Phishing URL Detector Report

A Comprehensive Analysis of the Phishing Detection System.

Prepared on June 4, 2025

1. Introduction

This document provides detailed overview of the **Phishing URL Detector**, a python-based application designed to identify potentially malicious URLs using a combination of machine learning, feature extraction and third-party API integrations. The system leverages structural URL analysis, a Random Forest model, and external security APIs to deliver robust phishing detection capabilities.

2. System Overview

The phishing URL detector is built using python and the flask web framework. It combines local machine learning predictions with external API checks to classify URLs, as SAFE, SUSPICIOUS, or PHISHING. Key components include:

- **Feature Extraction:** Analyses URL structure for suspicious patterns.
- **Machine Learning:** Uses a Radom Forest Classifier for local predictions.
- **API Integration:** Incorporates VirusTotal and Google Safe Browsing for enhanced analysis.
- **Web Interface:** Provides a user-friendly Flask-based interface and API endpoint.

3. Dataset

Source and loading

The system utilizes the dataset Phishing_legitimate_full.csv, which contains labelled URL dataset for training and evaluation. The dataset is loaded and processed as follows:

- **File check:** if a pre-cleaned version Phishing_dataset_clean.csv exists, it is loaded directly.
- **Processing:** Numeric columns are selected, missing values are dropped, and the cleaned dataset is saved as phishing_dataset_clean.csv.
- **Label:** The target variable is CLASS_LABEL, where 1 typically indicates phishing and 0 indicates legitimate URLs.

Data Splitting

The dataset is split for training:

- **Train-Test Split:** 80% training, 20 % testing (random state = 42).
- **Scaling:** Feature are standardized using Standard Scaler from scikit-learn.

4. Feature Extraction

URLFeatureExtractor Class

The URLFeatureExtractor Class extracts structural and security-related features from URLs, including:

- **Basic Features:** Number of dots, dashes, underscores, URL length, presence of @ or other symbols including special characters, etc.

  Domain Features: Hostname length, path length, query length, and IP address detection.
- **Security Features:** Checks for double slashes, random strings, embedded brands names, and suspicious keywords (e.g., “login”, “verify”, “PayPal”,).

Suspicious keywords

The system flags URLs containing sensitive words:

- Keywords: http, login, verify, account, update, security, banking, etc.

Default Features

For robustness, default values are set for optional features (e.g., PctExtHyperlinks = 0.5, ExtFavicon = 0).

5. Machine Learning Model

Model Type

The system employs a RandomForestClassifier from scikit-learn with the following parameters

- n_estimators: 150
- max_depth: 12
- min_samples_leaf: 1
- class_weight: balanced
- random_state: 42
- n_jobs: -1 (uses all available cores)

Training Process

Loading: attempts to load pre-trained model, scaler, and feature names from phishing_model.pkl, scaler.pkl, and feature_names.pkl.

Training: If files are absent, trains a new model, scales features, and saves the model and scaler.

Evaluation: Reports training and test accuracy.

Prediction

The model predicts phishing probability by:

1. Extracting URL features.
2. Scaling feature using the training Standard scaler.
3. Predicting with the Random Forest model and returning a probability score.

6. API Integration

Third-Party APIs

The system integrates the following APIs for enhanced detection:

VirusTotal API

- **Key:** Loaded from environment variable VIRUSTOTAL_API_KEY.
- **Function:** Checks URL reputation via https://www.virustotal.com/api/v3/urls.
- **Process:** Retrieves existing reports or submits URLs for analysis.
- **Output:** Malicious, suspicious, harmless, and undetected counts from security engines.
- **Caching:** Users Iru_cache with a size of 1000 to reduce redundant calls.

Google Safe Browsing API

- **Key:** Loaded from environment variable GOOGLE_API_KEY.
- **Function:** Checks for malware, social engineering, and unwanted software via a HTTP request POST https://safebrowsing.googleapis.com/v4/threatMatches:find
- **Output:** Boolean indicating if the URL is flagged as malicious.

Configuration

- **API Keys:** Stored in a .env file and loaded via dotenv.
- **Retries:** Maximum of 3 attempts per requests.

Hybrid Check

The hybrid check function combines:

- Local model predictions.
- VirusTotal results (weighted by malicious and suspicious counts).
- Google Safe Browsing results (highest weight).
- Heuristic checks (e.g., brand impression, no HTTPS, suspicious TLDs).

The final decision (SAFE, SUSPICIOUS, PHISHING) is based on a weighted confidence score.

7. Web Application

Framework

Built using Flask, the application offers:

Routes:

- / and / index: Renders the homepage (index.html).
- /scan (POST): Processes URL scans and displays results (results.html).
- /api/ scan (POST): JSON- based API endpoint for programmatic scans,

Templates: Uses Flask's render_template for user interface.

Error Handling

- Ensures URLs are valid and prepends http:// if needed.
- Returns user-friendly error messages for invalid inputs or scan failures.

Conclusion

The phishing URL Detector combines structural URL analysis, a Random Forest machine learning model, and third-party API checks such as VirusTotal and Google Safe Browsing to provide accurate and reliable phishing detection. The dataset Phishing_legitimate_full.csv from Kaggle enables robust training, while the Flask application delivers an accessible interface for users and developers.