CS 3700 - Networks and Distributed Systems

Project 4: Web Crawler

This project is due at 11:59pm on Wednesday, March 25, 2020.

Description

This assignment is intended to familiarize you with the HTTP protocol. HTTP is (arguably) the most important application level protocol on the Internet today: the Web runs on HTTP, and increasingly other applications use HTTP as well (including Bittorrent, streaming video, Facebook and Twitter's social APIs, etc.).

Your goal in this assignment is to implement a web crawler that gathers data from a fake social networking website that we have set up. The site is available here: Fakebook.

What is a Web Crawler?

A web crawler (sometimes known as a robot, a spider, or a screen scraper) is a piece of software that automatically gathers and traverses documents on the web. For example, lets say you have a crawler and you tell it to start at www.wikipedia.com. The software will first download the Wikipedia homepage, then it will parse the HTML and locate all hyperlinks (i.e. anchor tags) embedded in the page. The crawler then downloads all the HTML pages specified by the URLs on the homepage, and parses them looking for more hyperlinks. This process continues until all of the pages on Wikipedia are downloaded and parsed.

Web crawlers are a fundamental component of today's web. For example, Googlebot is Google's web crawler. Googlebot is constantly scouring the web, downloading pages in search of new and updated content. All of this data forms the backbone of Google's search engine infrastructure.

Fakebook

We have set up a fake social network for this project called Fakebook. Fakebook is a very simple website that consists of the following pages:

- **Homepage**: The Fakebook homepage displays some welcome text, as well as links to several random Fakebook users' personal profiles.
- **Personal Profiles**: Each Fakebook user has a profile page that includes their name, some basic demographic information, as well as a link to their list of friends.
- **Friends List**: Each Fakebook user is friends with one or more other Fakebook users. This page lists the user's friends and has links to their personal profiles.

To browse Fakebook, you must first login with a username and password. We will email each student to give them a unique username and password.

DO NOT TEST YOUR CRAWLERS ON PUBLIC WEBSITES

Many web server administrators view crawlers as a nuisance, and they get very mad if they see strange crawlers traversing their sites. **Only test your crawler against Fakebook. Do not test it against any other website.**

High Level Requirements

Your goal is to collect 5 *secret flags* that have been hidden somewhere on the Fakebook website. The flags are unique for each student, and the pages that contain the flags will be different for each student. Since you have no idea what pages the secret flags will appear on, and the Fakebook site is very large (tens of thousands of pages), your only option is to write a web crawler that will traverse Fakebook and locate your flags. If you chose to work in a team, you must submit the flags for every member on the team to qualify to receive full points.

Your web crawler must execute on the command line using the following syntax:

```
./webcrawler [username] [password]
```

`username` and `password` are used by your crawler to log-in to Fakebook. You may assume that the root page for Fakebook is available at http://fring.ccs.neu.edu/fakebook/. You may also assume that the log-in form for Fakebook is available at http://fring.ccs.neu.edu/accounts/login/?next=/fakebook/.

Your web crawler should print **exactly fives lines of output**: the five *secret flags* discovered during the crawl of Fakebook. If your program encounters an unrecoverable error, it may print an error message before terminating. Secret flags may be hidden on any page on Fakebook, and their exact location on each page may be different. Each secret flag is a 64 character long sequences of random alphanumerics. All secret flags will appear in the following format to make them easier to identify:

```
<h2 class='secret_flag' style="color:red">FLAG: 64-characters-of-random-alphanumerics</h2>
```

Language

You can write your code in whatever language you choose, as long as your code compiles and runs on **unmodified** Khoury College Linux machines **on the command line**. Do not use libraries that are not installed by default on the Khoury College Linux machines, or that are disallowed for this project. You may use IDEs (e.g. Eclipse) during development, but make sure you code has **no dependencies** on your IDE and do not turn in your project without a Makefile.

