Malicious URL Classification: Machine Learning and LLM Approaches

Group Members: **Hakim Ali, Qasim Saeed**

March 27, 2025

1 Introduction

The rapid expansion of the internet has facilitated a surge in cyber threats, with malicious URLs serving as a major attack vector for phishing, malware distribution, and spam. Traditional blacklisting methods have limited effectiveness due to the dynamic nature of cyber threats. This necessitates the use of machine learning (ML) and large language models (LLMs) to classify URLs more accurately based on extracted patterns.

In this study, we evaluate multiple approaches, including traditional ML models, deep learning architectures, and transformer-based LLMs, to assess their effectiveness in malicious URL classification. A critical aspect of our evaluation includes analyzing the training convergence time, computational demands, and performance across various feature extraction techniques.

2 Dataset and Preprocessing

2.1 Data Merging and Cleaning

The dataset used for training was constructed by merging two publicly available malicious URL datasets to create a five-class classification problem with labels: benign, defacement, phishing, malware, and spam. Merging datasets introduced inconsistencies such as missing values and duplicate records, which were handled using the following steps:

- Removal of duplicate entries to prevent data leakage.
- Cleaning of URLs by normalizing case sensitivity and removing unnecessary URL parameters.
- Handling missing values by applying domain-specific rules for feature extraction.

2.2 Feature Engineering

Feature extraction played a crucial role in model performance. We derived three types of features:

1. **Lexical and Structural Features:** Extracted features such as URL length, number of special characters, presence of numerical sequences, number of subdomains, and presence of suspicious keywords.
2. **NLP-Based Embeddings:** - TF-IDF vectorization was applied to transform URLs into numerical representations. - Word2Vec embeddings were trained on the dataset to capture word relationships within URLs. - Transformer-based embeddings (e.g., BERT, DistilBERT) were used to extract contextual representations.
3. **Sequence-Based Features:** - URLs were treated as character sequences, allowing deep learning models such as LSTMs to learn sequential patterns.

To address class imbalance, we applied oversampling (SMOTE) and undersampling techniques to balance the minority and majority classes.

3 Exploratory Data Analysis (EDA)

Exploratory data analysis was conducted to understand the distribution of URL attributes, frequency of malicious patterns, and correlations between features. Key insights included:

- Malicious URLs tend to be longer and contain more special characters than benign URLs.
- Phishing URLs frequently mimic legitimate domain structures but contain slight spelling variations.
- Malware and spam URLs exhibit a higher number of subdomains.

Figure 1 provides an overview of the distribution of URL lengths across different categories.

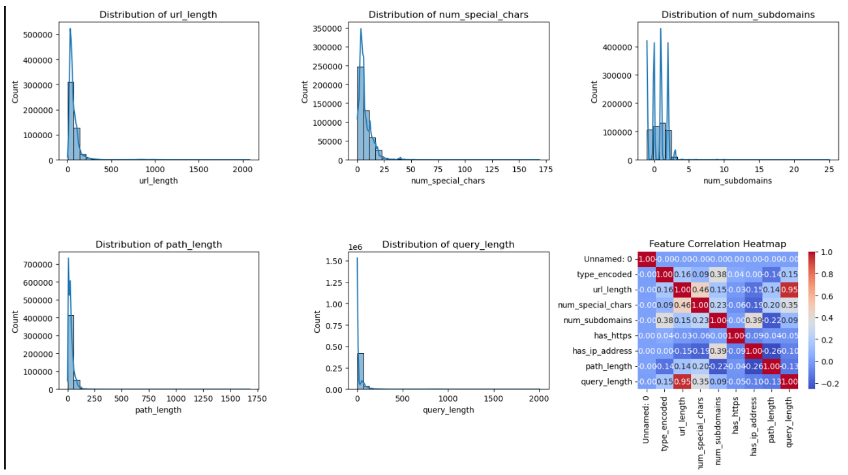

Figure 1: Exploratory Data Analysis (EDA) on URL dataset

4 Model Selection and Training

Three types of models were implemented for classification:

4.1 Traditional ML Models

- **Random Forest:** Utilized an ensemble of decision trees to capture non-linear relationships. - **Logistic Regression:** A linear model that served as a strong baseline due to its interpretability.

