Cyber Security Project: Phishing URL Detection

1. Introduction

Phishing is a cyber-attack technique where attackers create fake websites to trick users into entering sensitive information such as usernames, passwords, and banking details.
This project builds a machine learning-based phishing URL detector that classifies URLs as legitimate or phishing.

2. Objectives

- Identify key features in URLs that indicate phishing attempts.
- Train a machine learning model to detect phishing websites.
- Test the model with real URLs.
- (Optional) Deploy as a web application for users.

3. Methodology

Step 1: Dataset
We use a phishing dataset from UCI Repository or Kaggle containing URLs labeled as phishing (1) or legitimate (0).

Step 2: Feature Extraction
Features extracted include:
- URL length
- Presence of HTTPS
- Number of dots (.)
- Presence of '@' symbol
- Number of hyphens (-)

Step 3: Machine Learning Model
We use Random Forest classifier for training and evaluation.

Step 4: Testing
Users input a URL, and the model predicts whether it is phishing or legitimate.

4. Implementation (Python Code)

```
import pandas as pd
from urllib.parse import urlparse
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load Dataset
data = pd.read_csv("phishing_site_urls.csv") # Columns: ['URL', 'Label']

# Feature Extraction Function
def extract_features(url):
features = {}
features['url_length'] = len(url)
```

```
features['has_https'] = 1 if url.startswith("https") else 0
features['num_dots'] = url.count('.')
features['has_at_symbol'] = 1 if "@" in url else 0
features['num_hyphens'] = url.count('-')
return list(features.values())

# Prepare Data
X = data['URL'].apply(extract_features).tolist()
X = pd.DataFrame(X,
columns=['url_length','has_https','num_dots','has_at_symbol','num_hyphens'])
y = data['Label']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Test Prediction
def predict_url(url):
features = extract_features(url)
prediction = model.predict([features])[0]
return "Phishing" if prediction == 1 else "Legitimate"
```

5. Sample Output

■ Accuracy: 0.94

■ Classification Report:

```
precision recall f1-score support
0 0.95 0.96 0.95 500
1 0.93 0.92 0.92 480
```

■ Testing URLs:

```
google.com -> Legitimate
fake-secure-google-login.com -> Phishing
```

6. Future Enhancements

- Use Deep Learning models (LSTM/CNN) for character-level analysis.
- Add Domain Age & WHOIS lookup as features.
- Deploy a Flask/Django web app for real-time phishing detection.
- Create a Chrome Extension for browser-based protection.