Phisherman

Phishing website detection using Extreme Machine Learning

Pikala Ritwik Durga
Seemakurthi Nandan Sri Siva Ramakrishna
Ogireddy Sree Sudeep Reddy

Submitted in partial fulfillment of the requirements for the
course Machine Learning(CSN-382)
in Computer Science and Engineering

Supervisor : Pravendra Singh

April, 2025

Abstract

This progress report mainly discusses the different reasons for why we chose this project and other major decisions made in the model. This report starts with some background and then dives into the implementation details. Different reasons for using Extreme Learning Machine (ELM) over other models and feature selection is also mentioned. Some improvements over the initial model are discussed along with different comparisons of the accuracy of different models. Later, the report moves on to the front-end implementation and the integration with the model. The report ends with some probable future improvements and references.

Contributions

- **Pikala Ritwik Durga**: Trained ELM model and it's variations
- **Seemakurthi Nandan Sri Siva Ramakrishna**: Frontend(Browser extension), Flask-based API and backend integration
- **Ogireddy Sree Sudeep Reddy**: Feature extraction and preprocessing

Contents

Chapter 1

Introduction

With the rise of digital interactions, phishing websites have become a major cybersecurity threat, tricking users into giving away sensitive information like passwords and financial details. These fake websites mimic real ones, making them hard to detect. Phishing attacks are increasing rapidly, causing huge financial losses and reducing trust in online systems. Traditional detection methods, like blacklists and manual reporting, are too slow and ineffective against constantly evolving phishing tactics. This project focuses on developing a smarter, real-time solution to detect phishing websites more accurately and efficiently, helping to protect users and organizations from online fraud.

This project tackles the problem of phishing website detection using Extreme Learning Machine (ELM), a fast and scalable machine learning method that can quickly spot malicious websites. Unlike older detection methods, ELM trains faster and adapts better to new threats. To make this technology easily accessible, we're building a system where an ELM model runs on a Flask web server, allowing users to check websites through an API. We're also creating a browser extension that gives real-time warnings when users visit suspicious sites. By combining smart machine learning with a user-friendly tool, this project turns cybersecurity research into something people can actually use to stay safe online.

The objective of this report is to detail the design, implementation, and evaluation of this system, demonstrating its effectiveness in combating phishing attacks. The scope focuses on desktop browser environments, with an emphasis on real-time detection and user accessibility, though limitations such as internet dependency are acknowledged. The report is structured as follows: Section 2 reviews the background and related work, Section 3 outlines the implementation details, Section 4 presents results and analysis, Section 5 discusses future work and the report concludes with key contributions and bibliography. Through this work, we aim to enhance cybersecurity resilience and provide a scalable framework for thwarting phishing threats in an increasingly digital world.

Chapter 2

Background and Literature Review

2.1 Overview of Phishing Attacks

Phishing attacks represent a significant threat in the cybersecurity domain, exploiting human vulnerabilities to steal sensitive information such as login credentials, financial details, and personal data. These attacks often manifest as fraudulent websites that mimic legitimate entities, using tactics like domain spoofing, social engineering, and URL obfuscation. The economic and societal impacts are profound, with billions lost annually to phishing-related fraud and a growing erosion of trust in digital systems. As phishing techniques evolve—utilising HTTPS encryption and zero-day exploits—traditional detection methods like blacklists and manual reporting have proven useless.

2.2 Extreme Learning Machine (ELM)

Extreme Learning Machine (ELM) is a single-layer feedforward neural network model designed for efficient and high-performing classification and regression tasks. Unlike traditional gradient-based neural networks, which iteratively adjust weights and risk local minima traps, ELM randomly assigns input weights and biases, analytically computing output weights. This approach, as outlined by Huang et al.[1], reduces training time and enhances generalization, making it suitable for real-time applications. Sönmez and Gökal [5] used ELM to classify 30 phishing website features from the UCI Machine Learning Repository, achieving a test accuracy of 95.34%, outperforming methods like Support Vector Machines (SVM) and Naive Bayes (NB). Their work highlights ELM's efficacy in handling structured feature sets, such as URL-based, HTML-based, and domain-based attributes, which are critical for phishing detection.

