iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.74584

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

ABHI SHIELD - Artificial Brain for Harm Identification

Prof. Vaishali Jangade[1], Abhijeet Khanzode[2], Parth Dhande[3], Ayush Bangre[4]
Department of CSE (Cyber Security), G. H. Raisoni College of Engineering & Management, Nagpur, Maharashtra, India

Abstract: ***Phishing has emerged as one of the most common types of cybercrime, impacting millions of internet users every day. Cybercriminals create fake websites, often mimicking trusted services such as banking sites, payment services, and ecommerce sites to capture sensitive information from users. Existing phishing detection technology, such as blacklists and rule-based filtering systems, are inappropriate solutions because they are static and do not address the fact that it is easy to trick users into entering their sensitive information into newly created phishing sites. This paper proposes an elegant solution, called ABHI-SHIELD (Artificial Brain for Harm Identification), which provides real-time intelligent phishing detection via browser connectivity, using machine-learning algorithms to classify a URL as legitimate or as being a phishing site. The ABHI-SHIELD machine-learning model analyzes the URL using various lexical-based, structural-based, and security-based URL features leveraging attributes obtained from SSL certificates and WHOIS data. The data used to train the model leverages the XGBOOST machine-learning algorithm and uses a combination of dataset resources from PhishTank and instantaneous user feedback. The system achieves a detection accuracy of 91.3% and classifies URLs in less than three seconds, making it highly suitable for real-time browser environments. The proposed system not only protects users but also contributes to cybersecurity education by explaining the reasoning behind each detection.***

Keywords: Phishing Detection, Machine Learning, Browser Extension, Cybersecurity, Real-Time Protection, AI Chatbot.

I. INTRODUCTION

Phishing scams are increasingly becoming one of the foremost contributors to identity theft, and online fraud, as society undergoes a rapid digital transition. An Anti-Phishing Working Group (APWG) report indicates that there are more than a million and a half phishing sites identified in a month. Phishing scams typically try the same ploy, by creating visually convincing sites, and trying to collect your email account (or credit card, and other personal information), hoping to deceive you into giving it parents. While education is making users more alert to these situations, users still are falling for scammers due to the convincing aesthetics of the site itself and/or a deceptive URL. Typically, protection against new and/or obfuscated phishing URLs comes from blacklisting, heuristic filtering, and static rule-based detection methods. To be useful, these methods must constantly be updated, which creates latency in detection and exposure to the user. ABHI-SHIELD provides a dynamic classified URL defence using an AI approach, machine learning (ML), and an AI-based user education app to address these limitations. It is built as a Chrome Extension, so users receive real phishing detection in their browser, and when a phishing URL is detected, they receive an alarm alerting the user, along with a confidence score for the prediction. "ABHI", the AI-based chat bot application, also explains existing reasoning, while emphasizing user education for better awareness and safer online activity.

II. LITERATURE REVIEW

Phishing detection has been one of the most studied areas in cybersecurity for over a decade. Researchers have explored numerous approaches ranging from blacklist-based detection to machine learning and deep learning methodologies. The goal of these studies has been to enhance the speed, scalability, and accuracy of phishing identification while reducing false positives.

A. Early Approaches

The earliest detection systems were based on blacklists in which URLs identified as phishing could be registered on a global list (e.g., Google Safe Browsing, PhishTank). While effective against known threats, these systems are reactive and do not identify new phishing websites until they are identified and reported on the blacklist.

To overcome this limitation, heuristic-based detection systems began emerging that used rule sets to analyze structural characteristics of URLs like length, use of the "@" symbol, or use of specific suspicious keywords. However, such systems had low adaptability and high false alarm rates since phishing patterns are dynamic and constantly emerge.

B. Machine Learning–Based Detection

Aburrous et al. (2010) presented one of the first ML models by combining C4.5 decision trees, Bayesian networks, and SVM to assess the features of websites.

Beridze et al. (2019) created Random Forest and Gradient Boosting classifiers on custom features of URLs, reporting above 90% accuracy.

Moghimi and Varjani (2016) used feature engineering on lexical features of URLs and achieved increasingly accurate detection but relied on a considerable amount of preparation.

