LBS Institute Of Technology For Women

AI-POWERED BROWSER EXTENSION FOR DETECTING PHISHING & MALICIOUS SCRIPT

OUR TEAM

- **Aditya ML (LBT23IT003)**
- **Anjum Aysha (LBT23IT013)**
- **Aparna S S (LBT23IT015)**
- **Nikitha S Nair (LBT23IT043)**

Department: *Information Technology*
College: *LBS Institute of Technology for Women*
University: *APJ Abdul Kalam Technological University*
Project Guide: *Prof. Raji P S*

Academic Year: *2025–2026*

INTRODUCTION

Phishing and malicious websites trick users into revealing logins and banking details. Existing browser and antivirus warnings often appear only after a page loads, allowing harmful scripts to run. As modern attacks target user behavior through fake links and deceptive pages rather than system flaws, this project provides earlier, stronger browser-level protection for students, professionals, shoppers, and organizations.

PROBLEM STATEMENT

- Existing phishing and malware detection systems usually alert users only after a suspicious website has already loaded, allowing malicious scripts or fake login forms to execute.

- Current systems lack early-stage detection that can warn users before accessing harmful links and often provide only generic alerts without explaining the reason for risk.

For example, a user may click on a phishing link and see a warning only after a fake login page appears, increasing the chance of credential theft.

OBJECTIVES

1. To design and develop an AI powered browser extension that can detect phishing and malicious websites in real time.
2. To analyze website behavior by monitoring URLs, JavaScript execution, and DOM activities before and during page load.
3. To implement intelligent phishing detection using Python-based machine learning models.
4. To provide a dynamic website safety score with clear risk explanations for improved user awareness.

LITERATURE SURVEY

BASE PAPER OVERVIEW & METHODOLOGY

Md. Redwan Ahmed, Md. Manowarul Islam, Md. Abu Layek (2024)

Phishing URL Detection using Comprehensive Feature Extraction and Machine Learning Techniques

(IEEE CS BDC Symposium 2024)

Objective of the Paper

- Focuses on enhancing phishing URL detection through a combination of feature extraction and machine learning techniques.
- To extract and utilize comprehensive static URL and domain features.
- To integrate reputation-based APIs (Open Page Rank, Google Safe Browsing, VirusTotal) for validation.

Methodology

- Collection of phishing and legitimate URLs from public datasets
- Uses external security APIs like Open Page Rank, Google Safe Browsing, and VirusTotal to check URL reputation and known threats.
- Extraction of static URL and domain features such as:
  - URL length
  - Special characters etc.
- Training multiple machine learning classifiers such as Decision Tree, Random Forest, and XGBoost.

Technologies & Protocols

- Programming Language: Python
- Libraries: Scikit-learn
- Data Source: PhishTank, OpenPhish
- Protocol Focus: HTTP/HTTPS URL structure analysis

Algorithm Used

- Uses supervised machine learning classification
- Trains Decision Tree, Random Forest, and XGBoost models
- Compares model performance to select the best classifier
- Decision Tree achieves the highest detection accuracy

Approach

- Collects phishing and legitimate URLs from public datasets
- Extracts static URL and domain-based features
- Validates URLs using security reputation APIs
- Classifies URLs as phishing or legitimate using ML models

Pros:

- High accuracy in phishing URL detection
- Effective use of comprehensive URL and domain features
- Employs reliable and well-known ML algorithms
- Computationally efficient compared to deep learning models
- Provides a strong baseline for phishing detection research

Cons:

- Relies only on static URL and domain analysis
- Does not inspect JavaScript or DOM behavior
- No real-time browser-level implementation
- Provides only binary output without user explanation

CONNECTIVITY

- Establishes a strong machine learning foundation for phishing detection
- Demonstrates effective URL and domain feature extraction
- Uses practical ML models such as Decision Tree, Random Forest, and XGBoost
- Shows the effectiveness of feature-based classification
- Highlights the limitation of static analysis approaches
- Indicates the need for real-time, behavior-aware phishing detection
- Suggests future focus on user-centric detection systems

RESEARCH GAP

- Uses only static URL and domain feature analysis
- Does not analyze dynamic JavaScript or DOM behavior
- Detection occurs after the URL is accessed
- Provides only binary results without explanations
- Not suitable for real-time, user-facing browser protection
- Not designed for lightweight, client-side deployment

S. Adolphine Shyni et al. (2025)
Phish Defender: Real-Time Detection of Phishing Websites Using a Browser Extension (Springer, 2025)

Objective

- To develop a real-time browser extension that detects phishing websites during browsing.
- To use heuristic rules, URL analysis, and domain reputation checks for phishing detection.
- To provide instant alerts to users about suspicious websites.