The diagram 2.1 illustrates a typical feedforward neural network architecture composed of three layers: an **input layer**, a **hidden layer**, and an **output layer**. The input layer consists of n input nodes labeled $X(1), X(2), \ldots, X(n)$, each receiving one feature from the input data. These input nodes are fully connected to the m neurons in the hidden layer, meaning each

hidden neuron receives weighted inputs from all input nodes. Each hidden neuron also includes a bias term denoted as $b(1), b(2), \ldots, b(m)$. The hidden layer neurons then pass their outputs through an activation function and forward the signals to the output layer, which consists of p neurons generating the final outputs $y(1), y(2), \ldots, y(p)$. This structure is

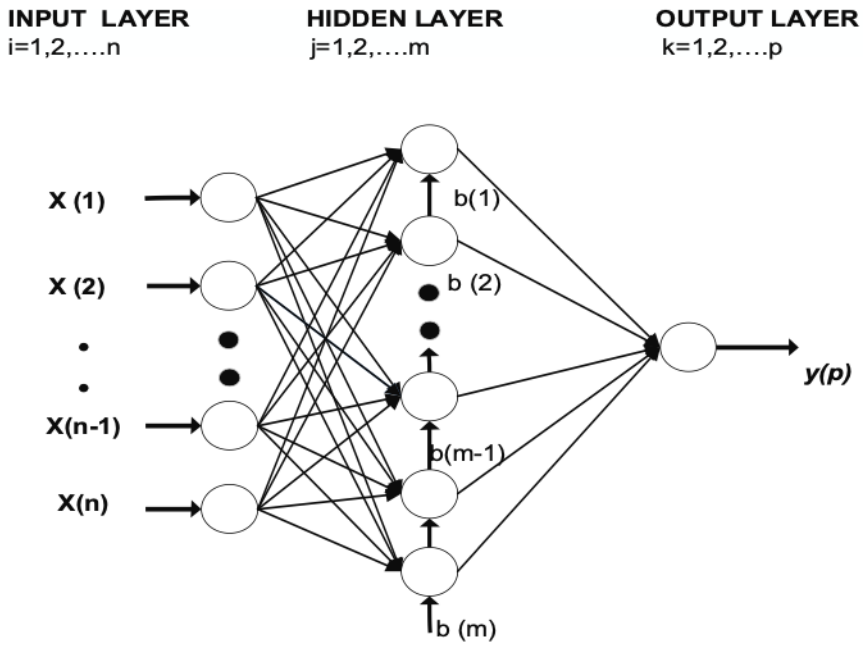

FIGURE 2.1: An artificial neural network model with a single hidden layer with forwardfeed [5]

2.3 Related Work

Phishing detection has been extensively studied, with approaches ranging from rule-based systems to machine learning techniques. Mohammad et al. [3] proposed a self-structuring neural network for phishing prediction, achieving a test accuracy of 92.18%, though it required significant computational tuning. Hadi et al. [4] introduced a fast associative classification algorithm, emphasizing speed but lacking real-time deployment mechanisms. Moghimi and Varjani [2] developed a rule-based method focusing on URL anomalies, yet it struggled with adaptability to evolving threats. Sönmez and Gökal's study [5] stands out by systematically extracting 30 features—categorized into address bar, abnormal, HTML/JavaScript,

and domain-based features—and applying ELM for classification. Their feature set, including indicators like HTTPS token presence and domain registration length, provides a robust foundation for identifying phishing sites.

However, most existing studies, including Sönmez and Gökal's,[5] focus on offline classification rather than practical, user-facing tools. Browser-based solutions, such as blacklist extensions (e.g., Google Safe Browsing), offer real-time protection but fail against zero-day attacks. Machine learning-based extensions, while promising, often lack integration with scalable backends. Flask, a lightweight Python framework, has been used in web applications for hosting ML models (e.g., RESTful APIs), but its application in phishing detection remains underexplored.

