iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.52217

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Malware Detection Using Machine Learning

Kunal Bhat[1], Tejas Khairnar[2], Sharayu Phatangare[3], Tanmay Narkhedkar[4]
Department of Computer Engineering, D Y Patil institute of engineering and technology, Ambi, Pune

Abstract: ***The sophistication of malicious software, known as malware, continues to advance. Previous approaches to detecting malware have predominantly focused on software-based detectors, which are susceptible to compromise. Consequently, recent efforts have suggested the adoption of hardware-assisted malware detection. In this research, we present a fresh framework for hardware-assisted malware detection that utilizes machine learning to monitor and classify patterns of memory access. This framework offers enhanced automation and coverage by reducing the reliance on specific malware signatures from the user. Our work is based on the fundamental understanding that malware must modify control flow and/or data structures, thereby leaving identifiable traces in program memory accesses. Expanding on this insight, we propose an online framework for malware detection that employs machine learning to classify malicious behaviour based on patterns of virtual memory access. Key elements of this framework include techniques for gathering and summarizing memory access patterns at the function and system call levels, as well as a two-level classification architecture.***

Keywords: (Dynamic analysis, Machine learning, Malicious software, Obfuscation techniques, Static analysis, API calls, Ransomware.)

I. INTRODUCTION

The continuous rise of malicious software, commonly known as malware, poses a significant security threat that requires ongoing research efforts for effective detection. The initial step in detecting malware involves analysis, which can be performed through either static or dynamic methods. Typically, this analysis takes place offline and involves the expertise of human professionals. The results of the analysis are condensed into what is known as a "signature." One approach for detecting malware involves using static signatures to inspect programs after they are loaded but before execution. However, this method can be circumvented by malware that utilizes obfuscation techniques to evade detection. As a response to this challenge, dynamic behaviour-based detection has been proposed. These techniques monitor system behaviour using instrumentation based on the operating system or hypervisor, aiming to identify malicious activities. Both static and dynamic signatures can be derived through either deterministic or statistical techniques. Statistical techniques employ machine learning to uncover patterns associated with malicious behaviour. On the other hand, deterministic signatures are typically constructed based on the meticulous analysis performed by human experts.

A. Malware Definition:

The term "malware" is derived from "malicious software" and serves as a comprehensive label encompassing various types of harmful programs, including viruses, Trojans, worms, and more. These programs possess a range of capabilities, such as unauthorized acquisition, encryption, or removal of sensitive data, manipulation or takeover of fundamental computer operations, and monitoring of user activities, often without explicit permission.

B. Types of Malwares

1) *Virus*: A computer virus is typically an unauthorized program that can be installed without the user's consent, leading to potential harm to both the computer's operating system and its physical hardware components. The effects produced by such viruses can be diverse and detrimental to the overall functionality and stability of the computer system.
2) *Worms:* Computer worms are programs with destructive effects that use communication between computers to spread. Worms have common features with viruses, ii. Worms are able to multiply like viruses, but not locally, but on other computers. It uses computer networks to spread to other systems.
3) *Trojan horse*: Trojan horses are deceptive programs designed to create vulnerabilities in an operating system, enabling unauthorized access for users. Unlike computer viruses, Trojans lack the ability to self-replicate on their own. These "disguised" programs attempt to exploit file-sharing networks to infiltrate systems and compromise their security.
4) *Ransomware:* Ransomware is a form of malicious software that restricts the victim's computer access and extorts a payment as ransom. The specific ransom amount and the stated justification for payment vary depending on the particular variant of the virus.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

II. OBJECTIVES

The objective of this research is to explore the implementation of machine learning techniques in malware detection for the identification of unknown malware. The aim is to develop a software solution that utilizes machine learning to effectively detect unknown malware instances. The primary goal is to validate the effectiveness of machine learning-based malware detection by achieving a high accuracy rate while minimizing false positives.

III. METHODOLOGY

A. *Feature Extraction*

There are many format features in PE files, but most of those features are not helpful in distinguishing malware and benign software. Based on our empirical studies and in-depth analysis of the format features of the PE files, we extracted 54 features that have the potential to distinguish between benign software and malware, from given PE files. These features are summarized in. In the below discussion, we gave a brief description of the extracted features.

