Homoglyph (Homographic) Detector - Proof of Concept (Enhanced)

Author: [Ninad Divekar]
Intern ID: [256]

❖ Overview

This Python-based PoC detects domain spoofing attacks using homoglyphs — deceptive Unicode characters that mimic standard ASCII characters — to trick users into trusting malicious domains (e.g., "google.com" vs. "google.com").
The script compares user-input domains against a trusted whitelist after applying Unicode normalization and evaluates string similarity to detect spoofing attempts.

❖ Key Features

- Unicode Normalization using NFKC for consistent string comparison.
- Similarity Detection using difflib to flag suspiciously close matches.
- Whitelist-Based Comparison with commonly trusted domains.
- Domain Format Validation using regex for basic syntax checking.
- Case Normalization to ensure consistent detection.
- Modular Function Design for ease of testing and future expansion.
- User Interaction with clear safety/suspicion feedback.
- Extensible Design to support further integrations or enhancements.

❖ Tools & Libraries Used

Tool/Library	Purpose
unicodedata	Unicode normalization of domain strings.
difflib	String similarity scoring.
re	Domain format validation (regex).
Python 3.x	Core programming language for the PoC.

❖ What I Learned

- How Unicode normalization (NFKC) is critical for detecting homoglyph attacks.
- Realized the security risks of visual spoofing and how they affect user trust.
- Explored string similarity algorithms like difflib.SequenceMatcher.
- Gained practical skills in Python for building modular detection tools.
- Learned how attackers can exploit visually confusable characters to trick users.

❖ Short Confusable Examples

Input Domain	Trusted Match	Explanation
google.com	google.com	Latin small letter script G (U+0261)
faceb**oo**k.com	facebook.com	Greek omicron instead of English 'o'
amazon.com	amazon.com	Cyrillic 'a' (U+0430) vs ASCII 'a'
rnicrosoft.com	microsoft.com	‘rn’ combo mimics 'm' visually
youtube.com	youtube.com	Latin small letter T with stroke (U+01AD)

Code Implementation:

```
import unicodedata
import difflib
import re

# Whitelist of known legitimate domains
whitelist = [
    'google.com', 'amazon.com', 'facebook.com',
    'microsoft.com', 'youtube.com', 'apple.com',
    'paypal.com', 'instagram.com', 'linkedin.com'
]

def normalize_domain(domain):
    """
    Normalize domain using NFKC (Normalization Form KC)
    to standardize Unicode homoglyphs.
    """
    return unicodedata.normalize('NFKC', domain)

def validate_domain_format(domain):
    """
    Validate if the domain input is in proper format using regex.
    """
    pattern = r'^(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,}$'
    return bool(re.match(pattern, domain))

def is_suspicious(domain):
    """
    Compare normalized domain against a whitelist to detect
    high similarity (potential spoofing).
    """
    normalized = normalize_domain(domain.lower())
    for safe in whitelist:
        ratio = difflib.SequenceMatcher(None, normalized, safe).ratio()
        if ratio > 0.88 and normalized != safe:
            return True, safe
    return False, None

if __name__ == "__main__":
    user_input = input("Enter domain to check: ").strip()

    if not validate_domain_format(user_input):
        print("Invalid domain format. Please try again.")
```

```
    else:
        flagged, matched_domain = is_suspicious(user_input)
        if flagged:
            print(f"Suspicious: '{user_input}' is similar to trusted '{matched_domain}'")
        else:
            print(f"Safe: '{user_input}' does not resemble any known safe domain.")
```

❖ Output:

```
$ python homoglyph_detector.py
Enter domain to check: google.com
Suspicious: 'google.com' is similar to trusted 'google.com'
```

❖ Enhancements Over Base Version

- Input validation for stronger error handling.
- Expanded domain whitelist for broader coverage.
- Code modularization for reusability and testing.
- Real test case examples included.
- Normalization and case handling to detect more variants.

❖ Future Improvements

- GUI or Web App Interface using Flask/Streamlit
- Integrate AI for smart phishing detection
- Real-time domain monitoring/logging
- Browser plug-in for active protection
- Use a Homoglyph Mapping Dictionary for advanced character substitution analysis