IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

WEB INTEGRITY SHIELD: REAL TIME AI BASED PHISHING DETECTION SYSTEM

R Sai Krishna[1], and N Dinesh[2]

[1,2] *Students, Department of Emerging Technologies (CSE-AI&ML), Mahatma Gandhi Institute of Technology, Hyderabad, India.*

Abstract: Phishing attacks remain one of the most persistent and evolving threats to internet security, causing significant financial losses and compromising sensitive data worldwide. To address this challenge, Web Integrity Shield presents an intelligent, real-time phishing detection system aimed at safeguarding users by proactively identifying malicious websites before harm occurs. By combining traditional URL-based heuristics with advanced machine learning, the system provides a proactive layer of defense that evolves alongside emerging phishing tactics.

The proposed solution employs hybrid AI architecture integrating multiple technologies for accuracy and scalability. Initially, a machine learning model is trained in Python using a comprehensive dataset containing URL-based heuristic features. To further enhance detection, the system leverages Selenium WebDriver for automated, headless browser analysis within a secure, sandboxed backend. This enables dynamic extraction of features from the website's DOM structure, JavaScript execution patterns, and visual layout. The enriched feature set is then processed by the trained model, serialized into ONNX format, and deployed through a Java Spring Boot backend. A lightweight JavaScript browser extension functions as the client, performing quick feature extraction and querying the backend API for instant predictions.

This modular architecture ensures scalability, high accuracy, and separation of data science and deployment. The system delivers real-time protection against evolving phishing threats while remaining efficient and user-friendly.

Index Terms - Artificial Intelligence, Machine Learning, Phishing Detection, Cybersecurity, Web Security, Feature Extraction, Ensemble Learning, Real-Time Systems, Browser Extensions, ONNX Runtime, Selenium WebDriver

I. INTRODUCTION

Phishing attacks have emerged as one of the most persistent and evolving threats to internet security, leading to widespread financial loss and data breaches. With attackers increasingly creating highly convincing replicas of legitimate websites, users often struggle to differentiate between genuine and malicious platforms.

To address these issues, **Web Integrity Shield: A Real-Time AI-Based Phishing Detection System** is proposed as proactive and intelligent solution. This approach that combines URL-based heuristic analysis with real-time content inspection to accurately identify phishing websites. It is designed to

provide instant, reliable predictions and alert users before they interact with potentially harmful content, ensuring a safer browsing experience. The proposed system adopts a scalable and modular architecture.

It integrates a machine learning model trained in Python and deployed using ONNX within a Java Spring Boot backend, enabling fast and efficient inference. Additionally, Selenium-based sandboxed analysis is used for deeper inspection of suspicious websites, while a lightweight browser extension facilitates real-time interaction with users. This integrated approach ensures improved accuracy, adaptability, and protection against evolving cyber threats.

A. Motivation

The motivation behind Web Integrity Shield is to make internet security more accessible and user-centric. User often struggle to identify sophisticated phishing attacks due to limited technical awareness. Traditional security tools fail to keep pace with evolving and the system integrates machine learning, backend processing, and real-time content analysis for accurate detection and more reliable browsing. It aims to provide reliable, proactive protection and a safer browsing experience for all users.

B. Problem Definition

Phishing attacks are increasingly complex, making them difficult to detect using traditional methods. Existing systems rely on blacklists and simple heuristics, which fail against dynamic and zero-day attacks. The problem is the absence of a scalable system that combines fast client-side analysis with deep backend inspection to provide accurate real-time detection.

C. Existing System

Current systems rely on static blacklists or basic URL analysis, which are ineffective against new attacks. They lack dynamic content analysis and fail to examine DOM structures or JavaScript behavior. Additionally, most systems do not integrate client-side speed with backend processing, resulting in incomplete protection. These limitations make them vulnerable to zero-day attacks and advanced phishing techniques. They also struggle to adapt to evolving threat patterns due to limited scalability and update mechanisms. As a result, users remain exposed to potential data theft and financial risks despite existing security measures.