Methodology

- Input: Webpage URL and domain.
- Content filtering
- SSL certificate verification
- Heuristic rules applied to detect suspicious URL patterns.
- Domain reputation checked against known blacklists.
- Detection: Alerts triggered after the page loads if phishing is suspected.
- Output: User receives alert indicating whether the site is phishing or safe.

Protocols / Technologies

- Browser Extension APIs – to intercept navigation and show alerts.
- Heuristic Rule Engine – lightweight rule-based detection.
- Domain Reputation Services – check against blacklists and known phishing domains.
- JavaScript / DOM Inspection – minimal; mainly post-load URL checks.

Algorithm Used

- Heuristic-based detection for URL structure (suspicious characters, abnormal length).
- Domain reputation checks (lookup against blacklists).
- No complex ML or deep learning in this paper; purely rule-based for real-time efficiency.

Approach

- Browser-based detection: integrated as a real-time extension.
- Two main steps:
- Monitor URL and domain at runtime.
- Apply heuristic rules and domain reputation checks post-page load.
- User alert: only after page is loaded if the site is suspicious.

Pros

- Practical browser extension deployment.
- Lightweight and fast — no heavy ML or backend processing.
- Provides real-time phishing alerts during browsing.
- Efficient for common phishing patterns.

Cons

- Alerts appear only after the page loads; malicious scripts may already run.
- No pre-navigation warning to prevent accidental clicks.
- Does not analyze dynamic JavaScript or DOM behavior.
- Alerts are binary and rule-based, without explanations for the user.
- Limited defense against advanced phishing attacks using scripts or hidden forms.

CONNECTIVITY

- Base idea: browser extension for phishing detection.
- Our project builds upon it by:
- Adding pre-navigation URL warnings.
- Performing runtime JavaScript and DOM inspection for malicious behavior.
- Providing dynamic safety scores with explainable alerts.
- Using a lightweight ML model for better detection and adaptability.

RESEARCH GAP

- Lack of pre-navigation detection before the user clicks a suspicious link.
- Limited explainability in alerts.
- Detection primarily rule-based, not learning-based, reducing adaptability to new phishing techniques
- Detection occurs only after the webpage loads, allowing malicious scripts to execute before alerts.
- No pre-navigation or hover-based URL warning to prevent impulsive clicks.
- Dynamic JavaScript behavior is not analyzed, so advanced script-based phishing can bypass detection.

Web-Based Phishing URL Detection Model Using Deep Learning Optimization Techniques

(Kousik Barik, Sanjay Mishra, Raghini Mohan)

Objective

- Reviews and summarizes existing phishing detection techniques.
- EGSO-CNN deep learning model to improve detection accuracy and reduce false positives.
- Applies data preprocessing (StandardScaler) and feature extraction using VAE.
- Validates the proposed model on a different dataset and compares results with existing studies.

Methodology

- Data collection and Dataset Creation
- Preprocessing & Feature Extraction
- Deep Learning Model: EGSO-CNN
- Training & Evaluation

Protocols / Technologies

- HTTP/HTTPS-system checks whether a site uses HTTPS
- TLS/SSL (Transport Layer Security)
- Browser Extension Protocols (WebExtension APIs)
- Client-Side Processing Protocol.

Algorithm Used

- CNN – main phishing detection model
- EGSO (Enhanced Grid Search Optimization) – used to optimize CNN performance
- EGSO-CNN – the proposed hybrid model of the paper
- VAE– used for feature extraction
- StandardScaler – used for data preprocessing

Approach

- Collect web phishing dataset from trusted sources
- Preprocess data using StandardScaler
- Extract important features using Variational Autoencoder (VAE)
- Train deep learning models (CNN, LSTM, GRU)
- Optimize and classify using the proposed EGSO-CNN model

Pros

- Achieves very high accuracy (~99.4%) with reduced false positives.
- Demonstrates consistent performance across multiple datasets.
- Shows potential for real-world deployment, including browser-based phishing detection.

Cons

- Phishing techniques evolve rapidly.
- The model cannot detect phishing sites that use embedded objects (e.g., Flash or Java-based content).
- Detection of embedded malicious elements requires additional external tools.
- No real-world testing outcomes, or user feedback.

