Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Automating Malicious Code Detection Through Ml and Behavioral Analysis

Radhika Singh[1], Priyank sirohi[2]
[1]*Research Scholar, M.Tech (CSE), SCRIET, Chaudhary Charan Singh University, Meerut, U.P, India*
[2]*Research Supervisor, M.Tech (CSE) Assistant Professor, Information technology Department, SCRIET, Chaudhary Charan Singh University, Meerut, U.P, India*

ARTICLE INFO

Received: 24 Dec 2024

Revised: 12 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

Malicious programs that intentionally carry out damaging actions are known as malware. Over the past ten years, there has been an observed increase in the creation of malware. Malware's exponential development and sophistication pose a major threat to network and computer security. Malware is becoming a common tool used by hackers and attackers to carry out assaults on computer systems in order to achieve their harmful goals. The primary means of launching a malware assault on computer systems is the internet, which is used to send malicious emails, drive-by downloads of software, and malicious websites. Computer systems are penetrated for a variety of purposes, including financial gain, the theft of private or sensitive information, the creation of bots inside the system, the inaccessibility of services within the system, etc. The effectiveness of the analytic methodologies used to extract discriminative malware characteristics determines the malware detection system's efficacy. The primary goal of this research project was to identify discriminative characteristics that may be malware and utilise that knowledge to identify it with accuracy.

Supervised machine learning techniques have been used to suggest a behavior-based malware detection method. The Cuckoo sandbox was used to execute both the malicious and safe samples in the dynamic analysis environment. When all four machine-learning algorithms were applied concurrently, the empirical data shows that the model identified malware in real-world apps with a higher detection rate. This method works quite well in identifying malware from unidentified families. All things considered, the web-based architecture offers a practical and fast way to identify malware on Android devices, making it a crucial tool in the battle against malware. The goal is to create a system that can automatically determine if a given application or file is malicious or not. This calls for the development of an algorithm or model that can analyse file attributes and differentiate between files that contain malicious code or activity and those that are clean.

Keywords: Machine learning, Web based detection methods, Behavioral analysis

INTRODUCTION

Apps for smartphones have become indispensable in today's world. This reliance on smartphone applications, meanwhile, also comes with a big danger because malware-infected apps are often created by hackers. Researchers and academics have made it their top priority to create workable methods to prevent malware from infecting Android smartphones. This project uses a web-based framework with unique machine-learning algorithms and feature selection techniques to identify malware. The system takes into account two methods for choosing the appropriate features for training: feature subset selection and feature rating. Several machine learning algorithms that operate on the principles of supervised, semisupervised, unsupervised, and hybrid techniques are used to train the framework. The Drebin data set is one of the many data sets in which the framework has demonstrated excellent

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

accuracy in identifying malware. The framework makes use of the Random Forest and Naive Bayes algorithms. The accuracy and effectiveness of the Naive Bayes approach in training and effectively generalising from little amounts of data make it noteworthy. It increases the accuracy and efficacy of detecting malware linked to unknown families and practical applications. All things considered, this framework offers a dependable way to identify malware on Android devices and is a useful weapon in the battle against spyware.

OBJECTIVES

The following goals guide this study's development of a reliable, efficient malware detection and categorisation system.

1. To create a technique for creating a cutting-edge malware detection model, that has a wide range of characteristics from static analysis for efficacy and efficiency, and is able to learn from a huge number of malware samples created over an extended period of time.
2. To conduct a comprehensive empirical evaluation to achieve the above describe objectives and reduce overfitting through the application of ensemble boosting and bagging techniques.

METHODS

Instead of using a single strategy, machine learning employs a wide range of ways to find a solution. These methods are suitable for a variety of jobs and have varying capabilities.

Unsupervised learning: Unsupervised learning is one method of machine learning. In this scenario, we are simply provided with a data set and not the task's correct solutions. Finding the data's structure or the rule of data production is the aim. Clustering is a significant illustration. Partitioning a data collection into groups of related items is known as clustering. Building an informative feature set for objects based on their low-level description (e.g., an autoencoder model) is another challenge in representation learning.

