URL Phishing Detection using Machine Learning Algorithms

Atharva Kasar

Contents

1. Abstract

Phishing attacks have emerged as one of the most prevalent and dangerous cyber threats in modern digital ecosystems, primarily exploiting deceptive and malicious URLs to impersonate trusted entities and trick users into revealing sensitive information such as login credentials, financial details, and personal data. As online services, digital payments, and cloud-based platforms continue to grow, traditional phishing detection techniques based on blacklists and static rules have become increasingly ineffective due to the short lifespan and rapidly evolving nature of phishing websites. This project proposes a machine learning–based phishing URL detection model that leverages lexical and structural characteristics of URLs to accurately classify them as phishing or legitimate. Multiple supervised learning algorithms, including Decision Tree, Support Vector Machine, Logistic Regression, K-Nearest Neighbours, Naive Bayes, and Random Forest, are implemented and evaluated on multiple datasets containing labeled phishing and legitimate URLs. The model follows a structured pipeline involving data preprocessing, feature extraction, model training, and performance evaluation using metrics such as accuracy, precision, recall, and F1-score. Experimental analysis demonstrates that ensemble-based models, particularly Random Forest, consistently outperform individual classifiers by effectively handling feature variability and reducing overfitting, achieving high and stable detection accuracy across datasets. The proposed approach highlights the practicality of URL-based machine learning techniques for phishing detection and provides a scalable foundation for deployment in web-based security applications aimed at enhancing user protection and digital trust.

2. Introduction

Phishing is a sophisticated form of cybercrime in which attackers manipulate users into disclosing sensitive information by impersonating legitimate organizations, services, or individuals. These attacks commonly rely on fraudulent URLs that closely resemble authentic web addresses, exploiting users' trust and lack of awareness. With the rapid digitalization of financial services, e-commerce platforms, cloud applications, and social media, phishing attacks have increased significantly in both frequency and complexity. Malicious actors continuously modify URL structures, use URL shortening services, and deploy large numbers of short-lived phishing websites to evade detection. Traditional phishing detection mechanisms, such as blacklist-based approaches and manually crafted heuristic rules, are increasingly inadequate because they fail to identify newly generated or previously unseen phishing URLs, commonly referred to as zero-day attacks. As a result, there is a growing need for intelligent and adaptive detection systems that can analyze URL patterns and automatically distinguish malicious links from legitimate ones in real time.

Machine learning has emerged as a powerful solution to address these challenges by enabling systems to learn discriminative patterns from historical data and generalize to unseen phishing attempts. In particular, URL-based phishing detection has gained significant attention due to its efficiency and low computational cost, as it does not require downloading webpage content or executing scripts. By extracting lexical and structural features from URLs—such as length, special character frequency, abnormal subdomain usage, and token patterns—machine learning models can effectively capture the underlying characteristics of phishing behavior. This project focuses on designing and evaluating a machine learning–based phishing URL detection system using multiple supervised learning algorithms, including Decision Tree, Support Vector Machine, Logistic Regression, K-Nearest Neighbours, Naive Bayes, and Random Forest. Through extensive experimentation on multiple datasets and rigorous performance evaluation, the study aims to identify the most reliable classification model and demonstrate the feasibility of deploying an accurate, scalable, and real-time phishing detection solution to enhance cybersecurity defenses.

3. System Design

The system design of the proposed phishing URL detection framework is structured to ensure accuracy, scalability, and real-time applicability. The overall architecture follows a modular machine learning pipeline in which each stage performs a well-defined function, allowing the system to be easily extended or improved in the future. The design emphasizes URL-based analysis, making the system lightweight and efficient while maintaining high detection accuracy. The complete workflow consists of data collection, data preprocessing, feature extraction, model training, model evaluation, final model selection, and deployment through a web-based interface.

