NANKAI UNIVERSITY

College of Electronic Information and Optical Engineering

Development of Website Update Detection System

Haoyu Gong

Advisor: Yigang Han

***Abstract*-Website update is one of the important ways to obtain new information in people's daily life, and it is also the foundation of knowledge of many Web development theories. This article uses the method of detecting cookie changes to judge whether the website is updated, obtains the website page content at the same time when the update occurs, and notifies the user via email. The whole article starts with discussing several ways to judge whether a website is updated. Through understanding and exploring the HTTP protocol and website cookie mechanism, it gradually deepens the principle of website updating and completes the realization of system functions. The theoretical exploration phase of this paper has strong applicability and practicality, guide the implementation of the detection system, and it is the knowledge base for many Web developments. The cookie detection update method used in the system as a whole is not only more accurate than other methods, but most of the function modules can be run separately.**

I. INTRODUCTION

In the information age, one of the main ways people get the latest news is through website updates, and there are many ways to detect website updates. The main research object of this article is the website that is often used to "post notices", such as the official website of the college, or various web pages that show the remaining status of the exam seats. These website updates will have the corresponding detail change, grasp the basic principles of these details change, it is equivalent to master the key to judge the site update.

A. Ways to determine if your website is being updated.

The process of updating the website is varied, and the means of judgment are varied. No matter from which point of view to determine whether the website is updated, the current detection software basically uses one of the following methods of judgment, namely, HTTP message status code, "Last-Modification" file modification time, and MD5 digital signature.

1. Judging by HTTP message status code

After you open the browser developer option, the detailed process of changing the client's access to the web page is displayed. In addition to other content transmitted by the page (such as pictures, videos, etc.), the response and request message of the visit process can often be seen at the top. If there is no update to the page on the second visit, the status code here will appear as "304". The status code, which is generally "200" when visiting a Web page first or normally, is the most commonly occurred status code, indicating that communication is normal.

Many browsers now use this method to determine whether a website is updated because it is intuitive and convenient enough, but it is actually a lot more cumbersome to implement with Python. The browser is simply the original display message data, and the separate implementation of judgment updates requires the use of analytical messages. The first time you access a web page, use Python network programming to get the HTML format file for that page and save it locally. The HTML file is then modified for the if-modification-After request to observe and determine whether the status code returned by the web server is "304". If so, there is no update and you can proceed to the next step.

2. Judging the timing of modifications based on the "Last-Modification" file

For most static pages, you can tell if the site is updated by comparing the time of modification of the "Last-Modification" file. This approach is simple, and no matter what method is used to send a website visit request, the server often returns "Last-Modified", an important tool for recording the last time a website is updated. Then determine if the site has an update to compare "Last-Modified" with interval polling.

Corresponding approach also has its own drawbacks, because most of the dynamic web updates are real-time, can not use this method to judge. At this point, "Last-Modified" only indicates the time when the site server sent back a response message, and is no longer related to the "site update" to follow.

3. Judging by MD5 digital signature

MD5 digital signature can be understood as the site's unique "digital fingerprint", using its principle of judging the website update is more rigid. It records all the data returned by the web site server as a digital signature and stores it in the cache. If you follow this method every time you visit, you can tell if your site is updated by comparing the fingerprints before and after there is a difference.

So far, these methods have highlighted a "comparative" detection method, MD5 digital signature is no exception. Similarly, this approach applies only to static sites, not to dynamic web pages.

4. Judging by whether cookies have changed

Cookies are files stored locally that are assigned to clients by the server to mark identity. Its main role is to further facilitate the server to identify users during subsequent visits, as well as to personalize the transmitted pages according to the characteristics of the user. One of the most obvious examples is when users browse items on a shopping site such as Taobao or Amazon, the site records the user's footprint, making it easier to display content that is more user-friendly when they login.

This means that when a user visits the website twice, the request is carried with the last remaining cookie. The website server identifies identity and personal information based on cookies in a client's request, and determines whether the cookie is updated by the client according to its needs. If a change in the reassigned cookie sits, it means that the website has been updated for whatever reason, including the update of the website itself or the user's personalized content.

Thus, judging whether cookies are updated or not is a high degree of reliability of the choice, but also the fourth chapter of the system to implement the method of judgment. In practice, you can choose to use Python network programming instead of browser to complete the process of sending a message carrying a cookie request, and detect cookie changes to determine the website update.

B. The reason why choose the detection of cookies for website updates

First of all, cookie judgment is different from the principle of most update detection software or plug-ins on the market today, more complex and more accurate. Secondly, in the process of implementing the update by cookie judgment, we need to know more about the KNOWLEDGE including HTTP protocol, cookie mechanism, socket link, Redis database, etc., where http protocol and cookie principle is also the necessary basis for network crawlers. Moreover, the maintenance method of implementing a large number of cookie pools is not unique, there is still a great possibility of exploration. The forms and methods of network programming are complicated, but their essence remains the same. Thus, the study detects changes in cookies to determine whether the website is updated has a strong practicality and universality.

II. EXPLORE HTTP PROTOCOL

Different types of websites, different protocols used, and their corresponding appropriate methods of judgment vary widely. For example, RTMP and HLS protocols are commonly used for live webcasts, and HTTP protocols are common on a wide variety of social news sites. There is also the RTP and RTCP protocols based on the UDP protocol, which are generally suitable for non-repeatable multimedia data streams. In view of the types of websites that need to be tested and the actual needs of life, this paper selects a broader http protocol for research. HTTP protocol is extremely important because it now covers almost every aspect of the Internet, whether it is to study whether the site is updated or a network crawler, it needs to have a deep understanding.

