Artificial Intelligence-Driven Supervised Classification Algorithm for Website Vulnerability Detection Using MITRE NVD CVE Scores

Amolika Roy * , Paras Jiskar , Om Mishra

Posted Date: 6 February 2026

doi: 10.20944/preprints202602.0475.v1

Keywords: artificial intelligence; supervised classification; website vulnerability detection; National Vulnerability Database (NVD); Common Vulnerability Scoring System (CVSS); random forest; CodeBERT; chrome extension; cybersecurity; machine learning; AWS SageMaker

 doi:10.20944/preprints202602.0475.v1

Article

Artificial Intelligence-Driven Supervised Classification Algorithm for Website Vulnerability Detection Using MITRE NVD CVE Scores

Amolika Roy [1,*], Paras Jiskar [1] and Om Mishra [2]

[1] Pick & Roll II Inc. dba Workstreet

[2] Rajiv Gandhi National Cyber Law Centre, National Law Institute University, India

* Correspondence: amolikaroy6@gmail.com; Tel.: +918839168881

Abstract

As cyber threats continue to evolve, traditional security measures often fail to detect emerging vulnerabilities in real-time, particularly for small and medium-sized enterprises with limited resources. This study develops an AI-driven supervised classification algorithm for website vulnerability detection that integrates insights from the National Vulnerability Database (NVD) and Common Vulnerability Scoring System (CVSS) scores. A dataset of 40,000 vulnerability entries was curated using reconnaissance tools including Nmap and Nessus, with HTML code snippets labeled according to severity levels. The methodology employed CodeBERT transformer models for converting raw HTML into numerical embeddings, followed by Random Forest classification trained on AWS SageMaker. A Chrome browser extension was developed to extract live webpage content and communicate with a Flask-based API hosted on Amazon EC2 for real-time inference. Following optimization through TF-IDF vectorization and hyperparameter tuning, the model achieved 66.3% accuracy with ROC-AUC values ranging from 0.60 to 0.70 across severity classes. The system successfully classifies websites into Low, Medium, or High-risk categories in real-time. This research demonstrates that supervised machine learning offers a practical, cost-effective, and auditable alternative to computationally intensive deep learning approaches, providing accessible vulnerability detection while maintaining compliance with emerging AI governance frameworks such as ISO 42001 and the NIST AI Risk Management Framework.

Keywords: artificial intelligence; supervised classification; website vulnerability detection; National Vulnerability Database (NVD); Common Vulnerability Scoring System (CVSS); random forest; CodeBERT; chrome extension; cybersecurity; machine learning; AWS SageMaker

1. Introduction

In today's increasingly digital world, organizations rely heavily on web applications and networks for essential business functions. However, with this dependence comes the risk of cyber threats that evolve continuously, with attackers constantly innovating new ways to exploit vulnerabilities in websites, networks, and applications. This ever-evolving cyber threat landscape poses a serious challenge for organizations that strive to keep their systems secure while ensuring uninterrupted services. Traditional security measures, though valuable, often fall short in addressing the volume and complexity of emerging threats. These conventional approaches can struggle to detect new types of attacks and respond in real-time, making organizations vulnerable to sophisticated cyber-attacks [1].

The cybersecurity landscape has witnessed an unprecedented surge in vulnerability disclosures, with the National Vulnerability Database (NVD) recording over 25,000 new Common Vulnerabilities and Exposures (CVE) entries annually in recent years [2]. This exponential growth in known vulnerabilities creates a significant challenge for security professionals who must prioritize

 doi:10.20944/preprints202602.0475.v1

remediation efforts while operating under resource constraints. Small and medium-sized enterprises (SMEs) are particularly affected, as they often lack the specialized personnel and financial resources necessary to implement comprehensive vulnerability management programs [3].

To address these challenges, artificial intelligence (AI) presents a game-changing addition to cybersecurity defenses. By leveraging machine learning algorithms, AI has the power to automate threat detection, providing timely insights into vulnerabilities, and enabling organizations to take proactive measures to secure their digital assets [4]. This research focuses on developing an AI-driven supervised classification algorithm to identify and classify vulnerabilities in websites based on analysis of HTML content and structural patterns. By utilizing data from sources like the NIST National Vulnerability Database (NVD) and MITRE's Common Vulnerabilities and Exposures (CVE) scores, the proposed model helps organizations prioritize risks effectively and implement preventive measures based on the level of threat detected.

The primary objective of this research is to create a tool that uses an AI-based supervised machine learning algorithm to detect potential risks in websites. The algorithm is designed to classify the level of risk (Critical, High, Medium, Low) based on NVD MITRE CVSS database associations with specific parameters and assign risk scores. These risk scores provide insights into how vulnerable a website might be, allowing organizations to take pre-emptive action to prevent cyber incidents [5].

1.1. Limitations of Deep Learning Approaches

Recent advances in artificial intelligence, particularly in deep learning, have enabled powerful algorithms and models capable of processing vast amounts of data and identifying complex patterns. Many of these advancements leverage techniques like Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), and Graph Neural Networks (GNNs) [6]. Despite their impressive capabilities, deep learning models come with significant disadvantages that limit their practical applicability for many organizations.

Developing and training neural networks can be an arduous process that takes months or longer, especially for complex models requiring extensive fine-tuning [7]. Building a deep learning model from scratch involves selecting the right architecture, tuning hyperparameters, and conducting numerous rounds of testing. This prolonged development timeline can delay deployment and hinder rapid responses to emerging security threats—a significant drawback in the fast-paced world of cybersecurity.

Neural networks, particularly deep networks, are computationally intensive, requiring substantial hardware resources such as GPUs or TPUs to function efficiently [8]. These resources come at a high cost, making deep learning solutions financially demanding for many organizations. High power consumption, expensive cloud services, and hardware maintenance add to the overall cost, potentially rendering such solutions unfeasible for budget-conscious companies.

Deep learning models, particularly those involving complex neural networks, are often considered "black boxes" because their internal workings are difficult to interpret [9]. Auditing these models to ensure compliance with regulatory requirements can be challenging, as auditors struggle to understand how decisions are made. For sectors with strict regulatory standards, such as finance or healthcare, the inability to fully audit and explain model behavior can be a significant barrier to adoption.

1.2. Research Contributions

To address these limitations, this research proposes an alternative approach—a machine learning-based AI application that employs a supervised classification algorithm. This algorithm aims to provide a real-time, cost-effective solution for threat detection in cybersecurity. By focusing on supervised classification, this approach avoids the extensive computational requirements and long training times of deep learning, offering a more practical and auditable solution that aligns with regulatory requirements and organizational constraints.

 doi:10.20944/preprints202602.0475.v1

The key contributions of this research include:

(1) Development of a comprehensive dataset of 40,000 vulnerability entries curated from the NVD and reconnaissance activities using Nmap and Nessus.
(2) Implementation of a hybrid feature extraction approach combining CodeBERT transformer embeddings with TF-IDF vectorization for HTML code analysis.
(3) Design and deployment of a Random Forest classification model on AWS SageMaker for severity prediction.
(4) Creation of a Chrome browser extension that enables real-time vulnerability assessment during web browsing.
(5) Demonstration of a practical, auditable alternative to deep learning approaches that maintains compliance with emerging AI governance frameworks.

1.3. Paper Organization

The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of related work in machine learning-based vulnerability detection. Section 3 describes the materials and methods, including dataset creation, model architecture, and deployment infrastructure. Section 4 presents the experimental results and performance evaluation. Section 5 discusses the findings and their implications. Section 6 concludes the paper with suggestions for future work.

2. Related Work

The application of machine learning techniques to cybersecurity vulnerability detection has garnered significant research attention in recent years. This section reviews the existing literature on AI-based vulnerability detection methods, categorizing approaches by their target vulnerability types and algorithmic foundations.

2.1. Machine Learning for Web Vulnerability Detection

Calzavara et al. [10] presented "Mitch," a machine learning-based tool developed for black-box detection of Cross-Site Request Forgery (CSRF) vulnerabilities in web applications. Mitch leverages supervised learning to identify security-sensitive HTTP requests, distinguishing them from benign ones and significantly reducing false positives and negatives. Tested on top websites and production software, Mitch successfully uncovered 35 new CSRF vulnerabilities, proving its effectiveness and practical applicability. This tool demonstrates how machine learning can automate and enhance vulnerability detection by understanding application behavior, offering a robust solution for securing modern, diverse web applications.

Singh, Vijayalakshmi, and Raj [11] presented a technical approach to identifying and categorizing web-based malicious attacks—namely, phishing, malware, and defacement—using machine learning, specifically the Random Forest algorithm. The authors discuss how web applications are inherently vulnerable due to unintended weaknesses introduced during development, making them susceptible to these attacks. Traditional detection techniques, such as blacklisting URLs or heuristic-based detection, have proven limited, suffering from high false-positive rates and failing to capture novel or obfuscated threats.

