Homoglyph Detection Tool for Suspicious Domain Identification

Intern Name: Sayali Vijay Pol
Intern ID: 345
Organization: Digisuraksha parhari foundation

1.Introduction

Cyber attackers often register domain names that look almost identical to trusted websites by replacing normal characters with homoglyphs — visually similar Unicode characters from other alphabets (e.g., Latin, Cyrillic, Greek). This technique, known as IDN homograph attack, tricks users into believing they are visiting a legitimate site, leading to phishing, credential theft, or malware delivery.

The Homoglyph Detection Tool is designed to detect such suspicious domains by:

- Scanning each character in the given domain/URL.
- Checking for the presence of Unicode characters that closely resemble standard ASCII letters.
- Comparing the domain against a whitelist of trusted domains.
- Flagging any suspicious substitutions for further review.

By implementing this tool, organizations and individuals can prevent users from being misled by fake domains that look right but are dangerously wrong.

2.Objective

The objective of this PoC is to demonstrate a tool that can detect **malicious or suspicious domain names** that use **Unicode homoglyphs** (characters that look similar to normal ASCII characters) to mimic legitimate websites and carry out phishing attacks.

3. Background

Phishing attackers often register domains that **look identical** to real websites but contain **visually similar Unicode characters** instead of normal ASCII ones. Example:

- Real: google.com
- Fake: google.com (where “g” is a Unicode script G, not normal "g")

To a user, they look the same, but to a browser, they are **different domains**. This is called a **homoglyph attack**.

4. Scope

The tool:

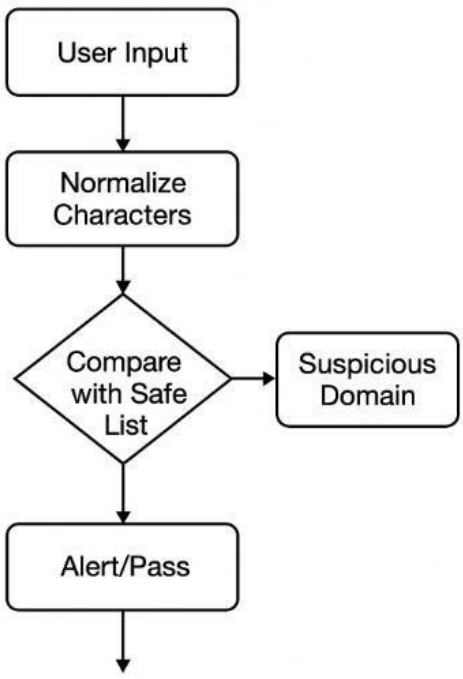
- Takes an input domain name.
- Normalizes it by replacing homoglyphs with equivalent ASCII characters.
- Compares it with a whitelist of trusted domains.
- Flags suspicious domains that are too similar to a trusted one.

5. Tools & Technologies Used •

Language: Python 3

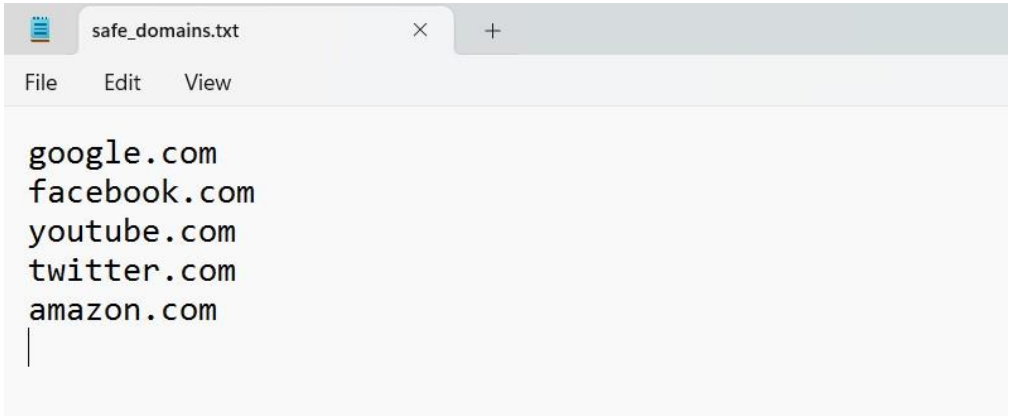
- **Libraries:**
  - json – Store homoglyph mappings
  - difflib – Compare string similarity
- **Files:**
  - safe_domains.txt – Trusted domain list
  - homoglyphs.json – Mapping of homoglyphs to ASCII

