JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Automated Malware Detection using Machine Learning Algorithms

B. Pragathi[1], S. Rahul[2], T.Thakshith Kumar[2],S. Varun[2], V.G. Suraj Kumar Yadav[2]
[1]Assistant Professor, Department of IT, Malla Reddy Engineering College, Hyderabad, Telangana, India
[2]B.Tech IV Year, Department of IT, Malla Reddy Engineering College, Secunderabad, Telangana, India

Abstract: The evolving landscape of computer technology has led to a significant shift towards virtual environments, accompanied by a rise in the threat posed by malware. Traditional methods of malware detection are struggling to keep pace with the increasing sophistication of malware variants, which often employ advanced packing and obfuscation techniques to evade detection. In response to this challenge, there is a growing interest in exploring new techniques for effectively combating malware. One promising approach is the utilization of deep learning methods, which offer the potential to detect complex and new malware variants more effectively than traditional machine learning methods. This abstract presents an overview of the current challenges in malware detection and highlights the potential of deep learning methods, particularly ensemble learning approaches, for enhancing cybersecurity measures.

KeyWords: *Malware, Cyber Security, Ensemble learning.*

1. INTRODUCTION

Cybersecurity is rapidly emerging as a critical area of concern for network engineers and computer scientists, driven by the continuous evolution of technology and its pervasive integration into various aspects of daily life. As a result, the detection and mitigation of malware threats have become paramount. Among the plethora of malware variants, malicious URLs have garnered significant attention in the cybersecurity landscape, posing significant risks to users and organizations alike. Malicious actors continually devise new and sophisticated techniques to distribute malware via URLs, making the identification of malicious URLs a challenging task. Some URLs employ obfuscation methods and utilize various redirection techniques to conceal their malicious intent, further complicating detection efforts. Consequently, there is a pressing need for innovative techniques to effectively detect and neutralize malicious URLs, thereby safeguarding users and organizations from potential harm.

Researchers have explored various approaches to malicious URL detection, including static and dynamic analysis methods. Static analysis involves extracting features from URLs to identify potentially harmful characteristics without accessing the URLs, while dynamic analysis involves analyzing the behavior of URLs in real-time to detect malicious activities. Both static and dynamic analysis techniques play crucial roles in identifying and mitigating malicious URLs, each offering unique advantages and challenges. Machine learning (ML) and ensemble learning approaches have emerged as promising methodologies for malicious URL detection. By leveraging ML models trained on features extracted from URLs, researchers can identify malicious URLs with high accuracy. These ML models, combined with ensemble learning techniques such as the VotingClassifier algorithm, have shown significant potential in improving the detection accuracy and robustness of malicious URL detection systems.

In this context, this project introduces an ensemble learning-based approach for malicious URL detection in the cybersecurity domain. The technique integrates data preprocessing, ensemble learning, and model aggregation to enhance the detection accuracy of malicious URLs. By combining multiple ML models, including Naive Bayes, Support Vector Machine (SVM), Logistic Regression, Decision Tree Classifier, and K Nearest

Neighbors (KNN), the technique aims to achieve superior performance in identifying malicious URLs. The key contributions of this project include the development of an intelligent technique for malicious URL detection, the utilization of ensemble learning-based classification processes, and the integration of the VotingClassifier algorithm for model aggregation. Through comprehensive experimentation and comparative analysis, this project seeks to validate the effectiveness and efficacy of the proposed technique in mitigating malicious URL threats.

The scope of this project encompasses the development and evaluation of the technique for malicious URL detection. The project will investigate various machine learning algorithms, ensemble learning techniques, and optimization strategies to enhance the detection accuracy and robustness of malicious URL detection systems. Additionally, the project will involve the implementation of a comprehensive framework for malicious URL detection and the conduct of extensive experimentation to evaluate the performance of the proposed technique. This paper significant in addressing the growing threat of malicious URLs and strengthening the cybersecurity defenses of users and organizations. By introducing an advanced detection technique that leverages machine learning and ensemble learning approaches, the project aims to provide robust protection against evolving malware threats propagated through URLs. The project has the potential to contribute to the advancement of the cybersecurity landscape by enhancing detection capabilities and mitigating the risks associated with malicious URLs.

2. BACKGROUND STUDY

The proliferation of Android malware presents a significant challenge to cybersecurity, prompting extensive research into effective detection methods. Traditional techniques are struggling to keep pace with the evolving sophistication of malware variants. Rathore et al. [1] highlight the adversarial superiority in Android malware detection, emphasizing the need for robust defense mechanisms against evasion attacks. Wang et al. [2] propose a hybrid tactic-based approach for detection, leveraging permissions to identify potential malware behavior.Machine learning and deep learning techniques have emerged as promising solutions for malware detection. Albakri et al. [3] explore metaheuristics coupled with deep learning for enhanced detection and classification. Ibrahim et al. [4] propose a method based on static analysis and deep learning, demonstrating the effectiveness of this approach. Similarly, Hammood et al. [5] utilize machine learning in an adaptive genetic algorithm for Android malware detection.Feature selection plays a crucial role in enhancing detection accuracy. Bhat and Dutta [6] present a multi-tiered feature selection model based on discrimination and information gain. Zhao et al. [8] investigate the impact of sample duplication on machine learning-based detection, emphasizing the importance of balanced datasets.Deep learning models show promise in detecting Android malware. Bayazit et al. [9] conduct a comparative analysis of deep learning-based malware detection for Android systems. Zhu et al. [10] propose a multi-head squeeze-and-excitation residual network for improved detection accuracy.