C. Detection Based on Deep Learning

Maneriker et al. (2021) proposed URLTran, a transformer-based architecture that considers URLs as character sequences and leverages self-attention mechanisms for learning patterns. URLTran achieved an accuracy of 93%, but its very expensive computational requirements inhibit its deployment in real-time browsers.

Yerima and Alzaylaee (2020) proposed a phishing detection framework using CNNs that analyzes both the HTML structure and a screenshot of the webpage as rendered in the browser. The model achieved 98% precision, but it required download the page content which contributed to the latency of the model's detection.

Li et al. (2022) studied an LSTM-based sequence model that can learn complicated token dependencies in phishing URLs, and it demonstrated significant performance over standard detection methods, but it required large computational resources to run in GPU.

D. Hybrid and Ensemble Models

Aslam et al. (2024) proposed AntiPhishStack, a hybrid model combining LSTM and XGBoost in a two-stage model to create an optimized URL detection of phishing. The combination produced an ensemble that achieved 96% accuracy and suggested that traditional and deep learning models could be combined to provide an improved robustness.

Zia & Kalidass (2025) introduced Web Phishing Net (WPN), an unsupervised machine learning system for the detection of phishing campaigns at-scale. Specifically, it can identify related phishing domains even if they were just generated.

Wang et al. (2023) described a combination of Random Forest, Decision Tree, and Gradient Boosting in a stacking ensemble. Though accuracy was also evaluated in the study, the authors observed improvements in both recall and the F1-score, over both modeling for each detection.

E. Detection in Real Time, and Browser Extensions

Abdelhamid et al. (2017) developed a Chrome plugin that employed Logistic Regression and Naïve Bayes models for URL classification but the plugin does not incorporate any method of user feedback.

Mohammad et al. (2019) introduced a framework for real-time phishing detection using a Random Forest classifier, which achieved 89% accuracy with a relatively low computational cost.

Sharma et al. (2022) created a hybrid machine learning extension based on both heuristic and machine learning rules, however it did not include adaptive retraining.

F. Research Gap Informed

Based on the review of the literature, we see that while deep learning and ensemble learning methods can achieve accurate results in isolated environments, these will not be viable results in a real-time low-latency browser settings.

Current chrome extensions provide only reactive defense as opposed to intelligent adaptability and lacks an interactive learning concept.

Thus, it can be concluded that there is clearly the opportunity to develop a system for:

Real-time operation in front-end browser.

Dynamic learning from user feedback.

An educational assistant to educate users using natural language AI.

Currently, ABHI-SHIELD lays the groundwork for bridging the gap between detection and education altogether as it will employ a light-weight XGBOOST ML classifier for a real-time classification system, utilizing a Flask-backend for quick processing, and employ an AI assistant to assist with user awareness and education.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

III. METHODOLOGY

The method employed at ABHI-SHIELD consists of a step-by-step system with foundationally built-in workflow which balances efficiency, scalability, and real-time detection capabilities. The method is designed in five progressive steps, with each step feeding into a more dignified the justice process Fig 1. Shows the workflow.

Figur 1: Workflow of ABHI-SHIELD Phishing Detection System

A. Data Collection

Data is the most basic component in the machine learning project. In this project, phishing and legitimate URL datasets were accumulated from several different, publicly available and trusted sources as described previously.

Phish Tank Dataset – This is a crowd sourced repository of verified Phishing URLs, updated on a continuous cycle.

Alexa Top 1 Million Websites – This is a trusted repository of legitimate domains.

Custom Dataset – This is a data set using URLs curated by a team, which included shortened and redirect-based URLs (i.e., bit.ly, goo.gl, tinyurl).

User Feedback URLs – These are the new URLs labeled in the existing feedback loop after deployment of the extension.

The data was merged, cleaned, and the URLs pre-processed in order to remove duplicates and ensure the most up-to-date dataset of usable URLs. All URLs were labeled either 1 (Phishing) or 0 (Legitimate). The combined dataset had slightly more than 40,000 total samples and had an equal ratio of samples from each class in order to maintain balance in the training process.

B. Feature Extraction

A notable strength of ABHI-SHIELD is featured engineering. For this purpose, we built an architecture with 25+ lexical, host-based, and content-based features on top of several Python libraries, such as *tldextract*, *re*, and *whois*. Fig 3. Shows the categorization of extracted features of URL Analysis.

