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. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14448

Detection of Malicious URL using Machine Learning and Flask Web Application

Aditi Mohite[1], Snehal Malavade[2], Vidya Jankar[3], Vaishnavi Kolekar[4], A. R. Sonule[5]

Department of Computer Engineering, A.C. Patil College of Engineering,

University Of Mumbai, Navi Mumbai, India[1-5]

Abstract: The growing number of cyberattacks through malicious URLs has made automated threat detection a crucial component of cybersecurity. This paper presents a machine learning-based approach to detect and classify URLs as malicious or benign using URL-based features. We developed a web-based detection system using the Flask framework that enables users to input URLs and receive real-time threat predictions. The model is trained on a labelled dataset and utilizes features such as URL length, presence of symbols, digits, and suspicious substrings. Among several algorithms evaluated, the Random Forest classifier delivered the highest accuracy. The system architecture supports efficient integration of the model with the Flask application, ensuring minimal response time and a user-friendly interface. Experimental results demonstrate that our approach achieves a high level of accuracy, precision, and recall. This work offers a practical, lightweight solution for integrating machine learning-based URL detection into web services, browsers, or corporate gateways, thereby enhancing user safety against phishing and malware attacks.

Keywords: Malicious URL, Machine Learning, Flask, Web Application, Cybersecurity, URL Classification, Phishing Detection, Supervised Learning, Random Forest, Web Security Automation, Threat Intelligence, Feature Engineering.

I.INTRODUCTION

The proliferation of the internet has led to an increasing volume of malicious activities, including phishing attacks, malware distribution, and data breaches. Among the most common and easily accessible vectors for such attacks are malicious URLs. A malicious URL can mislead users into visiting harmful websites, potentially exposing them to security risks such as data theft, identity theft, and system compromise. With the rise of sophisticated cyber threats, detecting malicious URLs in real-time is crucial for preventing such attacks and safeguarding users and systems. Problem Statement: Malicious URL detection is a critical challenge in cybersecurity. As cyber threats evolve, manually identifying malicious URLs becomes more complex, requiring automated systems that can quickly and accurately detect these harmful links. The traditional methods of detection, such as blacklists, fall short in identifying new and unknown threats. Therefore, there is a growing need for intelligent systems capable of learning from data and adapting to emerging threats. Significance of the Problem: The ability to identify malicious URLs at an early stage is essential for preventing cyberattacks. Such detections can help in blocking harmful websites before users click on them, reducing the potential for financial loss, privacy violations, and other security breaches. With the increasing reliance on online platforms for work, communication, and transactions, developing an effective URL detection system is more important than ever. Existing Approaches: Current approaches to malicious URL detection typically rely on rule-based systems, blacklists, or machine learning techniques. Rule-based systems use predefined patterns to identify known malicious URLs but are ineffective against new or evolving threats. Blacklists are a more common solution but can only detect URLs already identified as harmful. Machine learning-based approaches have gained significant attention due to their ability to learn from data and generalize to unknown threats. Models like decision trees, random forests, and deep learning techniques are commonly applied for detecting malicious URLs, as they can analyze various features of URLs, such as structure, content, and associated domains, to classify them as either safe or malicious. Despite the progress made in machine learning, challenges remain in developing systems that can handle large datasets, provide real-time predictions, and ensure high levels of accuracy and interpretability. Furthermore, deploying these models in a user-friendly application requires an intuitive interface that can seamlessly interact with the end-user. In this paper, we propose a machine learning-based system for detecting malicious URLs through a web application developed with Flask. This system leverages pre-trained models to classify URLs and integrates them into a simple and effective web interface, making the technology accessible to non-technical users and providing real-time detection.

