TECHNICAL REPORT

AIML FOR NETWORKING
TEAM NAME: ENCRYPTO KNIGHTS
NO. OF PEOPLE: 3

Problem Statement

Modern networks face increasing challenges in monitoring and securing traffic due to the exponential growth of data, encrypted communication, and sophisticated cyber threats. Traditional rule-based security measures and deep packet inspection (DPI) techniques are becoming less effective in detecting and classifying threats, especially in encrypted traffic. Manual intervention in network traffic classification is inefficient, leading to delayed threat detection and security vulnerabilities. To address these issues, Al-driven solutions can analyze traffic patterns, detect anomalies, classify applications, and enhance security in real-time, ensuring adaptive and intelligent network defence.

Abstract

This project focuses on developing a machine learning-based system to detect phishing websites through URL analysis. Various classification models were explored and evaluated to identify the most effective approach for accurate prediction. The system is integrated into a user-friendly web application built with Streamlit, allowing real-time detection of suspicious URLs. By analyzing structural patterns, domain information, and other URL-based indicators, the application helps users determine whether a given link is legitimate or potentially malicious. This solution aims to support internet safety by offering a quick, accessible, and reliable tool for phishing threat identification.
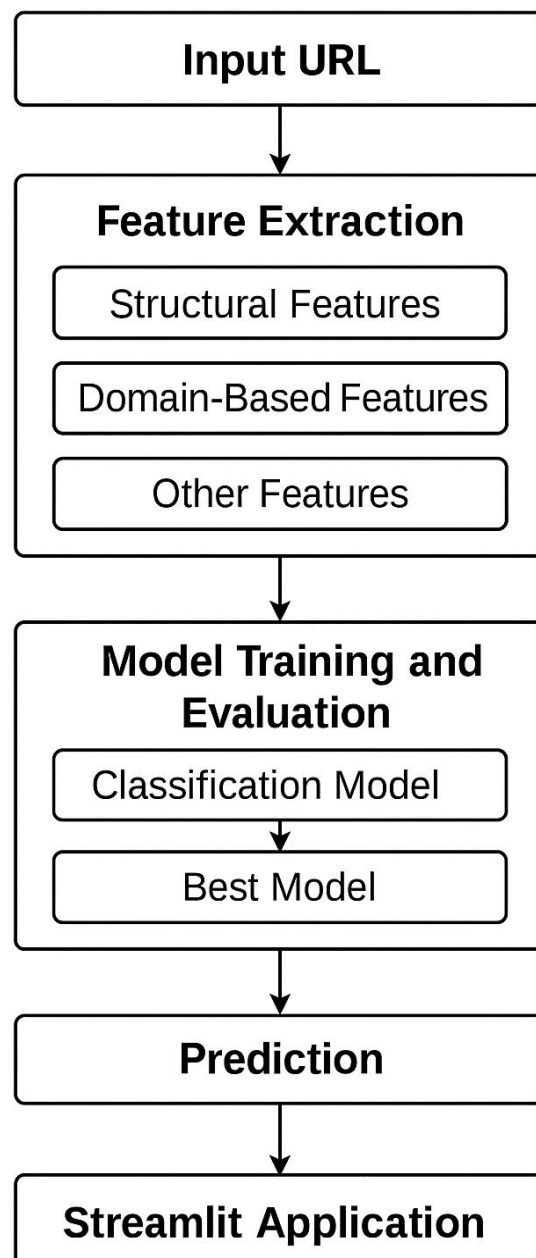
Introduction

Phishing is a widespread cyber threat where attackers trick users into revealing sensitive information through fake websites that closely resemble legitimate ones. These fraudulent websites are often accessed via deceptive URLs sent through emails, social media, or messaging platforms. Identifying such malicious links before users interact with them is crucial to improving online safety.

This project aims to develop an intelligent URL-based phishing detection system using machine learning techniques. By analyzing various characteristics of URLs — such as their structure, domain information, SSL certificate status, and DNS validity — the system can effectively classify whether a given URL is legitimate or potentially harmful. Multiple machine learning models were evaluated to determine the most accurate and efficient one for this task.

To make the system accessible and easy to use, a Streamlit-based web application was developed. It allows users to input any URL and instantly receive a prediction, enhancing real-time threat detection. This project demonstrates the

power of combining machine learning with real-time feature extraction to combat phishing attacks and protect internet users from falling victim to online scams.

