IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024

DOI: 10.17148/IJARCCE.2024.13422

Enhanced Malware Detection Using Machine Learning Algorithms

Naveen Sundar Kumar P[1], Veera Prasad Singiri[2], Sujatha Perapogu[3], Yasaswini Kunam[4], Vamse Krishna Mallela[5]

Assistant Professor, Department of Computer Science and Engineering, Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, Andhra Pradesh, India[1]

Department of Computer Science and Engineering, Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, Andhra Pradesh, India[2-5]

Abstract: Malware detection is a critical aspect of cybersecurity, and the integration of machine learning techniques has shown promising results in enhancing detection capabilities. This paper proposes the integration of the Extreme Gradient Boosting (XGBoost) algorithm for detecting malware. XGBoost, known for its scalability and efficiency, has been successfully applied in various domains, including malware detection. By leveraging XGBoost, this research aims to improve the accuracy and efficiency of malware detection systems. Additionally, the XGBoost algorithm has been compared with other machine learning models like Random Forest and Adaboost, consistently showing superior performance in terms of detection accuracy. The proposed approach in this paper builds on the strengths of XGBoost to enhance the detection of diverse malware variants. By optimizing XGBoost parameters and leveraging ensemble learning techniques, this research contributes to advancing the capabilities of malware detection systems, thereby strengthening cybersecurity measures in the face of evolving cyber threats.

Keywords: Malware Detection, Xgboost, Machine Learning, Dynamic Analysis, DT,KNN,RF, Static analysis.

I. INTRODUCTION

Cybersecurity is seriously threatened by malware, as hackers are always developing new strategies to avoid detection and infiltrate networks. Sophisticated and polymorphic malware strains are frequently difficult to identify using conventional signature-based detection techniques [4]. Consequently, there is an increasing demand for more sophisticated and effective detection methods. Machine learning algorithms [3][1] have become an effective instrument for malware detection because of their capacity to examine vast amounts of data and spot trends that point to the presence of malicious activity.

1.1 Malware

Malware, short for malicious software, refers to any software intentionally designed to cause damage, disrupt, or gain unauthorized access to computer systems, networks, or devices. It encompasses a broad range of malicious programs including viruses, worms, Trojans, ransomware, spyware, and adware. Malware can steal sensitive information, corrupt files, degrade system performance, and even render systems or networks inoperable. It typically spreads through various means such as email attachments, infected websites, or removable media, posing significant threats to individuals, organizations, and society as a whole. As technology advances, malware continues to evolve in sophistication, requiring constant vigilance and proactive security measures to mitigate its risks.

1.2 Malware Detection Techniques

Malware detection techniques encompass a diverse array of methods aimed at identifying and mitigating the presence of malicious software within computer systems, networks, and devices. These techniques leverage a combination of signature-based [4], heuristic-based [6], behavioral-based, and machine learning-based [3][1] approaches to detect known and emerging threats. Signature-based [4] detection relies on predefined patterns or signatures of known malware to identify malicious files or code. Heuristic-based [6] detection employs algorithms to detect suspicious behavior or characteristics indicative of malware, even if the exact signature is unknown. Behavioral-based detection observes the actions of programs or processes to identify anomalous behavior that may indicate malware activity. Additionally, machine learning-based [3][1] techniques analyze large datasets to identify patterns and trends associated with malware, enhancing detection accuracy and adaptability to evolving threats

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024
DOI: 10.17148/IJARCCE.2024.13422

1.3 Key Findings

In this paper, a novel behavior-based malware detection technique is introduced, leveraging dynamic analysis within a Cuckoo sandbox environment. Both malware and benign samples are subjected to runtime analysis, generating comprehensive behavioral reports. Notably, the approach meticulously investigates printable strings at a granular word level using text mining methodologies, a significant departure from previous techniques. By constructing a matrix of words through count vectorization and applying Singular Value Decomposition (SVD) for dimensionality reduction, the research enhances the understanding of string features within binary files. Moreover, the assessment incorporates Shannon entropy calculations on API calls and Printable String Information (PSI) to gauge malware randomness, enriching the training feature set. Beyond these aspects, the paper integrates additional runtime features encompassing file, registry, network, and mutex activities, further enhancing the training feature set's comprehensiveness. Ultimately, the research employs machine learning algorithms, including XGBoost, alongside a comparative analysis with various other classifiers, to develop robust malware classifiers. This multifaceted approach not only highlights the efficacy of XGBoost but also underscores the novelty of the study in its comprehensive exploration of runtime features and meticulous analysis of printable strings, thus contributing significantly to the field of behavior-based malware detection.