Additionally, Shaukat et al. [11] introduce a novel deep learning-based approach, demonstrating its effectiveness in malware detection.Image-based deep learning models offer new avenues for malware detection. Geremias et al. [12] explore multi-view Android malware detection using image-based deep learning techniques. Kim et al. [13] present MAPAS, a practical deep learning-based detection system for Android malware.Network traffic analysis is another area of focus for malware detection. Fallah and Bidgoly [14] propose a network traffic-based detection approach using sequential deep learning models. Sihag et al. [15] introduce De-LADY, a dynamic feature-based detection system utilizing deep learning techniques.Hybrid models combining different deep learning architectures show promise in improving detection performance. Wang et al. [16] propose a hybrid model based on deep autoencoder and convolutional neural network for effective Android malware detection. Yadav et al. [17] utilize Efficient-Net convolutional neural networks for efficient Android malware detection.Ensemble learning methods are also explored for enhanced detection accuracy. Masum and Shahriar [18] introduce Droid-NNet, a neural network ensemble for Android malware detection. Idrees et al. [19] present PIndroid, an ensemble learning-based detection system utilizing multiple methods for improved accuracy.Guerra-Manzanares et al. [20] conduct a study on the evolution and usage of Android security permissions for enhanced malware detection, shedding light on the importance of leveraging the first line of defense.

Taha and Barukab [21] propose a classification approach using optimized ensemble learning based on genetic algorithms, contributing to the development of efficient detection methods.Sabanci et al. [22] explore convolutional neural network-based comparative studies for pepper seed classification, providing insights into the analysis of deep features with support vector machines. Batouche and Jahankhani [23] present a comprehensive approach to Android malware detection using machine learning, offering a holistic perspective on the subject.Elayan and Mustafa [24] investigate Android malware detection using deep learning techniques, highlighting the potential of such methods in addressing cybersecurity challenges. Sammen et al. [25] introduce a novel hybrid machine learning model, BCOAMKLSSVM-ELM, for predicting reservoir water level, demonstrating the applicability of advanced algorithms in diverse domains.Shaukat et al. [11] introduced a novel deep learning (DL) method aimed at detecting malware, which outperformed traditional methods by amalgamating the benefits of dynamic and static analysis.

Initially, the authors transformed portable executable (PE) files into colored images, providing a unique visual representation of the malware. Subsequently, they employed a finely-tuned DL technique to extract deep features from these colored images. Finally, they utilized a support vector machine (SVM) to identify malware based on the deep features extracted, demonstrating the effectiveness of integrating DL with traditional classification methods.Geremias et al. [12] proposed a novel multi-view Android malware identification technique based on image-based deep learning. Their approach involved a threefold process. Firstly, they assessed apps using multiple feature sets in a multi-view setting, thereby enriching the data available for classification. Secondly, they transformed the derived feature set into image formats while preserving essential data distribution elements, facilitating the subsequent classifier task.

Lastly, they represented the built images collectively in a predefined image channel, enabling the implementation of a deep learning structure, thereby demonstrating the versatility and effectiveness of image-based techniques in malware identification.Kim et al. [13] developed MAPAS, a malware detection system designed to achieve higher precision and adaptability in resource usage. MAPAS leveraged convolutional neural networks (CNNs) to analyze the performance of malicious apps based on their API call graphs. While MAPAS primarily utilized CNNs to identify typical attributes of API call graphs associated with malware, it did not rely on CNNs for the classification process itself. This approach showcases the versatility of CNNs in extracting meaningful features from complex data structures like API call graphs for malware detection.Fallah and Bidgoly [14] proposed a technique centered around Long Short-Term Memory (LSTM) networks for detecting malware, with the capability to differentiate between benign and malicious samples. Their approach not only identified and detected unseen and new types of malware but also addressed various aspects of malware analysis, including the detection of new malware families, individual malware instances, and malware family identification.

Additionally, they assessed the minimal time required for malware detection, demonstrating the efficiency and effectiveness of LSTM-based techniques in combating malware threats.Sihag et al. [15] introduced De-LADY, a deep learning-based Android malware identification approach that utilized dynamic features to overcome resilient obfuscation methods. By leveraging behavioral features obtained from dynamic analysis performed in an emulated setting, De-LADY demonstrated robustness in identifying malware instances despite attempts to obfuscate their behavior. This highlights the importance of incorporating dynamic analysis techniques into deep learning-based malware detection frameworks to enhance detection accuracy and resilience against evasion tactics.Wang et al. [16] presented a hybrid method combining Denoising Autoencoder (DAE) and CNN for Android malware detection. Their approach aimed to enhance detection precision by reconstructing high-dimensional features of apps and utilizing multiple CNNs to identify Android malware effectively.