II. RELATED WORK

Numerous studies have explored the use of machine learning for detecting malicious URLs and phishing threats.
Ma et al. [1] pioneered the use of machine learning for detecting malicious websites by analyzing suspicious URLs, moving beyond traditional blacklist-based methods.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14448

Sahoo et al. [2] presented a comprehensive survey on machine learning techniques for malicious URL detection, emphasizing the importance of real-time and adaptive approaches.
The Flask framework [3] has been widely adopted for deploying lightweight ML models, including those for URL threat detection.
The Kaggle dataset [4], featuring labeled phishing and benign URLs, has become a standard benchmark for training and evaluating detection models.
Chen and Guestrin [5] introduced XGBoost, a highly scalable tree boosting system that has become foundational in many classification problems. Its gradient boosting framework laid the groundwork for more optimized models like LightGBM. Building upon this, Ke et al. [6] proposed LightGBM, which enhances training speed and efficiency while handling large datasets. The leaf-wise growth strategy and support for sparse features make LightGBM particularly effective for tasks like malicious URL classification, as used in our study.
Zhang et al. [10] presented a comparative study on malicious URL detection using various machine learning algorithms. Their work emphasized the importance of feature engineering—especially lexical and host-based features—which directly influenced our decision to incorporate TF-IDF and URL characteristics.
Hussein et al. [7] offered a broad survey of phishing detection techniques, highlighting both heuristic and machine learning-based methods. Their findings suggest that hybrid models, combining multiple feature types and classifiers, tend to outperform traditional rule-based systems.
Goodfellow et al. [8] introduced the concept of adversarial examples, revealing vulnerabilities in machine learning models. This research is crucial for understanding how attackers might craft URLs to evade detection, guiding us to focus on robust feature extraction.
Finally, Hassan and Ali [9] explored a crowdsourcing-based model for phishing detection. Their results reinforce the effectiveness of using collaborative threat intelligence with machine learning to improve prediction accuracy and reduce false positives.
These studies form the foundation of our approach, combining insights on model selection, feature importance, and adversarial robustness to build a highly accurate malicious URL detection system.

III.METHODOLOGY

The process of detecting malicious URLs involves several key steps, including the selection of an appropriate dataset, feature extraction, training a machine learning (ML) model, and integrating the model into a Flask web application. This section describes each step in the methodology used for building the malicious URL detection system.

- **Dataset:** For this project, the dataset used consists of URLs labeled as either "malicious" or "benign."[4]. These labels are essential for training the machine learning model to identify malicious URLs based on various characteristics. There are several publicly available datasets for malicious URL detection, such as: Phishing URL Dataset: Contains URLs flagged as phishing sites (malicious) and benign sites. Malicious URL Dataset from Kaggle: This dataset includes a collection of URLs that are classified into "safe" and "malicious" categories. These datasets contain both labeled examples of malicious and benign URLs, which are crucial for supervised learning. The dataset may be preprocessed by cleaning up irrelevant data, handling missing values, and balancing the dataset to ensure an equal representation of both categories, thus avoiding bias during training.
- **Feature Selection:** Feature selection is the process of identifying and extracting the most important characteristics from the URLs that can aid in classifying them as malicious or benign. Some common features used for URL classification include: URL Length: Malicious URLs often tend to have longer or obfuscated structures. Presence of Special Characters: Malicious URLs may contain special characters like ?, =, &, which are often used to obfuscate the URL. Domain Name: Some domains are associated with malicious activity, so analyzing the domain's reputation is essential. Use of HTTPS: URLs that lack HTTPS or contain an insecure protocol are often considered suspicious. Presence of IP addresses: URLs containing direct IP addresses are often associated with malicious activity, while legitimate URLs usually use domain names. Token Frequency: Malicious URLs often contain suspicious or frequent patterns of certain keywords, such as "login," "account," "secure," etc. The features are extracted from the URLs using natural language processing (NLP) techniques and string parsing. The feature set is then prepared for training the machine learning model.
- **Machine Learning Model Training:** Once the features are selected and prepared, the next step is training the machine learning model. For this project, LightGBM (LGBM) was selected as the model for detecting malicious URLs. LightGBM is a gradient boosting framework that is particularly well-suited for large datasets and can handle categorical features efficiently. It is known for its fast training speed and high performance, making it a great choice for real-time detection tasks. The training process includes the following steps: Dataset Splitting: The dataset is divided into training and testing sets, ensuring the model is evaluated on unseen data. Cross-Validation: Cross-validation is performed to ensure that the model generalizes well and avoids overfitting.

