Malicious URL Detection with Deep Learning

1st Kenisha Stills
Computer Science
Hamilton College
kstills@hamilton.edu

***Abstract*—This proposal outlines the datasets, tools, and model architectures selected for a malicious URL detection NLP project. The project uses Deep Learning specifically focusing on Bidirectional RNNs to classify URLs as benign, defacement, malware, or phishing.**

***Index Terms*—Bi-LSTM, Bi-GRU, RNN, NLP, Cybersecurity**

I. Introduction

Malicious URLs pose serious problems for people, businesses, institutions, and organizations connected to the web. The urls of mass phishing campaigns sent from VPN-cloaked servers, impersonate corporate identities to steal personally identifiable information or access critical data of the target. They're used to hijack and deface company or government websites, rendering critical data inaccessible and inoperable. Malware could slip into software download from a suspicious website, providing another point of access for hackers to break into a network. These urls if ever clicked or visited even for a second could do grave damage and irreparable harm to anyone online.

The current standard methods for detecting malicious urls (i.e malware, phishing, and defacement) rely on blacklists — databases of confirmed bad urls — and heuristics — manually formulated rules to identify bad urls based on an analysis of confirmed bad urls. But, these techniques for detecting malicious urls. Detecting malicious urls based on a pre-defined set of characteristics and a database of collected bad urls proves unreliably effective against attackers because they constantly adjust the patterns of their urls and bypass heuristic rules.

Cybersecurity and AI researchers have been exploring new and more robust methods in the area of machine learning and deep learning that predict malicious urls more reliably and rapidly. Deep learning models still rely on large and frequently updated datasets, but the trade off for enhanced malicious url detection capabilities could better support corporations, organizations, and governments working around the clock to respond to increasingly challenging and evolving cybersecurity threats.

A. Deep Learning Frameworks and Technologies

- **PyTorch** – The popular framework used for building deep learning architectures and handling tensor operations.
- **Pandas** – Used for CSV data ingestion, dataframe manipulation, and merging multiple datasets (Tranco, OpenPhish, etc.).
- **Lightning Module** – Used for organizing the training loop and visualizing training progress, accuracy, and loss metrics.
- **scikit-learn** – Offers utilities for dataset splitting (train/test/val), label encoding, and calculating performance metrics like F1-score.
- **kagglehub** – Simple API for accessing Kaggle datasets.

B. Model Architecture

I have chosen architectures optimized for sequential data processing: Bi-LSTM and Bi-GRU.

- **Bidirectional LSTM (Bi-LSTM):** Processes the URL character sequence in both forward and backward directions, allowing the model to capture context from both the start (protocol/subdomain) and end (TLD/path) of the URL.
- **Bidirectional GRU (Bi-GRU):** Similar to LSTM but computationally more efficient. It is ideal for detecting patterns in long URL strings without the vanishing gradient problem.

These architectures lend themselves well to analyzing URLs because URLs are structured sequences of characters where the relationship between adjacent characters (like "http", ".exe", or randomized strings) determines malicious intent.

C. Related Research

The following study provided inspiration for the project:

- Rafsanjani et al. [1] propose a framework leveraging priority coefficients for feature evaluation in malicious URL detection.

D. References

(1) A. S. Rafsanjani, N. Binti Kamaruddin, M. Behjati, S. Aslam, A. Sarfaraz and A. Amphawan, "Enhancing Malicious URL Detection: A Novel Framework Leveraging Priority Coefficient and Feature Evaluation," in IEEE Access, vol. 12, pp. 85001-85026, 2024.

(1) S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

(2) K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734, 2014.

II. Data Collection

A. Public Datasets

I have aggregated data from sources routinely used by cybersecurity professionals:

- **Kaggle Malicious URL Dataset**: A collection of labeled URLs for multi-class classification. https://www.kaggle.com/
- **Openphish**: Provides up-to-the-minute phishing URL feeds and a 30-day archive. https://openphish.com/phishing_feeds.html
- **URLHaus**: Provides a full database dump of websites actively distributing malware. https://urlhaus.abuse.ch/api/
- **Tranco**: Maintains an active list of benign (safe) URLs, accessible via the Tranco Python library. https://pypi.org/project/tranco/

I have sampled approximately 60,000 URLs collectively across these four datasets.

B. Self-Generated Data Collection

To test the robustness of the model against novel attacks, I am using Generative AI to create synthetic ”fake” URLs that mimic malicious url patterns (e.g., slightly misspelled domain names) to serve as an adversarial test set.

III. Preprocessing the Data

Since URLs are text data, the preprocessing pipeline focuses on cleaning and normalizing the urls.

- **URL Normalization**: Converting all urls to lowercase, removing the protocol (i.e. www), and trailing spaces
- **IP Address Filtering**: Removed urls with IP address hostnames
- **Label Encoding**: Converting the target labels (i.e, ”benign”, ”phishing”, ”malware”) into numerical values for multi-class classification.
- **Class Balancing**: Balance all classes to equal sample counts
- **Tokenization**: Breaking down the URL string into individual characters or sub-words. This converts a string like ”google.com” into a list of tokens.
- **Vocabulary Building**: Creating a mapping of unique characters to integers. Rare characters may be replaced with an ”unknown” token to reduce noise.
- **Padding and Truncation**: Neural networks require inputs of a fixed size. Short URLs are padded with zeros, and extremely long URLs are truncated to a maximum sequence length (e.g., 50 characters).

IV. Data Examples

Below are examples of the data structure from the collected sources (Table I).

TABLE I
EXAMPLES OF COLLECTED URL DATA

Source	**URL Sample**	**Label**
Tranco	*google.com*	Benign
URLHaus	*http://192.168.1.1/malware.exe*	Malware
Kaggle	http://www.protect-effect.nl/diensten/62-telefonie	Defacement
OpenPhish	*http://secure-login-apple.id.com*	Phishing
AI Generated URL.	*www.amaz0n-security-check.com*	Phishing