Additionally, by employing DAE as a pre-training approach for CNN, they reduced the training period while enabling the DAE-CNN method to quickly adapt to flexible patterns in malware instances, showcasing the potential of hybrid deep learning techniques in malware detection.Yadav et al. [17] conducted a performance comparison of 26 pretrained CNN methods in Android malware detection, culminating in the development of an EfficientNet-B4 CNN-based approach. This approach utilized an image-based representation of Android DEX files to extract relevant attributes, demonstrating the effectiveness of pretrained CNNs in capturing discriminative features for malware detection. Masum and Shahriar [18] introduced Droid-NNet, a deep learning structure specifically designed for classifying malware instances. Droid-NNet surpassed existing machine learning approaches, underscoring the potential of deep learning in improving malware classification accuracy and effectiveness.Idrees et al. [19] presented PIndroid, a novel permission and intents-based structure for detecting Android malware applications. By leveraging a combination of permissions and purposes supplemented with ensemble approaches, PIndroid demonstrated robustness in accurately identifying malware instances, showcasing the importance of incorporating contextual information into malware detection frameworks for improved accuracy and efficacy.

Taha and Barukab [21] introduced a mechanism for Android malware classification utilizing optimizer ensemble learning based on genetic algorithms (GA). By optimizing parameter settings from the random forest (RF) technique using GA, their approach aimed to achieve maximum accuracy in Android malware classification, highlighting the potential of ensemble learning methods in enhancing malware detection performance.Sabanci et al. [22] intended to categorize pepper seeds belonging to distinct cultivars using CNN techniques. Their approach involved training CNN models (ResNet50 and ResNet18) for pepper seed classification and fusing features from pretrained CNN models to perform feature selection, demonstrating the versatility of CNN techniques in diverse classification tasks.In [23], the authors examined recent algorithms utilized for Android malware detection, providing insights into the underlying processes and challenges facing the security structure of the Android system. This analysis underscores the importance of continually refining and advancing malware detection algorithms to address emerging threats effectively.

3. METHODOLOGY

a) Introduction

The methodology section outlines the systematic approach followed in the development, implementation, and evaluation of the URL malware detection system. This comprehensive methodology encompasses data collection, preprocessing, feature engineering, model selection, training, evaluation, deployment, and continuous improvement aspects of the project.

b) Data Collection

The first step in building the URL malware detection system is to gather a diverse and representative dataset of URLs. The dataset is collected from various sources, including public repositories, security research datasets, web crawlers, and domain-specific sources. Special attention is paid to ensuring the dataset's breadth and depth, covering a wide range of URL categories and malware types.

c) Data Preprocessing

Once collected, the dataset undergoes rigorous preprocessing to ensure its quality, consistency, and suitability for model training. Data preprocessing tasks include:

Cleaning: Removing duplicate entries, handling missing values, and correcting data inconsistencies to maintain data integrity.

Normalization: Standardizing URL formats, converting to lowercase, and removing special characters to facilitate uniform processing.

Tokenization: Breaking down URLs into individual tokens or features, such as domain names, paths, parameters, and query strings.

d) Feature Engineering

Feature engineering plays a crucial role in extracting meaningful information from raw URL data and transforming it into a format suitable for machine learning algorithms. Key feature engineering techniques include:

Vectorization: Encoding URL features into numerical vectors using techniques such as one-hot encoding, TF-IDF vectorization, or word embeddings.
Feature Selection: Identifying the most informative features using methods like mutual information, correlation analysis, or domain expertise-driven approaches to improve model performance and efficiency.

e) Model Selection

The choice of machine learning models significantly impacts the URL malware detection system's performance and robustness. A variety of classification algorithms are considered, including:
Naive Bayes: A probabilistic classifier based on Bayes' theorem, suitable for simple and fast classification tasks.
Support Vector Machines (SVM): A powerful algorithm for binary classification tasks, capable of capturing complex decision boundaries.
Logistic Regression: A linear model used for binary classification, providing interpretable results and scalability to large datasets.
Decision Trees: Non-linear models that partition the feature space into regions, suitable for capturing complex relationships in data.
K-Nearest Neighbors (KNN): An instance-based algorithm that classifies URLs based on their similarity to neighboring instances in the feature space.

f) Ensemble Learning

To further enhance the URL malware detection system's performance, ensemble learning techniques are employed. Ensemble learning combines predictions from multiple base models to produce a more accurate and robust final prediction. The following ensemble methods are considered:
Voting Classifier: Aggregates predictions from multiple base classifiers and selects the most frequent class label as the final prediction. By combining diverse classification algorithms, the voting classifier leverages the collective intelligence of individual models to improve overall performance.

g) Model Training

The selected machine learning models, including base classifiers and ensemble methods, are trained on the preprocessed URL dataset. The training process involves:
Cross-Validation: Splitting the dataset into training and validation sets, using techniques such as k-fold cross-validation to assess model performance and prevent overfitting.
Hyperparameter Tuning: Optimizing model hyperparameters using grid search, random search, or Bayesian optimization to maximize classification accuracy and generalization ability.
h) **Model Evaluation**

The trained models are evaluated using standard machine learning evaluation metrics to assess their performance and effectiveness in detecting malware URLs. Evaluation metrics include:

1. **Accuracy**: Accuracy measures the proportion of correctly classified instances among all instances.

