Phishing URL Detection Using Machine Learning and Web Interface

Rujuta Shetkar
Dept.of Information Technology
Sathaye College
rujutashetkar05@gmail.com

Pratiksha Swami
Dept.of Information Technology
Sathaye College
pratikshaswami781@gmail.com

***Abstract*— In the age of digitization, phishing attacks have emerged as a predominant cyber threat targeting individuals and organizations. These attacks rely heavily on deceptive URLs designed to trick users into revealing sensitive information such as login credentials and financial data. This paper presents the development and evaluation of a machine learning-based phishing URL detection tool featuring an interactive web interface. The system employs supervised learning techniques, trained on a labeled dataset of phishing and legitimate URLs, and incorporates key lexical and host-based features. The model is integrated into a Flask-powered web application that provides real-time detection results. Through rigorous evaluation, the system achieves a high detection accuracy of over 96%. This research demonstrates the feasibility of integrating machine learning into cybersecurity applications and emphasizes the practical, ethical, and market implications of such a tool.**

***Keywords*— *Phishing Detection, Machine Learning, Cybersecurity, URL Analysis, Random Forest, Web Application, Phishing URLs, Real-Time Detection, Feature Engineering, Flask Framework*.**

I. PROBLEM STATEMENT AND OBJECTIVE

Phishing attacks exploit human trust by masquerading as legitimate entities through fake websites or URLs. The increasing sophistication of phishing techniques, including the use of HTTPS, domain spoofing, and URL obfuscation, makes it challenging for average users to distinguish real from fake. Traditional detection methods, such as blacklists, are reactive in nature and cannot effectively counter rapidly emerging threats. As a result, there is a growing need for automated systems capable of identifying phishing attempts in real-time. The main objective of this research is to leverage machine learning to develop a system that can accurately classify URLs based on extracted features, offering a more proactive and intelligent defense mechanism. Additionally, the integration of this system with a web-based interface ensures its accessibility to a wider range of users, including those with limited technical expertise.

Phishing continues to be one of the most persistent and damaging forms of cybercrime, leveraging deceptive websites and spoofed URLs to manipulate users into divulging sensitive credentials, financial information, or installing malware. Traditional defense mechanisms such as browser blacklists, rule-based filters, and manual site reviews struggle to keep up with the sheer volume and sophistication of newly generated phishing domains, many of which are active for just a few hours. This creates a critical need for automated, scalable, and intelligent systems capable of detecting malicious URLs in real time. The objective of this research is to develop such a system—one that leverages supervised machine learning and domain-specific feature extraction to distinguish phishing URLs from legitimate ones based solely on their lexical and structural attributes. By focusing on URL-level analysis rather than page content, the system ensures faster processing, platform independence, and broader applicability across different devices and operating environments.

II. LITERATURE SURVEY

Phishing detection has been an area of intense research interest, especially as cybercriminal tactics become more sophisticated. Early detection systems primarily relied on blacklists and signature-based approaches, where URLs previously identified as malicious were stored and compared with new entries. Although fast and simple, these methods are reactive in nature and incapable of detecting new or cleverly disguised phishing URLs. Researchers soon recognized the limitations of these static methods and began exploring heuristics—rules based on suspicious patterns such as URL length, use of numbers, or uncommon domain extensions. However, heuristic systems struggled with high false-positive rates and lacked adaptability. The application of machine learning marked a significant advancement in phishing detection. Ma et al. (2009) demonstrated that lexical and host-based features could effectively distinguish between phishing and legitimate URLs using logistic regression. Mohammad et al. (2015) proposed a rule-based machine learning classifier with an expanded feature set and achieved promising results. More recently, Sahingoz et al. (2019) compared several supervised learning algorithms including Random Forests and Gradient Boosting, concluding that ensemble methods outperformed linear models in terms of accuracy and robustness. Meanwhile, deep learning methods, such as LSTM networks, have shown promise in analyzing URL character sequences, although their implementation remains complex and resource-intensive. Our research builds upon this body of work by adopting a Random Forest model, balancing performance and interpretability, and integrating it into a scalable web-based platform.

