Hybrid Phishing Detection System – Project Documentation

Jagadeesh Kommineni

Hybrid Phishing Detection System – Project Documentation

Project Title

Hybrid Phishing Detection System Using Rule-Based Logic and Machine Learning

---

Table of Contents

1. Overview
2. Objective
3. Technologies and Libraries Used
4. Key Features
5. System Architecture
6. Code Breakdown
7. User Interface Design
8. Security Considerations
9. Future Enhancements
10. Conclusion

---

1. Overview

This project is a **desktop-based phishing URL detection tool** that combines **rule-based heuristics** and a **machine learning model** to determine the legitimacy of URLs entered by users. The system provides an interactive interface, processes URL characteristics, and delivers clear, color-coded verdicts based on its analysis.

---

2. Objective

The primary objective of this project is to:

- Build a real-time phishing detection tool for educational and practical security use cases.
- Integrate both **heuristic (rule-based)** and **data-driven (machine learning)** techniques to increase detection accuracy.

- Offer an intuitive GUI for non-technical users to assess website safety.
- Ensure modular, maintainable, and scalable design for future integration with browsers or larger systems.

---

3. Technologies and Libraries Used

Technology/Library	Purpose
Python	Programming language
Tkinter	GUI creation
re (Regex)	URL pattern matching
urllib.parse	URL parsing and analysis
joblib	Model serialization/deserialization
scikit-learn (used in model)	ML model training (external)
os	Path and file validation

---

4. Key Features

- **Hybrid Detection Logic**: Combines handcrafted rules and trained ML models.
- **Interactive GUI**: User-friendly interface using Tkinter.
- **URL Validation and Warning System**: Flags suspicious domains and IP-based URLs.
- **Machine Learning Prediction**: Uses pre-trained model features for prediction.
- **Failsafe Integration**: Warns users if the model is missing.

---

5. System Architecture

[User Input (URL)]
↓
[Rule-Based Heuristics]
↓
[Feature Extraction for ML Model]
↓
[Machine Learning Prediction]
↓
[Final Decision Logic]
↓
[Result Display in GUI]

---

6. Code Breakdown

a. rule_based_check(url)

This function flags suspicious URLs using manually defined rules:

- Uses IP address in domain
- Excessive URL length
- Use of '@' character
- Presence of hyphens
- Not using HTTPS
- Excessive dot count
- Double slashes in non-standard locations

If **3 or more rules** are matched, the URL is marked suspicious.

---

b. extract_ml_features(url)

Extracts numerical features required for the machine learning model:

- Length of URL
- IP-based URL check
- '@' symbol presence
- Hyphen in domain
- HTTPS usage
- Count of periods in domain

Returns a list of numerical features as input to the classifier.

---

c. model = joblib.load(...)

Loads the pre-trained machine learning model. If not found, an error popup appears to instruct the user to train the model using a separate script (train_model.py).

---

d. detect_url()

The main detection function:

- Collects the URL
- Runs both rule-based and ML detection
- Combines results to determine severity:
    - High risk: Both rule-based and ML detect as phishing
    - Moderate risk: Either rule-based or ML detects phishing
    - Safe: Both rule-based and ML mark as safe
- Displays a text verdict in color (red for phishing, orange for suspicious, green for safe)

---

7. User Interface Design

The UI has a professional dark theme and consists of:

- **Title Label**: Clearly defines purpose of the app.
- **URL Entry Field**: For user input with enhanced visibility.
- **Analyze Button**: Initiates detection logic.
- **Result Label**: Displays verdict with appropriate color (green, orange, red).
- **Font**: Monospaced 'Consolas' for a technical look.

All widgets are styled for consistency and accessibility, with a strong emphasis on clarity and minimalism.

---

8. Security Considerations

- **No external network requests** are made, ensuring the tool is safe for offline use.
- The tool is non-invasive and performs static URL analysis only.
- This makes it suitable for use in restricted, corporate, or academic environments without privacy concerns.

---

9. Future Enhancements

Feature	Description
Online Phishing Database Integration	Cross-check with real-time phishing databases like PhishTank
URL Shortener Expansion	Unwrap shortened URLs before analysis
Browser Extension Integration	Convert into a Chrome/Firefox add-on
GUI Enhancements	Add copy-paste buttons, history tracking, and alerts

Feature	Description
Confidence Score	Display prediction probability alongside decision
Model Training UI	Add a module to train models from within the GUI

10. Conclusion

This project combines the power of **heuristics** and **machine learning** in a neatly packaged GUI-based solution. It reflects a solid understanding of:

- Cybersecurity threats like phishing
- Natural URL behavior patterns
- Python GUI development with Tkinter
- Feature engineering and binary classification

The hybrid detection system demonstrates a practical application of theory in the domain of web safety and can be further extended into a full-scale security tool for browsers, SOCs, or corporate networks.