Weekly Task Document 1: Homographic (Homoglyph) Detector

Intern Name: Sumith Bangera

Intern ID: 130

Objective

The primary goal of this task is to design and implement a basic detection mechanism to identify suspicious domain names that exploit visually similar Unicode characters (homoglyphs) to mimic legitimate and trusted domain names, which is a common tactic in phishing and social engineering attacks.

Task Overview

Homoglyph attacks exploit the visual similarity between Unicode characters from different scripts and ASCII characters. This tool detects such domains using Unicode normalization, homoglyph character mapping, and string similarity comparison against a whitelist of known legitimate domains.

Key Concepts Used

• Unicode Normalization (NFKC) to standardize characters. • Custom homoglyph mapping to replace suspicious Unicode characters with ASCII equivalents. • Whitelist comparison with fuzzy matching to detect similar domains. • Libraries used: unicodedata, difflib.

Code Snippet

```
import unicodedata
import difflib

# Step 1: Whitelist of safe domains
SAFE_DOMAINS = [
    "google.com",
    "facebook.com",
    "amazon.com",
    "microsoft.com",
    "apple.com"
]

# Step 2: Known homoglyph mapping
HOMOGLYPHS = {
    "■": "a",  # Cyrillic 'a'
    "■": "e",  # Cyrillic 'e'
    "■": "o",  # Cyrillic 'o'
    "■": "i",  # Ukrainian 'i'
    "■": "c",  # Cyrillic 'c'
    "■": "p",  # Cyrillic 'p'
    "■": "g",  # Latin small letter script g
}

def normalize_domain(domain):
    """Replace homoglyphs with their ASCII equivalent."""
    normalized = ""
    for ch in domain:
        normalized += HOMOGLYPHS.get(ch, ch)
    return unicodedata.normalize('NFKC', normalized)
```

```
def detect_homoglyph(domain):
    """Check if the domain is suspicious."""
    normalized = normalize_domain(domain)
    if normalized != domain:
        print(f"[WARNING] Possible homoglyphs detected in: {domain}")
        print(f"          Normalized form: {normalized}")

    matches = difflib.get_close_matches(normalized, SAFE_DOMAINS, cutoff=0.8)
    if matches:
        print(f"Looks similar to: {matches}")
    else:
        print("No close match found.")

# Example test cases
test_domains = [
    "■oogle.com",
    "𝗀𝗈𝗈𝗀𝗅𝖾.com",
    "google.com",
    "fa■■book.com"
]

for d in test_domains:
    detect_homoglyph(d)
```

Challenges Faced

• Building a comprehensive homoglyph map was challenging. • Some Unicode characters were visually identical to ASCII but escaped detection initially. • Ensuring normalization worked correctly across various inputs. • Balancing fuzzy matching to reduce false positives.

Conclusion

A functional prototype was successfully created that detects homograph domain attacks using Unicode normalization, character mapping, and fuzzy comparison. This forms the basis for enhancing the detection system with broader coverage and real-world applicability.