PHISHING & CYBER RISK SCORE DETECTION

CONCEPTS

About This Document

This Conceptual Study is part of a broader portfolio series designed to bridge the gap between technical execution and strategic understanding. While the main documentation explains *what* was built and *how*, this companion document explores the deeper *why* behind it.

You'll find here:

- Foundational concepts that support the project’s methodology
- Business relevance and real-world applications
- Algorithm intuition and implementation logic
- Opportunities for extension and learning paths

Whether you're a curious learner, a recruiter reviewing domain expertise, or a professional looking to adopt similar methods — this document is meant to offer clarity beyond code.
If you're eager to understand the reasoning, strategy, and impact of the solution — you're in the right place.

Happy Learning!

Introduction

Cybersecurity has become one of the most critical domains in modern technology, where organizations are not only tasked with detecting threats but also predicting and prioritizing their response to potential risks. Two of the most prevalent threats in this ecosystem are **phishing emails**—crafted messages designed to trick users into revealing confidential information—and **software vulnerabilities**, often rated using the CVSS (Common Vulnerability Scoring System) framework, that expose systems to exploitation.

This project merges these two domains under one AI-powered umbrella, building:

- An NLP-based phishing detection model for identifying harmful emails.
- A vulnerability risk scoring system that predicts the severity of software threats based on CVSS parameters.

The goal of this conceptual study is to explain the **underlying technical logic, architectural design, modelling rationale, and future implications** of this project—beyond implementation-level documentation.

1. NLP and Machine Learning for Phishing Email Detection

1.1. Text Vectorization with TF-IDF

Natural Language Processing begins by converting raw unstructured email text into structured numerical representations. This is achieved using **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorization:

- **Term Frequency** measures how frequently a term appears in a document.
- **Inverse Document Frequency** scales down terms that appear frequently across many documents (i.e., common words).
- Together, TF-IDF emphasizes rare yet potentially meaningful words, which are often critical in phishing detection (e.g., "click", "account suspended", "urgent").

This transformation produces a **sparse, high-dimensional feature matrix**, where each email becomes a vector of weighted word frequencies.

1.2. Supervised Classification with Random Forest

A **Random Forest Classifier** is used due to its interpretability, low tuning overhead, and effectiveness with high-dimensional, noisy text data. Random Forest:

- Is an ensemble of decision trees.
- Handles non-linearity and interactions between features.
- Supports feature importance, which aids explainability.

The model is trained to distinguish between legitimate and phishing emails using the vectorized input matrix and binary labels.

1.3. Handling Imbalanced Data with SMOTE

Phishing datasets are typically imbalanced—legitimate emails significantly outnumber phishing ones. This imbalance can cause the model to underperform on minority (phishing) classes.

To counter this, **SMOTE (Synthetic Minority Over-sampling Technique)** is used:

- Generates synthetic samples of the minority class by interpolating between existing examples.
- Ensures a more balanced training set.
- Leads to improved **recall**—critical in phishing detection where false negatives are dangerous.

1.4. Threshold Tuning for Operational Precision

Instead of using the default classification threshold of 0.5, the system employs **threshold tuning (e.g., 0.65)** to:

- Optimize for recall or precision based on business needs.
- Reduce false positives while still capturing most phishing cases.
- Allow better integration into enterprise risk workflows (where alerts need high confidence).

2. CVSS-Based Cyber Risk Scoring

2.1. Why CVSS?

The **Common Vulnerability Scoring System (CVSS)** provides a standardized way to evaluate the severity of software vulnerabilities. In this project, the key features include:

- **CVSS score (0–10)**: Quantitative measure of the base severity of a vulnerability.
- **Attack complexity**: Qualitative feature indicating how difficult it is to exploit the vulnerability (LOW, MEDIUM, HIGH).

2.2. Feature Encoding and Input Normalization

Because ML models require numerical input:

- The **complexity** feature is **label-encoded** into integers.
- The **severity** (target variable) is also label-encoded into categories: LOW, MEDIUM, HIGH, and CRITICAL.
- The **CVSS scores are standardized** using StandardScaler to ensure that features on different scales do not distort the model.

Normalization ensures that features are weighted fairly during model training.

2.3. Class Integrity via Data Injection

Real-world datasets often lack representation of all possible classes. To ensure the model learns to predict all severity levels, the system:

- Manually adds synthetic examples for missing classes (e.g., LOW, MEDIUM).
- Ensures **stratified training** across all risk levels.
- Prevents "class collapse" during inference (e.g., the model always predicting HIGH or CRITICAL).

2.4. Risk Prediction Using Random Forest

Again, a **Random Forest Classifier** is employed, due to:

- Its ability to handle mixed-type features.
- Its robustness against overfitting, particularly with limited samples per class.
- Its support for feature importance visualization, which aligns well with explainability goals.

The final output is a predicted **severity label** (e.g., HIGH), which can be used in vulnerability management systems.

3. Explainability via LIME

3.1. The Role of Explainability in Cyber AI

In sensitive domains like cybersecurity, black-box models are often rejected due to lack of transparency. Stakeholders need to understand:

- Why a certain email is flagged as phishing.
- Why a vulnerability is deemed critical.

LIME (Local Interpretable Model-agnostic Explanations) is used to address this need.

3.2. LIME for Text

- Applies perturbation-based sampling to identify **words that most influenced the classification**.
- Returns a visual plot showing which words contributed positively or negatively to phishing detection.
- Enables **security analysts** to verify the model's logic and refine policies accordingly.

3.3. LIME for Tabular Data

- Used to explain severity classification in the CVSS risk scoring module.
- Visualizes how much the **CVSS score,** and **complexity** pushed the prediction toward a specific severity label.
- Offers a **local explanation**, i.e., per-input basis rather than global feature importance.

While LIME is inherently localized, it is valuable for interactive dashboards and real-time analysis.

4. Deployment Architecture & Real-Time Inference

The entire system is modular and deployable using:

4.1. Streamlit UI

- A lightweight Python web framework used for real-time user interaction.
- Provides:
  - Textbox inputs for email or vulnerability features.
  - Real-time inference and explanation display.
  - Clean, no-server-needed deployment.

4.2. Model Serialization

To enable reuse and separation of concerns:

- All models, encoders, and scalers are **saved using joblib**.
- Allows loading models during inference without retraining.
- Ensures consistency across training, development, and deployment environments.

Final Thoughts & Strategic Conclusion

This project represents a **dual-pronged, real-world AI solution for cybersecurity**—targeting both human and system-level threats through explainable machine learning models. The choice of:

- TF-IDF with Random Forests for phishing detection,
- CVSS + complexity-based risk scoring,
- LIME explainability for model transparency,
- and Streamlit for end-user accessibility,

…reflects a pragmatic and modular approach to deploying AI at the edge of IT operations.

From a strategic standpoint:

- **Security analysts** can use this tool to rapidly screen suspicious emails or evaluate CVEs.
- **Product teams** can integrate the risk scoring module into their software security pipelines.
- **Enterprise architects** can adopt the explainability outputs to meet compliance and audit needs.

By combining interpretability, predictive accuracy, and deployment readiness, this project builds a strong foundation for **AI-integrated SOC (Security Operations Center) tools**, where automated triage and analyst support go hand in hand.

Future enhancements can integrate deep learning (e.g., BERT for phishing), global explanation tools (SHAP), and automated remediation strategies, evolving this into a full-stack AI cybersecurity framework.

Let me know if you'd like a downloadable PDF version or a visual concept map alongside this study.