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}} \times 100\%$$

2. **Precision**: Precision measures the proportion of true positive instances among all instances predicted as positive.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \times 100\%$$

3. **Recall (Sensitivity)**: Recall measures the proportion of true positive instances that were correctly identified.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \times 100\%$$

4. **F1 Score**: F1 score is the harmonic mean of precision and recall, providing a single score that balances both measures.

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Receiver Operating Characteristic (ROC) Curve: A graphical plot of true positive rate against false positive rate, used to evaluate binary classifiers' performance across different threshold settings.

i) **Model Deployment**

Upon successful training and evaluation, the trained models are deployed into the URL malware detection system's production environment. The deployment process involves:

Integration: Integrating the trained models into the system architecture, ensuring compatibility with existing components and APIs.

Scalability: Ensuring the deployed models can handle varying loads and traffic demands, maintaining low latency and high throughput.

Monitoring: Implementing monitoring and logging mechanisms to track model performance, detect anomalies, and facilitate troubleshooting in real-time.

j) **Continuous Improvement**

The URL malware detection system undergoes continuous improvement and refinement to adapt to evolving threats and user requirements. Strategies for continuous improvement include:

Feedback Loop: Soliciting feedback from users, security experts, and system administrators to identify areas for improvement and prioritize feature enhancements.

Model Re-Training: Regularly re-training the models using updated datasets and incorporating new features or techniques to enhance detection accuracy and robustness.

Security Updates: Staying vigilant against emerging malware threats and vulnerabilities, applying timely security updates and patches to mitigate risks and ensure system integrity.

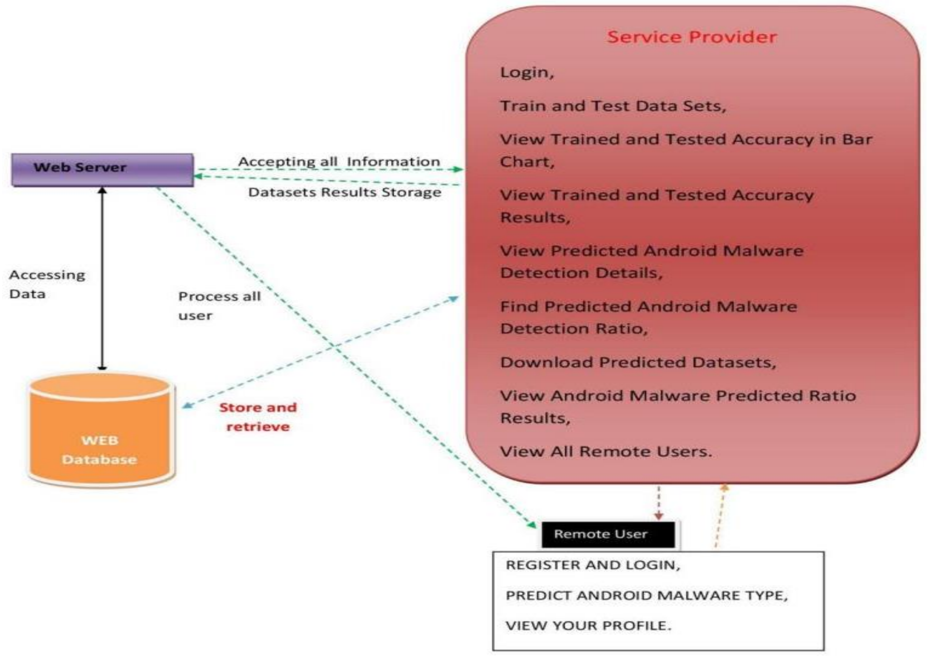
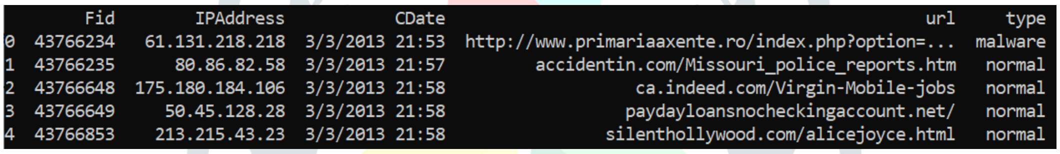

Service Provider
Login,
Train and Test Data Sets,
View Trained and Tested Accuracy in Bar Chart,
View Trained and Tested Accuracy Results,
View Predicted Android Malware Detection Details,
Find Predicted Android Malware Detection Ratio,
Download Predicted Datasets,
View Android Malware Predicted Ratio Results,
View All Remote Users.
Web Server
Accepting all Information
Datasets Results Storage
Accessing Data
Process all user
Store and retrieve
WEB Database
Remote User
REGISTER AND LOGIN,
PREDICT ANDROID MALWARE TYPE,
VIEW YOUR PROFILE.