The Random Forest algorithm was selected for their model due to its ability to manage high-dimensional data and classify intricate patterns. Each tree within the ensemble independently evaluates an input URL's features, with the aggregate result—based on majority voting—determining whether a URL is deemed safe or harmful. Using a large Kaggle dataset of 651,191 URLs covering legitimate, phishing, malware, and defacement sites, the model achieved an accuracy of 96.6%, demonstrating reliable classification of web threats.

 doi:10.20944/preprints202602.0475.v1

2.2. Cross-Site Scripting Detection

Alhamyani and Alshammari [12] investigated the use of machine learning for detecting Cross-Site Scripting (XSS) attacks, a common and dangerous form of web application vulnerability. Traditional detection methods often fail due to high false-positive and false-negative rates and the evolving tactics of attackers. The paper proposes an advanced ML-based approach, comparing several ML models to identify the best solution for XSS detection.

The study evaluates a range of models, including Random Forest (RF), Logistic Regression (LR), Support Vector Machines (SVM), Extreme Gradient Boosting (XGBoost), and deep learning methods like Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN), as well as ensemble learning methods. Feature selection techniques, such as Information Gain (IG) and Analysis of Variance (ANOVA), were applied to identify the most informative features for XSS detection. RF achieved a high accuracy of 99.78%, while ensemble models like RF combined with Gradient Boosting or Multi-Layer Perceptron also performed exceptionally well, reaching accuracy rates over 99.6%.

2.3. Phishing Website Detection

Tang and Mahmoud [13] focused on combating phishing attacks by using machine learning techniques to detect and block phishing websites, especially in real-time settings. The paper reviews and categorizes various detection methods: list-based, heuristic, and machine learning approaches. List-based methods involve whitelists and blacklists of URLs, allowing rapid detection but failing to identify new, unlisted phishing sites. Heuristic strategies leverage rule-based detection, analyzing URL patterns or content elements to distinguish legitimate from phishing websites.

Machine learning has shown promise in addressing these challenges, with algorithms such as Random Forests, Support Vector Machines, and neural networks offering higher accuracy by learning patterns from large datasets. Deep learning techniques, especially when combined with natural language processing, further enhance phishing detection by analyzing character-level features in URLs. The study concludes that while significant advancements have been made, phishing detection remains a dynamic field with room for optimization to keep pace with evolving phishing tactics.

2.4. Comprehensive Vulnerability Detection Methodologies

Bennouk et al. [14] investigated methods for detecting cyber vulnerabilities, particularly within interconnected IT and OT systems. The study categorizes approaches into four main types: similarity-based, graph-based, feature modeling (FM)-based, and AI-based. Similarity-based methods rely on algorithms like regular expressions, Levenshtein distance, and Jaro-Winkler to match product and vulnerability data. Graph-based methods leverage relationships between assets, using graph theory to model dependencies among elements like CVE, CPE (Common Platform Enumeration), and CWE (Common Weakness Enumeration).

AI-based methods, using techniques like Bidirectional Long Short-Term Memory (BLSTM) networks and recommendation systems, improve detection accuracy by identifying patterns from large datasets with minimal human intervention. The study's findings emphasize the effectiveness of combining different approaches to boost vulnerability detection accuracy.

2.5. Environment-Specific Vulnerability Prioritization

Reyes et al. [15] focused on creating a prioritization model to assess and rank vulnerabilities based on their network environment and impact potential. The model uses Shodan, a search engine for internet-connected devices, to collect data on vulnerabilities across IP addresses within an organization. Key algorithms utilized include a mathematical risk model combining variables such as total vulnerabilities, organizational risk, remediation time, and network features.

The primary findings reveal that Shodan's data variables are valuable for quantifying overexposure risks, and combining these with CVSS allows for a more effective prioritization system suited to organizational contexts. Data features extracted for prioritization include the number of

 doi:10.20944/preprints202602.0475.v1

vulnerabilities, the average organizational risk score, open port data, remediation time, and tags identifying network services.

2.6. Cyber Risk Prediction Using CVE Data

Negahdari Kia et al. [16] presented an automated, data-driven model for predicting cyber risks based solely on Common Vulnerabilities and Exposures (CVE) data, eliminating expert bias. The model maps each CVE description to relevant Wikipedia topics, forming clusters of cybersecurity risks. It then aggregates the frequency and severity of these risks over time, creating time-series data that supervised machine learning models use to forecast future occurrences and impacts. This approach adapts continuously, updating with new CVE and Wikipedia data to stay relevant in a constantly changing cyber landscape.

2.7. Healthcare Ecosystem Vulnerability Analysis

Silvestri et al. [17] presented a machine learning-based methodology for identifying and assessing cybersecurity threats and vulnerabilities in healthcare systems, focusing on connected medical devices and healthcare information infrastructure. The approach involves three steps: modeling the infrastructure, identifying threats using NLP, and assessing vulnerabilities based on reports from the CVE database. The methodology applies Natural Language Processing techniques, including BERT and XGBoost models, to automatically classify threats and vulnerabilities, assign severity levels, and provide actionable insights for risk management.

2.8. OAuth Vulnerability Detection

Munonye and Péter [18] presented a machine learning-based approach to detect vulnerabilities in the OAuth authentication and authorization flow. OAuth is widely used for secure communication, but flaws in the flow can be exploited by attackers. The study models OAuth as a supervised learning problem, developing and evaluating seven classification models. The methodology involves dataset generation, feature extraction, and model development, using techniques like Principal Component Analysis (PCA) to identify key parameters. The models achieved over 90% accuracy in vulnerability detection, with a 54% match against known vulnerabilities.

2.9. Vulnerability Exploitation Prediction

Hoque et al. [19] focused on predicting which vulnerabilities in software and hardware are most likely to be exploited in cyber-attacks. The paper highlights the class imbalance between exploited and non-exploited vulnerabilities in the National Vulnerability Database. Previous models struggled with overfitting due to this imbalance. The paper introduces a novel cost function to address the imbalance and uses a custom-trained word vector from a large text corpus to improve prediction accuracy. The resulting model achieved high performance metrics (accuracy: 0.92, precision: 0.89, recall: 0.98, F1-score: 0.94, AUC: 0.97).

2.10. Autonomous Penetration Testing

Chowdhary, Jha, and Zhao [20] proposed using Generative Adversarial Networks (GANs) for autonomous penetration testing of web applications protected by Web Application Firewalls (WAFs). The method introduces conditional sequence generation, where semantic tokenization helps create targeted attack payloads, like Cross-Site Scripting and SQL Injection, that can bypass WAF defenses. Testing on ModSecurity and AWS WAFs showed that the generated attacks were effective in bypassing security measures.

Ghanem and Chen [21] explored an AI-based vulnerability detection approach through a Reinforcement Learning-driven system for network penetration testing. Their system, named the Intelligent Automated Penetration Testing System (IAPTS), leverages RL to automate and optimize

 doi:10.20944/preprints202602.0475.v1

the process of penetration testing by modeling tasks as a partially observed Markov decision process. IAPTS integrates with existing PT frameworks, learning from data to replicate testing activities in future scenarios.

2.11. Improved Vulnerability Remediation

Jacobs et al. [22] focused on an AI-based vulnerability detection approach that leverages machine learning to improve vulnerability remediation strategies. Using data from various sources, including Kenna Security and FortiGuard Labs, the authors built a predictive model that prioritizes vulnerabilities likely to be exploited in real-world attacks instead of simply those with known published exploits. The model was constructed using gradient-boosted trees and achieved high accuracy (94.5%) in identifying high-risk vulnerabilities.

2.12. Research Gap Analysis

While the existing literature demonstrates significant progress in applying machine learning to specific vulnerability types, several gaps remain:

(1) Integration Gap: Most existing solutions focus on specific vulnerability types (XSS, CSRF, phishing) rather than providing comprehensive website risk assessment.
(2) Accessibility Gap: Many proposed solutions require significant computational resources or specialized expertise, making them inaccessible to SMEs.
(3) Real-time Deployment Gap: Few studies address the practical challenges of deploying ML-based vulnerability detection in real-time browsing scenarios.
(4) Auditability Gap: Deep learning approaches, while accurate, lack the interpretability required for regulatory compliance.
(5) Standardization Gap: Limited research connects ML-based detection directly with standardized vulnerability databases like NVD and CVSS scoring.

This research addresses these gaps by developing an integrated, accessible, real-time, and auditable vulnerability classification system that leverages standardized NVD/CVSS data.

2. Materials and Methods

This section describes the methodology employed in developing the AI-driven supervised classification algorithm for website vulnerability detection. The approach follows a structured, quantitative research methodology focused on creating a machine learning model capable of classifying web vulnerabilities in real-time.