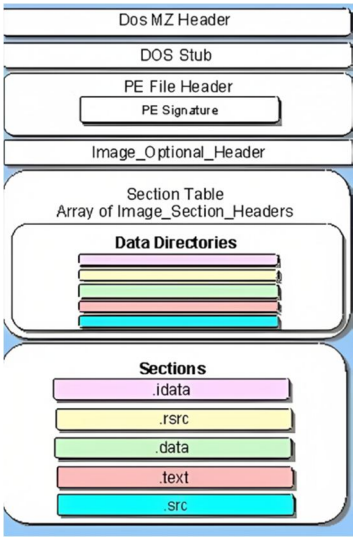


Fig. 1 PE File Features

B. *Algorithms:*

1) *Random Forest*: Random forests are an effective alternative for classification tasks, utilizing trees with fixed structures and random features. The algorithm generates a collection of independent decision trees, which can be easily parallelized. Each tree is built as a complete binary tree, with randomly selected features (often with replacement) used for branching. The leaves of the tree are completed based on training data. The resulting classifier combines the predictions of all the trees through a voting mechanism. This approach works well when all the features are relevant, as each tree only selects a small number of features. The strength of random forests lies in the fact that while some trees may make random predictions due to unnecessary features, others will query important features and make accurate predictions based on the training data.
2) *Ada Boost*: AdaBoost, also known as Adaptive Boosting, was proposed by Freund and Schapire in 1996 as an ensemble boosting classifier. Its purpose is to enhance accuracy by combining multiple classifiers. AdaBoost follows an iterative approach to construct a strong classifier by combining weaker ones. During each iteration, the weights of classifiers and training data samples are adjusted to ensure precise predictions. The training procedure involves the iterative selection of training data subsets based on the accuracy of the previous iteration, with higher weights assigned to incorrectly classified observations. The weight of each classifier is adjusted proportionally to its accuracy. This iterative process continues until either the training data is accurately classified or the maximum number of iterations is reached.
3) *Gradient Descent*: Gradient boosting is a technique that enhances the strength of a weak learner or learning algorithm by making a series of tweaks to it. It is based on the concept of Probability Approximately Correct Learning (PAC) and investigates the complexity of machine learning problems. The Gradient Boosting Classifier relies on a differentiable loss function, such as logarithmic loss for classification or squared errors for regression.

©IJRASET: All Rights are Reserved

4) *Decision Tree:* A decision tree is a hierarchical structure where features are represented by nodes, decision rules are represented by branches, and outcomes are represented by leaf nodes. The topmost node is called the root node, and the tree recursively partitions based on attribute values. The visual nature of decision trees, resembling flowcharts, makes them easy to comprehend and interpret, reflecting human thought processes. Decision trees are non-parametric and do not require assumptions about probability distributions. With high accuracy, decision trees can effectively handle high-dimensional data.
5) *Naïve Bayes:* Naive Bayes is a simple yet powerful classification algorithm based on Bayes' theorem. Naive Bayes calculates the probability of a data point belonging to each class and selects the class with the highest probability. Naive Bayes requires a relatively small amount of training data to estimate the necessary probabilities. It is computationally efficient and performs well in high-dimensional datasets. However, the assumption of feature independence may limit its accuracy in some complex scenarios. Bayes' theorem offers a method to compute the posterior probability P(c—x) using the prior probabilities P(c), P(x), and the likelihood P(x—c).

The posterior probability P(c—x) represents the probability of a class (c, target) given a predictor (x, attributes). The prior probability P(c) is the initial probability of the class. The likelihood P(x—c) denotes the probability of the predictor given the class. Lastly, the prior probability P(x) refers to the initial probability of the predictor.