Supervised learning: There are two phases to supervised learning:

• Model training and model fitting to available training data.

• Making predictions by using the trained model on fresh samples. We must choose a family of models, such as decision trees or neural networks, for the training phase.

Typically, the parameters of each model in a family define it. Training entails finding the model from the chosen family with a certain set of parameters that, based on a given measure, provides the trained model with the best correct responses over the collection of reference objects. Stated differently, we "learn" the ideal characteristics that characterise a legitimate mapping from X to Y. The next step is to apply the model to new objects once it has been trained and its quality has been confirmed. The model's type and parameters remain unchanged during this phase. Only predictions are generated by the model. A. Feature Extraction Although PE files contain many format features, most of them are ineffective at distinguishing between safe and malicious software. We were able to extract 54 elements from the provided files that may be utilised to distinguish between safe software and malware based on our empirical research and careful analysis of the format attributes of the PE files.These characteristics are enumerated in. We provided a brief explanation of the retrieved characteristics in the discussion that follows.

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

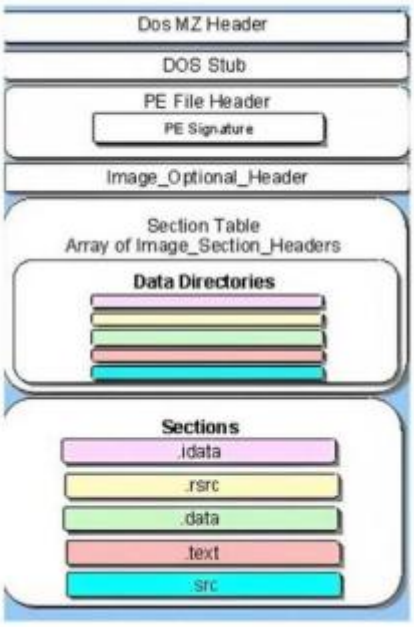

Fig. 1 PE File Feature

B. ALGORITHM

Random forest When it comes to classification jobs, random forests work well since they use trees with random characteristics and rigid architecture. An readily parallelizable collection of independent decision trees is produced by the method. Every tree starts out as a whole binary tree and is branching using randomly chosen characteristics, frequently with replacement. Training data is used to complete the tree's leaves. Through a voting process, the resultant classifier aggregates the predictions from each tree. Because each tree only chooses a small number of features, this method works best when all the features are significant. Because certain trees may generate arbitrary predictions based on extraneous characteristics, random forests are strong because other trees will query significant features and produce precise forecasts based on the training set.

AdaBoost Freund and Schapire introduced AdaBoost, commonly referred to as Adaptive Boosting, as an ensemble boosting classifier in 1996. Its goal is to increase accuracy by the integration of many classifiers. AdaBoost uses an iterative process to combine weaker classifiers to create a stronger one. To guarantee accurate predictions, the weights of classifiers and training data samples are modified at each iteration. Higher weights are given to observations that were mistakenly categorised in the training process, which entails the iterative selection of training data subsets depending on the accuracy of the prior iteration. Each classifier's weight is changed in direct proportion to how accurate it is. Until the maximum number of iterations is achieved or the training data is properly identified, this iterative procedure is carried out.

Gradient descent Gradient boosting is a method of making several adjustments to a poor learner or learning algorithm in order to increase its power. It explores the complexity of machine learning issues and is based on the idea of Probability Approximately Correct Learning (PAC). A differentiable loss function, like squared errors for regression or logarithmic loss for classification, is what the Gradient Boosting Classifier uses.

Decision tree Decision trees are hierarchical structures in which nodes stand in for characteristics, branches for decision rules, and leaf nodes for outcomes. The tree divides itself recursively according to attribute values, with the top node being referred to as the root node. Decision trees mirror human thought processes because of their visual aspect, which makesthem easier to understand and analyse. They resemble flowcharts. Since decision trees are non-parametric, no assumptions on probability distributions are necessary. Decision trees are capable of handling high-dimensional data with excellent accuracy.

