Adaptive Phishing Detection in Real-Time Messaging Platforms

Aaron Weng
Department of Computer Science & Engineering
Texas A&M University
College Station, TX 77843 USA
mwkyawaw@tamu.edu

***Abstract*—This project focuses on the development of a phishing detection system specifically designed for real-time messaging platforms, such as Discord, which are increasingly becoming prime targets for phishing attacks. Traditional phishing detection systems have been primarily designed for email-based threats, where the language structure, content, and attack methods are more standardized. In contrast, real-time messaging platforms pose a unique challenge due to their dynamic and informal communication styles. This system leverages advanced Natural Language Processing (NLP) techniques and adaptive machine learning models that are capable of interpreting the nuances of informal language and rapidly changing conversation contexts. The core of the system is built upon deep learning algorithms, particularly Recurrent Neural Networks (RNNs), that analyze message content, detect potential phishing attempts, and classify the messages accordingly. Furthermore, the system dynamically learns from evolving phishing patterns, continually improving its detection capabilities in response to new threats. By using a feedback loop that allows real-time learning and adaptation, the system ensures that it stays updated and capable of providing real-time protection. This approach not only improves detection accuracy but also enhances the system's ability to adapt to sophisticated phishing strategies that may emerge on messaging platforms.**

***Index Terms*—Spam, Phishing, Machine Learning, Detection, Natural Language Processing**

I. Introduction

With the growing popularity and widespread adoption of real-time messaging applications like Discord, Slack, and WhatsApp, phishing attacks have become an increasing threat in these environments. These platforms, once primarily used for socializing and gaming, have now expanded to encompass professional and business communication. As a result, they are increasingly targeted by malicious actors who deploy sophisticated phishing tactics to deceive users into revealing sensitive information, such as login credentials, payment details, and personal data. Unlike traditional phishing attacks that occur via email and follow more predictable patterns, phishing attempts on real-time messaging platforms exploit the informal, fluid, and often fast-paced nature of conversations.

This project seeks to counter this by developing an adaptive phishing detection system specifically tailored to the needs of real-time messaging platforms. By integrating Natural Language Processing (NLP) techniques, this system is able to analyze conversational context, tone, and language patterns to accurately detect phishing attempts. Additionally, the system leverages deep learning models, particularly Recurrent Neural Networks (RNNs), to capture the sequential nature of conversations and identify patterns that might indicate phishing. These models are trained to recognize subtle cues in the language and context of messages, such as suspicious links, urgent calls to action, and deceptive wording.

What sets this system apart from traditional phishing detection methods is its adaptability combined with its use in messaging applications. By utilizing adaptive machine learning models, the system is capable of continuously learning from new data, evolving phishing strategies, and user feedback. This ensures that the system can stay ahead of emerging phishing threats, providing dynamic protection in real-time. The model is designed to improve over time as it encounters more phishing attempts, allowing it to refine its accuracy and minimize false positives.

Ultimately, this project aims to offer a real-time, scalable, and adaptable phishing detection solution for modern messaging platforms, helping to protect users from evolving phishing attacks that threaten their security and privacy. By combining cutting-edge machine learning techniques with real-time adaptability, the system seeks to provide a more robust and effective defense against phishing in dynamic and informal messaging environments.

II. Related Works

Past phishing detection models rely heavily on static rule-based systems or machine learning models that primarily target email content. While effective in traditional email phishing, these approaches fail to capture the nuances of messaging platform conversations. Recent advancements include natural language processing models, but few are tailored for the rapid, context-specific interactions of real-time chat applications. Previous research typically doesn't combine both message filtering as well as URL classification in one model as well.

A. PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System (2020)

PhishHaven is a robust AI-driven system designed to detect both AI-generated and human-crafted phishing URLs in real-time [1]. The authors emphasize the escalating sophistication of phishing attacks, which now leverage deep learning systems like DeepPhish to create malicious URLs capable of bypassing traditional detection mechanisms. PhishHaven addresses this challenge using a hybrid approach that combines ensemble machine learning methods, multi-threading, and lexical feature analysis to improve accuracy and efficiency. The experimental results demonstrate that PhishHaven achieves exceptional performance, particularly against AI-generated phishing URLs, with a 98% accuracy.