1.4 Structural Layout

This research paper is structured as follows:
1. In Section II, an exploration of related literature is conducted.
2. Section III delves into the presentation of various machine learning algorithms employed in the paper.
3. The proposed technique is elucidated in Section IV.
4. Section V encompasses a thorough discussion on the experimental setup and resultant findings.
5. Finally, the paper is bought to a close with a conclusion.

II. RELATED WORK

In their paper, [1] "A Review on Malicious Software Detection using Machine Learning Algorithms" authored by Ravi Kumar Tirandasu and Dr. Y. Prasanth, the authors investigate the efficacy of various machine learning algorithms for detecting malicious software. They evaluate classifiers such as K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machines (SVM), Naive Bayes (NB), and Random Forest (RF) on a dataset obtained from Cuckoo Sandbox. Through their analysis, they find that Random Forest achieves the highest accuracy among the classifiers, with an accuracy level of 97.46%. However, the study acknowledges several limitations, including the reliance on a single dataset and the need for further validation across diverse datasets to ensure the generalizability of the findings approaches.

In the paper authored by H. Shahriar, M. Islam and V. Clincy [7], a novel approach to detect anomalous Android applications is proposed, leveraging Latent Semantic Indexing (LSI) to identify relevant permission categories and confirming behaviors through dynamic analysis in an emulator environment. The study utilized machine learning algorithms for dimensionality reduction and similarity computation, with an emphasis on cosine distance for relevance assessment. By applying this multifaceted approach, the paper achieved promising results in accurately identifying malware applications, with average accuracies ranging from 76% to 89% across different categories.

The paper [7] authored by Sunita Choudhary and Anand Sharma presents a comprehensive study on malware detection and classification using machine learning techniques, particularly focusing on behavior-based detection methods. Various machine learning algorithms such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Decision Trees, and Multi-layer Perceptron (MLP) are employed and evaluated for their effectiveness. Results indicate high detection rates, with SVM and J48 Decision Tree achieving detection rates above 90% and overall accuracies surpassing 90%. However, limitations may arise concerning the generalization of these techniques across diverse malware families and the adaptability of the models to rapidly evolving malware tactics and evasion techniques. Additionally, the study underscores the potential of machine learning algorithms over traditional signature-based methods, especially in the face of the increasing diversity and sophistication of malware threats.

The paper's [3] contributions were made by Xinning Wang and Chong Li. They developed a novel approach for detecting malware on Android platforms using a multiple dimensional, kernel feature-based framework and feature weight-based detection (WBD) method. The authors also developed a software agent for automated data collection and storage, enabling the scanning of thousands of benign and malicious apps. By examining 112 kernel attributes and prioritizing 70 significant features through dimensional reduction, they achieved higher classification accuracy compared to existing techniques. Specifically, memory- and signal-related features contributed to more precise classification, with memory-related features offering fine-grain policies for higher classification precision.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024
DOI: 10.17148/IJARCCE.2024.13422

Experimental results showed that their method achieved 94%-98% accuracy and 2%-7% false positive rate, outperforming existing techniques with reduced-dimensional features. However, limitations may arise regarding the generalization of the approach across diverse malware families and the adaptability to rapidly evolving malware behaviors and evasion techniques.

III. METHODOLOGY

3.1 Classification algorithms

Many classification techniques are essential in the field of malware detection for differentiating harmful software from benign software. Large datasets can benefit from the simplicity and computational efficiency of Gaussian Naive Bayes (GNB). K-Nearest Neighbors (KNN), on the other hand, depends on instance similarity but has difficulties in high-dimensional feature spaces commonly found in malware investigation. While they provide transparency, decision trees (DT) may not work well with complicated decision boundaries. Algorithms that use gradient boosting, including XGBoost and Gradient Boosting Machines (GBM), are very good at identifying complex correlations in malware datasets. The efficiency of their iterative refining process is further increased by XGBoost's improvements, which further improve model performance and scalability. Together, these algorithms—each with its own advantages—provide strong malware detection systems that are essential for protecting digital ecosystems from ever changing threats.

