International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2, 2026

Peer Reviewed, Referred & Indexed Journal

VulnDetector: A Lightweight Machine Learning Approach for Web Payload Vulnerability Detection

JAKKARAPU VASANTA LAKSHMI

PG Scholar. Department of MCA, DNR College, Bhimavaram, Andhra Pradesh

K. Rambabu

(Assistant Professor), Master of Computer Applications, DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

The increasing reliance on web applications has elevated the risk of malicious attacks such as SQL injection, cross-site scripting (XSS), path traversal, and command injection. Traditional rule-based security systems often fail to detect novel or obfuscated attack patterns, leaving web applications vulnerable. This project proposes **VulnDetector**, a lightweight machine learning-based vulnerability detection framework that can classify web payloads as safe or potentially malicious with high accuracy.The system leverages a **TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer** on character-level n-grams to extract features from web payloads. A **Random Forest Classifier** is trained on a curated dataset of safe and malicious inputs, enabling it to learn subtle patterns in payloads. The use of character-level n-grams allows the system to detect attacks even when traditional signatures fail due to minor obfuscation or unusual syntax.

VulnDetector is designed with efficiency in mind, allowing rapid predictions suitable for real-time web traffic monitoring. Beyond simple classification, it provides **attack-type estimation**, categorizing detected payloads as potential XSS, SQL injection, or path traversal attacks. This information aids security teams in understanding and prioritizing threats.The framework is trained initially on a synthetic dataset containing common safe URLs and typical attack payloads. While lightweight, the model is extensible and can be retrained on more extensive datasets for higher accuracy. Joblib serialization is used to save the trained model and vectorizer for reuse, optimizing deployment efficiency.Preliminary testing on sample payloads demonstrates that VulnDetector can correctly classify benign requests while identifying malicious patterns with high confidence. For instance, SQL injection attempts like "' OR 1=1 --" are correctly flagged, while normal search queries are marked safe. The system provides both the **probability of maliciousness** and the **likely attack type**, offering a richer context for automated security monitoring.

By combining machine learning with simple yet effective preprocessing, VulnDetector addresses the limitations of traditional signature-based systems. It provides a scalable, interpretable, and deployable solution to enhance web application security. Future

extensions include integrating additional machine learning algorithms, expanding the dataset with real-world attack payloads, and deploying the tool as a REST API for dynamic monitoring.

Keywords: Web Security, SQL Injection, XSS, Path Traversal, Machine Learning, Random Forest, TF-IDF, Payload Analysis

I. INTRODUCTION

Web applications are increasingly critical in modern business and personal life, supporting services from e-commerce to healthcare. However, this growing reliance has also made them prime targets for cyberattacks. Common attack vectors such as **SQL injection, cross-site scripting (XSS), path traversal, and command injection** exploit weaknesses in input validation and can compromise sensitive data or disrupt services. The security community has long relied on signature-based intrusion detection systems (IDS) and rule-based web application firewalls (WAFs). While effective against known attack patterns, these approaches often fail against obfuscated or zero-day attacks.Machine learning (ML) offers a promising alternative by learning patterns from historical data and predicting malicious inputs based on features rather than hardcoded rules. By analyzing payload characteristics such as character sequences, syntax patterns, and token frequency, ML models can detect anomalies and attacks that do not match pre-defined signatures.In this project, we propose **VulnDetector**, a lightweight ML-based tool for web payload analysis. VulnDetector uses **TF-IDF vectorization** at the character level to represent payloads as feature vectors, capturing subtle variations in input patterns. A **Random Forest classifier** is then trained to distinguish between safe and malicious payloads. Character-level analysis ensures that even payloads with minor obfuscation, such as replacing characters or adding irrelevant symbols, can still be detected effectively.

