PROJECT REPORT

Group Members: Muhammad Saad
Fatima Ali
Suleman Malik

Course: CS221 DSA

This report explains a system designed to detect and reduce phishing threats. It has two main parts: Domain and URL Analysis to spot suspicious links, and Message Content Analysis to detect risky or urgent language.

MESSAGE CONTENT ANALYSIS:

1. **Keyword Pattern Scanner:**

- ***Data Structures Used:*** *Trie (Aho–Corasick Automaton)*
- ***Purpose:*** *Efficiently detect multiple phishing-related keywords in text.*
- ***Implementation:***
    - *A **trie** is built where each node represents a character.*
    - *Leaf nodes mark the end of a keyword.*
- ***How It Works:***
    - *The text is scanned once, and all matches are reported in linear time.*

2. **Sentiment and Urgency Detector:**

- ***Data Structures Used:*** *Hash Map, Arrays*
- ***Purpose:*** *Identify urgent or manipulative language in messages.*
- ***Implementation:***
    - *A **hash map** (using an array of structures) stores words/phrases with urgency scores.*
    - ***Arrays** are used for n-gram generation (1-gram, 2-gram, 3-gram).*
- ***How It Works:***

- *A sliding window generates n-grams, which are looked up in the hash map.*
- *Cumulative urgency scores determine the message's risk level.*

3. Attachment Risk Analyzer

- ***Data Structures Used:*** *Trie*
- ***Purpose:*** *Identify and classify dangerous file extensions.*
- ***Implementation:***
  - *A **trie** is built where each node represents a character in a file extension.*
  - *Leaf nodes store a risk level (1–5) instead of a simple end-of-word flag.*
- ***How It Works:***
  - *The trie is traversed to detect file extensions in the text.*
  - *Detected extensions are classified by their risk score.*

4. Confidence Explanation System:

- ***Data Structures Used:*** *Singly Linked List*
- ***Purpose:*** *Provide users with clear, rule-based explanations of detected risks.*
- ***Implementation:***
  - *Each node in the **linked list** contains:*
    - *Rule name (e.g., "too_many_links")*
    - *Explanation text*
    - *Pointer to the next node*
- ***How It Works:***
  - *The system traverses the linked list to fetch explanations for triggered rules.*
  - *Explanations are displayed in a clean, user-friendly format.*

5. Multi-Language Detection System

- ***Data Structures Used:*** *Linked List, Wide Strings (Unicode)*
- ***Purpose:*** *Detect homograph attacks using mixed-script characters.*
- ***Implementation:***
  - *A **linked list** stores mappings between Unicode characters and their ASCII equivalents.*
  - *Each node contains:*
    - *Unicode character (e.g., Cyrillic 'a')*
    - *ASCII equivalent ('a')*
    - *Next pointer*
- ***How It Works:***
  - *The input string is converted to a wide string for Unicode support.*
  - *Each character is checked against Unicode ranges (e.g., Latin, Cyrillic, Greek).*
  - *The linked list is used to identify suspicious mappings.*

6. Link Ratio Detector

- ***Data Structures Used:*** *Arrays, Strings parsing*
- ***Purpose:*** *Calculate the ratio of links to words in a message.*
- ***Implementation:***
    - ***Character arrays*** *and* ***string manipulation*** *functions are used for parsing.*
    - ***Integer counters*** *track:*
        - *linkCount*
        - *wordCount*
    - ***Floating-point*** *variables compute the ratio.*
- ***How It Works:***
    - *Word boundaries are detected using spaces, tabs, and newlines.*
    - *URL patterns are matched using string::find() and length checks.*
    - *The ratio is compared against thresholds to determine risk.*

URL and Domain Analysis Tools

1. URL Heuristic Analysis

- ***Data Structures Used:*** *Strings and Boolean flags*
- ***Purpose:*** *Evaluate URLs based on suspicious features like length, symbols, and keywords.*
- ***Implementation:***
    - *Utilizes C++ Standard Library's string class for URL storage and manipulation*
    - *Employs character-level parsing algorithms for pattern detection*
    - *Implements heuristic scoring through integer accumulation and string concatenation for result formatting.*
- ***How It Works:***
    - *The system applies multiple heuristic checks including length validation, symbol detection, IP address parsing, and keyword matching*
    - *Each heuristic violation contributes weighted scores to a cumulative risk assessment*
    - *Final classification follows threshold-based categorization into LOW RISK (0-14), SUSPICIOUS (15-29), or HIGH RISK (30+) categories*
    -

2. Visual Spoof Analysis

- ***Data Structures Used:*** *Linked List*
- ***Purpose:*** *Detect homoglyph attacks by normalizing domains.*
- ***Implementation:***
    - *A* ***singly linked list*** *stores mappings of visually similar characters (e.g., 'o' → '0', 'rn' → 'm').*

- ***How It Works:***
    - *The normalizeDomain function traverses the linked list to replace suspicious characters.*
    - *Normalized domains are compared using a simple character-by-character match.*

3. Domain Similarity Analysis

- ***Data Structures Used:*** *2D Array (Dynamic Programming Table)*
- ***Purpose:*** *Quantify similarity between domains using the Levenshtein distance.*
- ***Implementation:***
    - *A **2D array** is used to store intermediate values for edit distance calculations.*
- ***How It Works:***
    - *The algorithm fills the DP table to compute the minimum edits needed to transform one domain into another.*
    - *The result is converted into a similarity percentage.*