D. Proposed System

The proposed system uses a hybrid AI approach that combines a browser extension for real-time feature extraction, a machine learning model for prediction, and Selenium-based deep content analysis. The model is trained in Python, converted to ONNX format, and deployed in a Java Spring Boot backend to ensure efficient integration. This architecture enables fast inference, scalability, and seamless interaction between components. It also supports real-time detection with minimal latency while maintaining high accuracy. Additionally, the modular design allows easy updates and integration of improved models without affecting the overall system.

II. LITERATURE SURVEY

Potential research on phishing detection and web security is discussed in the following paragraphs. Researchers have explored various techniques to identify malicious websites, ranging from static blacklist methods to advanced machine learning models, aiming to achieve accurate, real-time detection and enhance user security.

Table I: Highlights selected literature on this topic

S.No	Year	Author	Title	Methodology
1	2020	J. S. Rao, A. Kumar, and S. Verma	Heuristic- Based Phishing Detection	Uses traditional static blacklists and rule- based heuristics for identifying malicious sites
2	2021	M. Ahmed and A. N. Al- Marridi	PhishGuard: A Machine Learning Approach for Detecting Phishing	Considers the URL string itself; trains models on features like URL length, keyword counts.
3	2021	G. S. Bhat and L. Tamil	Analyzing HTML Content for Phishing Classification	Performs static analysis of webpage HTML and DOM
4	2022	Dr. R.Srinivasa Rao, Dr.Purnachandra	Detecting Phishing Websites using CNN	Uses CNN-based visual analysis
5	2022	K. Gaurav and A. Kumar	A Lightweight Phishing Detection Browser Extension	Audits diagnostic tooperforms all feature extraction
6	2023	Ahmad Ayid Ahmad and Huseyin Polat	Phishing Detection Using LSTM	Applies a deep learning (LSTM) model to treat the URL string as a sequence of characters.
7	2023	Harshit Gupta and Lakshya Dahiya	Real-Time PhishNet using Scikit- learn and Flask	EvauatesPhishing Detection using a random forest model.
8	2024	Rayan Alanazi	A Hybrid Model for Robust Phishing Detection	Reviews ML algorithm and proposes a hybrid filter

III. METHODOLOGY

A. Dataset Collection

The dataset combines phishing and legitimate URLs, structured with features extracted from each webpage. It includes lexical, host-based, and page-based attributes to capture diverse web characteristics. Data is sourced from platforms like PhishTank, OpenPhish, and Tranco for reliability and relevance. A binary label (1: phishing, 0: legitimate) enables accurate supervised learning and generalization.

B. Data Preprocessing

The system uses a comprehensive dataset with URL-based heuristics, which is preprocessed to remove noise and handle missing values while ensuring consistency. Raw inputs such as URLs and DOM data are transformed into standardized numerical features for accurate analysis. The preprocessing pipeline includes label encoding and normalization to make the data suitable for machine learning models.

Additionally, client-side feature extraction mirrors the training logic to maintain consistency and ensure reliable predictions.

C. Model Selection

The system uses machine learning models such as Logistic Regression, SVM, and ensemble methods to detect malicious websites using URL-based features, trained and optimized in Python for high accuracy, precision, and recall. Advanced techniques like ensemble learning and hyperparameter tuning improve performance, with evaluation on unseen data ensuring real-world generalization. An ensemble model such as Random Forest or Gradient Boosting is selected for its strong performance on tabular data and ability to prevent overfitting. The model is further optimized using metrics like F1-score and deployed efficiently via ONNX within a scalable backend for reliable phishing detection.

D. Backend Development

The backend is deployed using a high- performance Java Spring Boot Framework to handle API requests, run prediction models, and manage Selenium-based analysis. It is designed to be modular and scalable, ensuring seamless integration between the ONNX model, browser extension and analyse environment. A Secure Rest API enables safe communication and decouples front-end interaction from back-end processing. Robust error handling ensures reliabilityby managing valid inputs and system failures effectively.

