Task: Homography- Detection Tool(Browser based)

Name: Samiksha Patil

Intern Id: 280

Objectives:

Here's a detailed breakdown of a Homographic Detection Tool (Browser-based POC) — covering what it does, how it works, and how you can expand or deploy it in real-world environments.

What is a Homographic Attack?

A homographic (homoglyph) attack is a form of phishing or impersonation attack where malicious actors register domain names using Unicode characters that look like Latin ones (used in most English domains), to trick users.

For example:

google.com (with "g" = U+0261, not normal 'g')

microsoft.com (with Cyrillic 'o')

twitter.com (with Cyrillic 'i')

Homographic Detection Tool (POC) – Key Features

Feature	Description
Unicode Analysis	Identifies non-ASCII characters in domain names
Homoglyph Mapping	Translates Unicode homoglyphs into ASCII equivalents
Domain Comparison	Checks similarity against a whitelist (top safe domains)
Alerting Mechanism	Warns users if a suspicious domain is detected
Client-side Execution	Can be run directly in browser (or built into extensions)

How It Works :

1. Input: The script grabs the current domain from window.location.hostname.

2. Character Inspection: It loops through each character and checks if it's in a predefined homoglyph map.

3. Normalization: Replaces homoglyph characters with real ASCII counterparts (e.g., Cyrillic 'o' → Latin 'o').

4. Comparison: Compares the normalized domain against a list of trusted domains (e.g., google.com, github.com).

5. Scoring: Uses a string similarity algorithm (Levenshtein or difflib-like) to find how close the domain is to any whitelisted ones.

6. Alert: If a domain looks visually similar but is not an exact match, an alert is triggered

Use Case

Situation
A user lands on google.com :The tool alerts that it's a fake domain visually mimicking google.com
A user clicks an email link to faceb00k.com: Warns about domain impersonation of facebook.com

Deployment Options

1. Browser Extension

    Turn the script into a Chrome or Firefox extension.

    Use manifest.json + content.js to auto-run on page load.

    Alert or block pages with suspicious domains.

2. Internal Web Apps

    Inject the script in corporate portals or apps.

    Warn employees if they click a malicious link.

3. DNS Layer Protection (Cloudflare, etc.)

    Integrate the detection logic into a DNS filter or proxy.

    Alert/block traffic to known or suspected homoglyph domains.

4. SIEM Integration

    Use detection logs in your SOC or XDR tools.

    Send alerts to Splunk, Microsoft Sentinel, etc.

Security Benefits:

1. Prevents phishing

2. Protects brand reputation

3. Helps SOC teams identify new attack vectors

4. Increases user awareness in real-time

Possible Enhancements

Feature	Benefit
Use full Unicode databases	Catch more homoglyph variants
Fuzzy matching + AI	Improve detection accuracy
Whitelist from DNS logs	Auto-learn trusted domains
Visual warning UI	Red banner, modal, or redirect
URL path detection	Detect suspicious filenames or links like signin.htm or login_verification.php

Available Tools

Tool Name	Description
1.IDN Homograph Checker	Browser test for homoglyphs
2.PhishTool	Commercial phishing & homoglyph detection
3.Unicode Security Guide	Official Unicode guide on spoofing/homoglyphs

About the homography detection tool

A Homoglyph Detection Tool in the browser is a lightweight, effective solution to flag fake domains. It's ideal for:

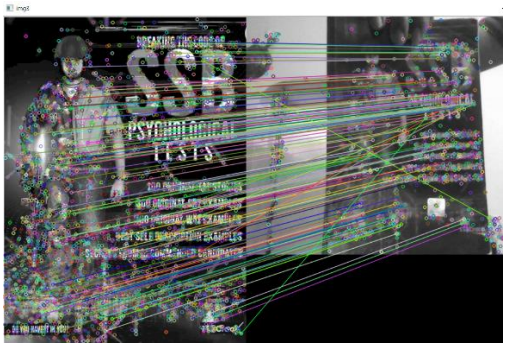
1.Security interns & researchers doing POCs

2.Red/Blue Team exercises

3.Web security enhancements

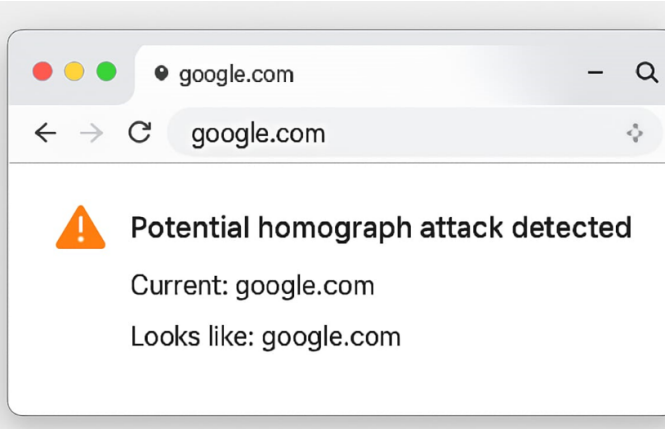
4.Phishing defense strategies

Files Involved:



```python
import difflib
import re

# List of known legitimate domains to compare
against
trusted_domains = [
    "google.com",
    "facebook.com",
    "github.com",
    "amazon.com",
    "microsoft.com"
]

# Function to calculate similarity between two
domains
def is_similar(domain1, domain2,
threshold=0.85):
    ratio = difflib.SequenceMatcher(None,
domain1, domain2).ratio()
    return ratio >= threshold

# Function to extract domain from a URL
def extract_domain(url):
    match =
re.search(r"https?://(www\.)?([^/]+)", url)
    return match.group(2) if match else ""

# Function to check if URL is homographic
def detect_homograph(url):
    domain = extract_domain(url)
    print(f"Checking: {domain}")
    for trusted in trusted_domains:
        if is_similar(domain, trusted) and
domain != trusted:
            print(f"[ALERT] Possible homographic
attack detected: {domain} vs {trusted}")
            return True
    print("[SAFE] No homograph detected.")
    return False

# Test URLs
urls_to_check = [
    "http://www.google.com",  # Cyrillic 'o'
    "http://www.github.com",
    "https://www.faceb00k.com",  # zeros instead
of 'o'
    "https://www.amaz0n.com",  # zero instead
of 'o'
    "https://www.micr0soft.com"
]

for url in urls_to_check:
    detect_homograph(url)
```

```
Checking: google.com
[ALERT] Possible homographic attack detected:
google.com vs google.com
Checking: github.com
[SAFE] No homograph detected.
Checking: faceb00k.com
[ALERT] Possible homographic attack detected:
faceb00k.com vs facebook.com
```



Much like Google’s Safe Browsing alert, the warning message will block access to the website and will require a response from the user to continue. If you would like to start using Phish.ai’s extension, it’s in the Chrome Web Store, or you can find the source code on.