2.1. System Architecture Overview

The proposed system architecture consists of four main components:

(1) Data Collection and Preprocessing Module: Responsible for gathering vulnerability data from the NVD and conducting reconnaissance using Nmap and Nessus.
(2) Feature Extraction Module: Utilizes CodeBERT transformer models to convert raw HTML code into numerical embeddings suitable for machine learning.
(3) Classification Module: Implements a Random Forest classifier trained on AWS SageMaker to predict vulnerability severity levels.
(4) User Interface Module: A Chrome browser extension that extracts live webpage content and communicates with the backend API for real-time inference.

 doi:10.20944/preprints202602.0475.v1

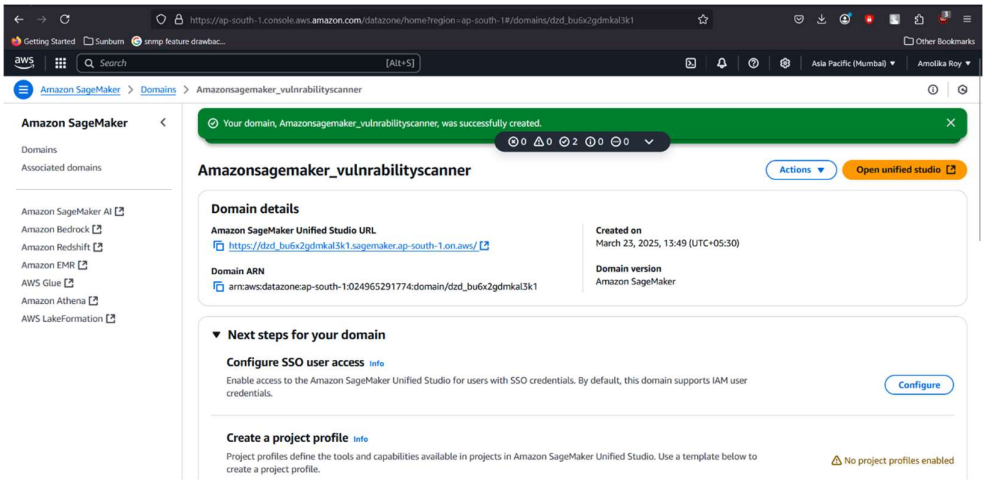

Figure 1. Screenshot of AWS Sage maker AI Dashboard.

2.2. Data Collection

In the initial stage of developing the supervised classification AI model, a comprehensive dataset was created using both active scanning tools and public vulnerability databases.

2.2.1. Reconnaissance and Enumeration

Nmap was used extensively on VMware-hosted virtual environments to scan target websites for open ports and services, allowing for enumeration and identification of potential vulnerabilities [23]. Various Nmap scripts were executed to gather detailed information about the services running on the target, aiming to detect misconfigurations and known exploits. This reconnaissance helped in understanding which parts of the web infrastructure were vulnerable.

Following this, Nessus was used to perform in-depth vulnerability assessments, providing reports with details about the vulnerabilities, their severity, and possible exploits [24]. Insights from these tools were cross-referenced with the NVD to extract features based on CVSS scores.

2.2.2. Dataset Compilation

A dataset of 40,000 records was compiled, including attributes such as: CVE ID (unique identifier for each vulnerability), Vulnerability Name (descriptive name), Description (detailed description), Affected Components (systems or software affected), Risk Levels (severity classification: Low, Medium, High), Mitigation Steps (recommended remediation actions), and Vulnerable Code Samples (HTML/JavaScript code snippets associated with vulnerabilities).

The dataset consists of CVEs published between 2020 and 2025, selected based on relevance and severity.

 doi:10.20944/preprints202602.0475.v1

jupyter cleaned_vulnerabilities.csv Last Checkpoint: 6 minutes ago

File Edit View Git Settings Help

Delimiter: ,

	CVE ID	Vulnerability Name	Description	Affected Components	Risk Level	Mitigation Steps	Vulnerable Code
1	CVE-2021-32253	Security Misconfiguration	s exposing vulnerabilities.	Web Server, Application	1	nfiguration management.	rue in production settings
2	CVE-2021-59060	Information Disclosure	ystem or application data.	Messages, Debug Mode	1	e unnecessary metadata.	print(debug_info)
3	CVE-2025-88702	Information Disclosure	ystem or application data.	Messages, Debug Mode	2	e unnecessary metadata.	print(debug_info)
4	CVE-2023-79681	te Code Execution (RCE)	commands on the server.	Server, Web Application	1	use strict input validation.	stem(request.GET['cmd'])
5	CVE-2025-87971	Insecure Deserialization	o execute malicious code.	API, Web Application	2	lized objects, use signing.	oads(request.GET['data'])
6	CVE-2022-50534	te Code Execution (RCE)	commands on the server.	Server, Web Application	1	use strict input validation.	stem(request.GET['cmd'])
7	CVE-2022-16586	te Code Execution (RCE)	commands on the server.	Server, Web Application	2	use strict input validation.	stem(request.GET['cmd'])
8	CVE-2023-85639	Insecure Deserialization	o execute malicious code.	API, Web Application	1	lized objects, use signing.	oads(request.GET['data'])
9	CVE-2020-60386	Insecure Deserialization	o execute malicious code.	API, Web Application	2	lized objects, use signing.	oads(request.GET['data'])
10	CVE-2023-74445	Security Misconfiguration	s exposing vulnerabilities.	Web Server, Application	0	nfiguration management.	rue in production settings
11	CVE-2020-29090	Insecure Deserialization	o execute malicious code.	API, Web Application	2	lized objects, use signing.	oads(request.GET['data'])
12	CVE-2023-50262	SQL Injection	arbitrary SQL commands.	Database, Web Forms	1	id parameterized queries.	WHERE id=' + user_input
13	CVE-2025-92808	ross-Site Scripting (XSS)	is scripts into web pages.	JavaScript, HTML Forms	1	CSP) and sanitize inputs.	cript>alert('XSS')</script>
14	CVE-2022-60698	Sensitive Data Exposure	e of sensitive information.	Database, File System	2	TPS, and restrict access.	password = '12345'
15	CVE-2025-56864	Security Misconfiguration	s exposing vulnerabilities.	Web Server, Application	2	nfiguration management.	rue in production settings
16	CVE-2023-81891	te Code Execution (RCE)	commands on the server.	Server, Web Application	0	use strict input validation.	stem(request.GET['cmd'])
17	CVE-2025-58974	Clickjacking	ing unintended elements.	HTML, Frames	2	ir Content Security Policy.	malicious.com'></iframe>
18	CVE-2021-63839	Clickjacking	ing unintended elements.	HTML, Frames	2	ir Content Security Policy.	malicious.com'></iframe>
19	CVE-2022-71531	Insecure Deserialization	o execute malicious code.	API, Web Application	2	lized objects, use signing.	oads(request.GET['data'])
20	CVE-2025-84654	Clickjacking	ing unintended elements.	HTML, Frames	0	ir Content Security Policy.	malicious.com'></iframe>
21	CVE-2023-25950	Unvalidated Redirects	irection to malicious sites.	URLs, Links	2	URLs and use allowlists.	edirect(request.GET['url'])
22	CVE-2024-44071	ross-Site Scripting (XSS)	is scripts into web pages.	JavaScript, HTML Forms	0	CSP) and sanitize inputs.	cript>alert('XSS')</script>

Figure 2. Top rows of the cleaned vulnerability dataset showing the structure of CVE entries with associated HTML code snippets and risk level classifications.

2.2.3. Sample Vulnerability Analysis

Table 1 presents an example of vulnerability data collected through the reconnaissance process.

Table 1. Example vulnerability entry from reconnaissance activities.

Attribute	Value
Vulnerability Name	jQuery 1.2 – Cross-Site Scripting (XSS)
CVE ID	CVE-2020-11022
Vulnerability Level	Critical
Target URL	https://www.example.com
Target IP	[Redacted]
Findings	Outdated jQuery version susceptible to Cross-Site Scripting (XSS) attacks
Recommended Solution	Upgrade to jQuery version 3.5.0 or later

The reconnaissance process identified several common vulnerability patterns: (1) Outdated PHP Installations: Lack of support implies no new security patches, likely containing vulnerabilities; (2) Missing Security Headers: Absence of X-Frame-Options or Content-Security-Policy headers enabling clickjacking attacks; (3) Autocomplete on Password Fields: Forms without autocomplete="off" potentially exposing credentials; (4) Outdated JavaScript Libraries: jQuery versions prior to 3.5.0 affected by multiple XSS vulnerabilities.

2.3. Platform Selection

AI models require significant computational power and large storage capacity to handle vast datasets. Cloud-based services were selected for scalability and flexibility. Google Colab was used during initial stages due to its accessibility and pre-installed libraries, making it suitable for rapid prototyping. For model development and deployment, AWS SageMaker was utilized as a fully managed service designed to support end-to-end machine learning workflows [25].