A. Introduction to HTTP protocol

The vast majority of ways to access web content in everyday life are related to HTTP protocols, because most of the current Web development is built on them. From people entering the URL link to browse the website on the browser and confirming it, to the full page presented in front of us in the short seconds, is the HTTP protocol supports several communications

between the various clients and the website server.

This is actually a lot of things that happen in just a few seconds, and after the client is able to confirm the URL link, the browser (i.e. the client) needs to know who it wants next to get the page the user needs, that is, the IPof the corresponding server for the website Address. First, the browser will look up from its own DNS cache if there has been a corresponding record of the domain name and an IP address, and some will access it directly, no cache is found, or the cache is invalidated. Look in the native operating system and, if you still don't find it, start looking from the hard drive host file. If the method you're looking for doesn't work, the browser will initiate a DNS resolution request directly, so that you can get the IP address you want. As soon as it is available, the browser establishes a TCP link to the appropriate server, which is often referred to as a "three-way handshake". After a successful connection is established, the client send HTTP request message containing information that the client needs to let the server know, the main content can be described as who the source of the request is and what the request wants to do. When the server is available, it returns the response message and the required HTML upon request and marks the client (i.e. the cookie involved below). Once the browser is taken, it is processed accordingly to present the original appearance of the page to the user.

B. Implementation of a simple Web server

Because people are usually very accustomed to the operation of the client, after understanding the HTTP protocol, then look at how the web site's server works, which requires writing a simple Web server yourself.

From the server's point of view, your task is to wait for someone else to connect, to resolve the request when it receives it, and to generate the required content in the format required for the request and send it back. What the user needs to do is create a working directory locally, complete the resolution and return of the requested content, and handle various anomalies. This part of the content chose to be implemented in the Linux environment through Python, and using httpie to simulate the daily browser to send requests to test the results of the experiment, all aspects of the requirements to be minimal, to make the experimental content more pure.

1. simple page content

The essence of the server is actually very simple, creating a file in py format under a custom named folder and editing it, which becomes the prototype of the server. The various libraries in Python are extremely powerful, where baseHTTPServer of the standard library can handle client requests. Under the current HTTP1.0 protocol, there are only three request methods: GET, POST, and HEAD. Excluding THE POST used to send form data and HEAD, which only receives back-to-head information, the GET method is most commonly used daily. Introducing BaseHTTPServer and using the RequestHandler class, you can use the do-GET to process the SENT GET request, specifically by self.send-response to return the "200" status code, and self.send. the header to define the type of content returned. It's important to note that the input is "text/html", which means to tell the client how to handle the returned content (html); serverAddress is written as 8080.

At this point, run this already written py format file, and enter the appropriate address on the browser to view, you can see their own simple Web server, as shown in Figure 1, where 127.0.0.1 is the native return address.

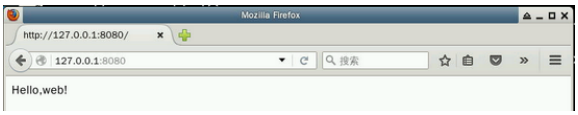

Figure 1 The prototype of a simple Web server

2. Updates to the contents of the server page

The prototype of the server is ready, and the next job is to gradually enrich its functionality. Obviously, just one line of content is absolutely not enough, and for convenience, the page still needs to show the details of the accepted request. The essence of this step is to expand the content of the previous do-GET, before the page's main structure is written out. The page structure code is as follows:

```
Page = '''\
```

```
<html>
<body>
<table>
<tr> <td>Header</td> <td>Value</td> </tr>
<tr> <td>Date and time</td> <td>{date_time}</td>
</tr>
<tr> <td>Client host</td> <td>{client_host}</td>
</tr>
<tr> <td>Client port</td> <td>{client_port}</td> </tr>
<tr> <td>Command</td> <td>{command}</td> </tr>
<tr> <td>Path</td> <td>{path}</td> </tr>
</table>
</body>
</html>
 '''
```

The new addition of the create-page and send-content content allows it to process requests accordingly, and the results of this step are shown as shown in the Figure 2.

Figure 2 Web server runs

So far, web servers can be described as largely complete, but there are still many features to be perfected. For example, the current access to the server returns the correct content when the response status code is 200, if there is a problem somewhere in which the normal access, you need to add a function that returns the 404 status code incorrectly.

3. Jump ingress on the wrong return page

The return of the 404 status code requires the processing of static pages and the addition of an exception handling class to the primary server file (for which a new sys, os library is introduced). Whenever the condition sits, the HTML page is retrieved from the local, with the local HTML page code as follows.

```
<html>
<head>
<title>Plain Page</title>
</head>
<body>
<h1>Plain Page</h1>
<p>Nothin' but HTML.</p>
</body>
</html>
```

The ServerException class is then added to the primary server file and the contents of the previous section are re-changed. It's worth noting that when you write the error page, the quotation marks are single quotes, and you can't forget to update the send-content and main in the main server file, otherwise it will cause the last run to go wrong. Access to the wrong file is pictured Figure 3.

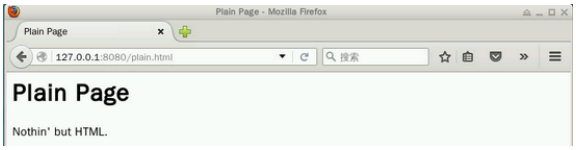

Figure 3 The error

But from the terminal to open the page is still returned is 200 status code, after the entire program check, found that the server file in the “handle error” and “send content” content to be modified, that is, add the corresponding status code, and write a "Not Found." Normal and wrong content is handled, and finally you need to write each situation into a conditional judgment class, add cases, so that you can more easily deal with a variety of problems.