3.2 Dimensionality reduction algorithms

We highlight Singular Value Decomposition (SVD) as the dimensionality reduction approach used to improve string feature comprehension in binary data. SVD is a popular method for lowering a data set's dimensionality while maintaining its fundamental structure. To do this, the original matrix is broken down into three constituent matrices, which represent orthogonal factors. Processing and analysis can be done more quickly and effectively since the final representation highlights the most important patterns in the data.

The integration of Printable String Information (PSI) and Shannon entropy computations on API requests for evaluating malware unpredictability is also mentioned in the article. Although it isn't stated clearly, Shannon entropy is a measure of uncertainty or unpredictability in a dataset rather than a strategy for reducing dimensionality. That being said, adding entropy computations in addition to A thorough method for removing unnecessary dimensions from the dataset and extracting valuable information is shown by the use of SVD for feature engineering and selection.

Algorithm.1: Algorithm for Dimensionality Reduction

```
Function SVD_Dimensionality_Reduction(matrix M, int k):
  Input:
    - M: Input data matrix of shape (m, n)
    - k: Number of dimensions to retain

  Output:
    - Reduced matrix of shape (m, k)

  Step 1: Perform Singular Value Decomposition (SVD) on M
    U, Sigma, V_transpose = SVD(M)
    # U: Left singular vectors matrix
    # Sigma: Singular values diagonal matrix
    # V_transpose: Right singular vectors transpose matrix

  Step 2: Select top-k singular values and corresponding columns of U
    Sigma_k = Top_k_Singular_Values(Sigma, k)
    U_k = Select_Corresponding_Columns(U, k)

  Step 3: Compute the reduced matrix
    Reduced_Matrix = U_k * Sigma_k

  Return Reduced_Matrix
```

Overall, the paper's dedication to using cutting-edge methods for dimensionality reduction and feature engineering in the context of malware detection is demonstrated by the use of SVD in conjunction with entropy estimates.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024

DOI: 10.17148/IJARCCE.2024.13422

IV. PROPOSED WORK

The system architecture for the suggested malware detection technique is shown in Fig. 1. It consists of six steps, each of which is thoroughly explained.

4.1 Sample Acquisition

The first thing the system does is gather samples of possible malware. These can come from a variety of places: people who think their devices may have malware on them can send data for examination.

It is possible for security software to upload files that it finds suspicious automatically. Furthermore, honeypots—decoy systems intended to attract and seize malware—can add useful samples to the system's repertoire.

4.2 Cuckoo Sandbox Study

Once obtained, a Cuckoo Sandbox isolates the file and examines its behavior in a safe setting. While simulating an actual system, this virtual sandbox stops malware from harming the genuine system. The sandbox functions as a watchful observer, keeping an eye on all of the file's interactions, including file modifications, network connections, changes to the system registry, and system call (API) interactions with the operating system.

Cuckoo Sandbox is an open-source automated malware analysis system that is used to analyze and report on the behavior of suspicious files. It is designed to be a standalone tool that can be installed on a virtual machine or a physical machine.

Cuckoo Sandbox works by executing a suspicious file in a controlled environment, monitoring its behavior, and generating a detailed report that includes information about the file's actions, network traffic, and system changes. Once the analysis is complete, Cuckoo Sandbox generates a detailed report that includes information about the file's behavior. The report includes details such as:

File information (name, size, hash, etc.)

Behavioral analysis (system calls, network traffic, etc.)

Screenshots of the analysis environment

Memory dumps and other artifacts that report can be used for identifying the malware

4.3 Report Creation

Following the Cuckoo Sandbox analysis, a comprehensive report is produced. This report functions as a digital version of the investigation's case file, akin to that of a detective. It contains important information about the file, like its size and creation date. The report then delves further, tracking all connections made, including URLs browsed and data exchanged, and providing a detailed account of the file's network activities.