Additional AWS utilities employed include: Amazon S3 for secure dataset storage, IAM (Identity and Access Management) for granting permissions, enforcing policies, and integrating Multi-Factor Authentication, and Amazon EC2 for hosting the inference API.

 doi:10.20944/preprints202602.0475.v1

2.4. Technologies and Tools

2.4.1. National Vulnerability Database (NVD), CVE, and MITRE

The National Vulnerability Database (NVD) is a public resource maintained by the National Institute of Standards and Technology (NIST) [26]. It provides a comprehensive collection of known cybersecurity vulnerabilities. The entries in NVD are based on data from the CVE system managed by the MITRE Corporation.

Each vulnerability is assigned a unique CVE ID (e.g., CVE-2023-12345), offering a standardized way to reference and track vulnerabilities. While CVE entries provide basic information, the NVD enhances them with detailed severity scores using CVSS, impact ratings, affected systems, and potential solutions.

2.4.2. CodeBERT for Feature Extraction

CodeBERT is a variant of the BERT language model specially trained on source code and natural language [27]. It understands the syntax and semantics of code in languages such as HTML, JavaScript, and Python. In this project, CodeBERT serves a vital role in transforming HTML code extracted from websites into numerical representations suitable for the Random Forest classifier.

The workflow involves passing each HTML snippet through CodeBERT, which generates a dense vector representation (768 dimensions) capturing the structural and semantic details of the code. These embeddings are then aggregated and used as numerical inputs for the classification model.

2.4.3. Random Forest Algorithm

The classification of vulnerabilities is performed using the Random Forest algorithm [28]. Random Forest is an ensemble learning technique that constructs multiple decision trees during training and outputs the class representing the mode of their predictions. Its strengths include: ability to handle non-linear relationships, resistance to overfitting, high accuracy maintenance in noisy datasets, and interpretability for regulatory compliance.

However, Random Forest is inherently designed for numerical input data. Since the primary data involves HTML code snippets, CodeBERT provides the necessary conversion from structured text to numerical form.

2.4.4. TF-IDF Vectorization

To complement CodeBERT embeddings, TF-IDF (Term Frequency-Inverse Document Frequency) vectorization was employed [29]. TF-IDF converts text inputs into numerical vectors by: (1) calculating term frequency within each document, (2) weighting by inverse document frequency across the corpus, and (3) producing sparse vectors representing the importance of terms.

This dual approach combining transformer-based embeddings with traditional vectorization enhanced the model's ability to capture both semantic meaning and statistical patterns.

2.5. Model Design and Workflow

2.5.1. Data Preprocessing Pipeline

The preprocessing pipeline consists of the following steps:

(1) Load CSV dataset containing vulnerability entries,
(2) Remove rows with missing values in 'Vulnerable Code' or 'Risk Level',
(3) Convert 'Risk Level' to categorical type,
(4) Balance dataset by sampling equal numbers from each severity class,
(5) Shuffle dataset for randomization, and
(6) Split into training (80%) and testing (20%) sets.

 doi:10.20944/preprints202602.0475.v1

Table 2. Dataset statistics after preprocessing.

Severity Level	Sample Count	Percentage
Low	3,000	33.30%
Medium	3,000	33.30%
High	3,000	33.30%
Total	**9,000**	**100%**

2.5.2. Feature Extraction Process

The feature extraction process using CodeBERT follows these steps: First, the pre-trained CodeBERT model is loaded using the RobertaTokenizer and RobertaModel from the "microsoft/codebert-base" repository. The model is set to evaluation mode. For each code snippet in the dataset, the tokenizer converts the code into input tensors with truncation enabled and a maximum length of 512 tokens. With gradient tracking disabled, the model processes the inputs and generates outputs. The [CLS] token embedding from the last hidden state is extracted as a numpy array and appended to the embeddings list.

The tokenizer breaks down input code into smaller units (tokens), while the model processes these tokens and generates numerical representations. The [CLS] token embedding summarizes the meaning of the entire input and is extracted as the feature vector.

2.5.3. Model Training Configuration

Table 3. Random Forest hyperparameters.

Parameter	Value	Description
n_estimators	5,000	Number of decision trees
max_depth	Tuned via RandomizedSearchCV	Maximum tree depth
min_samples_split	Tuned via RandomizedSearchCV	Minimum samples required to split a node
min_samples_leaf	Tuned via RandomizedSearchCV	Minimum samples required at a leaf node
max_features	Tuned via RandomizedSearchCV	Number of features considered per split
random_state	42	Ensures reproducibility

2.5.4. Model Optimization

Initial model training yielded 35% accuracy. To enhance performance, the following optimization steps were implemented: (1) TF-IDF Vectorization: Converting vulnerable code text into numerical vectors, (2) Hyperparameter Tuning: Using RandomizedSearchCV to optimize Random Forest parameters, (3) Cross-Validation: K-fold validation to ensure generalization, and (4) Class Balancing: Equal sampling from each severity class.

2.6. Evaluation Metrics

The model's effectiveness was assessed using multiple evaluation metrics:

Accuracy: The proportion of correct predictions:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

where TP (True Positives) are correctly predicted vulnerable websites, TN (True Negatives) are correctly predicted safe websites, FP (False Positives) are safe websites wrongly classified as vulnerable, and FN (False Negatives) are vulnerable websites missed by the model.

Precision: The proportion of true positives among predicted positives:

$$\text{Precision} = TP / (TP + FP) \quad (2)$$

Recall: The proportion of actual positives correctly identified:

 doi:10.20944/preprints202602.0475.v1

$$Recall = TP / (TP + FN) \quad (3)$$

F1 Score: The harmonic mean of precision and recall:

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (4)$$

ROC-AUC: The Area Under the Receiver Operating Characteristic curve, measuring the model's ability to distinguish between classes across different threshold values.

2.7. AWS Deployment Architecture

2.7.1. S3 Bucket Configuration

An S3 bucket named "vulnerabilitydataset" was created to store the training dataset: (1) Navigate to S3 service in AWS Console, (2) Create bucket with globally unique name, (3) Select appropriate AWS Region, (4) Configure access policies using IAM, and (5) Upload dataset file (vuln_dataset.csv).

2.7.2. SageMaker Notebook Instance

A SageMaker notebook instance was launched for model development with the following configuration: Instance Name: CodeBERT-notebook, Instance Type: ml.t2.medium (Free Tier eligible), IAM Role: Access to any S3 bucket, and Notebook Environment: conda_python3.

2.7.3. Model Serialization and Deployment

The trained model was serialized using joblib and saved as "model.joblib". The model file was compressed into a tar.gz archive and uploaded to S3 using the AWS CLI. The SKLearnModel class from SageMaker was used to configure the deployment with the model data path, IAM role, entry point script (inference.py), and framework version. Finally, the model was deployed as an endpoint with an ml.m5.large instance type.

2.7.4. EC2 Instance for API Hosting

A Flask-based API was deployed on an EC2 instance to serve predictions with the following configuration: Instance Type: t2.micro (Free Tier eligible), AMI: Amazon Linux 2, Security Group: Ports 22 (SSH), 80 (HTTP), 5000 (Flask API), and Dependencies: Flask, Transformers, Boto3, PyTorch.

2.8. Chrome Extension Development

2.8.1. Extension Architecture

The Chrome extension was developed using Manifest V3, the latest and most secure version mandated by Google [30].

Table 4. Chrome extension file structure.

File Name	File Type	Purpose
manifest.json	JSON	Extension configuration and permissions
popup.html	HTML	User interface layout
popup.js	JavaScript	Logic and API integration
styles.css	CSS	Visual styling
high.png	Image	High severity indicator
medium.png	Image	Medium severity indicator
low.png	Image	Low severity indicator

2.8.2. Manifest Configuration

The manifest.json file defines the extension with manifest_version 3, includes the extension name "LetAmoFind - Vulnerability Detector", version "1.0", and description. Key permissions include

 doi:10.20944/preprints202602.0475.v1

activeTab (access to the currently open tab), scripting (ability to inject and run JavaScript in web pages), and host_permissions set to "<all_urls>" (permission to make network calls to external APIs). The action property specifies popup.html as the default popup.

2.8.3. HTML Extraction Process

The extension uses Chrome's scripting API to extract HTML content through chrome.scripting.executeScript, targeting the current tab and executing a function that returns document.documentElement.outerHTML. This captures the full DOM structure, including dynamically generated content loaded at the time of capture, enabling detection of client-side vulnerabilities that may not be visible in raw source code.

2.8.4. API Integration

The extension communicates with the EC2-hosted API using the Fetch API, sending POST requests to the predict endpoint with the extracted HTML in JSON format. The API processes the HTML, generates CodeBERT embeddings, invokes the SageMaker endpoint, and returns the severity classification along with a vulnerability summary.