B. Adaptive Phishing Detection: Harnessing the Power of Artificial Intelligence for Enhanced Email Security (2023)

"Adaptive Phishing Detection: Harnessing the Power of Artificial Intelligence for Enhanced Email Security" explores the increasing sophistication of phishing attacks and the inadequacies of traditional detection methods in combating them [2]. The authors emphasize how phishing attacks have evolved, leveraging advanced AI techniques to generate convincing fraudulent emails that evade detection systems. The proposed solution incorporates machine learning (ML) and natural language processing (NLP) to create an adaptive system capable of detecting phishing emails in real-time while continuously learning from new attack patterns.

The evaluation of the proposed system demonstrates its superior performance over traditional methods, achieving higher accuracy and adaptability. Specifically, the AI-driven approach yields a high true positive rate (TPR) and a low false positive rate (FPR), highlighting its ability to detect phishing attempts with precision while minimizing disruptions caused by misclassifications. The system's adaptive nature allows it to learn from emerging threats and dynamically improve its detection capabilities over time, addressing the limitations of static rule-based systems and blacklists.

C. PhishDetector (2016)

The PhishDetector paper proposes a novel rule-based phishing detection system tailored for internet banking websites [3]. It introduces two innovative feature sets to evaluate webpage content and identify access protocols for its resources. The first feature set assesses the relationship between webpage content and its URL using approximate string matching (Levenshtein Distance). The second feature set examines the security protocol (HTTPS) used to load webpage resources such as images, scripts, and CSS files.

The experimental evaluation highlights the model's exceptional performance, with a true positive rate of 99.14% and false negative rate of 0.86%. The sensitivity analysis demonstrates that six out of eight proposed features significantly improve classification accuracy, while the model maintains great efficiency performing its classification in 1.5 seconds on average.

III. Methodology

The initial idea was to create something that could protect users from spam and phishing attacks. There are many email platforms that have built-in filters for spam, such as Gmail. Some of these filters might even have built-in adaptation using machine learning algorithms, which has been touched on in many previous research papers. However, it is more rare to see real-time messaging platforms with safeguards against malicious messages and links, which is the direction that was decided upon in this research. The primary objectives of this study include the development of a detection system using machine learning that can dynamically adjust to threats.

A. Dataset

The spam SMS dataset, created in 2016, was obtained from Kaggle [4]. It contains 5574 messages labeled "spam" if it is spam and "ham" if it is a regular message. There is only 1 feature in this dataset, which is the text content of each message.

The phishing URL dataset, created in 2021, was also obtained from Kaggle [5]. It contains a substantial 651,191 URLs, all labeled as "benign", "phishing", "malware", or "defacement". To simplify the model, everything besides a benign URL was treated as a harmful phishing URL, making the classification binary. Similar to the spam SMS dataset, the 1 feature is the text content of each URL.

B. Idea and System

The core of this project is to develop an adaptive phishing detection system specifically tailored for real-time messaging platforms (mainly focused on Discord), addressing the increasing threat of phishing attacks in these environments. While many email services like Gmail have implemented advanced spam filters that leverage machine learning algorithms to adapt over time, similar protections are notably lacking in instant messaging applications. This gap presents a critical area for research and development.

The proposed system will utilize machine learning techniques, particularly natural language processing (NLP) and deep learning, to identify spam messages and mitigate phishing attempts in real time. By analyzing messages and links sent within platforms like Discord, the system aims to detect malicious content that traditional filters may overlook due to not having real-time updates.

C. Design

The design of the system integrates two key components: a spam detection model for textual messages and a URL classification model to identify malicious links. These models were developed using a combination of deep learning and traditional machine learning techniques to ensure robust performance and scalability.

For spam detection, the SMS Spam Collection dataset was used, which contains labeled SMS messages. The data was preprocessed by mapping the labels to binary values, with "spam" as 1 and "ham" as 0. The text data was tokenized using TensorFlow's Keras Tokenizer, converting the SMS messages into integer sequences. These sequences were padded to a uniform length of 100 tokens to ensure compatibility with the neural network model. A Recurrent Neural Network (RNN) with Bidirectional Long Short-Term Memory (LSTM) layers was selected for its ability to capture contextual relationships in text data. The model architecture included an embedding layer to learn word representations, followed by two Bidirectional LSTM layers—one with 64 units (returning sequences) and another with 32 units—paired with a dropout layer to reduce overfitting. The model concluded with a dense layer using a sigmoid activation function for binary classification.

$$\text{Input} \xrightarrow{\text{Embedding}} \text{Bidirectional LSTM}_{64} \quad (1)$$

$$\xrightarrow{\text{Dropout}} \text{Bidirectional LSTM}_{32} \xrightarrow{\text{Dense}} \text{Sigmoid Activation} \quad (2)$$

Fig. 1. The sequence of steps of the RNN architecture.