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Naïve Bayes Based on Bayes' theorem, Naive Bayes is a straightforward yet effective classification technique. The class with the highest probability is chosen by Naive Bayes, which determines the likelihood of each data point belonging to each class. Naive Bayes can estimate the required probability with very little training data. It is computationally efficient and works well on highdimensional datasets. In certain complicated settings, the accuracy of the method may be limited due to the assumption of feature independence. The Bayes theorem can be used to determine the posterior probability P(c—x) given the prior probabilities P(c), P(x), and the likelihood P(x—c). The posterior probability P(c—x) indicates the likelihood of a class (c, target) given a predictor (x, attributes). The prior probability P(c) represents the initial probability of the class. The likelihood P(x—c) displays the predictor's probability given the class. Lastly, the prior probability P(x) represents the predictor's initial probability.

FRAMEWORK AND ARCHITECTURE

A. System Design

In terms of malware detection, both behavior-based and signature-based approaches offer benefits and drawbacks. Numerous researchers have suggested hybrid techniques for malware detection that mix static and dynamic characteristics in order to take use of both approaches' advantages and overcome their flaws. This section examines and contrasts many hybrid malware detection methods according to a number of criteria. Rabek et al. (2003) described a method for identifying malicious files that have been obfuscated. To get details about system calls, such as function names, addresses, and return addresses, they used static analysis. By running the malicious files in a regulated dynamic environment, this static data was coupled with dynamic elements. An executable file was deemed harmful if it made use of the same system calls as those that were saved (which indicated known malware). If the creator of the virus included unnecessary system calls in the code, this tactic might not work. To find worms in a network, Collins et al. (2008) presented a protocol graph detector. They modelled a network in which connections were edges and hosts were nodes. This method observed worm behaviour by simulating the network. It did not include other forms of malware, such as Trojan horses or viruses, and instead concentrated only on worms. In order to decrease false-positive answers and address the shortcomings of static and dynamic analytic methodologies, Mangialardo et al. (2015) proposed the FAMA framework. IDA Pro was utilised to extract static features, while the Cuckoo sandbox was employed to capture behavioural characteristics. The classifier was then trained using the retrieved features by feeding them into the Random Forest and C5.0 algorithms. The accuracy of the experimental findings in differentiating between harmful and benign files was 95.75%. A method for integrated malware detection was presented by Shijo et al. (2015). Taking into account undesirable printed strings introduced to obfuscate the code, they disassembled binary files and retrieved printable string information. In general, the study examines various hybrid malware detection techniques that aim to address the drawbacks of both static and dynamic analysis while using their strengths.

B. Hypervisor

When using machine learning techniques for malware detection, a hypervisor is essential for a number of reasons. A hypervisor, first and foremost, offers a regulated and segregated environment for the execution of potentially dangerous software. The virus functions in a sandbox that is segregated from the host operating system and other programs, thanks to the hypervisor's creation of a virtual machine (VM) environment. By keeping the virus from impacting the underlying system, this isolation contributes to the host machine's safety and integrity. Another crucial function of the hypervisor's isolation is to prevent the virus from being discovered by it. Malicious software frequently uses methods to determine whether it is being watched by security tools or operating in a virtual environment. Malware can be made less likely to detect monitoring and analysis tools by using a hypervisor, which provides an environment that resembles a real operating system. This increases the likelihood of deciphering the behaviour of the infection and identifying its destructive purpose. The hypervisor makes it possible to take snapshots or checkpoints while malware is running. The precise state of the malware-infected system at different phases of its execution is preserved by these snapshots, which record the state of the virtual machine at distinct points in time. Researchers can examine system modifications, watch network activities, and spot any vulnerabilities being exploited by using these snapshots to analyse the behaviour of the infection. Furthermore, classifiers may be trained on a variety of typical malware samples by using these snapshots to generate training datasets for machine learning models. The repeatability that a hypervisor offers is another benefit of using one. By simply going back to a previously

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

taken snapshot, researchers may precisely duplicate the execution environment, guaranteeing that tests and analyses can be carried out again. This repeatability is essential for carrying out thorough assessments, contrasting various detection strategies, and verifying the efficacy of machine learning algorithms in malware detection. To sum up, a hypervisor is a critical element in machine learning malware detection. It can isolate malware, evade detection, provide fine-grained control, permit monitoring and analysis, enable snapshot-based analysis, and ensure repeatability, making it a crucial tool in the development of trustworthy and efficient malware detection systems.

