International Journal for Modern Trends in Science and Technology
Volume 11, Issue 09, pages 71-78.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol11issue09.html
DOI: https://doi.org/10.5281/zenodo.17148931

Machine Learning for Web Vulnerability Detection

SK.K.K.B Vali Basha[1] , V.Navyasri[2], S.Ramya[2], Ch.Vanitha[2], S.Navya[2]

[1]*Associate Professor Department of CSE, Vijaya Institute of Technology for Women, Enikepadu, AP, INDIA.*
[2]*Department of CSE, Vijaya Institute of Technology for Women, Enikepadu, AP, INDIA.*

To Cite this Article
SK.K.K.B Vali Basha , V.Navyasri, S.Ramya, Ch.Vanitha & S.Navya (2025). Machine Learning for Web Vulnerability Detection. International Journal for Modern Trends in Science and Technology, 11(09), 71-78. https://doi.org/10.5281/zenodo.17148931

Article Info
Received: 07 August 2025; Accepted: 31 August 2025.; Published: 05 September 2025.

KEYWORDS

Machine Learning,
Random Forest,
Logistic Regression,
Deep Learning,
Cross-site Scripting.

ABSTRACT

With the rapid expansion of web applications, cybersecurity threats have escalated, making web vulnerability detection a crucial aspect of modern security. Traditional techniques, including static and dynamic analysis, often suffer from limited scalability, high false positives, and an inability to detect zero-day vulnerabilities. To overcome these challenges, this project employs machine learning-based automated web vulnerability detection, enhancing accuracy and adaptability.

By training models on historical security vulnerabilities, the system analyzes source code, web requests, and user inputs to predict potential security risks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Remote Code Execution (RCE). Using supervised learning algorithms like Random Forest, Support Vector Machines (SVM), and Deep Neural Networks (DNNs), it significantly improves detection efficiency.

This AI-driven approach minimizes manual security assessments while enabling real-time identification of vulnerabilities. Unlike rule-based security tools, the system learns dynamically from new attack patterns and cyber threat intelligence feeds, making it highly adaptable. Future enhancements will integrate deep learning, adversarial training, and automation tools, ensuring robust web application security against evolving cyber threats.

1. INTRODUCTION

The rapid With growth of the internet, web security has become a significant concern for organizations and individuals alike. The digital world is expanding at an unprecedented rate, making it an essential part of everyday life. From online banking to e-commerce, social media, and cloud computing, users rely on web-based platforms for various activities. However, this expansion has also led to an increase in cyber threats, including malicious URLs, phishing attacks, malware infections, and data breaches.

Cybercriminals exploit vulnerabilities in web systems to conduct attacks that compromise sensitive data, disrupt services, and cause financial losses. Traditional security mechanisms such as firewalls and blacklists have been used to prevent such threats, but they are no longer sufficient in tackling advanced and evolving cyber threats.

One of the most common and dangerous attack vectors is the use of malicious URLs, which are fraudulent web addresses designed to deceive users into revealing personal information or downloading harmful software. Attackers manipulate URLs to make them appear legitimate, often using typosquatting, shortened links, and obfuscated domains to evade detection. As traditional methods struggle to keep up with the dynamic nature of these attacks, Machine Learning (ML) has emerged as an effective solution for automated web vulnerability detection.

This project focuses on developing a Machine Learning-based system for detecting malicious URLs. The system extracts features from URLs, analyzes patterns, and classifies them as either benign or malicious. By leveraging Flask for deployment, this solution provides an efficient, real-time web security mechanism that can help prevent cyber threats.

1.1. Importance of Web Security

Web security is a critical aspect of modern cybersecurity due to the increasing reliance on internet-based services. The following factors highlight the importance of web security:

1.2.1 Protecting Sensitive Information

With interact web platforms by sharing sensitive data such as login credentials, financial details, and personal information. If web security is compromised, this data can be stolen and misused, leading to identity theft, financial fraud, and reputational damage. Organizations handling large volumes of user data, such as banks, healthcare providers, and e-commerce platforms, must ensure robust security mechanisms to protect user privacy.

a) 1.2.2 Preventing Malware Attacks

Hackers use malicious websites to distribute malware, such as ransomware, spyware, and trojans. Unsuspecting users clicking on malicious links may unknowingly install malware that can steal data, corrupt files, or hijack computer systems. Web security measures, including machine learning-based URL detection, help in preventing malware infiltration and mitigating cyber risks.