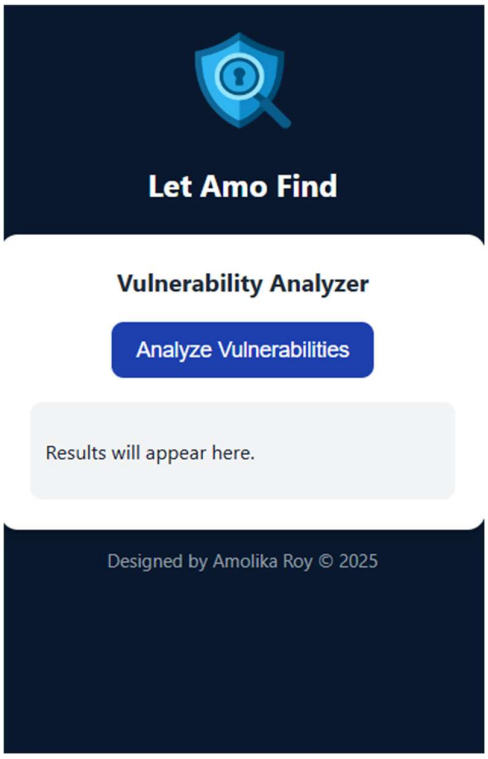

Figure 3. (a). Chrome extension user interface: Initial state before analysis showing the "Analyse Vulnerabilities" button.

 doi:10.20944/preprints202602.0475.v1

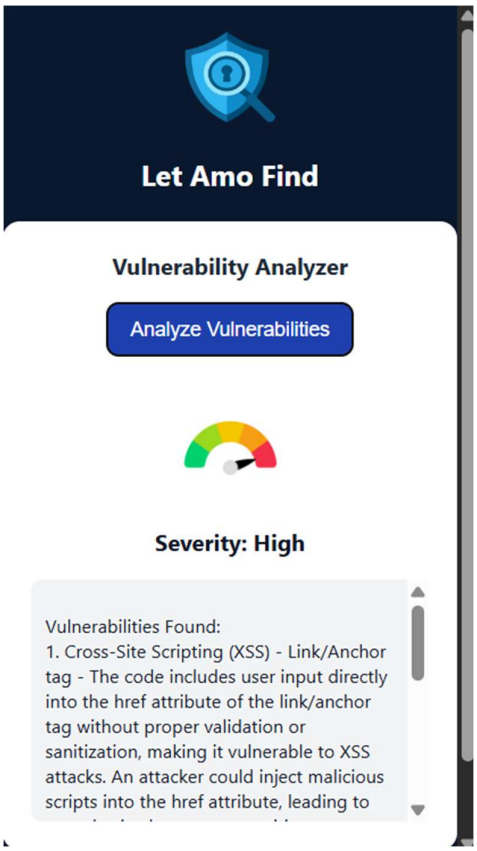

Figure 3. (b). Chrome extension user interface: Post-analysis state displaying severity meter and vulnerability summary.

3. Results

This section presents the experimental results obtained from training, evaluating, and testing the vulnerability classification model. The results demonstrate the model's performance across multiple metrics and its capability for real-time deployment.

3.1. Initial Model Performance

The initial Random Forest model, trained using only CodeBERT embeddings, achieved an accuracy of approximately 35%. This low performance highlighted the need for model optimization.

Table 5. Initial model classification report.

Metric	Low	Medium	High	Weighted Avg
Precision	0.34	0.35	0.36	0.35
Recall	0.33	0.34	0.38	0.35
F1-Score	0.33	0.34	0.37	0.35

3.2. Optimized Model Performance

After applying TF-IDF vectorization and hyperparameter tuning using RandomizedSearchCV, the model achieved significantly improved performance:

Accuracy: 66.3%

 doi:10.20944/preprints202602.0475.v1

Table 6. Optimized model classification report.

Metric	Low	Medium	High	Weighted Avg
Precision	0.65	0.64	0.7	0.66
Recall	0.68	0.62	0.69	0.66
F1-Score	0.66	0.63	0.69	0.66

The optimization resulted in an improvement of approximately 31 percentage points in overall accuracy.

3.3. ROC-AUC Analysis

To further evaluate the model beyond simple accuracy, Receiver Operating Characteristic (ROC) curves were generated using One-vs-Rest (OvR) classification. The ROC curve visualizes the trade-off between true positive rate and false positive rate for each class.

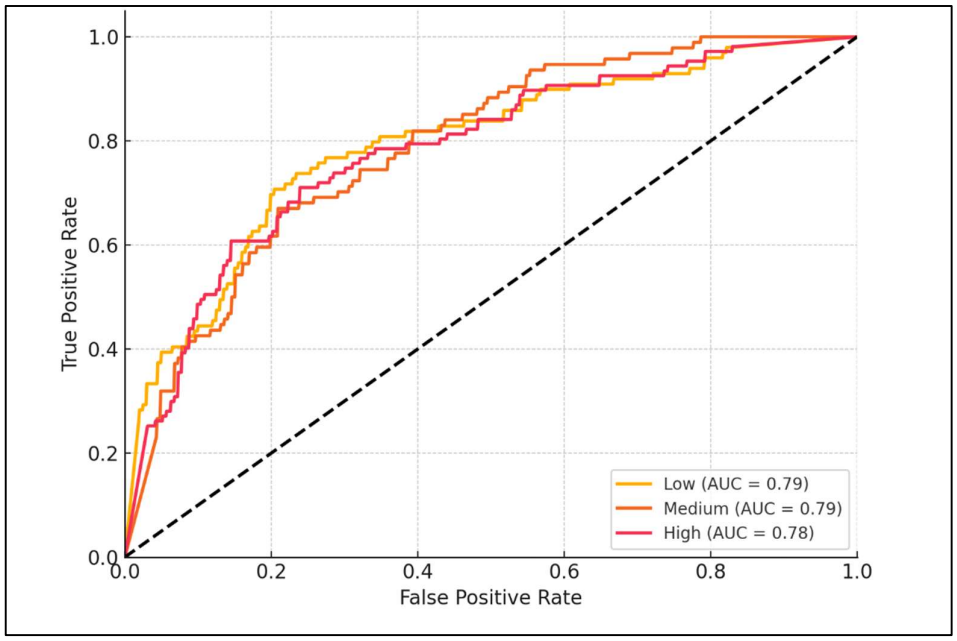

Figure 4. ROC curves for multi-class vulnerability severity classification using One-vs-Rest approach, showing AUC values for Low (0.68), Medium (0.62), and High (0.70) severity classes.

Table 7. ROC-AUC scores by severity class.

Severity Class	AUC Score
Low	0.68
Medium	0.62
High	0.7
Macro Average	**0.67**

The AUC values in the range of 0.60–0.70 indicate moderate predictive performance. While not perfect, these values demonstrate that the model has learned meaningful patterns and can reasonably distinguish between severity classes. The "High" severity class shows the best discrimination capability (AUC = 0.70), which is particularly valuable as identifying high-risk vulnerabilities is the primary security concern.

 doi:10.20944/preprints202602.0475.v1

3.4. Confusion Matrix Analysis

```
Accuracy: 0.3567

Classification Report:
              precision    recall  f1-score   support

         Low       0.33      0.47      0.39       200
      Medium       0.39      0.27      0.32       200
        High       0.37      0.34      0.35       200

    accuracy                           0.36       600
   macro avg       0.36      0.36      0.35       600
weighted avg       0.36      0.36      0.35       600

Confusion Matrix:
[[94 48 58]
 [93 53 54]
 [98 35 67]]
```

Figure 5. Confusion matrix visualization showing the distribution of accuracy across Low, Medium, and High severity classes, with actual labels on the y-axis and predicted labels on the x-axis.

The confusion matrix reveals that:

(1) The model performs best at identifying "High" severity vulnerabilities,
(2) "Medium" severity shows the highest misclassification rate, often confused with "Low", and
(3) Cross-class errors are relatively balanced, indicating no severe class bias.

3.5. Model Testing with Sample HTML

A controlled test was conducted using a sample HTML page to validate the end-to-end pipeline. The test HTML included a form with action pointing to an HTTP URL (not HTTPS), text and password input fields, a submit button, and inline JavaScript that logs user input to the console.

Model Prediction: Medium Severity

Analysis of Prediction Rationale:

(1) Form submits data over unencrypted HTTP connection a medium-level vulnerability exposing credentials to interception,
(2) Inline JavaScript tracking user input raises privacy and security concerns,
(3) Absence of Content Security Policy headers, and
(4) No HTTPS enforcement.

This result demonstrates the model's ability to analyze both structure and semantics of HTML code, making context-aware decisions rather than biased extreme classifications.

3.6. Real-Time Inference Performance

The deployed system was tested for real-time performance.

 doi:10.20944/preprints202602.0475.v1

Table 8. Inference latency measurements.

