

Name Mohsin Nawaz

## BrainWave Matrix Solutions

### Cyber Security and Ethical Hacking Intern TASKS

Question : Build a Python Cyber Security - Phishing Link Scanner

#### Answer

Installation of python files in kali linux terminal :

```
(kali@kali)-[~]
$ sudo apt-get update
sudo apt-get install python3 python3-pip

Hit:1 http://http.kali.org/kali kali-rolling InRelease
Hit:2 https://dl.google.com/linux/chrome/deb stable InRelease
Get:3 https://download.docker.com/linux/debian bookworm InRelease [43.3 kB]
Fetched 43.3 kB in 3s (15.2 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.11.8-1).
python3-pip is already the newest version (24.2+dfsg-1).
The following packages were automatically installed and are no longer required:
  cython3 debtags debugedit distro-info-data finger gvimd-common kali-debtags libabsl20220623 libaio1 libbytes-random-secure-perl libcbor0.8
  libcrypt-random-seed-perl libdaxctl1 libdlt2 libfile-listing-perl libfont-afm-perl libfsverity0 libgphoto2-l10n libgumbo1
  libgupnp-igd-1.0-4 libgvm22 libhiredis0.14 libhtml-form-perl libhtml-format-perl libhtml-tree-perl libhttp-cookiejar-perl
  libhttp-cookies-perl libhttp-daemon-perl libhttp-negotiate-perl libio-multiplex-perl libipc-shareable-perl libjim0.81
  libmagiccore-6.q16-6-extra libmath-random-isaac-perl libmath-random-isaac-xs-perl libmosquitto1 libmozilla-publicsuffix-perl libmujs2
  libncurses5 libndctl6 libnet-cidr-perl libnet-http-perl libnet-ip-perl libnet-netmask-perl libnet-whois-ip-perl libnsl-dev
  libpaho-mqtt1.3 libpmem1 libpod-parser-perl libpthread-stubs0-dev libradcli4 libregexp-assemble-perl librpm-build9 librpm-sign9
  libstring-crc32-perl libstring-random-perl libtie-ixhash-perl libtinof5 libtirpc-dev libtry-tiny-perl libucl1 libwww-robotrules-perl
  libxml-regexp-perl libxml-writer-perl libxml-xpathengine-perl linux-headers-amd64 lua-lpeg medusa mosquito notus-scanner nsis
  nsis-common numba-doc openvas-scanner osdp-openvas perl-openssl-defaults pg-gvm python-apt-common python-odf-doc python-odf-tools
  python-tables-data python3-aioredis python3-apscheduler python3-apt python3-backcall python3-bottleneck python3-cryptography37
  python3-debian python3-defusedxml python3-diskcache python3-future python3-git python3-gitdb python3-gnupg python3-jdcal python3-llvmlite
  python3-mistune0 python3-numba python3-numexpr python3-odf python3-paho-mqtt python3-pandas python3-pandas-lib python3-pendulum
  python3-pickleshare python3-promise python3-psutil python3-py python3-pyexploitdb python3-pyfiglet python3-pyminifier python3-pypdf2
  python3-pyrsistent python3-pyshodan python3-pytz-deprecation-shim python3-pytzdata python3-quamash python3-requests-toolbelt
  python3-rfc3986 python3-rx python3-smmmap python3-tables python3-tables-lib python3-tld python3-unicodcsv python3-yaswfp python3-zapv2
  rpm rwho rdhwd sparta-scripts wapiti xsltproc
```

Installing the required libraries:

```
(kali@kali)-[~]
$ pip3 install requests beautifulsoup4 urllib3

Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.31.0)
Requirement already satisfied: beautifulsoup4 in /usr/lib/python3/dist-packages (4.12.3)
Requirement already satisfied: urllib3 in /usr/lib/python3/dist-packages (2.0.7)
Requirement already satisfied: soupsieve>1.2 in /usr/lib/python3/dist-packages (from beautifulsoup4) (2.5)
```

Create a Python script named `phishing_link_scanner.py` using a text editor like nano

```
GNU nano 8.1 phishing_link_scanner.py *
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse, urljoin
import re

def is_phishing(url):
    """
    Check if a URL contains common phishing indicators.
    """
    parsed_url = urlparse(url)
    domain = parsed_url.netloc

    # Check for short domain names
    if len(domain) < 10:
        return True

    # Check if the URL contains common phishing keywords
    phishing_keywords = ["login", "verify", "secure", "account", "update", "bank", "confirm", "password"]
    if any(keyword in url.lower() for keyword in phishing_keywords):
        return True

    # Check for HTTP (not HTTPS)
    if not url.startswith("https://"):
        return True

    return False

def detect_malicious_content(soup):
    """
    Detect malicious content by looking for suspicious elements in the page's HTML.
    """
    # Check for hidden iframes
    iframes = soup.find_all('iframe')
    for iframe in iframes:
        if 'hidden' in iframe.attrs.get('style', ''):
            return True

    # Check for suspicious forms (e.g., forms asking for sensitive information)
    forms = soup.find_all('form')
    for form in forms:
        action = form.get('action', '').lower()
        if any(keyword in action for keyword in ['login', 'verify', 'secure', 'password']):
            return True

    # Check for suspicious scripts
    scripts = soup.find_all('script')
    for script in scripts:
        if script.get('src') and 'tracker' in script.get('src').lower():
            return True

    return False

def scan_url(url):
    """
    Scan a URL to detect phishing or malicious content.
    """
```

```
GNU nano 8.1 phishing_link_scanner.py *
def detect_malicious_content(soup):
    """
    Detect malicious content by looking for suspicious elements in the page's HTML.
    """
    # Check for hidden iframes
    iframes = soup.find_all('iframe')
    for iframe in iframes:
        if 'hidden' in iframe.attrs.get('style', ''):
            return True

    # Check for suspicious forms (e.g., forms asking for sensitive information)
    forms = soup.find_all('form')
    for form in forms:
        action = form.get('action', '').lower()
        if any(keyword in action for keyword in ['login', 'verify', 'secure', 'password']):
            return True

    # Check for suspicious scripts
    scripts = soup.find_all('script')
    for script in scripts:
        if script.get('src') and 'tracker' in script.get('src').lower():
            return True

    return False

def scan_url(url):
    """
    Scan a URL to detect phishing or malicious content.
    """
```

```
GNU nano 8.1 phishing_link_s
return False

def scan_url(url):
    """
    Scan a URL to detect phishing or malicious content.
    """
    try:
        response = requests.get(url, timeout=10)
        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser')
            if detect_malicious_content(soup):
                return "Warning: The page contains suspicious elements!"
            title = soup.title.string if soup.title else 'No title'
            return f"Page title: {title}"
        else:
            return "Failed to retrieve page"
    except requests.exceptions.RequestException as e:
        return f"Request failed: {e}"

if __name__ == "__main__":
    url_to_scan = input("Enter the URL to scan: ")
    if is_phishing(url_to_scan):
        print("Warning: The URL looks suspicious!")
    else:
        scan_result = scan_url(url_to_scan)
        print(scan_result)
```

## Explanation of the Code:

- **is\_phishing Function:** This function checks for common phishing indicators like short domain names, common phishing keywords in the URL, and the use of HTTP instead of HTTPS.
- **detect\_malicious\_content Function:** This function checks the page's content for hidden iframes, suspicious forms, and scripts that might indicate malicious activity.
- **scan\_url Function:** This function fetches the webpage content using requests, parses it with BeautifulSoup, and uses detect\_malicious\_content to identify any suspicious elements.
- **Main Functionality:** The script takes a URL as input, checks for phishing indicators, and scans the webpage for malicious content.

Make the script executable so

```
(kali@kali)-[~]  
$ chmod +x phishing_link_scanner.py
```

Now executing the python file so:

```
(kali@kali)-[~]  
$ python3 phishing_link_scanner.py  
Enter the URL to scan: https://pftpedu.org/  
Page title: PFTP - Professional Freelancing Training Program
```

```
(kali@kali)-[~]  
$ python3 phishing_link_scanner.py  
Enter the URL to scan: https://www.kali.org/docs/  
Page title: Kali Docs | Kali Linux Documentation
```

```
(kali@kali)-[~]  
$ python3 phishing_link_scanner.py  
Enter the URL to scan: https://caniphish.com/free-phishing-test/phishing-website-templates  
Warning: The page contains suspicious elements!
```

## Enhancing the Phishing Link Scanner

### Integrate with Threat Intelligence APIs

- Consider integrating APIs like VirusTotal or Google Safe Browsing to cross-check URLs against a known database of phishing and malicious domains.

### Add Logging

- Implementing logging to record the results of each scan. Python's `logging` module can be used for this.

### Implement Machine Learning

- Add a machine learning component to classify URLs based on a dataset of known phishing and non-phishing URLs for more accurate detection.

## Security Considerations

- **Environment:** Ensure that to run the script in a secure environment since it fetches and processes content from external sites.
- **Ethical Use:** Always use the phishing link scanner ethically, ensuring the respect privacy and comply with legal guidelines.

This detailed implementation will provide us with a functional phishing link scanner in Kali Linux, capable of identifying potentially malicious URLs based on various indicators.

\*\*\*\*\*