Difficulty: Tier-0

Learning Outcome: Students will learn how to interact programmatically with web applications using high-level libraries.

GitHub Classroom:

Description:

Web scrapers or spiders are common utilities used by security professionals. These tools are commonly used by software such as web application vulnerability scanners to enumerate content on websites. They allow web application vulnerability scanners to discover additional components of the web application that should be scanned for vulnerabilities. This is done by looking for references to other resources within the HTML of a site.

For example, at the time of writing this assignment, www.rit.edu the following references in its html:

```
<link rel="image_src" href="https://www.rit.edu/_assets/images/og-image.jpg" />
```

```
</a></li><li class="dropdown-item" id="menu-item-0-1-2"><a href="/uniquely-rit" class="  ">
        Uniquely RIT
```

```
</a></li><li class="dropdown-item" id="menu-item-0-5-3"><a href="/research/centers-and-institutes" class="  ">
        Key Research Centers and Institutes
```

All of these examples are references to resources that exist on the www.rit.edu domain. There may be resources that exist on other domains as well. For example, at the time of writing, www.rit.edu following *external* references.

```
<iframe src="https://www.googletagmanager.com/ns.html?id=GTM-TKNM7FX" height="0" width="0" style="display:none;visibility:hidden"></iframe>
```

```
</a></li><li class="dropdown-item" id="menu-item-0-5-6"><a href="https://library.rit.edu/" class="  ">
        Libraries
```

The former example is clearly a resource not hosted on www.rit.edu. The later example, however, should properly be considered an “external” link as it references a subdomain. The host library.rit.edu is very likely a different underlying host than www.rit.edu.

In this assignment, you will create a web scraper in Python. This web scraper will take three pieces of input: a domain, a URL, and a depth. The output of the web scraper will be a file containing all links to all resources discovered bounded by the domain (1st input) and depth (3rd) input.

Your Python script will begin at the URL provided as the second input. It will search the HTML/CSS/JS for all links. Any links within the HTML/CSS/JS that are on the domain provided as the first input will be recorded and followed. Following a link will mean fetching the HTML/CSS/JS at that URL and searching it for additional links. Each time you follow a link, the depth will increase by 1.

For example, suppose your initial input is www.rit.edu, www.rit.edu, and depth=2. www.rit.edu would have a depth of 0 and should be scanned for all links.

Suppose you discover a link to library.rit.edu within that HTML/CSS/JS. That link should not be recorded as it resides on a different subdomain.

Suppose you discover a link to /uniquely-rit or a link to www.rit.edu/careerservices on www.rit.edu. These resources exist on the domain provided as input #1. You would record each of these as links discovered. Your script should then request the HTML/CSS/JS for each of these and scan the HTML for additional links (which would have a depth of 1. Suppose that www.rit.edu/careerservices has a reference to www.rit.edu/careerservices/news. You should follow *that* link, which would have a depth of 2. Any links on the www.rit.edu domain discovered on www.rit.edu/careerservices/news should be recorded, but not followed, as their depth would be 3.

In a nutshell, you are being asked to iteratively or recursively search a series of strings that happened to be fetched from remote sources.

Finally, you should ensure that duplicates are removed. This can be done as the iteration is occurring or as a post-processing step as you prefer.

Note: As the depth increases, the time taken to run your script will almost certainly increase significantly.

Assumptions/Recommendations:

- You may assume that the user input is valid
- You do not need to and it is not recommended that you test your script for very large depth values
- You may assume that only need to search for links in .html, .css, or .js files.
- You should record, but do not need to search, files that are likely to be media (.jpg, .png, .mp4, etc.)
- You should assume that links which do not have an explicit file type (ex: www.rit.edu/careerservices) have HTML content.

- There are likely to be edge cases that occur which are not explicitly covered by this project description. You may make your own judgement calls about how to address those edge cases, but make sure that you document this assumption in your code.
- You do not need to worry about sites that reject HTTP requests from your script. Amazon, for example, filters out HTTP requests based on the User-Agent to prevent scraping. Although this is an easy control to bypass, it is out of scope for this assignment.
- It is not recommended that you test this against small websites or servers. There is the possibility that use of this script may be detected as a denial of service attempt by security controls such as network firewalls or web application firewalls.
- It is recommended, but not required, that you use multi-threading to improve the efficiency of your scraper. Scrapers that are not multi-threaded may take a large amount of time to execute and hamper testing.
    - Students have reported that similar projects in past semesters have taken so long that it hampered development/testing.
- You may make use of HTTP/HTML libraries such as urllib, requests, lxml, or beautifulsoup. You may not make use of a library that does the iteration or recursion for you.
- You may not invoke a command-line too, such as dirbuster, OWASP ZAP, etc. to do the spidering for you.

Requirements:

- Your submission must be in Python3 (unless explicit permission to use another language has been granted).
- Your submission must be thoroughly documented.
- Your input must be exactly three command-line arguments in the order listed above: domain in scope, starting URL, depth.
- Your output must be exactly one text file containing all of the links.

Deliverables:

a. By 8:00 PM on 09/01/2022, you must submit the following to the DropBox on MyCourses:
    a. A copy of your Python script
        i. A zip is fine if MyCourses gives you issues uploading .py files.
        ii. This will be used to ensure your GitHub repository has not been changed.
    b. A GitHub classroom link to the repository containing your submission. The contents of this GitHub repository must match your uploaded file.