

PROOF OF CONCEPT (PoC) REPORT

Tool Name: Homoglyph Detection Tool

Author: Piyush Babele

Intern Id: 386

Executive Summary

The **Homoglyph Detection Tool** is a Python-based command-line utility designed to identify and flag suspicious domain names containing homoglyph characters. Homoglyphs are visually similar characters from different scripts (Latin, Cyrillic, Greek, etc.) that attackers use in phishing campaigns to mimic trusted domains.

This PoC demonstrates how the tool detects such domains, explains the differences, and assists cybersecurity analysts in mitigating phishing risks.

Objective

The goal of this PoC is to validate the functionality of the Homoglyph Detection Tool by:

- Detecting homoglyph-based phishing domains.
- Comparing suspicious domains against a whitelist of legitimate domains.
- Highlighting the exact character differences.
- Demonstrating a realistic use case scenario in a cybersecurity context.

Scope

- **In Scope:**
 - Detection of homoglyphs in domain names.
 - Comparison against known safe domains (`safe_domains.txt`).
 - Identification of suspicious characters and their Unicode details.
 - Highlighting and explaining exact character substitutions.
- **Out of Scope:**
 - URL reputation analysis.
 - Automated blocking of detected URLs.
 - Integration with external threat intelligence platforms.
 - Detection of non-homoglyph phishing techniques such as typosquatting without character replacements.

Tool Overview

The tool processes user-provided URLs/domains, normalizes them using **Unicode NFKC normalization**, and compares them against a whitelist using

string similarity matching. Suspicious domains are flagged, with replaced characters highlighted and explained.

Key Features:

- Unicode homoglyph detection.
- Visual highlighting of suspicious characters.
- Detailed reasoning for each flagged character.
- Lightweight, CLI-based, and easy to integrate.

Requirements

Software Requirements:

- Python latest version
- Built-in modules: `unicodedata`, `difflib`

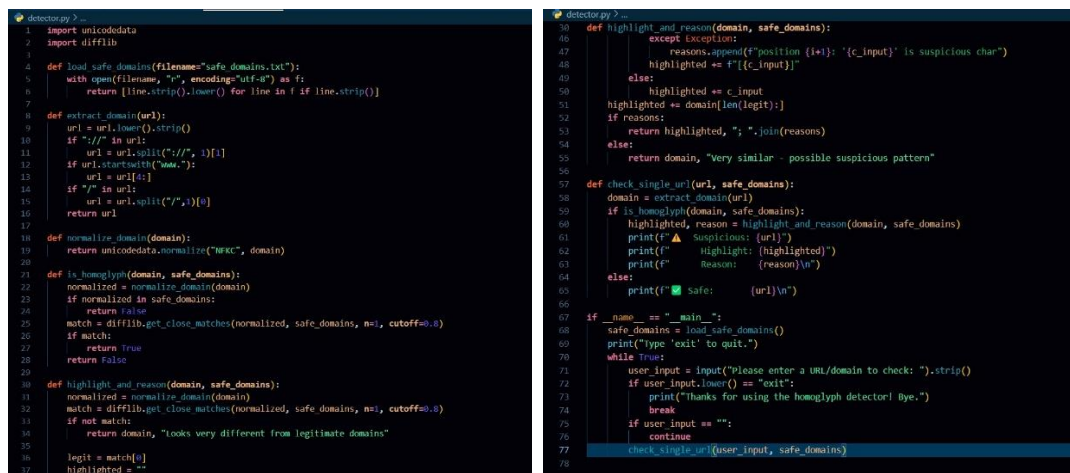
Data Requirements:

- `safe_domains.txt` — whitelist of legitimate domains.
- `sample_input.txt` — test URLs for batch processing.
- `detector.py` — python script for running the tool.

Steps to Run the Tool

1. Install Python Latest Version

Ensure Python is installed and accessible via the terminal or command prompt, and prepare the `detector.py` script with all required commands and formatting so the tool runs correctly.



```
1 import unicodedata
2 import difflib
3
4 def load_safe_domains(filename="safe_domains.txt"):
5     with open(filename, "r", encoding="utf-8") as f:
6         return [line.strip().lower() for line in f if line.strip()]
7
8 def extract_domain(url):
9     url = url.lower().strip()
10    if "://" in url:
11        url = url.split("://")[1]
12    if url.startswith("www."):
13        url = url[4:]
14    if "/" in url:
15        url = url.split("/")[0]
16    return url
17
18 def normalize_domain(domain):
19     return unicodedata.normalize("NFKC", domain)
20
21 def is_homoglyph(domain, safe_domains):
22     normalized = normalize_domain(domain)
23     if normalized in safe_domains:
24         return False
25     match = difflib.get_close_matches(normalized, safe_domains, n=1, cutoff=0.8)
26     if match:
27         return True
28     return False
29
30 def highlight_and_reason(domain, safe_domains):
31     normalized = normalize_domain(domain)
32     match = difflib.get_close_matches(normalized, safe_domains, n=1, cutoff=0.8)
33     if not match:
34         return domain, "looks very different from legitimate domains"
35     legit = match[0]
36     highlighted = ""
37
38     def highlight_and_reason(domain, safe_domains):
39         except Exception:
40             reasons.append(f"position {i+1}: '{c_input}' is suspicious char")
41             highlighted += f"[{c_input}]"
42         else:
43             highlighted += c_input
44             highlighted += domain[len(legit):]
45     if reasons:
46         return highlighted, "; ".join(reasons)
47     else:
48         return domain, "very similar - possible suspicious pattern"
49
50 def check_single_url(url, safe_domains):
51     domain = extract_domain(url)
52     if is_homoglyph(domain, safe_domains):
53         highlighted, reason = highlight_and_reason(domain, safe_domains)
54         print(f"⚠ Suspicious: {url}")
55         print(f"    highlight: {highlighted}")
56         print(f"    reason: {reason}\n")
57     else:
58         print(f"✅ Safe: {url}\n")
59
60 if __name__ == "__main__":
61     safe_domains = load_safe_domains()
62     print("Type 'exit' to quit.")
63     while True:
64         user_input = input("Please enter a URL/domain to check: ").strip()
65         if user_input.lower() == "exit":
66             print("Thanks for using the homoglyph detector! Bye.")
67             break
68         if user_input == "":
69             continue
70         check_single_url(user_input, safe_domains)
```