Recent studies have increasingly emphasized the efficacy of machine learning techniques in detecting phishing attacks by analyzing lexical and structural features of URLs. For instance, Sahingoz et al. (2019) demonstrated the potential of NLP-based URL analysis combined with classifiers such as Random Forest and Gradient Boosting, achieving over 95% accuracy. Other researchers explored hybrid approaches, such as combining blacklist verification with heuristic and ML techniques to improve real-time detection (Abdelhamid et al., 2014). Notably, Sharma et al. (2021) investigated the use of URL shorteners and suspicious top-level domains as significant predictors of phishing behavior, echoing the features used in this study. These contributions form the foundation upon which our work builds, aiming to improve generalizability and ease of deployment through lightweight, explainable, and real-time classification strategies using only URL-based features—without requiring webpage content or network overhead.

III. Research Methodology

The research methodology adopted for this project is grounded in a systematic, data-driven approach to solving the problem of phishing URL detection using supervised machine learning. The methodology encompasses all phases of the machine learning pipeline—beginning with the identification and acquisition of a reliable dataset, followed by rigorous preprocessing, detailed feature engineering, model training, and real-world deployment through a web-based interface. The primary goal is to create an end-to-end solution that is both accurate and efficient, capable of classifying URLs in real time based purely on lexical and structural characteristics. This approach eliminates the need for content scraping or network-level analysis, which can be resource-intensive or restricted due to security and privacy concerns. Each stage of the methodology was carefully designed to preserve data integrity, maintain feature consistency, and optimize performance. The final model was integrated into a lightweight Flask web application to ensure accessibility, ease of use, and rapid prediction feedback for users. This methodology not only ensures the academic rigor of the research but also emphasizes practical applicability and real-world effectiveness in detecting phishing threats.

A. Data Pre-processing:

The foundation of this research is the phishing_site_urls.csv dataset, sourced from Kaggle. This dataset contains a comprehensive and balanced collection of URLs labeled as either phishing (1) or legitimate (0). It includes over 10,000 samples, providing ample data for robust model training. Each record is a simple string-label pair, making it easy to manipulate and integrate into Python-based ML workflows. The selection of this dataset was based on several factors: availability, credibility, real-world relevance, and compatibility with supervised learning techniques. The dataset reflects actual URLs collected from public blacklists and verified legitimate sources, offering high-quality input for training a model that generalizes well across unseen data.

B. Data Pre-Processing:

Data preprocessing was essential to ensure consistency, reduce noise, and make the data suitable for machine learning. Initially, the dataset was inspected for duplicate URLs, null entries, and formatting inconsistencies. All URLs were converted to lowercase to standardize feature extraction, especially for keyword-based checks. URLs were validated using regular expressions to remove malformed entries. The dataset was then stratified into training and test sets using an 80/20 split to maintain the class balance. This was crucial in preventing bias and ensuring fair performance metrics across both phishing and legitimate samples. In addition, labels were encoded to a binary format (0 and 1), and URLs were retained as-is without tokenization, since the classification is based on structural features rather than semantic interpretation.

C. Feature Engineering:

Feature engineering played a pivotal role in the effectiveness of the model. Instead of relying on content-based features (e.g., page text or HTML), which require more resources and introduce privacy risks, the model uses handcrafted features extracted directly from the URL string. These features are:

- URL Length (url_length): Measures the total character count in the URL. Phishing URLs tend to be unusually long to disguise their intent.
- Presence of "@" Symbol (has_at): Indicates a potential attempt to redirect or obscure the actual domain.
- Presence of IP Address (has_ip): Flags whether the URL contains a raw IP instead of a domain name—common in phishing.
- Use of HTTPS (has_https): While HTTPS is a sign of security, its presence is still tracked to analyze trends in phishing URLs.
- Phishing Keywords (phishing_keywords): Checks for high-risk terms such as login, verify, secure, bank, etc.
- Suspicious Extensions (suspicious_extension): TLDs like .tk, .ml, .xyz are often used in free domain registration for phishing.
- Subdomain Count (subdomain_count): A high number of subdomains can indicate trickery or attempts to mimic legitimate domains.
- Use of URL Shorteners (uses_shortener): Services like bit.ly and t.co are commonly used to mask malicious links.
- Digit-to-Letter Ratio (digit_letter_ratio): A high digit content may indicate randomly generated or suspicious domain names.