The file also documents any changes it makes to the system registry. The report may also include screenshots that were taken during the investigation (optional), process trees that show how the file created new processes, and more. This report offers a plethora of data for additional examination and training of the machine learning model, which is the next line of defense.

4.4 Feature Extraction

The Cuckoo Sandbox report serves as a goldmine of information for the machine learning model's training. However, this data needs to be translated into a format that the model can interpret. This phase, referred to as feature extraction, entails taking pertinent data points out of the report and formatting them appropriately. This could entail translating numerical data into useful values, such as the size of the file or the number of network connections.

Text data may also be extracted, such as suspicious URLs browsed, registry key names changed, or particular strings detected inside the file. Extracting visual features from screenshots taken during the sandbox analysis (if applicable) may occasionally be a part of advanced analysis.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024

DOI: 10.17148/IJARCCE.2024.13422

Algorithm.2: Algorithm for Feature Extraction

1. Define function submit_sample(sample_file) to submit the sample file.
2. Define function execute_dynamic_analysis(sample_file) to execute the binary files and save the behavioral reports.
3. Define function save_behavioral_reports(behavioral_reports) to save the reports.
4. Define function extract_features(behavioral_reports) to extract implicit features.
5. Set sample_file as input.
6. Call submit_sample(sample_file) to submit the sample file.
7. Call execute_dynamic_analysis(submitted_sample) to execute dynamic analysis.
8. Call save_behavioral_reports(behavioral_reports) to save the reports.
9. Call extract_features(behavioral_reports) to extract features.
10. Output: Store extracted features in API_System_Calls, PSI_List, File_Reg_List, Network_Artifacts, and Other_features lists.

4.5 Training the Machine Learning Model

The machine learning model is trained using the retrieved features and distinct datasets of classified malware and benign files. In essence, the training dataset is a compilation of actual cases from which the model will be trained. This is where the magic happens: the selected machine learning method, like Random Forests or Support Vector Machines (SVM), examines the features that were taken out of files that are known to be benign or infected with malware. Through this training process, the model has the ability to recognize patterns in the extracted data that distinguish malicious from lawful file activity. This gives the model the ability to forecast new, unseen files with confidence.

4.6 Decision Outcome

After training, the model can handle newly received files. The system pulls features from newly supplied files for analysis that are comparable to those from the training phase. The trained model is then fed these features. After examining the features, the model renders a verdict based on the patterns it has learnt from the training data: "malware" or "benign." Security experts can assess if the file is dangerous and take appropriate action based on this classification.

4.7 Training Data Gathering

The quality and completeness of the training data determine how effective the system is as a whole. One of the most important functions of security researchers is gathering different types of malware samples from different sources and carefully classifying them as "malware" or "benign." The machine learning model learns from this labeled data, which enhances its capacity to recognize novel and changing threats. By continuously adding new samples to the training data, the system remains alert and adjusts to the malware's ever-changing environment.

