Proof of Concept – Homograph Attacks Detection

Name : Sandhya Singh

Intern ID : 233

**1 . Definition**

Homograph attack detection is the process of analyzing text (especially URLs, domain names, usernames, or file names) to identify characters that look visually similar to legitimate ones but are actually different Unicode code points. These characters are used by attackers to create spoofed identities, tricking users into visiting malicious websites or trusting fake applications.

* **Real** : facebook.com
* **Fake** : faceboTk.com (Here, the attacker uses **Cyrillic capital letter “І”** (U+0406) instead of the Latin “I”.

**2. Approach: How the Detection Works:**

The detection system scans URLs or domain names character by character to identify **suspicious characters**.

1. **The script checks:**

* If characters are **not part of standard ASCII** (A-Z, a-z, 0-9, punctuation).
* If they are **Unicode characters that look deceptive**.
* If they are **invisible** or **non-printable traps**.

1. **Detection Process:**

* **Extract domain** from the full URL (e.g., from http://facebook.com/page, extract facebook.com).
* **Scan each character** in the domain.
* **Identify suspicious characters** that are not standard or safe

**3 . Logic Behind the Detection Script (homograph\_core.py)**

This section explains the **algorithm** used in the detection script:

**Step-by-Step Logic:**

* **Define Safe Characters**:
  + Standard ASCII: A-Z, a-z, 0–9, punctuation marks.
  + Safe invisible characters like \n (newline), \t (tab) are allowed.
* **Scan Each Character** in the domain:
  + If it's **not safe**, fetch:
    - Its **Unicode name** (e.g., "CYRILLIC CAPITAL LETTER І")
    - Its **code point** (e.g., U+0406)
* **Extract Domain** from URL:
  + From https://facebook.com/profile, it extracts facebook.com for analysis

**4 : Explaination of the code :**

import unicodedata

import string

from urllib.parse import urlparse

* unicodedata: (Not used here but useful for Unicode analysis)
* string: Gives access to ascii\_letters, digits, punctuation
* urlparse: Extracts the domain name from a URL

**2. Standard Character Set**

STANDARD\_CHARS = set (string.ascii\_letters + string.digits + string.punctuation + " ")

* Defines the **valid, safe characters** → A–Z, a–z, 0–9, symbols like . / @ etc.
* These are characters you'd expect in normal domains like facebook.com

**3. Check for Suspicious Characters**

def is\_suspicious\_character(ch):

return ch not in STANDARD\_CHARS

* This function checks if a character is **not in the standard ASCII set**.
* Example: Cyrillic а (U+0430) looks like English a but is *not* standard.

**4. Normalize the URL**

def normalize\_url(url):

parsed = urlparse(url)

return parsed.netloc or parsed.path

* This extracts just the **domain name** from a URL.
* Example: from http://faceboIk.com/page, it returns faceboIk.com

**5 : Conclusion**

The project successfully shows that **even simple character-level analysis** can help in **detecting Unicode-based homograph attacks**.