These features were engineered using Python and verified with test samples to ensure consistency. A features_list.pkl file was created to preserve feature ordering during model prediction.

D. Model Selection and Training:

Several machine learning algorithms were evaluated to determine the best fit for this classification task. Algorithms such as Logistic Regression, Naive Bayes, and SVM were tested initially. However, Random Forest outperformed others in terms of both accuracy and generalization. Random Forest, an ensemble learning method, was selected for its ability to handle non-linear relationships, robustness to overfitting, and interpretability. The model was trained on the processed dataset using Scikit-learn, with class weights balanced to address any remaining minor class imbalance. The model training included 5-fold cross-validation to ensure that results were stable and not biased by data splits. Feature importance was also derived, confirming the relevance of features like url_length, has_ip, and phishing_keywords.

E. Hyper parameter tuning:

To enhance model performance, hyperparameters were tuned using a Grid Search strategy. The following hyperparameters were adjusted:

- n_estimators: Number of decision trees in the forest.
- max_depth: Maximum depth of each tree.
- min_samples_split: Minimum samples required to split a node.
- criterion: Metric used for node impurity (gini or entropy).

The best-performing configuration was selected based on the F1-score and ROC-AUC values, ensuring a balance between precision and recall. The final model demonstrated a high

degree of stability, minimizing both false positives and false negatives.

F. Model Evaluation:

In the evaluation phase of this research, model performance was assessed using accuracy as the primary metric. Accuracy measures the proportion of correctly classified URLs—whether phishing or legitimate—out of the total number of predictions. This metric was selected for its simplicity and effectiveness in balanced classification tasks, which applies to our dataset due to its nearly equal distribution of phishing and legitimate URLs. The trained Random Forest model was evaluated on a separate test set comprising 20% of the dataset. The model achieved a commendable accuracy of over 96%, indicating its strong ability to correctly identify malicious URLs based on structural and lexical characteristics alone.

While additional metrics such as precision, recall, and F1-score provide deeper insights, this study prioritized accuracy to maintain simplicity in implementation and interpretation. Furthermore, the confusion matrix was not explicitly computed, as the high accuracy score already suggested limited misclassifications.

This approach makes the model well-suited for real-world applications where a high correct classification rate is critical for user trust and safety. However, for future improvements, incorporating more advanced evaluation metrics and visual tools like ROC curves or confusion matrices could help identify edge cases and fine-tune model sensitivity.

G. Real Time- Testing:

To ensure that the model performs effectively in real-world scenarios, a dedicated real-time testing phase was conducted through the deployed Flask-based web interface. This interface allows users to input any URL into a simple form and receive an instant classification result—either "⚠️ This URL is likely PHISHING" or "✅ This URL appears LEGITIMATE". The goal of this phase was to validate whether the model, when removed from the controlled training environment, still exhibits robust and consistent behavior. Multiple categories of URLs were tested, including those from public phishing databases (such as PhishTank) and safe domains like banks, e-commerce sites, and social media platforms. The model successfully flagged suspicious links containing obfuscated domain names, excessive subdomains, and phishing-related keywords. Legitimate URLs were also accurately classified, with only a small number of edge cases triggering false positives due to unusual patterns in the domain structure (e.g., long legitimate URLs or brand names with digits). The real-time performance was found to be highly responsive, with most predictions returned within a fraction of a second. This responsiveness is critical for usability, especially in security applications where time-sensitive decision-making is required. The lightweight design of the Flask app, combined with the pre-trained model and streamlined feature extraction process, ensured that the solution could be deployed on minimal infrastructure and accessed by end-users without requiring any technical expertise.

