection system based on hybrid hierarchical classifiers,'' Wireless Pers. Commun., vol. 121, no. 1, pp. 659–686, Nov. 2021.

[20] I. Kumar, N. Mohd, C. Bhatt, and S. K. Sharma, ''Development of IDS using supervised machine learning,'' in Soft Computing: Theories and Applications: Proceedings of SoCTA 2019. Cham, Switzerland: Springer, 2020, pp. 565–577.

[21] M. Goyal and R. Kumar, ''Machine learning for malware detection on balanced and imbalanced datasets,'' in Proc.Int. Conf.Decis.AidSci.Appl. (DASA), Nov. 2020, pp. 867–871.

[22] A. Hota, S. Panja, and A. Nag, ''Lightweight CNN-based malware image classification for resource-constrained applications,'' Innov. Syst. Softw. Eng., pp. 1–14, Jul. 2022.

[23] S. Song, N. Gao, Y. Zhang, and C. Ma, ''BRITD: Behavior rhythm insider threat detection with time awareness and user adaptation,'' Cybersecurity, vol. 7, no. 1, p. 2, Jan. 2024.

[24] V. Tanksale, ''Intrusion detection system for controller area network,'' Cybersecurity, vol. 7, no. 1, p. 4, Feb. 2024.

[25] A. Kumar, K. Abhishek, K. Shah, D. Patel, Y. Jain, H. Chheda, and P. Nerurkar, ''Malware detection using machine learning,'' in Proc. 1st Indo-Amer. Conf., 2nd Ibero-American Conf. Knowl. Graphs Semantic Web(KGSWC),Mérida,Mexico.Cham,Switzerland:Springer, Nov. 2020, pp. 61–71.

[26] V. Patel, S. Choe, and T. Halabi, ''Predicting future malware attacks on cloud systems using machine learning,'' in Proc. IEEE 6th Intl Conf. Big Data Secur. Cloud (BigDataSecurity), IEEE Intl Conf. High Perform. Smart Comput., (HPSC), IEEE Intl Conf. Intell. Data Secur. (IDS), May 2020, pp. 151–156.

[27] M. Masum, M. J. Hossain Faruk, H. Shahriar, K. Qian, D. Lo, and M. I. Adnan, ''Ransomware classification and detection with machine learning algorithms,'' in Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC), Jan. 2022, pp. 0316–0322.

[28] I. Shhadat, B. Bataineh, A. Hayajneh, and Z. A. Al-Sharif, ''The use of machine learning techniques to advance the detection and classification of unknown malware,'' Proc. Comput. Sci., vol. 170, pp. 917–922, Jan. 2020.

[29] R. Damaševičius, A. Venčkauskas, J. Toldinas, and Š. Grigaliunas, ''Ensemble-based classification using neural networks and machine learning models for windows PE malware detection,'' Electronics, vol. 10, no. 4, p. 485, Feb. 2021.

[30] J. C. Kimmell, M. Abdelsalam, and M. Gupta, ''Analyzing machine learning approaches for online malware detection in cloud,'' in Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP), Aug. 2021, pp. 189–196.