At this point, a simple Web server is written with Python, although the open page is very simple, but the important functions are complete enough to help understand how the server works in the HTTP protocol.

C. Meaning of each field of HTTP message

The second half of this article is part of the implementation of the website update detection, its choice of the basis for judging whether the website is updated is to see whether the cookie is updated. This requires the use of code to implement the use of code to send HTTP requests with local cookies, which are contained in the Request head. So understanding http message fields is equally important. And by observing

the response and request, we can deepen the understanding of TCP connection. The most intuitive way to view Response and Request is to use the Chrome browser "Check" column to browse the relevant web pages, where the Network column displays every detail of the client's "communication" on the service side, where you can see the specify content of the response and request message. Next, we'll cover a few key HTTP message fields.

1. User-Agent field in Request head

Referring to User-Agent, it is necessary to say first that the Robots protocol (network crawler exclusion criteria) may exist on the site, which has the effect of telling the site which supports crawlers and which do not support crawling. The way to view the robots protocol for different sites is generally to add "/robots.txt" to the home page domain name, but there are special cases. Take the JD.com robots protocol, as an example, as shown in Figure 4.

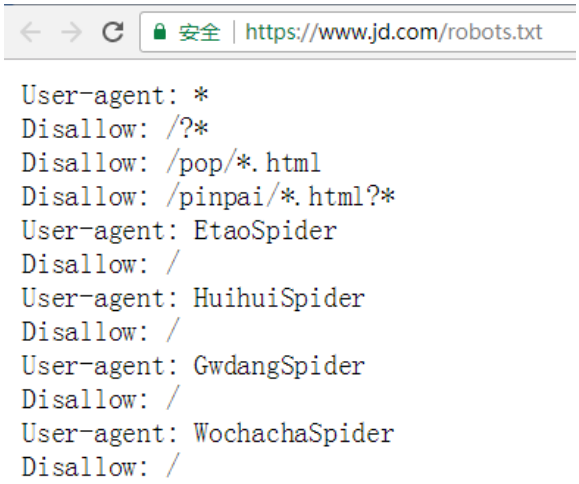

Figure 4 JD.com robots protocol

Understanding the robots protocol can assist in understanding the User-Agent field, which is an "ID card" that identifies the client owner and is used to tell the server who is accessing it. Some defined access is blocked if the User-Agent identity indicated above is present. The goal is to block access to some creepers that some servers don't like or illegal, but if a small crawler written on its own is in this situation, it can be disguised by rewriting the USER-Agent of the HTTP request header.

2. Cookies in Request's Head

If User-Agent functions like an "ID card," cookies are more like "private diaries". Cookies are used to record the user information that the website wishes to record, and when the user then visits the same website, the request head will generally bring the last saved cookie, so that the website can understand the last operation the user performed and arrange the page presented to the user. In a nutshell, cookies are the footprints of users required by the website to be saved, and with cookies, some websites can be used for second or countless account-free password logins, and websites can be customized private pages according to user preferences.

Although a cookie is a mixture of seemingly irregular numbers and letters that are long or short, it can contain unimaginable information. This also raises the question of the legality and confidentiality of the use of cookies. If you do not pay attention to network security, resulting in the disclosure of important websites may suffer unimaginable losses, many websites will place the site's cookie privacy statement on the home page, such as Bing.

There are also many ways to view cookies, such as emptying the address bar after opening the website to enter an instruction statement to query cookies, or using the Chrome "check" column, or using Wireshark to grab the process of visiting the web page, the cookie can also be seen in http messages.

It is important to note that cookies are time-to-life and can be customized, but some are continuous. A detailed description of the specific cookie will be described in detail.

3. Set-Cookie in Response Head

The Set-Cookie field is located at the response head and is literally comprehensible, a cookie sent by the server to the client. It usually appears when the client first or when the cookie needs to be updated, as shown in Figure 5.

Figure 5 First visit to Nankai University International Academic Exchange Office Summer Camp Project Web Page.

You can see that there are no cookies in the request because it is the first time you visit the page. Set-cookies appear at this point. And the second time the page is opened, set-cookies disappear, and cookies appear in the request, which is the first time the server has given itself, as shown in Figure 6.

Figure 6 Second visit to Nankai University International Academic Exchange Branch Summer Camp Project Web Page

III. EXPLORE THE COOKIE MECHANISM

Cookies have been briefly described above, and this section is an experiment to observe how cookies are generated and saved by writing their own simplest account password login page, as well as exploring the principle of login to an account with a cookie without a password. The experiment was compiled in a Windows environment using pycharm in python, a process that, although small in amount of code, was very representative and easy to understand.

A. Initial construction of the web page

This section is very much like the Web server in Chapter 2, the difference is that it now needs to be implemented in a Windows environment, and not too many tedious steps.

Before you begin, you need to determine what framework to use for the whole process, and here's a clean Tornado. It processes faster and actually works significantly better than most other Python frameworks. Using the process is the part of the "from tornado import" that follows your needs immediately.

The entire program consists of three small parts, namely, the server, routing, and landing processing module. First write the login processing module, which defines a class called "Login Handler" that is essentially an edit for page returns. In the initial build phase of the web page, write only one "login" to test whether the build is successful

Next is the routing section. The function of routing is to recognize the URL entered by the client that matches itself, thereby handing over the client's request to the login processing module. Here you define your URL, followed by "/login". If the URL matches, turn back to login Handler login processing module.

Finally, the server itself is here. The server's code framework is more fixed, using the ioloop in tornado, the main function is to listen and send back the routing section, i.e. application

At this point, the entire web frame has been initially set up. Write the URL "127.0.0.1:8080/login" on Google Chrome to see the page that has been written. As shown in Figure 3.1, the word "login" appears on the web page.