IV. Tool Implementation

The implementation of the phishing URL detection tool was designed to be lightweight, efficient, and accessible for end users via a web interface. The project integrates machine learning techniques with a user-friendly web deployment using Python-based technologies. The system is divided into three major components: backend model training, feature extraction, and real-time web-based prediction.

A. Programming Language and Environment:

The core programming language used for this project is Python due to its extensive support for machine learning and web development. The model training, data preprocessing, feature engineering, and serialization were performed using Python 3.10. The environment was set up using Jupyter Notebook for experimentation and VS Code for structured development.

B. Libraries and Frameworks Used:

Several Python libraries and frameworks were employed throughout the development lifecycle:

- Pandas: For data loading, manipulation, and analysis of the phishing_site_urls.csv dataset.
- Scikit-learn: Used for building, training, and serializing the Random Forest classification model (model.pkl), as well as preprocessing and feature scaling.
- Pickle: To serialize both the trained model and the list of selected features (features.pkl).
- Flask: A lightweight Python web framework used to create the real-time web interface that communicates with the model and displays results to the user.
- HTML/CSS: For front-end rendering of the interface (index.html), providing a clean and intuitive user experience.

C. Key Feature Extraction:

To ensure the model remains efficient and content agnostic, only URL-based features are used. These functions are implemented in Python and include:

- contains_phishing_keywords(url)
- suspicious_extension(url)
- subdomain_count(url)
- has_shortener(url)
- digit_letter_ratio(url)
- plus direct checks like URL length, presence of '@', and HTTPS usage.

Each of these functions is crafted to produce binary or numerical features that are easily interpretable by the model and efficient to compute at runtime.

V. Ethical Impact and Market Relevance

The deployment of phishing URL detection systems carries significant ethical importance in the broader context of cybersecurity and digital trust. In an increasingly interconnected world, users are regularly exposed to deceptive links through emails, social media, and websites—often without the technical knowledge to differentiate between legitimate and malicious sources. The proposed solution directly addresses this issue by empowering users to make informed decisions, thereby reducing the likelihood of

identity theft, data breaches, and financial fraud. By operating purely on URL structures and avoiding content scraping or user behavior tracking, the tool respects individual privacy and data protection norms. No personally identifiable information (PII) is stored or analyzed, ensuring compliance with legal standards such as the GDPR and India's Personal Data Protection Bill. Furthermore, the system does not rely on blacklists that may introduce bias or delay in identifying new threats, but instead uses intelligent, feature-based classification that works across a broad spectrum of domains and attack strategies. Ethically, this model promotes digital literacy and user agency by serving as a decision support tool rather than a black-box authority. The clear and simple interface also ensures that users of all backgrounds—technical or otherwise—can understand and utilize the tool effectively without requiring prior cybersecurity knowledge.

Phishing attacks are one of the most prevalent and damaging forms of cybercrime, accounting for billions of dollars in global losses each year. As remote work, online banking, and e-commerce continue to grow, so does the attack surface for phishing. Consequently, there is a rapidly growing market for lightweight, intelligent, and real-time phishing detection tools that can be embedded in browsers, email clients, or standalone security platforms.

The proposed system aligns well with market needs for the following reasons:

- **Real-time Response**: Its fast and accurate prediction capability makes it suitable for integration into security software, browser extensions, and corporate firewalls.
- **Low Resource Requirement**: The model's efficiency and independence from external API calls or content parsing make it scalable and easy to deploy in both cloud-based and edge computing environments.
- **Open Integration Potential**: Built with Python and Flask, the solution is highly customizable and open to integration with enterprise-level software or third-party applications.
- **Educational and Awareness Value**: The tool also serves as an educational aid in cybersecurity training programs, increasing public awareness of phishing threats.

Given these capabilities, the system has the potential for wide adoption in sectors ranging from education and banking to government and healthcare—any environment where secure online communication is critical.

VI. Future scope

The current system demonstrates high accuracy and real-time usability in detecting phishing URLs using handcrafted lexical and structural features. However, several enhancements can be incorporated in future iterations to further improve performance, coverage, and adaptability to evolving threats.

First, integrating advanced machine learning models such as gradient boosting algorithms (e.g., XGBoost, LightGBM) or deep learning architectures (e.g., LSTMs for sequential URL analysis) could improve accuracy in edge cases where current feature-based heuristics may fall short. These models may better capture subtle patterns in phishing URLs that static rules might miss.

Second, expanding the feature set to include contextual and behavioral indicators—such as DNS information, WHOIS data, SSL certificate validation, and domain age—would enable more comprehensive detection. While these may increase computational load, they can significantly reduce false positives and improve robustness against more sophisticated phishing campaigns.

Third, the tool could be enhanced to support browser extensions or email client plugins, enabling users to detect phishing links before clicking on them. This would allow for proactive threat mitigation across various platforms like Gmail, Outlook, or Chrome, creating real-world defense points where users interact most.

Fourth, the system could be deployed in an enterprise environment with an admin dashboard for monitoring phishing detection logs, alerting on suspected attacks, and feeding real-time phishing data back into model training loops for continual learning.

Lastly, implementing periodic retraining pipelines to keep the model updated with the latest phishing techniques and URL patterns scraped from trusted threat intelligence feeds (like PhishTank, OpenPhish, and Spamhaus) will ensure the system remains effective in a rapidly evolving cybersecurity landscape.

In conclusion, this tool provides a strong foundational model for phishing detection and sets the stage for future innovation that combines machine learning, threat intelligence, and human-centric design for a safer digital environment.

VII. Conclusion

In this research project, we have successfully developed a machine learning-based phishing URL detection system integrated with a web interface for real-time evaluation. By leveraging lexical and structural features of URLs and training a Random Forest classifier on the publicly available *phishing_site_urls.csv* dataset from Kaggle, the system achieves reliable performance in distinguishing phishing websites from legitimate ones. The lightweight Flask-based implementation ensures usability and responsiveness, making it suitable for deployment in educational, corporate, or personal environments. Through extensive evaluation—both offline using test datasets and online through real-time user input—the tool demonstrated strong accuracy and practical relevance. Beyond technical success, this project also contributes to raising cybersecurity awareness by providing an accessible and interpretable solution for end-users. While the model presently uses only static features, it lays a solid foundation for future expansions such as integrating dynamic signals, threat intelligence feeds, and adaptive learning mechanisms. The tool serves as a stepping stone toward more comprehensive anti-phishing solutions, addressing a critical and growing threat in the digital ecosystem. Overall, this project illustrates the power and promise of applying artificial intelligence to solve real-world security problems in an ethical and user-friendly manner.

References

1. Jain, A. K., & Gupta, B. B. (2018). Phishing detection: analysis of visual similarity based approaches. Security and Privacy, 1(1), e7. https://doi.org/10.1002/spy2.7

2. Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. arXiv preprint arXiv:1701.07179. https://arxiv.org/abs/1701.07179

3. Basnet, R., Mukkamala, S., & Sung, A. H. (2008). Detection of phishing attacks: A machine learning approach. In Soft Computing Applications in Industry (pp. 373–383). Springer. https://doi.org/10.1007/978-3-540-77465-5_23

4. Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based on rough set theory. Expert Systems with Applications, 41(13), 5948–5959. https://doi.org/10.1016/j.eswa.2014.03.061

5. Marchal, S., Saari, K., Singh, N., & Asokan, N. (2016). Know your phish: Novel techniques for detecting phishing sites and their targets. In 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS) (pp. 323–333). IEEE. *https://doi.org/10.1109/ICDCS.2016.*