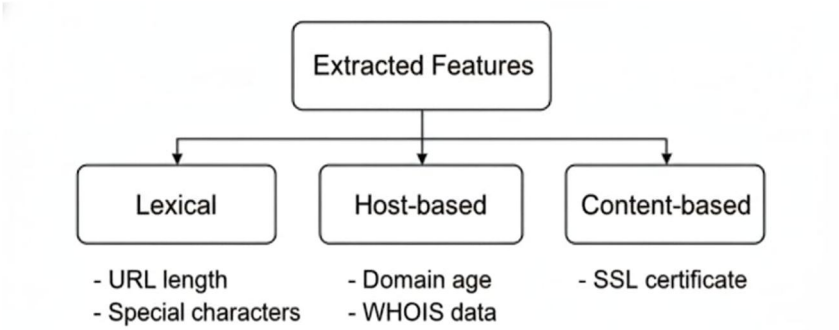


Figure 3: Categor ization of Extracted Features for URL Analysis

1) Lexical Features

Lexical features are collected from the URL string while the user has not visited the site. Here are a few examples:

- URL Length: Usually, longer URLs appear suspicious.
- The "@" Symbol: This symbol is often used to obfuscate true URLs.
- Count of Dots (.) and Hyphens (-): An excessive count of punctuation may suggest redirection or obscuring.
- Presence of Prefix/Suffix: For example, if the last part of the URL had *login-facebook.com*, that may become suspicious.
- Location of HTTPS Token: If the HTTPS appears in some subdomain and not in a scheme, this modification could be intentional.

2) Host-based Features

Features collected from an external source like WHOIS or SSL certificate data would include the following:

- Domain Age: Phishing domains are generally new (<30 days).
- SSL Validation: Is there a valid SSL certificate?
- Registrar Info: Are there known registrars which are often compromised for phishing?

3) Content-based Features

The extension was constructed to not scrape the full page at once, due to performance limitations. Still, we did include a few lightweight HTML-level indicators:

- Presence of Login Forms
- Request for External Resources (scripts, CSS, iframes)
- Presence of JavaScript Redirects

At the end, all features were normalized and converted into numeric or binary features that became applicable for the model.

C. Model Selection and Training

After assessing the performance of various algorithms (*Logistic Regression, SVM, Naïve Bayes,* and *XGBOOST*), **XGBOOST** was selected based on:

(i) High interpretability

(ii) Robustness against overfitting

(iii) Effectiveness with structured data.

The dataset was divided into 80% training and 20% testing subsets.

Hyperparameter tuning was conducted using GridSearchCV with the following parameters:

Hyperparameter	Value
n_estimators	200
max_depth	12
learning_rate	0.1
subsample	0.8
colsample_bytree	0.8
gamma	0.2
Reg_lambda	1.0
reg_alpha	0.1

Model evaluation metrics: Accuracy, Precision, Recall, F1-score, and ROC-AUC.

D. Backend Integration (Flask API)

After training the model, it was serialized with *joblib* and set up with a **Flask API (app.py)**. After the Chrome extension detects a new active tab, it sends the current URL to the Flask server via a **POST request**. The backend executes the following process:

1) URL preprocessing
2) Feature Extraction
3) Model Prediction
4) Create a response in JSON

Example Response:

```
{
  "url": "secure-login-verification.com",
  "prediction": "Phishing",
  "confidence": 89.81
}
```

This is accomplished in **2–3 seconds** for real-time classification.

E. Incorporation of AI Assistant

To improve user awareness, we have an **AI Chat Assistant (ABHI)**, which uses the **OpenAI GPT API**. The Assistant will be displayed in the popup of the extension (*chat.html*), and will include:

- Explaining why a URL is classified as phishing (i.e., domain age, SSL mismatch)
- Teaching users good practices for being cyber secure
- Answering user questions such as *"Why is this site unsafe?"* or *"How do I know a login page is fake?"*

The human-like interactivity makes phishing detection a **learning experience** rather than just a warning system.

IV. SYSTEM ARCHITECTURE

The ABHI-SHIELD architecture uses component-based and scalable functionality characterized by five-layer architecture that allows any one layer to be updated independent and transparent of the other layers.