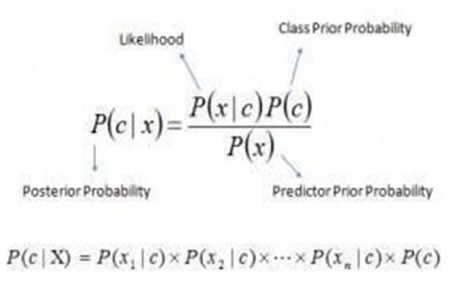


Fig. 2 Naïve Bayes

IV. FRAMEWORK and ARCHITECTURE

A. *System Design*

Both signature-based and behavior-based techniques have advantages and limitations in malware detection. To leverage the benefits of both methods and address their shortcomings, many researchers have proposed hybrid approaches that combine static and dynamic features for malware detection. This section discusses various hybrid malware detection techniques and compares them based on several parameters. In their work, Rabek et al. (2003) presented a technique for detecting obfuscated malicious files. They performed static analysis to gather information about system calls, including function names, addresses, and return addresses. This static information was combined with dynamic features by executing the malware files in a controlled dynamic environment. If an executable file invoked the same system calls as those stored (representing known malware), it was classified as malicious. However, this technique could fail if the malware developer embedded irrelevant system calls in the code. Collins et al. (2008) proposed a protocol graph detector to identify worms in a network. They created a network representation where hosts were nodes and connections were edges. This technique simulated the network to observe worm behavior. It specifically focused on worms and did not address other types of malwares like Trojan horses or viruses. Mangialardo et al. (2015) introduced the FAMA framework to overcome the weaknesses of static and dynamic analysis techniques and reduce false-positive responses. Static features were extracted using IDA Pro, while behavioral features were captured using the Cuckoo sandbox. The extracted features were then fed into Random Forest and C5.0 algorithms for training the classifier. Experimental results demonstrated a 95.75% accuracy in distinguishing between benign and malicious files. Shijo et al. (2015) proposed an integrated malware detection approach. They disassembled binary files and extracted printable string information, taking into account unwanted printable strings inserted to obfuscate the code. Overall, the study discusses various hybrid malware detection techniques that aim to leverage the strengths of both static and dynamic analysis while addressing their limitations.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