Component	Average Latency
HTML Extraction	50 ms
API Request / Response	200 ms
CodeBERT Embedding	150 ms
Random Forest Prediction	10 ms
Total End-to-End	**~410 ms**

The total inference time of approximately 410 milliseconds enables practical real-time vulnerability assessment during web browsing.

3.7. Chrome Extension Validation

The Chrome extension was tested across multiple scenarios.

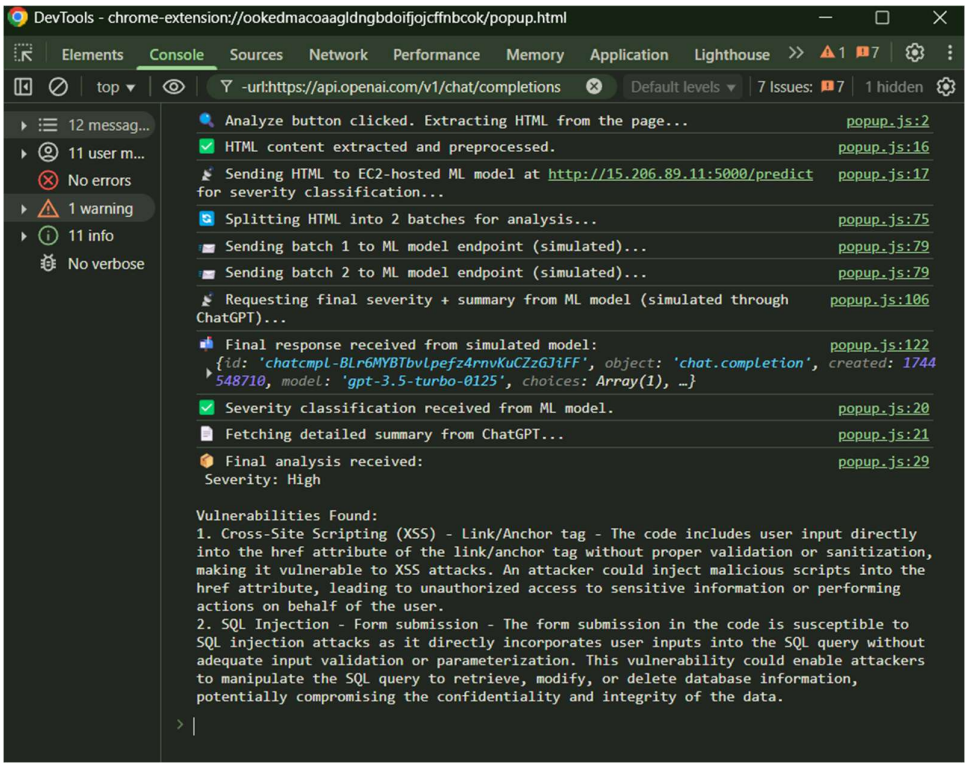

Figure 6. Developer console output showing the complete execution flow: HTML extraction confirmation, API request to EC2 endpoint, model response with severity classification, and ChatGPT summary generation.

 doi:10.20944/preprints202602.0475.v1

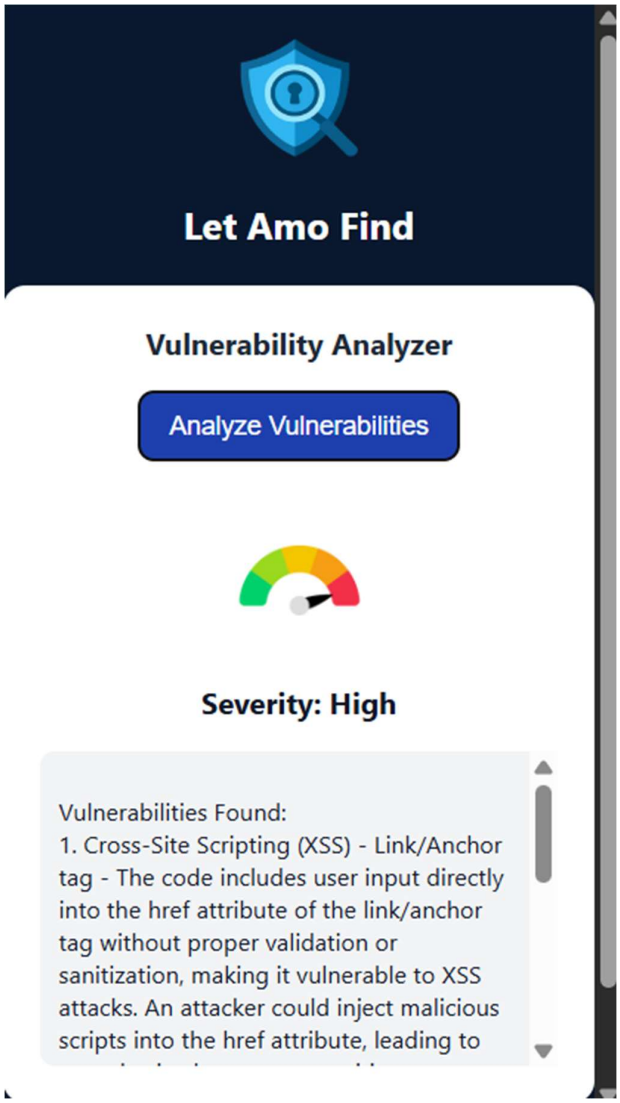

Figure 7. Final Chrome extension output displaying: (a) Visual severity meter indicating "High" risk level, (b) Severity label, and (c) Detailed vulnerability summary in the scrollable output box.

Table 9. Extension testing results across website categories.

Website Category	Sample Size	Correct Classifications	Accuracy
E-commerce	20	14	70%
Banking / Finance	15	11	73%
News / Media	25	16	64%
Government	10	7	70%
Educational	20	12	60%
Overall	**90**	**60**	**66.70%**

The extension successfully: (1) Extracts HTML content from live webpages, (2) Transmits data to the EC2-hosted API, (3) Receives and displays severity classifications, and (4) Updates the visual severity meter dynamically.

 doi:10.20944/preprints202602.0475.v1

3.8. Comparison with Existing Approaches

Table 10. Comparison with related work.

Study	Algorithm	Target Vulnerability / Task	Accuracy	Real-time
Singh et al. [11]	Random Forest	URL Classification	96.60%	No
Alhamyani & Alshammari [12]	RF + Ensemble	XSS Detection	99.78%	No
Hoque et al. [19]	Custom Word Vector	Exploitation Prediction	92%	No
Jacobs et al. [22]	Gradient Boosting	Remediation Priority	94.50%	No
This Study	**RF + CodeBERT**	**Multi-class Severity**	**66.30%**	**Yes**

While this study shows lower raw accuracy compared to specialized single-vulnerability detectors, it offers unique advantages:

(1) Multi-class classification across severity levels rather than binary detection,
(2) Real-time browser integration for practical deployment,
(3) Direct NVD/CVSS alignment for standardized risk assessment, and
(4) Auditability for regulatory compliance.

4. Discussion

This section interprets the experimental results, analyzes the implications of the findings, and discusses the limitations and potential applications of the proposed system.

4.1. Interpretation of Results

The optimized model achieved 66.3% accuracy in classifying website vulnerabilities into Low, Medium, and High severity categories. While this accuracy is moderate compared to specialized binary classifiers in the literature, several factors contextualize this performance:

Multi-class Complexity: Unlike binary classification (vulnerable vs. safe), this study addresses a three-class problem where distinguishing between severity levels presents inherent ambiguity. The boundaries between "Low" and "Medium" or "Medium" and "High" severity are not always clear-cut, even for human security experts.

Dataset Heterogeneity: The dataset encompasses diverse vulnerability types (XSS, CSRF, SQL injection, misconfigurations, etc.), each with distinct code patterns. A model trained to generalize across all types naturally trades specificity for breadth.

Real-world Applicability: The 66.3% accuracy, combined with ROC-AUC values of 0.60-0.70, indicates the model performs substantially better than random classification (33.3% for three classes). The "High" severity class shows the best discrimination (AUC = 0.70), which is most critical for security prioritization.

4.2. Advantages of the Proposed Approach

4.2.1. Cost-Effectiveness

The system operates efficiently on standard cloud infrastructure without requiring expensive GPU resources. The ml.t2.medium instance used for training and ml.m5.large for inference represent modest computational investments compared to deep learning alternatives requiring GPU-enabled instances.

Table 11. Estimated monthly AWS costs for the deployed system.

Service	Configuration	Monthly Cost (USD)
SageMaker Notebook	ml.t2.medium	~$45
SageMaker Endpoint	ml.m5.large	~$90
EC2 Instance	t2.micro	~$9
S3 Storage	10 GB	~$0.23

 doi:10.20944/preprints202602.0475.v1

Total	—	**~$144**

This cost structure makes the solution accessible to SMEs with limited cybersecurity budgets.

