Intern id:263

POC: Homograph Generator & Detector Tool

1.Objective:

This tool demonstrates and detects **homograph attacks**, where attackers use **visually similar Unicode characters** to create deceptive domain names (e.g., google.com with a Cyrillic "o").

Key Features:

- **Educates** users on how homograph attacks work in phishing and spoofing.
- **Detects** suspicious domains by analyzing Unicode characters.
- **Compares** legitimate vs. spoofed domains visually and via Unicode.
- **Flags risks** using script mixing or lookalike characters.
- **Provides tips** to prevent falling for such attacks (e.g., punycode checks, browser behavior).

Ideal for **security awareness**, **phishing simulations**, and **domain validation**.

2.Background on Homograph Attacks

What is a Homograph Attack?

A **phishing technique** where attackers use **Unicode characters** that look like standard **ASCII characters** to create **deceptive domain names**.

Example:

- **Legitimate**: www.google.com
- **Spoofed**: www.google.com (the second “o” is a **Cyrillic** character)

Unicode & IDN (Internationalized Domain Names)

- **Unicode** supports characters from various global scripts.
- **IDNs** allow domain names with these characters.
- **Attackers exploit** this to register domains that **look identical** to real ones, but **lead to malicious websites**.

Impact:

Credential theft (fake login pages).

Malware delivery.

Social engineering and phishing campaigns.

3.Features of tools

• **Homograph Generator:**

- Converts user-input domains by replacing ASCII characters with Unicode homoglyphs.
- Visually highlights swapped characters (e.g., in red) for clear comparison.

• **Homograph Detector:**

- Scans and analyzes domains for the presence of suspicious Unicode characters.
- Alerts users with a “Warning” if homoglyphs are detected.

- Displays the normalized (safe) version of the domain for verification.

4.Technical Workflow

1. **Input Parsing**
User enters a URL or domain name into the tool.

2.**Homoglyph Substitution**
The tool uses a predefined lookup table to replace standard ASCII characters with visually similar **Unicode homoglyphs** (e.g., replacing "o" with Cyrillic "о").

3.**Detection Logic**
Each character is analyzed. If any character matches known homoglyphs from non-Latin scripts, the domain is **flagged as suspicious**.

4. **Output Rendering**

- The generator **highlights substituted characters** for clarity.
- The detector displays a **"Safe"** or **"Warning"** message, along with the normalized or punycode version of the domain.

5.Key Security Considerations:

- Homograph attacks bypass user awareness because the URL looks legitimate visually.
- Attackers can obtain valid SSL certificates for spoofed domains, making detection harder.
- Phishing filters often fail if Unicode characters are validly encoded.

6.Example Attack Scenarios:

- Fake banking sites: hdfcbank.com → ɦdfcbank.com (uses Cyrillic "ɦ")

- Fake social media login: facebook.com → facebook.com (Cyrillic "a", "e", "o")

5.Advantages of the Tool:

- Educates users about Unicode-based phishing attacks.
- Provides a testing platform for cybersecurity training.
- Lightweight and browser-based; no installation required. Can be integrated into phishing awareness workshops.

6.Future Enhancements:

🔹 Support for punycode decoding (e.g., xn--pple-43d.com → apple.com).

🔹 Integration with threat intelligence APIs to check domain reputation.

🔹 Real-time scanning in browsers as a browser extension.

🔹 Expand homoglyph tables for more scripts (Greek, Cyrillic, Armenian).

7.Testing Scenarios:

Input URL and its Detection Result https://www.google.com : ✅

Safe https://www.google.com :⚠ Warning (Cyrillic "o")!

http://facebook.com :⚠ Warning (Cyrillic mix)

https://example.com :✅ Safe

8.Mitigation Strategies Against Homograph Attacks:

- Use punycode display in browsers.
- Implement domain whitelisting in corporate networks.
- Educate users to copy-paste URLs instead of clicking links blindly.
- Employ multi-factor authentication (MFA) to reduce phishing risks.
- Utilize email security filters that check for Unicode domains.