3.1 Overall Architecture

The architecture of the system is designed as a sequential flow of interconnected modules. The input to the system is a raw URL provided either from the dataset during training or by the user during real-time prediction. This URL is first processed by the preprocessing and feature extraction modules, which transform it into a numerical feature vector. The processed features are then passed to trained machine learning models that classify the URL as phishing or legitimate. The output is presented to the user through a graphical web interface. This modular architecture ensures separation of concerns, making the system easier to debug, maintain, and scale.

3.2 Data Collection

The datasets used in this project consist of URLs labeled as phishing or legitimate, obtained from publicly available phishing repositories and benchmark datasets commonly used in academic research. Each dataset contains thousands of URL samples along with extracted lexical features representing the structural properties of URLs. The use of multiple datasets allows the system to learn diverse phishing patterns and reduces the risk of overfitting to a single data source. The binary labeling scheme, where phishing URLs are assigned label “1” and legitimate URLs are assigned label “0”, enables supervised learning and straightforward performance evaluation.

3.3 Data Preprocessing

Data preprocessing is a critical stage in the system design, as raw URL data often contains inconsistencies and class imbalance issues. Initially, the datasets are cleaned by removing missing values and invalid records. Since phishing datasets are typically imbalanced, with fewer phishing samples compared to legitimate ones, oversampling techniques are applied to balance the class distribution. This ensures that machine learning models do not become biased toward the majority class. Additionally, feature scaling and normalization are performed where required to improve convergence and performance, particularly for distance-based and margin-based algorithms such as KNN and SVM.

3.4 Feature Extraction

Feature extraction plays a central role in distinguishing phishing URLs from legitimate ones. Instead of relying on webpage content, the system focuses on URL-based lexical and structural features, which can be extracted quickly and without accessing external resources. Key features include URL length, number of subdomains, frequency of special characters (such as '@', '-', '.', and '//'), presence of IP addresses instead of domain names, and abnormal token patterns. These features capture common characteristics of phishing URLs, such as excessive use of special characters and deceptive domain structures. The selected feature set provides a strong balance between computational efficiency and classification accuracy.

3.5 Machine Learning Model Selection

To perform a comprehensive comparative analysis, multiple supervised machine learning algorithms are incorporated into the system design. Decision Trees are used for their interpretability and ability to model non-linear relationships. Support Vector Machines are included due to their effectiveness in high-dimensional feature spaces. Logistic Regression serves as a baseline linear classifier, while K-Nearest Neighbours offers a distance-based perspective. Naive Bayes is implemented as a probabilistic classifier assuming feature independence. Random Forest, an ensemble learning method, is included to combine the strengths of multiple decision trees and improve generalization. The inclusion of diverse algorithms ensures a thorough evaluation of different learning paradigms.

3.6 Model Training and Validation Strategy

Each machine learning model is trained using preprocessed and balanced datasets. The datasets are split into training and testing subsets using multiple train-test ratios to evaluate model robustness under varying conditions. During training, models learn decision boundaries based on extracted features, while validation is performed using unseen test data. This approach ensures that the models generalize well and do not simply memorize training samples. Hyperparameters are tuned where necessary to optimize performance while avoiding overfitting.

3.7 Performance Evaluation Metrics

The system design incorporates multiple evaluation metrics to provide a comprehensive assessment of model performance. Accuracy is used to measure overall classification correctness, while precision evaluates the reliability of phishing predictions. Recall is emphasized as a critical metric, as failing to detect phishing URLs can have severe security consequences. The F1-score is used to balance precision and recall, offering a single metric for comparative analysis. These metrics collectively ensure that the selected model performs reliably across different datasets and real-world scenarios.

3.8 Final Model Selection and Deployment Design

Based on performance evaluation across multiple datasets, the best-performing model is selected for deployment. The system is designed to integrate the trained model into a Flask-based web application that serves as a user-friendly interface. Users can input URLs into the application, which processes the input, extracts features, and returns the classification result in real time. This deployment design demonstrates the practical applicability of the system and highlights its potential for integration into larger cybersecurity frameworks such as browser extensions or email filtering systems.