HTTP and Libraries

All HTTP request and response code must be written from scratch. You need to implement creating and sending HTTP requests and parse HTTP responses. You may use any libraries available on login.khoury.neu.edu to create socket connections, parse URLs, and parse HTML. However, you may not use **any** libraries/modules/etc. that implement HTTP or manage cookies for you.

The following libraries are known to be legal/illegal in the context of this assignment:

- Legal
  - Python
    - *socket*
    - *parseurl*
    - *html*
    - *html.parse*
    - *urllib.parse*
    - *xml*

- Illegal
  - Python
    - *urllib*
    - *urllib2*
    - *html5lib*
    - *httplib*
    - *lxml*
    - *requests*
    - *pycurl*
    - *cookielib*
    - *BeautifulSoup*
  - Java
    - *java.net.CookieHandler*
    - *java.net.CookieManager*
    - *java.net.HttpCookie*
    - *java.net.HttpUrlConnection*
    - *java.net.URLConnection*
      (Note that this includes *java.net.URL::openConnection* and *java.net.URL::openStream*)
    - *java.net.URL::getContent*
    - *Jsoup*

Please post any questions about the legality of any libraries to Piazza. It is much safer to ask ahead of time, rather than turn in code that uses a questionable library and receive points off for the assignment after the fact.

Implementation Details and Hints

In this assignment, your crawler must implement HTTP/1.1 (not 0.9 or 1.0). This means that there are certain HTTP headers like *Host* that you must include in your requests (i.e. they are required for all HTTP/1.1 requests). We encourage you to implement *Connection: Keep-Alive* (i.e. pipelining) to improve your crawler's performance (and lighten the load on our server), but this is not required, and it is tricky to get correct. We also encourage you to implement *Accept-Encoding: gzip* (i.e. compressed HTTP responses), since this will also improve performance for everyone, but this is also not required. If you want to get crazy, you can definitely speed up your crawler by using multithreading or multiprocessing, but again this is not required functionality.

One of the key differences between HTTP/1.0 and HTTP/1.1 is that the latter supports *chunked encoding*. HTTP/1.1 servers may break up large responses into chunks, and it is the client's responsibility to reconstruct the data by combining the chunks. Our server may return chunked responses, which means your client must be able to reconstruct them. To aid in debugging, you might consider using HTTP/1.0 for your implementation. Once you have a working 1.0 implementation, you can switch to 1.1 and add support for chunked responses.

In order to build a successful web crawler, you will need to handle several different aspects of the HTTP protocol:

- HTTP GET - These requests are necessary for downloading HTML pages.
- HTTP POST - You will need to implement HTTP POST so that your code can login to Fakebook. As shown above, you will pass a username and password to your crawler on the command line. The crawler will then use these values as parameters in an HTTP POST in order to log-in to Fakebook.
- Cookie Management - Fakebook uses cookies to track whether clients are logged in to the site. If your crawler successfully logs in to Fakebook using an HTTP POST, Fakebook will return a session cookie to your crawler. Your crawler should store this cookie, and submit it along with each HTTP GET request as

it crawls Fakebook. If your crawler fails to handle cookies properly, then your software will not be able to successfully crawl Fakebook.

In addition to crawling Fakebook, your web crawler must be able to correctly handle HTTP status codes. Obviously, you need to handle 200, since that means everything is okay. Your code must also handle:

- 301 - Moved Permanently: This is known as an HTTP redirect. Your crawler should try the request again using the new URL given by the server in the *Location* header.
- 403 - Forbidden and 404 - Not Found: Our web server may return these codes in order to trip up your crawler. In this case, your crawler should abandon the URL that generated the error code.
- 500 - Internal Server Error: Our web server may **randomly** return this error code to your crawler. In this case, your crawler should re-try the request for the URL until the request is successful.

I highly recommend the HTTP Made Really Easy tutorial as a starting place for learning about the HTTP. Furthermore, the developer tools built-in to Chrome and Firefox are both excellent for inspecting and understanding HTTP requests.

