

Proof of Concept (PoC) – Homoglyph Domain Detection

1. Title

Homoglyph Attack Detection Tool – Proof of Concept

2. Objective

The objective of this PoC is to demonstrate a Python-based detection mechanism that identifies potentially malicious domain names or URLs that use homoglyph characters (visually similar Unicode characters) to mimic legitimate domains (e.g., replacing the Latin "a" with the Cyrillic "a").

3. Background

Homoglyph attacks, also known as **IDN (Internationalized Domain Name) homograph attacks**, exploit the visual similarity between characters from different scripts (Latin, Cyrillic, Greek, etc.) to create deceptive domain names.

Example:

- Legitimate: google.com
- Malicious: google.com (Cyrillic small letter "g")

Such domains are often used for **phishing attacks**, **credential harvesting**, or **malware delivery**.

4. Scope

- Detect suspicious domains using Unicode normalization and homoglyph replacement.
 - Compare normalized domains against a whitelist of known legitimate domains.
 - Identify and flag domains that have high similarity scores with trusted domains.
-

5. Methodology

The PoC follows these steps:

Step 1 – Input URL

The user provides a URL suspected of being malicious.

Step 2 – Domain Extraction

Extracts the domain name from the URL using regex.

Step 3 – Unicode Normalization

Uses Python's unicodedata module to normalize the text into NFKC form.

Step 4 – Homoglyph Replacement

Maps suspicious Unicode characters to their ASCII equivalents using a predefined homoglyph dictionary.

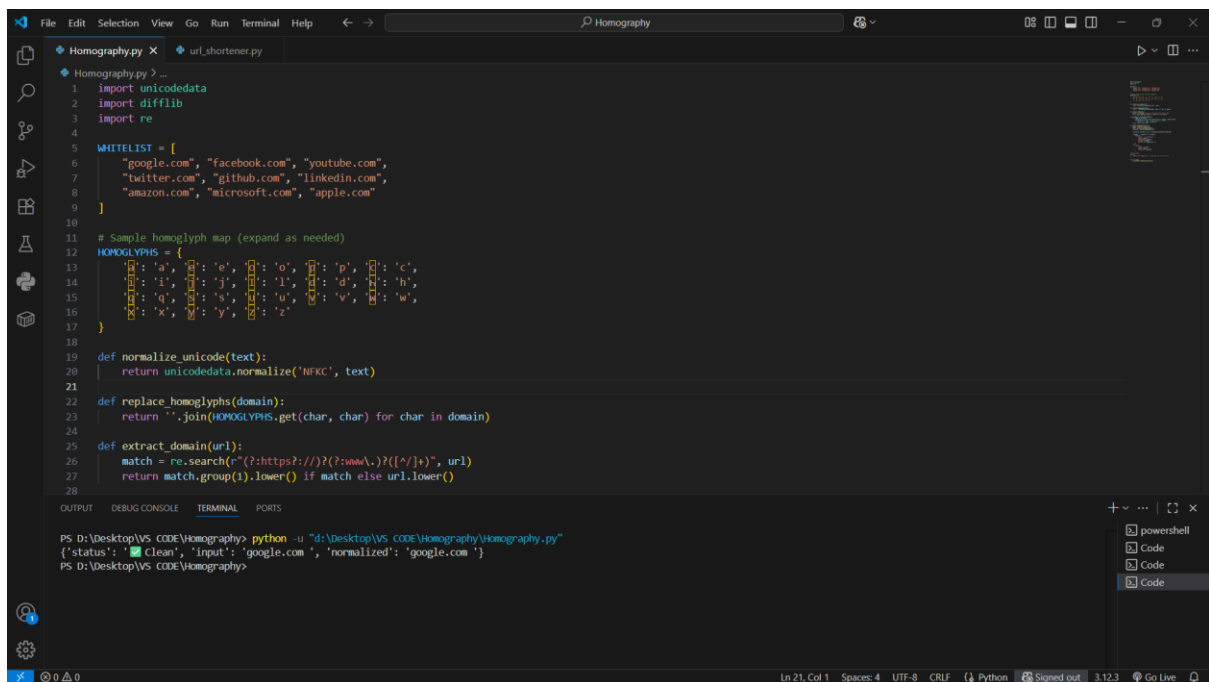
Step 5 – Similarity Check

Uses difflib.SequenceMatcher to compare the normalized domain with whitelisted legitimate domains.

If the similarity score > **0.85**, it's flagged.

Step 6 – Output

Returns JSON-like results indicating whether the domain is **Clean** or **Suspicious**, along with similarity score.



```
File Edit Selection View Go Run Terminal Help
Homography
Homography.py x url_shortener.py
1 import unicodedata
2 import difflib
3 import re
4
5 WHITELIST = [
6     "google.com", "facebook.com", "youtube.com",
7     "twitter.com", "github.com", "linkedin.com",
8     "amazon.com", "microsoft.com", "apple.com"
9 ]
10
11 # Sample homoglyph map (expand as needed)
12 HOMOGLYPHS = {
13     'a': 'ᵃ', 'e': 'ᵉ', 'o': 'ᵒ', 'p': 'ᵖ', 'c': 'ᶜ',
14     'i': 'ᵢ', 'j': 'ᵐ', 'l': 'ᵕ', 'd': 'ᵈ', 'h': 'ᵕ',
15     'q': 'ᵕ', 's': 'ᵍ', 'u': 'ᵘ', 'v': 'ᵛ', 'w': 'ᵛ',
16     'x': 'ᵗ', 'y': 'ʏ', 'z': 'ᶜ'
17 }
18
19 def normalize_unicode(text):
20     return unicodedata.normalize('NFKC', text)
21
22 def replace_homoglyphs(domain):
23     return ''.join(HOMOGLYPHS.get(char, char) for char in domain)
24
25 def extract_domain(url):
26     match = re.search("(?https?://)?(?:(www\.)?([a-z0-9-]+\.[a-z0-9-]+\.[a-z0-9-]+\.[a-z0-9-]+))", url)
27     return match.group(1).lower() if match else url.lower()
28
29
30 PS D:\Desktop\VS_CODE\Homography> python -u "d:\Desktop\VS_CODE\Homography\Homography.py"
{'status': 'Clean', 'input': 'google.com ', 'normalized': 'google.com '}
```

7. Limitations

- The homoglyph mapping dictionary is not exhaustive.
- Threshold-based similarity may lead to false positives or negatives.
- Does not actively block access — only detects.

8. Conclusion

This PoC successfully demonstrates the detection of homoglyph-based phishing domains. Expanding the homoglyph dictionary and integrating this into a real-time security tool could enhance web browsing safety.