Connectivity

- Their approach validates that AI techniques are effective in identifying phishing URLs based on learned patterns.
- The proposed project addresses real-time usability and deployment limitations, as the referenced work operates mainly in an offline, dataset-based environment.
- Unlike the existing paper, the project performs runtime JavaScript and DOM behavior monitoring after page load to detect phishing activity beyond URL patterns.

Research Gap

- URL-Only Focus-model .
- Static, Offline Detection-approach is an offline classifier.
- Dataset Dependency and Evolution
- Limited Handling of Obfuscated Content-model does not consider embedded content (like encoded scripts or hidden forms).
- Lack of Browser-Level Integration

Real-Time Phishing URL Detection Framework Using Knowledge-Based and Learning Approaches (Taylor & Francis, 2024)

Objective

- To detect phishing URLs in real time while users browse the web.
- To combine rule-based checks and machine learning techniques for better accuracy.
- To accurately classify phishing and legitimate websites.
- To reduce phishing attacks and protect users from credential theft.

Methodology

- Collected and labeled a large dataset of phishing and legitimate URLs.
- Extracted relevant URL text patterns and features.
- Trained a knowledge-distilled ELECTRA deep learning model for phishing detection.
- Integrated the trained model into a browser extension.
- Intercepted URLs during browsing and classified them in real time.
- Displayed instant phishing warnings to users.

Protocols / Technologies

- HTTP/HTTPS protocols to monitor and analyze URLs during browsing.
- Browser extension technology to detect phishing links in real time.
- Machine learning (ELECTRA model) for classifying phishing and legitimate URLs.
- Backend processing system to handle heavy model computation.
- Efficiently Learning an Encoder that Classifies Token Replacements.

Algorithm Used

- .Uses rule-based checks to quickly identify suspicious URL patterns.
- Applies a deep learning ELECTRA model to classify URLs as phishing or legitimate.
- Uses knowledge distillation to make the ELECTRA model faster and lighter.
- Performs binary classification to decide whether a URL is safe or phishing.

Approach

- User clicks or enters a website URL.
- Browser extension captures the URL during browsing.
- URL is analyzed using rules and machine learning (ELECTRA)
- System alerts the user if the URL is phishing.

Pros

- Detects phishing URLs in real time.
- Uses machine learning (ELECTRA) for high accuracy.
- Better than traditional blacklist methods.
- Works during active user browsing.

Cons

- Depends on backend/server processing.
- Uses heavy models (not browser-friendly).
- No early warning before page load.
- Does not detect malicious JavaScript or DOM behavior.

CONNECTIVITY

- THE PAPER GIVES THE IDEA OF USING MACHINE LEARNING FOR PHISHING DETECTION.
- URL CHECKING USED IN THE PAPER IS TAKEN AS THE FIRST STEP IN OUR PROJECT.
- THE PAPER DETECTS PHISHING WHILE BROWSING; OUR PROJECT ADDS EARLY WARNING BEFORE OPENING LINKS.
- LIMITS OF BACKEND AND STATIC CHECKING IN THE PAPER LEAD TO OUR CLIENT-SIDE DESIGN.
- THE PAPER FOCUSES ONLY ON URLS; OUR PROJECT ADDS SCRIPT AND WEBPAGE BEHAVIOR CHECKING.

Research Gap

- DETECTION TIMING: PAPER DETECTS PHISHING AFTER BROWSING STARTS; YOUR PROJECT CAN AIM FOR FASTER, PRE-CLICK DETECTION.
- SCOPE OF ANALYSIS: PAPER ONLY CHECKS URLS; YOUR PROJECT CAN INCLUDE WEBPAGE CONTENT AND SCRIPT ANALYSIS FOR ADVANCED THREATS.
- MODEL DEPLOYMENT: PAPER USES HEAVY BACKEND MODELS; YOUR PROJECT CAN USE LIGHTWEIGHT, CLIENT-SIDE MODELS FOR REAL-TIME PERFORMANCE.
- USER INTERACTION: PAPER PROVIDES LIMITED ALERTS; YOUR PROJECT CAN GIVE CLEAR WARNINGS AND GUIDANCE TO USERS.
- ADAPTABILITY: PAPER MAY MISS NEW PHISHING SITES; YOUR PROJECT CAN INCLUDE ADAPTIVE OR CONTINUOUSLY LEARNING DETECTION.