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (3)$$

Fig. 2. The Sigmoid Activation Function, which maps any real-valued input to a value between 0 and 1. It is commonly used for binary classification tasks.

The spam detection model was trained using the binary cross-entropy loss function and the Adam optimizer over five epochs with a batch size of 32. After training, the model was saved as spam_model.h5 for deployment. Additionally, a fine-tuning mechanism was implemented to allow the model to be updated with new feedback data, enabling continuous learning through additional training.

For URL classification, the Malicious URLs dataset was utilized, where malicious categories, including phishing, malware, and defacement, were mapped to 1, while benign URLs were labeled as 0. The preprocessing stage involved removing invalid or missing entries and converting URLs into numerical features using Scikit-learn's HashingVectorizer. This method uses character-level n-grams (3-4 characters) to tokenize URLs and map them into a fixed 5000-dimensional feature space. The dataset was then split into training and testing sets using a 70-30 split. A Stochastic Gradient Descent (SGD) Classifier with a log-loss function was selected for binary classification due to its efficiency with large datasets.

$$\text{Log-Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right) \quad (4)$$

Fig. 3. The log-loss (or binary cross-entropy) function used for binary classification, where y_i is the true label and $\hat{y}_i$ is the predicted probability of the sample being in class 1. This function is used in training the SGDClassifier to minimize the error between predicted probabilities and true labels.

The model was trained with a maximum of 800 iterations to ensure convergence. The trained classifier was saved as phishing_model.pkl. To support real-time updates, an incremental learning mechanism was implemented using the partial_fit method, allowing the classifier to incorporate new feedback data without retraining from scratch.

D. Classification Models

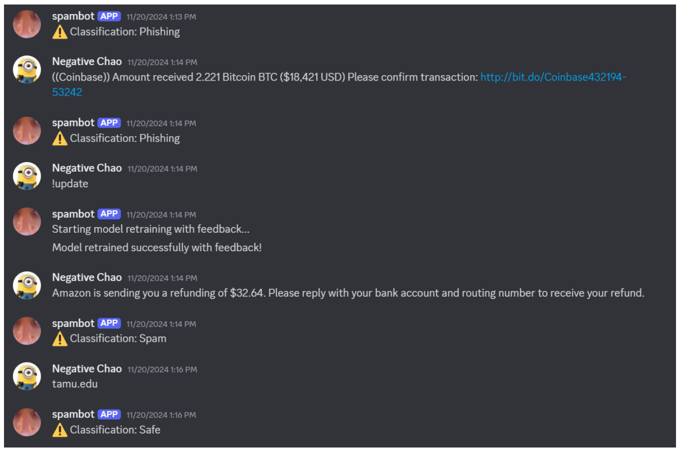
- RNN (Recurrent Neural Network) - The spam detection model leverages a Bidirectional LSTM architecture, which is well-suited for analyzing sequential text data. The Bidirectional LSTM captures contextual patterns by processing text both forward and backward, ensuring that relationships between words are fully utilized. The model's embedding layer learns meaningful word representations, while the dropout layer reduces overfitting during training. This deep learning model demonstrated strong performance after five epochs of training, effectively distinguishing between spam and non-spam messages.
- SGD (Stochastic Gradient Descent) - For URL classification, the HashingVectorizer was used to convert URLs into a numerical format suitable for machine learning algorithms. The SGDClassifier, optimized with a log-loss function, was chosen for its scalability and ability to provide probabilistic outputs for binary classification. The classifier's incremental learning capability via partial_fit allows it to adapt to new data efficiently, ensuring that the system remains up-to-date with evolving URL threats.
- Combined Model - By combining deep learning for text analysis with a lightweight, scalable machine learning approach for URL classification, the system balances performance, accuracy, and efficiency. If there is text without a definable URL, then the combined approach only uses the spam prediction model to determine the message category. If there is just a text along with a URL or just a URL, then both models are used, with the spam model having 0.4 weight and phishing model having 0.6 weight. The idea is that if a dangerous URL is used, then it doesn't matter if the text surrounding it is considered harmless. The spam classifier looks at all text including the URL, while the phishing classifier only looks at the URL in these cases. The final decision is a weighted combination of phishing and spam probabilities determines whether the message is phishing, spam, or ham.

