🛡️ Week One: Homoglyph Detector

🎯 Objective

Detect IDN homograph attacks by identifying visually deceptive Unicode characters in domain names using normalization and string similarity.

🧰 Tools & Libraries

- unicodedata: Unicode normalization (NFKC)
- difflib: String similarity comparison

📋 Whitelist of Known Safe Domains

```
whitelist = [
    'google.com',
    'amazon.com',
    'facebook.com',
    'microsoft.com',
    'youtube.com'
]
```

🔍 Unicode Homoglyph Examples

Fake Character	Unicode	Looks Like	Legitimate
g	U+0261	g	g
o	U+03BF	o	o
c	U+0441	c	c
a	U+0430	a	a
e	U+0435	e	e

🧠 Core Logic

```
import unicodedata
import difflib

# Whitelist of known safe domains
whitelist = [
    'google.com',
    'amazon.com',
    'facebook.com',
    'microsoft.com',
    'youtube.com'
]

def normalize_domain(domain: str) -> str:
    """
    Normalize domain using NFKC to unify Unicode representations.
    """
    return unicodedata.normalize('NFKC', domain)

def is_suspicious(domain: str):
    """
    Compare normalized domain to whitelist using string similarity.
    Flags if similarity > 80% but not an exact match.
    """
    normalized = normalize_domain(domain)
    for safe in whitelist:
```

```
        ratio = difflib.SequenceMatcher(None, normalized, safe).ratio()
        if ratio > 0.8 and normalized != safe:
            return True, safe
    return False, None

if __name__ == "__main__":
    user_input = input("Enter domain to check: ").strip()
    flag, matched = is_suspicious(user_input)
    if flag:
        print(f"⚠️ Domain '{user_input}' looks similar to '{matched}'")
    else:
        print("✅ Domain appears safe.")
```

🧪 Sample Output

```
Enter domain to check: www.google.com
⚠️ Domain 'www.google.com' looks similar to 'google.com'

Enter domain to check: www.abc.com
✅ Domain appears safe.
```

🧠 How It Works

- Normalization: Converts deceptive Unicode characters to canonical form using NFKC.
- Similarity Check: Uses difflib.SequenceMatcher to compare against trusted domains.
- Threshold: Flags domains with >80% similarity but different code points.

📘 What You Learned

- How Unicode spoofing works in phishing attacks.
- How to use Python for domain normalization and comparison.
- Importance of whitelisting and string similarity in detection.

✅ Conclusion

This Week One implementation demonstrates how deceptively simple Unicode homoglyphs can be weaponized in phishing and impersonation attacks. By leveraging Python's built-in normalization and string comparison tools, we've built a lightweight yet effective detector that flags suspicious domains based on visual similarity. While this version is rule-based, it lays the groundwork for more advanced defenses—such as machine learning models, browser plugins, and real-time scanning tools. As you continue building, integrating dynamic homoglyph databases and expanding detection logic will be key to staying ahead of evolving threats.

Name: Dhruv Singh

Intern id: 222