Aditya Kulkarni, Vivek Balachandran, Tamal Das (2024)

Phishing Webpage Detection: Unveiling the Threat Landscape and Defense Strategies (IEEE Communications Surveys & Tutorials)

Objective

- To systematically review existing phishing webpage detection techniques
- To analyze attack trends, detection strategies, and limitations
- To identify research gaps and future directions for effective phishing defense

Methodology

- Comprehensive survey and categorization of phishing detection methods
- Analysis of:
- URL-based
- Content-based
- Behavior-based
- Machine Learning & Deep Learning approaches

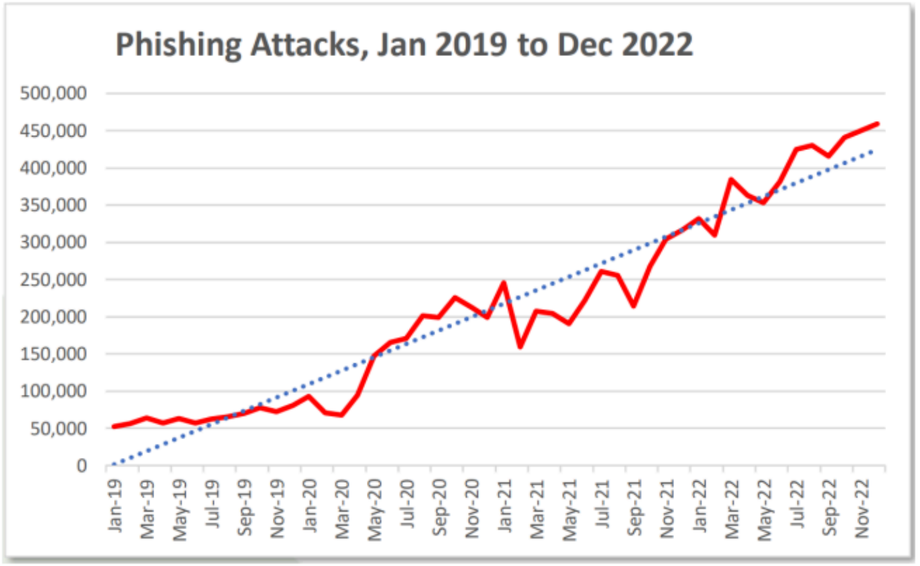


Fig. 1: Phishing Attacks from Jan 2019 to Dec 2022: APWG [6]

Algorithms & Technologies

- Machine Learning: Decision Tree, Random Forest, Naïve Bayes
- Deep Learning: CNN, RNN based
- Technologies: URL/domain analysis, HTML parsing, browser security mechanisms
- (No implementation — survey only)

Approach

- Analytical and comparative evaluation of existing research
- Focus on effectiveness vs real-world usability
- Highlights shortcomings in user-facing phishing protection

