

Proof of Concept – Homograph Attack

Name: Harshali Kasurde

Intern id: 299

1. Definition

Homograph attacks are a type of cyberattack where attackers register domain names using visually similar Unicode characters to impersonate legitimate websites.

For example:

- Real: apple.com
- Fake: apple.com (using Cyrillic letters that look like Latin letters)

This attack bypasses human detection and can evade security filters, making it an effective vector for phishing, malware distribution, and credential theft.

2. Approach

This script focuses on detecting suspicious characters in domain names or URLs by checking if they:

- Are not part of standard ASCII character sets (letters, numbers, punctuation).
- Have Unicode values and names that differ from safe characters.
- Could be deceptive in visual appearance.

It works by:

- Extracting the domain from a given URL.
- Scanning each character of the domain.
- Identifying and reporting characters that are unfamiliar, non-standard, or invisible Unicode traps.

3. Need and Significance Why This Matters in Cybersecurity:

- Traditional systems may not flag Unicode-based spoofing.
- Attackers exploit this to mimic well-known brands.
- This is widely used in phishing and typosquatting attacks.
- Protecting users from visually deceptive domains is essential in:

Secure browsing

Email filtering

DNS-level blocking

Zero-trust security environments

4. Logic Behind homograph_core.py

Step-by-Step Logic:

1. Define safe characters:

a. Standard ASCII (A–Z, a–z, 0–9, symbols) are marked as safe.

b. A few control characters like newline \n, tab \t is allowed.

2. Scan domain character-by-character:

a. For each character:

If it is not part of the safe set

Fetch its Unicode name and code point (e.g., U+0430).

Report it as a suspicious Unicode character.

3. Extract domain from a URL:

a. If a full URL is given (e.g., http://google.com), extract just google.com for scanning.

5. Code python

Python file: homoTool.py

```
import unicodedata
```

```
import string
```

```
from urllib.parse import urlparse
```

```
# Set of standard printable ASCII characters (A-Z, 0-9, punctuation, space)
```

```
standard_chars = set(string.ascii_letters + string.digits + string.punctuation + " ")
```

```
safe_invisible_chars = {'\n', '\r', '\t'}
```

```
# Check if character is standard ASCII
```

```
def is_allowed_standard_char(ch):
```

```
    return ch in standard_chars
```

Check if a character is suspicious (non-standard or unsafe control char)

```
def is_suspicious(ch):
```

```
    if ch in safe_invisible_chars:
```

```
        return False
```

```
    if ch in standard_chars:
```

```
        return False
```

```
    return True
```

Scan text for suspicious characters and return details

```
def scan_text(text):
```

```
    results = []
```

```
    for ch in text:
```

```
        if is_suspicious(ch):
```

```
            try:
```

```
                name = unicodedata.name(ch)
```

```
            except ValueError:
```

```
                name = "Unknown or Non-character"
```

```
                codepoint = f"U+{ord(ch):04X}"
```

```
                results.append((ch, name, codepoint))
```

```
    return results
```

Extract domain part from a full URL (or treat input as a domain)

```
def extract_domain(url):
```

```
    parsed = urlparse(url)
```

```
    return parsed.netloc or parsed.path
```