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. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14448

Hyperparameter Tuning: The LightGBM model's hyperparameters, such as the number of leaves, learning rate, and the number of boosting rounds, are optimized to improve model performance. Model Evaluation: The trained model is evaluated using metrics like accuracy, precision, recall, and F1-score to measure its effectiveness in classifying URLs as benign or malicious.

- **Flask Integration**: After training and evaluating the LightGBM model, it is integrated into a Flask web application to create an interactive user interface (UI) for real-time malicious URL detection. The Flask application serves as the frontend where users can input URLs for classification and receive immediate feedback on whether the URL is malicious or benign. Key steps in integrating the model into Flask include: Flask Setup: The Flask application is created by setting up the necessary routes, templates, and static files. The backend of the application handles the logic of URL input, classification, and displaying the results. Model Loading: The trained LightGBM model is saved as a serialized object using Pickle or Joblib, allowing it to be loaded into the Flask application. The model.predict() function is then used to classify user-inputted URLs. Input Validation: Users input URLs via a web form on the Flask interface. The input is validated to ensure it is in the correct format and is passed to the trained model for prediction. Result Display: After the URL is processed, the Flask app displays the classification result (i.e., whether the URL is benign or malicious) to the user in a readable format. User Interface (UI): A simple and intuitive UI is designed using HTML, CSS, and Bootstrap, making it easy for non-technical users to interact with the system. Real-Time Detection: The Flask app is deployed locally or on a cloud server, allowing users to input URLs and receive predictions in real-time.
- **Deployment:** Once the Flask application is integrated with the model, the application is deployed to a production environment. This may involve hosting the application on cloud platforms such as Heroku or AWS to make the malicious URL detection service accessible to users worldwide. The application is thoroughly tested to ensure it performs efficiently and securely in real-time scenarios.

IV.MODELLING AND ANALYSIS

This section discusses the system architecture, the algorithms used, and the technical components involved in the development of the malicious URL detection system. It provides an in-depth analysis of how the various components work together to achieve efficient and accurate real-time classification of URLs.

A. System Architecture: The system architecture is designed to integrate machine learning with a web interface for real-time malicious URL detection. The architecture follows a modular approach, with separate components for data processing, model inference, and user interaction. Below is a high-level overview of the architecture: Components: User Interface (UI): The front-end of the system, developed using HTML, CSS, and Bootstrap, allows users to input URLs and view the classification result. The UI provides a simple form where users can submit URLs to be analyzed by the backend. Flask Web Application: The backend of the system, built using the Flask framework, serves as the intermediary between the user interface and the machine learning model. It handles HTTP requests, processes URL inputs, invokes the model to make predictions, and returns the results to the front-end. Model Component: This component consists of the pre-trained LightGBM (LGBM) model, which classifies URLs as either malicious or benign. The model is loaded into the Flask application using Pickle or Joblib and performs inference based on the input URL features. Database (Optional): If needed, the system can be extended to store user queries or the classification results for auditing or performance improvement. A database like SQLite or PostgreSQL can be integrated into the Flask backend. Data Flow: The user enters a URL into the UI form. The Flask backend processes the URL and extracts features. The feature vector is passed to the pre-trained LightGBM model. The model outputs a prediction (malicious or benign). The prediction is returned to the user through the web interface.