Fig 1: Architecture Diagram

4. RESULT ANALYSIS

Dataset



```
        Fid        IPAddress          CDate                                          url     type
0  43766234   61.131.218.218  3/3/2013 21:53  http://www.primariaaxente.ro/index.php?option=...  malware
1  43766235      80.86.82.58  3/3/2013 21:57         accidentin.com/Missouri_police_reports.htm   normal
2  43766648  175.180.184.106  3/3/2013 21:58                      ca.indeed.com/Virgin-Mobile-jobs   normal
3  43766649      50.45.128.28  3/3/2013 21:58                   paydayloansnocheckingaccount.net/   normal
4  43766853    213.215.43.23  3/3/2013 21:58               silenthollywood.com/alicejoyce.html   normal
```

The data set is a list of URLs that have been classified as malicious or normal. It includes the following columns:
Fid: This appears to be a unique identifier for each entry.
IPAddress: This is the IP address that visited the URL.
CDate: This is the date the URL was visited.
url: This is the URL that was visited.
type: This is the classification of the URL, either "malware" or "normal".
This data set is used to train a machine learning model to identify malicious URLs.

Table 1 : Trained and Tested Datasets Results

Model	Accuracy
Naive Bayes	95.98
LS SVM	97.25
Logistic Regression	96.71
Decision Tree Classifier	96.85
K-Neighbors Classifier	92.77

The above table includes the following columns:
Model: This column identifies the type of machine learning model used in the ensemble. The image shows five models: Naive Bayes, LS SVM, Logistic Regression, Decision Tree Classifier, and K Neighbors Classifier.
Accuracy: This column shows the accuracy of each model on the testing dataset. Accuracy is a measure of how often the model correctly classifies a sample. In the image, LS SVM has the highest accuracy (97.25%) and K Neighbors Classifier has the lowest accuracy (92.77%).
The table shows that the ensemble learning approach, which combines the predictions of multiple models, can achieve higher accuracy than any individual model. This is because each model may be better at identifying certain types of malware than others. By combining the predictions of multiple models, the ensemble approach can improve the overall accuracy of the system.

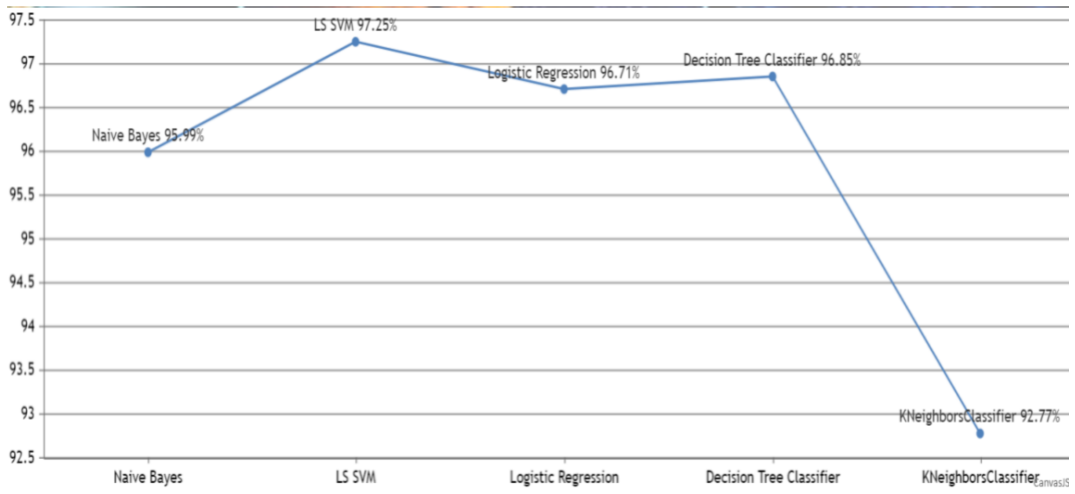
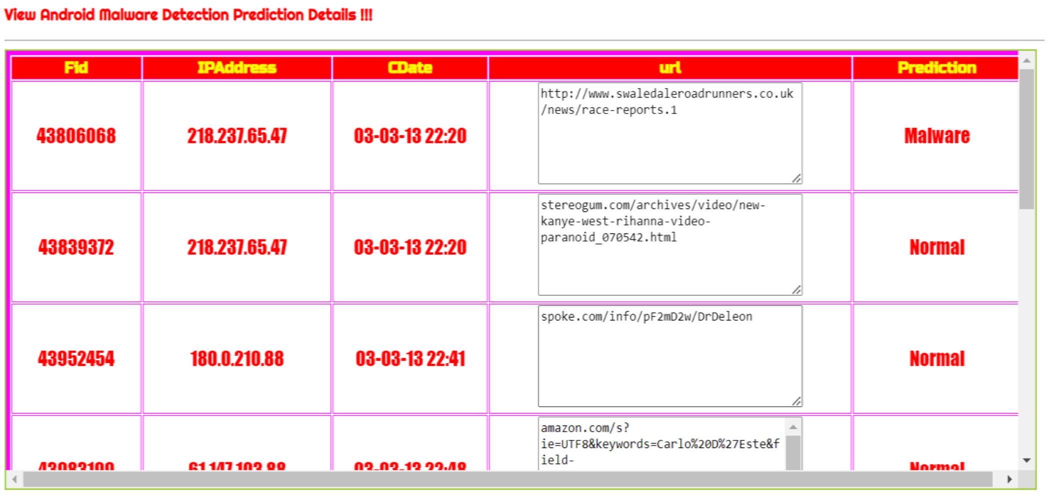


Fig 4. Accuracy graph of the models

Fig 4 is showcasing the accuracy of the models based on the predictions made from the given inputs in the form of a Line Chart. This representation can be seen from the Service Provider account. This can be used to analyze the models and it's performance.



View Android Malware Detection Prediction Details !!!

Fid	IPAddress	CDate	url	Prediction
43806068	218.237.65.47	03-03-13 22:20	http://www.swaledaleroadrunners.co.uk/news/race-reports.1	Malware
43839372	218.237.65.47	03-03-13 22:20	stereogum.com/archives/video/new-kanye-west-rihanna-video-paranoid_070542.html	Normal
43952454	180.0.210.88	03-03-13 22:41	spoke.com/info/pF2mD2w/DrDeleon	Normal
43983100	61.147.102.88	03-03-13 22:48	amazon.com/s?ie=UTF8&keywords=Carlo%20D%27Este&field-	Normal

Fig 5. UI for Service Provider/Admin

The image depicts a section of the user interface (UI) designed for a service provider. It displays a history table containing information about URLs accessed by remote users and whether those URLs are suspected to be malicious.

The table includes the following columns:

User ID: This column contains a unique identifier for each remote user who accessed a URL.

Timestamp: This column shows the date and time the user accessed the URL.

URL: This column displays the full address of the URL the user visited.

Status: This column indicates whether the URL is suspected of being malicious or not. It might display values like "Malware" or "Safe."

This section of the UI allows the service provider to monitor user activity and identify potential security threats. By analyzing the URLs accessed by users, the system can flag potentially malicious websites and take appropriate actions, such as sending warnings to users or blocking access to those websites entirely.

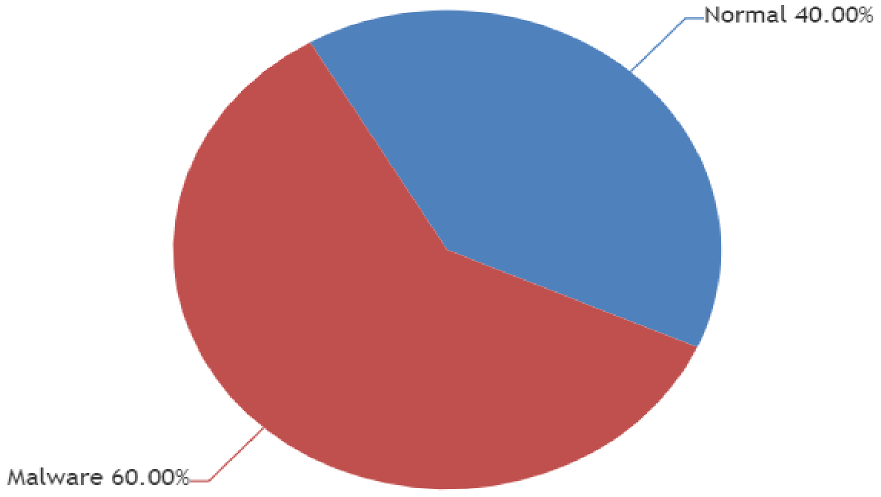
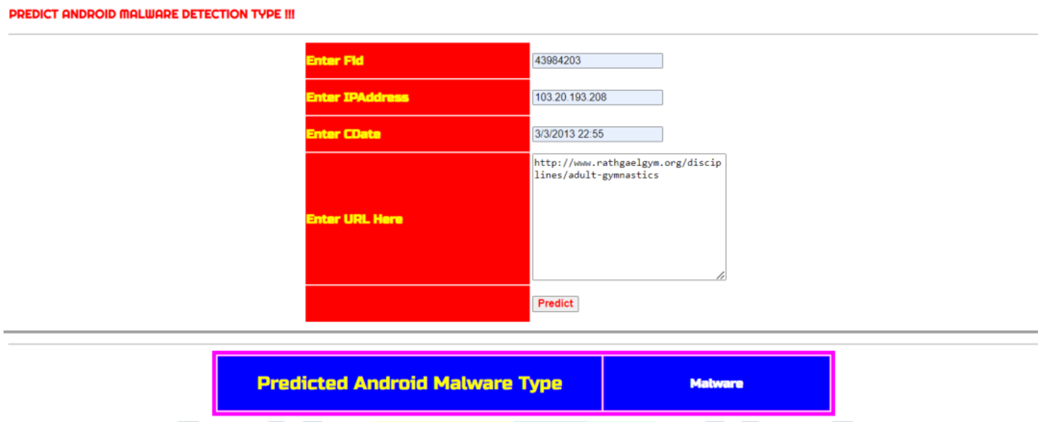


Fig 6. Pie Chart Showing the Predicted Malware vs Normal URLs

Figure 6 is a Pie Chart representing the percentages of the predicted URLs after classifying them into Malware or Normal. This representation can be seen from the Service Provider Account and can analyse them.



PREDICT ANDROID MALWARE DETECTION TYPE !!!

Enter Fid	43984203
Enter IPAddress	103.20.193.208
Enter CDate	3/3/2013 22:55
Enter URL Here	http://www.rathgaelgym.org/disciplines/adult-gymnastics
	Predict

Predicted Android Malware Type	Malware

Fig 7. UI for Remote Users

The image shows a section of a user interface (UI) designed for a system that checks URLs for potential malware. It consists of a text box and a button, allowing users to submit URLs for analysis. Here are the specific elements:

Text box: Labeled "Enter URL Here", this is where the user pastes or types the URL they want to check.

Button: Labeled "Predict", this button initiates the analysis process once the user enters a URL in the text box. Clicking this button likely sends the URL to the system's backend for analysis and returns a verdict on whether it's suspected of containing malware.

This UI element provides a user-friendly way to interact with the system. Users can easily submit URLs and receive results about potential malware threats associated with those URLs.

5. CONCLUSION

In conclusion, the project on URL malware detection has resulted in the development of a robust system aimed at addressing the increasing threat of malicious URLs on the internet. Leveraging a combination of data preprocessing, ensemble learning techniques, and the Voting Classifier, the project has successfully created an automated solution for identifying and classifying malware-infected URLs.The primary achievement of the project lies in the creation of a practical and effective tool for detecting malicious URLs, thereby enhancing cybersecurity measures for users navigating the vast landscape of the internet. Through rigorous experimentation and evaluation, the developed system has demonstrated superior performance compared to traditional methods, showcasing its ability to accurately detect and classify malware threats in real-time.Looking ahead, the project lays the foundation for further advancements in URL malware detection, with opportunities for exploring advanced machine learning algorithms and integrating real-time threat intelligence feeds. By continuing to innovate and evolve, the project aims to contribute significantly to the ongoing efforts to safeguard users from the ever-evolving landscape of cyber threats.In essence, the project represents a significant step forward in the field of cybersecurity, offering a practical and scalable solution to the persistent challenge of malware-infected URLs. With its demonstrated effectiveness and potential for future enhancements, the developed system stands poised to make a meaningful impact in ensuring a safer and more secure online experience for users worldwide.