Complete scan pipeline for a domain or full URL

```
def scan_domain(domain_or_url):
```

```

domain = extract_domain(domain_or_url)

return scan_text(domain)

if __name__ == "__main__":

    url = input("Enter a domain or full URL to scan: ").strip()

    suspicious = scan_domain(url)

    if suspicious:

        print("Suspicious characters found:")

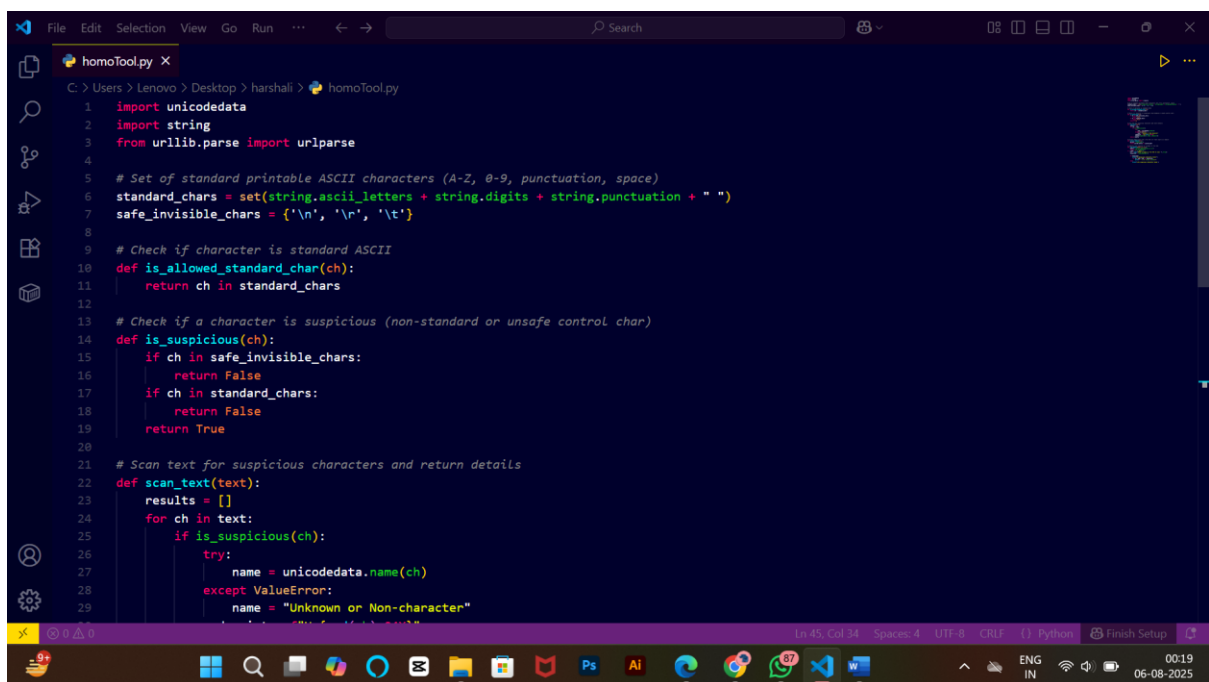
        for ch, name, code in suspicious:

            print(f" - {ch}: {name} ({code})")

    else:

        print("No suspicious characters found.")

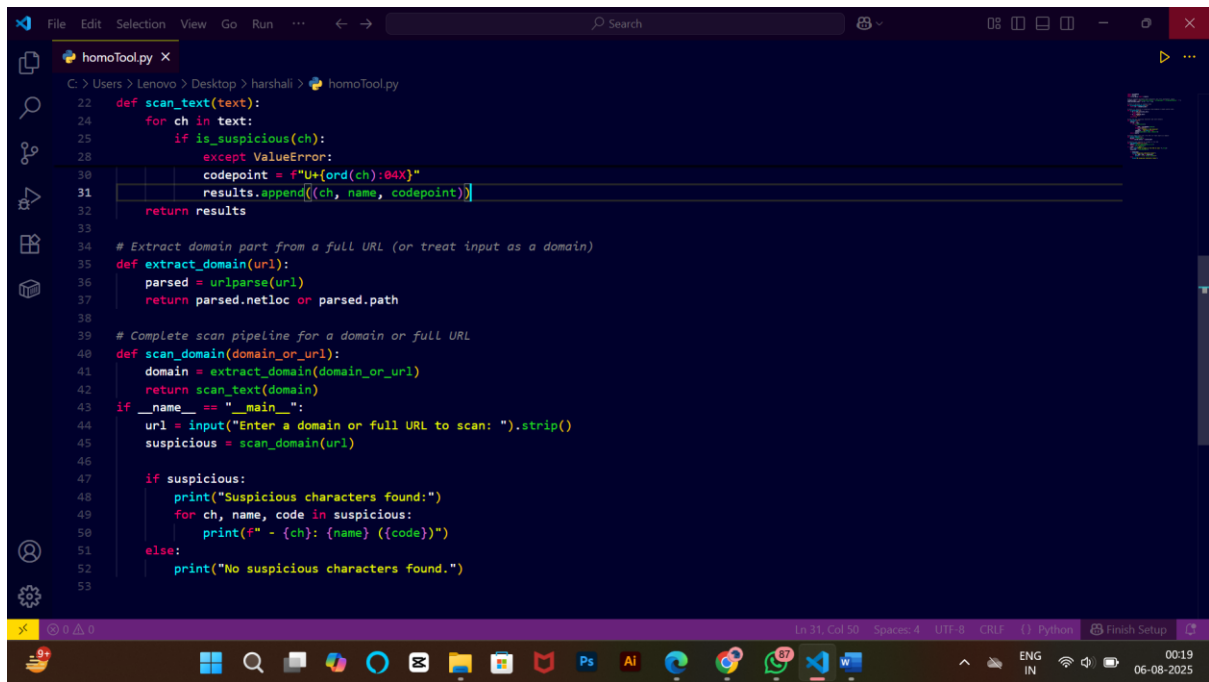
```



```

File Edit Selection View Go Run ... Search
homoTool.py X
C:\Users\Lenovo\Desktop>harshali\homoTool.py
1 import unicodedata
2 import string
3 from urllib.parse import urlparse
4
5 # Set of standard printable ASCII characters (A-Z, 0-9, punctuation, space)