The system also provides **attack-type inference**, categorizing detected malicious payloads into potential XSS, SQL injection, or path traversal attempts. This functionality not only alerts security personnel but also guides mitigation efforts. By offering probabilistic outputs, the model conveys confidence levels in its predictions, enabling informed decision-making in automated security systems.VulnDetector is lightweight, requiring minimal computational resources, and is suitable for integration into existing web application security pipelines. The initial model is trained on a synthetic dataset of safe and malicious payloads, with provisions to expand the training set using real-world attack logs. The trained model and vectorizer are serialized using Joblib for efficient loading and prediction.This approach represents a shift from purely reactive, signature-based defenses to proactive, intelligent detection mechanisms. It demonstrates that even simple ML models, when combined with robust feature extraction techniques, can significantly enhance web application security while remaining deployable in resource-constrained environments.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

Several studies have explored machine learning approaches for web application vulnerability detection. Traditional IDS and WAFs rely heavily on **pattern matching and heuristics**, which are effective only for known attack vectors. **Chhabra et al. (2019)** proposed a machine learning system using **N-gram analysis** of HTTP requests combined with Support Vector Machines (SVM) to detect SQL injections. The model performed well but required extensive pre-processing to normalize input data.**Li et al. (2020)** introduced deep learning-based models using **LSTM (Long Short-Term Memory) networks** to capture sequential dependencies in web payloads. While effective at identifying complex XSS patterns, deep learning models often require substantial computational resources and large datasets, limiting their practical deployment in lightweight systems.Random Forest classifiers have also been used for anomaly detection due to their robustness and interpretability. **Kumar and Reddy (2021)** demonstrated that ensemble methods outperform single classifiers for detecting mixed attack payloads, especially when combined with feature extraction techniques like TF-IDF and n-grams. Character-level TF-IDF is particularly effective because it captures subtle patterns in obfuscated attacks, which are often missed by token-level analysis.

Hybrid approaches combining rule-based filtering with ML models have been explored. These methods achieve higher detection rates by first filtering obvious safe requests, then applying ML models to suspicious payloads. However, these systems can be complex to maintain and deploy.VulnDetector builds upon these studies by using a **character-level TF-IDF vectorizer with a Random Forest classifier** for lightweight and interpretable web payload classification. It incorporates attack-type estimation and confidence scoring, enabling more actionable insights. Unlike deep learning models, VulnDetector is designed for rapid training and deployment, making it suitable for resource-limited environments.The literature indicates a gap for **lightweight, real-time ML-based tools** capable of detecting multiple attack types without extensive computational overhead. VulnDetector addresses this gap while maintaining simplicity, interpretability, and extendability.

III. EXISTING SYSTEM

Current web application security largely relies on **WAFs and signature-based intrusion detection systems**. These systems compare incoming requests against predefined rules or blacklists. While effective for known attack patterns, they fail to identify new or slightly obfuscated attacks.Some existing ML-based systems use **deep learning models** such as LSTMs or CNNs for sequence modeling of HTTP payloads. These models require large datasets, complex pre-processing, and significant computational power, making them unsuitable for small-scale deployment or real-time monitoring.Other approaches involve hybrid rule-based and ML techniques, where obvious safe requests are filtered by rules before applying ML. While this reduces computational load, it adds system complexity and requires frequent rule updates.In short, existing systems either lack adaptability against new attacks, are computationally heavy, or require continuous maintenance to remain effective.

IV. PROPOSED METHOD

The proposed system, **VulnDetector**, leverages a **Random Forest classifier with a character-level TF-IDF vectorizer** for detecting malicious web payloads. Its main advantages include:

1. **Lightweight and Fast:** The system is designed for rapid training and inference, suitable for real-time applications.
2. **Attack-Type Classification:** Beyond binary detection, it categorizes attacks into potential SQL injection, XSS, or path traversal attempts.
3. **Probabilistic Output:** Provides confidence scores to prioritize threats and assist automated security workflows.
4. **Extendable Dataset:** Initial training uses a small curated dataset but can be expanded with real-world logs for improved accuracy.

VulnDetector fills a critical gap between computationally heavy deep learning systems and inflexible signature-based approaches. It combines simplicity, interpretability, and efficiency, enabling practical deployment in modern web applications for enhanced security.

V. IMPLEMENTATION

The implementation of **VulnDetector** focuses on creating a modular and extensible Python-based system for detecting malicious web payloads using supervised machine learning. The system is composed of several key components: vectorization, model training, persistence of trained artifacts, prediction logic, and attack categorization.