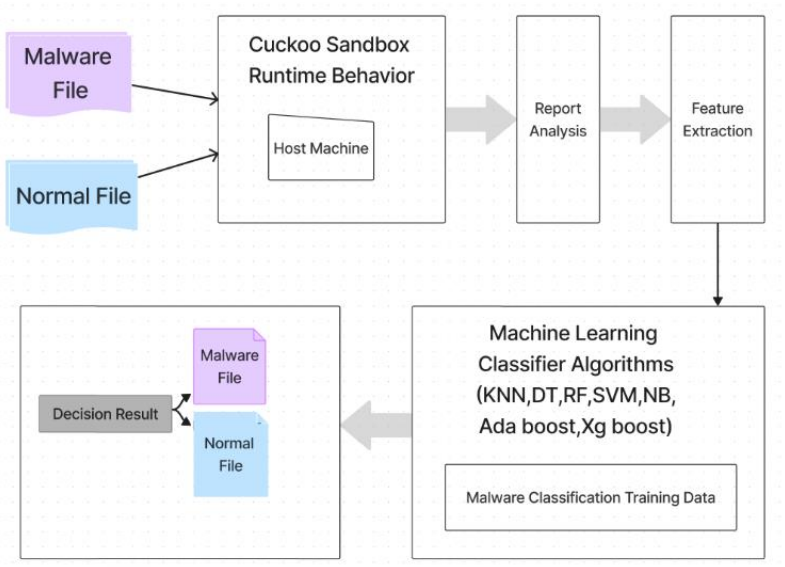

Fig.1: System architecture for malware detection

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024

DOI: 10.17148/IJARCCE.2024.13422

V. EXPERIMENTAL RESULTS

5.1 Data Sample Description

The API Call Sequence Dataset records the sequential pattern of API calls performed by processes or applications on a system. It contains information like timestamps, process IDs, API method names, parameters, and return values. These properties provide vital insights into process activity, allowing for the detection of anomalies or suspicious actions that indicate the existence of malware. Patterns such as process injection, privilege escalation, and unexpected file actions can be recognized by analyzing API call sequences. This dataset enables the construction of detection algorithms capable of recognizing malware based on its behavior patterns within the system's API calls.

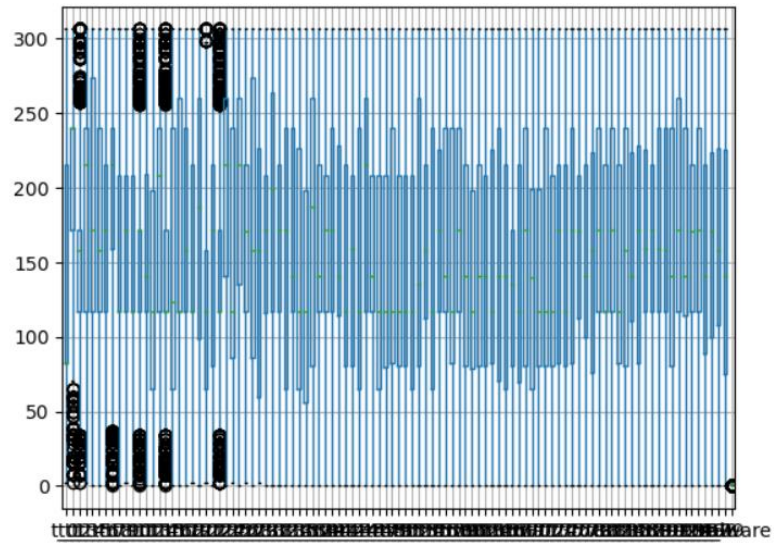

Fig.2: Boxplot for API Call sequence

The File, Registry, and Network Activity Dataset tracks file, registry, and network activity within a system. This dataset's attributes include activity kinds (e.g., file creation, registry change, network connection), file locations, registry keys, network addresses, ports, and protocol. Monitoring and recording these actions allows you to discover a variety of dangerous behavior, such as the creation or alteration of malicious files, illegal changes to registry keys, or unusual network connections linked with malware activities. Combining file, registry, and network activity data provides a full perspective of system behavior, making it possible to identify potential threats and design effective malware detection algorithms.

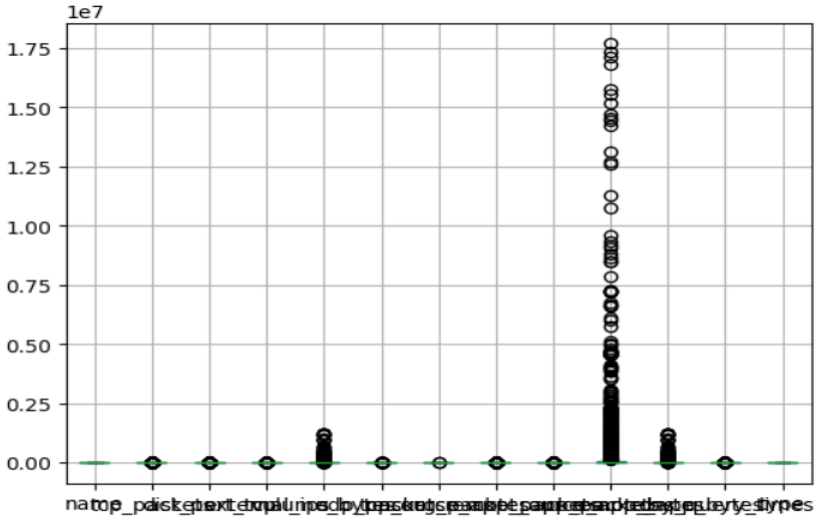

Fig.3: Box plot for File,Registry and Network activities

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024
DOI: 10.17148/IJARCCE.2024.13422