6. Workflow Diagram

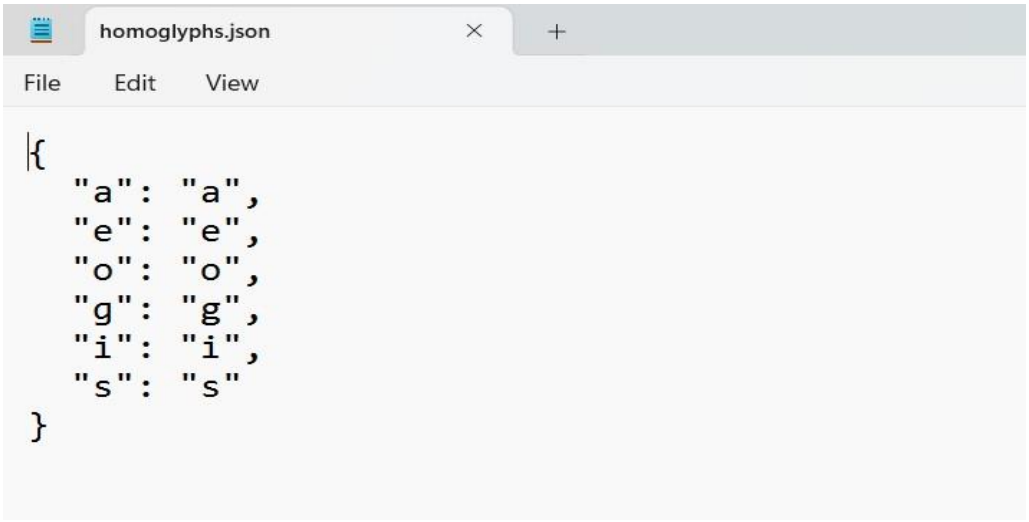


7. Implementation Steps

Step 1: Create safe_domains.txt containing trusted domains.



Step 2: Create homoglyphs.json with Unicode-to-ASCII mappings.



Step 3: Write main.py Python script to:

- Load mappings and safe domains.
- Normalize suspicious domain.
- Compare similarity.

main.py - C:/Users/sayali/Desktop/homoglyph_detector/main.py (3.12.4)

File Edit Format Run Options Window Help

```
import json
from difflib import SequenceMatcher

# Load homoglyph mappings
with open("homoglyphs.json", "r", encoding="utf-8") as f:
    homoglyphs = json.load(f)

# Load safe domain list
with open("safe_domains.txt", "r") as f:
    safe_domains = [line.strip() for line in f]

# Function to replace homoglyphs
def normalize_domain(domain):
    return ''.join(homoglyphs.get(char, char) for char in domain)

# Function to check similarity
def is_similar(domain, legit_domain, threshold=0.8):
    return SequenceMatcher(None, domain, legit_domain).ratio() >= threshold

# User input
suspicious_domain = input("Enter a domain to check: ")

# Normalize
normalized = normalize_domain(suspicious_domain)

# Compare
for legit in safe_domains:
    if is_similar(normalized, legit) and suspicious_domain != legit:
        print(f" Suspicious: '{suspicious_domain}' looks like '{legit}'")
        break
else:
    print(f"Safe: '{suspicious_domain}' appears safe.")
```

Step 4: Run the tool and test multiple domains.

IDLE Shell 3.12.4

File Edit Shell Debug Options Window Help

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sayali/Desktop/homoglyph_detector/main.py
Enter a domain to check: g00gle.com
 Suspicious: 'g00gle.com' looks like 'google.com'
>>>
========== RESTART: C:/Users/sayali/Desktop/homoglyph_detector/main.py =========
Enter a domain to check: google.com
Safe: 'google.com' appears safe.
```

8. Code Snippet (Full code for

main.py) import json from difflib

import SequenceMatcher

```
# Load homoglyph mappings
with open("homoglyphs.json", "r", encoding="utf-8") as f:
   homoglyphs = json.load(f)

# Load safe domains with
open("safe_domains.txt", "r") as f:
safe_domains = [line.strip() for line in f]

def normalize_domain(domain):
   return ''.join(homoglyphs.get(char, char) for char in domain)

def is_similar(domain1, domain2, threshold=0.8):
   return SequenceMatcher(None, domain1, domain2).ratio() >= threshold

suspicious_domain = input("Enter a domain to check: ") normalized
= normalize_domain(suspicious_domain)

for legit in safe_domains:    if is_similar(normalized, legit) and
suspicious_domain != legit:        print(f" Suspicious:
'{suspicious_domain}' looks like '{legit}'")
      break
else:
   print(f"Safe: '{suspicious_domain}' appears safe.")
```

9. Future Enhancements

- Add a web-based interface using Flask.
- Build a browser extension to automatically detect homoglyph domains.

- Use real-time threat intelligence feeds to update homoglyph lists.

10. Conclusion

This PoC successfully demonstrates the ability to detect suspicious domains that use homoglyphs to mimic legitimate sites. Such a tool can help prevent phishing attacks and improve cybersecurity awareness.