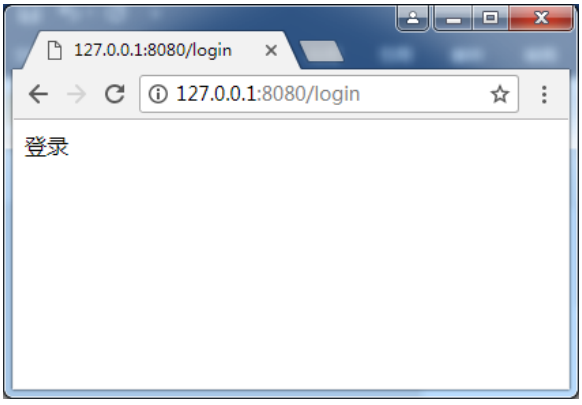

Figure 7 Web form

B. Implement user password login

The rest of the work after the basic framework is written is much easier. Under the premise that the frame is basically left still, the whole process is not

required to display the word "login", but the need to enter a user name and password place, here need to log in processing module "self.write" inside the content swass modified. and need to write a new HTML file, in the render parentheses to introduce the HTML.

Now completed as long as you follow the established steps to log in to the URL, will return the written HTML. However, the function is still incomplete, but also need to write in the main program the content of the specified user name and password, in order to achieve normal login.

When it comes to entering data into a web page, including user names and passwords, it is implemented using post. So the content of the newly added login processing module is about the same as the existing framework, except that the get is replaced with a post.

The final step is to set up the specified username and password content. It has already been shown above, followed by an if judgment statement to determine whether the username and password are correct. If correct, write a cookie and return "Login Success". Once the program is running, it can be viewed in Chrome and filled in with a set username and password, as shown in Figure 8.

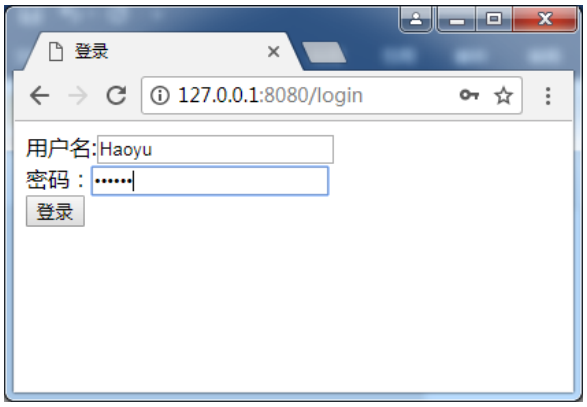

Figure 8 Login page implementation

Click to sign in and you'll see the "Successful login" screen.

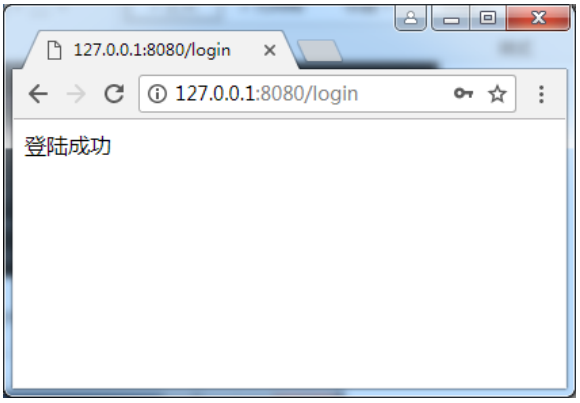

Figure 9 Success

By the end of this section, a new page has been implemented to log in with a username and password. The next section begins with a study of the cookie mechanism.

C. The process by which cookies are generated

It's still the process of opening the web page above and entering your username and password. But this time open Chrome's check to see what the client has done in these operations. At this point, The Set-cookie appears in the Response message, and then enter "document.cookie" from the console in the check to view the cookies on the website, as shown in Figure 10.

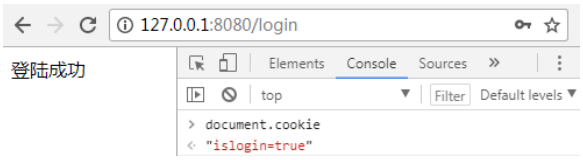

Figure 10 View cookie

You can see from the figure above that the website cookie is set in the program before, indicating that the whole process has been successful so far.

The following to deal with is to give up the login success only after the page back to "login success" four words, is to change the imaging jump "login.html" like jump a new html page. To have something to jump, you must first have an HTML, named "index.html". Then rewrite the jump statement in the main program and change it to "self.redirect()". Now that you have one more page, you need to add an extra section on index in the routing section, in the

same format as login.

There is an additional box of "IndexHandler" here but the class has not been defined before the program. So need to redefine one, visible to pull a full body. At this point, run the main program, log in normally, you can jump into a new page, its URL end "/index".

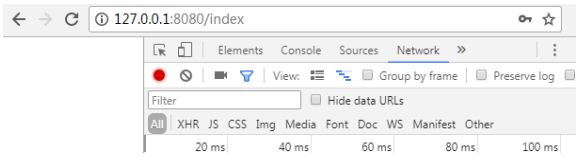

Figure 11 Jump success

And there is also a "true" under the pycharm, as shown in Figure 12, indicating that the operation was successful.

```
C:\Users\Administrator\PycharmProjects\(
true
true
```

Figure 12 Pycharm shows run successfully

To this functionality is basically complete, but you still need to write an if judgment statement to return index.html. This step is done by replacing the original "print"" with an if statement.

Run again, you'll see what it looks like to return to a new page after you've successfully logged in. Check its cookies in the checktool and everything is fine, as shown in Figure 13.

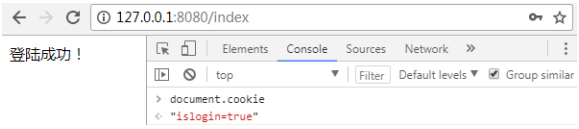

Figure 13 View cookies again

The last case is the result of a normal visit, and the cookie already exists and has been saved. In addition, an unexpected situation in which a cookie is not legal needs to be dealt with. The specific approach is to add an else directly under the if statement, if not legal, return the original "127.0.0.1:8080/login" page, waiting for re-entry until the correct username and password are typed, the code operation is as follows.

```
if islogin == 'true' :
        self.render('index.html')
    else :
        self.redirect('/login')
```

D. Use cookies to make websites login-free

The browser can achieve a secret-free login for re-visiting after the browser saves the cookies left by the user's visit to the website. This practice is a double-edged sword, in the convenience of the user's own actual use of the experience at the same time, but also to give others to take advantage of the opportunity.

First clear all cookies in Chrome, to ensure that the first login is no cookies, at this time try to open the "/index" suffix page is not open, will jump directly to the "/login" page with the login box. After ensuring that all interference is eliminated, log in again to produce cookies, as shown in Figure 14.

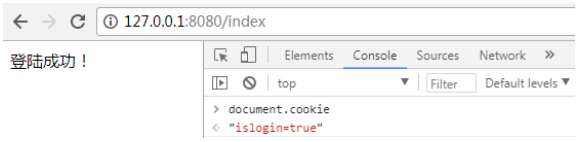

Figure 14 First login after cookie swashes cleared

Now copy the displayed cookie value and turn on Chrome's settings option again to clear all cookies, confirming that direct access to "/index" will be jumped. This time opens Chrome's check, writes the previously copied cookie value, and still uses "document.cookie", the procedure is figure 15.

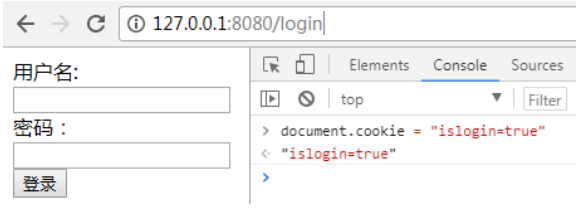

Figure 15 Directly write to previous cookies

At this point, go directly to the "/index" page, because there is the correct cookie value, so will no longer jump to the login window, but directly display the "login success" page, as shown in Figure 16.

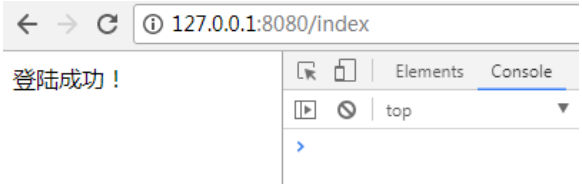

Figure 16 Successful lying in direct lying in without

input

As can be seen from this, carrying cookies to access web pages makes it easy for servers to identify themselves. But this convenience has two sides, from the point of view of the network crawler, it can more effectively hide their true identity, improve the efficiency of the crawler, but on the other hand, if you do not pay attention to the management of native cookies lead to the disclosure of important website cookies, will lead to unintended consequences.

E. How long cookies exist

Cookies are life-long and can be defined by the user themselves. Before defining, you need to introduce a time library, which is to write "import time" before the main program. Setting the cookie for a period of time only requires the addition of "expires" to the set-cookie statement.

For example, if you want the cookie to exist for only 10 seconds. Now let's verify that this cookie really exists for only 10 seconds. Open Chrome as usual, log in and view the cookies, the result is Figure 17.

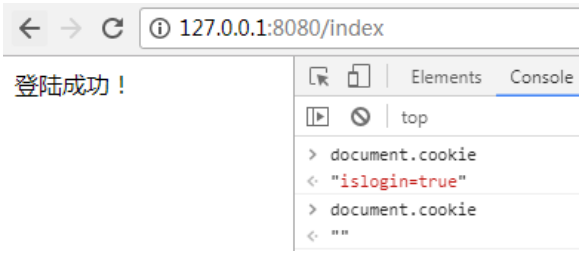

Figure 17 Verify cookie survival time

It can be seen that there is a view of the cookie value immediately after the successful login. But after 10 seconds, viewing the cookie again no longer exists, which means that the program has successfully set the life of the cookie to 10 seconds.

IV. IMPLEMENTATION OF THE SPECIFIC FUNCTIONS OF THE WEBSITE UPDATE DETECTION SYSTEM

Understanding http protocol and cookie mechanism, already has the basic ability to implement the system. Website update detection system, as the name implies its core role is to determine whether the site is updated. But only with this core function is not enough, the realization of the system also needs to be in line with the reality of life, to ensure a strong practicality. Therefore, all the specific functions of the system to achieve in order to maintain the cookie pool and update information push, the following are implemented separately.

A. Maintain the implementation of cookie pools

This paper uses the method of detecting cookie changes to determine website updates, which means that detecting cookie changes is the key.

1. Reasons to use the "cookie pool"

Cookie pools, as the name suggests, are where many cookies are centrally managed. Because the implementation of a complete system can not rely on each program writing alone, so will want to track the update of the site's cookies together unified management, which is required to detect which is the most convenient and convenient way. So this place that centralizes the centralized unified management of cookies is called the "cookie" pool. It can realize the classification management of all website cookies, detect changes in cookies and update them in a timely manner, is the core function of this system.