E. Additional Framework

The Python Flask API handles some front-end and the back-end logic for spam and phishing detection. It includes endpoints to classify messages and URLs allowing for requests to be made in external applications to use the classifier models.

A Discord application was set up to use the API for classification in real time. This was the most straightforward way to get real results from the model on a messaging platform. In Discord, the functionality of an application comes in the form of a bot that can read messages, give responses, and accept commands. To allow for adaptation, commands were enabled to give feedback to the bot's predictions. For example, if a user types "yes phishing" or "no phishing" in response to a prediction in Discord, the feedback classification (phishing, in this case) will be added to an external file that will be used to incrementally retrain the model when "!update" is used in

Discord. Because the model already uses a great deal of data (especially the URLs), any feedback is repeated 100 times in the external file to make it more effective when retraining.



spambot APP 11/20/2024 1:13 PM
⚠ Classification: Phishing

Negative Chao 11/20/2024 1:14 PM
((Coinbase)) Amount received 2.221 Bitcoin BTC ($18,421 USD) Please confirm transaction: http://bit.do/Coinbase432194-53242

spambot APP 11/20/2024 1:14 PM
⚠ Classification: Phishing

Negative Chao 11/20/2024 1:14 PM
!update

spambot APP 11/20/2024 1:14 PM
Starting model retraining with feedback...
Model retrained successfully with feedback!

Negative Chao 11/20/2024 1:14 PM
Amazon is sending you a refunding of $32.64. Please reply with your bank account and routing number to receive your refund.

spambot APP 11/20/2024 1:14 PM
⚠ Classification: Spam

Negative Chao 11/20/2024 1:16 PM
tamu.edu

spambot APP 11/20/2024 1:16 PM
⚠ Classification: Safe

Fig. 4. The bot making classifications and updating the models.

IV. Study & Evaluation

To evaluate the combined model, the predictors are loaded to analyze each dataset individually using the combined predictions outlined previously. This uses the bidirectional LSTM model, the SGD classifier, as well as a tokenizer used to process inputs. The first evaluation uses the spam dataset, with labeled SMS messages categorized as spam or ham and predicting the probability that a message is spam. True labels and predicted labels are compared to generate a confusion matrix as well as a classification report.

Spam Dataset Results

Confusion Matrix

Actual / Predicted	**Spam (1)**	**Ham (0)**
Spam (1)	475	253
Ham (0)	45	4780

Fig. 5. The confusion matrix for the combined model on the spam dataset.

Classification Report

Class	**Precision**	**Recall**	**F1-score**	**Support**
Spam	0.91	0.64	0.75	747
Ham	0.95	0.99	0.97	4825
Macro avg	0.93	0.81	0.86	5572
Weighted avg	0.94	0.94	0.94	5572

Fig. 6. The classification report for the combined model on the spam dataset.

The combined model has good precision, but unfortunately, does not have a great recall when classifying spam in comparison to its precision on the same set. This is also evident when looking at how many spam messages are incorrectly classified as ham. However, it is important to note that there is a great deal of variation when it comes to classification accuracy considering that the models can go through numerous retraining processes when adapting. The first fix to this instance of poor recall would be to add each incorrect classification to the feedback dataset and retrain.

Next, we will find the evaluate the combined classifier using the phishing URL dataset. For this, it was decided to sample only 2% of the original dataset (which was still more than 10,000 URLs), otherwise the evaluation process would take an impractical amount of time. Like before, true labels and predicted labels are compared to generate a confusion matrix as well as a classification report.

Phishing Dataset Results

Confusion Matrix

Actual / Predicted	**Phishing (1)**	**Ham (0)**
Phishing (1)	2912	1471
Ham (0)	99	8542

Fig. 7. The confusion matrix for the combined model on the phishing dataset.