4. System Implementation

The system implementation translates the conceptual design of the phishing URL detection framework into a functional and deployable machine learning application. The implementation is carried out using Python due to its extensive ecosystem of data science and machine learning libraries, which facilitate efficient data processing, model development, and deployment. The implementation follows a modular approach, where each stage of the system pipeline is independently developed and later integrated to form a complete solution. This modularity ensures code maintainability, scalability, and ease of enhancement.

4.1 Development Environment and Tools

The implementation environment consists of Python as the primary programming language, along with widely used open-source libraries. Pandas and NumPy are used for data handling and numerical computations, enabling efficient manipulation of large datasets. Scikit-learn is employed for implementing machine learning algorithms, preprocessing techniques, model training, and performance evaluation. Flask is used to develop a lightweight web application for real-time phishing URL detection, providing a user-friendly interface for interaction. This combination of tools ensures a balance between performance, flexibility, and ease of deployment.

4.2 Dataset Handling and Preparation

Each dataset is loaded into the system using Pandas data structures, allowing efficient access to features and labels. Initial exploratory analysis is performed to understand feature distributions, detect anomalies, and identify class imbalance. Missing or inconsistent records are removed to ensure data integrity. Since phishing datasets often exhibit imbalance between phishing and legitimate URLs, oversampling techniques are applied to increase the representation of minority classes. This step is critical to prevent biased learning and improve recall for phishing URLs.

4.3 Feature Processing and Transformation

The extracted URL-based features are transformed into a numerical feature matrix suitable for machine learning algorithms. Feature scaling and normalization are applied where required, particularly for algorithms such as K-Nearest Neighbours and Support Vector Machines, which are sensitive to feature magnitude. This transformation ensures that no single feature disproportionately influences the learning process. The processed feature vectors serve as the standardized input for all classifiers, enabling fair comparison across different models.

4.4 Machine Learning Model Implementation

Each machine learning algorithm is implemented independently using Scikit-learn. Decision Tree classifiers are constructed by recursively partitioning the feature space based on information gain, allowing the model to learn hierarchical decision rules. Support Vector Machines are implemented to identify optimal hyperplanes that separate phishing and legitimate URLs with maximum margin. Logistic Regression models probabilistic relationships between features and class labels using a sigmoid function. K-Nearest Neighbours classifies URLs based on proximity to neighboring samples in feature space, while Naive Bayes applies probabilistic inference under the assumption of feature independence. Random Forest is implemented as an ensemble of multiple decision trees trained on random subsets of data and features, significantly improving robustness and generalization.

4.5 Training and Testing Workflow

The datasets are divided into training and testing subsets using different train-test ratios to evaluate model stability. During training, models learn patterns associated with phishing behavior from labeled data. The trained models are then tested on unseen data to evaluate generalization performance. This separation ensures that evaluation metrics reflect real-world performance rather than memorization of training data. Hyperparameters such as tree depth, number of estimators, and neighborhood size are tuned to optimize performance while minimizing overfitting.

4.6 Performance Metric Computation

After prediction, model performance is evaluated using multiple statistical metrics. Accuracy measures the proportion of correctly classified URLs. Precision evaluates the correctness of phishing predictions, while recall measures the model's ability to detect all phishing URLs. The F1-score provides a harmonic balance between precision and recall, offering a comprehensive performance indicator. These metrics are computed for each model and dataset, enabling detailed comparative analysis and informed model selection.

4.7 Model Selection and Persistence

Based on experimental results, the Random Forest classifier is selected as the final model due to its consistent superiority across datasets and metrics. The trained model is serialized and stored using model persistence techniques, allowing it to be reused without retraining. This ensures efficient deployment and faster response during real-time prediction.

4.8 Web Application Integration

To demonstrate practical usability, the selected model is integrated into a Flask-based web application. The application accepts user-input URLs through a graphical interface, preprocesses the input, extracts relevant features, and feeds them into the trained model. The prediction result—phishing or legitimate—is displayed instantly to the user. This implementation highlights the feasibility of deploying machine learning–based phishing detection systems in real-world environments such as email gateways, browsers, and enterprise security platforms.