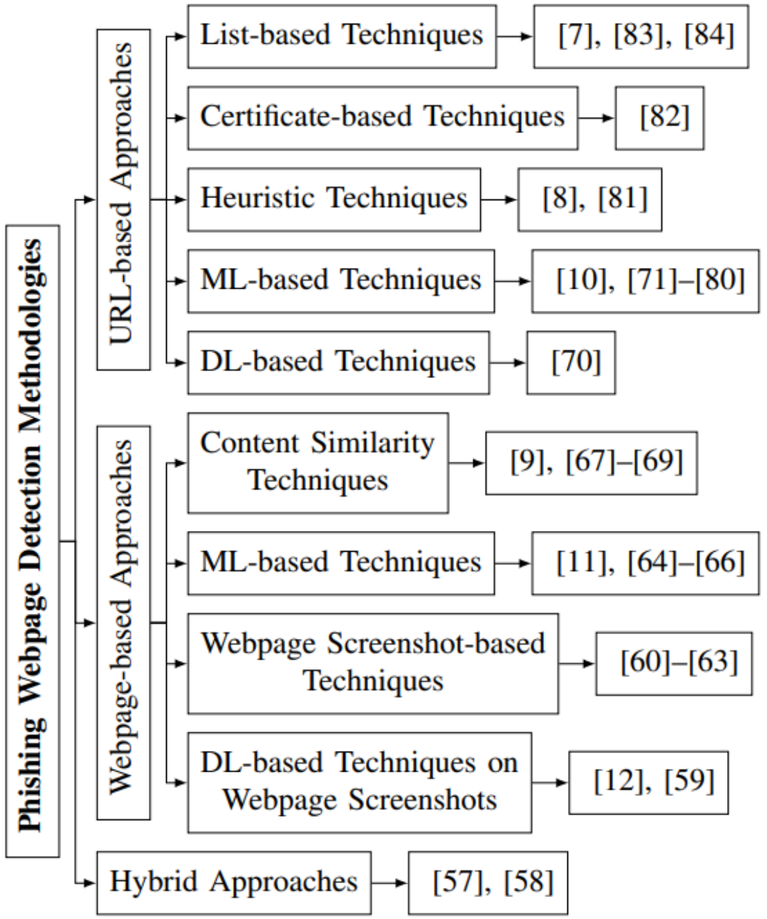


Fig. 3: Phishing Webpage Detection Methodologies

PROS

- HIGHLY COMPREHENSIVE AND AUTHORITATIVE SURVEY
- CLEARLY IDENTIFIES LIMITATIONS OF CURRENT SYSTEMS
- STRONG EMPHASIS ON FUTURE RESEARCH NEEDS

CONS

- NO PRACTICAL SYSTEM OR IMPLEMENTATION
- NO BROWSER EXTENSION OR REAL-TIME MODEL
- DOES NOT PROVIDE EARLY WARNING MECHANISMS

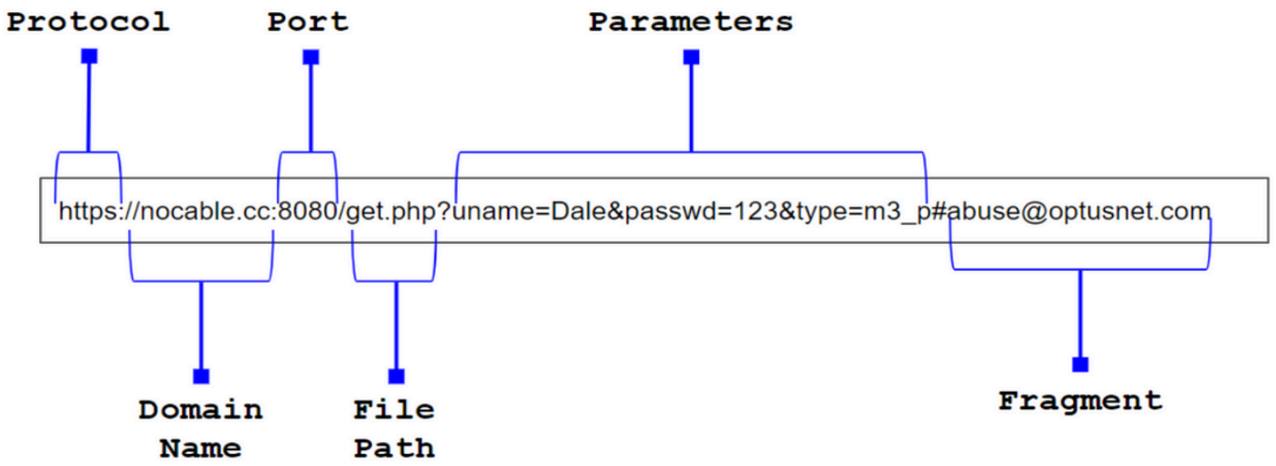


Fig. 2: URL Structure

RESEARCH GAP

- EXISTING PHISHING DETECTION SYSTEMS MAINLY RELY ON STATIC URL, DOMAIN, OR HTML CONTENT ANALYSIS
- DETECTION IS USUALLY POST-NAVIGATION, ALLOWING MALICIOUS SCRIPTS OR FAKE PAGES TO LOAD BEFORE WARNING
- LACK OF REAL-TIME JAVASCRIPT AND DOM BEHAVIOR ANALYSIS TO DETECT DYNAMIC PHISHING ATTACKS
- MOST SYSTEMS PROVIDE BINARY OUTPUTS (SAFE/UNSAFE) WITHOUT EXPLAINABLE RISK REASONING
- HEAVY RELIANCE ON BACKEND/CLOUD PROCESSING, UNSUITABLE FOR LIGHTWEIGHT BROWSER EXTENSIONS
- LIMITED FOCUS ON PROACTIVE, USER-CENTRIC, AND CLIENT-SIDE PROTECTION MECHANISMS