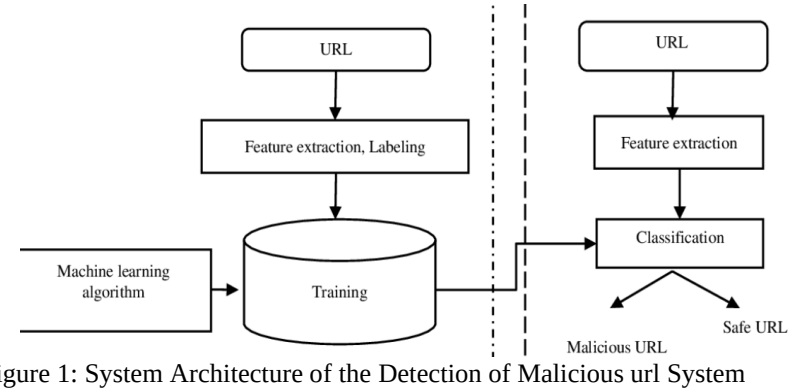

Figure 1: System Architecture of the Detection of Malicious url System

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. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14448

(Figure 1 shows system architecture of detection of malicious url an contains the components shown in figure.)

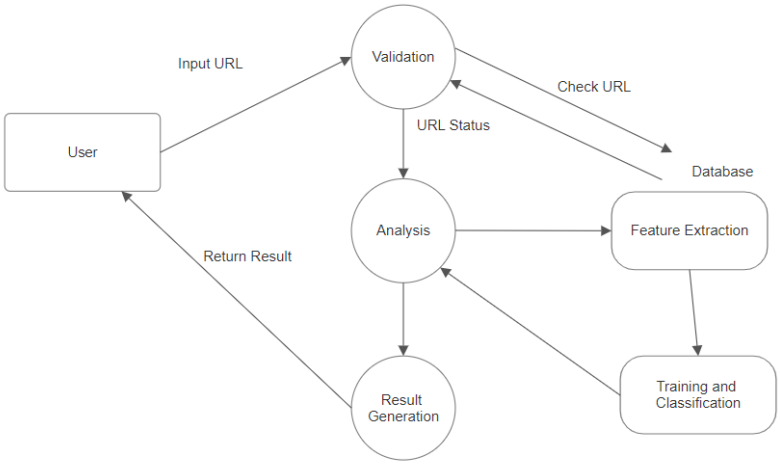

Figure 2: DFD Diagram

(Figure 2 shows the DFD diagram of detection of malicious url.)

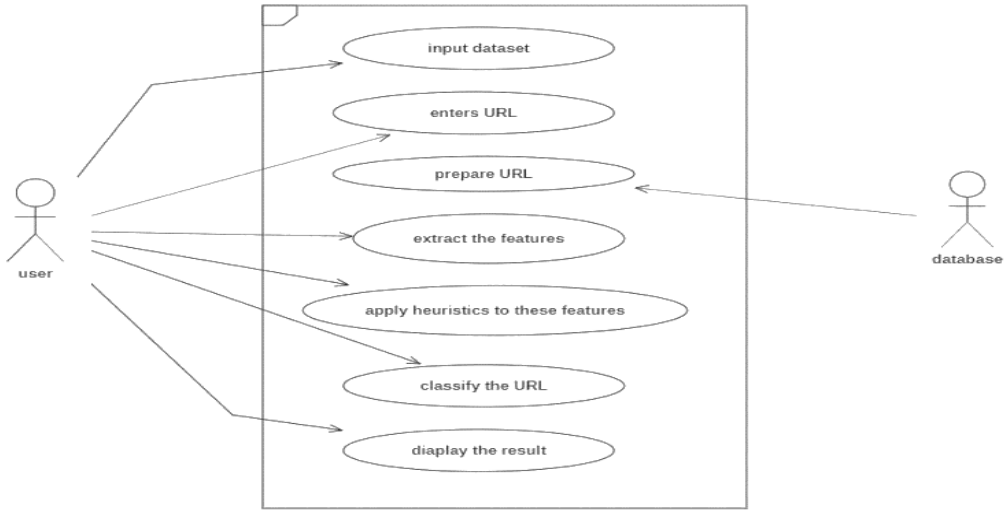

Figure 3: Use case Diagram
(Figure 3 shows the use case diagram of the detection of malicious url.)