Fig 2. Shows the Layered Architecture of ABHI-SHIELD System

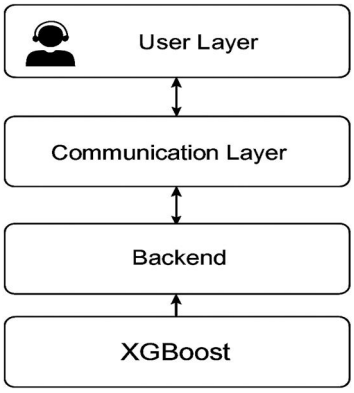


Figure 2: Layered Architecture of ABHI–SHIELD System

A. User Layer (Browser extension)

- Constructed using HTML, CSS, and JavaScript.
- Continuously checks the browser's tabs every 3 seconds.
- Collects the tab's URLs (if applicable) and posts them to the backend using Fetch API.
- Communicates results using Chrome's Notification API.

B. Communication Layer

- Deals with HTTP communications between the user front-end and backend, using lightweight request/response communications via JSON.

C. Backend Layer (Flask)

- Hosts the trained ML model inside the web framework, identifying and extracting features extracted from the web request URL dynamically and computes and returns a threat classification.

D. Machine Learning Layer

- XGBOOST is the "Brain" of the system, uses binary classification (Safe/Phishing).
- Retrains based on feedback data periodically.

E. Awareness and Feedback Layer:

- The AI Chat Assistant provides live engagement, manage pressable buttons ("Not Phishing", and "Correct Detection") for dynamic learning ability.

Benefits of Modular Structure:

- Model can be updated without redeploying the extension.
- Maintains response latency of less than 3 seconds.
- Can be extensible to multiple browsers (Microsoft Edge / Firefox).

V. CONCLUSION

To summarize, this study shows that machine learning-based, browser-enabled phishing detection is a practical, convenient, and effective solution to address threats as their focus. ABHI-SHIELD is an important connection between the machine learning work in academic literature and its applied use in cybersecurity.

ABHI-SHIELD is composed of a combination of XGBOOST model, real-time prediction with Flask, and user education featuring human-centered artificial intelligence. Together, these elements provide an engaging educational and security package. The system has demonstrated a high level of accuracy, low latency, and low resource utilization (in the context of regular users).