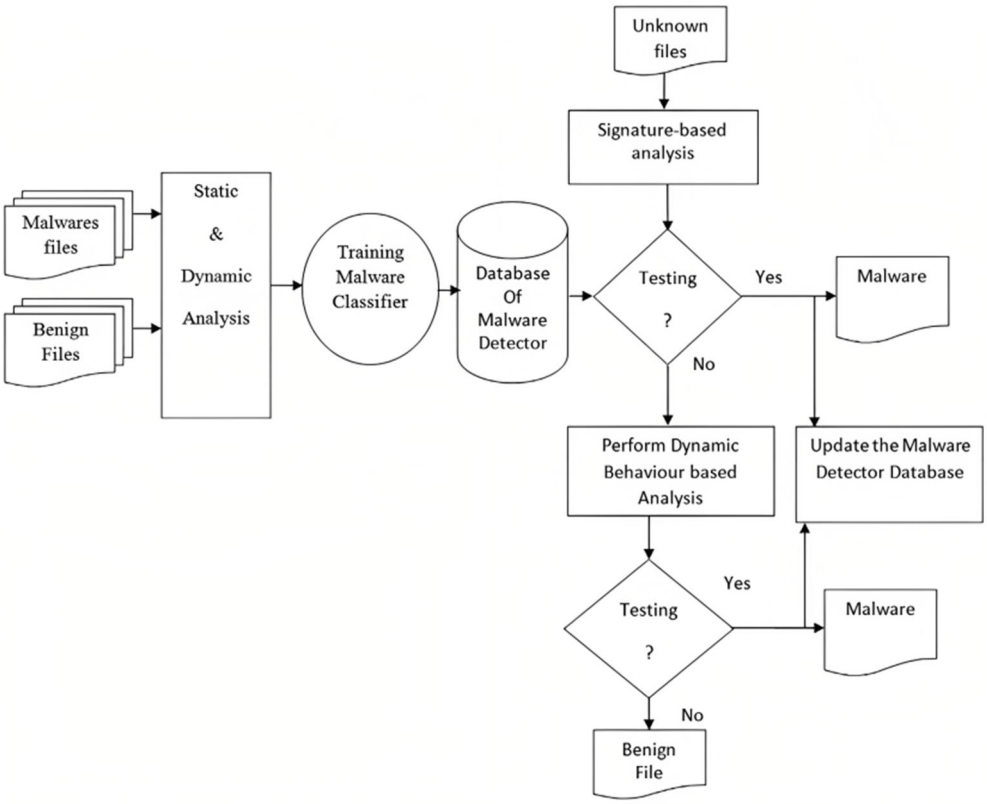


Fig. 3 Schematic Architecture of the proposed Hybrid Malware Detection Technique

B. Hypervisor

A hypervisor plays a crucial role in malware detection when employing machine learning techniques due to several key reasons. First and foremost, a hypervisor provides an isolated and controlled environment in which potentially malicious software can be executed. By running the malware within a virtual machine (VM) environment created by the hypervisor, it operates in an isolated sandbox, separate from the host operating system and other applications. This isolation helps prevent the malware from affecting the underlying system, ensuring the safety and integrity of the host machine. The isolation offered by the hypervisor also serves another important purpose: evading detection by the malware itself. Malicious software often employs techniques to identify whether it is running in a virtual environment or being monitored by security tools. By utilizing a hypervisor, which presents the malware with an environment that mimics a genuine operating system, the malware is less likely to detect the monitoring and analysis tools that are in place. This enhances the chances of effectively analyzing the malware's behavior and detecting its malicious intent. The hypervisor enables the creation of snapshots or checkpoints during the execution of malware. These snapshots capture the state of the VM at different points in time, preserving the exact state of the malware-infected system at various stages of its execution. These snapshots can be instrumental in analyzing the behavior of the malware, allowing researchers to investigate changes made to the system, observe network interactions, and identify potential vulnerabilities being exploited. Moreover, these snapshots can be used to create training datasets for machine learning models, enabling the training of classifiers on diverse and representative malware samples. Another advantage of employing a hypervisor is the repeatability it provides. Researchers can easily recreate the exact execution environment by reverting to a previously captured snapshot, ensuring that experiments and analyses can be repeated