Homograph Detection Tool

Name: Chanchal Teli

Intern ID: 125

Introduction

Homograph attacks are phishing techniques where malicious URLs use characters from different alphabets that look visually similar to English letters. For example, the Cyrillic letter 'a' looks like the Latin 'a'. This trick can fool users into visiting fraudulent websites.

Problem Statement

Attackers exploit the visual similarity of characters from different Unicode scripts to create deceptive URLs. These homoglyphs can bypass traditional URL filters, posing a security risk by enabling phishing and malware distribution.

Technology Used

- Programming Language**:** Python was chosen because of its powerful standard library, especially the unicodedata module that provides Unicode character properties and makes script detection easy.
- Key Python Modules:
  - unicodedata: To identify the Unicode script/block of each character.
  - Built-in data structures like dictionaries and sets to efficiently map homoglyphs and store scripts.

Approach and Methodology

Our program detects homograph attacks by analyzing the characters in a URL to identify suspicious homoglyphs. The detection involves:

- Mapping known homoglyphs**:** Characters from other scripts that look like Latin letters are stored in a dictionary.
- Script detection**:** Each character's Unicode script (e.g., LATIN, CYRILLIC) is identified using Python's unicodedata library.

- Suspicion criteria**:** A URL is suspicious if it contains at least one homoglyph character, and uses a mix of Latin and non-Latin scripts.

Code Explanation

- Homoglyph Dictionary:
  A Python dictionary maps homoglyph characters to their Latin counterparts. Example: 'a' (Cyrillic) → 'a' (Latin).
- get_script(char) Function:
  Uses unicodedata.name(char) to find the Unicode character name, then extracts the script block (like 'LATIN' or 'CYRILLIC').
  If the character is unknown, it returns 'UNKNOWN'.
- detect_homograph(url) Function:
  - Loops through each character in the input URL.
  - Adds the script of each character to a set to keep track of all scripts present.
  - Checks if the character is a known homoglyph and stores it.
  - After checking all characters, it confirms the URL is suspicious if:
    - At least one homoglyph is found,
    - Both Latin and non-Latin scripts are present.
  - Returns whether the URL is suspicious, the list of suspicious characters, and the scripts found.
- Example Logic:
  For URL "www.google.com" (with a Cyrillic e), the program detects mixed scripts (LATIN and CYRILLIC) and flags the URL as suspicious because of the homoglyph.

Examples and Demonstrations

URL	**Result**	**Explanation**
www.google.com	Clean	Only Latin letters detected
www.google.com	Suspicious	Contains Cyrillic 'e' homoglyph
www.apple.com	Suspicious	Contains Cyrillic 'a' homoglyph
xn--example.com (punycode)	Depends on input Punycode	requires additional check

Real-World Applications

- Browsers and security tools use similar detection to prevent phishing.
- Helps organizations safeguard users by filtering deceptive URLs.
- Useful for email security, web filtering, and user awareness.

Conclusion

This PoC demonstrates a simple yet effective technique to detect homograph attacks by analyzing Unicode scripts and homoglyph mappings in URLs. Using Python and its Unicode tools allows efficient detection of suspicious mixed-script URLs that could be phishing attempts.