2.4 Relevance to the Proposed Solution

This project builds on Sönmez and Gökal's ELM-based classification by adapting their 30-feature framework for real-time phishing detection. While their study demonstrated ELM's superior accuracy and speed in a controlled environment, our work extends this into a deployable system. By hosting the ELM model on a Flask server and integrating it with a browser extension via API calls, we address the gap between theoretical models and practical tools. The extension extracts webpage URL dynamically, sends this to the Flask API, and delivers instant feedback to users—a step beyond static dataset analysis. This approach combines ELM's computational efficiency, Flask's scalability, and the extension's accessibility, offering a novel contribution to phishing prevention.

Chapter 3

Implementation Details

This section describes the design and implementation of a real-time phishing website detection system using Extreme Learning Machine (ELM), and a browser extension. The methodology uses the feature-based classification approach from Sönmez and Gökal [research paper] into a web extension.

3.1 System Architecture

The proposed system comprises three main components:

- **ELM Model**: A single-layer feedforward neural network trained to classify websites as phishing or legitimate based on extracted features. The model leverages ELM's random weight assignment and analytical output computation for efficiency.
- **Flask Server**: A lightweight web server hosting the trained ELM model, exposing it through a RESTful API endpoint (e.g., /predict) that accepts feature inputs and returns predictions.
- **Browser Extension**: A client-side tool built for desktop browsers (e.g., Chrome), which extracts webpage features, communicates with the Flask API, and displays real-time alerts to users.

The workflow begins with the extension capturing webpage data (e.g., URL, HTML content) as the user browses. These features are preprocessed, sent to the Flask server via an API call, processed by the ELM model, and returned as a binary classification (phishing or legitimate), triggering an appropriate user notification.

3.2 Implementation Steps

The system was developed through the following procedural steps:

3.2.1 Data Collection and Preprocessing

The dataset was sourced from the UCI Machine Learning Repository, as used by Sönmez and Gökal, containing approximately 11,000 website samples with 30 predefined features. These features span four categories: address bar-based (e.g., HTTPS token presence), abnormal-based (e.g., request URL percentage), HTML/JavaScript-based (e.g., iframe redirection), and domain-based (e.g., age of domain). Missing values, if any, were handled via imputation or exclusion to maintain dataset integrity.

3.2.2 Feature Extraction

To extract meaningful indicators for phishing detection, a custom feature extraction pipeline was implemented in Python using a combination of web scraping, WHOIS data retrieval, and URL parsing libraries. Key modules included requests, tldextract, socket, ssl, BeautifulSoup, and whois. The pipeline evaluated both static and dynamic attributes of the webpage and domain, such as IP address usage, URL length, redirection patterns, HTTPS certificate details, domain registration length, and HTML behaviors (e.g., usage of iFrames or JavaScript-based redirections). Environmental variables and API keys for services like SerpAPI and VirusTotal were securely managed using the dotenv library. This multi-faceted approach enabled the extraction of over 30 features across four categories, supporting robust classification by the ELM model.

3.2.3 ELM Model Training

The ELM model was implemented as a single-layer feedforward neural network designed for fast training. It randomly assigns input weights and biases, then analytically computes output weights using the Moore-Penrose pseudoinverse. This implementation enhances the basic ELM with multiple activation functions, L2 regularization (alpha parameter) to prevent overfitting, and advanced weight initialization strategies.

The model processes input features (e.g., the 30 phishing website features) through a hidden layer with a default of 1000 neurons, applies the chosen activation function, and computes output weights to classify websites as phishing (-1) or legitimate (1). The achieved accuracy is approximately 95.40%, which is close to the 95.34% reported in the research paper.

Ensemble ELM with Weighted Averaging

An ensemble of ELM models was created where predictions are combined using weighted averaging. Weights are determined by each model's performance on a validation set, with diverse configurations:

Input (Features)		Output (Class)
1.1. Address Bar based Features		
1.1.1	Using the IP Address	
1.1.2	Long URL to Hide Suspicious Part	
1.1.3	Using URL Shortening Services	
1.1.4	URL's having "@" Symbol	
1.1.5	Redirecting using "//"	
1.1.6	Adding Prefix/Suffix by (-)	
1.1.7	Sub Domain and Multi Sub Domains	
1.1.8	HTTPS (SSL)	
1.1.9	Domain Registration Length	
1.1.10	Favicon	
1.1.11	Using Non-Standard Port	
1.1.12	"HTTPS" Token in Domain	
1.2. Abnormal Based Features		
1.2.1	Request URL	-1 Phishing / 1 Legitimate
1.2.2	URL of Anchor	
1.2.3	Links in Meta, Script, Link tags	
1.2.4	Server Form Handler (SFH)	
1.2.5	Submitting Info to Email	
1.2.6	Abnormal URL	
1.3. HTML and JavaScript based Features		
1.3.1	Website Forwarding	
1.3.2	Status Bar Customization	
1.3.3	Disabling Right Click	
1.3.4	Using Pop-up Window	
1.3.5	IFrame Redirection	
1.4. Domain based Features		
1.4.1	Age of Domain	
1.4.2	DNS Record	
1.4.3	Web Traffic	
1.4.4	PageRank	
1.4.5	Google Index	
1.4.6	Number of Links to Page	
1.4.7	Statistical Reports Feature	

TABLE 3.1: Features of Websites

- Hidden units: 800, 1000, 1200, 1500, 2000
- Activations: ReLU, leaky ReLU, ELU, Swish

The process involves splitting the training data into training and validation sets, training each model on the training set, evaluating accuracy on the validation set, assigning weights proportional to validation accuracy, normalizing them, retraining models on the full dataset,

and combining predictions using these weights. This approach achieved an accuracy of 96.36%.

Bagging Ensemble ELM

This method implements bootstrap aggregating (bagging) with Enhanced ELM models. Each model (default 10 estimators) is trained on a random subset (80% sample ratio) of the training data with replacement, using 1000 hidden units and ReLU activation.

The workflow includes generating bootstrap samples for each base model, training each model independently on its sample, and combining predictions via majority voting (averaged and rounded). The resulting accuracy is 95.91%.

Stacking Ensemble ELM

An ensemble method that combines predictions from multiple base Enhanced ELM models using a meta-learner (another Enhanced ELM). The base models vary in hidden units and activation functions.

The process involves training base models on a subset of the training data, using their predictions on a validation set as features for the meta-learner, which is trained with 100 hidden units and sigmoid activation to make the final prediction. Base models are then retrained on the full dataset for deployment, achieving an accuracy of 96.51%.

3.2.4 Browser Extension Development

A browser extension was developed for Chromium based browsers using JavaScript, HTML, and CSS. The extension operates as follows:

- **Feature Extraction**: Upon page load, the URL is automatically captured by the extension. This URL is sent to the flask server where it extracts the 30 features (e.g., chrome.tabs for URL, DOM parsing for HTML content). Rules from Sönmez and Gökal (e.g., "HTTPS token in domain $\rightarrow$ phishing") guide the extraction logic.
- **User Interface**: The response triggers a visual alert—e.g., a red popup for phishing sites or a green icon for legitimate ones—integrated into the browser toolbar.

3.2.5 Integrating the Model With the Frontend

We successfully trained the model using the Extreme Learning Machine (ELM) algorithm in Python. To bridge the trained model with the user interface, we used Flask, a lightweight Python web framework. Flask was used to host the model on a local server and expose an API endpoint that could accept input and return model predictions.

On the frontend, built using Vanilla JavaScript, we implemented API calls using `fetch()` to send user input to the Flask server and receive the model's response. This allowed real-time communication between the frontend and backend, enabling seamless interaction for the user. The Chrome extension now reflects the model's output dynamically based on user inputs.

3.3 Tools and Technologies

- **Python:** Used for ELM model training.
- **JavaScript/HTML/CSS:** Core technologies for the Chrome extension, with the Chrome Extension API for browser integration.
- **Flask:** Lightweight Python web framework used to build the backend API for communication between the model and the extension.
- **Chrome Extension API:** Used for browser integration and extension functionality.

3.4 Model Evaluation

Model performance was assessed using accuracy, precision, recall, and F1-score, calculated on the test set. Results were compared against baseline methods (e.g., SVM, NB) to validate ELM's superiority, drawing on Sönmez and Gökal's findings. Later other variants of ELM are also evaluated.

Chapter 4

Results and Analysis

This section evaluates the performance of the proposed phishing website detection system, encompassing the Enhanced Extreme Learning Machine (ELM) model, the Flask-hosted API, and the browser extension. Experiments were conducted to assess classification accuracy, system latency, and real-world usability, with results compared to baseline methods and prior work by Sönmez and Gökal [5].

4.1 ELM Model Performance

The performance of the ELM model, along with its ensemble variations, was evaluated on the 30 features from Sönmez and Gökal's dataset. The evaluation utilized a 70%-15%-15% split for training, validation, and testing, along with 5-fold cross-validation to ensure reliable results.

The performance metrics—accuracy, precision, recall, and F1-score—were computed for each method, with results shown in Table 4.1. The following methods were evaluated:

- **Generalized ELM:** Achieved an accuracy of 95.40%.
- **Bagging Ensemble ELM:** Achieved an accuracy of 95.91%.
- **Ensemble ELM with Weighted Averaging:** Achieved an accuracy of 96.36%.
- **Stacking Ensemble ELM:** Achieved an accuracy of 96.51%.

4.2 Comparison with Baseline Methods

To validate the effectiveness of the ELM model and its ensemble methods, performance was benchmarked against traditional machine learning models such as Support Vector Machines (SVM) and Naive Bayes (NB), using the same dataset and feature set. The results are presented in Table 4.1.

TABLE 4.1: Accuracy of Machine Learning Methods

Method	Train Accuracy	Test Accuracy
ELM (Generalized)	100%	95.40%
Bagging Ensemble ELM	100%	95.91%
Ensemble ELM (Weighted Averaging)	100%	96.36%
Stacking Ensemble ELM	100%	96.51%
SVM	100%	93.1%
NB	100%	92.5%

4.3 Frontend outputs

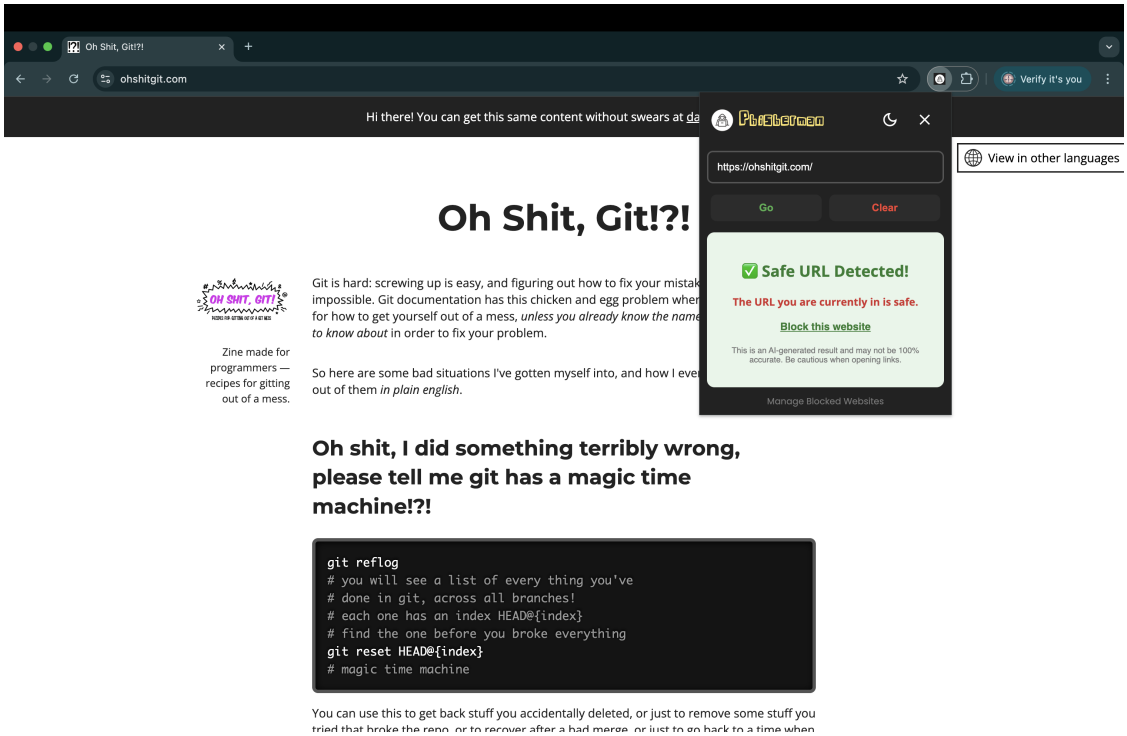

FIGURE 4.1: Output when URL is safe

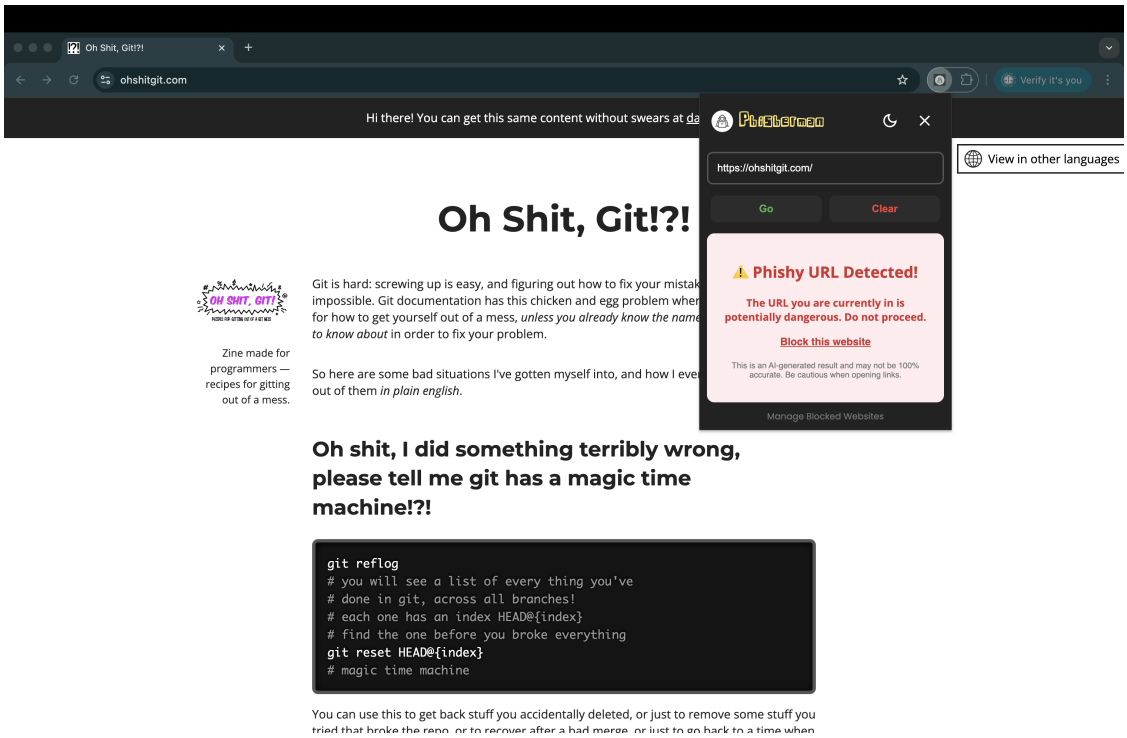

FIGURE 4.2: Output when URL is unsafe

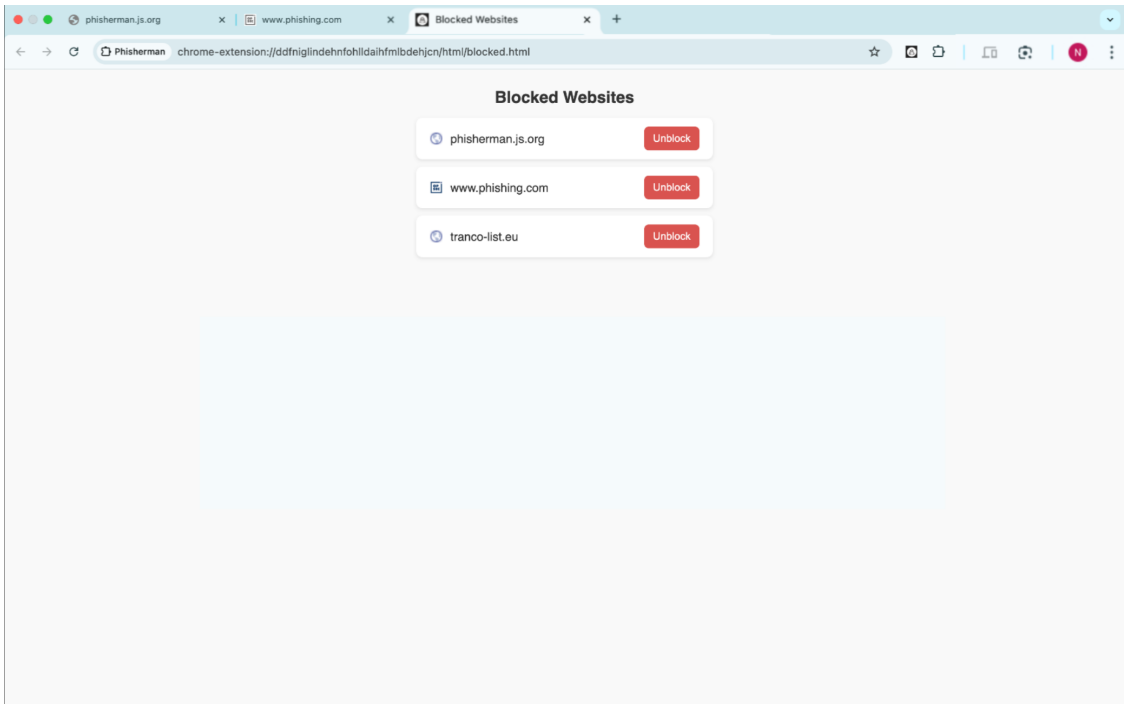

FIGURE 4.3: List of blocked URL's

Chapter 5

Future Work

We have successfully trained the phishing detection model using the Enhanced Extreme Learning Machine (ELM) and developed a basic front-end using VanillaJS. While the foundation has been laid, several important features and improvements remain to be implemented in order to complete the system and enhance its effectiveness.

5.1 Hosting the Model Online

To further scale and increase accessibility, the model will be deployed on a cloud platform. This ensures high availability and scalability for users across different regions. Additionally, the browser extension will be published to popular web stores, such as the Chrome Web Store, to make the tool publicly available. This broader distribution will help gather user feedback and real-world usage data, which can be used to fine-tune the model and improve detection accuracy over time.

5.2 User Reporting Feature

We aim to introduce a user reporting feature within the extension. This will allow users to report suspicious URLs directly through the browser interface. These crowdsourced reports will be reviewed by the development team, and verified URLs will be added to the internal phishing database. This mechanism not only engages users in the detection process but also helps keep the model updated with emerging phishing threats, thereby improving its responsiveness and long-term accuracy.

By combining intelligent detection with user interaction, we aim to build a robust, scalable, and community-driven phishing defense system.

List of Figures

List of Tables

Bibliography

[1] Chee-Kheong Siew Guang-Bin Huang, Qin-Yu Zhu. Extreme learning machine: Theory and applications. https://www.sciencedirect.com/science/article/pii/S0925231206000385.

[2] M. Moghimi and A. Y. Varjani. New rule-based phishing detection method. https://www.sciencedirect.com/science/article/pii/S0957417416000385.

[3] Lee McCluskey Rami M. Mohammad, Fadi Thabtah. Predicting phishing websites based on self-structuring neural network. https://link.springer.com/article/10.1007/s00521-013-1490-z.

[4] F. Aburub W. Hadi and S. Alhawari. A new fast associative classification algorithm for detecting phishing websites. https://www.sciencedirect.com/science/article/pii/S1568494616303970.

[5] Hüseyin Gökal Engin Avcı Yasin Sönmez, Türker Tuncer. Phishing web sites features classification based on extreme learning machine. https://ieeexplore.ieee.org/document/8355342.