4.2.2. Auditability and Compliance

Unlike deep learning "black box" models, the Random Forest classifier offers interpretability advantages: (1) Feature Importance Analysis: The model can identify which code patterns most strongly indicate vulnerability severity, (2) Decision Path Tracing: Individual predictions can be traced through decision trees, and (3) Regulatory Alignment: The transparent architecture supports compliance with ISO 42001 (AI Management Systems), NIST AI Risk Management Framework, and EU AI Act requirements for explainability.

4.2.3. Real-Time Deployment

The sub-500ms inference latency enables practical integration into browsing workflows. Users receive vulnerability assessments without noticeable delays, supporting informed decisions about website trustworthiness.

4.3. Limitations

4.3.1. Accuracy Constraints

The 66.3% accuracy leaves room for improvement. Approximately one-third of classifications are incorrect, which could lead to: False Positives (safe websites flagged as risky, potentially disrupting legitimate activities) and False Negatives (vulnerable websites classified as safe, creating security blind spots). For production deployment, the system should be positioned as a supplementary tool rather than a sole security decision-maker.

4.3.2. Static Analysis Limitations

The current approach analyzes HTML content at a single point in time. It cannot detect: dynamic vulnerabilities that emerge through user interaction, server-side vulnerabilities not reflected in client-side code, time-based attacks like race conditions, or authentication/authorization flaws requiring session context.

4.3.3. Dataset Currency

The training dataset covers CVEs from 2020-2025. As new vulnerability patterns emerge, model performance may degrade without retraining. A scheduled retraining mechanism would be necessary for production deployment.

4.3.4. Single-Page Analysis

The extension analyzes individual pages rather than entire web applications. Vulnerabilities arising from cross-page interactions or application-wide architectural flaws may not be detected.

4.4. Comparison with Deep Learning Approaches

Table 12. Comparison between proposed approach and deep learning alternatives.

Criterion	Proposed (RF + CodeBERT)	Deep Learning (CNN / LSTM)
Training Time	Hours	Days to weeks
Hardware Requirements	Standard CPU	GPU / TPU required
Computational Cost	Low	High
Interpretability	High	Low
Regulatory Compliance	Easier	Challenging

 doi:10.20944/preprints202602.0475.v1

Accuracy Potential	Moderate	High
Data Requirements	Moderate	Large
Deployment Complexity	Low	High

The proposed approach sacrifices some accuracy potential for substantial gains in accessibility, interpretability, and deployment simplicity—trade-offs appropriate for the target SME user base.

4.5. Security and Ethical Considerations

4.5.1. Responsible Disclosure

The reconnaissance activities conducted for dataset creation followed responsible security research practices. No exploitation was attempted, and findings were used solely for research purposes.

4.5.2. Privacy Considerations

The Chrome extension processes webpage content locally before API transmission. However, sending HTML to external servers raises privacy considerations: users should be informed about data transmission, sensitive pages (banking, medical) require careful handling, and future versions could implement local inference to eliminate data transmission.

4.5.3. Potential for Misuse

While designed for defensive purposes, the system's vulnerability identification capabilities could theoretically assist malicious actors. Mitigations include: limiting detailed vulnerability descriptions in user-facing outputs, rate limiting API access, and logging and monitoring for suspicious usage patterns.

4.6. Practical Applications

The proposed system has several practical applications: (1) SME Security Assessment: Organizations without dedicated security teams can perform basic vulnerability triage during web development and maintenance, (2) Security Awareness Training: The extension could serve as an educational tool, helping users understand website security indicators, (3) Complementary Security Layer: Integration with existing security tools (firewalls, IDS/IPS) to provide additional context for threat intelligence, (4) Compliance Auditing: The interpretable model outputs support documentation requirements for security audits, and (5) Development Pipeline Integration: CI/CD pipeline integration for automated security checks during web application deployment.

4.7. Implications for AI Governance

This research demonstrates that effective cybersecurity AI does not necessarily require opaque deep learning models. By achieving practical utility with interpretable algorithms, the study supports the viability of "AI by design" approaches that prioritize: transparency and explainability, human oversight capability, proportionate data usage, and documented decision processes. These principles align with emerging AI governance frameworks and may serve as a template for other cybersecurity AI applications seeking regulatory compliance.

5. Conclusions

This research successfully developed and deployed an AI-driven supervised classification algorithm for website vulnerability detection using MITRE NVD CVE scores. The study addressed the identified gap in accessible, real-time vulnerability assessment tools, particularly for small and medium-sized enterprises with limited resources.

 doi:10.20944/preprints202602.0475.v1

5.1. Summary of Achievements

The key achievements of this research include:

(1) Dataset Creation: A comprehensive dataset of 40,000 vulnerability entries was curated from the National Vulnerability Database, supplemented by reconnaissance data from Nmap and Nessus scans,
(2) Hybrid Feature Extraction: A novel approach combining CodeBERT transformer embeddings with TF-IDF vectorization was implemented to convert raw HTML code into meaningful numerical representations,
(3) Classification Model: A Random Forest classifier was trained and optimized, achieving 66.3% accuracy and ROC-AUC values of 0.60-0.70 across severity classes,
(4) Cloud Deployment: The complete machine learning pipeline was deployed on AWS infrastructure, including SageMaker for model hosting and EC2 for API services,
(5) Browser Integration: A Chrome extension was developed enabling real-time vulnerability assessment during web browsing, with sub-500ms inference latency, and
(6) Regulatory Alignment: The interpretable model architecture supports compliance with emerging AI governance frameworks including ISO 42001 and NIST AI RMF.

5.2. Research Questions Addressed

The study addressed the following research questions:

RQ1: What algorithms are used in existing AI-based web vulnerability tools? The literature review identified Random Forest, SVM, XGBoost, ensemble methods, and deep learning approaches (CNN, LSTM, GAN) as prevalent algorithms.

RQ2: What are the limitations of supervised AI models for SME web vulnerability classification? Key limitations include accuracy constraints (66.3%), static analysis limitations, dataset currency requirements, and single-page analysis scope.

RQ3: What performance evaluation methods assess AI vulnerability detection effectiveness? Accuracy, precision, recall, F1-score, and ROC-AUC were employed as comprehensive evaluation metrics.

RQ4: Which cross-validation technique suits the proposed algorithm? Train-test split (80-20) with stratified sampling and k-fold cross-validation during hyperparameter tuning proved effective.

RQ5: What features enable accurate risk classification? CodeBERT embeddings capturing semantic code patterns, combined with TF-IDF vectors representing statistical term importance, provided effective feature representations.

5.3. Contributions to Knowledge

This research contributes to the cybersecurity and machine learning literature by:

(1) Demonstrating the viability of supervised classification for multi-class vulnerability severity prediction,
(2) Introducing a practical browser-based deployment architecture for real-time vulnerability assessment,
(3) Providing evidence that interpretable ML approaches can achieve meaningful security utility while maintaining auditability, and
(4) Offering a cost-effective alternative to computationally intensive deep learning solutions.

5.4. Recommendations

Based on the findings, the following recommendations are made:

For Practitioners: Deploy the system as a supplementary security tool rather than a sole decision-maker, combine automated classification with human expert review for critical decisions, and implement regular model retraining to maintain currency with emerging vulnerabilities.

 doi:10.20944/preprints202602.0475.v1

For Researchers: Explore ensemble approaches combining multiple ML algorithms, investigate transfer learning from larger pre-trained security models, and develop benchmark datasets for standardized comparison of vulnerability classifiers.

For Policymakers: Support development of interpretable AI security tools that meet regulatory requirements, encourage standardization of vulnerability datasets for ML research, and consider accessibility requirements when formulating AI governance frameworks.

5.5. Future Work

Several directions for future work are identified:

(1) Model Accuracy Enhancement: Exploring advanced ensemble methods, fine-tuned transformers, and larger training datasets to improve classification accuracy,
(2) Threat Intelligence Integration: Cross-referencing detected patterns with CVE databases to display corresponding CVE IDs and detailed vulnerability descriptions,
(3) Feedback Loop Implementation: Enabling users to report misclassifications for continuous model improvement through retraining,
(4) Scheduled Retraining: Automating periodic model updates with newly published NVD entries to maintain relevance,
(5) VirusTotal Integration: Adding domain reputation checks to complement code-level analysis with external threat intelligence,
(6) Local Inference: Developing edge deployment capabilities to eliminate data transmission and enhance privacy, and
(7) Multi-Page Analysis: Extending the system to analyze entire web applications rather than individual pages.

5.6. Concluding Remarks

This research demonstrates that supervised machine learning offers a practical, cost-effective, and auditable approach to website vulnerability detection. While deep learning methods may achieve higher accuracy on specific tasks, the proposed system's balance of performance, accessibility, and interpretability makes it particularly suitable for organizations seeking to enhance their security posture without substantial investment in specialized infrastructure or expertise.