5. Results & Analysis

The performance of the proposed phishing URL detection system was evaluated through extensive experimentation on multiple datasets using several supervised machine learning algorithms. The objective of this evaluation was not only to measure classification accuracy, but also to analyze the reliability and robustness of each model in detecting phishing URLs, which is critical in real-world cybersecurity applications. Since false negatives (undetected phishing URLs) can have severe consequences, special emphasis was placed on recall and F1-score in addition to accuracy.

The evaluation metrics used for analysis include Accuracy, Precision, Recall, and F1-Score. Accuracy provides an overall measure of correct classification, while Precision indicates how many URLs classified as phishing are actually phishing. Recall measures the model's ability to detect all phishing URLs, and F1-score balances Precision and Recall, making it particularly suitable for imbalanced datasets such as phishing detection.

5.1 Dataset 1 – Performance Evaluation and Analysis

Model	Accuracy	Precision	Recall	F-1 score
Decision Tree	0.9620	0.9639	0.9600	0.9619
Random Forest	0.9775	0.9809	0.9740	0.9774
SVM	0.8665	0.8323	0.9180	0.8730
KNN	0.8830	0.8613	0.9130	0.8864
Naïve Bayes	0.8585	0.9367	0.7690	0.8446

[1] Phishing_Legitimate_full

Analysis:

The results for Dataset 1 clearly demonstrate the superiority of the Random Forest classifier, which achieves the highest accuracy (97.75%) along with excellent precision and recall values. This indicates that Random Forest is not only accurate but also highly reliable in detecting phishing URLs without generating excessive false alarms. Decision Tree also performs well, benefiting from its ability to capture non-linear decision boundaries, though it slightly underperforms compared to its ensemble counterpart.

Support Vector Machine and KNN show moderate performance, with relatively high recall but lower precision, indicating a tendency to misclassify legitimate URLs as phishing. Naive Bayes exhibits high precision but significantly lower recall, suggesting that while its phishing predictions are often correct, it fails to detect a substantial number of phishing URLs. This makes Naive Bayes less suitable for security-critical applications where missing malicious URLs is unacceptable.

5.2 Dataset 2 – Performance Evaluation and Analysis

Model	Accuracy	Precision	Recall	F-1 score
Decision Tree	0.9480	0.9574	0.9488	0.9531
Random Forest	0.9742	0.9696	0.9846	0.9770
SVM	0.9489	0.9402	0.9699	0.9548
KNN	0.9466	0.9477	0.9569	0.9523
Naïve Bayes	0.6006	0.9971	0.2835	0.4415

[2] Phishing Dataset

Analysis:

Dataset 2 further reinforces the effectiveness of ensemble-based learning. Random Forest again achieves the highest accuracy and the highest recall (98.46%), making it particularly effective in identifying phishing URLs. The extremely high recall indicates that very few phishing URLs escape detection, which is essential in real-world cybersecurity systems.

Naive Bayes presents an interesting case in this dataset, achieving extremely high precision but very poor recall. This implies that although nearly all URLs it flags as phishing are indeed phishing, it fails to detect the majority of phishing URLs. Such behavior is undesirable in phishing detection systems, where the cost of false negatives is much higher than false positives. Overall, Random Forest maintains the best balance between all performance metrics.

5.3 Dataset 3 – Performance Evaluation and Analysis

Model	Accuracy	Precision	Recall	F-1 score
Decision Tree	0.8854	0.8344	0.8543	0.8442
Random Forest	0.8861	0.8663	0.8119	0.8382
SVM	0.8663	0.8505	0.7668	0.8065
KNN	0.8778	0.8364	0.8252	0.8308
Naïve Bayes	0.7147	0.8784	0.2493	0.3883

[3] Web page Phishing Detection Dataset

Analysis:

Dataset 3 is comparatively more challenging, as reflected by the overall lower performance across all models. Despite this, Random Forest still achieves the highest accuracy and maintains balanced precision and recall values. The reduced performance suggests higher feature complexity or overlaps between phishing and legitimate URLs in this dataset, highlighting the importance of robust models capable of handling noisy or ambiguous data.