consistently. This repeatability is critical for conducting rigorous evaluations, comparing different detection approaches, and validating the effectiveness of machine learning algorithms in detecting malware. In conclusion, a hypervisor is an indispensable component in malware detection using machine learning. Its ability to isolate malware, evade detection, provide fine-grained control, facilitate monitoring and analysis, enable snapshot-based analysis, and ensure repeatability makes it an invaluable tool in the development of robust and accurate malware detection systems.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

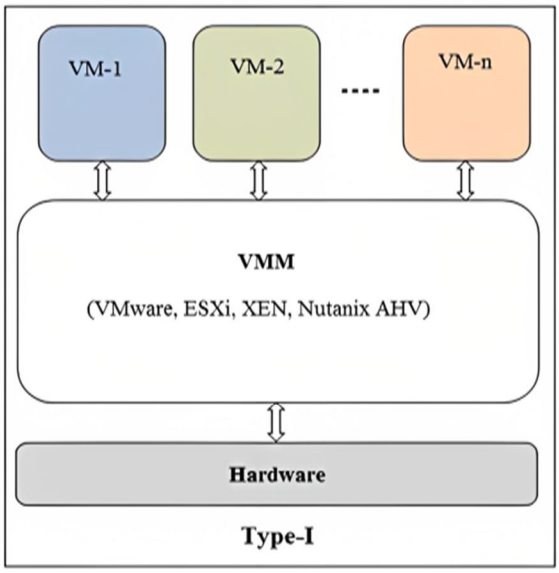


Fig. 1. Architecture of Hypervisor Type-I.

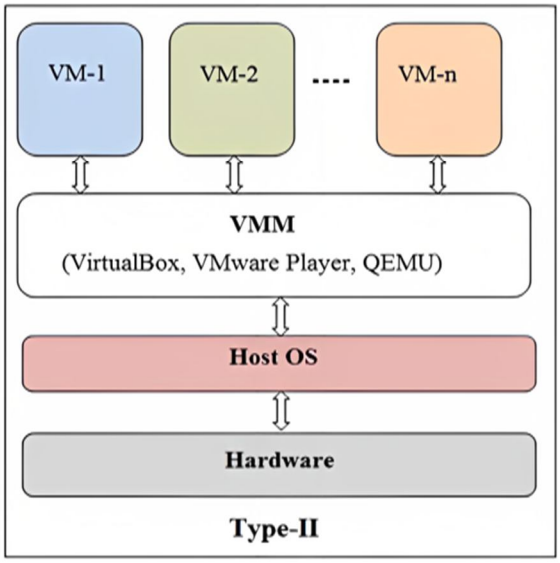


Fig. 4 Type II Hypervisor Architecture

C. Architecture

The behavior-based approach to malware detection relies on identifying malicious activities exhibited by malware during its execution. This approach considers various features such as APIs, browser events, system events, network events, etc., to define the behavior. These parameters are classified into three main categories: file activities, registry activities, and network activities.The underlying concept behind malware detection is the detection of anomalies, which are unusual activities performed by malware. In anomaly-based detection systems, the malware detector is trained by analyzing only benign files. Through static or dynamic analysis, the benign files are examined, and the classifier is trained using their normal activities. In contrast, the anomaly and benign-based approach involves analyzing both malware files and benign files, making it a superior approach for distinguishing between benign and malicious activities. This approach captures both normal and malicious activities. However, training the detector in this approach is more time-consuming compared to the anomaly-based approach. Heuristic techniques serve as an extension of behavior-based malware detection methods. In comparison to traditional malware detection methods, machine learning plays a significant role in effectively detecting complex malware.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

