Task : Homographic Detection Tool
(For Logo-Based Detection)

Name : Shivani Yadav
Intern ID : 279

Objectives:

The objective of this project is to design and implement a desktop-based security tool capable of detecting potential brand spoofing attacks by identifying and analyzing logos used on suspicious websites or phishing domains.

The tool aims to automatically extract logo images from websites, compare them against a set of known trusted brand logos, and determine if the site is attempting to visually impersonate a well-known organization.

This approach enhances traditional domain-based detection methods by introducing visual verification, helping users recognize threats that bypass textual detection by using legitimate brand assets like logos.

What is a Homography Attack?:

(Also called a Homoglyph or IDN Homograph Attack)

A homography attack is a phishing or impersonation attack where an attacker registers a domain name using characters that look visually similar to the real ones — but are actually different Unicode characters.

This tricks people into thinking they’re visiting a legitimate website, when in reality, they’re on a malicious site.

Example

- Legit: google.com (all standard ASCII characters)
- Fake: **g**oogle.com — where the “**g**” is U+0261 (Latin Small Letter Script G), not a normal "g".

Both look almost the same in a browser address bar, but they point to different domains.

Purpose of the Attack:

- Phishing: Steal login credentials, payment details, personal data.
- Malware Delivery: Trick users into downloading malicious files.
- Brand Abuse: Damage trust in a company by hosting fake content.
- Bypass Filters: Evade basic security tools that only check exact domain spelling.

How It Works

1. Attacker picks a target (e.g., PayPal).
2. Finds Unicode characters that look like normal letters (a, o, l, etc.).
3. Registers a fake domain using Internationalized Domain Name (IDN) support.
4. Hosts a lookalike website with same logo, colors, and design.
5. Tricks victims into clicking links via email, SMS, or ads.

Real Examples

- microsoft.com (Cyrillic "o")
- twitter.com (Cyrillic "i")
- facebook.com (Cyrillic "o" twice)

How to Prevent

- Use Unicode domain detection tools (like your project's homoglyph detection).
- Implement browser extensions or DNS filters.
- Enable punycode display in browsers (e.g., xn--oogle-9sb.com instead of **g**oogle.com).
- Educate users about suspicious links.

Use Cases / Scenarios for Homography (Homoglyph) Attacks

1. Phishing Login Pages

Scenario:
A victim receives an email claiming their Google account is compromised.
The link in the email goes to **g**oogle.com (with Unicode "**g**"), which looks identical to google.com.
The fake page asks for the victim's username and password, which are then stolen.

---

2. Fake Banking Websites

Scenario:
An attacker registers sbi.com (Cyrillic "s"), which looks like sbi.com.
The site mimics the official bank website, tricking users into entering their net banking credentials.

---

3. Fake E-commerce Platforms

Scenario:
A user clicks on a Facebook ad for a "huge Amazon sale."

The link is to amazon.com (Cyrillic “o”).
The site collects payment details and delivers no products.

---

4. Malware Delivery

Scenario:
A fake tech support ad directs victims to microsoft-support.com.
The site prompts users to download a “security patch”, which is actually malware.

---

5. Brand Reputation Damage

Scenario:
An attacker creates a fake company site using a homograph domain and publishes false information or offensive content.
Visitors think it’s the real brand, harming reputation.

---

6. Social Media Impersonation

Scenario:
A homograph link to twitter.com (Cyrillic “i”) is shared in a viral post.
Victims who try to log in give attackers their credentials.

Deployment Options (Key & Concise)

1. Standalone Desktop Application
    - Runs locally on Windows/Linux/Mac.
    - Easy to use, no internet required for core detection.

2. Browser Extension
    - Detects spoofed logos in real-time while the user browses.
    - Works automatically without manual scanning.

3. Email Security Integration
    - Scans logos from links in incoming emails.
    - Blocks phishing attempts before reaching the user.

4. SOC/SIEM Integration
    - Sends detection logs to Security Operation Centers.
    - Helps analysts track spoofing incidents centrally.

5. Cloud API Service *(Optional)*
    - Logo detection available as an API.
    - Can be integrated into other security tools.

Possible Enhancements

Feature	Benefit
AI-Powered Logo Recognition	Use deep learning models (YOLO, TensorFlow) to detect logos even if they are resized, rotated, or slightly modified.
Multi-Logo Detection	Identify multiple logos on a single webpage and flag any suspicious combinations (e.g., PayPal + Bank of America).
Watermark & Partial Logo Analysis	Detect partial logo usage or watermarked brand images that are altered to avoid detection.
Fuzzy Image Matching	Implement perceptual hashing with higher tolerance for color changes, background edits, or low-resolution logos.

Whitelist & Blacklist Management	Allow organizations to maintain their own sets of trusted and blocked brand logos.

Available Tools

Tool / Library	Description
Pillow (PIL)	Python Imaging Library for opening, processing, and saving images in various formats.
ImageHash	Library for generating perceptual hashes to compare image similarity regardless of size or format changes.
OpenCV	Open-source computer vision library for advanced image processing and feature matching.
BeautifulSoup	Web scraping library to parse HTML and extract logo image URLs from websites.
Requests	Python HTTP library to fetch HTML content and download images from target websites.

Security Benefits

1. **Enhanced Phishing Protection**
    - Detects phishing websites that use stolen or modified brand logos to trick users.
    - Goes beyond domain checks to protect against visual impersonation attacks.

2. **Multi-Layered Security**

- Complements homoglyph/text-based detection by adding image-level verification.
- Reduces the risk of attacks that bypass text-based filters.

3. **Brand Reputation Safeguard**
   - Helps companies identify unauthorized use of their logos on fraudulent websites.
   - Prevents damage to brand image caused by fake or malicious pages.

4. **Early Threat Detection**
   - Flags spoofed websites before users enter sensitive information.
   - Useful in corporate environments for proactive security.

5. **Security Awareness & Training**
   - Helps train employees and end-users to spot visual phishing attempts.
   - Can be used in Red/Blue team exercises for realistic simulations.

About the Logo-Based Brand Spoof Detection Tool:

The Logo-Based Brand Spoof Detection Tool is a desktop security application designed to identify visual phishing attempts by detecting and analyzing logos used on suspicious websites. Unlike traditional text-based detection methods, which focus on domain names and homoglyphs, this tool adds a visual verification layer to catch threats that mimic trusted brands using stolen or modified logos.

It automatically extracts logo images from the target website, compares them against a trusted database of official brand logos, and calculates a similarity score using perceptual hashing and/or computer vision techniques. If the similarity exceeds a defined threshold, the tool alerts the user of a potential spoofing attempt.

This solution is ideal for security teams, corporate users, and individual internet users who want protection against sophisticated phishing attacks that bypass conventional domain-based filters.

The tool can be deployed as a standalone desktop application, integrated into email security systems, or adapted into a browser extension for real-time protection.

Code:

D: > Shivani Practice > Cybersecurity > Logo-Based-Detection-Tool.py > ...

```python
import requests
from bs4 import BeautifulSoup
from PIL import Image
import imagehash
from io import BytesIO
from urllib.parse import urljoin

# ---- CONFIG ----
TRUSTED_LOGO_PATH = "google_logo.png"  # official brand logo file
PHASH_THRESHOLD = 8  # <=8 means likely similar

def get_logo_url(page_url):
    html = requests.get(page_url, timeout=5).text
    soup = BeautifulSoup(html, "lxml")
    for img in soup.find_all("img"):
        src = img.get("src")
        if src and "logo" in src.lower():  # basic heuristic
            return urljoin(page_url, src)
    return None

def compare_logos(trusted_path, suspect_url):
    trusted_logo = Image.open(trusted_path).convert("RGB")
    trusted_hash = imagehash.phash(trusted_logo)

    resp = requests.get(suspect_url, timeout=5)
    suspect_logo = Image.open(BytesIO(resp.content)).convert("RGB")
    suspect_hash = imagehash.phash(suspect_logo)

    diff = trusted_hash - suspect_hash
    return diff

# ---- MAIN ----
page = input("Enter website URL: ").strip()
logo_url = get_logo_url(page)

```

```python
# ---- MAIN ----
page = input("Enter website URL: ").strip()
logo_url = get_logo_url(page)

if not logo_url:
    print("❌ No logo found on the page.")
else:
    print(f"🔍 Found logo: {logo_url}")
    diff_score = compare_logos(TRUSTED_LOGO_PATH, logo_url)
    print(f"Hash difference: {diff_score}")
    if diff_score <= PHASH_THRESHOLD:
        print("⚠️ Possible brand spoof detected!")
    else:
        print("✅ Logo does not closely match trusted brand.")

```