Algorithm	Accuracy	Precision	Recall	F1-score
NB	0.8893	0.9842	0.9008	0.9407
KNN	0.9843	0.9855	0.9985	0.9920
RF	0.9891	0.9895	0.9994	0.9944
XGBoost	0.9898	0.9903	0.9992	0.9948

Table.1: classification algorithm results using api call sequence.

Algorithm	Accuracy	precision	Recall	F1-score
NB	0.5069	0.5035	0.9558	0.6596
KNN	0.9190	0.9198	0.9228	0.9200
RF	0.9635	0.9653	0.9616	0.9634
XGBoost	0.9846	0.9137	0.9760	0.9845

Table.2: classification algorithm results using file, registry and network activities.

5.2 Results Comparison

The measures used to assess algorithm performance are accuracy, precision, recall, and F1-score.
Accuracy is defined as the ratio of correct predictions to total input samples. A high accuracy shows that the classifier is functioning properly.

Precision is calculated as the ratio of genuine positive predictions to total anticipated positives. High precision indicates that the classifier has a low false positive rate.

Recall is defined as the ratio of true positive predictions to total real positives. A high recall indicates that the classifier has a low false negative rate.

The F1-score is the harmonic mean of precision and recall, which assigns equal weight to both measurements. It offers a balanced appraisal of the classifier's performance.

Table 1 displays the performance results of various machine learning methods used to detect malware via API call sequences. The algorithms compared include Naive Bayes (NB), K-Nearest Neighbors (KNN), Random Forest (RF), and XGBoost.

According to the results, XGBoost outperforms the other algorithms in terms of Accuracy, Precision, Recall, and F1-score. This suggests that XGBoost is a suitable technique for detecting malware via API call sequences.

In summary, Table 1 compares four machine learning algorithms for malware detection based on API call sequences, with XGBoost outperforming the others. The algorithms are evaluated using criteria such as accuracy, precision, recall, and F1-score.

Table 2 compares the performance of various machine learning techniques for detecting malware based on file, registry, and network events. The algorithms compared include Naive Bayes (NB), K-Nearest Neighbors (KNN), Random Forest (RF), and XGBoost.

According to the results, XGBoost outperforms the other algorithms in terms of Accuracy, Precision, and F1-score. However, RF has a greater Recall score than XGBoost, implying that it has a reduced false negative rate.

In summary, Table 2 compares four machine learning algorithms for malware detection based on file, registry, and network activities, with XGBoost outperforming the others in terms of accuracy, precision, and F1-score. The algorithms are evaluated using criteria such as accuracy, precision, recall, and F1-score. Meanwhile, XGBoost has the highest in terms of accuracy, precision, and F1-score, RF has a higher Recall score, suggesting that it may be a superior alternative for applications where false negatives are extremely costly.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024