Classification Report

Class	**Precision**	**Recall**	**F1-score**	**Support**
Phishing	0.97	0.66	0.79	4383
Ham	0.85	0.99	0.92	8641
Macro avg	0.91	0.83	0.85	13024
Weighted avg	0.89	0.88	0.87	13024

Fig. 8. The classification report for the combined model on the phishing dataset.

Like for spam classification, our precision is good but it seems recall is a big weakness for our combined model. We still must take note that there are large variations in the model's effectiveness based on retraining and adaptation on feedback data as well. This time, there is also variations in the sample itself because the sample is such as small part of the original dataset. To improve our recall, we can retrain on misclassified data points with increased weight (similar to our feedback adaptation process) to make sure the model classifies them correctly next time.

V. Conclusion

For this project, inspiration was taken from PhishHaven's methodology with the addition of extending it to real-time messaging platforms like Discord [1]. While PhishHaven focuses primarily on phishing URLs, the system outlined in this paper integrates text-based spam detection using an RNN with Bidirectional LSTMs and URL classification using an SGDClassifier. By combining deep learning and traditional machine learning, we aimed to achieve similar levels of accuracy and efficiency for dynamic, informal communication channels. PhishHaven's success highlights the importance of robust feature engineering, which was partially incorporated in this system through tokenization for text analysis and HashingVectorizer for URL processing.

PhishHaven demonstrates the effectiveness of hybrid approaches in addressing evolving phishing threats. Our system builds upon these principles, adapting them to the unique challenges posed by real-time messaging platforms while ensuring both spam and phishing content are accurately identified and mitigated.

The adaptive learning concept presented in the study by Andriu is also mirrored in our system's design [2]. However, our method also expands on the usable area of detection by allowing use in messaging platforms. By enabling continuous retraining of both models using new feedback data, our system ensures that it stays updated with evolving phishing patterns. Additionally, Andriu's focus on minimizing false positives aligns with this implementation, where weighted predictions balance the results from URL and message classifiers in an attempt to provide more accurate final decisions.

PhishDetector's emphasis on real-time detection and its independence from third-party services closely aligns with this project's goals of creating a dynamic phishing detection system for real-time messaging platforms [3]. While PhishDetector focuses on browser-based detection for internet banking, this study extends the concepts used in PhishDetector to messaging platforms like Discord. The use of NLP for message analysis and a hybrid machine learning approach mirrors PhishDetector's ability to handle evolving phishing strategies effectively.

Finally, PhishDetector demonstrates the importance of using novel features and rule-based systems to address zero-day phishing attacks, reinforcing the significance of adaptation in real-time models like the one proposed in this study.

Future work will focus on further refining the model, expanding the dataset to include more data as well as a wider variety of phishing tactics, and improving the system's efficiency and accuracy. Additionally, the application can be improved to be hosted on an online service rather than on a local machine. We hope for better adaptability, automatically updating models when given feedback without needing to use a command for updates. Also, a Discord bot might not be necessary, as the classification should be able to run through a user interface given further work. A final improvement can come through the system's efficiency, since the model is reloaded every time it is called to classify by API to keep it dynamic. Ultimately, this project contributes to the broader field of cybersecurity by pioneering a comprehensive approach to detecting and preventing phishing in dynamic messaging environments, ensuring users are better protected against evolving threats.

VI. Updates from Progress Report

Things that were added since the progress report:

- Improved length of abstract and introduction
- Added detail to related work section, including results
- All figures (there were no figures on the progress report)
- Evaluation steps and results (also includes new code file eval.py)
- Conclusion elaborated upon, linked current work to related work
- New citations as well as corrected reference citations
- In-line citations added to related work section and conclusion

References

[1] Sameen, Maria, Kyunghyun Han, and Seong Oun Hwang. "PhishHaven—An efficient real-time AI phishing URLs detection system." IEEE Access 8 (2020): 83425-83443.

[2] Andriu, Adrian-Viorel. "Adaptive Phishing Detection: Harnessing the Power of Artificial Intelligence for Enhanced Email Security." Romanian Cyber Secur. J 5.1 (2023): 3-9.I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Mahmood Moghimi, Ali Yazdian Varjani. "New rule-based phishing detection method." Expert Systems with Applications, Volume 53, 2016, Pages 231-242, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2016.01.028.

[4] Kaggle, https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset/data.

[5] Kaggle, https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset.