Proof of Concept (POC): Homoglyph Domain Detector Tool

Name : Akkshat shah

Intern id : 261

🧠 Objective

The objective of this project is to create a simple Python-based tool that can detect potentially malicious domain names that use **Unicode homoglyphs** — characters from non-English scripts (like Cyrillic or Greek) that look almost identical to English letters. These are often used in phishing to trick users into visiting fake versions of legitimate websites like `google.com` instead of `google.com`.

🔍 Understanding Homoglyph Attacks

In cyberattacks like phishing or spoofing, attackers often use Unicode characters that look like common English letters. These **homoglyphs** can make a fake domain appear visually identical to a real one. For example:

- `google.com` uses the Greek 'g' instead of 'g'.
- `facebook.com` uses Cyrillic 'o' instead of English 'o'.

Because the characters look the same to the human eye, it's easy to get tricked — but under the hood, these are completely different domain names. This tool helps detect such tricks by analyzing and normalizing suspicious domains.

🛠️ How the Tool Works

The tool is written in Python and uses only the standard library to stay lightweight. Here's the step-by-step process it follows:

1. **Input Collection**: The user is prompted to enter a domain name.
2. **Unicode Normalization**: It uses the `unicodedata` library to clean and standardize the domain using NFKC form.
3. **Homoglyph Replacement**: A mapping dictionary is used to replace homoglyph characters with their ASCII equivalents.
4. **Non-ASCII Check**: If any characters are not standard ASCII, they are marked as suspicious.
5. **Whitelist Comparison**: The cleaned domain is compared to a known list of safe domains (e.g., google.com, facebook.com).
6. **Similarity Check**: If the cleaned domain looks >80% similar to a trusted domain, it is flagged as a possible phishing attempt.

🧬 Code Modules Explained

- `homoglyphs`: A Python dictionary that stores fake → real letter mappings.
- `clean(domain)`: Takes in a domain name, normalizes it, and replaces homoglyphs.
- `check(domain)`: Main logic that detects suspicious domains and checks similarity.

- `difflib`: Used to calculate how close a suspicious domain is to a real one.
- `unicodedata`: Helps normalize strange-looking characters.

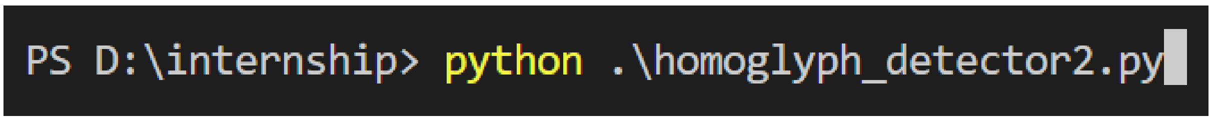
Example homoglyph map:

{ 'a': 'a', 'e': 'e', 'i': 'i', 'o': 'o', 'p': 'p', 's': 's', 'g': 'g', 'l': 'l', 'h': 'h', 'ʏ': 'y' }

⚙️ How to Use the Tool (Step-by-Step)

1. Save the script as `homoglyph_detector.py`.
2. Open Command Prompt or terminal in the folder where the script is saved.

Run the command:



```
PS D:\internship> python .\homoglyph_detector2.py
```

3. Enter a domain when prompted (e.g., `google.com` or `facebook.com`).
4. The tool will tell you:
    - If Unicode homoglyphs are found
    - What the cleaned domain looks like
    - If it closely matches any real domains from the whitelist

---

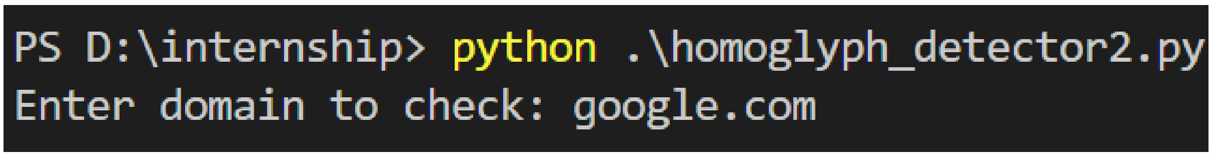
💻 Real-World Use Cases

- Investigating phishing emails and suspicious links

- Checking URLs in readme files, documentation, or user messages
- Preventing spoofed domains in user-submitted forms
- Demonstrating social engineering risks during cybersecurity training
- Adding an extra safety check before whitelisting a domain

🧪 Example Usage (PoC)

Let’s say your input is:

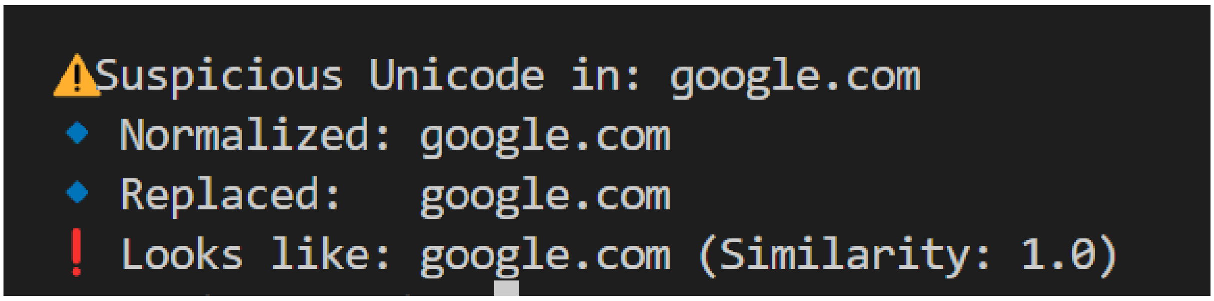


```
PS D:\internship> python .\homoglyph_detector2.py
Enter domain to check: google.com
```

The tool will:

- Normalize the domain using Unicode NFKC
- Replace the homoglyph 'g' with 'g'
- Detect that the domain is suspicious
- Compare it with the real domain `google.com`

Output:



```
⚠️Suspicious Unicode in: google.com
🔹 Normalized: google.com
🔹 Replaced:   google.com
❗ Looks like: google.com (Similarity: 1.0)
```

🧠 Suggestions for Future Enhancements

- Accept bulk domain inputs from a `.txt` file
- Add GUI for non-technical users
- Export suspicious results to CSV/PDF
- Integrate with threat intelligence APIs
- Add a scoring system for threat level
- Build a browser plugin for live link checking

✅ Advantages of the Tool

- Lightweight — runs on any PC with Python installed
- Requires no internet connection
- Easy to extend and modify
- Useful for both training and real-world use
- Detects phishing links before they cause damage
- Works across Windows, Mac, and Linux

🧾 Conclusion

Homoglyph attacks are dangerous because they exploit how humans visually interpret letters. This tool helps detect such attacks before any harm is done. It's simple to use, easy to extend, and can be integrated into larger security pipelines or used independently by security teams, educators, or even casual users who want to stay safe online.