4.2 Deep Learning Model

- **LSTM with Embeddings:** - Implemented a character-level LSTM model using pre-trained embeddings (TF-IDF and Word2Vec). - Captured sequential dependencies within URL structures to improve classification accuracy. - Required extensive hyperparameter tuning to balance convergence speed and overfitting.

4.3 LLM-Based Model

- **Transformer-Based Approach (Fine-Tuned BERT Model):** - Utilized a transformer model pre-trained on web-based textual data. - Fine-tuned on our dataset to leverage contextual understanding of URLs. - Required high computational power but achieved faster convergence due to prior knowledge from pretraining.

5 Training and Convergence Analysis

5.1 Computational Challenges

One of the major challenges encountered was the training time and computational cost associated with different models:

- **Traditional ML models** (Random Forest, Logistic Regression) converged quickly within minutes, requiring minimal computational resources.
- **LSTM models** took significantly longer (several hours) to converge, even when using embeddings. Despite hyperparameter tuning, achieving optimal performance required fine-tuning dropout rates, sequence lengths, and batch sizes.
- **LLM-based models** required extensive GPU resources due to their transformer architecture. Although fine-tuning took fewer epochs compared to LSTMs, the total computation cost was high because of the pretraining-finetuning paradigm.

5.2 Performance Comparison

Table 1 summarizes the performance of all models:

5.3 Results Visualization

Figure 2 shows the ROC curves for different models.

Model	Accuracy	Precision	Recall
Random Forest	94.3%	93.4%	94.02%
Logistic Regression	90.6%	90.02%	90.16%
LSTM with Embeddings	90.32%	90.23%	90.4%
Final LLM Model	90.01%	90.3%	90.4%

Table 1: Model Performance Comparison

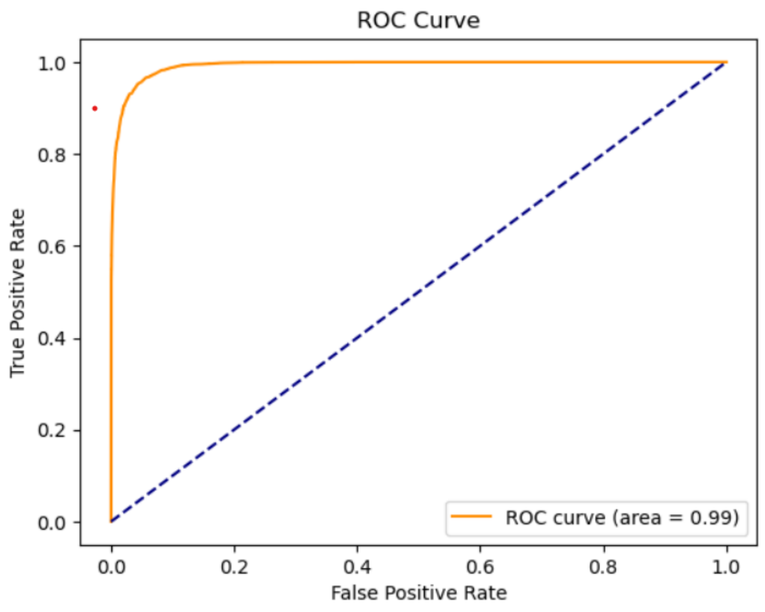

Figure 2: ROC Curve Comparison of Models

6 Critical Analysis

- While traditional ML models provided high accuracy, they lacked deep contextual understanding. - LSTMs required extensive training but effectively captured sequential URL dependencies. - The fine-tuned LLM model leveraged its prior knowledge, leading to rapid convergence despite its computational intensity. - Transformer-based embeddings significantly improved feature representations, making the LLM model superior in contextual analysis.

7 Conclusion and Future Work

This study demonstrated that while traditional ML models are computationally efficient, deep learning models offer greater flexibility in handling URL structures. LLM-based models, despite their high resource requirements, achieve faster convergence due to pre-training.

Future improvements include:

- Exploring hybrid models that combine traditional ML with transformer-based embeddings.
- Expanding datasets to improve generalization across unseen malicious patterns.

- Deploying real-time detection systems with adaptive model updates.