# Homoglyph Detection Tool

**Name:** Vinit Chaskar
**Intern ID:** 395
**Date:** 27-07-2025
**Tool Name:** Homoglyph Detector
**Category:** Threat Detection / Digital Forensics / Domain Intelligence
**Platform:** Python-based (Cross-platform CLI tool)

## ■ Objective

To build a detection tool that identifies homoglyph attacks, where attackers use visually similar Unicode characters (e.g., Cyrillic '■' instead of Latin 'a') to trick users into visiting malicious domains that appear to be legitimate.

## ■ Use Case

Homoglyph attacks are commonly used in phishing domains, brand impersonation, and credential harvesting. This tool helps identify such domains and alerts analysts about potential spoofing attempts.

## ■ Technology Stack

Python 3 - Core programming language
unicodedata - Unicode normalization difflib
- Domain similarity matching

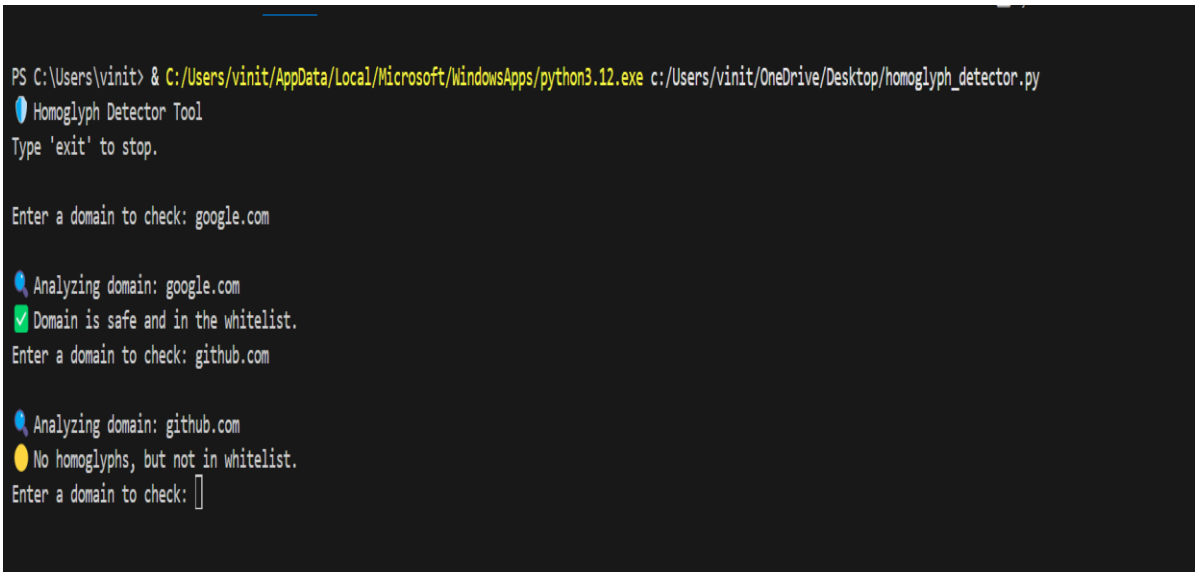
## ■ How It Works

1. Normalization: Converts the input domain using Unicode normalization (NFKC form) to detect hidden character differences.
2. Homoglyph Mapping: Scans the domain for characters in a predefined HOMOGLYPHS dictionary and replaces them with their ASCII lookalikes.
3. Similarity Check: Compares the cleaned domain against a trusted whitelist using fuzzy matching.
4. Alerts Analyst: Displays Unicode details, mapped output domain, and closest matching domain.

## ■ Whitelist Used

```
WHITELIST = [
    "google.com", "facebook.com",
    "youtube.com", "amazon.com",
    "microsoft.com", "instagram.com",
    "twitter.com"
]
```

## Output


PS C:\Users\vinit> & C:/Users/vinit/AppData/Local/Microsoft/WindowsApps/python3.12.exe c:/Users/vinit/OneDrive/Desktop/homoglyph_detector.py
Homoglyph Detector Tool
Type 'exit' to stop.
Enter a domain to check: google.com
Analyzing domain: google.com
Domain is safe and in the whitelist.
Enter a domain to check: github.com
Analyzing domain: github.com
No homoglyphs, but not in whitelist.
Enter a domain to check:

## ■ Threat Relevance

Homoglyph domains are used in advanced phishing campaigns. This tool helps identify such threats during reconnaissance before they become active attacks.

## ■ Advantages

- Works on any system with Python installed.
- No external libraries needed.
- Useful for SOC teams, forensic analysts, and threat hunters.
- Easily extendable for real-time monitoring.

## ■ Reference

Wikipedia on IDN Homograph Attack, Threat Labs, inspired by homoglyph detection principles from threat intel platforms.

## ■ Python Script (Homoglyph Detector)

```
importunicodedata import difflib

HOMOGLYPHS = {
        '■': 'a', '■': 'e', '■': 'o', '■': 'c', '■': 'p', '■': 'h',
        '■': 'i', '■': 'j', '■': 'l', '■': 'g', '■': 'd', '■': 'i',
        '■': 'a', '■': 'e', '■': 'o', '■': 'r', '■': 's', '■': 't', '■': 'u',
}

WHITELIST = [
        "google.com", "facebook.com", "youtube.com",
        "amazon.com", "microsoft.com", "instagram.com", "twitter.com"
]

def normalize_domain(domain):
        return unicodedata.normalize("NFKC", domain)

defdetect_homoglyphs(domain):
        suspicious_chars = [] cleaned = ''
        for char in domain:
                if char in HOMOGLYPHS:
                    suspicious_chars.append((char,HOMOGLYPHS[char])) cleaned +=
                    HOMOGLYPHS[char]
                else:
                        cleaned += char
                return suspicious_chars, cleaned

def check_similarity(domain, whitelist, threshold=0.8):
        matches = difflib.get_close_matches(domain, whitelist, n=1, cutoff=threshold) return matches[0] if matches else None

def analyze_domain(input_domain):
        print(f"\n■  Analyzing domain: {input_domain}")
```

```
    normalized = normalize_domain(input_domain)
    suspicious, mapped_domain = detect_homoglyphs(normalized)

    if suspicious:
        print("■■  Suspicious characters found:") for s in suspicious:
            code_point = hex(ord(s[0]))
            print(f"          - '{s[0]}' (U+{code_point[2:].upper()}) □ '{s[1]}'") print(f"■ Cleaned
        version: {mapped_domain}")
        similar = check_similarity(mapped_domain, WHITELIST) if similar:
            print(f"■ Looks similar to: {similar}")
        else:
            print("■ No close match found in whitelist.")
    else:
        if input_domain in WHITELIST:
            print("■ Domain is safe and in the whitelist.") else:
            print("■ No homoglyphs, but not in whitelist.")

if __name__ == "__main__":
    print("■■  Homoglyph Detector Tool") print("Type 'exit' to
    stop.\n")
    while True:
        user_input = input("Enter a domain to check: ").strip() if user_input.lower() ==
        "exit":
            print("Exiting tool. Stay safe!") break
        if user_input: analyze_domain(user_input)
```