Naive Bayes again demonstrates poor recall, reaffirming its limitations for phishing detection. Decision Tree and KNN show competitive but slightly lower performance, while SVM struggles with recall in this dataset. The results confirm that ensemble-based models are more resilient to dataset variability and complexity.

5.4 Comparative Analysis and Key Observations

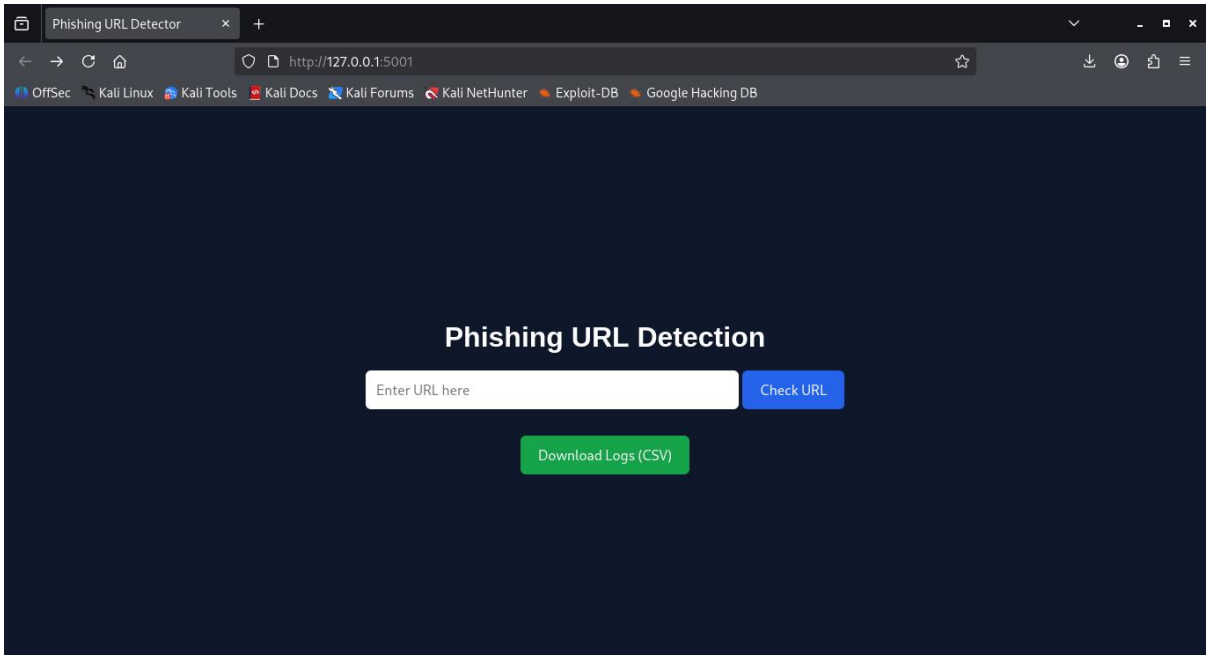
Across all datasets, several consistent patterns emerge:

- Random Forest consistently outperforms other classifiers in terms of accuracy and balanced performance.
- Ensemble learning effectively reduces overfitting and improves generalization.
- Naive Bayes, despite high precision in some cases, is unsuitable due to poor recall.
- High recall is critical for phishing detection, as undetected phishing URLs pose serious security risks.

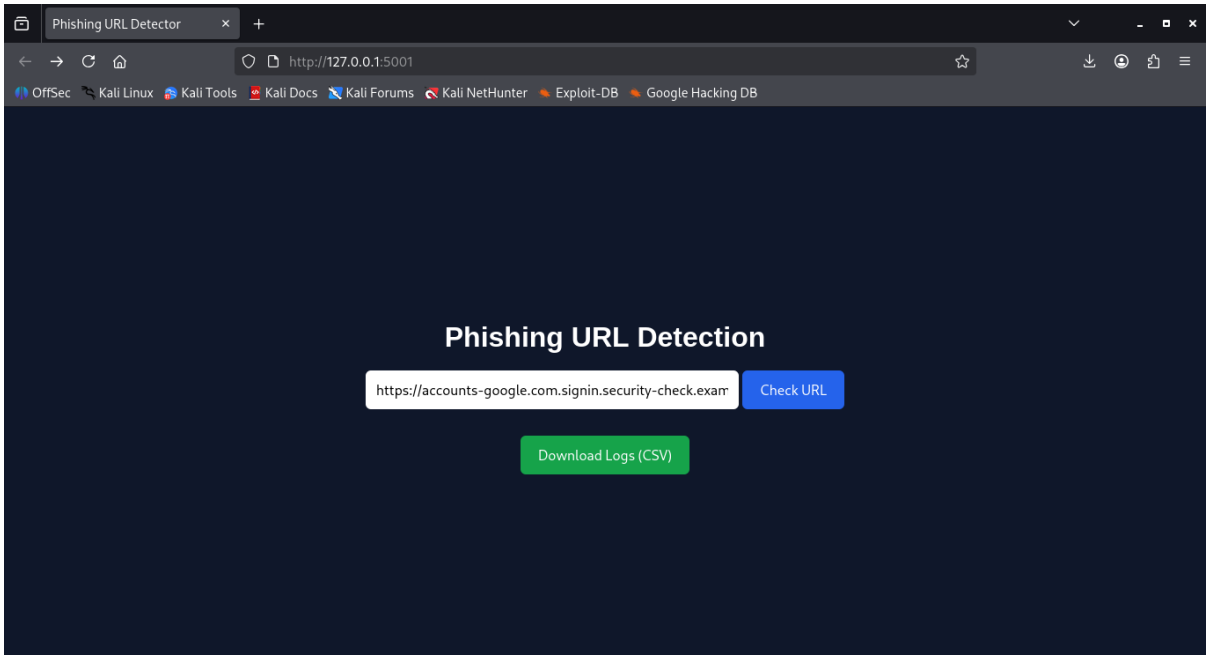
The experimental findings validate the choice of Random Forest as the final model for deployment and align closely with previously published research results in phishing URL detection.

5.5 Outputs

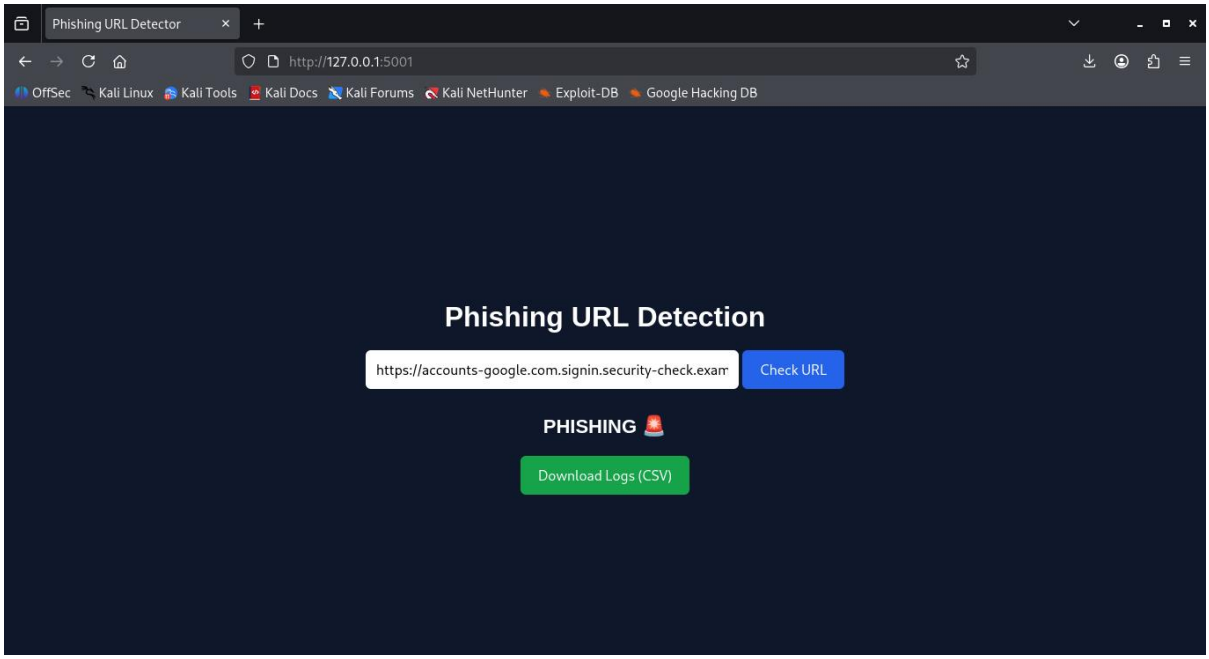
5.5.1 Home Page of the system.



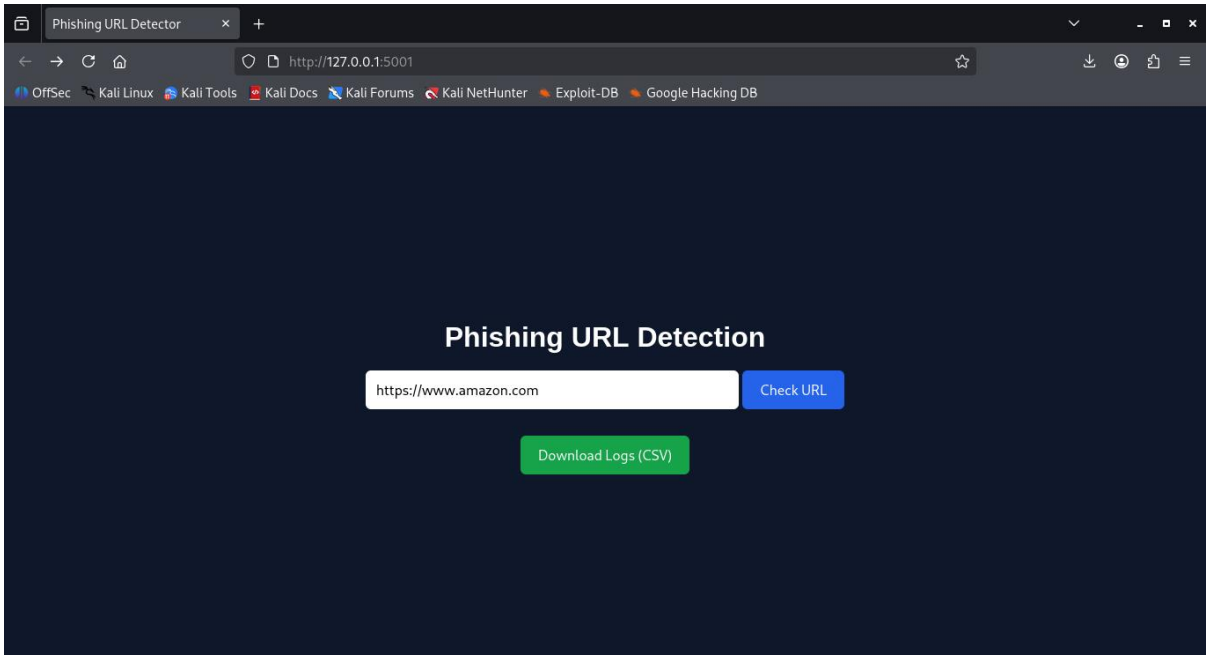
5.5.2 Link 1- Input first link in search bar.