E. Frontend Development

The frontend is implemented as a lightweight JavaScript browser extension focused on simplicity and usability for users of all technical levels. It uses standard web technologies (JavaScript, HTML5, CSS3) to create an interactive interface with real-time alert mechanisms. The extension performs on-the-fly feature extraction and instantly displays security verdicts to warn users before sensitive input. It acts as a user-friendly client that communicates with the backend while preventing data theft and fraud.

F. Hybrid AI Integration

The system integrates a Hybrid AI Detection System by training models in Python and deploying them using the ONNX format within a high-performance Java Spring Boot backend for real-time predictions. This ensures fast and accurate results. The backend also incorporates Selenium WebDriver for deeper analysis. Combined with a lightweight browser extension, this architecture enables a scalable, robust, and high-performance user protection system.

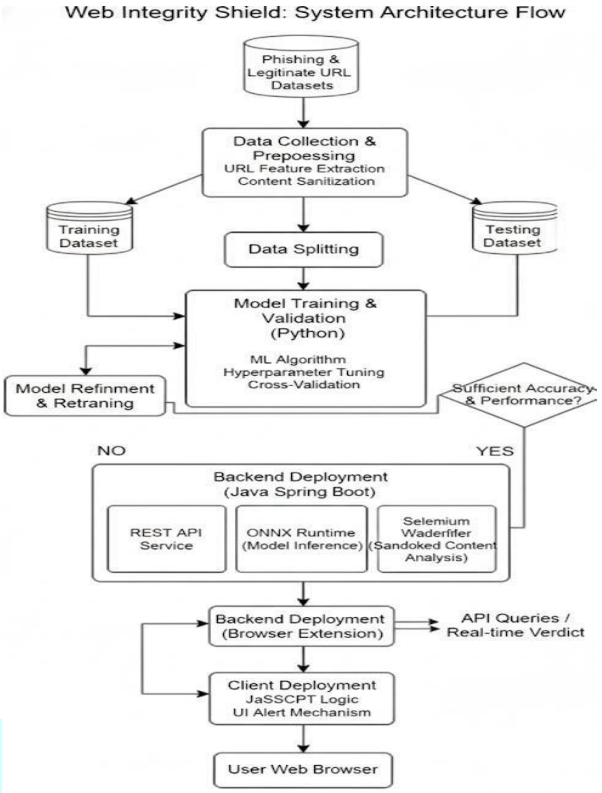
G. Deployment

The system is deployed with a cloud-based Spring Boot backend for scalable, secure real-time processing. The browser extension is distributed via official stores for easy access. Backend setup includes Docker-based Selenium and optimized APIs. Monitoring and updates ensure performance and evolving threat handling.

IV. SYSTEM DESIGN

A. Dataflow Diagram

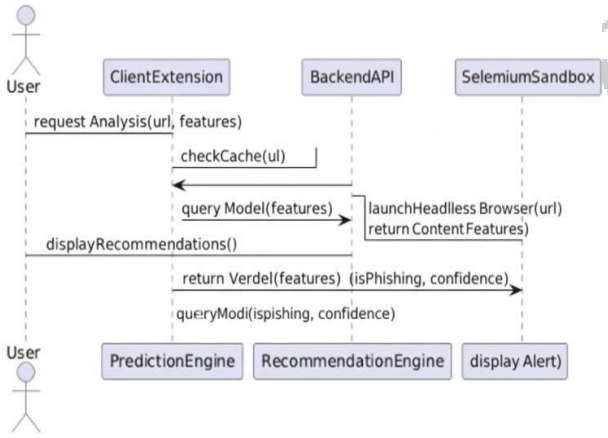
Figure 1 outlines the workflow of the Web Integrity Shield system, beginning with dataset collection, preprocessing, and feature extraction. The processed data is used for model training, validation, and refinement before deployment using a Java Spring Boot backend integrated with ONNX Runtime and Selenium for analysis. A browser extension then interacts with the backend to deliver real-time phishing detection and user alerts.