Architecture Diagram

```
┌─────────────────────────────┐
│          Input URL          │
└─────────────────────────────┘
               │
               ▼
┌─────────────────────────────┐
│      Feature Extraction     │
│  ┌───────────────────────┐  │
│  │   Structural Features │  │
│  └───────────────────────┘  │
│  ┌───────────────────────┐  │
│  │  Domain-Based Features│  │
│  └───────────────────────┘  │
│  ┌───────────────────────┐  │
│  │     Other Features    │  │
│  └───────────────────────┘  │
└─────────────────────────────┘
               │
               ▼
┌─────────────────────────────┐
│    Model Training and       │
│        Evaluation           │
│  ┌───────────────────────┐  │
│  │  Classification Model │  │
│  └───────────────────────┘  │
│             │               │
│             ▼               │
│  ┌───────────────────────┐  │
│  │      Best Model       │  │
│  └───────────────────────┘  │
└─────────────────────────────┘
               │
               ▼
┌─────────────────────────────┐
│          Prediction         │
└─────────────────────────────┘
               │
               ▼
┌─────────────────────────────┐
│     Streamlit Application   │
└─────────────────────────────┘
```

Work Flow

1. **User Input (Frontend)**

   o User enters a URL into the Streamlit web app.

   o Clicks "Predict" button.

2. **Feature Extraction (Backend Processing)**

   o The system parses the URL using urlparse(), re, and other standard libraries.

   o It generates 16 features including:

     ▪ Structural features (length, depth, special characters)

     ▪ Domain features (WHOIS domain age, expiration)

     ▪ SSL status

     ▪ DNS resolution

   o Some features are fetched live (e.g., WHOIS, DNS, SSL check).

3. **Model Loading**

   o A pre-trained ML model is loaded from XGBoostClassifier.pickle.dat using pickle.

4. **Prediction**

   o The extracted feature vector (length 16) is passed into the model's .predict() method.

   o The model outputs a prediction: 0 (Legitimate) or 1 (Phishing).

5. **Result Display (Frontend)**

   o Based on prediction, the app shows:

     ▪ ☐ "Legitimate" for 0

     ▪ 🔴 "Phishing" for 1

Optional Control Paths

- **Error Handling**:

- o If feature extraction fails or whois/DNS times out → default fallback values are used.

- **Input Validation**:

  - o If the user enters an invalid or empty URL, the app prompts them for valid input.

Contribution

Person 1 (Koushik Maharushi): Model Development Lead

Responsibilities:

- Data preprocessing and cleaning

- Feature engineering (e.g., URL parsing, WHOIS info, SSL checks)

- Train and compare machine learning models (XGBoost, etc.)

- Evaluate models (accuracy, precision, recall, etc.)

- Save the final trained model as a .pickle file

Deliverables:

- model_training.ipynb or train_model.py

- XGBoostClassifier.pickle.dat

- Documentation of model selection and performance

Person 2 (V. Sreeja): Streamlit App Developer

Responsibilities:

- Design and build the Streamlit app (app.py)

- Implement real-time feature extraction from input URLs

- Integrate model predictions into the app

- Handle exceptions, validation, and result display

- UI polishing and responsiveness

Deliverables:

- app.py

- Functional UI with input → prediction flow

- Error handling and validation logic

**Person 3 (S. Vennela): Deployment & Documentation Lead**

Responsibilities:

- Prepare and organize project structure for deployment

- Create and test requirements.txt

- Deploy the project on Streamlit Cloud (or locally)

- Write project report: abstract, introduction, architecture, etc.

- Create visuals: architecture diagram, flowcharts, control flow

Deliverables:

- requirements.txt

- GitHub repo with all files

- Final project report / documentation / presentation

Result
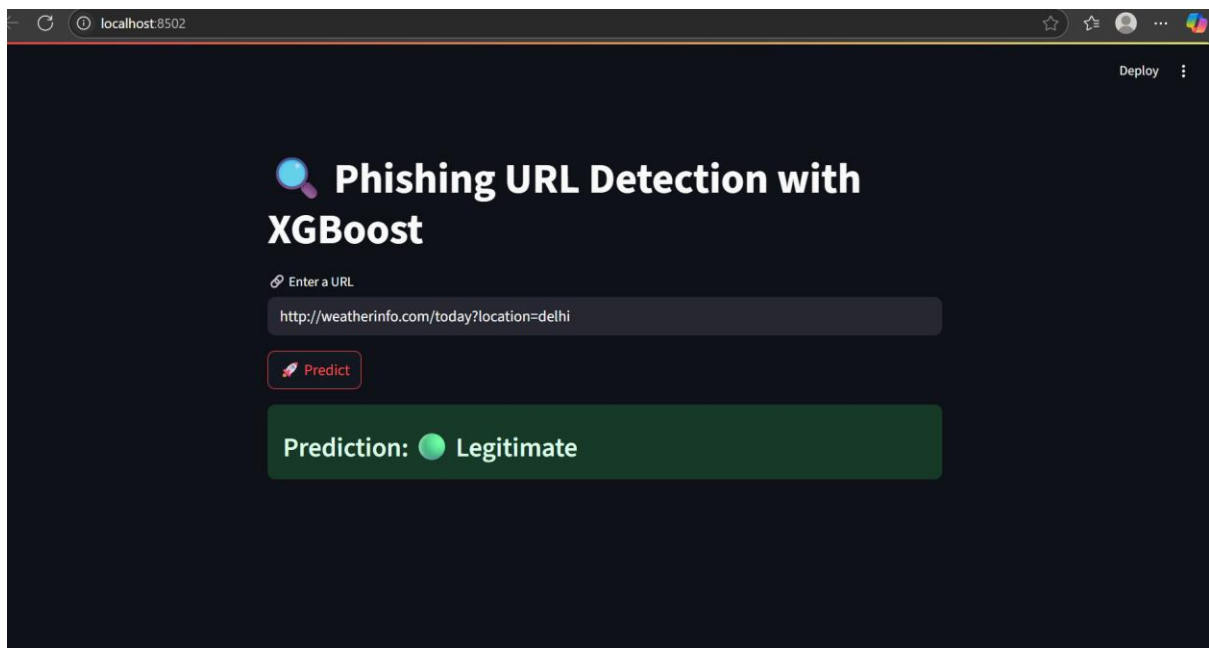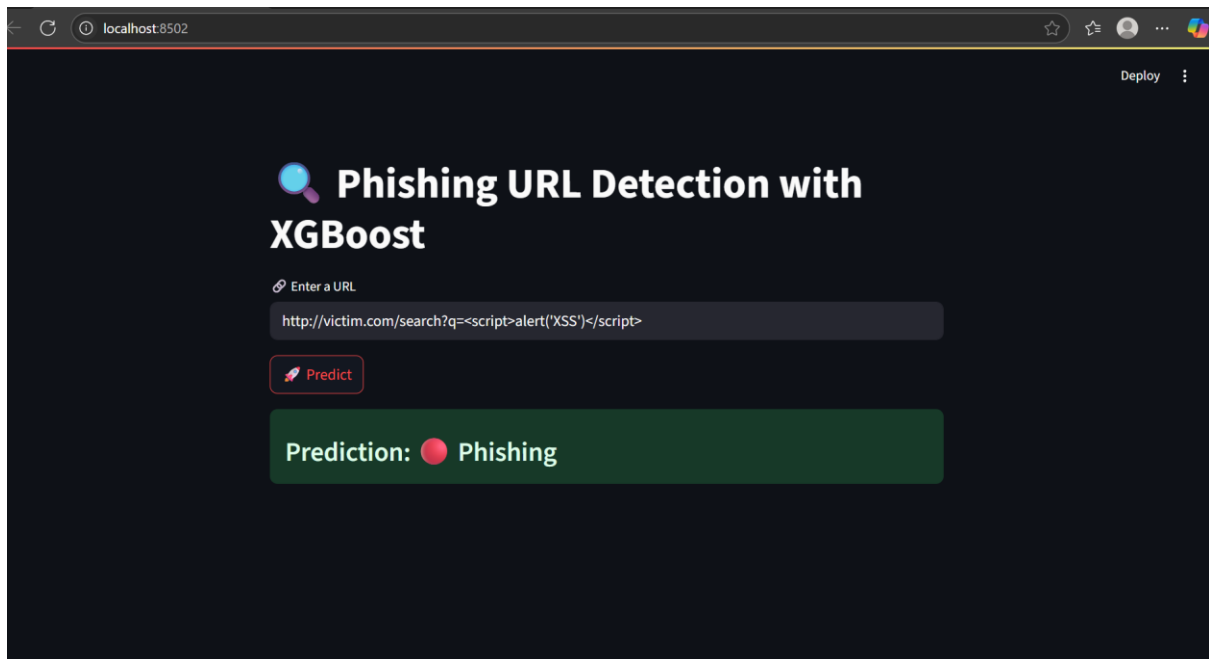
Links of the result:

Result model link:
https://drive.google.com/file/d/1kGU12jrPp37lHjCTQihXkZcgQntIGEtD/view?usp=drive_link

GitHub link: https://github.com/Koushik-Maharushi/IntelUnnatiInternship