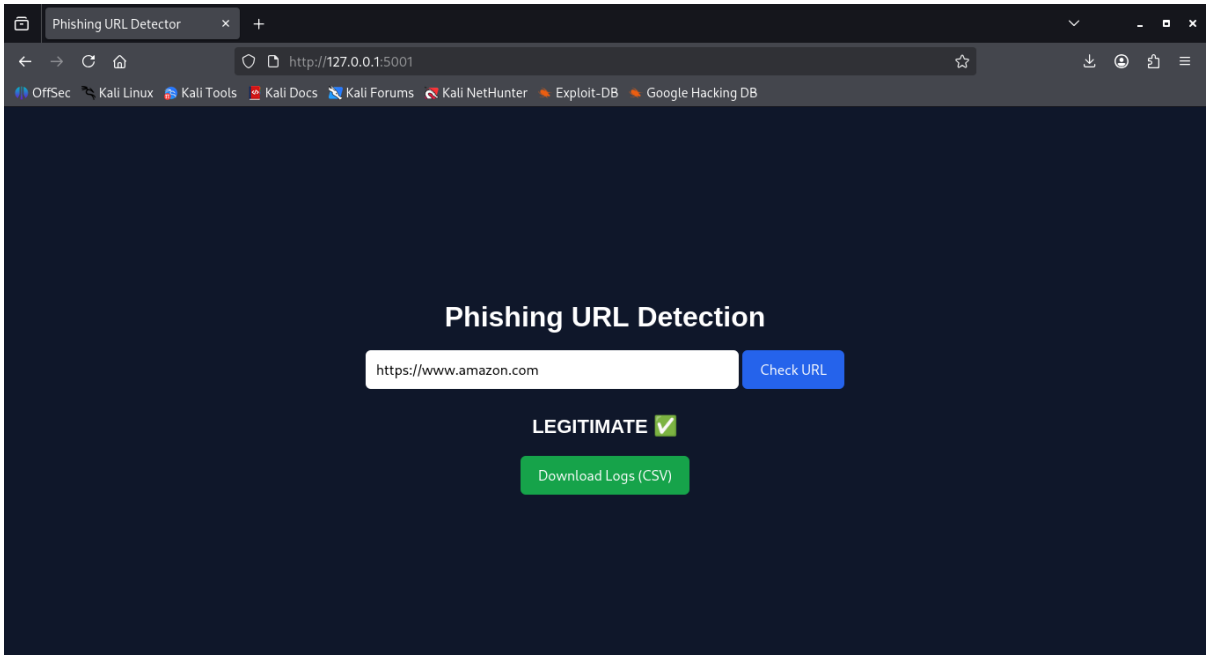
5.5.3 Result 1 – Result of first link is displayed.



5.5.4 Link 2 – Input second link in search bar.



5.5.5 Result 2 – Result of second link is displayed.



6. Conclusion & Future Scope

This project successfully demonstrates the effectiveness of machine learning techniques for phishing URL detection by leveraging URL-based lexical and structural features to accurately classify malicious and legitimate URLs. Through systematic experimentation on multiple datasets and a comparative evaluation of several supervised learning algorithms, the study establishes that ensemble-based methods, particularly the Random Forest classifier, consistently deliver superior and more reliable performance compared to individual classifiers. The results highlight that Random Forest achieves high accuracy while maintaining a strong balance between precision and recall, which is crucial in security-sensitive applications where undetected phishing URLs can lead to severe consequences. The modular design, robust preprocessing pipeline, and comprehensive evaluation strategy ensure that the proposed system is both scalable and adaptable. Overall, the project confirms that URL-based machine learning approaches offer a practical, efficient, and reliable solution for real-time phishing detection and can significantly enhance existing cybersecurity defenses.

The scope of this work can be extended by incorporating deep learning models for automated feature learning, integrating real-time browser or email gateway deployment, and combining URL-based analysis with webpage content and network-level features. Additionally, continuously updating the training datasets to include emerging phishing patterns can further improve detection accuracy and system resilience against evolving cyber threats.

7. References

1. Dataset 1 – https://www.kaggle.com/datasets/amj464/phishing
2. Dataset 2 – https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning
3. Dataset 3 – https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset
4. Github Repository Link - https://github.com/magnusec/URL-Phishing-Detection-using-Machine-Learning-Algorithms
5. Research Paper - https://ieeexplore.ieee.org/document/11208974