DOI: 10.17148/IJARCCE.2024.13422

5.3 Performance Metrics

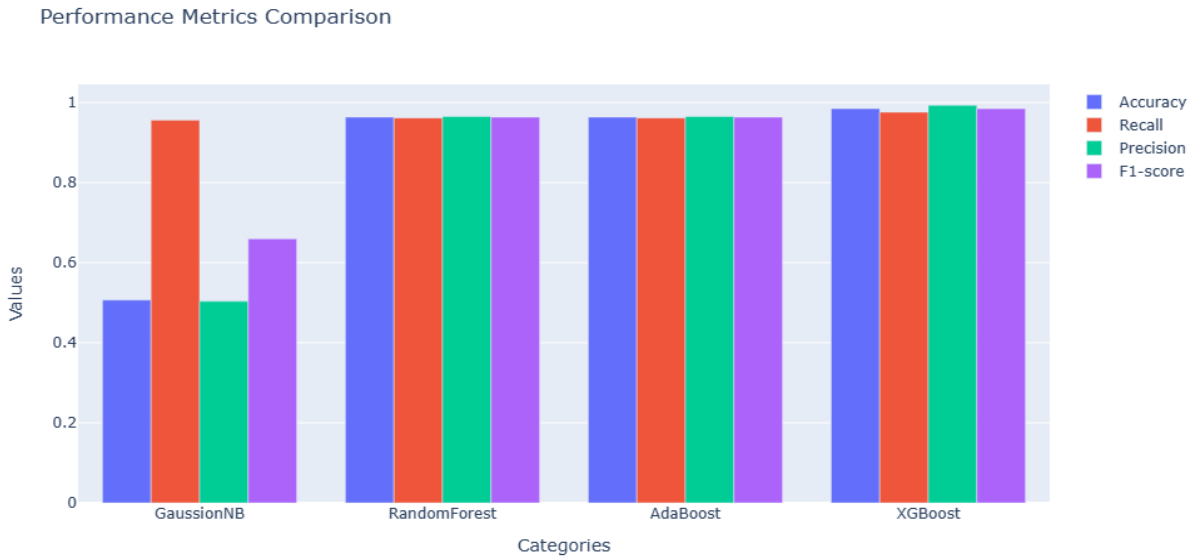

Fig 4: Performance Metrics Comparison

The image above compares the performance metrics of three machine learning techniques for malware detection: Gaussian Naive Bayes (GaussionNB), Random Forest (RandomForest), and XGBoost. The algorithms are compared using four performance metrics: accuracy, recall, precision, and F1-score.

The values in the graphic indicate the algorithms' performance on a malware detection test, with the categories identified as E. The x-axis depicts the performance measures, while the y-axis indicates their values.

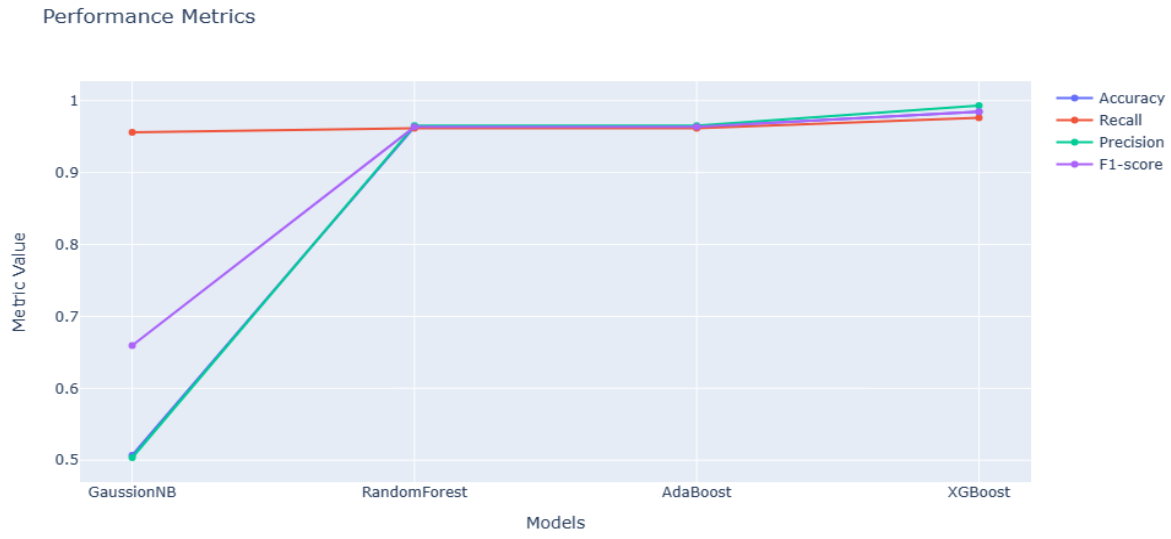

Fig.5: Performance Evaluation

The image shows that XGBoost outperforms the other two algorithms across all performance criteria. XGBoost achieves the greatest values among the three, with an accuracy of 0.9898, recall of 0.9992, precision of 0.9902, and F1-score of 0.9948.

