Homograph Attacks Detection

Name : Khushi Patwa

Intern ID : 234

(A) Definition

Homograph attack detection is the process of identifying and mitigating malicious attempts to deceive users by using visually similar characters from different writing systems to mimic legitimate domain names, URLs, or text strings. This type of detection helps protect against phishing, spoofing, and other social engineering attacks by analyzing and flagging suspicious character patterns that exploit visual similarities between letters from different scripts.

- **Real** : google.com
- **Fake** : goagle.com (Typosquatting or impersonation — not Unicode, but still suspicious)

(B) Approach How the Detection Works

To understand how homograph attack detection works, it's important to break the process into layers — from low-level character inspection to high-level domain reputation. Here's a step-by-step explanation of the approach detection systems use.

1. Input Normalization

Goal: Prepare the input (usually a domain or URL) for analysis.

- Convert domain to lowercase.
- Decode Punycode (e.g. `xn--pple-43d.com` → `apple.com`).
- Strip prefixes like `https://`, `www.`, etc.

2. Script Analysis

Goal: Identify if the domain mixes different writing systems (scripts), which is a red flag.

- Use Unicode data to detect the script of each character.
- If the domain includes characters from multiple scripts, especially non-Latin ones, it's suspicious.

3. Confusable Character Detection

Goal: Detect characters that are visually similar to others (even if from different scripts).

- Use a confusables mapping to replace characters with their base equivalents.

- Compare normalized domain to known domains.

(C) logic behind detection script (homograph_core.py

1. Domain Normalization

Purpose: Standardize the input to reveal hidden Unicode characters

- Many phishing domains use Punycode encoding to represent Unicode characters in an ASCII-compatible form (e.g. xn--pple-43d.com).
- The script decodes these into their actual Unicode representation (e.g. apple.com).

2. Script Detection

Purpose: Identify if the domain mixes characters from multiple writing systems

- Each character in Unicode belongs to a script (Latin, Cyrillic, Greek, etc.).
- Legitimate domain names almost always use a single script (usually Latin).
- If a domain contains characters from more than one script, it is suspicious.

3. Confusable Character Mapping

Purpose: Normalize visually similar characters to a common form

- Unicode contains many characters that look nearly identical (called homoglyphs).
- The script replaces these characters with their Latin equivalents using a confusables database.

4. Similarity Measurement

Purpose: Compare the suspicious domain to known/trusted domains

- Once the suspicious domain is normalized, it is compared to a list of trusted domains (like google.com, apple.com).
- A similarity score is calculated using algorithms like Levenshtein distance or SequenceMatcher.

5. Decision Making

Purpose: Flag the domain as malicious or safe based on rules

- The domain is flagged if:
  - It uses multiple scripts OR
  - It has a high similarity to a trusted domain (e.g., >90%)

(D) Explain of the Code

1. Punycode Decoding

idna. Decode (domain)

- Converts domains like xn--pple-43d.com into readable Unicode: apple.com
- This is essential because homograph attacks often rely on internationalized domain names (IDNs)

2. Script Detection

unicodedata.name(char). split() [0]

- Every character in Unicode has a name, like "CYRILLIC SMALL LETTER A" or "LATIN SMALL LETTER A"
- The script extracts the script type (like CYRILLIC, LATIN)
- If a domain uses multiple scripts (e.g., a mix of Cyrillic and Latin), it's very suspicious

3. Confusables Normalization

confusables.confusable_replace(text)

- Uses a mapping of visually similar Unicode characters (like I, I, o, o, etc.) to their closest Latin equivalent
- Converts apple.com → apple.com
- Makes it easier to compare visually similar domains

4. Similarity Scoring

SequenceMatcher(None, a, b).ratio()

- Measures how similar the normalized spoofed domain is to a trusted domain
- Returns a score between 0.0 and 1.0
- If the similarity is above a threshold (e.g., 0.9 or 90%), it is likely a spoof attempt

5. Trusted Domain Comparison

trusted_domains = ['apple.com', 'google.com']

- List of safe, legitimate domains
- The script checks if the suspicious domain is visually similar to any of them
- You can expand this list based on your environment (corporate domains, partners, etc.)

6. Flagging the Domain

```
if has_mixed_scripts(decoded) or similarity_score > 0.9:
    return True
```

- The detection logic flags the domain if either:
    - It contains mixed scripts, or

- It is highly similar to a trusted domain

This ensures both blatant and subtle attacks are caught.

Conclusion:

Logic Behind `homograph_core.py` in Homograph Attack Detection

The `homograph_core.py` script is designed to detect homograph attacks, a subtle and dangerous form of phishing that leverages visually deceptive characters in domain names to trick users into visiting malicious websites.