9.Source Code:

```
<!DOCTYPE html>
<html>
<head>
 <title>Homograph Generator & Detector (All-in-One)</title>
 <style>
  body { font-family: Arial, sans-serif; background: #f7f7f7; padding: 30px; color: #333; }
```

```
    .tool-container { max-width: 500px; margin: 25px auto; background: #fff;
 padding: 22px 28px; border-radius: 10px; box-shadow: 0 2px 12px #e0e0e0;
}

   h2, .logo { text-align: center; font-weight: bold; margin-bottom: 15px;
color: #25a244; }     input { width: 100%; padding: 10px; font-size: 15px;
margin-bottom:
12px; border-radius: 5px; border: 1px solid #bbb; }

   button { cursor: pointer; border: none; color: #fff; font-size: 14px;
padding: 8px 12px; border-radius: 5px; margin: 5px 3px; }

   .btn-green { background: #25a244; }

   .btn-green:hover { background: #188038; }

   .btn-blue { background: #1976d2; }

   .btn-blue:hover { background: #145ba1; }

   .output, .alert { padding: 10px; border-radius: 6px; margin-top: 10px;
font-size: 16px; word-break: break-word; }

   .output { background: #f4f4f4; font-family: monospace; }

   .swapped { color: #d7263d; font-weight: bold; }

   .note { font-size: 12px; color: #777; margin-top: 5px; }

   .alert.danger { background: #ffefef; color: #d7263d; border: 1px solid
#f6cac7; }

   .alert.safe { background: #eeffe6; color: #218c5a; border: 1px solid
#b2daf7; }

  </style>

</head>

<body>


  <!-- Homograph Generator -->

  <div class="tool-container">

    <div class="logo">Homograph Generator</div>
```

```
  <input id="inputText" type="text" placeholder="Enter text (e.g. google.com)" />
  <button class="btn-green" onclick="generate()">Generate</button>
  <div class="output" id="output"></div>
  <div class="note">Red = swapped Unicode character</div>
 </div>

 <!-- Homograph Detector -->
 <div class="tool-container">
  <h2>Homograph Detector</h2>
  <input type="text" id="domainInput" placeholder="Enter domain (e.g. facebook.com)" />
  <button class="btn-blue" onclick="checkHomograph()">Check</button>
  <div id="result"></div>
 </div>

 <script>
  /* ---------- Homograph Generator ---------- */
const homoglyphTable = {
    'A': 'A','B': 'B','C': 'C','E': 'E','H': 'H','I': 'I','J': 'J','K': 'K','M': 'M',
    'N': 'N','O': 'O','P': 'P','S': 'S','T': 'T','X': 'X','Y': 'Y','a': 'a','c': 'c',
    'e': 'e','i': 'i','j': 'j','o': 'o','p': 'p','s': 's','x': 'x','d': 'd','q': 'q',
    'y': 'y','r': 'r','v': 'v','w': 'w'
  };
  function generate() {
    const text = document.getElementById('inputText').value;
let result = '';      for (let ch of text) {
```

```
      result += homoglyphTable[ch] ? `<span class="swapped">${homoglyphTable[ch]}</span>` : ch;
    }
    document.getElementById('output').innerHTML = result || '(No input)';
  }

  /* ---------- Extended Homograph Detector (Upper + Lower) ---------- */
const extendedHomographs = {
    "A": ["A","A","A","Á","À","Â","Ä","Ã","Å","Ă","Ą"],
    "B": ["B","B","B","Ḃ","Ƀ"],
    "C": ["C","C","Ç","Ċ","Ĉ","Ć"],
    "D": ["ԁ","d","Ď","Đ","Ḋ"],
    "E": ["E","E","ℰ","É","È","Ê","Ë","Ē","Ĕ","Ė","Ę","Ě"],
    "F": ["Ғ","Ғ","Ḟ"],
    "G": ["G","Ĝ","Ğ","Ġ","Ģ"],
    "H": ["H","Ĥ","Ḧ","Ħ"],
    "I": ["I","I","Í","Ì","Î","Ï","Ĩ","Ī","Ĭ","Į","İ"],
    "J": ["J","Ĵ"],
    "K": ["K","Ƙ","Ķ","Ḱ"],
    "L": ["L","Ⅼ","Ĺ","Ļ","Ľ","Ł"],
    "M": ["M","M","Ḿ"],
    "N": ["N","Ñ","Ń","Ņ","Ň","Ṅ"],
    "O": ["O","O","0","Ó","Ò","Ô","Ö","Õ","Ō","Ŏ","Ő","Ø"],
    "P": ["P","P","Ṕ","Ƥ"],
    "Q": ["Q"],
    "R": ["Г","Ŕ","Ř","Ṙ"],
```

"S": ["S","Ś","Ŝ","Ş","Š","Ṡ"],

"T": ["T","T","Ť","Ţ","Ṫ"],

"U": ["U","Ú","Ù","Û","Ü","Ũ","Ū","Ŭ","Ů","Ű","Ų"],

"V": ["V","Ṽ","Ṿ"],

"W": ["W","Ꙍ","Ŵ","Ẁ","Ẃ","Ẅ"],

"X": ["X","X"],

"Y": ["Y","Y","Ý","Ŷ","Ÿ","Ỳ","Ẏ"],

"Z": ["Z","Ź","Ż","Ž","Ẓ"],

"a": ["a","á","à","â","ä","ã","å","ă","ą"],

"b": ["ь","ƀ","ɓ","ḃ"],

"c": ["c","ç","ċ","ĉ","ć"],

"d": ["d","ď","đ","ḋ"],

"e": ["e","é","è","ê","ë","ē","ĕ","ė","ę","ě"],

"f": ["ғ","ƒ","ḟ"],

"g": ["ĝ","ğ","ġ","ģ"],

"h": ["ʜ","ĥ","ḧ","ħ"],

"i": ["i","ı","ì","î","ï","ĩ","ī","ĭ","į","ɪ"],

"j": ["j","ĵ"],

"k": ["ĸ","ĸ","ķ","ḱ"],

"l": ["l","ĺ","ļ","ľ","ŀ"],

"m": ["м","ḿ"],

"n": ["п","ñ","ń","ņ","ň","ṅ"],

"o": ["o","o","ó","ò","ô","ö","õ","ō","ŏ","ő","ø"],

"p": ["p","ρ","ṕ","ƥ"],

"q": ["q"],

"r": ["r","ŕ","ř","ṙ"],

```
    "s": ["s","ś","ŝ","ş","š","ṡ"],
    "t": ["т","τ","ť","ţ","ṫ"],
    "u": ["u","ú","ù","û","ü","ũ","ū","ŭ","ů","ű","ų"],
    "v": ["v","ṽ","ṿ"],
    "w": ["w","ŵ","ẁ","ẃ","ẅ"],
    "x": ["χ","x"],
    "y": ["ɣ","ý","ŷ","ÿ","ỳ","ẏ"],
    "z": ["ζ","ź","ż","ž","ẓ"]
  };

  function checkHomograph() {
    const domain = document.getElementById('domainInput').value;
let suspicious = false;      let converted = "";

    for (let ch of domain) {
      let found = false;
      for (let [base, homoglyphs] of Object.entries(extendedHomographs)) {
if (homoglyphs.includes(ch)) {          suspicious = true;          converted +=
base;          found = true;          break;
        }
      }
      if (!found) converted += ch;
    }

    const resultDiv = document.getElementById('result');
resultDiv.innerHTML = suspicious
```

? `<div class="alert danger">⚠ Warning: This may be a homograph attack!<br>Converted: <strong>${converted}</strong></div>`

: `<div class="alert safe">✅ This domain looks safe (no obvious homoglyphs found).</div>`;

}

</script>

</body>

</html>

10.Screenshots:

Homograph Generator

www.google.com

Generate

www.google.com

Red = swapped Unicode character

Homograph Detector

www.google.com

Check

⚠ Warning: This may be a homograph attack!
Converted: **www.google.com**

Homograph Generator

www.google.com

Generate

www.google.com

Red = swapped Unicode character

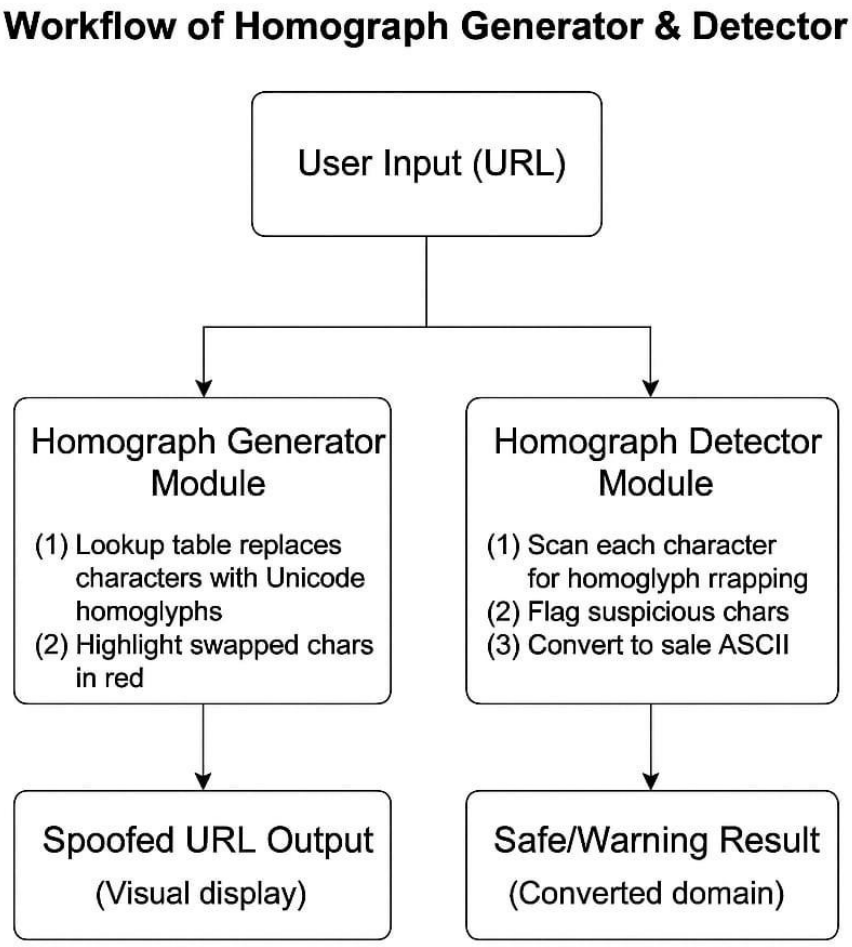
Homograph Detector

www.google.com

Check

✅ This domain looks safe (no obvious homoglyphs found).

11.Flow Diagram



12.Conclusion

This tool successfully demonstrates how homograph attacks can be generated and detected. It can be used for security awareness, red team exercises, and training environments to educate users about modern phishing threats.