
# WEB - 100

# Python Scripting Basics

## Training Task:

Web Server Technology Detection and Web Spider Script Report

**Prepared by:**

Maria Abu Sammour

Nadeen Moreb

Sondos Farrah

**Date:** 01/03/2025

# Table of Contents

# Table of Figures:

# 1. Overview:

This Python script determines the technology the web server is working on of a given URL, e.g (Apache, Microsoft, tomcat…). Based on this technology, it calls a list of default files and URLs, and it starts spidering inside the website and extracting internal links/folders and comparing them with that default list.

Then it generates a comparison matrix between the default URLs. The results are saved in a file for further analysis. The script relies on various Python libraries, including requests, BeautifulSoup, and pandas.

# 2. Key Features and Steps

- **Technology Detection**:

  Firstly, The script analyzes the HTTP response and reads the ‘Server’ header to detect the web server technology (e.g., Apache, Nginx, IIS).

- **Default File Detection:**

  After detecting server technology, the script loads the corresponding web server default files list (e.g., default_nginx.txt, default_apache.txt), which are used later as a reference to compare the extracted URLs of the given website.

- **Web Spidering:**

  The script recursively spiders inside the website, and gathers all internal URLs, which belong to the same domain as the target URL, up to a specific depth (depth of 4 here).

- **URL Comparison:**

  The found URLs are compared with the list of default files: URLs that match default files are classified as "default," while the rest are considered "non-default."

- **Cross Matrix Creation:**

  A cross-matrix comparison is generated to show which URLs are found in the default files and which are not. This matrix includes hashed values of the URLs to help avoid any exposure of sensitive information.

- **Results:**

  Finally, the script saves the results in a list of default URLs found, a list of non-default URLs found and the comparison matrix showing the default and non-default URLs.

# 3. Code and Used Functions

- **detect_technology(url)**:

  This function sends an HTTP request to the target URL and analyzes the 'Server' header to determine the server technology. It returns the server type as a string or None if detection fails.

```
1 usage
def detect_technology(url):
    """
    Detects the web server technology by analyzing the 'Server' header in the HTTP response.
    :param url: Target URL to analyze.
    :return: Server technology string or None if detection fails.
    """
    try:
        response = requests.get(url, timeout=5)
        return response.headers.get('Server', 'Unknown').lower()
    except requests.exceptions.RequestException as e:
        print(f"Error: {e}")
        return None
```

*Figure 1 detect technology*

- **get_default_file_path(server_header)**:

  Selects the associate default file list based on the server technology (Apache, Nginx, IIS, etc.). If the server technology is not recognized, a generic file is returned.

```python
def get_default_file_path(server_header):
    """
    Determines the default file list based on the detected server technology.
    :param server_header: Detected web server technology.
    :return: Filename of the corresponding default file list.
    """
    if "nginx/1.19.0" in server_header:
        return "default_nginx_1_19_0.txt"
    elif "apache/2.4.62" in server_header:
        return "default_apache_2_4_62.txt"
    elif "apache" in server_header:
        return "default_apache.txt"
    elif "nginx" in server_header:
        return "default_nginx.txt"
    elif "iis" in server_header:
        return "default_iis.txt"
    elif "tomcat" in server_header.lower():
        return "default_tomcat.txt"
    elif "lighttpd" in server_header.lower():
        return "default_lighttpd.txt"
    elif "jetty" in server_header.lower():
        return "default_jetty.txt"
    elif "hiawatha" in server_header.lower():
        return "default_hiawatha.txt"
    elif "abyss" in server_header.lower():
        return "default_abyss.txt"
    elif "zeus" in server_header.lower():
        return "default_zeus.txt"
    elif "tengine" in server_header.lower():
```

*Figure 2 get default file path*

- **load_default_files(file_path)**:

  Loads a list of default filenames from the specified file. These filenames are used as reference URLs to compare with the found URLs.

```python
def load_default_files(file_path):
    """
    Loads the default file list from the given file.
    :param file_path: Path to the default file list.
    :return: A list of default filenames.
    """
    try:
        with open(file_path, 'r') as file:
            return [line.strip() for line in file if line.strip()]
    except FileNotFoundError:
        print(f"Default file list not found: {file_path}")
        return []
```

*Figure 3 load default files*

- **hash_value(value)**:

  Computes the SHA-256 hash of a given string. We use to hash the URLs in the cross matrix comparison to achieve confidentiality.

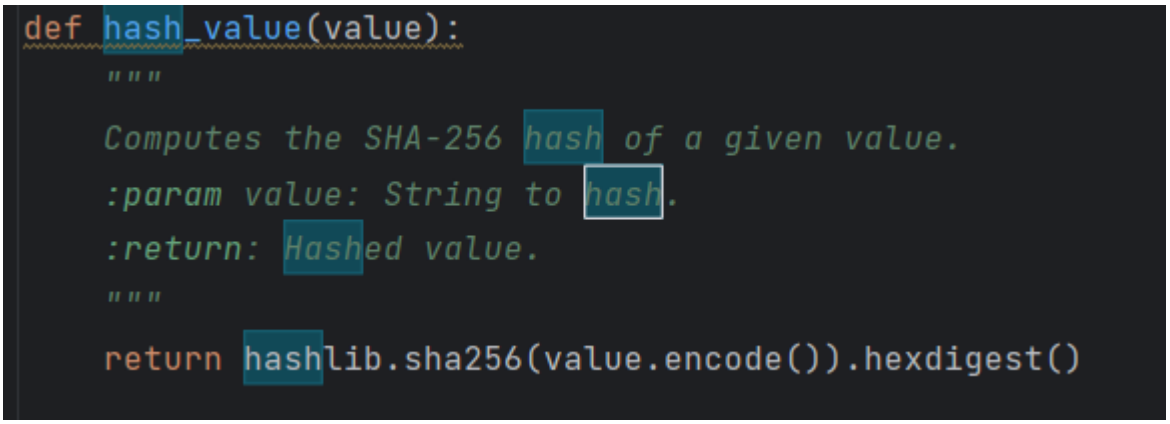


*Figure 4 hash value*

- **crawl_site(url, max_depth=4)**:

  This crawls the website starting from the given URL and recursively to the internal links up to a specified depth, to return a set of the found URLs.

```python
def crawl_site(url, max_depth=4):
    """
    Crawls a website to extract internal URLs up to a specified depth.
    :param url: Base URL to start crawling.
    :param max_depth: Maximum depth for recursive crawling.
    :return: A set of discovered URLs.
    """
    visited, found_urls = set(), set()
    to_visit = [url]
    for _ in range(max_depth):
        new_links = []
        for link in to_visit:
            if link in visited:
                continue
            try:
                response = requests.get(link, timeout=5)
                visited.add(link)
                soup = BeautifulSoup(response.text, 'html.parser')
                for a_tag in soup.find_all('a', href=True):
                    absolute_url = urljoin(url, a_tag['href'])
                    if absolute_url.startswith(url):
                        found_urls.add(absolute_url)
                        new_links.append(absolute_url)
            except requests.exceptions.RequestException:
                continue
        to_visit = new_links
    return found_urls
```

*Figure 5 crawl site*

- **compare_urls(found_urls, default_files, base_url)**:

  Compares the extracted URLs with the default file list and returns two sets: one containing the URLs that match default files, and the other containing URLs that do not match the default files.

```python
def compare_urls(found_urls, default_files, base_url):
    """
    Compares discovered URLs with the default file list.
    :param found_urls: Set of URLs found during crawling.
    :param default_files: List of default files that should exist.
    :param base_url: The base website URL.
    :return: Sets of default files found and non-default files.
    """
    default_urls = {urljoin(base_url, df) for df in default_files}
    return found_urls & default_urls, found_urls - default_urls
```

*Figure 6 compare urls*

- **create_cross_matrix(default_set, non_default_set, label1, label2)**:

  Generates Pandas DataFrame that contains the comparison matrix of default and non-default URLs, showing the hashed values along with which URLs are considered default or non-default.

```python
def create_cross_matrix(default_set, non_default_set, label1, label2):
    """
    Generates a comparison matrix between default and non-default URLs.
    :param default_set: Set of URLs that match default files.
    :param non_default_set: Set of URLs that do not match default files.
    :param label1: Label for default URLs.
    :param label2: Label for non-default URLs.
    :return: Pandas DataFrame containing the comparison matrix.
    """
    all_items = sorted(default_set | non_default_set)
    matrix = [[hash_value(item), item in default_set, item in non_default_set] for item in all_items]
    return pd.DataFrame(matrix, columns=['Hashed Item', label1, label2])
```

*Figure 7 create cross matrix*

- **save_results(default_found, non_default, url_matrix)**:

  Saves the crawling and comparison results to a text file (spidering_results.txt), which includes the lists of default and non-default URLs, and the comparison matrix.

```python
def save_results(default_found, non_default, url_matrix):
    """
    Saves the results of the web scanning process to a file.
    :param default_found: Set of default URLs found.
    :param non_default: Set of non-default URLs found.
    :param url_matrix: Pandas DataFrame containing the comparison matrix.
    """
    with open("spidering_results.txt", "w") as file:
        file.write("=== Default URLs Found ===\n")
        file.writelines(f"{url}\n" for url in default_found)
        file.write("\n=== Non-Default URLs Found ===\n")
        file.writelines(f"{url}\n" for url in non_default)
        file.write("\n=== Cross Matrix Comparison ===\n")
        file.write(url_matrix.to_string(index=False))
```

*Figure 8 save results*

# 4. Execution Flow

1. The script prompts the user to input a target URL (e.g., http://example.com).
2. It detects the server technology of the website.
3. Based on the detected technology, the script loads the corresponding default file list.
4. The website then is crawled to extract internal URLs.
5. The found URLs are compared to the list of default files to classify them as default or non-default.
6. A cross matrix is generated to compare the URLs.
7. The results, including the URLs and comparison matrix, are saved to a text file for further review.

# 5. Results

Test Url: [http://testphp.vulnweb.com/](http://testphp.vulnweb.com/)

```
C:\Users\AL-YAMEN\PycharmProjects\pythonWebSpider\venv\Scripts\python.exe C:\Users\AL-YAMEN\PycharmProjects\pythonWebSpider\web_s.py
Enter target URL (e.g., http://example.com): http://testphp.vulnweb.com/
Results saved to spidering_results.txt

Process finished with exit code 0
```

*Figure 9 testing a url*

After the script completes, the results are saved in the spidering_results.txt file. This file includes:

- The list of default URLs found

```
=== Default URLs Found ===
http://testphp.vulnweb.com/index.php
```

*Figure 10 default URLs list*

- The list of non-default URLs found

```
=== Non-Default URLs Found ===
http://testphp.vulnweb.com/artists.php?artist=2
http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg
http://testphp.vulnweb.com/listproducts.php?artist=2
http://testphp.vulnweb.com/guestbook.php
http://testphp.vulnweb.com/privacy.php
http://testphp.vulnweb.com/listproducts.php?artist=3
http://testphp.vulnweb.com/signup.php
http://testphp.vulnweb.com/listproducts.php?cat=2
http://testphp.vulnweb.com/artists.php?artist=1
http://testphp.vulnweb.com/artists.php?artist=3
http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg
http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg
http://testphp.vulnweb.com/listproducts.php?cat=1
http://testphp.vulnweb.com/product.php?pic=3
http://testphp.vulnweb.com/product.php?pic=1
http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg
http://testphp.vulnweb.com/AJAX/index.php
http://testphp.vulnweb.com/product.php?pic=4
http://testphp.vulnweb.com/?pp=12
http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg
http://testphp.vulnweb.com/Details/web-camera-a4tech/2/
http://testphp.vulnweb.com/product.php?pic=2
http://testphp.vulnweb.com/product.php?pic=6
http://testphp.vulnweb.com/listproducts.php?artist=1
http://testphp.vulnweb.com/product.php?pic=7
http://testphp.vulnweb.com/userinfo.php
http://testphp.vulnweb.com/hpp/
http://testphp.vulnweb.com/listproducts.php?cat=4
http://testphp.vulnweb.com/listproducts.php?cat=3
```