In addition, cookie pools are widely used. For example, when crawling sina Weibo, because the setting of microblogging is required to log in to use all the functions, users need to build a cookie pool, the cookies in the request header before accessing the micro-blog. Not only that, the cookie pool also solves the problem of many websites prohibiting repeated crawl requests, because the cookie pool can maintain a large number of accounts, with a considerable number of legitimate requests to meet the rules.

2. Principles for implementing "cookie pools"

The Redis database was selected for the implementation of the "cookie pool". It is very versatile, not only to build cookie pools, but also to build IP pools. Its essence can be understood as a database, and there are many numeric types that

support storage, which perfectly meet the requirements of the cookie pool. As shown in Figure 18, many account cookies and passwords for Sina Weibo managed by The Redis client are the same as other websites.

Figure 18 Redis client at a glance

According to the requirements of the system function, the "cookie" pool to be built has the following three requirements: automatic verification of updates, detection filtering at any time, API interface provided. It's easy to implement cookie changes that track a single website, but maintaining changes in the entire cookie pool is not easy, and the entire process is a huge amount of code, and the key steps will be described only next.

The implementation body of the whole program is divided into two parts, one is the program's total switch, the other is the branch of the program, the program is also implemented using pycharm. First, create a frame py file on the outermost of the entire program file to act as an entry point for the program.

The "run.py" file is the entry to the program, and the Scheduler scheduler is used, and the real scheduling is implemented in its "run" method. In addition to this, config files have been created as switches for the main function of the program. The switch not only contains the cookie generator, the authenticator, the API service switch, but also defines the API address and cycle cycle of the cookie. If the three modules in this want to implement one or more of the functions, you only need to change the corresponding equal sign after the "False" to "True", and do not need the function written as "False". These three modules support the entire cookie pool and are the core of the entire detection system.

3. Scheduler

Scheduler uses the Scheduler class, and defines three methods in the program, which correspond to the three modules of the total program, namely the generator, the validator, the API. Whenever a program receives a corresponding instruction, it starts with the "run" method and calls the corresponding module. There are three judgment statements in the run method that can help you jump to the appropriate module. This part is the scheduler's "branch small switch" that corresponds to the three branch modules below.

Where the implementation of the validator needs to rely on a dead cycle, but also need to define the interval between each cycle, i.e. timed detection, timed sleep, cycle interval can be achieved by "time.sleep". There is also a new class in this loop, defined as "tester", which is still implemented and jumped using the "run" method. This section can help jump to the new "tester.py". The rest of the generator and API are structurally identical, using the same method for jumping. At this point, the entire scheduler is implemented.

4. Call the Redis database

The Redis database has a client, and in order to make contact with the program it is writing, it is necessary to write an extra file to make the call. This call file needs to define a "RedisClient". It needs to be inherited by two "cookiesRedisClient" and "AccountRedisClient" classes, because many of the operational steps of RedisClient are common at initialization time, and these generic operations need to be placed in the subclasses to play their role accordingly.

The "cookiesRedisClient" contains "domain" and "name" that correspond to the corresponding key value, i.e. the cookie value, in the Redis client.

The corresponding account password corresponds to "domain" and "name" in AccountRedisClient, and the code is as similar as the above. Sina Weibo was used in the first operation, but "name" can vary depending on the name of the website you want to track. There are also many small features that can be defined in the three classes, all of which can be implemented using "def". However, there are three things that must be implemented in subclasses: set, get, and delete. Using the "set" example in

"CookiesRedisClient", it calls the action on set in the Redis API. If the process goes wrong, it throws an exception and needs to be handled.

There are many kinds of exceptions that may occur in this section, and one more file needs to be written for classification of the exception. Not only cookies, the same account part will also appear abnormal, so write an additional "error.py" for jumping

5.Tester

The validator tester is connected to the scheduler Scheduler's request jump, and two classes are mainly defined in the implementation process. Where "ValidTester" is a parent class, defining a lot of initialization operations, that is, the relevant generic methods. This setting also throws an exception, which means that there must be a corresponding exception workaround in the subclass, otherwise an error will occur at the time of the call. And the "run" of this parent class is the core, it is responsible for redis the corresponding all the cookies are taken out one after another using the for loop to get the test method for detection.

"WeiboValidTester" as a subclass is responsible for implementing some of the relevant unique methods. Different site validation methods are different, and these tests need to be overridden in subclasses. This part of the verification method is more general, it uses these cookies in the request header to see if they can return normal results.

Next is the lower half of the subclass, which continues only if the return code is "200" and the connection is proven to be normal. It can be seen that if the cookie has failed it will be deleted. The above is the implementation of the validator tester.

6.Generator

The essence of the generator is to put the many accounts in Redis, take out one-by-one logins, and put the cookies obtained after successful login back in the Redis queue. As with the validator, there are two classes in the generator. Where the parent class is similar to the validator's parent class, performs a series of initialization operations, as well as a series of variable settings, including setting "userAgent" to prevent access from being restricted. It also defines the part that produces the new cookie, which is inherited by the subclass.

Next is the composition of the subclass section. Subclasses also require some initialization, such as their name and implementation of the Cloud Code platform section, which has borrowed an identification verification code. The main job for subclasses is to ensure that new cookies are obtained after successful login.

Part of it is the implementation of the "new-cookies" in the parent class, which the child class needs to write in detail. Initial lying it removes all cookies currently stored, re-requests and requires a detailed implementation to borrow third-party websites to identify possible QR codes.

The above is the implementation of the generator idea.

7. API Interface Module

The API interface provides a domain name link to view cookies for a variety of purposes. The API module here calls Flask and implements several more general functions, such as the "random" function, which can be randomly obtained for cookies.

At this point, you need to close the authenticator and generator, and only open the API module. Enter "localhost:5000/weibo/random" in your browser to see the randomly given cookie values in the cookie pool and can change as refreshes.

At this point, the required API modules are completed, and new requirements can be extended at any time.

B. Implementation of updating information push capabilities

Having studied so much, it takes the last step to make a complete system, that is, let users know the results of the website's timely update. After all, users will not always sit in front of the computer staring at the program observation, so do not delay doing other things without the premise of receiving the site updated instant message. The specific way to implement this is to follow the maintenance changes

in the cookie pool and place the latest website HTML files in the mailbox of the message's attachments.

1. Download the HTML file for the web page

After the core function has been implemented, all to do is the message push, the system abandoned the direct use of reptile creep method, but chose to directly save the transferred HTML file. There are many benefits to this, because if you use a crawler it's easy to completely void the program because users have different concerns about the updated site, and even the dilemma of wanting to see a new website is going to write a new reptile. Doing so will make the entire website update detection system meaningless, users have to write a crawler time, why not open the web page to see it yourself?

On the contrary, saving the HTML file received back by the client directly as push content has a great advantage. Compared to the reptile it is very simple, there is no complex and cumbersome analysis steps, there is no site robots protocol constraints. Putting saved HTML files in attachments is easily and quickly easily sent to users by mail, allowing them to find information they want to know.

The process of getting the HTML of a website is simple, and you need to manually fill in the URL you want to capture the site, or jump from the cookie pool. Then pay attention to transcode the file can be, note that writing the file is used with bytes instead of str.

2. Send mail automatically

Before this part of the function can be implemented, the SMTP service for the mailbox that will be used for the shipment is opened so that the program can call the corresponding library.

Downloading HTML files and sending messages are actually one-size-a-like, all of which belong to message push. To implement sending a message requires calling a "smtplib" library, which is specifically used to implement sending mail. Second, set up information in front of the program, such as sending and receiving mailboxes.

The program first needs to set up the details of the message, including the subject, content, and so on. The body can be edited on its own or you can use the contents of an HTML file directly. If you use HTML pages, you can actually be more flexible, you can attach your own picture address and format adjustments, and so on.

Messages can also choose to add txt format files or add pictures, they can be placed in attachments, the implementation code is as follows. You need to add the attachment name and fill in the local address of the attachment exactly.

The last part is the scaffolding code of the mailing program, which needs to be filled in in any program that has the automatic e-mail function. It uses the initial SMTP protocol, invokes the sender and receiver information that has been filled in above, and so on. Also need to note that the SMTP port number of this place varies depending on the sender's email address, the program uses 163 mailboxes, so the port number is 25.

At this point, the push step for updating the message is complete.

V. PRACTICAL TESTING OF SYSTEM FUNCTIONALITY

The email program will be modified or added a portion of the content to enable it to add HTML format files. Using if statements and jump commands to combine cookie pools and information push components, it constitutes a complete website update detection system. In summary, the whole system contains three switches, namely the generator, the validator, the API interface module, and then verify the integrity of its functions, other websites the same.

A. Turn on the authenticator and API interface module

By writing the generator in the switch as "False" and the other two as "True", you can verify that all cookies in the Redis database are still valid. In layman's terms, this step is the total system verification switch, because the system contains timing function, open the authenticator will be automatically detected. By directly representing all

cookies in the Redis database using an API interface, you can detect the cookies directly by accessing the interface. The test results are displayed in the "Terminal" terminal of pycharm, taking Sina Weibo as an example, the detection process and results are as shown in Figure 19.

```
CookiesPool git:(master) ✗ python3 run.py
Checking Cookies
Testing Account vwfmbke6597007@163.com
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Valid Cookies vwfmbke6597007@163.com
Testing Account uqhhdret80@163.com
Valid Cookies uqhhdret80@163.com
Testing Account 15273807294
Valid Cookies 15273807294
Testing Account 15873072166
Valid Cookies 15873072166
```

Figure 19 Validator and API module validation results

You can see that the website cookies corresponding to these accounts are still valid, indicating that these two parts are complete.

B. Turn on generator only

The function of the generator is to generate new cookies, whether used for first login or for existing cookies that need to be updated. The on command is to write the generator "generator" in the switch to "True" and the other two to "False". The process involves the corresponding account password, corresponding to the cookie placed in the request header to access, as well as the verification code recognition, the same take Sina Weibo as an example, the process is as shown in Figure 20.

```
CookiesPool git:(master) ✗ python3 run.py
Generating Cookies
Getting 527 accounts from Redis
Getting Cookies of  weibo tdbhddyn9@163.com 7sg0e2xznb
Generating Cookies of tdbhddyn9@163.com
出现验证码，开始识别验证码
Retrying:  1400247052 Count:  1
{'method': 'result', 'cid': 1400247052}
{'ret': -3002, 'cid': 1400247052, 'text': ''}
云打码验证码还在识别
Retrying:  1400247052 Count:  2
{'method': 'result', 'cid': 1400247052}
{'ret': -3002, 'cid': 1400247052, 'text': ''}
云打码验证码还在识别
Retrying:  1400247052 Count:  3
```

Figure 20 Start identifying the QR code

After several times to identify the verification code, the identification was successful, and the content of the two-dimensional code was displayed, the result is Figure 21.

```
Retrying:  1400247052 Count:  9
{'method': 'result', 'cid': 1400247052}
{'ret': -3002, 'cid': 1400247052, 'text': ''}
云打码验证码还在识别
Retrying:  1400247052 Count:  10
{'method': 'result', 'cid': 1400247052}
{'ret': 0, 'cid': 1400247052, 'text': 'W545S'}
登录成功
```

Figure 21 Identify success and log in

Once you're successful, you'll get a new cookie and put it automatically into the Redis database. The whole process does not need to be manually involved, to obtain the results as shown in Figure 22.

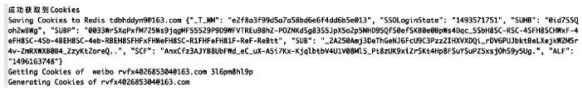

Figure 22 Successful access to cookie

You can see that the generator of the cookie pool is fully functional and you can proceed to the next step.

C. Update push process validation

As soon as the cookie is detected to have been updated, the HTML acquisition program is launched and saved to a unified path. The process is simpler, and pycharm displays a success page as shown in Figure 23.

```
C:\Users\Administrator\PycharmProjects\练习下载豆瓣照片\venv\Scripts\python.exe C
下载成功

Process finished with exit code 0
```

Figure 23 Get HTML page success

For example, the website of the Faculty of Education of Nankai University, the page opened as shown in Figure 24.

Figure 24 Page details

Next is the verification of the automatic message. The first is the automatic sending verification without attachments, eliminating the tedious steps of adding attachments, and the implementation process is fast. The process uses two private mailboxes of individuals to communicate, the actual application can be arbitrarily changed the recipient, the result is Figure 25.

Figure 25 Successfully received an attachment-free message

The process of automatically sending mail with attachments is more complex, and the results are shown in Figure 26.

Figure 26 Successfully received a message with an attachment

At this point, all functional tests of the website update detection system have been completed and have been fully implemented.

D. Deficiencies

This article integrated website update monitoring system is fully functional, but there are still many places worth upgrading, and in the process of implementation has also appeared a lot of problems, some have been resolved, and some have not been perfectly resolved.

1. HTML notification reliability questionable

The verification of the above cookie pool selected Sina Weibo, because the micro-blogging update is very fast, suitable for the cookie pool research. But the update push verification section chose a page from the school's education website because the HTML page saved by the microblogging page was blank, as shown in Figure 27.

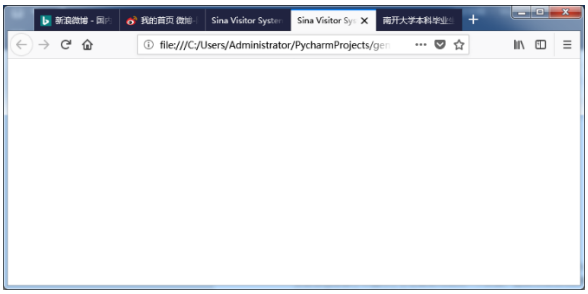

Figure 27 HTML page blank

Such a blank page is certainly not the purpose of letting users know about the update. Not only Sina Weibo, there are many pages saved is garbled. The creep-grabbing key information method was initially abandoned because each user has a different focus, and the code writing method varies widely. HTML notification method, although the code is stable, but the notification effect may be very unstable.

2. Spam issues

There has been a problem with the company's withdrawal of letters when the auto-mail code has just

been written and running. Later changed the theme and body of the might be more popular, but there were still several times when the inbox was classified as spam, as shown in Figure 28.

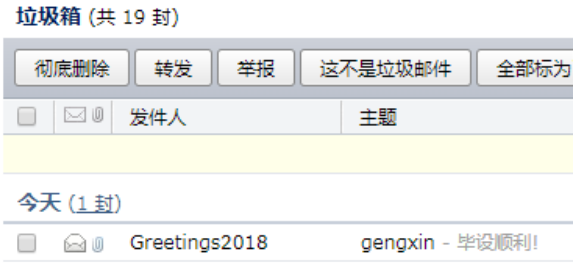

Figure 28 Messages are classified as spam

If an updated message is classified as spam, there is no difference between actually not being notified. Even if many situations are received normally, the risk sits so that it does not achieve the perfect effect.

3. Other questions

The HTTP protocol targeted by the system can actually contain the vast majority of websites, but similar to some live video sites, it is likely that the HTTP protocol will not be used. For these live sites, users may have a need to get their subscription stod anchor son-in-chief messages, which means the system is not "all-powerful."

In addition, the system is still a collection of some py-format files. Although the functionality is already complete, but can not be separated from pycharm. If you want to use the system alone, you also need to design a user interface to be encapsulated as exe.

VI. SUMMARY AND OUTLOOK

At this point, the implementation and analysis of the whole system is basically completed. Review all the learning practice process, both harvest and shortcomings, but also on the relevant prospects of thinking.

A. Summary

Based on http protocol design, this system in-depth study of cookie mechanism and network programming and other content, to achieve the detection and notification of website updates. The whole process is not only very close to the reality of life, but also many of these functional modules alone can also be used perfectly. Learning is not only the process of imitation, but also the process of thinking, although the whole system is fully functional and has a reference role, but put into practice there are still many issues to consider.

Website update is a complex but simple-principle process, the above exploration is only a small part of it, there are still many principles and communication process worthy of more in-depth exploration. Not only that, the Internet protocol itself is constantly being updated, so adapting to the latest protocols and even developing new protocols is the direction of the efforts of countless Web engineers.

B. Outlook

More effort is required to make a detection update system for various types of protocols that does not produce unexpected situations with smooth process. Not only to learn to master the principles of various types of website updates, but also to constantly eliminate all the accidents that may occur in the process. More smooth, you can also transplant the system to the mobile platform, so that the site update more convenient, more free from the "spam problem" trouble.

At present, the rapid development of the network, to really do a good job of website update detection system not only pay close attention to the reality of life, but also at any time to follow up the development of network protocol. Just making this one detection system is not the end, this involves enough knowledge to extend to all areas of the network, diligent thinking and more attempts is the developer's first state and normal.