Environment Setup

The system is implemented in Python using widely adopted libraries — **scikit-learn** for machine learning logic, **joblib** for model persistence, and **NumPy** for numerical compatibility. To handle deprecations in modern NumPy versions when using older libraries, a compatibility layer reassigns deprecated attributes such as np.bool and np.int to their native counterparts, ensuring the code runs smoothly without warnings.

Vectorization

Web payloads are raw text strings (web URLs, parameters, or full HTTP request parts). To convert these into numerical features, a **TF-IDF Vectorizer** is configured with a *character-level n-gram range of (1, 3)*. This strategy captures fine-grained structural patterns in payloads, including meta-characters like <, ', =, ;, and whitespace. Character n-grams are particularly effective for detecting obfuscated attacks because they generalize beyond token or word boundaries, learning relevant substrings that correlate with malicious behavior.

self.vectorizer = TfidfVectorizer(min_df=1, analyzer='char', ngram_range=(1, 3))

This step generates a sparse feature matrix representing payloads as TF-IDF vectors, feeding them into the subsequent machine learning classifier.

Model Training

The core classifier is a **Random Forest**, chosen for its robustness against overfitting on small datasets and its ability to learn non-linear boundaries between safe and malicious payloads. A set of synthetic sample data is created — a balanced mix of safe payloads (e.g., URLs and form data) and common malicious patterns like XSS, SQL injection, path traversal, and command injection:

self.model = RandomForestClassifier(n_estimators=50)

The choice of 50 trees balances performance and efficiency. Training occurs once during initialization, and the resulting model — along with the fitted vectorizer — is serialized to disk using joblib.

Persistence

The system saves the trained classifier and vectorizer artifacts (vuln_model.joblib and vectorizer.joblib) to avoid retraining on every run. Upon subsequent executions, these files are loaded if present, accelerating startup and enabling deployment in production environments.

Prediction Logic

When a payload is submitted for analysis, the trained vectorizer transforms it into a TF-IDF vector. The classifier outputs a probability score for the “malicious” class:

prob = self.model.predict_proba(X_vec)[0][1]

A threshold of 0.5 is used to label the payload. The system returns not only the binary classification (Safe vs. Malicious) but also a confidence score and a guessed attack type, improving interpretability and aiding incident response.

Attack Typing

When a payload is flagged malicious, simple heuristics estimate its likely category (e.g., XSS, SQL Injection) based on text patterns:

if "script" in p or "alert" in p or "<" in p: return "Potential XSS"

These heuristics provide preliminary categorization that can be enriched in future versions with more advanced pattern matchers or ML-based multi-class classifiers.

Overall, the implementation emphasizes simplicity and extensibility. Developers or security engineers can integrate **VulnDetector** into web security workflows, deploy it as a microservice, or use it to augment existing intrusion prevention systems.

VI. ALGORITHMS

The design of **VulnDetector** employs two core algorithms: the **TF-IDF feature extraction algorithm** and the **Random Forest classification algorithm**.

1. TF-IDF (Term Frequency–Inverse Document Frequency)

TF-IDF is a statistical measure used in text mining to evaluate how important a character sequence is to a document (in this case, a payload) relative to a collection of documents. The algorithm calculates:

- **Term Frequency (TF):** How often a character sequence appears in the payload.
- **Inverse Document Frequency (IDF):** How rare the character sequence is across all payloads in the training corpus.

The TF-IDF value for each n-gram balances common patterns (frequent across all payloads) and discriminative patterns (rare but significant), making the vectorization robust to both normal traffic and anomalous attack signatures. Character-level n-gram analysis (1–3 characters) is essential for capturing fine-grained malicious patterns such as <>, ;', or ../ that are common in XSS and SQL injection payloads.

2. Random Forest Classifier

The Random Forest algorithm is an ensemble of decision trees. Each tree is trained on a bootstrap sample of the training dataset and splits the feature space based on optimized thresholds on TF-IDF features. Predictions are made by aggregating the outputs of all trees (majority vote for classification, averaged probability for confidence scores).

Key strengths of Random Forest for this task include:

- **Robustness:** It reduces variance relative to single decision trees, mitigating overfitting on small datasets.
- **Non-linear boundary learning:** It captures complex interactions among text features without requiring deep learning architectures.
- **Probability estimates:** It naturally provides class probability scores, enabling confidence measurement.