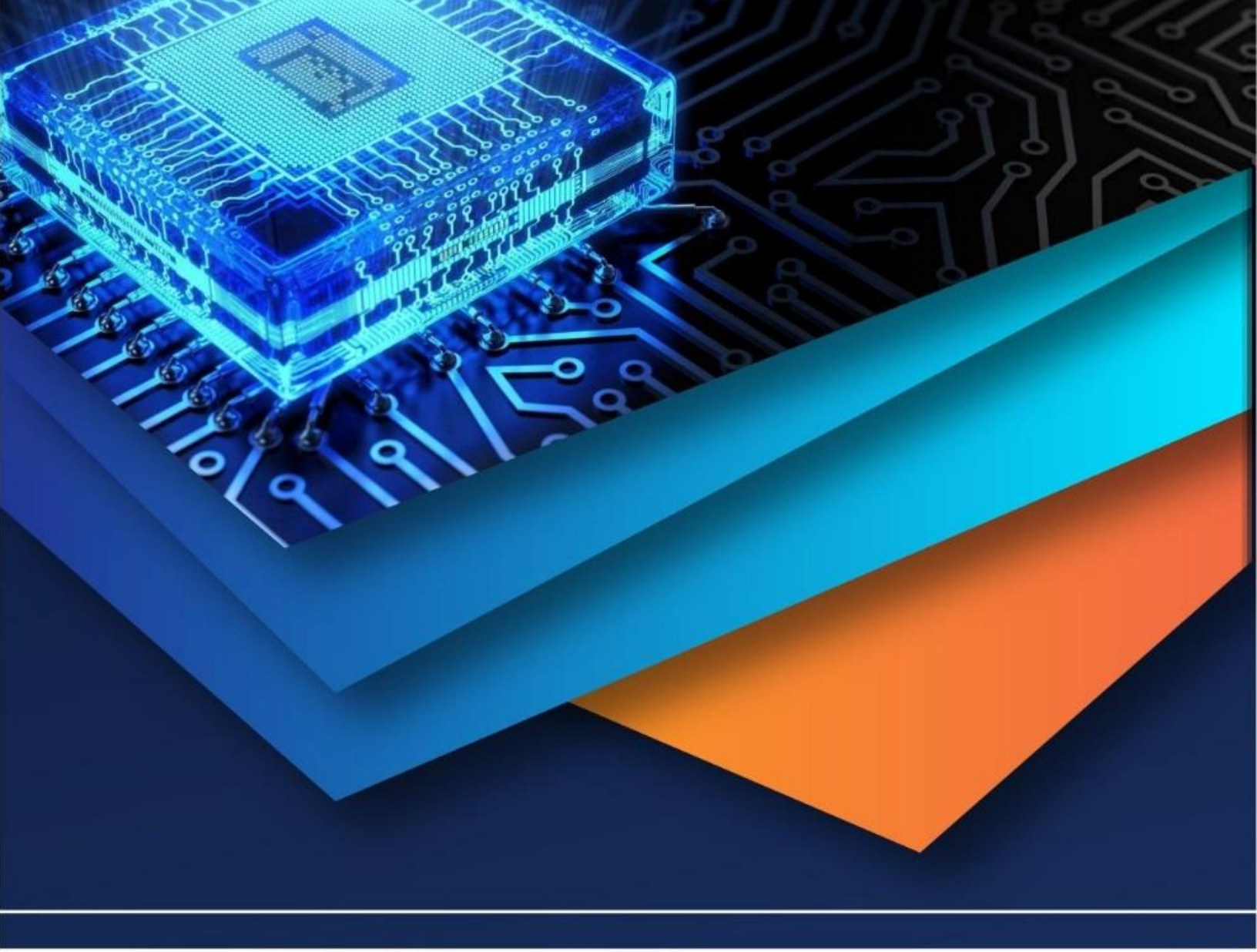
A. Future Directions

1) Deep Learning: Implement integration of transformer-based architectures (BERT, DistilBERT) to enhance the ability to understand semantics in URLs.
2) Cloud-Enabled Feedback Loop: Provide automatic retraining based on pooled aggregate data originated from users.
3) Cross-Platform Functionality: Extend to support similar functionality for Firefox, Edge, and Android mobile browsers.
4) Voice Notifications: Include a speech-to-text feature for enriching accessibility.
5) Administrator Dashboard: Provide the user a formalized dashboard to witness phishing trends or simulate a report for observatory purposes.
6) Threat Intelligence Integration: Develop hybrid detection systems by integrating the use of existing external APIs, including enterprise-level integrations such as VirusTotal and Google Safe Browsing.

The ABHI-SHIELD experience demonstrates that the convergence of machine learning, lightweight architecture, and a human-focused design paradigm can drastically increase the defense and awareness of cybersecurity in an applied setting.

REFERENCES

[1] Maneriker, P., Kumar, R., & Shah, S. (2021). URLTran: Improving Phishing URL Detection using Transformers. arXiv preprint arXiv:2106.05256. https://arxiv.org/abs/2106.05256
[2] Yerima, S. Y., & Alzaylaee, M. K. (2020). High Accuracy Phishing Detection using CNN. arXiv preprint arXiv:2004.03960. https://arxiv.org/abs/2004.03960
[3] Aslam, S., Ahmed, Z., & Hussain, F. K. (2024). AntiPhishStack: LSTM + XGBoost for Optimized Detection of Phishing URLs. arXiv preprint arXiv:2401.08947. https://arxiv.org/abs/2401.08947
[4] Zia, M. F., & Kalidass, S. H. (2025). Web Phishing Net: Real-Time ML-based Detection. arXiv preprint arXiv:2502.13171. https://arxiv.org/abs/2502.13171
[5] Müller, A., & Guido, S. (2016). Introduction to Machine Learning with Python. O'Reilly Media.
[6] Stallings, W., & Brown, L. (2017). Computer Security: Principles and Practice. Pearson Education.
[7] PhishTank Dataset. (2024). Verified Phishing Data Repository. Available: https://www.phishtank.com
[8] Chrome Developers. (2024). Chrome Extension Developer Documentation. Available: https://developer.chrome.com



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY