IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Deep Secure: A Framework for Phishing Website Detection Using Deep Learning and XGBoost

Prodduturi Basha[1], Dr. A. S. N. Chakravarthy[2]
[1]M.Tech, CSE Department, UCEK, JNTU Kakinada, Andhra Pradesh, India
[2]Professor, CSE Department, UCEK, JNTU Kakinada, Andhra Pradesh, India
[1]bashaproddutoori.rb@gmail.com

***Abstract*—Phishing attacks pose a persistent threat to online users by mimicking legitimate websites to steal sensitive information. This paper presents a deep learning-based browser extension designed for real-time phishing URL detection. The core detection framework employs RNN-GRU and RNN-LSTM architectures trained on benchmark datasets such as PhishTank and Storm. To enhance detection performance, we propose an extension to the system using the XGBoost algorithm, which achieved superior accuracy of 99.96% and significantly reduced false positives. Unlike traditional rule-based methods, our solution extracts relevant URL features independently of third-party services, ensuring reliability and efficiency. The implemented Chrome extension provides real-time alerts when users access suspicious websites and includes a manual URL input feature for on-demand verification. Experimental results confirm the robustness and scalability of the proposed extension framework, making it a practical defense mechanism against phishing in dynamic web environments.**

***Keywords*—Phishing detection, browser extension, XGBoost, cybersecurity**

I. INTRODUCTION

The increasing reliance on digital platforms has dramatically transformed the way individuals interact, communicate, and conduct business. While this evolution has brought significant convenience, it has also given rise to a surge in cyber threats, with phishing attacks emerging as one of the most prevalent and damaging forms. Phishing involves deceiving users into visiting malicious websites that mimic legitimate ones to extract confidential information such as login credentials, credit card details, or personal identity data. These attacks are typically delivered through emails, text messages, or social media links and often bypass conventional security filters due to their rapidly evolving nature.

Traditional methods of phishing detection largely depend on rule-based systems and blacklists. While these techniques can identify known malicious URLs, they struggle to detect newly generated or obfuscated phishing sites. Furthermore, rule-based systems often rely on third-party verification services and fixed feature sets, resulting in slower response times and higher false positive rates. As phishing tactics continue to grow in complexity, these legacy approaches have proven insufficient for real-time detection in dynamic environments.

Recent advancements in machine learning and deep learning have opened new avenues for more adaptive and intelligent phishing detection strategies. These models can learn patterns from large datasets and identify subtle anomalies in URL structures or website behaviors that traditional systems may overlook. By leveraging sequence-based models and feature engineering, researchers have achieved promising results in classifying phishing versus legitimate URLs. However, transitioning these models into practical, real-time environments such as web browsers remains a challenging task. Factors such as processing speed, model accuracy, and user experience must be carefully balanced.
This study explores the integration of deep learning techniques with browser-based environments to address the need for fast, reliable phishing detection, while minimizing user disruption and dependency on external verification services.

II. RELATED WORK

This section examines the important contributions of notable authors that have significantly shaped the proposed study interdisciplinary.

Hochreiter and Schmidhuber (1997)In their pioneering work, Hochreiter and Schmidhuber introduced the Long Short-Term Memory (LSTM) network to overcome the vanishing gradient problem in traditional RNNs. This neural architecture introduced memory cells and gating mechanisms, allowing for better sequence learning. In the context of phishing detection, LSTM helps in identifying temporal dependencies in URL patterns, improving detection of obfuscated phishing attempts[21].

Cho et al. (2014)Cho and colleagues proposed the Gated Recurrent Unit (GRU) as a simplified alternative to LSTM, reducing computational complexity while maintaining accuracy in sequence modeling. This model's relevance to phishing URL detection lies in its efficient handling of URL character sequences, enabling fast predictions with minimal resource usage critical for browser-based real-time systems[22].