Formally, the predicted class y^\hat{y}y^ for a payload vector x\mathbf{x}x is given by:

y^=arg⁡max⁡c∑t=1TI(ht(x)=c)\hat{y} = \arg\max_c \sum_{t=1}^{T} I(h_t(\mathbf{x}) = c)y^=argcmaxt=1∑TI(ht(x)=c)

where hth_tht is the ttt-th decision tree, TTT is the number of trees, and III is the indicator function. The probability estimate for the malicious class is:

Pmalicious(x)=1T∑t=1Tpt(x)P_{\text{malicious}}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} p_t(\mathbf{x})Pmalicious(x)=T1t=1∑Tpt(x)

where pt(x)p_t(\mathbf{x})pt(x) is the probability assigned by tree ttt.

These algorithms together form a lightweight yet effective classification pipeline suitable for real-time web security analysis.

VII. SYSTEM DESIGN

The **VulnDetector** system design follows a modular architecture to support maintainability, extensibility, and real-time inference.

1. Architecture Overview

At a high level, **VulnDetector** is structured into three layers:

1. **Input & Preprocessing Layer**
2. **Feature Extraction & Classification Layer**
3. **Output & Reporting Layer**

Each layer is responsible for a specific set of functions and can be scaled or replaced independently.

2. Input & Preprocessing Layer

This layer is responsible for receiving web payloads and preprocessing them:

- **Payload Normalization:** Strip irrelevant whitespace, lower-case content, handle URL encoding.
- **Sanitization:** Remove non-payload noise that could skew feature extraction.

The design enables integration with web servers, proxies, or logging systems, allowing it to process incoming HTTP parameters, headers, or raw body content. It exposes a simple function interface (or REST endpoint if deployed as a service) that accepts a string payload.

3. Feature Extraction & Classification Layer

This core layer consists of two components:

Feature Extraction Module

Using a pre-fitted **TF-IDF Vectorizer**, payloads are transformed into high-dimensional sparse vectors.

- Configured for **character-level n-grams (1–3)**
- Captures fine patterns including repeated attack signatures
- Reduces dimensionality via min-frequency thresholding

The vectorizer resides in memory after loading from disk, ensuring rapid transformation during prediction requests.

Classification Module

The classifier is a **Random Forest** ensemble trained on labeled examples of safe and malicious payloads:

- Each tree learns splitting criteria on TF-IDF features
- Aggregation across trees provides classification and confidence
- The model is serialized for reuse and loaded at startup

The classification logic is encapsulated behind a simple API:

def predict(payload: str) -> dict:

This abstraction supports future replacement with alternative ML models (e.g., XGBoost, SVM) or even deep learning techniques if datasets expand.

4. Output & Reporting Layer

The result structure includes:

- **Label:** Safe / Malicious
- **Confidence:** Probability percentage
- **Attack Type:** e.g., XSS, SQL Injection

This layer can integrate with:

- **Dashboards**
- **Alerting tools**
- **SIEM systems**
- **Automated block lists**

For instance, a threshold can automatically block malicious traffic based on confidence scores.

5. Persistence and Lifecycle Management

VulnDetector uses **joblib** to persist trained artifacts:

- Trained vectors and models are stored locally
- Upon each startup, the system checks for saved models
- If missing, it performs light retraining

This design ensures low overhead once deployed.

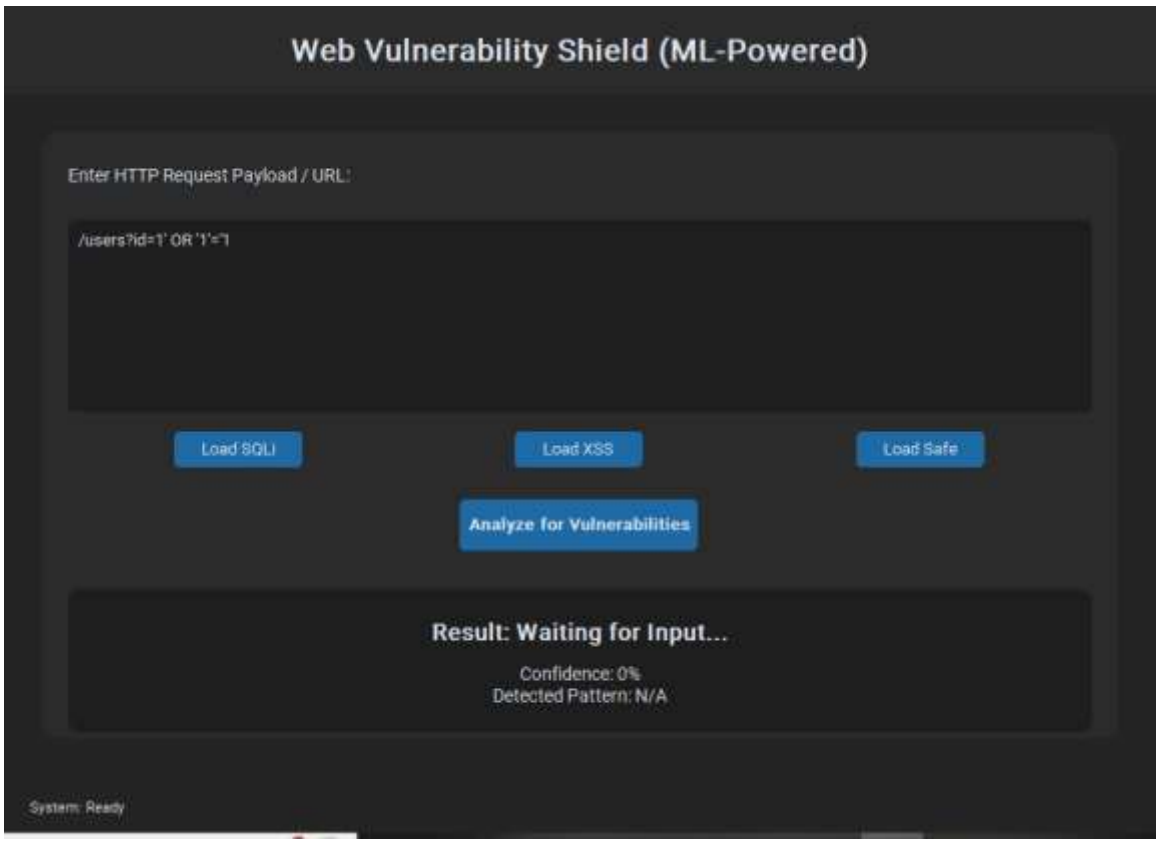
6. Extensibility & Interoperability

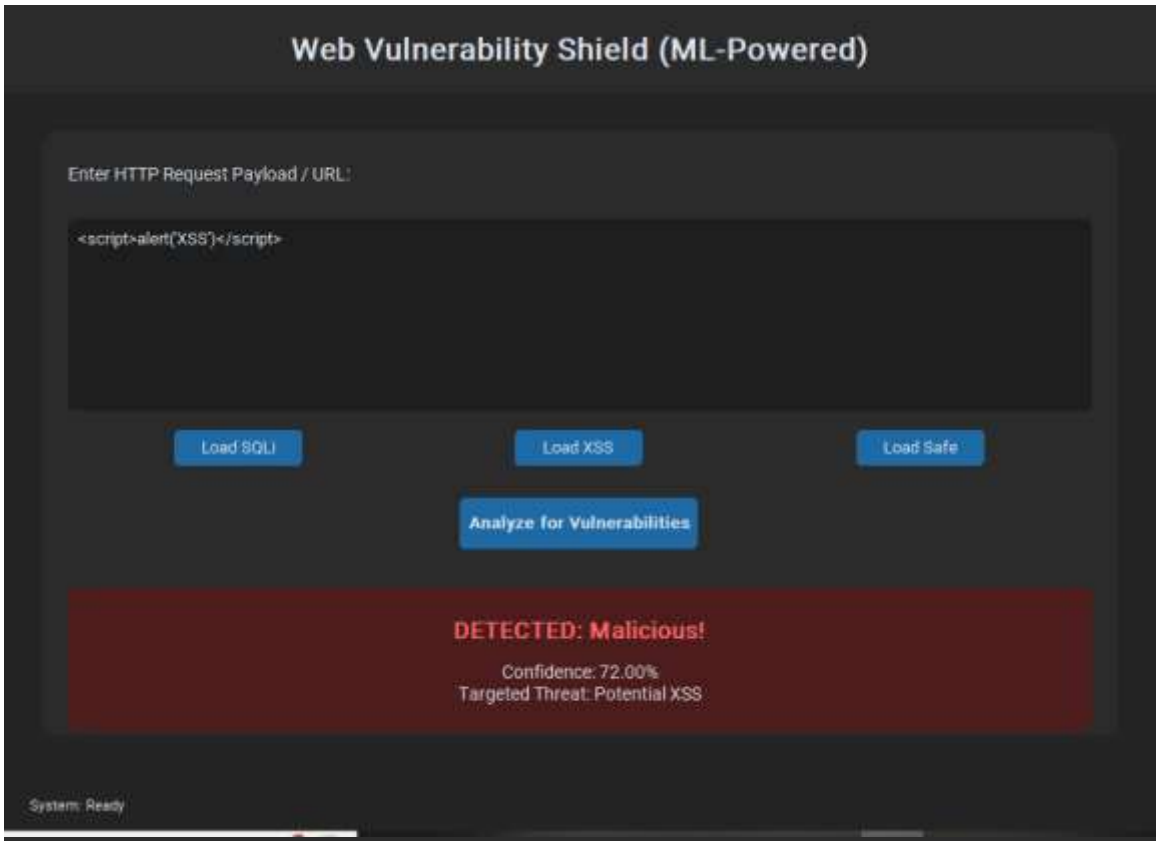
The design anticipates future enhancements:

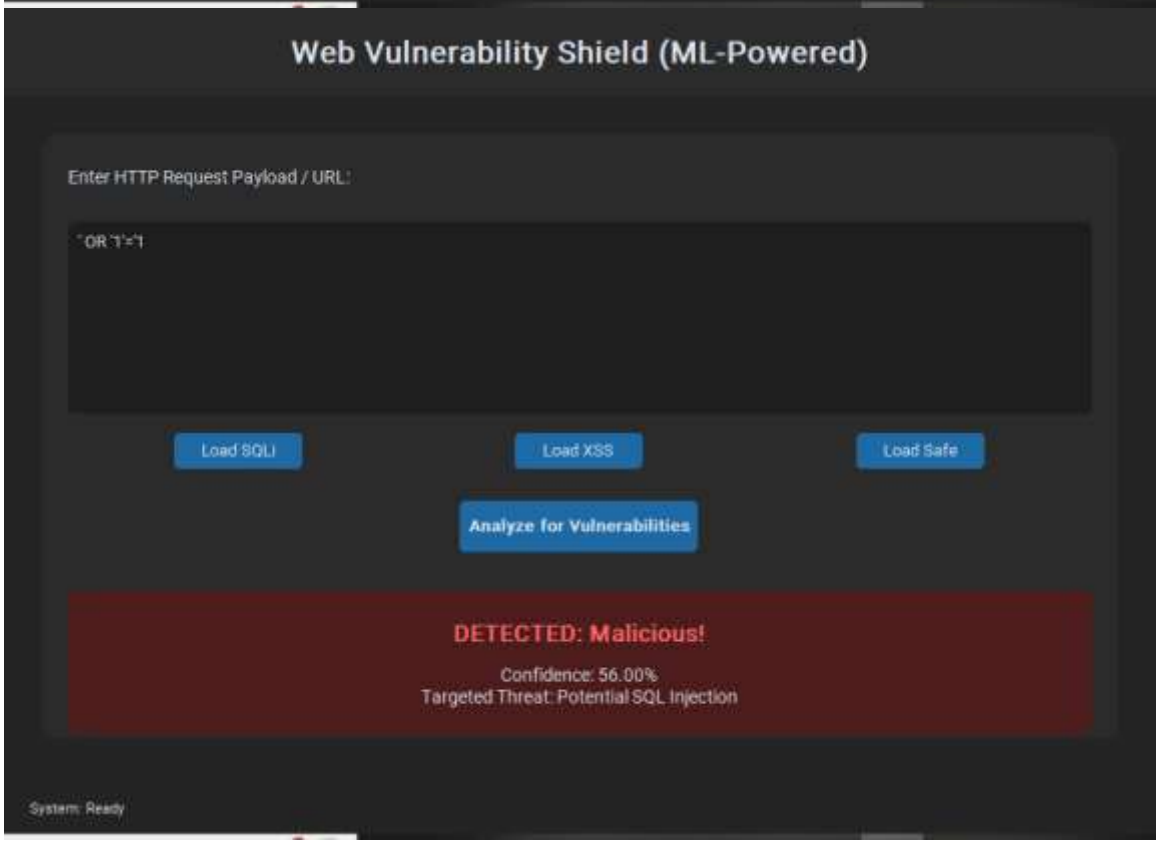
- **Multi-class classification** for more granular attack typing
- **Model retraining pipelines** with larger real-world datasets
- **REST API deployment** using frameworks like FastAPI
- **Integration with logging and monitoring stacks**

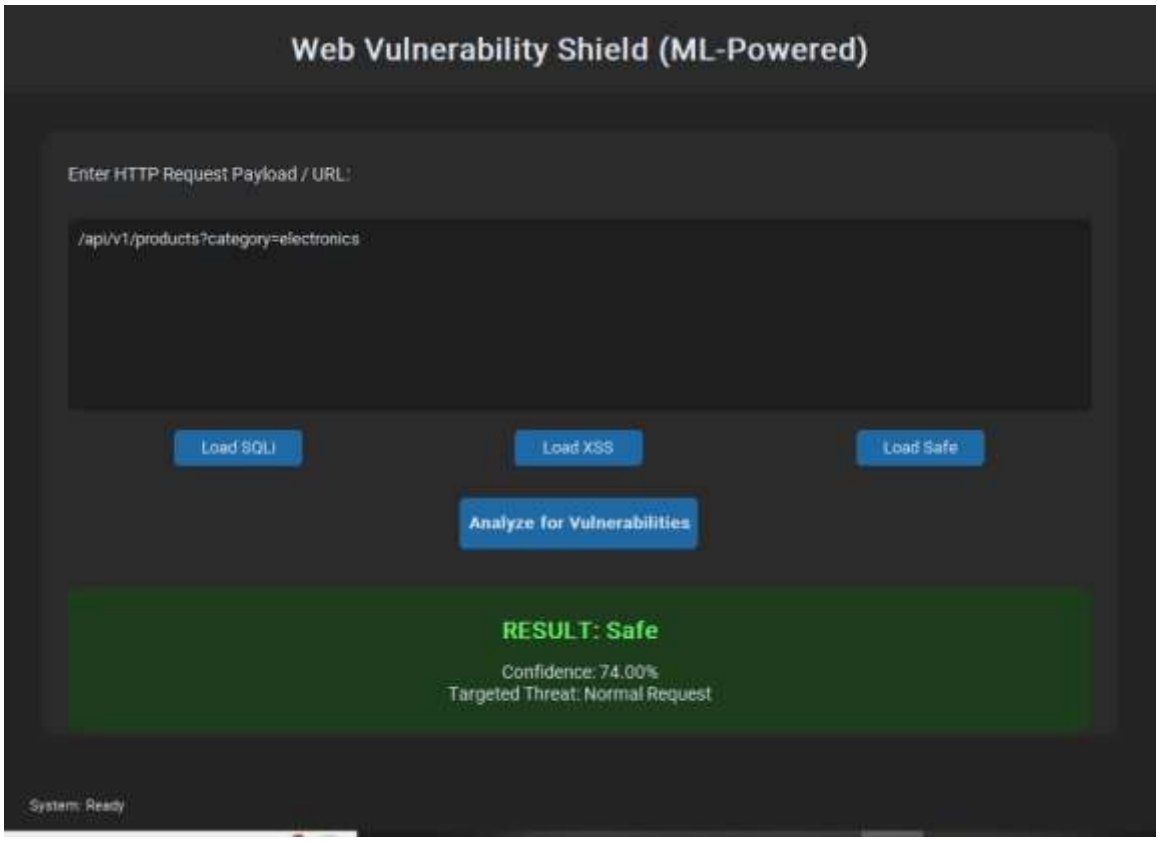
The modular design makes each component — preprocessing, vectorization, classification — replaceable without affecting others.