In conclusion, based on the performance metrics comparison shown in the image, XGBoost is the best algorithm for malware detection, with the highest Accuracy, Recall, Precision, and F1-score.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2024
DOI: 10.17148/IJARCCE.2024.13422

VI. CONCLUSION

Finally, this research describes a novel behavior-based malware detection technique that uses dynamic analysis in a Cuckoo sandbox environment. By submitting both malware and benign samples to extensive runtime analysis, the strategy provides behavioral reports that rigorously explore printable strings at the word level, employing text mining methodologies—a considerable divergence from past efforts. The research improves understanding of string properties within binary files by building a word matrix using count vectorization and reducing dimensionality with Singular Value Decomposition (SVD).

Furthermore, the evaluation uses Shannon entropy calculations on API calls and Printable String Information (PSI) to measure malware unpredictability, expanding the training feature set. Furthermore, the study incorporates new runtime features such as file, registry, network, and mutex activities, broadening the scope of the training feature set. Finally, the study uses machine learning algorithms, such as XGBoost, as well as a comparative analysis with other classifiers, to create robust malware classifiers. This holistic approach not only illustrates the performance of XGBoost.

This proposed technique produced an accuracy of 98.98% using XGBoost algorithm, but also emphasizes the study's uniqueness in its thorough examination of runtime features and painstaking analysis of printed strings. As a result, our work makes a substantial contribution to the field of behavior-based malware detection by expanding our understanding of malware behaviors and enhancing detection accuracy.

REFERENCES

[1]. R. K. Tirandasu and Y. Prasanth, "A Review on Malicious Software Detection using Machine Learning Algorithms," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021, pp. 1945-1948, doi: 10.1109/ICESC51422.2021.9532700.
[2]. Wang, X., & Li, C. (2021). Android malware detection through machine learning on kernel task structures. Neurocomputing, 435, 126–150. doi:10.1016/j.neucom.2020.12.088
[3]. Li, J., Li, Q., Zhou, S., Yao, Y., & Ou, J. (2017). A review on signature-based detection for network threats. 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN). doi:10.1109/iccsn.2017.8230284
[4]. H. Shahriar, M. Islam and V. Clincy, "Android malware detection using permission analysis," SoutheastCon 2017, Concord, NC, USA, 2017, pp. 1-6, doi: 10.1109/SECON.2017.7925347.
[5]. S. Choudhary and A. Sharma, "Malware Detection & Classification using Machine Learning," 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), Lakshmangarh, India, 2020, pp. 1-4, doi: 10.1109/ICONC345789.2020.9117547.
[6]. Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013, May). A survey on heuristic malware detection techniques. In *The 5th* Conference on Information and Knowledge Technology(pp. 113-120).IEEE
[7]. Hassan, A., & Mendki, H. (2020). A deep learning approach for malware detection using API call sequences. Journal of Intelligent Information Systems, 1-21. DOI: 10.1007/s10844-020-00604-2.
[8]. Yakura, E., & Nishikawa, T. (2018). Malware detection using API call sequences and deep neural networks. Proceedings of the 2018 International Conference on Information Systems Security, 102-109. DOI: 10.1109/IMPACT.2018.00024
[9]. Ugarte, J. M., Fernández, A., & Viedma, J. (2015). An analysis of machine learning techniques for malware detection. Information Security Journal: A Global Perspective, 24(1-3), 18-28. DOI: 10.1080/19393555.2015.1040337
[10]. Raff, N., & Kim, D. (2017). Malware detection using deep learning.Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 507-512. DOI: 10.1109/ICSE.2014.122.
[11]. Saxe, J., & Berlin, J. (2015). Deep neural networks for classifying malware. Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security, 1365-1376. DOI: 10.1145/2810103.2813675.
[12]. Arp, D., Schneider, F., & Kirda, E. (2014, May). DREBIN: effective and explainable malware detection on android. In International Conference on Software Engineering (pp. 1223-1233). DOI: 10.1109/ICSE.2014.122
[13]. Kruegel, C., & Valeur, J. (2005). Malware detection using data mining. Proceedings of the 2005 IEEE Symposium on Security and Privacy, 15-29. DOI: 10.1109/SP.2005.1