[Fig. 1: Dataflow Diagram]

B. Sequence Diagram

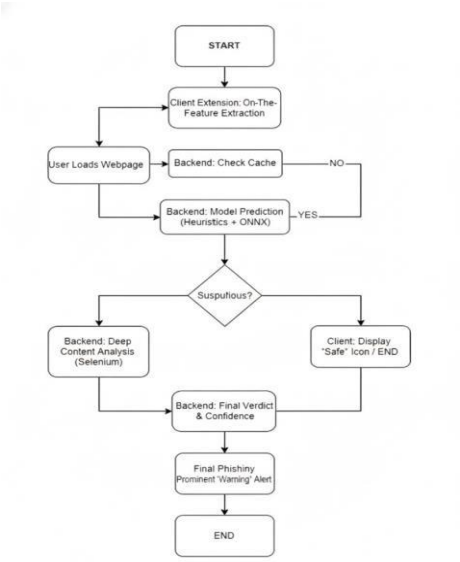
Figure 2 visualizes illustrates the interaction flow starting from the user request, where the browser extension checks the cache and forwards extracted features to the backend API for analysis. The backend processes the request using the prediction model and may trigger Selenium-based deep content analysis for suspicious URLs. Finally, the system returns the prediction results and recommendations, which are displayed to the user through alerts in the browser extension.



[Fig. 2: Sequence Diagram]

C. Workflow Diagram

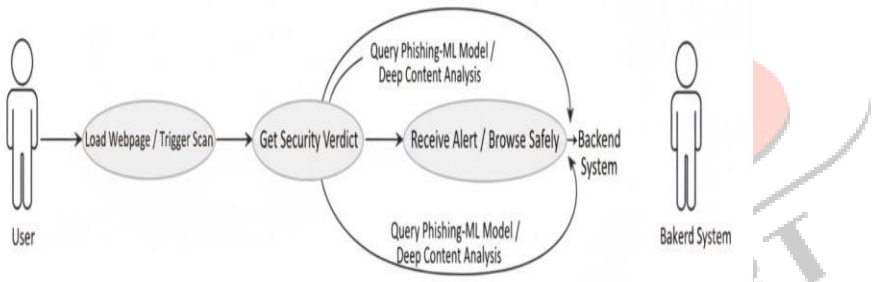
Figure 3 represents the process starting when a user loads a webpage, triggering feature extraction and backend model prediction. If the URL is identified as suspicious, the system performs deep content analysis using Selenium; otherwise, a safe indication is displayed. Finally, the backend generates a verdict with confidence and presents a phishing warning alert if necessary.



[Fig. 3: Workflow Diagram]

D. Use Case Diagram

Figure 3.18 illustrates how the web application can be used by a potential user and how easy it is to access and handle the prediction.



[Fig. 4: Usecase Diagram]

V. TESTING AND RESULTS

A. Testing

A thorough testing process was conducted on various aspects of the Phishing Detection system:

•Functional Testing: Verified real-time URL classification, correct risk scoring, and triggering of Level 2 analysis with end-to-end pipeline validation.

•Unit Testing: Validated individual backend and analysis components, including feature extraction, model inference, and security checks.