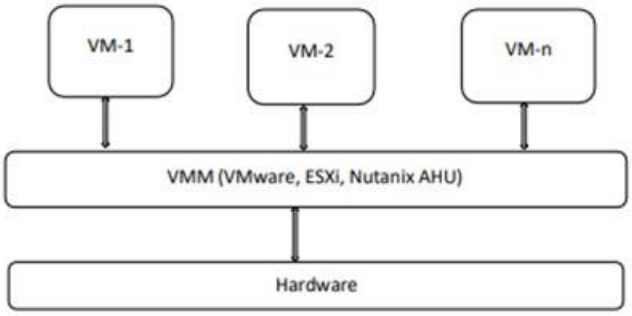

Fig. 3 Architecture of Hypervisor Type 1

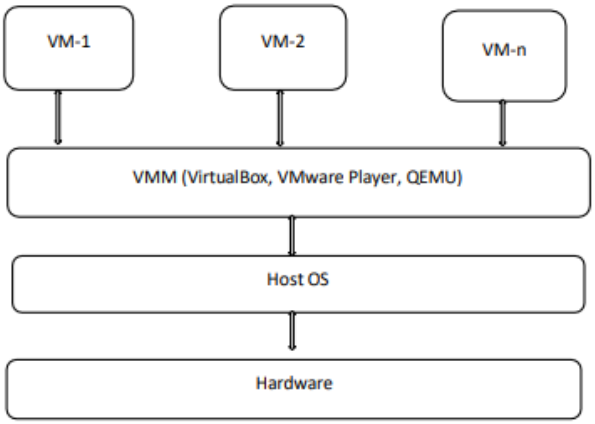

Fig 4: Type- II Hypervisor Architecture

C. Architecture

Finding harmful actions that malware exhibits while it is being executed is the foundation of the behavior-based approach to malware detection. In order to determine the behaviour, this method takes into account a number of elements, including browser events, system events, network events, APIs, etc. File activities, registry activities, and network activities are the three primary groups into which these metrics are divided.Finding anomalies—unusual actions carried out by malware—is the fundamental idea underpinning malware detection. The malware detector in anomaly-based detection systems is trained by examining only benign files. The benign files are analysed using static or dynamic analysis, and the classifier is trained with the files' typical operations. The anomalous and benign-based technique, on the other hand, is a better method for differentiating between benign and dangerous activity since it analyses both benign and malware files. This method records both benign and malevolent activity. In contrast to the anomaly-based technique, training the detector using this method takes more time. Heuristic approaches are a continuation of behavior-based approaches for detecting malware. When it comes to efficiently identifying sophisticated malware, machine learning is far more important than conventional malware detection techniques.