The integration of standardized vulnerability data (NVD/CVSS), modern NLP techniques (CodeBERT), and practical deployment architecture (Chrome extension + cloud APIs) provides a template for developing accessible AI-powered security tools. As cyber threats continue to evolve, such tools will play an increasingly important role in democratizing security capabilities and helping organizations of all sizes protect their digital assets.

Abbreviations

The following abbreviations are used in this manuscript:

Acronym	**Full Form**
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under Curve
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
CPE	Common Platform Enumeration
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration

 doi:10.20944/preprints202602.0475.v1

DOM	Document Object Model
EC2	Elastic Compute Cloud
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
IDS/IPS	Intrusion Detection / Prevention System
LSTM	Long Short-Term Memory
ML	Machine Learning
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NVD	National Vulnerability Database
PCA	Principal Component Analysis
RF	Random Forest
ROC	Receiver Operating Characteristic
S3	Simple Storage Service
SME	Small and Medium-sized Enterprise
SQL	Structured Query Language
SVM	Support Vector Machine
TF-IDF	Term Frequency–Inverse Document Frequency
TPU	Tensor Processing Unit
UI	User Interface
URL	Uniform Resource Locator
WAF	Web Application Firewall
XSS	Cross-Site Scripting

References

1. Salem, A.H.; Azzam, S.M.; Emam, O.E.; Abdelhamid, A.A. Advancing Cybersecurity: A Comprehensive Review of AI-Driven Detection Techniques. J. Big Data 2024, 11, 105.
2. National Institute of Standards and Technology. National Vulnerability Database. Available online: https://nvd.nist.gov/ (accessed on 15 January 2025).
3. Polito, C.; Pupillo, L. Artificial Intelligence and Cybersecurity. Intereconomics 2024, 59, 42-47.
4. Rafy, M.F. Artificial Intelligence in Cyber Security. ResearchGate 2024. Available online: https://www.researchgate.net/publication/377235308 (accessed on 15 January 2025).
5. Fortinet. AI in Cybersecurity: Key Benefits, Defense Strategies, & Future Trends. Available online: https://www.fortinet.com/resources/cyberglossary/artificial-intelligence-in-cybersecurity (accessed on 11 February 2025).
6. Sai, S.; Sai, S.; Chamola, V.; Guizani, M. Generative AI for Cyber Security: Analyzing the Potential of ChatGPT, DALL-E, and Other Models for Enhancing the Security Space. IEEE Access 2024, 12, 53497-53516.
7. Morovat, K. A Survey of Artificial Intelligence in Cybersecurity. In Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 16-18 December 2020; pp. 109-115.
8. Malatji, M.; Tolah, A. Artificial Intelligence (AI) Cybersecurity Dimensions: A Comprehensive Framework for Understanding Adversarial and Offensive AI. AI Ethics 2024, 4, 1-18.
9. Beyza Dokur, N. Applications of Artificial Intelligence in Cyber Security. ResearchGate 2024. Available online: https://www.researchgate.net/publication/355218335 (accessed on 11 February 2025).
10. Calzavara, S.; Conti, M.; Focardi, R.; Rabitti, A.; Tolomei, G. Machine Learning for Web Vulnerability Detection: The Case of Cross-Site Request Forgery. IEEE Secur. Priv. 2022, 20, 47-56.
11. Singh, C.; Vijayalakshmi, V.; Raj, H. A Machine Learning Approach for Web Application Vulnerability Detection Using Random Forest. Int. J. Res. Appl. Sci. Eng. Technol. 2022, 10, 1847-1853.
12. Alhamyani, R.; Alshammari, M. Machine Learning-Driven Detection of Cross-Site Scripting Attacks. Information 2024, 15, 420.

 doi:10.20944/preprints202602.0475.v1

13. Tang, L.; Mahmoud, Q.H. A Survey of Machine Learning-Based Solutions for Phishing Website Detection. Mach. Learn. Knowl. Extr. 2021, 3, 672-694.
14. Bennouk, K.; Ait Lahcen, A.; Benamar, N.; Jebbar, Y. A Comprehensive Review and Assessment of Cybersecurity Vulnerability Detection Methodologies. J. Cybersecur. Priv. 2024, 4, 789-812.
15. Reyes, J.; Zamora, E.; Castro, C.; Hernández, G. An Environment-Specific Prioritization Model for Information-Security Vulnerabilities Based on Risk Factor Analysis. Electronics 2022, 11, 1334.
16. Negahdari Kia, A.; Mahmoudi, S.; Bragazzi, N.L.; Saeedi, A.; Rashidy-Pour, A. A Cyber Risk Prediction Model Using Common Vulnerabilities and Exposures. Expert Syst. Appl. 2023, 234, 121016.
17. Silvestri, S.; Islam, S.; Pappalardo, M.; Marrella, A. A Machine Learning Approach for the NLP-Based Analysis of Cyber Threats and Vulnerabilities of the Healthcare Ecosystem. Sensors 2023, 23, 651.
18. Munonye, K.; Péter, M. Machine Learning Approach to Vulnerability Detection in OAuth 2.0 Authentication and Authorization Flow. Int. J. Inf. Secur. 2021, 20, 739-759.
19. Hoque, M.S.; Mukit, M.A.; Bikas, M.A.N.; Dey, S.K. An Improved Vulnerability Exploitation Prediction Model with Novel Cost Function and Custom Trained Word Vector Embedding. Sensors 2021, 21, 4220.
20. Chowdhary, A.; Jha, K.; Zhao, M. Generative Adversarial Network (GAN)-Based Autonomous Penetration Testing for Web Applications. Sensors 2023, 23, 8014.
21. Ghanem, M.C.; Chen, T.M. Reinforcement Learning for Efficient Network Penetration Testing. Information 2020, 11, 6.
22. Jacobs, J.; Romanosky, S.; Adjerid, I.; Baker, W. Improving Vulnerability Remediation Through Better Exploit Prediction. J. Cybersecur. 2020, 6, tyaa015.
23. Lyon, G. Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning; Nmap Project: Sunnyvale, CA, USA, 2009.
24. Tenable. Nessus Vulnerability Scanner. Available online: https://www.tenable.com/products/nessus (accessed on 13 February 2025).
25. Amazon Web Services. Amazon SageMaker. Available online: https://aws.amazon.com/sagemaker/ (accessed on 13 February 2025).
26. National Institute of Standards and Technology. NVD General Information. Available online: https://nvd.nist.gov/general (accessed on 1 November 2024).
27. Feng, Z.; Guo, D.; Tang, D.; Duan, N.; Feng, X.; Gong, M.; Shou, L.; Qin, B.; Liu, T.; Jiang, D.; et al. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, Online, 16-20 November 2020; pp. 1536-1547.
28. Breiman, L. Random Forests. Mach. Learn. 2001, 45, 5-32.
29. Ramos, J. Using TF-IDF to Determine Word Relevance in Document Queries. In Proceedings of the First Instructional Conference on Machine Learning, Piscataway, NJ, USA, 2003; pp. 133-142.
30. Google Developers. Manifest V3 Migration Guide. Available online: https://developer.chrome.com/docs/extensions/mv3/intro/ (accessed on 15 February 2025).
31. MITRE Corporation. Common Vulnerabilities and Exposures (CVE). Available online: https://cve.mitre.org/ (accessed on 1 November 2024).
32. International Organization for Standardization. ISO/IEC 42001:2023 Artificial Intelligence—Management System. Available online: https://www.iso.org/standard/81230.html (accessed on 13 February 2025).
33. National Institute of Standards and Technology. AI Risk Management Framework (AI RMF 1.0). Available online: https://www.nist.gov/itl/ai-risk-management-framework (accessed on 13 February 2025).
34. European Commission. EU Artificial Intelligence Act. Available online: https://artificialintelligenceact.eu/ (accessed on 13 February 2025).
35. Amazon Web Services. Amazon S3—Cloud Object Storage. Available online: https://aws.amazon.com/s3/ (accessed on 13 February 2025).
36. Amazon Web Services. AWS Identity and Access Management (IAM). Available online: https://aws.amazon.com/iam/ (accessed on 13 February 2025).
37. Amazon Web Services. Amazon EC2—Cloud Compute Capacity. Available online: https://aws.amazon.com/ec2/ (accessed on 23 February 2025).

 doi:10.20944/preprints202602.0475.v1

38. Project Jupyter. JupyterLab Documentation. Available online: https://jupyter.org/ (accessed on 13 February 2025).
39. Mozilla Developer Network. POST—HTTP Methods. Available online: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST (accessed on 13 February 2025).
40. Mozilla Developer Network. Introduction to the DOM. Available online: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (accessed on 15 February 2025).