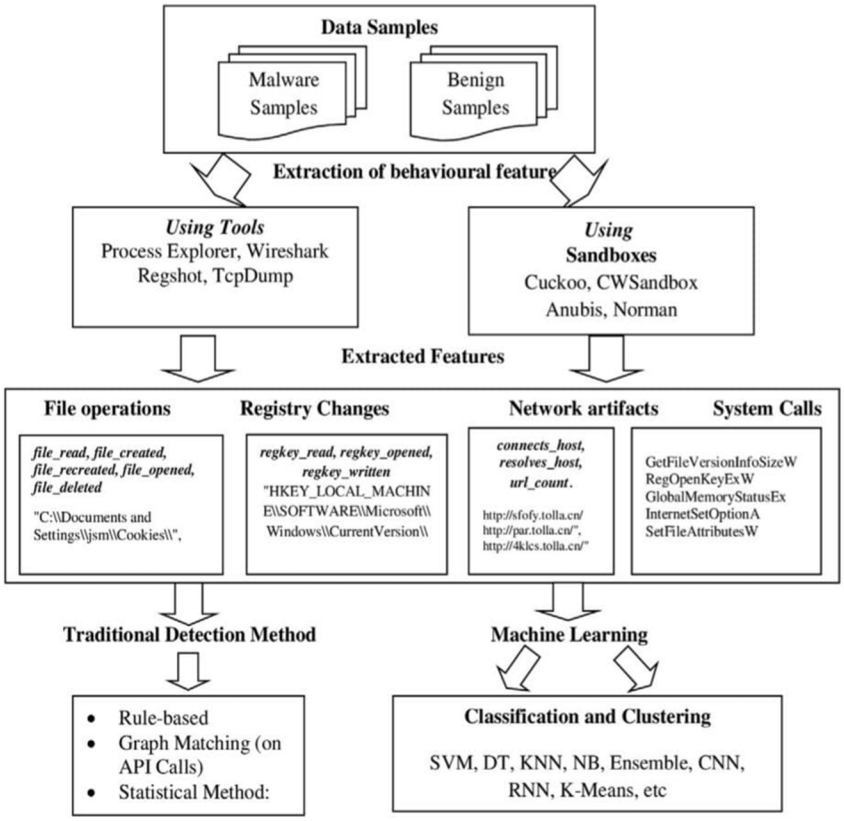


Fig. 5 Behavioural and machine learning based Architecture

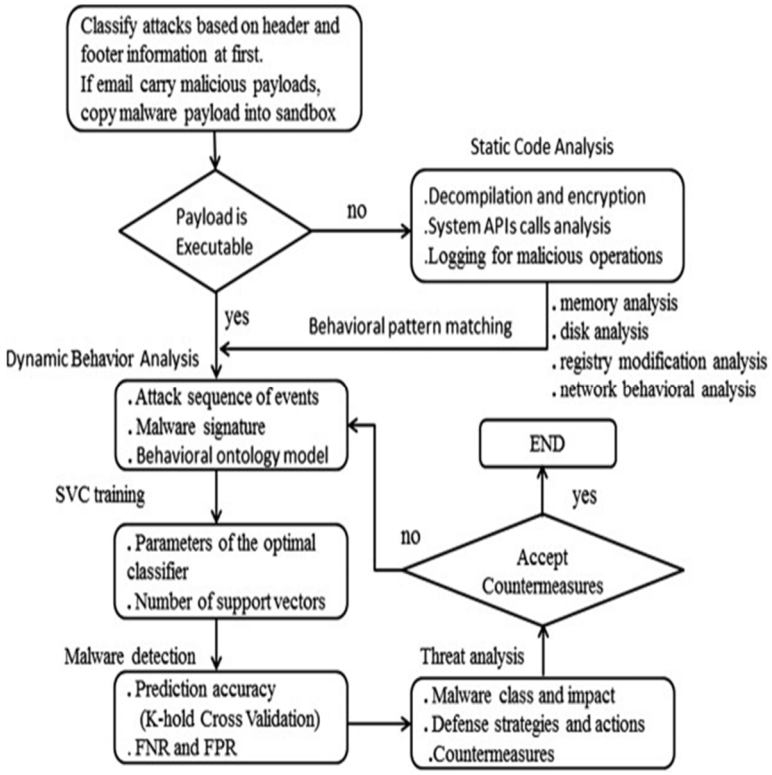


Fig. 6 Use Case Diagram

V. CONCLUSION

This project aims to enhance malware detection by utilizing run-time features. Malware is known for its complexity and rapid evolution. Malware analysis involves extracting valuable information from malware to detect and classify it. Two main techniques, static and dynamic analysis, are used for malware analysis. Signature-based (antivirus software) and behaviour-based anti-malware systems have been developed based on these techniques. However, signature-based techniques face two major challenges: they cannot detect new or unknown malware, and they can be easily evaded by malware variants. On the other hand, behaviour-based techniques are capable of detecting new and variant malware, and dynamic techniques are more resilient against malware obfuscation compared to signature-based techniques. Nonetheless, implementing dynamic techniques can be inflexible and time-consuming, while signature-based techniques are fast and effective in detecting known malware.