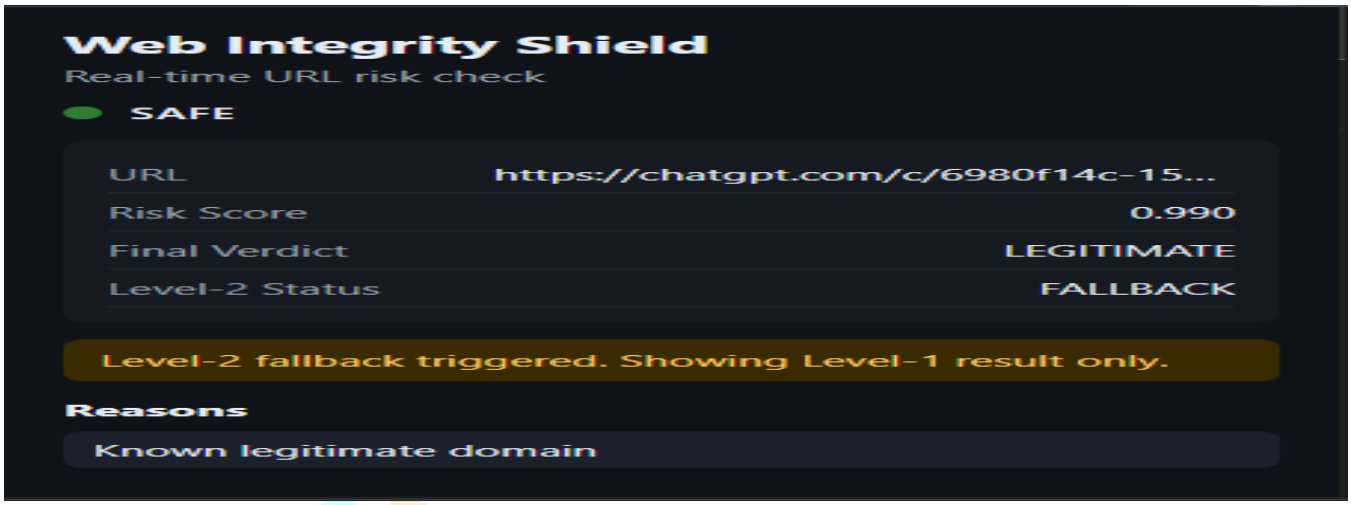
•Integration Testing: Ensured seamless communication across browser extension, backend, and Level 2 service with proper fallback handling.

•UI Testing: Confirmed clear display of results, alerts, and usability of the browser extension under various scenarios.

•Data Validation Testing: Ensured consistency, correctness, and compatibility of feature extraction and response data across the system.

•Performance Testing: Evaluated system responsiveness, scalability, and stability under concurrent load with acceptable latency.

B. Result



[Fig. 5: Application Interface]

As shown in Fig. 5, The figure depicts the Web Integrity Shield browser extension identifying a legitimate website, displaying a low-risk score along with a "LEGITIMATE" verdict. This demonstrates the system's ability to correctly classify safe URLs using the Level 1 model without triggering deep analysis.

[Fig. 6: Weather Input and Analysis Results Dashboard]

Fig. 6 depicts the figure depicts the browser extension flagging a suspicious or phishing URL, showing a high-risk score and a warning or "PHISHING/SUSPICIOUS" verdict. It also includes Level 2 analysis results and reasons, illustrating the system's multi-layer detection capability for enhanced security.

VI. CONCLUSION

The Web Integrity Shield: Real-time Phishing Detection System will be approached as a modern, scalable, and technology-driven solution to today's evolving cybersecurity challenges. By focusing on the persistent and rapidly changing nature of phishing attacks, including zero-day threatsthe system aims to proactively strengthen user security, prevent data theft, and minimize financial fraud. The approach emphasizes accessibility and reliability so that users with any level of technical expertise can benefit from instant, accurate, and automated protection while browsing the internet.

To achieve this, the project adopts a clear, structured methodology built around a Hybrid AI Detection Architecture. The workflow begins with a Python-trained machine learning model using comprehensive URL-based heuristic features, followed by deeper real-time content analysis through Selenium WebDriver for DOM-level, JavaScript, and visual-layout inspection. This enriched feature pipeline is then delivered through a high-performance Java Spring Boot backend, where the trained model is deployed in ONNX format to ensure fast inference and easy updates. A lightweight JavaScript browser extension serves as the user-facing client, performing quick feature extraction and querying the backend for real-time verdicts. This modular and decoupled architecture not only supports efficient development but also ensures long-term scalability for future enhancements—such as expanded threat types, multilingual interfaces, integration with threat intelligence feeds, and more advanced user-centric security features.