2. Prepare the Safe Domains List

Create a file named `safe_domains.txt` containing legitimate domains, one per line:

```

safe_domains.txt
1  google.com
2  github.com
3  microsoft.com
4  apple.com
5  facebook.com
6  twitter.com
7  instagram.com
8

```

3. Prepare Test URLs

Create `sample_input.txt` with URLs to check:

```

sample_input.txt
1  https://google.com
2  www.github.com
3  https://www.microsoft.com
4
5  https://google.com
6  www.microsoft.com
7  http://apple.com
8  https://facebook.com
9  https://twitter.com
10 www.instagram.com
11

```

4. Run Interactive Mode

```

PS C:\Users\CYBORG\Desktop\Homo> Python detector.py

```

- Enter URLs one by one.
- Type exit to quit.

PoC Test Execution

```

PS C:\Users\CYBORG\Desktop\Homo> Python detector.py
Type 'exit' to quit.
Please enter a URL/domain to check: https://google.com
▲ Suspicious: https://google.com
Highlight: g[o][o]gle.com
Reason: position 2: 'o' is CYRILLIC SMALL LETTER O, should be 'o' (LATIN SMALL LETTER O); position 3: 'o' is CYRILLIC SMALL LETTER O, should be 'o' (LATIN SMALL LETTER O)

Please enter a URL/domain to check: https://facebook.com
▲ Suspicious: https://facebook.com
Highlight: faceb[o][o]k.com
Reason: position 6: 'o' is GREEK SMALL LETTER OMICRON, should be 'o' (LATIN SMALL LETTER O); position 7: 'o' is GREEK SMALL LETTER OMICRON, should be 'o' (LATIN SMALL LETTER O)

Please enter a URL/domain to check: www.instagram.com
▲ Suspicious: www.instagram.com
Highlight: inst[a]gram.com
Reason: position 5: 'a' is CYRILLIC SMALL LETTER A, should be 'a' (LATIN SMALL LETTER A)

```

```
Please enter a URL/domain to check: www.google.com
✓ Safe:      www.google.com

Please enter a URL/domain to check: www.microsoft.com
✓ Safe:      www.microsoft.com

Please enter a URL/domain to check: www.facebook.com
✓ Safe:      www.facebook.com

Please enter a URL/domain to check: www.flipkart.com
✓ Safe:      www.flipkart.com

Please enter a URL/domain to check: exit
Thanks for using the homoglyph detector! Bye.
PS C:\Users\CYBORG\Desktop\Homo>
```

Use Case Scenario

A SOC analyst is reviewing email phishing logs and identifies multiple suspicious domains. Instead of manually checking each one, the analyst runs them through the Homoglyph Detection Tool. Within seconds, the tool highlights which domains are malicious lookalikes and specifies the exact deceptive characters used.

Advantages

- **Accurate Detection:** Finds even subtle character replacements.
- **Lightweight:** Minimal dependencies, easy to run anywhere.
- **Customizable:** Safe domains list can be expanded for different organizations.
- **Informative Output:** Detailed reasons make investigation faster.

Threat Impact Analysis

Threat Addressed:

- Homoglyph attacks in phishing campaigns.
- Credential theft via visually deceptive domains.
- Malicious redirects to attacker-controlled websites.

Potential Risks Without This Tool:

- Users unknowingly visiting malicious websites.
- Compromise of login credentials.
- Malware delivery via fake login portals.
- Loss of trust in brand/domain.

Impact Reduction:

This tool enables **early detection**, allowing security teams to block suspicious domains before phishing emails or malicious ads reach end-users.

Future Enhancements

- **Integration with Threat Intelligence Feeds** – automatically check detected domains against blacklists.
- **Real-Time Browser Extension** – warn users before visiting a suspicious link.
- **Email Gateway Integration** – scan incoming emails for homoglyph-based URLs.
- **Machine Learning Models** – improve detection accuracy and adapt to new homoglyph attack techniques.
- **Automated Blocking** – link with firewall or proxy rules for immediate protection.

Conclusion

This PoC confirms the Homoglyph Detection Tool's effectiveness in identifying visually deceptive domains. By combining **Unicode normalization** with **similarity analysis**, it offers a lightweight yet powerful defense against phishing. With further integration into real-time monitoring systems, it can significantly reduce phishing success rates.