**CHAPTER 3**

**PROPOSED SYSTEM**

The proposed machine-learning-based phishing detection system focuses exclusively on URL analysis to enhance network security. The key features of the system include:

**URL-Only Approach for Enhanced Network Protection:**

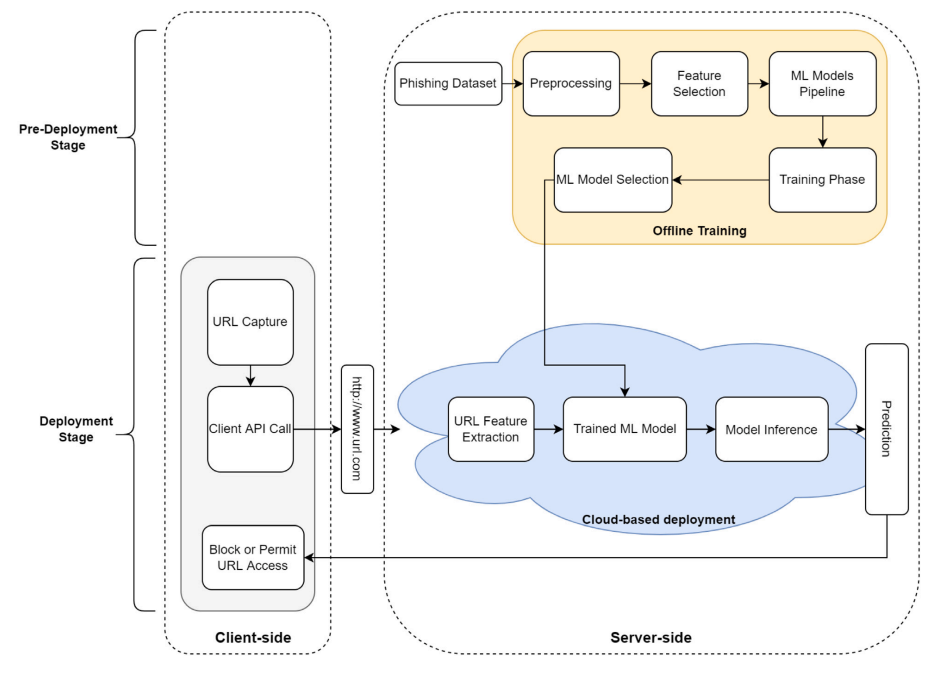
The system relies solely on URL analysis, eliminating the need to access the target webpage. This unique approach minimizes the attack surface, providing better network protection compared to methods requiring webpage access. By focusing on the URL, the system efficiently detects phishing attacks without exposing the network to potential threats from malicious web content.

**Efficient Feature Selection using Recursive Feature Elimination (RFE):**

The system utilizes Recursive Feature Elimination (RFE) during the feature selection stage. RFE helps identify and retain only the most critical features, improving efficiency by reducing the number of features input into the machine learning classifier. This not only enhances computational efficiency but also streamlines the data acquisition phase by extracting a minimal set of essential features for accurate phishing detection.

**Cloud-Based Deployment for Accessibility and Availability:**

The machine-learning-based phishing detection system is deployed on the cloud as an API (Application Programming Interface). This cloud deployment strategy offers high availability and easy accessibility for various networks. The system can be seamlessly integrated into different architectures, such as browser plugins or email client plugins. This cloud-based deployment enhances scalability, accessibility, and overall service availability, making it a robust solution for diverse network environments.



**Fig. 3.1: PhishNot system overview.**

The proposed phishing detection system is organized into two main stages: the pre-deployment stage and the deployment stage.

**Pre-deployment Stage:**

1. **Data Preprocessing:**
   * The dataset undergoes preprocessing to prepare the data, handle missing values, and address data balancing issues.
2. **Feature Reduction and Selection:**
   * A feature reduction and selection process is applied to the preprocessed data, resulting in a dataset with a reduced number of critical features.
3. **Machine Learning Classifier Training:**
   * A pipeline of machine learning classifiers is trained and tested using the feature-reduced dataset. The best-performing model is identified through systematic testing, ensuring its generalization beyond the training dataset.
4. **Model Storage:**
   * The selected model is stored for later deployment into a cloud environment.

**Deployment Stage:**

1. **Client Interaction:**
   * The client captures a URL and sends it to the cloud-based server through an API call.
2. **URL Encoding:**
   * To handle symbols in the URL, UTF-8 encoding is applied before sending it as an HTTP parameter in a simple API call.
3. **Server Processing:**
   * The server receives the encoded URL, extracts features, and requests additional external features from their sources.
4. **Feature Input to ML Classifier:**
   * Extracted features are used as input for the trained machine learning classifier.
5. **Prediction Output:**
   * The classifier predicts whether the URL is 'benign' or 'phishing,' and the server sends this prediction back to the client.
6. **Client Decision:**
   * Based on the received prediction, the client can decide to either block or allow access to the URL.

**Key Features:**

* Emphasis on URL analysis only, reducing the need to access target webpages.
* Recursive Feature Elimination (RFE) for efficient feature selection.
* Cloud-based deployment for high availability and accessibility.
* Systematic testing to ensure model generalization.
* UTF-8 encoding for handling symbols in URLs.
* Integration of external features to enhance prediction accuracy.

**DATASET**

**Dataset Overview:**

* **Total Instances:** 11,054 URLs
* **Benign:** 4,897
* **Phishing:** 6,157

**Feature Information:**

* **Total Features:** 31

**URL-Based Features:**

* Features were extracted by dissecting the URL into four parts.
* Examples of URL-based features include counts of specific special characters like '.' in the domain name or '/' in the directory part.
* Some features considered the entire URL, providing a comprehensive analysis.

**External Features:**

* External features were collected from sources such as Google search indexing.
* Examples of external features include domain age and the presence of an indexed Alexa website ranking.

**Dataset Characteristics:**

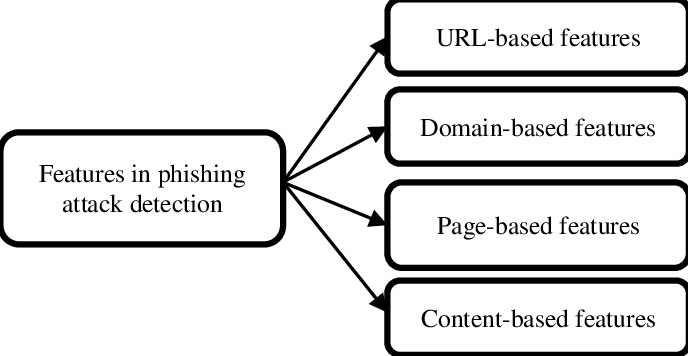
* Imbalance: More benign instances (58,000) than phishing instances (30,646).
* Features extracted to capture diverse aspects of URLs, including structural elements and character occurrences.

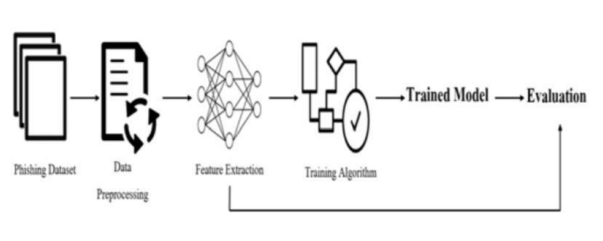
**Dataset Purpose:**

* The dataset is intended for training and testing machine learning models for phishing detection.
* Features were selected to capture both intrinsic URL characteristics and external factors affecting URL legitimacy.

**Implications:**

* The dataset's structure and features aim to provide a comprehensive understanding of URL properties for effective machine learning model training.
* External features enhance the dataset by incorporating domain-related information from sources like Alexa and Google.



****

**Chapter 4**

**DESIGN**

4.1 UML Diagrams

Use Case: A use case diagram is a visual aid that shows how a system communicates external entities, known as actors, to accomplish particular objectives. It is a component of the standardized Unified Modelling Language (UML), a modelling language used in software engineering for the purposes of designing, defining, building, and documenting software systems.

Fig.4.1 Use case Diagram

Activity Diagram: Another kind of diagram in the Unified Modelling Language (UML) that is used to represent a system's dynamic features is an activity diagram. Activity diagrams, which display the order of activities and the decision points that regulate the flow, are especially helpful for visualizing the flow of activities within a system or business process. Workflows, business procedures, and the logic of intricate operations are frequently modelled using them. Fig.4.2 Activity Diagram yes no 13 Class Diagram: In the Unified Modelling Language (UML), a class diagram is a particular kind of diagram that shows a system's structure by modelling its classes, their properties, methods, and the connections between them. Class diagrams are a useful tool for visualizing a system's static design view, which highlights the various entities and their interactions. Fig.4.3 Class Diagram