Overall, this project approach demonstrates how a hybrid AI strategy can bridge the gap between traditional detection methods and modern cybersecurity demands. By combining intelligent automation, scalable backend design, and user-first principles, Web Integrity Shield aims to deliver a reliable, adaptive, and forward-looking security solution for safe internet use.

The future scope of Web Integrity Shield focuses on enhancing detection accuracy and expanding threat coverage through advanced AI techniques. The system can evolve by incorporating deep learning models such as LSTM or transformer-based architectures to analyze sequential URL patterns and webpage content more effectively. Integration with real-time threat intelligence feeds (e.g., phishing databases, domain reputation APIs) can further improve detection of zero-day attacks. Additionally, expanding feature extraction to include visual similarity detection (e.g., identifying cloned websites using screenshot comparison) and JavaScript behavior analysis will significantly strengthen the Level-2 deep analysis layer.

From a system and product perspective, the project can be extended into a fully scalable, cloud-deployed security platform with support for large-scale concurrent users. Features such as user-specific risk profiling, centralized dashboards, analytics reporting, and alert systems can be added for enterprise use. The browser extension can be enhanced with multilingual support, customizable security levels, and real-time user feedback loops to continuously improve the model. Furthermore, integration with mobile browsers and development of APIs for third-party applications can broaden its applicability, making Web Integrity Shield a comprehensive, adaptive cybersecurity solution.

ACKNOWLEDGMENT

THE AUTHORS WOULD LIKE TO EXPRESS THEIR SINCERE GRATITUDE TO MS. P. M. TULASI AND MS. S. ASHA JYOTHI, ASSISTANT PROFESSORS IN THE DEPARTMENT OF EMERGING TECHNOLOGIES, MAHATMA GANDHI INSTITUTE OF TECHNOLOGY, HYDERABAD, FOR THEIR VALUABLE GUIDANCE, CONTINUOUS SUPPORT, AND ENCOURAGEMENT THROUGH OUT THE DEVELOPMENT OF THIS PROJECT. WE ALSO EXTEND

OUR HEARTFELT THANKS TO THE DEPARTMENT OF EMERGING TECHNOLOGIES (CSE-AI &ML), MAHATMA GANDHI INSTITUTE OF TECHNOLOGY, HYDERABAD, FOR PROVIDING THE NECESSARY RESOURCES AND A CONDUCIVE ENVIRONMENT TO CARRY OUT THIS RESEARCH WORK.

FINALLY, WE WOULD LIKE TO THANK OUR FAMILY AND FRIENDS FOR THEIR CONSTANT MOTIVATION AND SUPPORT.

REFERENCES

[1] Murad, A., & Rahimi, A. (2024), 'PhishGuard: Machine Learning-Powered Phishing URL Detection', *International Conference on Computer and Applications (ICCA)*.

[2] Gupta, B, & Kumaraguru P (2021), 'Heuristic based Approach for Phishing Site Detection Using URL Features', *Journal of Cyber Security and Mobility*.

[3] Rao Y & Li Y. (2020) 'Detection Method for Phishing Web Page Using DOM-Based Doc2Vec Model', *International Journal of Information and Education Technology*.

[4] Rao, G. S & Ali S. S. (2019) 'A Computer Vision Technique to Detect Phishing Attacks', *International Conference on Communication and Signal Processing*.

[5] Kumar, S & Das S. (2025) 'Browser Extension for Phishing Website Detection Using Machine Learning', *AIJR Proceedings*.

[6] Ferrara, E (2024) 'PhishLang: A Lightweight, Client-Side Phishing Detection Framework using MobileBERT for Real-Time, Explainable Threat Mitigation'.

[7] Singh, A (2023) 'A Hybrid Framework for Real-Time Phishing Detection Using URL, Content, and DOM Features with Interpretable ML', *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*.

[8] K S& L A (2025) 'Machine Learning and Neural Networks for Phishing Detection: A Systematic Review (2017–2024)', *MDPI, Electronics*.