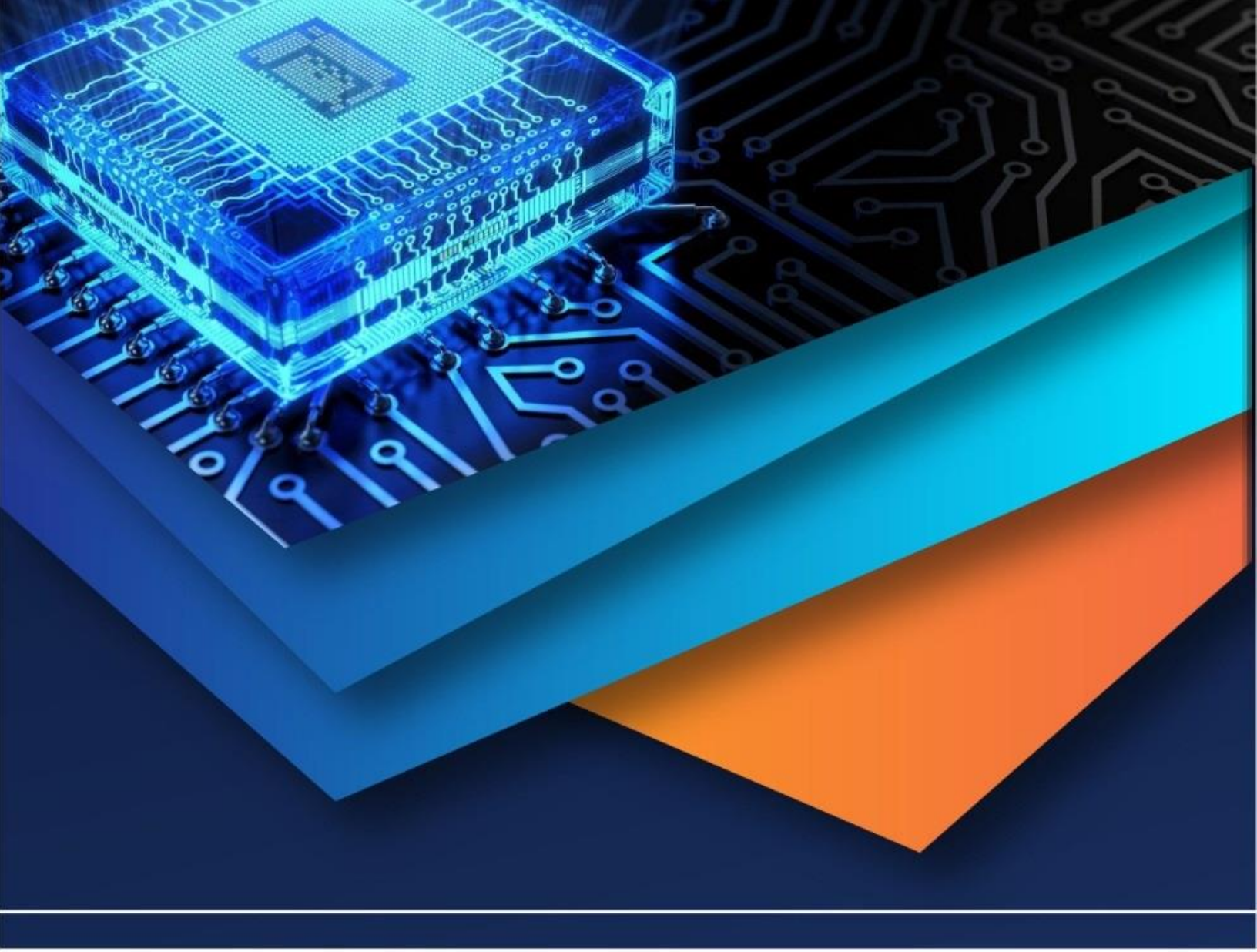
VI. ACKNOWLEDGEMENT

We're really indebted to D.Y Patil Institute of Engineering and Technology, Ambi, Pune for providing us an opportunity to undertake this project work as partial fulfilment of the BACHELOR's Degree in BACHELOR OF ENGINEERING curriculum. We would like to express our heartiest gratitude to Prof. Madhavi Patil and all the faculties of the BE Computer Department for their encouraging support and guidance in carrying out this project. We express our sincere thanks to D.Y Patil Institute of Engineering Technology, Ambi, Pune for permitting us to take this project work and for them instance of the good programming technique, which helped us to design and develop a successful MALWARE DETECTION USING ML. Finally, sincere thanks to our project members, mentors and all well-wishers for their esteemed guidance, support, valuable suggestions and constructive criticism.

REFERENCES

[1] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, MalDAE : Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics, Computer. Secure. (2019) 208–233, http://dx.doi.org/10.1016/j.cose.2019.02.007.
[2] P. Burnap, R. French, F. Turner, K. Jones, Malware classification using self-organising feature maps and machine activity data,Comput. Secur.73 (2017) 399–410, http://dx.doi.org/10.1016/j.cose.2017.11.016, http://linkinghub.elsevier.com/retrieve/pii/S0167404817302535.
[3] A. Damodaran, F.D. Troia, C.A. Visaggio, T.H. Austin, M. Stamp, A comparison of static, dynamic, and hybrid analysis for malware detection, J. Comput. Virol. Hacking Tech. 13 (1) (2017) 1–24, http://dx.doi.org/10.1007/s11416-015-0261-z.
[4] E.M. Dovom, A. Azmoodeh, A. Dehghantanha, D.E. Newton, R.M. Parizi, H. Karimipour, Fuzzy pattern tree for edge malware detection and categorization in iot, J. Syst. Archit. 97 (March) (2019) 1–7, http://dx.doi.org/10.1016/j.sysarc.2019.01.017
[5] M. Ficco, F. Palmieri, Leaf : An open-source cybersecurity training platform for realistic edge-iot scenarios, J. Syst. Archit. 97 (September 2018) (2019) 107–129, http://dx.doi.org/10.1016/j.sysarc.2019.04.004.
[6] K. Khan, A. Mehmood, S. Khan, M.A. Khan, Z. Iqbal, W.K. Mashwani, A survey on intrusion detection and prevention in wireless ad - hoc networks, J. Syst. Architecture. (2019) 101701, http://dx.doi.org/10.1016/j.sysarc.2019.101701.
[7] A. Bushby, F. Cybersecurity, How deception can change cyber security defences, Comput. Fraud Secur. Bull. 2019 (1) (2019) 12–14, http://dx.doi.org/10.1016/S1361-3723(19)30008-9.
[8] E. Gandotra, D. Bansal, S. Sofat, Malware analysis and classification: A survey,J. Inf. Secure.05 (02) (2014) 56–64, http://dx.doi.org/10.4236/jis.2014.52006.
[9] Raff, E., Barker, J., Sylvester, J., Brandon, T., Catanzaro, B., Nicholas, C. K., ... & Brandon, T. (2019). Malware detection by eating a whole exe. arXiv preprint arXiv:1806.04687.
[10] Chen, J., Jia, K., Chen, X., Cao, H., & Lu, Y. (2022). MalDroid: Android malware detection using convolutional neural networks. IEEE Transactions on Dependable and Secure Computing, 19(2), 405-418
[11] Zhang, H., Shen, Y., Li, X., Yu, S., & Lai, X. (2021). Malware detection based on deep learning with multiple tasks learning. IEEE Access, 9, 58207-58217.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)