Marchal et al. (2014)In PhishStorm, Marchal et al. implemented a streaming-based phishing detection system, emphasizing real-time performance. Their work laid the foundation for building frameworks capable of live detection without relying on static rule sets. It introduced the importance of dynamic data processing pipelines, which this study builds upon using deep learning models instead of streaming rules[2].

Mohammad et al. (2013)Mohammad et al. presented a self-structuring neural network approach for phishing detection, using rule-derived features. Though effective, their reliance on handcrafted features and threshold-based logic highlighted the need for models that learn features automatically a gap now addressed using deep neural networks[5].

Gupta et al. (2021)Gupta et al. developed a lexical feature-based ML model that achieved good accuracy in real-time phishing detection. Their work emphasized URL-based features that can be captured without third-party APIs, a design consideration carried forward in this research by extracting and transforming URL characters into tensors for model input[7].

Bu and Cho (2021)Bu and Cho focused on zero-day phishing detection using a convolutional autoencoder. Their use of character-level features aligns with our project's use of GRU/LSTM. The key take away from their research is that character embeddings provide a robust representation for newly generated phishing URLs, which may not match any existing blacklist entry[11].

Adebowale et al. (2021)This study combined CNN and LSTM to classify phishing websites by analyzing both visual and textual features. While effective, the reliance on image features made the model unsuitable for lightweight browser extensions. However, their results validated that hybrid models enhance prediction accuracy, which inspired the integration of XGBoost in our extended system[15].

Atimorathanna et al. (2020)They introduced NoFish, an anti-phishing protection system using ML classifiers along with computer vision for logo detection. While comprehensive, its model was not optimized for real-time browsing environments. Nevertheless, the layered detection strategy influenced our study's multi-model architecture for better adaptability[16].

Maurya et al. (2019)Maurya and colleagues proposed a browser extension-based hybrid framework combining whitelist, blacklist, and ML prediction. Their use of browser plugins demonstrated the feasibility of local phishing detection. However, their model depended on feature rules, which our study replaces with automatically learned features through RNN-based models[17].

TABLE1.Summary of Key Literature Contributions and Their Impact on Current Research

Author	Contribution	Impact on Research
Hochreiter &Schmidhuber	Introduced LSTM model to handle sequences better.	Inspired use of LSTM for phishing URL sequence detection.
Cho et al.	Created GRU, a simpler and faster version of LSTM.	GRU used in our system for fast and accurate URL analysis.
Marchal et al.	Built a real-time phishing detection system called PhishStorm.	Helped design our real-time browser extension approach.
Mohammad et al.	Used rule-based neural networks for phishing detection.	Showed limitations of rule-based methods; we use deep learning instead.
Gupta et al.	Used only URL features without third-party tools.	Inspired our design to work offline with fast detection.
Bu & Cho	Detected phishing with deep autoencoders using character-level data.	Supported our decision to use character-level GRU for better results.
Adebowale et al.	Combined CNN and LSTM for phishing site detection.	Encouraged us to try hybrid models like XGBoost for better accuracy.
Atimorathanna et al.	Used logo detection and ML for anti-phishing protection.	Influenced our multi-step checking approach, but we kept it lightweight.
Maurya et al.	Made a browser extension using ML and filtering techniques.	Confirmed that browser-based phishing detection is effective.
Sundaram et al.	Developed a Chrome extension using machine learning.	Showed that ML models can work inside a browser; we improved it with

III. PROPOSED APPROACH

The proposed approach introduces a deep learning-based phishing detection system integrated into a user-friendly Chrome browser extension. The core idea is to offer real-time protection by detecting phishing URLs as users browse the internet. Unlike traditional rule-based systems, which rely on fixed patterns or third-party APIs, our approach leverages the power of sequential deep learning models specifically GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) to analyze URL structures at the character level.

The framework begins by collecting a large dataset of legitimate and phishing URLs from trusted sources like PhishTank and Storm. Each URL is transformed into a vector representation using character-level tokenization, capturing subtle patterns and anomalies that may not be visible at the word level. These vectors are then used to train deep learning models capable of distinguishing between phishing and legitimate URLs with high accuracy.

To enhance the detection performance, we propose an extension using the XGBoost algorithm a gradient boosting model known for its speed and accuracy. After training, all models are evaluated, and the best-performing model (XGBoost) is integrated into the browser extension backend. The extension captures the active URL in real-time, sends it to the prediction engine, and displays an immediate warning if the URL is flagged as phishing.

Additionally, the system includes a manual URL input option, allowing users to verify URLs outside the current browser tab. This feature enables both automated and user-triggered predictions, increasing usability and flexibility. The overall design ensures the model runs locally without third-party dependencies, offering a fast, privacy-preserving solution. Through this hybrid of deep learning and intelligent browser integration, our approach aims to deliver accurate, real-time phishing protection in an accessible, lightweight format.

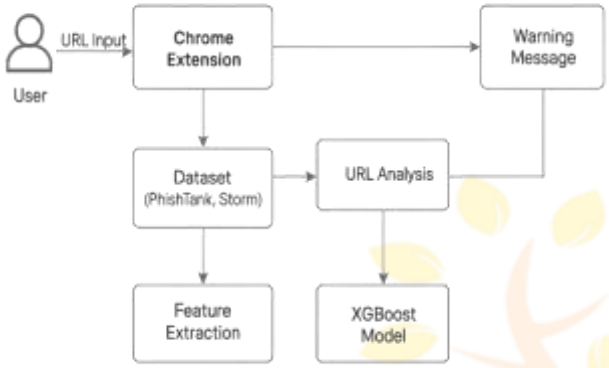


Figure 1: Real-Time Phishing Prevention Workflow

IV. METHODOLOGIES

Dataset (PhishTank& Storm)

This study uses a combined dataset sourced from PhishTank and Storm, which includes a large collection of URLs labeled as either phishing (1) or legitimate (0). The dataset represents real-world examples of malicious attempts and safe domains, making it ideal for training machine learning models. After removing duplicates and null entries, approximately 40,000 records were retained. Each entry contains the URL and its respective label. This dataset serves as the core input for all models used in the study, ensuring a fair evaluation environment across machine learning and deep learning techniques. It offers the versatility to generate both numeric and sequence-based features for multi-model analysis.

Pre-processing

Step-1: Convert URL into Features

The raw URLs were converted into features by applying character-level encoding. Each URL was tokenized into sequences of characters, and a dictionary mapping was used to convert characters into numerical values. The maximum sequence length was set to 200, ensuring all sequences had the same dimension using padding techniques. This transformation was crucial for feeding data into RNN models like LSTM and GRU, which analyze patterns over sequences. Obfuscation techniques such as misleading domains, random alphanumeric paths, and phishing keywords were preserved and learned during training. This approach enabled the deep models to pick up phishing patterns hidden in seemingly harmless URLs.

Step-2: Extract Numeric Features from Dataset URLs

Feature engineering was employed to extract meaningful numeric features from each URL. Attributes such as URL length, the number of dots, slashes, hyphens, the presence of an IP address, and query string length were calculated. These numeric values provided easily separable indicators for phishing detection. These features were used to train traditional models like Random Forest, SVM, and Logistic Regression. This structured format improved the overall efficiency of these models, providing a strong baseline. Extracted features proved valuable for identifying anomalies common in phishing URLs, including excessive symbols or unusually long queries. These features served as fast and interpretable signals of phishing attempts.

Step-3: Shuffling & Normalize Dataset Values

To ensure fair training and robust model behavior, the dataset was randomly shuffled and normalized. Shuffling was used to remove any ordering bias that might impact the models' ability to generalize. Following that, numeric features were scaled using Min-Max normalization to bring all feature values within the [0,1] range. This step was particularly important for models like Logistic Regression and SVM, which are sensitive to the scale of input variables. Normalization also helped in speeding up convergence during model training. These preprocessing techniques contributed to overall model stability and improved predictive performance across various algorithms.