6 standard_chars = set(string.ascii_letters + string.digits + string.punctuation + " ")
7 safe_invisible_chars = {'\n', '\r', '\t'}
8
9 # Check if character is standard ASCII
10 def is_allowed_standard_char(ch):
11     return ch in standard_chars
12
13 # Check if a character is suspicious (non-standard or unsafe control char)
14 def is_suspicious(ch):
15     if ch in safe_invisible_chars:
16         return False
17     if ch in standard_chars:
18         return False
19     return True
20
21 # Scan text for suspicious characters and return details
22 def scan_text(text):
23     results = []
24     for ch in text:
25         if is_suspicious(ch):
26             try:
27                 name = unicodedata.name(ch)
28             except ValueError:
29                 name = "Unknown or Non-character"

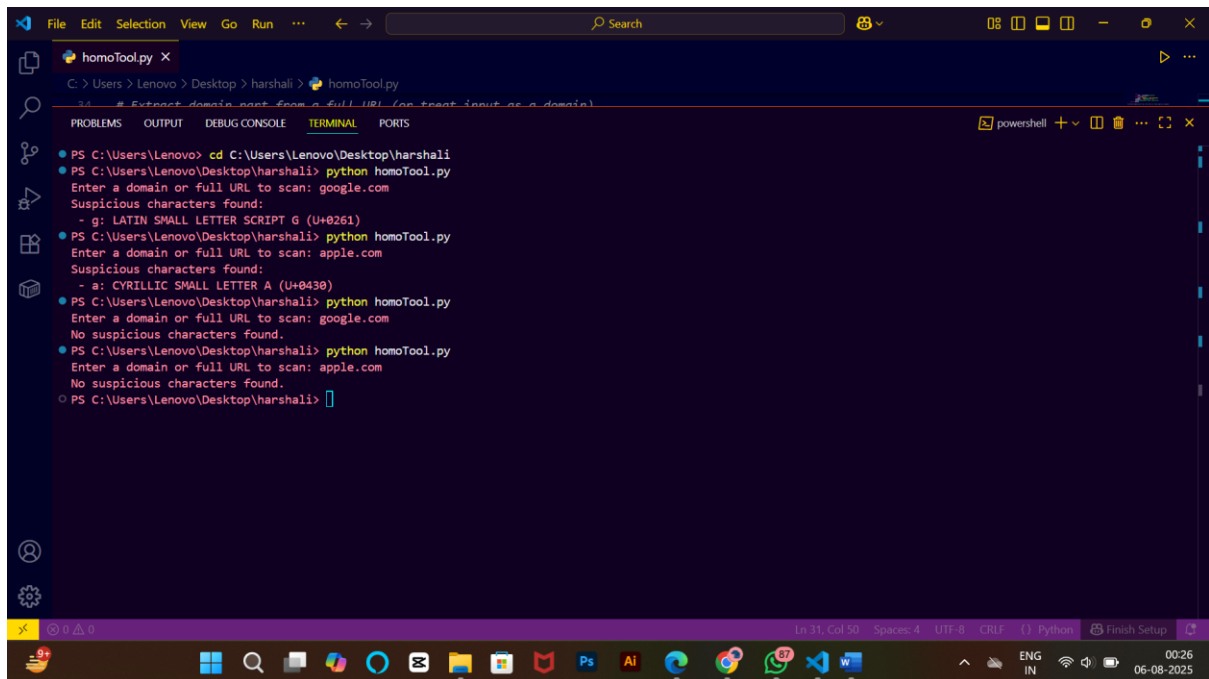
```



The screenshot shows a VS Code editor window with the file `homoTool.py` open. The script is a Python program designed to scan text or domains for suspicious characters. It includes functions for scanning text, extracting domains from URLs, and a main pipeline. The script is as follows:

```
22 def scan_text(text):
23     for ch in text:
24         if is_suspicious(ch):
25             except ValueError:
26                 codepoint = f"U+{ord(ch):04X}"
27                 results.append((ch, name, codepoint))
28     return results
29
30 # Extract domain part from a full URL (or treat input as a domain)
31 def extract_domain(url):
32     parsed = urlparse(url)
33     return parsed.netloc or parsed.path
34
35 # Complete scan pipeline for a domain or full URL
36 def scan_domain(domain_or_url):
37     domain = extract_domain(domain_or_url)
38     return scan_text(domain)
39
40 if __name__ == "__main__":
41     url = input("Enter a domain or full URL to scan: ").strip()
42     suspicious = scan_domain(url)
43
44     if suspicious:
45         print("Suspicious characters found:")
46         for ch, name, code in suspicious:
47             print(f" - {ch}: {name} ({code})")
48     else:
49         print("No suspicious characters found.")
```

6.Output



The screenshot shows the same VS Code editor window, but with the `TERMINAL` tab active. It displays the execution of the `homoTool.py` script. The output shows the script being run from a PowerShell prompt, with the user entering `google.com` and `apple.com` as domains to scan. The script identifies suspicious characters in `google.com` (LATIN SMALL LETTER SCRIPT G) and `apple.com` (CYRILLIC SMALL LETTER A).

```
PS C:\Users\Lenovo> cd C:\Users\Lenovo\Desktop\harshali
PS C:\Users\Lenovo\Desktop\harshali> python homoTool.py
Enter a domain or full URL to scan: google.com
Suspicious characters found:
 - g: LATIN SMALL LETTER SCRIPT G (U+0261)
PS C:\Users\Lenovo\Desktop\harshali> python homoTool.py
Enter a domain or full URL to scan: apple.com
Suspicious characters found:
 - a: CYRILLIC SMALL LETTER A (U+0430)
PS C:\Users\Lenovo\Desktop\harshali> python homoTool.py
Enter a domain or full URL to scan: google.com
No suspicious characters found.
PS C:\Users\Lenovo\Desktop\harshali> python homoTool.py
Enter a domain or full URL to scan: apple.com
No suspicious characters found.
PS C:\Users\Lenovo\Desktop\harshali>
```