

**Homographic (Homoglyph) Detector — Weekly Report****Objective:**

- ✓ Homoglyph mapping (homoglyph\_map dictionary)
- ✓ Maps visually similar Unicode characters (e.g., А (Cyrillic A) to a, і (Ukrainian i) to i) to their ASCII equivalents.
- ✓ This helps normalize suspicious domains into a standard form.
- ✓ sanitize\_domain(domain)
- ✓ Replaces homoglyph characters in a domain with their mapped ASCII versions.
- ✓ Example: "google.com" (with Cyrillic 'o') becomes "google.com".
- ✓ calculate\_similarity(a, b)
- ✓ Uses difflib.SequenceMatcher to calculate how similar two domain strings are (returns a score from 0 to 1).
- ✓ load\_file\_lines(filepath)
- ✓ Reads a list of domains from a file (used for both safe whitelist and test domains).
- ✓ detect\_homoglyph\_domains(safe\_domains, test\_domains, output\_path)

For each test domain:

- Normalizes it using the homoglyph map.
- Compares it to each safe (whitelisted) domain.
- If similarity score > 0.85, marks it as [RISKY].
- Otherwise marks it as [SAFE].
- Writes the result to an output file (results.log).

Detection Approach (summary)

- ✓ Normalize domain names using NFKC and decode punycode
- ✓ (IDNA).
- ✓ Identify non-ASCII characters in the domain's second-level
- ✓ name (SLD).
- ✓ Map confusing characters to standard ASCII equivalents using
- ✓ a confusable dictionary.
- ✓ Compare the transformed result with a list of safe domains.
- ✓ Raise flags on suspicious matches or unusual similarities for review.

Code:

```
results.log
1 Checked: google.com
2 + Normalized: google.com
3 + Result: [RISKY] - Similar to: google.com (1.00)
4
5 Checked: facebook.com
6 + Normalized: facebook.com
7 + Result: [RISKY] - Similar to: facebook.com (1.00)
8
9 Checked: youtube.com
10 + Normalized: youtube.com
11 + Result: [RISKY] - Similar to: youtube.com (1.00)
12
13 Checked: netflix.com
14 + Normalized: netflix.com
15 + Result: [RISKY] - Similar to: netflix.com (0.91)
16
17 Checked: amazon.in
18 + Normalized: amazon.in
19 + Result: [SAFE]
20
21 Checked: apple.com
22 + Normalized: apple.com
23 + Result: [RISKY] - Similar to: apple.com (0.89)
24
25 Checked: flipkart.com
26 + Normalized: flipkart.com
27 + Result: [RISKY] - Similar to: flipkart.com (1.00)
28
```

```
EXPLORES
domain_checker.py
whitelist_domains.txt
domains_to_test.txt
results.log

UNICODE HUNTER
domain_checker.py
domains_to_test.txt
results.log
whitelist_domains...

domain_checker.py 2...
1 import difflib
2 import unicodedata
3
4 # Extended homoglyph map
5 homoglyph_map = {
6     'a': 'ᄁ', 'A': 'ᄂ', 'u': 'ᄃ', 'U': 'ᄄ',
7     'e': 'ᄅ', 'E': 'ᄆ', 'i': 'ᄇ', 'I': 'ᄈ',
8     'l': 'ᄉ', 'L': 'ᄊ', '1': 'ᄋ', 'O': 'ᄌ',
9     'o': 'ᄍ', '0': 'ᄎ', '0': 'ᄏ',
10    'p': 'ᄐ', 'P': 'ᄑ',
11    'q': 'ᄒ', 'Q': 'ᄓ',
12    'u': 'ᄔ', 'U': 'ᄕ',
13    'y': 'ᄖ', 'Y': 'ᄗ',
14    'd': 'ᄘ', 'D': 'ᄙ',
15    't': 'ᄚ', 'T': 'ᄛ',
16    'c': 'ᄜ', 'C': 'ᄝ',
17    'h': 'ᄞ', 'H': 'ᄟ', 'm': 'ᄠ',
18 }
19
20 def sanitize_domain(domain):
21     normalized = ""
22     for char in domain:
23         normalized += homoglyph_map.get(char, char)
24     return normalized
25
26 def calculate_similarity(a, b):
27     return difflib.SequenceMatcher(None, a, b).ratio()
28
29 def load_file_lines(filepath):
30     try:
31         with open(filepath, "r", encoding="utf-8") as file:
32             return [line.strip() for line in file if line.strip()]
33     except Exception as e:
34         print(f"[!] Error reading {filepath}: {e}")
35     return []
36
37 def detect_homoglyph_domains(safe_domains, test_domains, output_path):
38     with open(output_path, "w", encoding="utf-8") as out:
39         for test in test_domains:
40             normalized = sanitize_domain(test)
41             status = "[SAFE]"
42             for safe in safe_domains:
43                 score = calculate_similarity(normalized, safe)
44                 if score > 0.85:
45                     status = f"[RISKY] - Similar to: {safe} ({score:.2f})"
```

Output:

```
UNIC...
domain_checker.py
domains_to_test.txt
results.log
whitelist_domains...

domains_to_test.txt
1 C:\Users\chandan kumar\OneDrive\Desktop\unicode-hunter\domains_to_test.txt
2 facebook.com
3 youtube.com
4 netflix.com
5 amazon.in
6 apple.com
7 flipkart.com
8
```

## What I Learned:

- ✓ Unicode allows dangerous visual deception.
- ✓ Skeleton mapping and normalization are vital first steps.
- ✓ Even simple tools can catch high-risk spoofing attempts.

- ✓ Manual review is always needed for critical decision-making.

**Conclusion:**

The implemented solution provides a foundation for identifying suspicious domain names using homoglyph characters. Although lightweight, it is effective for early detection and supports further enhancement using advanced techniques.