CONNECTIVITY

- THE PAPER CLEARLY IDENTIFIES THE NEED FOR PROACTIVE AND BEHAVIOR-BASED PHISHING DETECTION, WHICH OUR PROJECT IMPLEMENTS
- OUR SYSTEM PROVIDES PRE-NAVIGATION URL RISK ANALYSIS TO WARN USERS BEFORE ACCESSING HARMFUL LINKS
- INTRODUCES REAL-TIME JAVASCRIPT AND DOM INSPECTION TO DETECT PHISHING BEHAVIORS AFTER PAGE LOAD
- GENERATES DYNAMIC SAFETY SCORES WITH CLEAR EXPLANATIONS, ADDRESSING THE EXPLAINABILITY GAP
- DESIGNED AS A LIGHTWEIGHT BROWSER EXTENSION, ENABLING FAST, PRIVACY-PRESERVING CLIENT-SIDE DETECTION
- OUR PROJECT TRANSLATES THE FUTURE RESEARCH DIRECTIONS SUGGESTED IN THIS SURVEY INTO A PRACTICAL IMPLEMENTATION

Author & year	Title of the Paper	Methodology used	Limitations identified
Md.Redwan Ahmed, Md. Manowarul Islam, Md. Abu Layek (2024)	Phishing URL detection using comprehensive feature extraction and Machine Learning Techniques (IEEE CS BDC Symposium 2024)	The authors extract URL and domain features and use machine learning models such as Decision Tree, Random Forest, and XGBoost to classify phishing and legitimate websites.	The system relies only on static URL features, detects threats after the link is opened, and lacks real-time script analysis and clear user explanations.
S. Adolphine Shyni, A. Harish, P. Karthikeyan, S. Willfard Austen Zerro, R. Yogesh (2025)	Phish Defender: Real-Time Detection of Phishing Websites Using a Browser Extension (Springer, 2025)	This work introduces a browser extension that uses heuristic rules, URL analysis, and domain reputation to detect phishing websites during browsing.	Detection occurs after the page loads, allowing malicious scripts to run, and the system lacks early warnings and detailed script or DOM analysis.

Author & year	Title of the Paper	Methodology used	Limitations identified
Kousik Barik, Sanjay Misra, Raghini Mohan (2025)	Web-Based Phishing URL Detection Model Using Deep Learning Optimization Techniques (Springer – International Journal of Data Science and Analytics)	EGSO-CNN deep learning model that uses phishing URL datasets to improve classification accuracy and reduce false positives.	The deep learning method is computationally heavy, not suitable for lightweight browser extensions, and cannot provide client-side or pre-navigation warnings.
Authors as listed by Taylor & Francis (2024)	Real-Time Phishing URL Detection Framework Using Knowledge-Based and Learning Approaches (Taylor & Francis, 2024)	Rule-based checks and machine learning to detect phishing URLs in real time while users browse.	The approach depends on backend processing with heavy models, lacks a lightweight client-side design, and does not provide early warnings or real-time JavaScript threat detection.

Author & year	Title of the Paper	Methodology used	Limitations identified
Aditya Kulkarni, Vivek Balachandran, Tamal Das (2024)	Phishing Webpage Detection: Unveiling the Threat Landscape and Defense Strategies (IEEE Communications Surveys & Tutorials)	This survey paper reviews and categorizes phishing detection methods and behavior-based highlighting key trends and challenges.	It lacks practical implementation and notes gaps like delayed detection, poor explainability, and no pre-click warnings.

RESEARCH GAP

What is Missing in Existing Works:

Most systems rely on static URL/domain analysis and alert users only after page load. They lack real-time JavaScript/DOM monitoring, early pre-navigation warnings, explainable alerts, and lightweight client-side operation.

How Our Project Addresses It:

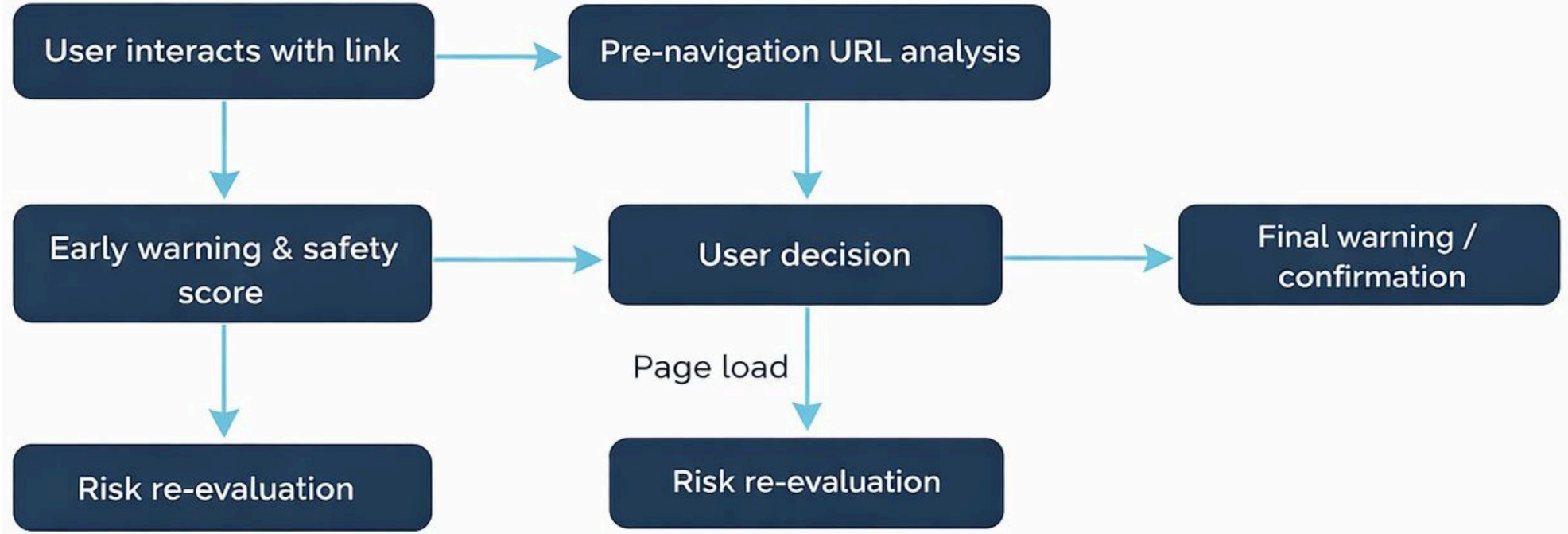
Our AI-powered browser extension detects phishing and malicious scripts in real time, analyzes website behavior before and during page load, provides dynamic safety scores with clear explanations, and runs mostly locally for proactive, user-friendly protection.

METHODOLOGY

The proposed system uses a two-stage detection approach. Before a website is opened, the extension analyzes the URL with heuristic rules such as suspicious patterns, abnormal length, special characters, and domain inconsistencies to provide an early warning. After the user enters the website, it performs real-time JavaScript and DOM inspection to detect malicious scripts, hidden forms, fake login elements, and phishing behaviors.

This approach improves upon existing systems by reducing impulsive clicks, minimizing early data exposure, and continuously updating risk levels instead of relying on a single detection step.

SYSTEM FLOW


User interacts with link
Pre-navigation URL analysis
Early warning & safety score
User decision
Final warning / confirmation
Page load
Risk re-evaluation
Risk re-evaluation

TECHNOLOGIES USED

1. **Programming Language:**
Python – for experimenting, training, and validating machine learning models for phishing detection.

2. **Libraries:**
- NumPy
- Pandas
- TensorFlow.js

3. **Tools / IDEs:**
- Visual Studio Code
- Git / GitHub
- Jupyter
- Chrome Developer Tools

4.

Browser / Extension APIs:

- Chrome Extension APIs
- DOM Inspection API
- JavaScript Runtime Analysis

5.

Data Sources:

Public phishing URL datasets (e.g., PhishTank, OpenPhish) for training and testing ML models.

6.

Visualization / UI Libraries :

Chart.js to display dynamic website safety scores and alerts in a user-friendly way.