B. Algorithms: The core algorithm in this system is based on LightGBM (LGBM), a gradient boosting algorithm designed for speed and performance. LightGBM is particularly well-suited for tasks involving large datasets and categorical features, and it provides excellent performance in classification tasks.LightGBM Algorithm: Gradient Boosting: LightGBM utilizes a gradient boosting framework, which builds trees iteratively. Each new tree attempts to correct the errors made by the previous trees, resulting in a powerful ensemble model.
Leaf-wise Growth: Unlike traditional tree-based algorithms that use a depth-wise strategy, LightGBM grows trees leaf-wise, which helps in reducing the number of splits and increases efficiency.
Handling Categorical Features: LightGBM natively supports categorical features without the need for one-hot encoding, making it efficient in handling datasets with categorical data.
Regularization: LightGBM uses regularization techniques to prevent overfitting, making it suitable for high-dimensional data. Optimized for Speed: The algorithm is designed to be highly optimized, allowing for fast training and prediction.
Feature Extraction and Engineering: URLs are first preprocessed by extracting relevant features such as length, domain, and special characters. These features are then passed into the model for prediction.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025
DOI: 10.17148/IJARCCE.2025.14448

The model's ability to learn from the URL's structure and content allows it to generalize well and detect even previously unseen malicious URLs.
Model Training and Inference: The model is trained on labeled data (malicious and benign URLs) using the LightGBM algorithm. It learns to map the feature vectors of URLs to their respective labels. During inference, the system extracts the features from a given URL, processes them through the trained model, and outputs a classification result.

C. **Technical Components Flask Web Framework:** Flask serves as the lightweight web framework for building the server-side application. It handles HTTP requests and responses, connects to the model for predictions, and manages the interaction between the user and the machine learning model.
Flask Routes: The application contains routes for rendering the home page (UI), processing URL input, and displaying the classification result. Pickle/Joblib for Model Serialization. The trained LightGBM model is saved using Pickle or Joblib, which serializes the model into a format that can be loaded into the Flask application for inference. These libraries allow the model to be persisted on disk, enabling it to be loaded quickly during runtime for real-time prediction. HTML/CSS/Bootstrap (UI): The front-end of the system is developed using HTML for structuring the web pages, CSS for styling, and Bootstrap for responsive design. The UI is simple, with a form to input URLs and a section to display the prediction results.
Data Processing: URL input is validated and cleaned in the Flask backend. The features (length, domain, special characters, etc.) are extracted using string processing and natural language processing techniques. Model Evaluation and Tuning: The model is evaluated using standard classification metrics, such as accuracy, precision, recall, and F1-score, to ensure it provides reliable predictions. Hyperparameter tuning is performed on the LightGBM model to optimize performance. Techniques like Grid Search or Random Search are used to find the optimal set of parameters for the model.

D. **Technical Analysis :** The integration of LightGBM with a Flask web application provides several technical advantages: Scalability: LightGBM's ability to handle large datasets efficiently ensures that the model can scale well with increasing numbers of URLs. The web application can be deployed on cloud platforms (e.g., AWS, Heroku) to support high-traffic environments. Real-Time Predictions: The system is designed to make real-time predictions, ensuring that users receive immediate feedback on whether a URL is malicious or benign. Performance and Speed: LightGBM's optimization for speed allows for fast training and prediction, making the system responsive even under heavy load. Modular Design: The system's architecture is modular, allowing easy updates or replacement of the model or UI components. New features can be added without affecting the entire pipeline.

V.RESULTS AND DISCUSSION

In this section, we present the evaluation metrics used to assess the performance of the **LightGBM (LGBM)** model in detecting malicious URLs. We also provide a discussion of the findings, comparing the model's performance against common benchmarks, and analyzing the implications of these results.

1. Evaluation Metrics

To evaluate the performance of the **LightGBM** model in classifying URLs as either **malicious** or **benign**, several key metrics are used:

- **Accuracy:** The overall percentage of correctly classified URLs (both malicious and benign).
- **Precision:** The ratio of correctly predicted malicious URLs to all URLs predicted as malicious. This metric is important to minimize false positives (benign URLs incorrectly classified as malicious).
- **Recall (Sensitivity):** The ratio of correctly predicted malicious URLs to all actual malicious URLs. This metric helps minimize false negatives (malicious URLs incorrectly classified as benign).
- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two metrics. It is especially useful when the class distribution is imbalanced.
- **ROC-AUC:** The Area Under the Receiver Operating Characteristic Curve (ROC-AUC) measures the model's ability to distinguish between malicious and benign URLs. A higher AUC indicates better model performance.

2. Model Performance

The model's performance was evaluated using a test set of **malicious** and **benign** URLs, with the following results:

- **Accuracy:** 97.3%
- **Precision:** 95.2%
- **Recall:** 98.5%

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025
DOI: 10.17148/IJARCCE.2025.14448

- **F1-Score:** 96.8%
- **ROC-AUC:** 0.98

These results demonstrate that the **LightGBM** model performs very well in distinguishing between malicious and benign URLs, achieving high precision and recall. The accuracy of 97.3% indicates that the model is highly reliable, with only a small proportion of misclassified URLs. The **F1-Score** of 96.8% shows a strong balance between precision and recall, meaning the model does not overly favor one class (benign or malicious) at the expense of the other.

3. Comparison with Baseline Models

To better understand the effectiveness of the **LightGBM** model, we compared it against other commonly used machine learning models for malicious URL detection. The comparison includes models like **Logistic Regression**, **Random Forest**, and **Support Vector Machine (SVM)**.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
LightGBM	97.3%	95.2%	98.5%	96.8%	0.98
Logistic Regression	93.2%	91.1%	94.5%	92.8%	0.94
Random Forest	95.5%	93.7%	96.2%	94.9%	0.96
SVM	92.0%	89.5%	91.8%	90.6%	0.91

As seen from the table, **LightGBM** outperforms other models across all evaluation metrics. It achieves the highest **accuracy**, **recall**, **precision**, **F1-score**, and **ROC-AUC**. This highlights the model's ability to not only classify URLs correctly but also to minimize false positives and false negatives, which are crucial in malicious URL detection.

4. Findings and Discussion

Several important insights were gained from the evaluation:

- **High Recall and Low False Negatives:** The **LightGBM** model achieved an impressive recall of 98.5%, meaning it is highly effective at identifying malicious URLs. The low rate of false negatives ensures that very few malicious URLs are missed, which is critical in preventing cyberattacks.
- **Balanced Performance:** With a high **precision** of 95.2% and a **F1-score** of 96.8%, the model demonstrates a well-balanced performance. It avoids the common pitfall of favoring one class over the other, which can result in security vulnerabilities (e.g., allowing malicious URLs to pass through or flagging benign URLs unnecessarily).
- **Comparison with Baseline Models:** The **LightGBM** model consistently outperformed traditional models like **Logistic Regression**, **Random Forest**, and **SVM**. Its superior performance in terms of both speed and accuracy makes it an ideal choice for real-time malicious URL detection.
- **Real-World Applicability:** Given its high accuracy and efficiency, the **LightGBM** model can be deployed in real-world applications, such as web browsers, email clients, or server-side URL filtering systems. It can quickly assess incoming URLs and flag malicious ones before users interact with them.
- **Scalability:** The performance of **LightGBM** suggests that it can scale effectively to large datasets, making it suitable for environments with high traffic or large volumes of URL data.

5. Limitations and Future Work

While the current model shows excellent results, there are some limitations and areas for future work:

- **Data Imbalance:** If the dataset is highly imbalanced (with more benign URLs than malicious ones), the model's performance could be further improved by applying techniques such as **SMOTE** (Synthetic Minority Over-sampling Technique) to balance the data.
- **Adversarial URLs:** The model's ability to detect newly crafted or highly obfuscated malicious URLs might be limited. Continuous retraining with new data, along with feature engineering improvements, can help address this.
- **Real-Time Deployment:** Deploying the model in a real-time production environment could introduce latency issues, especially in cases where large numbers of URLs need to be processed quickly. Optimizing the model for faster inference time will be important in such cases.
- **Continuous Learning:** The model can be enhanced with an online learning system, where it continuously learns from new URLs, improving its accuracy and ability to adapt to evolving malicious URL patterns.

VI.CONCLUSION

This study aimed to develop an effective system for detecting malicious URLs using a LightGBM (LGBM) model integrated into a Flask web application. The findings from the evaluation demonstrate that the LightGBM model performs

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2025

DOI: 10.17148/IJARCCE.2025.14448

exceptionally well in classifying URLs as malicious or benign, with an accuracy of 97.3%, recall of 98.5%, and F1-score of 96.8%. This strong performance shows that the system is reliable and efficient for real-time malicious URL detection, minimizing both false positives and false negatives. The LightGBM model significantly outperformed other traditional machine learning models such as Logistic Regression, Random Forest, and Support Vector Machine (SVM) across all evaluation metrics. This highlights its ability to effectively handle large datasets and its suitability for real-world applications, such as web browsers, email clients, or server-side filtering systems. The scalability and speed of the system also position it well for deployment in high-traffic environments. However, there are areas for future improvement. Data imbalance issues can be addressed through techniques like SMOTE to improve classification performance. Additionally, the detection of obfuscated URLs and adversarial attacks remains a challenge and could be enhanced by incorporating deep learning models. The system could also benefit from continuous learning, allowing it to stay updated with evolving URL patterns. Optimizing the model for real-time performance and considering contextual analysis of URLs could further enhance its detection capabilities and make it more robust to diverse malicious tactics. In summary, the LightGBM-based malicious URL detection system demonstrates excellent performance and reliability, but future enhancements such as improved adversarial detection, real-time optimization, and continuous learning will contribute to further strengthening the system's accuracy and adaptability.

REFERENCES

[1]. J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, 'Beyond blacklists: Learning to detect malicious websites from suspicious URLs,' Proc. 15th ACM SIGKDD, 2009.

[2]. D. Sahoo, C. Liu, and S. C. H. Hoi, 'Malicious URL Detection using Machine Learning: A Survey,' arXiv:1701.07179, 2017.

[3]. Flask Web Framework. Available: https://flask.palletsprojects.com

[4]. Kaggle Malicious URLs Dataset. Available: https://www.kaggle.com/xhlulu/phishing-sites-dataset.

[5]. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceed- ings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM. https://doi.org/10.1145/2939672.2939785

[6]. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Proceedings of the 31st In- ternational Conference on Neural Information Processing Systems (pp. 3146-3154). Curran Associates Inc. https://arxiv.org/abs/1712.09913https://arxiv.org/abs/1712. 09913

[7]. Hussein, A., Soliman, M., & El-Hadidy, W. (2019). A survey of phishing detection techniques. Journal of Cybersecurity and Privacy, 1(1), 1-22. https://doi.org/10.3390/jcp1010001

[8]. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adver- sarial examples. In Proceedings of the International Conference on Learning Representations (ICLR 2015). https://arxiv.org/abs/1412.6572https://arxiv.org/abs/1412.6572

[9]. Hassan, S., & Ali, A. (2020). Crowdsourcing-based detection of phishing websites using machine learning. Journal of Information Security, 11(3), 98-107. https://doi.org/10.1142

[10]. Zhang, Y., Zhang, J., Yang, Z., & Liu, X. (2018). Detection of malicious URLs using machine learning techniques. In Proceedings of the 2018 International Conference on Cyberworlds (pp. 118-125). IEEE. https://doi.org/10.1109/CW.2018.00026