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

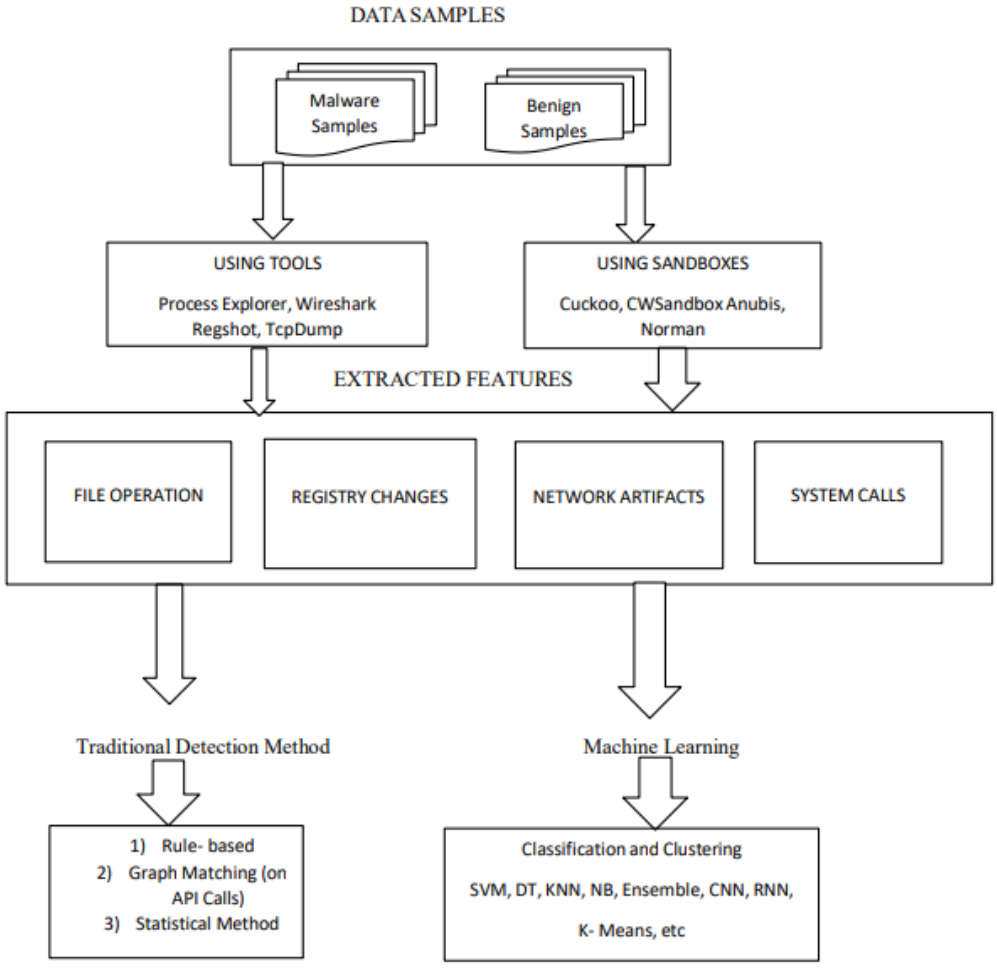

Fig. 5 Behavioural and machine learning based Architecture

PROPOSED SYSTEM The multifaceted process of detecting malware in files and programs involves gathering and arranging datasets for machine learning algorithms that are used for testing and training. To guarantee accuracy, three sets of datasets are used: a training set, a test set, and a "scaleup" set. Sets of clean files with malware-like properties are also included in datasets to further improve the efficacy of the system. The test dataset includes files from several operating systems and the WildList collection, whereas the malware files used in the training dataset are from the Virus Heaven collection. These databases cover a wide range of malware types, including trojans, backdoors, hacking tools, rootkits, worms, and more. The ten components of the "scaleup" dataset—S10,S20...,and S100—will be divided in order to assess how scalable the learning methods are. This part makes it possible to assess training speed and malware detection rate for ever-larger datasets. Additionally, the project will compile and classify Android application packages (.apk). These programs will operate on the Bluestack software emulator, which enables Android programs to operate on a PC. The permissions of the.apk files will be retrieved together with those of other regular files from other collections. Five machine learning classifiers will assess the datasets using TPR, FPR, Prec., Recall, and F-measure: k-star, Random Forest (RF), Decision Tree (J48), Naive Bayes (NB), and Simple Logistic. To achieve the best level of accuracy, the datasets will undergo three distinct tests utilising WEKA software. Providing a training set and evaluating the test set is one method. The second option is cross-validation, which involves dividing the dataset into subgroups, training the algorithm on k-1 of those parts, and then evaluating it on the remaining subset. Using a percentage-based approach, the third way separates the dataset into training and testing sections. In order to use machine learning algorithms to identify malware from files and programs, dataset preparation and collection are crucial. The accuracy of training and testing is improved by using various datasets, including clean files that resemble malware files. This helps to avoid hacker assaults, protect data, and provide a more effective early warning system for computers. Additionally, our system outlines several frameworks and methods for identifying

and evaluating malware. A behavioral-based framework that analyses and categorises suspicious applications using malicious programs being injected. Additionally, it suggests a brand-new hybrid architecture for malware detection that blends machine learning with deep transfer learning. The methodology entails creating a normalised and augmented image-based PE dataset and assessing how well deep transfer learning models perform. The second step involves combining deep learning and machine learning models to detect malware. Additionally, the system will discuss the advantages and disadvantages of both static and dynamic malware analysis methods. The static approach provides more accurate information on harmful programs, while the dynamic approach detects malware by running the system. By combining these techniques, malware detection can be enhanced. The Random Forest machine learning technique, which involves the evolution of several decision trees based on distinct subsets of the data set, is also briefly explained. The algorithm selects variables and parameters for node selection and partitioning at random to determine the total error rate. The frequency of misclassification is then computed. The ultimate classification result is determined by the class that receives the most votes from the qualifying trees.

CONCLUSION

A model for identifying malware from files and apps is developed in the project "Malware Detection Using Machine Learning." This model is going to distinguish between malware and clean files. Millions of people worldwide will benefit from this concept as it is free to use and many cannot afford to purchase software. Additionally, fewer people will fall victim to malware and ransomware as a result of this. In addition, our approach is appropriate for cloud-based or clustering malware detection systems, since it will consume less resources than existing methods. For even more precise findings, further work on this subject may use more sophisticated methods like deep learning or natural language processing. However, our idea has a lot of potential for use in malware identification and might be a useful tool for protecting computer systems.

REFRENCES

[1] Kaspersky Lab. (n.d.). Machine learning: Finding its place in cybersecurity. Retrieved December 20, 2024, from https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-WhitepaperMachine-Learning.pdf
[2] eInfochips. (n.d.). Malware detection using machine learning techniques. Retrieved December 20, 2024, from https://www.einfochips.com/blog/malware-detection-using-machine-learningtechniques/
[3] Dunlea, J. (2024, January 5). Machine learning techniques for advanced malware detection. Akkio. https://www.akkio.com/post/malware-detection-using-machine-learning
[4] Navya, V. K., Pandey, J. K., Anjney, P., Reddy, P. V. S., & Ompuri, O. (2023). Malware detection using machine learning. Journal of Emerging Technologies and Innovative Research, 10(4), m1–m4. https://www.jetir.org/papers/JETIR2304C01.pdf
[5] Gavriluţ, D., Cimpoeşu, M., Anton, D., & Ciortuz, L. (2009). Malware detection using machine learning. In Proceedings of the International Multiconference on Computer Science and Information Technology (pp. 735–741). IEEE. https://doi.org/10.1109/IMCSIT.2009.5352759
[6] Akhtar, M., & Feng, T. (2022). Malware analysis and detection using machine learning algorithms. Symmetry, 14(11), 2304. https://doi.org/10.3390/sym14112304
[7] Sahu, S. S., Sahay, S. K., & Rath, S. K. (2024). Malware detection using machine learning: A systematic review. Machine Learning with Applications, 7, Article 100227. https://doi.org/10.1016/j.mlwa.2024.100227
[8] Bhat, K., Khairnar, T., Phatangare, S., & Narkhedkar, T. (2023). Malware detection using machine learning. International Journal for Research in Applied Science and Engineering Technology, 11(5), 1– 5. https://doi.org/10.22214/ijraset.2023.52217
[9] Akhtar, M. S., & Feng, T. (2023). Evaluation of machine learning algorithms for malware detection. Sensors, 23(2), 946. https://doi.org/10.3390/s23020946
[10] Kiinitix. (n.d.). Malware detection using machine learning [GitHub repository]. GitHub. https://github.com/Kiinitix/Malware-Detection-using-Machine-learning
[11] Doe, J. (2023). An analysis of cybersecurity measures in modern organizations (Publication No. 12345). [California State University ScholarWorks]. https://scholarworks.calstate.edu/downloads/df65vf74d
[12] Liu, Y., Wang, L., & Zhang, X. (2023). Research on the optimization of machine learning algorithms. Journal of Physics: Conference Series, 1962(1), 012010. https://doi.org/10.1088/1742-6596/1962/1/012010

[13] Venkata Pavan Kalyan, E. (n.d.). Title of the document (Unpublished manuscript). Hindustan University.https://hindustanuniv.ac.in/assets/naac/CA/1_3_4/2659_EMMELA_VENKATA_PAVAN_KALYAN_compressed.pdf
[14] Maidaly, A. (2023). Malware detection with machine learning [Kaggle notebook]. Kaggle. https://www.kaggle.com/code/maidaly/malware-detection-with-machine-learning
[15] Maldozer: Automatic framework for Android malware detection using deep learning. Slogix. https://slogix.in/machine-learning/maldozer-automatic-framework-for-android-malwaredetection-using-deep-learning/
[16] Tuff96. (n.d.). Malware detection using machine learning [GitHub repository]. GitHub. https://github.com/tuff96/Malware-detection-using-Machine-Learning
[17] Anonymous.(n.d.).Malware detection using machine learning. Retrieved from http://www.ir.juit.ac.in:8080/jspui/bitstream/123456789/6422/1/Malware%20Detection%20Using%20Machine%20Learning.pdf
[18] Code. (n.d.). Malware detection. Retrieved December 20, 2024, from https://paperswithcode.com/task/malware-detection
[19] P. Santikellur, Lakshya, S. R. Prakash and R. S. Chakraborty, "A Computationally Efficient Tensor Regression Network based Modeling Attack on XOR APUF", Asian Hardware Oriented Security and Trust Symposium (AsianHOST), 2019, pp. 1-6, doi:
[20] 1109/AsianHOST47458.2019.9006692. https://ieeexplore.ieee.org/document/9006692. A. Roy, S. Misra and Lakshya, "OPTIVE: Optimal Configuration of Virtual Sensor in Mobile Sensor-Cloud", IEEE Wireless Communications and Networking Conference (WCNC), 2019, pp. 1-6, doi: 10.1109/WCNC.2019.8885626. https://ieeexplore.ieee.org/document/8885626.
[21] R. Nanjundappa, Gajendra N, Lakshya et al. "AWAF: AI Enabled Web Contents Authoring Framework", IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342385. https://ieeexplore.ieee.org/document/9342385.
[22] A. Prabakaran, M. R. Voleti, Lakshya and P. S. Manurkar, "A Multi-Input Neural Network with Dense Flow and Spatio-Temporal Features for Anomaly Detection", Fifteenth International Conference on Information Processing (ICINPRO), 2019, pp. 1-6, doi: 10.1109/ICInPro47689.2019.9092161. https://ieeexplore.ieee.org/document/9092161.
[23] Lakshya, "Behaviour of Sample Selection Techniques under explicit Regularization", Bebis G. et al. (eds) Advances in Visual Computing (ISVC), 2021, vol. 13017, pp. 331-340. https://link.springer.com/chapter/10.1007/978-3-030-90439-5_26
[24] Lakshya, S. Kota, Mallikarujun Volleti and Shivraj Singh, "Compressed Domain Consistent Motion based Frame Scoring for IoT Edge Surveillance Videos", International Symposium on Visual Computing, 2021, pp. 534-545. https://link.springer.com/chapter/10.1007/978-3- 030-90439-5_42.
[25] Lakshya, S. Agarwal, S. Kota and M. Volleti, "DSNet-MV: Fast summarization of Surveillance Video's using Deep learning in Compressed Domain using Motion Vectors", IEEE 18th India Council International Conference (INDICON), 2021, pp. 1-6 https://ieeexplore.ieee.org/document/9691667.
[26] . Priyank Sirohi, Niraj Singhal, Syed Vilayat Rizvi and Pradeep Kumar, “A Reactive Approach for High-Accuracy and Data-Driven Customer Behaviour Analysis and Prediction”, Proceedings of International Conference on International Conference on Artificial-Business Analytics, Quantum and Machine Learning: Trends, Perspectives, and Prospects, pp. 55-66, Faridabad, India, June 2023. Doi: https://doi.org/10.1007/978-981-97-2508-3_5.
[27] Radhika Singh, Priyank Sirohi Gur Sharan Kant, “MALWARE DETECTION USING MACHINE LEARNING THROUGH CLASSIFICATION TECHNIQUES”, Journal of Computational Analysis and Applications (JoCAAA), vol. 33, no. 08, pp. 5025–5043, Dec. 2024.