In addition to HTTP-specific issues, there are a few key things that all web crawlers must do in order function:

- **Track the Frontier**: As your crawler traverses Fakebook it will observe many URLs. Typically, these uncrawled URLs are stored in a queue, stack, or list until the crawler is ready to visit them. These uncrawled URLs are known as the frontier.
- **Watch Out for Loops**: Your crawler needs to keep track of where it has been, i.e. the URLs that it has already crawled. Obviously, it isn't efficient to revisit the same pages over and over again. If your crawler does not keep track of where it has been, it will almost certainly enter an infinite loop. For example, if users A and B are friends on Fakebook, then that means A's page links to B, and B's page links to A. Unless the crawler is smart, it will ping-pong back and forth going A->B, B->A, A->B, B->A, ..., etc.
- **Only Crawl The Target Domain**: Web pages may include links that point to arbitrary domains (e.g. a link on google.com that points to cnn.com). **Your crawler should only traverse URLs that point to pages on fring.ccs.neu.edu**. For example, it would be valid to crawl *http://fring.ccs.neu.edu/fakebook/018912/*, but it would not be valid to crawl *http://www.fakebook.com/018912/*. Your code should check to make sure that each URL is in the target domain before you attempt to visit it.

Logging in to Fakebook

In order to write code that can successfully log-in to Fakebook, you will need to reverse engineer the HTML form on the log-in page. *Carefully* inspect the form's code. It is not be as simple as it initially appears. The key acronym to be on the lookout for is *CSRF*.

Submitting Your Project

Before turning in your project, you and your partner must register your group. To register yourself in a group, execute the following script:

```
$ /course/cs3700sp20/bin/register project4 [team name]
```

This will either report back success or will give you an error message. If you have trouble registering, please contact the course staff. You and your partner must run this script with the same `[team name]`. This is how we know you are part of the same group.

To turn-in your project, you should submit your (thoroughly documented) code along with three other files:

- A Makefile that compiles your code.
- A plain-text (no Word or PDF) README.md file. In this file, you should briefly describe your high-level approach, any challenges you faced, and an overview of how you tested your code.
- A file called *secret_flags*. This file should contain the *secret flags* of all group members, one per line, in plain ASCII. For example, a group of two should have a file with exactly ten lines in it.

Your README.md, Makefile, secret_flags file, source code, etc. should all be placed in a directory. You submit your project by running the turn-in script as follows:

```
$ /course/cs3700sp20/bin/turnin project4 [project directory]
```

[project directory] is the name of the directory with your submission. The script will print out every file that you are submitting, so make sure that it prints out all of the files you wish to submit! The turn-in script will not accept submissions that are missing a README.md, a Makefile, or a secret_flags file. **Only one group member needs to submit your project.** Your group may submit as many times as you wish; only the last submission will be graded, and the time of the last submission will determine whether your assignment is late.

Double Checking Your Submission

To try and make sure that your submission is (1) complete and (2) will work with our grading scripts, we provide a simple script that checks the formatting of your submission. This script is available on the Khoury College Linux machines and can be executed using the following command:

```
$ /course/cs3700sp20/code/project4/project4_format_check.py [path to your project directory]
```

This script will attempt to make sure that the correct files (e.g. README.md, secret_flags, and Makefile) are available in the given directory, that your secret_key file contains at least ten 64-byte keys, that your Makefile will run without errors (or is empty), and that after running the Makefile a program named webcrawler exists in the directory. The script will also try to determine if your files use Windows-style line endings (\r\n) as opposed to Unix-style line endings (\n). If your files are Windows-encoded, you should convert them to Unix-encoding using the dos2unix utility before turning in.

Grading

This project is worth 8% of your final grade. You will receive full credit if:

1. Your code compiles, runs, and produces the expected output
2. You have not used any illegal libraries; and
3. You successfully submit the *secret flags* of all group members

All student code will be scanned by plagiarism detection software to ensure that students are not copying code from the Internet or each other.

You can see your grades for this course at any time by using the gradesheet program that is available on the Khoury College machines.

```
$ /course/cs3700sp20/bin/gradesheet
```