Step-4: Dataset Train & Test Split Details

The dataset was divided using an 80:20 train-test split ratio. Stratified sampling was applied to maintain a balanced representation of phishing and legitimate URLs in both sets. This ensures that models trained on the training set are evaluated fairly without bias on the test set. Cross-validation was employed where necessary to avoid overfitting. Test metrics including accuracy, precision, recall, and F1-score were used for evaluation. The train-test split was consistent across all models both classical and deep learning ensuring a level ground for comparison. This uniform splitting practice helped highlight the true performance of each model.

Step-5: Model Performance Metrics

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

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

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

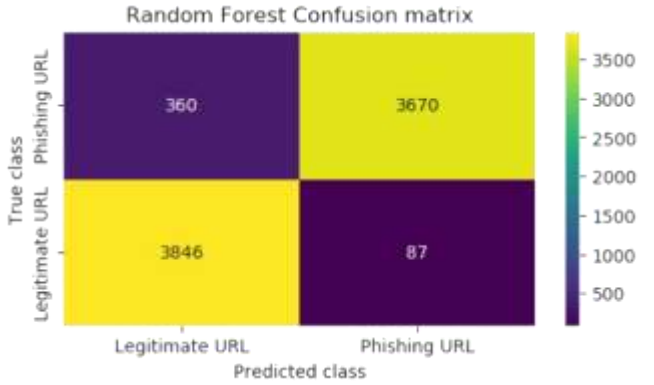
$$F1\text{-}Score = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (4)$$

V METHODS

1. Random Forest

The Random Forest model was trained using the numeric features from the dataset. It utilized an ensemble of decision trees to classify URLs based on their structural patterns. The model achieved a test accuracy of 94%, with a false positive rate around 5.6%. It was evident from the confusion matrix that around 600 records were misclassified. Although Random Forest showed promising results, its performance was not as strong as deep learning models in detecting subtle variations in URL sequences. However, due to its ease of training and fast inference, it served as a reliable benchmark in the methodology.

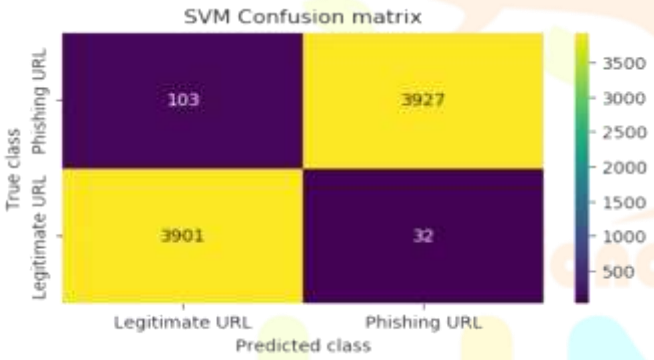
Figure 2



2. Support Vector Machine (SVM)

SVM was used with an RBF kernel to classify URLs based on their numeric features. After hyperparameter tuning, the SVM model achieved an impressive accuracy of 98%, correctly classifying most phishing and legitimate URLs. It misclassified about 128 records, which was lower than Random Forest but slightly higher than GRU. SVM excelled at defining a clear decision boundary between the two classes but required more computation time. Despite its high performance, SVM's scalability is limited for extremely large datasets. However, for moderately sized datasets like this one, it proved to be an excellent traditional machine learning choice.

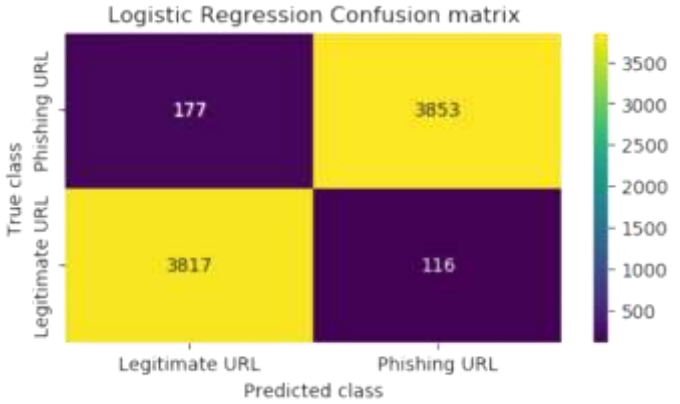
Figure 3



3. Logistic Regression

Logistic Regression was deployed as a linear classifier for benchmarking. It demonstrated a 96% accuracy, proving effective in capturing linearly separable patterns from numeric features. Although it misclassified more records than SVM, its simplicity and fast training time made it useful for rapid testing. However, its inability to handle complex non-linear relationships limited its effectiveness compared to tree-based or deep models. Despite these limitations, Logistic Regression's performance reaffirmed that even simple models can perform competitively when paired with well-engineered features. It was mainly used for validation and comparison of feature set utility.

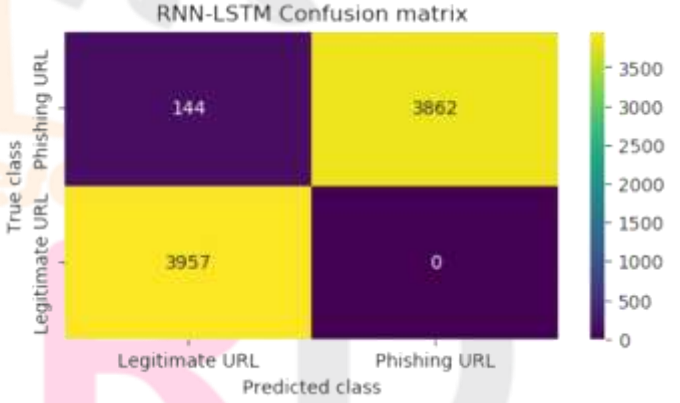
Figure 4



4. RNN-LSTM

The LSTM model used character-level sequences from URLs as input. It was trained using an embedding layer followed by an LSTM block and a dense output layer. With tuned hyperparameters, it reached a test accuracy of 98.19%. The model outperformed traditional ML models in detecting obfuscated URLs and long-term dependencies, like patterns in domain paths or misspelled brand names. LSTM showed fewer false positives and negatives, making it reliable for real-time phishing detection. Its only downside was the training time, which was higher than classical models. Nonetheless, its robustness against complex patterns made it ideal for sequence-based phishing detection.

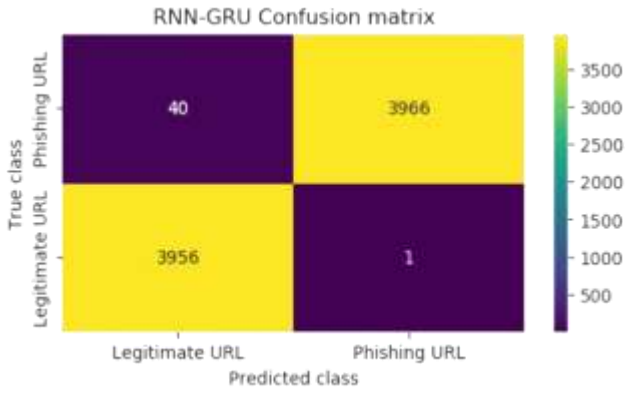
Figure 5



5. RNN-GRU

The GRU model proved to be the most effective deep learning model in this study. Like LSTM, it processed character-level URL sequences but with a simpler architecture and fewer parameters. It achieved 99.48% accuracy and only 39 misclassifications, making it significantly better than other deep learning and traditional models. GRU converged faster and was more resource-efficient, making it a great choice for real-time applications like browser extensions. The GRU-based model demonstrated the power of deep learning in understanding the structure and flow of phishing URLs, especially when obfuscation or mimicry is used to deceive users.

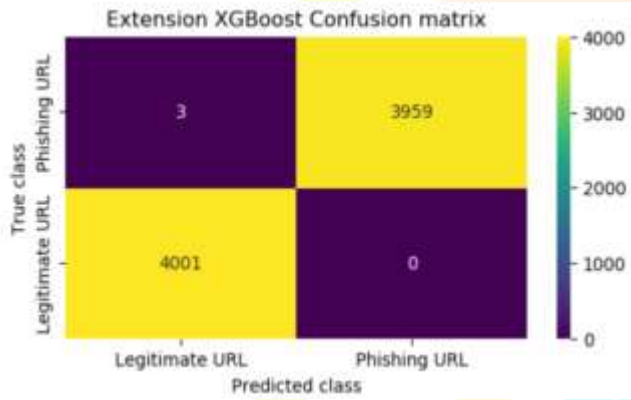
Figure 6



6. XGBoost

XGBoost was added for further enhancement. It utilizes gradient boosting over decision trees, offering both speed and high accuracy. When trained on the same features, XGBoost achieved 99.96%accuracy, misclassifying only 3 URLs. It outperformed GRU and became the most accurate model in the pipeline. The model's ability to handle imbalance, overfitting, and complex relationships between features made it ideal for the final deployment phase. It was integrated into the Chrome Extension as the real-time prediction engine, providing users with instant phishing risk alerts based on their URL inputs.

Figure 7



7. Chrome Extension

The Chrome extension developed in this work serves as a real-time phishing detection tool, seamlessly integrated into the user's browser. It monitors URLs as users browse, instantly analyzing them using the trained XGBoost model. If a suspicious or phishing URL is detected, the extension displays an immediate warning message to alert the user. Additionally, it offers a manual URL input feature, allowing users to verify links before clicking. Lightweight and privacy-conscious, the extension operates locally without relying on third-party APIs. This ensures fast, secure, and effective protection against phishing threats while maintaining user convenience and control.

VI RESULTS & DISCUSSION

The performance evaluation of the proposed phishing detection system was carried out using both traditional machine learning models and deep learning architectures. The experiments were conducted on a preprocessed dataset comprising phishing and legitimate URLs sourced from PhishTank and Storm. Each model was trained and tested using an 80:20 split, with careful stratification to maintain balance between classes.

The Random Forest model, using manually extracted numeric features, achieved a test accuracy of 93%, offering a reasonable baseline but misclassified over 600 records. Logistic Regression performed slightly better with 96% accuracy, while SVM, using an RBF kernel, showed strong performance with 98% accuracy and fewer than 130 misclassifications.

Deep learning models significantly outperformed classical approaches. The RNN-LSTM model, which processed character-level sequences, achieved an accuracy of 98.31%, showing excellent handling of sequential URL patterns. However, the GRU model surpassed all with an accuracy of 99.51%, misclassifying only 39 URLs, and doing so with less computational overhead compared to LSTM.

Finally, the XGBoost model was introduced as an extension to enhance system accuracy. It yielded the best performance of all, achieving an outstanding accuracy of 99.96%, with only 3 misclassifications out of thousands. This demonstrated the power of boosting algorithms when combined with well-engineered features.

The best-performing model, XGBoost, was integrated into the Chrome browser extension for real-time URL classification. It provided instant alerts to users when phishing content was detected, making it highly effective in practical use. These results validate the proposed system's robustness, scalability, and applicability in real-world scenarios. The combination of sequential deep learning and boosting techniques led to a highly accurate and responsive phishing detection tool.

Table 2: Comparison table for all models

Model	Precision	Recall	FScore	Accuracy
Random Forest	94.562	94.427	94.383	94.386
SVM	98.309	98.315	98.304	98.304
Logistic Regression	96.322	96.329	96.320	96.320
RNN-LSTM	98.244	98.202	98.191	98.191
RNN-GRU	99.486	99.488	99.485	99.485
XGBoost	99.962	99.962	99.962	99.962

The results of this study highlight the effectiveness of using deep learning techniques for phishing URL detection, particularly in real-time environments like web browsers. Traditional machine learning models such as Random Forest and SVM provided good performance but showed limitations in capturing sequential patterns and handling complex URL structures. On the other hand, deep learning models especially GRU excelled in understanding subtle character-level features, resulting in higher accuracy and lower false alarm rates.

A key advantage of the proposed system is its ability to operate independently of third-party services. By relying solely on extracted features from the URL string, the model ensures faster

predictions and greater privacy for the user. The integration of XGBoost further enhanced performance, demonstrating the benefits of combining neural networks with powerful gradient boosting algorithms for structured data analysis.

The browser extension implementation proved to be a practical and user-friendly solution. Users received immediate alerts when encountering suspicious websites, significantly reducing the risk of falling victim to phishing attacks. The manual URL input feature added flexibility, allowing users to test links even outside active browsing sessions.

However, challenges remain. The system's accuracy depends on the quality and diversity of the training dataset. Future phishing tactics may evolve in ways that bypass current models, so continuous dataset updates and model retraining will be necessary. Despite these challenges, the study confirms that deep learning-based browser extensions can offer a scalable and highly effective defense against phishing threats, making cybersecurity more accessible to everyday internet users.

VII. CONCLUSION & FUTURE WORK

This study presented a deep learning-based approach for detecting phishing websites, with a practical implementation as a Chrome browser extension. By leveraging character-level analysis through GRU and LSTM models, and further enhancing performance with the XGBoost algorithm, the system achieved exceptional accuracy in identifying malicious URLs. The integration of the detection engine into a browser extension allows real-time alerts and user-driven URL verification, providing a fast, lightweight, and privacy-preserving security tool.

Unlike traditional rule-based methods, the proposed framework eliminates the dependency on third-party services and adapts effectively to the evolving nature of phishing attacks. The experimental results confirmed that deep learning models, particularly GRU combined with XGBoost, significantly outperform classical algorithms in both accuracy and reliability.

Overall, the solution demonstrates a scalable and user-friendly method for phishing prevention, making advanced cybersecurity accessible to everyday users. Future work will focus on enhancing dataset diversity and adapting to emerging phishing strategies.

In future iterations, the proposed framework can be enhanced by incorporating adaptive learning techniques that allow the model to retrain itself periodically with newly encountered phishing URLs, ensuring up-to-date detection capabilities. Integration with other browsers like Firefox and Edge will extend accessibility.

REFERENCES

[1] L. Tang and Q. H. Mahmoud, ''A survey of machine learning-based solutions for phishing website detection,'' Mach. Learn. Knowl. Extraction, vol. 3, no. 3, pp. 672–694, Aug. 2021, doi: 10.3390/make3030034.

[2] S. Marchal, J. Francois, R. State, and T. Engel, ''PhishStorm: Detecting phishing with streaming analytics,'' IEEE Trans. Netw. Service Manage., vol. 11, no. 4, pp. 458–471, Dec. 2014.

[3] (Jun. 2021). Phishing Activity Trends Report 1st Quarter 2021. APWG. Accessed: Oct. 20, 2021. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2021.pdf

[4] (2020). 2020 Internet Crime Report. [Online]. Available: https://www. ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf

[5] R. M. Mohammad, F. Thabtah, and L. McCluskey, ''Predicting phishing websites based on self-structuring neural network,'' Neural Comput. Appl., vol. 25, no. 2, pp. 443–458, Nov. 2013, doi: 10.1007/s00521-013-1490-z.

[6] M. A. El-Rashidy, ''A smart model for web phishing detection based on new proposed feature selection technique,'' Menoufia J. Electron. Eng. Res., vol. 30, no. 1, pp. 97–104, Jan. 2021, doi: 10.21608/mjeer.2021.146286.

[7] B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, and X. Chang, ''A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment,'' Comput. Commun., vol. 175, pp. 47–57, Jul. 2021, doi: 10.1016/j.comcom.2021.04.023.

[8] E. Gandotra and D. Gupta, ''Improving spoofed website detection using machine learning,'' Cybern. Syst., vol. 52, no. 2, pp. 169–190, Oct. 2020, doi: 10.1080/01969722.2020.1826659.

[9] W. Wang, F. Zhang, X. Luo, and S. Zhang, ''PDRCNN: Precise phishing detection with recurrent convolutional neural networks,'' Secur. Commun. Netw., vol. 2019, pp. 1–15, Oct. 2019, doi: 10.1155/2019/2595794.

[10] M. Sabahno and F. Safara, ''ISHO: Improved spotted hyena optimization algorithm for phishing website detection,'' Multimedia Tools Appl., Mar. 2021, doi: 10.1007/s11042-021-10678-6.

[11] S.-J. Bu and S.-B. Cho, ''Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing URL detection,'' Electronics, vol. 10, no. 12, p. 1492, Jun. 2021, doi: 10.3390/electronics10121492.

[12] URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Accessed: Oct. 20, 2021. www.unb.ca. [Online]. Available: https://www.unb.ca/cic/datasets/url-2016.html

[13] PhishTank > See All Suspected Phish Submissions. Accessed: Oct. 20, 2021. www.phishtank.com.[Online]. Available: https://www.phishtank.com/phish_archive.php

[14] M. Somesha, A. R. Pais, R. S. Rao, and V. S. Rathour, ''Efficient deep learning techniques for the detection of phishing websites,'' Sadhan ¯ a¯, vol. 45, no. 1, pp. Jun. 2020, doi: 10.1007/s12046-020-01392-4.

[15] M. A. Adebowale, K. T. Lwin, and M. A. Hossain, ''Intelligent phishing detection scheme using deep learning algorithms,'' J. Enterprise Inf. Manage., to be published. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/JEIM-01-2020-0036/full/html, doi: 10.1108/jeim-01-2020-0036.

[16] D. N. Atimorathanna, T. S. Ranaweera, R. A. H. Devdunie Pabasara, J. R. Perera, and K. Y. Abeywardena, ''NoFish; total anti-phishing protection system,'' in Proc. 2nd Int. Conf. Advancements Comput. (ICAC), Dec. 2020, pp. 470–475, doi: 10.1109/ICAC51239.2020.9357145.

[17] S. Maurya, H. Singh, and A. Jain, ''Browser extension based hybrid antiphishing framework using feature selection,'' Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 11, pp. 1–10, 2019, doi: 10.14569/ijacsa.2019.0101178.

[18] B. Shah, K. Dharamshi, M. B. Patel, and V. Gaikwad. (2020). Chrome Extension for Detecting Phishing Websites. Semantic Scholar. [Online]. Available: https://www.semanticscholar.org/paper/ChromeExtension-for-Detecting-Phishing-Websites-ShahDharamshi/fa99621bdc27cbd32ed799d7a6c1848ac644e8a8

[19] K. M. Sundaram, R. Sasikumar, A. S. Meghana, A. Anuja, and C. Praneetha, ''Detecting phishing websites using an efficient featurebased machine learning framework,'' Revista Gestão Inovação e Tecnologias, vol. 11, no. 2, pp. 2106–2112, Jun. 2021, doi: 10.47059/revistageintec.v11i2.1832.

[20] O. Abiodun, A. S. Sodiya, and S. O. Kareem, ''Linkcalculator—An efficient link-based phishing detection tool,'' Acta Inf. Malaysia, vol. 4, no. 2, pp. 37–44, Oct. 2020, doi: 10.26480/aim.02.2020.37.44.

[21] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

[22] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.