Homoglyph Detection Tool Intern: Swadhin Das

Proof of Concept (PoC) Report Intern ID: 396

Submitted To: DigiSuraksha Parihari Foundation

Project Summary:

This tool is a web-based Homoglyph Detection system that identifies visually similar Unicode characters (homoglyphs) in user input. It is designed to detect phishing-style attacks in domain names, email addresses, or any text where malicious characters may impersonate legitimate ones.

Technologies Used:

- JavaScript
- HTML5
- Unicode (Homoglyph Mapping)
- JSON for extensibility

Problem Statement:

Attackers often use Unicode characters that resemble Latin letters to trick users into clicking fake URLs or reading malicious text. These characters are called homoglyphs. This tool aims to detect and flag such characters in real-time from text input.

Objective:

To implement a tool that:

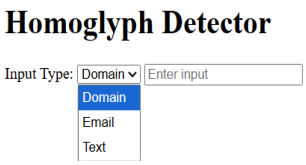
- Allows live input of domain, email, or text
- Detects visually deceptive characters (homoglyphs)
- Highlights the suspicious characters
- Displays explanations and suggested base letters

Tool Features:

- Homoglyph mapping using Unicode
- Real-time input detection
- Custom homoglyph list from JSON
- Type-specific detection (email, domain, text)

PoC Proofs:

Input:



We can choose Domain, Email, Text & input texts for Homoglphy Detection

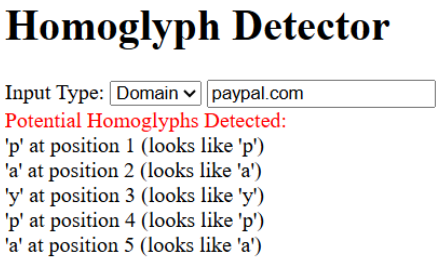
Input Example:

Input Handling:

let inputField = document.getElementById("inputText");
Selects the <input> or <textarea> field where the user types.

inputField.addEventListener("input", checkHomoglyphs);
Listens for real-time input, whenever the user types or pastes, it calls the function checkHomoglyphs() to analyze the content.

Homoglyph Detection:



Process of Homoglyph Detection:

let input = inputField.value;
- ➔ Gets the current value typed by the user — the actual text that needs to be scanned.

for (let i = 0; i < input.length; i++) {
  let char = input[i];
- ➔ Loops through each character in the input string.

for (let base in homoglyphs) {
  if (homoglyphs[base].includes(char)) {
- ➔ Compares each character to a predefined list of homoglyphs.If a character matches a homoglyph (e.g. Cyrillic 'o'), it's flagged as above otherwise no “No Homoglyph Detected.”

Output Generation:

found.push(`'${char}' at position ${i + 1} (looks like '${base}')`);
- ➔ If a suspicious character is found, it's added to an array found[] with:
  - The character , Its position, The letter it resembles

Displaying the Result:

result.innerHTML = "message here";

Dynamically updates the page to show whether homoglyphs were found or not, using innerHTML.

How the Tool Works:

The tool uses vanilla JavaScript to dynamically detect and flag homoglyphs typed or pasted by the user.

getElementById()	selects input/output elements
addEventListener("input")	triggers scanning in real-time
value	accesses user-typed content
for loops	iterate through characters and homoglyph definitions
includes()	checks for Unicode-based visual similarities
innerHTML	displays findings and warnings
fetch()	loads homoglyphs.json for extensibility (optional)

Tool Execution Flow & JavaScript Commands:

 User types in a domain/email/text — captured using:
→ document.getElementById("inputText").value

 As they type or press "Detect", detection function is called:
→ inputField.addEventListener("input", checkHomoglyphs)

 Tool checks for homoglyph characters by looping through:
→ for (let i = 0; i < input.length; i++)
→ and comparing with known homoglyphs:
homoglyphs[base].includes(char)

 Detected homoglyphs are stored:
→ found.push(...)

 Output is dynamically inserted into the HTML:
→ result.innerHTML = ...

 (If implemented) Suspicious characters are highlighted using:
→ ...

Conclusion:

- The tool detects visually deceptive Unicode characters (homoglyphs) in real-time.
- Built using **JavaScript**, it works directly in the browser — no installation needed.
- Can scan domain names, emails, or any text input for phishing risks.
- Uses a **customizable homoglyphs.json** file for easy updates.
- Helps improve **security awareness** and supports **anti-phishing efforts**.
- Can be extended into a browser extension or integrated into email filters.
- All this happens live in the browser using pure JavaScript.
- This lightweight approach requires **no external libraries** and runs entirely in the **browser any one can use it** easily.