*Figure 11 Non-default URLs list*

- The cross-matrix comparison of the URLs with hashed values

```
=== Cross Matrix Comparison ===
                                                  Hashed Item  Default URL  Non-Default URL
a3961d2837c3f4ac3c4cf02022f0352adbf09d06f215580b03ef286ff4a9232c        False            True
accd48f2529889f3243f8ef69427bcd6502a12e04132e634a353943507d73d8d        False            True
7c09633b7777bdf4aac3ab746c5a500c4832e027b54364a6d3d72fb78ffb1cf2        False            True
3280557caa415c54ba0fd6725e423c2dcd6ac66fc43dba29024e8c1c40b2723b        False            True
37ee42e8e10d317c4493bb27acb375306a95215b22e2db2d77e7377b5ab6c416        False            True
195d99dacfac794852241840d34f7ff5075475f1a2393770f59ad501497382b6        False            True
d1c45a178c2830164b33250ce40591a6c685b3a1b9dfe80d616cc0b878f643e0        False            True
c3819614be4f730d192bd9997f726f380ec54199cf5c01879d3492948766c017        False            True
52dd295714d338867607ac54ca8d72fc05156b666469b4d19ec88051303fb502        False            True
b2d18626f0f0b75a9e649153b03f400e9e200d95cca375bb8e846aefaaa12181        False            True
0224a39c4b0c592689002721e6e6e0c809da579fdb338839e460ef5ccd2d8491        False            True
967197804df7f69317d0a6d5aed47b2fbab88e762d7cb3a00d4d71920827f522        False            True
5e55c3ec819c35d653826abfae52a7ea7df1c53330857a76b76197695a2bbde8        False            True
c789ad9d7b69bee699d248f2420f064a1292f1b784b7d86e9f2d56e1e52ab362        False            True
5304e0614c8621cb12bb967fab2ab424c0bfa81a931edf0716477ab308e8b04a        False            True
06057b17d6d5869ef5013e5d92a14b10c731eea1a0b28e6c91fe351412867a9f        False            True
a51de6211847cc87a65f296394d77a59abe740380f5876773ee21fd7c862772b         True           False
2104c4a2351dee3160ec20d5e1e9dcd901d35a31423e68e149f15bbccd832032        False            True
986db4efb5222fb69930c95d830a3ba28387a466546e453e49599518f7771828        False            True
02a69c5087cf037aaaba0ccb901315f1f9af66da0185c4eb426280d0668f3895        False            True
7f424b60126f0426e2b56517b740687dbdd39543b2bfa60a8f563e39cec7e020        False            True
e9467d9e55df9c8dd89de2fef597508a6735e399230ac6d5d1c31fc04fc502e4        False            True
5507004c3a3ab6e0f442916a78c676cb1b5c83cacb3c0659b401916535268495        False            True
ccb89ad15a9ad37da5ed8da7224c9a7012a33d07cbaf7126d044360f7a13df75        False            True
4418d78255061243f2aa055546bb8fdcd1e9fb87e159ef4bd56c23e992a9ee49        False            True
c09df570d596461750ea9f62bc89c9ec781b3ae7ec45d9cdf86339cac5fdca76        False            True
f39f74c972a573c4a61702191862bc29a0c10dbbb86008a9a5a80760a319db33        False            True
644a4e36050a11bb15bdaad9ee9710fab59b096e80d6b90b5f7620fd80540828        False            True
```

*Figure 12 cross matrix comparison*

# 6. Conclusion

This Python script serves as a useful tool as a web spider that lists internal URLs of a given website by comparing it against a list of default files based on the web server technology. It can help in identifying potential vulnerabilities related to the presence of default web server files and configurations.