MODEL USED

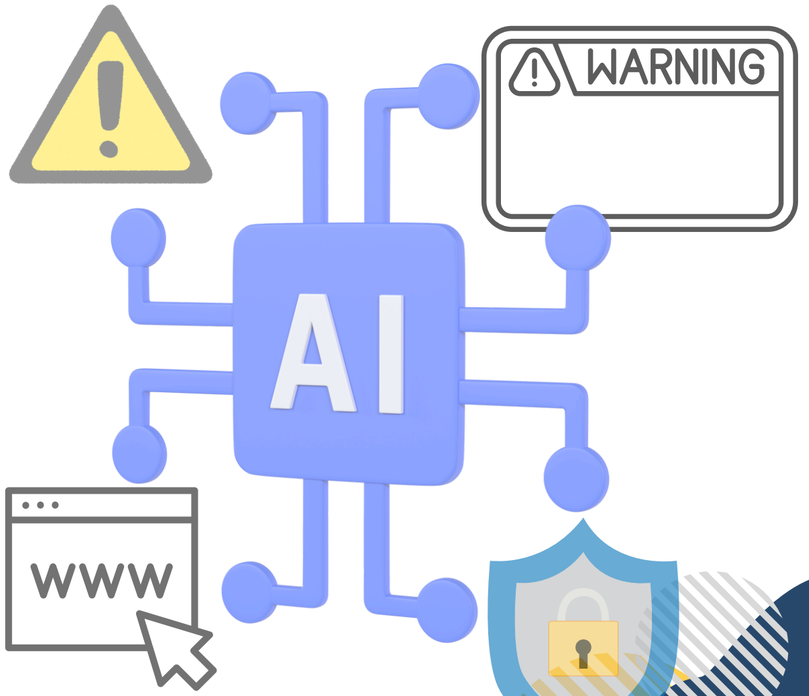
- The system uses a **Random Forest / lightweight machine learning model** for phishing and malicious script detection.
- This model was chosen because it is **robust**, **fast**, and **accurate** for classifying tabular features like URLs, domains, and webpage behavior, while remaining suitable for real-time browser execution.
- Key steps include extracting URL, domain, and JavaScript/DOM behavior features, training the model on labeled phishing and legitimate datasets, making real-time predictions during browsing and generating a dynamic website safety score with clear, explainable alerts for users.

DATASET DESCRIPTION

- **Source:** Public phishing URL datasets (PhishTank, OpenPhish)
- **Number of Records:** ~10,000–15,000 URLs (phishing + legitimate)
- **Features:** URL patterns, domain info, hidden forms, JavaScript/DOM behaviors
- **Data Preprocessing Steps:** Cleaning, Encoding, Normalization, Feature Selection and Data Splitting.

IMPLEMENTATION

Input: Website URL and webpage content (JS/DOM).
Processing: Pre-navigation heuristic check → Runtime JS/DOM monitoring → ML classification → Generate safety score and alerts.
Expected Output: Safety score, alerts, and user guidance.
System Workflow: User navigates → URL intercepted → Heuristic check → Page loads → Runtime analysis → ML classification → Safety score & alert.



ADVANTAGES

- Provides early warning before website access
- Reduces risk of impulsive phishing clicks
- Lightweight and browser-friendly
- No heavy backend dependency
- Improves user awareness through explainable warnings

LIMITATIONS

The system depends on heuristic rules and available datasets, which may not cover all zero-day attacks. Advanced obfuscated scripts may bypass detection. Browser extension permissions and performance constraints also limit deep inspection capabilities.

APPLICATIONS

1. educational institutions
2. e-commerce browsing
3. corporate environments
4. general internet security
5. online banking platforms

CONCLUSION

This project developed an AI-powered browser extension for real-time detection of phishing websites and malicious scripts. The system combines heuristic rules, lightweight machine learning, and runtime monitoring of JavaScript and DOM behavior to provide dynamic safety scores and clear risk alerts.

The outcome is a proactive, user-friendly tool that reduces exposure to phishing and malware threats.

This project is significant because it enhances safe browsing, improves user awareness, and demonstrates how AI can be effectively integrated into everyday web security.

REFERENCES

1. Browser Extension for Phishing Website Detection Using Machine Learning – Aditya Kulkarni, Vivek Balachandran, Tamal Das, IEEE, 2024.
2. Phishing URL Detection Using Comprehensive Feature Extraction and Machine Learning Techniques – Md. Redwan Ahmed, Md. Manowarul Islam, Md. Abu Layek, IEEE CS BDC Symposium, 2024.
3. Phish Defender: Real-Time Detection of Phishing Websites Using a Browser Extension – S. Adolphine Shyni, A. Harish, P. Karthikeyan, S. Willfard Austen Zerro, R. Yogesh, Springer, 2025.
4. Web-Based Phishing URL Detection Model Using Deep Learning Optimization Techniques – Kousik Barik, Sanjay Misra, Raghini Mohan, Springer, 2025.
5. Real-Time Phishing URL Detection Framework Using Knowledge-Based and Machine Learning Techniques – Authors as listed by Taylor & Francis, Taylor & Francis, 2024.

THANK YOU