1.2.3 Ensuring Business Continuity

Cyberattacks can lead to **website downtime, data loss, and system failures**, affecting business operations and customer trust. Organizations must prioritize web security to **ensure seamless service delivery** and protect their brand reputation. **E-commerce platforms, financial institutions, and government agencies** are particularly vulnerable to cyber threats, making it crucial to implement **automated threat detection systems**.

1.2.4 Combating Phishing and Fraudulent Activities

Phishing is one of the most prevalent cyber threats where attackers trick users into providing confidential information by impersonating legitimate websites. Fake login pages, fraudulent payment portals, and social engineering techniques are commonly used to deceive users. Machine Learning models trained to detect suspicious URL patterns can help mitigate phishing attacks and improve overall cybersecurity.

1.2.5 Compliance with Cybersecurity Regulations

Many industries are required to comply with **data protection laws and cybersecurity regulations**, such as:

- **General Data Protection Regulation (GDPR)** – Protects user privacy in the European Union.
- **Health Insurance Portability and Accountability Act (HIPAA)** – Ensures the security of patient data in the healthcare sector.
- **Payment Card Industry Data Security Standard (PCI-DSS)** – Protects online payment transactions.

Failure to adhere to these regulations can result in **legal penalties, financial losses, and reputational damage**. Implementing advanced **web security solutions** such as ML-based **malicious URL detection** helps organizations stay compliant with cybersecurity frameworks.

1.3 Rise of Malicious URLs

1.3..1 What Are Malicious URLs?

A malicious URL is a web link created with the intention of harming users or compromising computer systems. Attackers craft deceptive URLs to steal sensitive data, spread malware, or redirect users to fraudulent sites. These URLs often appear legitimate, making it difficult for users to differentiate between safe and harmful links.

1.3.2 Common Types of Malicious URLs

1.Phishing URLs – Imitate trusted websites to steal usernames, passwords, and banking details.

2.Malware-hosting URLs – Distribute harmful software such as viruses, trojans, and spyware.

3. Redirect URLs – Automatically send users to an unauthorized destination, leading to fraudulent websites.

4. Shortened URLs – Used to disguise malicious intent, often found in emails and social media.

5.Drive-by Download URLs – Trigger the automatic installation of malware when visited.

(1) 1.3.3 Techniques Used by Cybercriminals

Hackers utilize various techniques to make malicious URLs appear authentic, including:

- **Typosquatting** – Registering domain names similar to legitimate ones (e.g., **g00gle.com** instead of **google.com**).
- **Obfuscation** – Using **random characters** to disguise URLs (**bit.ly/randomcode**).
- **Unicode Spoofing** – Using **Unicode characters** that look similar to real letters.
- **HTTPS Manipulation** – Fake websites using HTTPS to **appear secure** but still serve malware.

1.3.4 Challenges in Detecting Malicious URLs

Detecting malicious URLs is challenging because attackers constantly modify domain structures, content, and redirections to bypass traditional security measures. Signature-based methods fail to detect zero-day threats (newly emerging attacks), making machine learning-based detection a promising solution.

1.4 Role of Learning Machine in Cybersecurity

Machine Learning algorithms analyze patterns in URLs, domain names, and website behaviors to classify them as either benign or malicious. Unlike traditional rule-based methods, ML models:

- Continuously **learn from new threats**.
- Detect **previously unknown attacks**.
- Identify **subtle malicious patterns** that human analysts might miss.

Proposed System

System Overview

With the increasing complexity and sophistication of web-based attacks, traditional security mechanisms like signature-based intrusion detection and rule-based firewalls struggle to keep up with evolving threats. To address these limitations, we propose an intelligent Machine Learning (ML)-based Web Vulnerability Detection System that can effectively identify and mitigate cyber threats in real time.

2. System Architecture

The proposed system consists of the following key components:

URL Feature Extraction Module: Extracts key features from a given URL, such as length, number of special characters, subdomain depth, and presence of suspicious patterns.

- **Machine Learning Classifier:** Uses a trained model to classify URLs as malicious or benign.
- **Real-Time Detection System:** Deploys the ML model in a web application that can predict web vulnerabilities dynamically.
- **Logging and Monitoring System:** Maintains logs of detected vulnerabilities for security analysis and improvement

3.Working Mechanism

1.Data Collection & Preprocessing:

- The system gathers data from web traffic logs, security databases, and user inputs.
- ○URLs are analyzed based on extracted features like domain structure, length, entropy, and character frequency.

2.Feature Extraction and Selection:

- Key features influencing malicious web behaviors are extracted, such as:
- Presence of IP addresses in URLs (often linked to phishing).

- Number of redirections (can indicate malicious intent).
- Use of uncommon Top-Level Domains (TLDs) (e.g., .xyz, .top).

1.Model Training and Classification:

- A machine learning model (Random Forest, SVM, or Deep Learning model) is trained on labeled datasets to classify web vulnerabilities.
- The model learns from past attack patterns and continuously updates itself to recognize new threats.

2.Deployment & Real-Time Detection:

- The trained model is deployed in a **Flask-based web application** to assess URLs in real-time.
- Users can submit a URL, and the system will instantly classify it as "Malicious" or "Benign."

Advantages Over Existing Solutions

The proposed system offers several advantages over traditional web security mechanisms:

Real-Time Detection

Unlike traditional static rule-based security measures, our system **analyzes URLs in real-time** and instantly provides classification results.

- This is crucial for protecting users from phishing attacks, drive-by downloads, and other web-based threats.

Adaptability to Evolving Threats

Traditional security methods struggle with zero-day vulnerabilities and new attack techniques.

- Our **machine learning model continuously learns** from new data, ensuring it remains effective against emerging cyber threats.

System Architecture

The **Machine Learning-Based Web Vulnerability Detection System** is designed to classify URLs as **malicious or benign** based on extracted features. The architecture consists of the following components:

Key Components of the System

1.User Interface (UI) & Web Application

- A web-based front-end where users submit URLs for classification.
- Built using Flask, allowing interaction with the backend ML model.

2.Data Collection & Preprocessing Module

- Gathers labeled datasets of malicious and benign URLs from sources like:
  - OpenPhish
  - PhishTank
  - Alexa Top Sites
- Cleans and preprocesses data for feature extraction.

3.Feature Extraction Module

- Extracts relevant **URL-based, lexical, and domain-related features**, including:
  - URL Length
  - Number of Special Characters
  - Presence of Suspicious Keywords (e.g., "login", "secure", "bank")
  - Use of IP Addresses instead of Domains

Prediction & Decision Engine

- Takes extracted features as input and applies the trained model.
- Outputs a classification result: "**Malicious**" **or** "**Benign.**"

Logging & Monitoring System

- Maintains records of classified URLs.
- Allows security analysts to track and analyze attack trends.

Model Training & Continuous Learning

- Periodic retraining of the model using updated
- threat intelligence. ○ Ensures adaptability to evolving cyber threats.

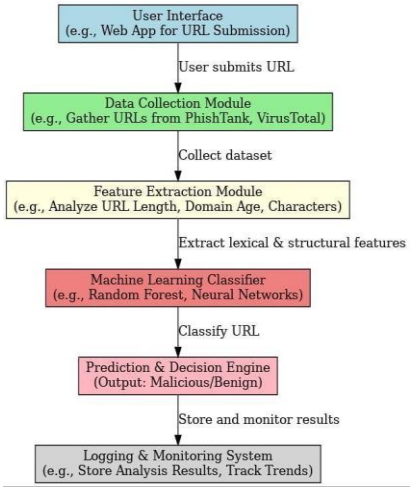


Fig. System Architecture

Workflow of URL Classification

The **URL classification workflow** follows a step-by-step process, from user input to final classification.

Step-by-Step URL Classification Process

User Step 1: Input (URL Submission)

- The user submits a URL for classification via the web application interface.
- The request is sent to the backend for processing.

Step 2: Feature Extraction

● The system extracts relevant **lexical and structural features** from the URL, including:

- ○ **Length of URL** (Longer URLs may be suspicious)
- ○ **Number of Special Characters** (Hyphens, dots, slashes)
- ○ **Presence of Keywords** (e.g., "bank", "secure", "login")
- ○ **Use of IP Addresses instead of Domain Names**
- ○ **Domain & Subdomain Length**

Step 3: Preprocessing & Normalization

- Extracted features are **standardized and normalized** to ensure consistency.
- Features are converted into a format suitable for the ML model.

Step 4: Machine Learning-Based Classification

The extracted features are passed through the trained **ML model** (Random Forest, Decision Tree, or Neural Network).

- The model predicts whether the URL is **benign or malicious** based on its learned patterns.

Step 5: Prediction Output

The ML model provides a classification result:

- ○ "**Benign**" → Safe URL
- ○ "**Malicious**" → Potential phishing/malware link

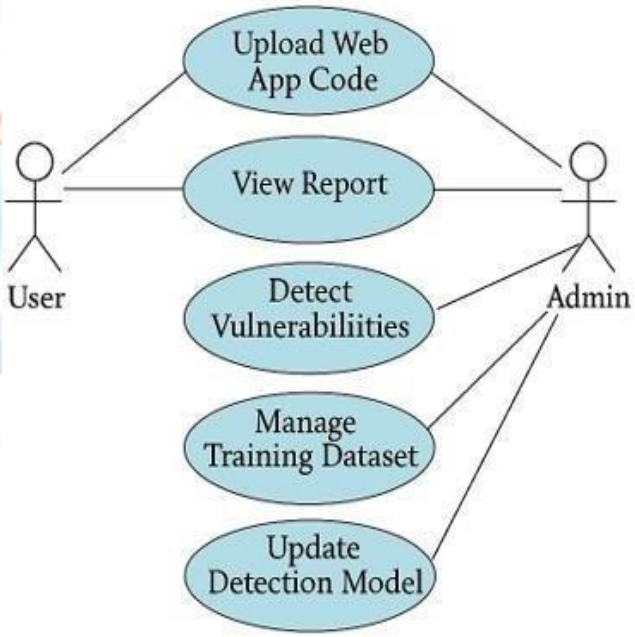
- The system returns this classification to the user through the web application.

Step 6: Logging & Security Monitoring

● The system **logs detected malicious URLs** for further analysis. ● Data is used for **retraining the model** and improving accuracy.

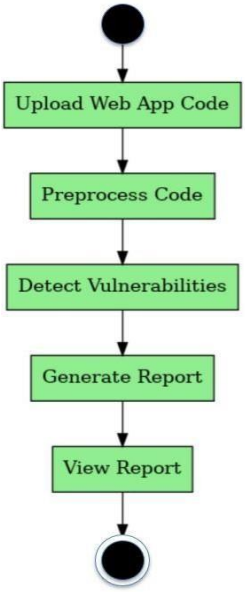
Use case Diagram:

This use case diagram illustrates the interaction between two main actors, the User and the Admin, in a web application vulnerability detection system. The User can upload web app code and view the generated report. The Admin has additional privileges, including detecting vulnerabilities, managing the training dataset, and updating the detection model. Both the User and Admin share access to uploading code and viewing reports, ensuring collaborative functionality while maintaining administrative control for system updates and analysis accuracy.



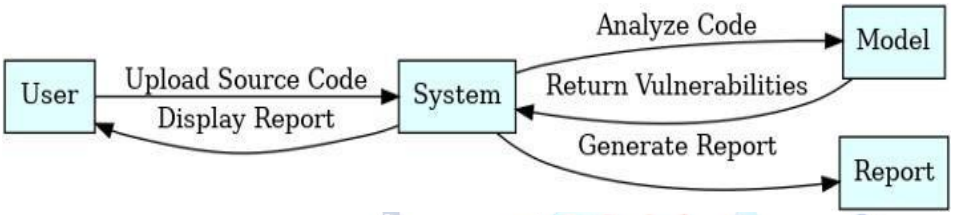
Activity Diagram:

activity diagram represents the step-by-step process of detecting in a web application. It starts with uploading the web app code, followed by preprocessing the code to prepare it for analysis. Next, the system detects vulnerabilities within the code. Once detection is complete, a report is generated based on the findings. Finally, the user can view the report to understand the vulnerabilities identified in the web application.



If drowsiness is detected for a sustained period, the system proceeds to trigger an alert.

Fig-ROC Curve for drowsiness detection

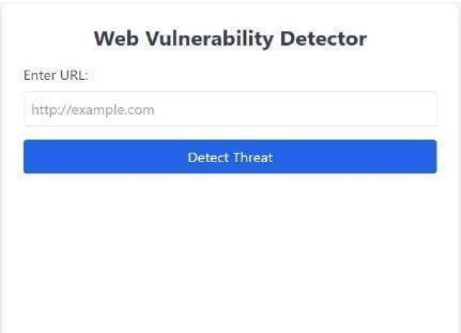


This sequence diagram illustrates the interaction between the User, System, Model, and Report in the process of vulnerability detection. The User begins by uploading the source code to the System. The System then sends the code to the Model for analysis. After the Model analyzes the code, it returns the identified vulnerabilities to the System. The System uses this information to generate a detailed report and finally displays the report back to the User.

Results & Evaluation

The given Flask-based machine learning application classifies URLs as either **Malicious** or **Benign** based on extracted features. The evaluation of the model's performance involves analyzing accuracy, precision, recall, and comparing it with other models.

INPUT SCREENS



(2) OUTPUT SCREENS

Fig:- Prediction of Benign URL

Fig. 15.2. Prediction of Malicious URL

(a) Accuracy, Precision, Recall Analysis

Accuracy:

Accuracy measures the overall correctness of the model's predictions.

True Positives (TP): Malicious URLs correctly classified as malicious.

- **True Negatives (TN):** Benign URLs correctly classified as benign.
- **False Positives (FP):** Benign URLs incorrectly classified as malicious.
- **False Negatives (FN):** Malicious URLs incorrectly classified as benign.

A high accuracy score indicates the model's effectiveness in distinguishing between safe and harmful URLs.

Precision:

Precision determines how many of the URLs classified as malicious were actually maliciousA high precision value means the model has fewer false positives, ensuring legitimate websites are not wrongly flagged.

Recall (Sensitivity):

Recall measures how many actual malicious URLs were correctly identified.A high recall means that the model detects most of the actual threats, reducing the risk of missing dangerous URLs.

F1 Score:

The F1 Score is the harmonic mean of precision and recall, providing a balanced This metric is particularly useful when there is an imbalance between benign and malicious URLs in the dataset.

2. Performance Comparison with Other Models

To evaluate the effectiveness of this URL malware detection model, we compare it with other traditional machine learning models used for similar tasks:

Conclusion :

The "Machine Learning for Web Vulnerability Detection" project successfully demonstrates the application of machine learning for real-time URL classification to detect potentially malicious or benign websites. By extracting key lexical and structural features from URLs, the trained model effectively identifies patterns commonly associated with web-based threats.

Key Achievements:

Automated Feature Extraction – Analyzes URL properties such as length, special characters, and domain structure to detect suspicious websites.

Machine Learning-Based Classification – Utilizes a pre-trained model for fast and accurate URL threat detection.

Flask API for Real-Time Detection – Deploys as a lightweight, interactive web application for easy use and integration.

Logging & Robust Error Handling – Ensures stability and maintainability of the system by tracking API requests and handling exceptions.

Future Scope & Enhancements:

- Implement deep learning models (e.g., CNNs, RNNs, Transformers) for more robust threat detection
- Enhance with real-time integration of external security databases (Google Safe Browsing, VirusTotal, PhishTank)
- Develop a browser extension or mobile app for user-friendly security checks
- Deploy as a scalable cloud-based microservice for enterprise-level cybersecurity applications

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs". In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).

[2] Sahoo, D., Liu, C., & Hoi, S. C. H. (2019). "Malicious URL Detection using Machine Learning: A Survey". arXiv preprint arXiv:1701.07179.
[3] Google Safe Browsing API – https://safebrowsing.google.com/
[4] VirusTotal Threat Intelligence API – https://www.virustotal.com/gui/home/url
[5] PhishTank: Open Phishing Database – https://www.phishtank.com/
[6] Joblib Documentation – https://joblib.readthedocs.io/en/latest/
[7] Flask Documentation – https://flask.palletsprojects.com/en/2.0.x/