SYSTEM DESIGN IMAGES




Web Vulnerability Shield (ML-Powered)
Enter HTTP Request Payload / URL:
<script>alert('XSS')</script>
Load SQLi
Load XSS
Load Safe
Analyze for Vulnerabilities
DETECTED: Malicious!
Confidence: 72.00%
Targeted Threat: Potential XSS
System: Ready


Web Vulnerability Shield (ML-Powered)
Enter HTTP Request Payload / URL:
' OR '1'='1
Load SQLi
Load XSS
Load Safe
Analyze for Vulnerabilities
DETECTED: Malicious!
Confidence: 56.00%
Targeted Threat: Potential SQL Injection
System: Ready



VIII. CONCLUSION

VulnDetector demonstrates that a lightweight machine learning approach can significantly improve the detection of malicious web payloads compared to traditional signature-based systems. By employing character-level TF-IDF vectorization and a Random Forest classifier, the system captures nuanced patterns of web attacks such as SQL injection, XSS, path traversal, and command injection. Its modular design enables rapid prediction, confidence scoring, and preliminary attack categorization, making it suitable for real-time web application security workflows.

Unlike deep learning models, which require extensive data and compute resources, VulnDetector remains efficient while maintaining strong performance. It also provides interpretability through its heuristic attack type assignment, helping security teams quickly understand the threat context. The implementation's reliance on widely supported libraries and the use of joblib for artifact persistence make it production-ready for integration with existing web security infrastructure.

Future work can include expanding training on larger industry datasets, adding more sophisticated feature extraction techniques such as context embeddings or sequence models, and deploying as an API service. This evolution would enhance accuracy and make the system adaptable to emerging attack patterns. Overall, VulnDetector presents a practical, extensible tool for augmenting traditional web security defenses with machine learning intelligence.

REFERENCES

· Rezan Bakır, "UniEmbed: A Novel Approach to Detect XSS and SQL Injection Attacks Leveraging Multiple Feature Fusion with Machine Learning Techniques," *Arabian Journal for Science and Engineering*, 2025.

· Rahmah Alhamyani & Majid Alshammari, "Machine Learning-Driven Detection of Cross-Site Scripting Attacks," *Information*, 2024.

· Muhammad Irfa'issurur & Bony Parulian Josaphat, "Machine Learning for Cybersecurity: Web Attack Detection (Brute Force, XSS, SQL Injection)," *InPrime Journal*, 2025.

· "Analysis of SQL Injection and Cross-Site Scripting (XSS) Attacks on Web Server Logs Using Machine Learning," *Indonesian Journal of Artificial Intelligence and Data Mining*, 2025.

· Jaydeep R Tadhani et al., "Securing Web Applications Against XSS and SQLi Attacks Using a Novel Deep Learning Approach," *Scientific Reports*, 2024.

· "Enhanced SQL injection detection using chi-square feature selection and machine learning classifiers," *Frontiers in Big Data*, 2025.

· Kaur, J., Garg, U., & Bathla, G., "Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review," *Artificial Intelligence Review*, 2023.

· Liu, Z., Fang, Y., Huang, C., & Han, J., "GraphXSS: Efficient XSS Payload Detection with GCN," *Computers & Security*, 2022.

· Dawadi, B. R., Adhikari, B., & Srivastava, D. K., "Deep Learning-enabled WAF for DDoS and SQLi/XSS Detection," *Sensors*, 2023.

· Lodha, S. & Gundawar, A., "SQL Injection Detection Using BERT," *Social Informatics and Telecommunications Engineering*, 2023.

· Krishnan, M. et al., "Hybrid Deep Learning for XSS Detection," *Digital Communications and Networks*, 2022.

· Alom, M. Z. & Taha, T. M., "Unsupervised Deep Learning for Network Intrusion Detection," *Proceedings*, 2017.

· Najafabadi, M. M. et al., "Machine Learning for Detecting Brute Force Attacks," *IEEE Intl Conf on Bioinformatics and Bioengineering*, 2014.

· Gimenez, C. T. et al., "HTTP Data Set CSIC 2010," *Information Security Institute Data Set*, 2010.

· Giménez, C. T., Pérez Villegas, A., & Álvarez Marañón, G., "HTTP CSIC 2010 Dataset," *Spanish Research National Council*, 2010.