Hacking the Great Firewall (HtGFW) User Manual

Mitchell Edwards

Introduction

By design, the greatest value of the Hacking the Great Firewall (HtGFW) platform resides in the data that I've already gathered and supplied for the user. However, I also include the source code for the Dump, Check, and Analysis modules as well as the Zhihu scraping API. These portions of the HtGFW platform are designed to be shared, changed, and built into new projects. Below, I will include details on the functionality built within each of the three modules. The functions are defined in function call format, ie **ReturnType functionName(ArgType arg1, ArgType arg2):** with a brief definition and description below.

Zhihu API

Requests.get() get_url(String url)

The **Get_Url** function is a wrapper for the requests.get() function, and returns the response from calling requests.get() with hardcoded cookies and headers and using the url (String) that a user supplies in the function call. This function is meant to fetch the url of a question or answer type post, while the **Get_Url_Zl** is designed to fetch posts from **Zhunlan**, which are long form posts classified as articles.

Requests.get() get_url_zl(String url):

The **Get_Url_Zl** function is a wrapper for the requests.get() function, and returns the response from calling requests.get() with hardcoded cookies and headers and using the url (String) that a user supplies in the function call. This function is meant to fetch the url of a long-form article type post, while the **Get_Url** is designed to fetch questions and answer type posts.

Bool ping_url(String url):

The **Ping_Url** function takes a url (String) and returns True if the status_code of the response is equal to 200, and False if the url is unreachable. This function is designed to work with the **Check** module to return the status of a post and determine whether or not it has been censored.

Bool ping_topic(String topic):

The **Ping_Topic** function takes a Topic Code (String; a unique identifier for each topic found in the URL pointing to the subtopic) and returns True if the topic still exists and False if it does not. This functionality isn't being used in the project as of now, as I'm not monitoring any topics I believe are sensitive enough to be taken down wholesale, but could be useful in the future.

String getTopicName(String topic):

The **GetTopicName** function takes a Topic Code (String; a unique identifier for each topic found in the URL pointing to the subtopic) and returns the name of the topic. This is useful when doing analysis, where the user can view the statistics coupled with the name of the topic instead of an arbitrary Topic Code.

String getPostBody(String postID, String posttype):

The **GetPostBody** function takes a postID (String; a unique numerical identifier for each post on each topic) and a post type (String; either question, answer, or article) and returns the body of the post as a String.

String getPostTitle(String postID, String posttype):
The **GetPostTitle** function functions the same as the **GetPostBody** function, accepting the same arguments, but returns the title of the post as a String.

String dumpTopic(String topicID):
The **DumpTopic** function forms the main functionality of the Zhihu API pertaining to the HtGFW platform. This takes a Topic Identifier as an argument and loops through the topic, returning 5 or so posts from the 'Hot' subsection (**dumpTopic_hot();** containing the most popular posts) and 5 or so posts from the 'Unanswered' subsection (**dumpTopic_unanswered();** containing newer posts that haven't been answered or discussed yet). The return value is a string containing the Post Type, Post ID, Author, Title, Link, and Content, each variable separated by one newline ('\n') and each post separated by two newlines ('\n\n'). The main function handles calls to both the **dumpTopic_hot()** function and the **dumpTopic_unanswered()** function, so a user only needs to call the **dumpTopic()** function with the TopicID to return posts from both subsections.

Dump Module
The Dump module takes in a hardcoded or user supplied list of topics and loops through each, 'Dumping' each individual post into a text file in the directory denoted by './TopicID/PostID.' The purpose of the **Dump Module** is to put the **dumpTopic** function of the Zhihu API into action by calling it on a list of topics supplied by the user. Coupled with automation functionality like the supplied shell scripts (**dump#.sh**) or, better yet, **crontab** in Linux operating systems, this function can be run automatically at different intervals. The hardcoded sleep() function calls are to be viewed as recommendations, as I tweaked the length of the sleep cycles throughout development and found these worked best.

Check Module
The Check module takes a hardcoded or user supplied list of topics and expects a file structure in the current directory created by the Dump module. It loops through the text file and checks each and every post within every Topic directory listed in the text file. The Check module uses the ping_url() function in the Zhihu API to check if a post still exists. If not, it copies the original text file into the directory './Censored/topicID/postID' to be analyzed as a newly censored post. The Check module then erases the file at the original location.

Analysis Module
The most rudimentary of the three, the only functionality currently in the Analysis module is to print a report of the total posts gathered (**totalPosts()**), the total posts censored (**totalCensored()**) and print a report containing the total posts gathered and censored as well as a breakdown of how many posts are censored in each topic (**printCensoredByTopic()**). This is the module that I believe I can improve on the most, but have yet to decide what kind of analysis I can do on the limited data set that I have.