Homographic Link Detection Tool

❖ Objective:

The purpose of this Poc is to demonstrate how malicious actors can use homoglyph characters to create phishing URLs that look visually identical to legitimate domains. We use a Python-based detection tool leveraging the homoglyph library to automatically identify such deceptive links.

❖ Background:

A homoglyp**h** is a character that looks like another character but comes from a different writing system (e.g., Latin vs Cyrillic).
Attackers register domains using homoglyphs to trick users into visiting malicious websites.
This is known as an IDN Homograph Attack**.**

Example:

Legit: google.com, Fake: google.com (contains Cyrillic "o" characters)

These two appear visually identical in most fonts but are different Unicode sequences

❖ Python Script:

```
import re
import sys

try:
import homoglyphs as hg
except ImportError:
print("Please install the homoglyphs package: pip install homoglyphs")
sys.exit(1)

def extract_links_from_text(text):
url_pattern = re.compile(
r'((?:https?://|www\.)[^\s<>"\'()]+|[a-zA-Z0-9-]+\.[a-zA-
Z]{2,}(?:/[^\s<>"\'()]*)?)',
re.IGNORECASE
)
raw_links = url_pattern.findall(text)
return [link.rstrip('.,);]>"\") for link in raw_links]

def is_link_suspicious(link, homoglyphs_obj):
ascii_versions = homoglyphs_obj.to_ascii(link)
return bool(ascii_versions and (link not in ascii_versions))
```

```
def check_links_in_text(text):
homoglyphs_obj = hg.Homoglyphs(languages={'en', 'ru', 'el'})
links = extract_links_from_text(text)
suspicious_links = [link for link in links if is_link_suspicious(link,
homoglyphs_obj)]

print(f"\nLinks found ({len(links)}):")
for link in links:
if link in suspicious_links:
print(f"  {link}  <-- (SUSPICIOUS)")
else:
print(f"  {link}")

print("\nSuspicious (potentially fake) links:")
if suspicious_links:
for link in suspicious_links:
print(f"  {link}")
else:
print("  None detected.")

if __name__ == "__main__":
text_to_check = """
https://google.com
https://google.com
example.com
www.paypa1.com
"""
check_links_in_text(text_to_check)
```

❖ Execution:

```
= RESTART: C:\python\homographic_link.py

Links found (3):
  https://google.com
  https://google.com  <-- (SUSPICIOUS)
  www.paypal.com

Suspicious (potentially fake) links:
  https://google.com
```

❖ Analysis:

The legitimate domain uses the Latin small letter "o" (U+006F), while the fake domain uses the Cyrillic small letter "o" (U+043E). Although these characters render identically in most browsers, they are actually different Unicode code points. This difference allows attackers to register domains that appear to be legitimate but resolve to entirely different destinations. When unsuspecting users visit these disguised domains, they may be exposed to phishing websites designed to steal credentials or distribute malware.

❖ Conclusion:

To reduce the risk of falling victim to such attacks, domain names should be normalized to ASCII using libraries like homoglyphs before being processed or displayed. Users and administrators can also enable punycode display in browsers to reveal the true structure of internationalized domains. In addition, implementing link validation within email filters and security gateways helps detect and block malicious homoglyph-based URLs before they can be clicked.