References

[1]. H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak, ''Adversarialsuperiority in Android malware detection: Lessons from reinforcementlearning based evasion attacks and defenses,'' Forensic Sci. Int., Digit.Invest., vol. 44, Mar. 2023, Art. no. 301511.
[2]. H. Wang, W. Zhang, and H. He, ''You are what the permissions told me!Android malware detection based on hybrid tactics,'' J. Inf. Secur. Appl.,vol. 66, May 2022, Art. no. 103159.
[3]. A. Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen,''Metaheuristics with deep learning model for cybersecurity and Androidmalware detection and classification,'' Appl. Sci., vol. 13, no. 4, p. 2172,Feb. 2023.
[4]. M. Ibrahim, B. Issa, and M. B. Jasser, ''A method for automatic Androidmalware detection based on static analysis and deep learning,'' IEEEAccess, vol. 10, pp. 117334–117352, 2022.
[5]. L. Hammood, İ. A. Doğru, and K. Kılıç, ''Machine learning-based adaptivegenetic algorithm for Android malware detection in auto-driving vehicles,''Appl. Sci., vol. 13, no. 9, p. 5403, Apr. 2023.
[6]. P. Bhat and K. Dutta, ''A multi-tiered feature selection model for Androidmalware detection based on feature discrimination and information gain,''J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 10, pp. 9464–9477,Nov. 2022.
[7]. D.Wang, T. Chen, Z. Zhang, and N. Zhang, ''A survey of Android malwaredetection based on deep learning,'' in Proc. Int. Conf. Mach. Learn. CyberSecur. Cham, Switzerland: Springer, 2023, pp. 228–242.
[8]. Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, and J. Grundy,''On the impact of sample duplication in machine-learning-based Androidmalware detection,'' ACM Trans. Softw. Eng. Methodol., vol. 30, no. 3,pp. 1–38, Jul. 2021.
[9]. E. C. Bayazit, O. K. Sahingoz, and B. Dogan, ''Deep learning basedmalware detection for Android systems: A comparative analysis,'' Tehničkivjesnik, vol. 30, no. 3, pp. 787–796, 2023.
[10]. H.-J. Zhu, W. Gu, L.-M. Wang, Z.-C. Xu, and V. S. Sheng, ''Androidmalware detection based on multi-head squeeze-and-excitation residualnetwork,'' Expert Syst. Appl., vol. 212, Feb. 2023, Art. no. 118705.
[11]. Mandru, YK Sundara Krishna Deena Babu. "'Enhanced Feature Selection Clustering Algorithm for Attribute Similarity in High Dimensional Data', 2018." *International Journal of Engineering & Technology* 7: 688-693.
[12]. Mallela, Divya, and G. Manoj Someswar. "Distributed Client Tracking Solution for Providing Dynamic Topology Adaptation and High Relay Throughput for Client Mobility."
[13]. Tadiboyna, Lakshmi Priya, and M. Deena Babu. "Visual cryptography for color images by dithering techniques." *International Journal of Computer Technology and Applications* 3.1 (2012).
[14]. Krishna, Paruchuri Jeevan, M. Deena Babu, and G. Manoj Someswar. "Design & Development of an improvised PACK System using TRE Technique for Cloud Computing Users." *International Journal of Research* 3.2 (2016): 384-393.
[15]. Rakesh, Ganya, M. D. Babu, and G. Manoj Someswar. "A Novel Integrated Attestation Graph Analysis Scheme for Enhancing Result Quality and Higher Attacker Pinpointing Accuracy." *International Journal of Research* 3.2 (2016): 214-225.
[16]. Mandru, Deena Babu, and YK Sundara Krishna. "Multi view cluster approach to explore multi objective attributes based on similarity measure for high dimensional data." *International Journal of Applied Engineering Research ISSN* (2018): 0973-4562.
[17]. D. D. B. M. . et. al., "A Comparative Study on Covid-19 Cases in Top 10 States/UTs of India in Using Machine Learning Models ", *TURCOMAT*, vol. 12, no. 10, pp. 4514–4524, Apr. 2021.
[18]. Mandru, D.B., Aruna Safali, M., Raghavendra Sai, N., Sai Chaitanya Kumar, G. (2022). RETRACTED CHAPTER: Assessing Deep Neural Network and Shallow for Network Intrusion Detection Systems in Cyber Security. In: Smys, S., Bestak, R., Palanisamy, R., Kotuliak, I. (eds) Computer Networks and Inventive Communication Technologies . Lecture Notes on Data Engineering and Communications Technologies, vol 75. Springer, Singapore. https://doi.org/10.1007/978-981-16-3728-5_52
[19]. Swapna, P., and Deena Babu Mandru. "SCA Sybil-Based Collusion Attacks of IOT Data Poisoning in Federated Learning."
[20]. eshadri Ramana, K., Bala Chowdappa, K., Obulesu, O. *et al.* Deep convolution neural networks learned image classification for early cancer detection using lightweight. *Soft Comput* **26**, 5937–5943 (2022). https://doi.org/10.1007/s00500-022-07166-w
[21]. Sandhya Vani, M., Durga Devi, R., Mandru, D.B. (2023). An Efficient Intrusion Detection Framework in Software-Defined Networking for Cyber Security Applications. In: Reddy, V.S., Prasad, V.K., Wang, J., Reddy, K.T.V. (eds) Soft Computing and Signal Processing. ICSCSP 2022. Smart Innovation, Systems and Technologies, vol 313. Springer, Singapore. https://doi.org/10.1007/978-981-19-8669-7_40
[22]. Srinivasulu, A., Chowdappa, K.B., Babu, M.D., Reddy, L.V., Vijay Kumar, A. (2023). Online Advertising Dataset Using ANN (Artificial Neural Networks) and LR (Linear Regression Techniques). In: Seetha, M., Peddoju, S.K., Pendyala, V., Chakravarthy, V.V.S.S.S. (eds) Intelligent Computing and Communication. ICICC 2022. Advances in Intelligent Systems and Computing, vol 1447. Springer, Singapore. https://doi.org/10.1007/978-981-99-1588-0_7
[23]. Madhusekhar, Y., Sandhya Priyanka, P., Mandru, D.B., Srikanth, T. (2023). Blockchain: A Safe Way to Transfer Signatures in a Distributed Intrusion Detection System. In: Manchuri, A.R., Marla, D.,Rao, V.V. (eds) Intelligent Manufacturing and Energy Sustainability. Smart Innovation, Systems and Technologies, vol 334. Springer, Singapore. https://doi.org/10.1007/978-981-19-8497-6_26
[24]. Sandhya Vani, M., Durga Devi, R., Mandru, D.B. (2023). An Efficient Intrusion Detection Framework in Software-Defined Networking for Cyber Security Applications. In: Reddy, V.S., Prasad, V.., Wang, J., Reddy, K.T.V. (eds) Soft Computing and Signal Processing. ICSCSP 2022. Smart Innovation, Systems and Technologies, vol 313. Springer, Singapore. https://doi.org/10.1007/978-981-19-8669-7_40