**Part1 Web Scraping**

* **Anti-scrape mechanisms**

The first mechanism is font anti-crawling which exists in the car details on the guazi web page, it replaces certain data on the page with a custom font, and some of the numbers are randomly generated, so it may get the wrong data content without using the correct decoding method. we tried the mobile webpage of this website and find it uses the second anti-scrape mechanism: parameter encryption, there is a **verify\_token** field in the query string parameters of the header when sending a request, so how did this token come about? We use postman to test and find that **client-time** and **verify-token** in the header have a strong correlation and that these two parameters determine whether the request is successful or not, so we go back to the source code of the website and search for verify-token

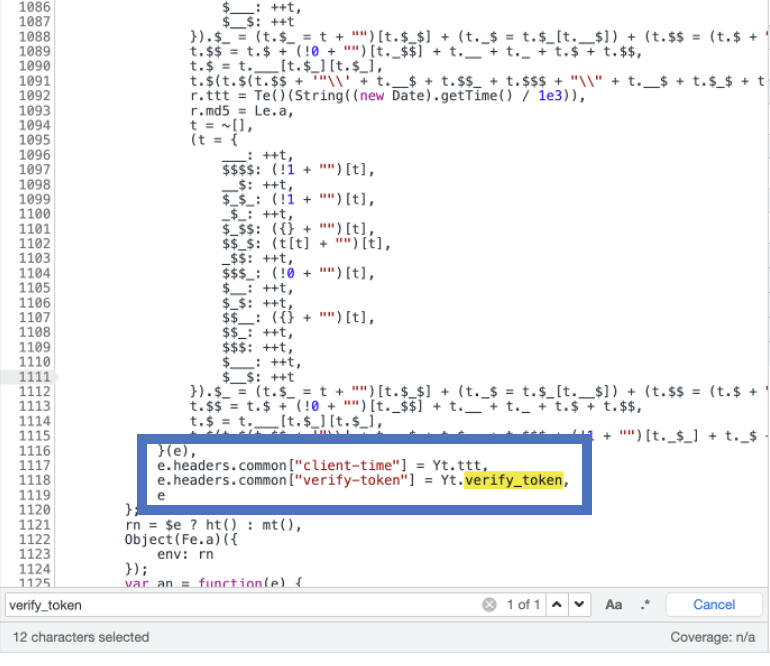
 

Figure 1. source code of the webpage

Then we try to search for the **verify\_token** character in the source code of this page, and we find that the source code of the page is also encrypted javascript code. But we roughly know this process, as shown in figure1,the value of the variable **ttt** is assigned to **client\_time** and the value of the variable **verify\_token** is assigned to **verify\_token**, so we add a breakpoint at the beginning of this process when defining **ttt** and execute all the steps one by one until a function called **verify\_token**.

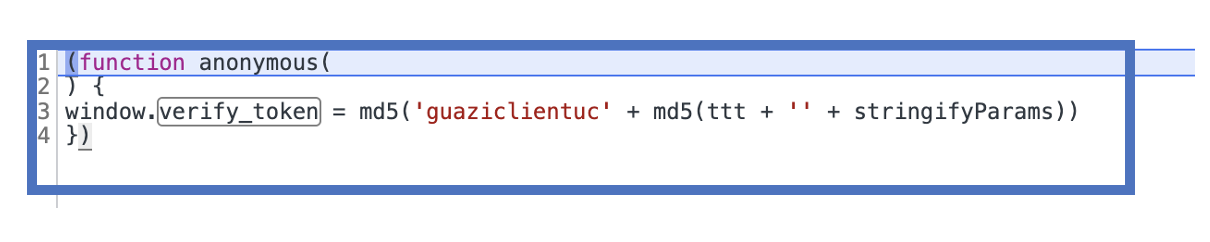


Figure 2. verify\_token function of the source code

Refer to figure 2, **ttt** first adds **stringifyParams** formd5 encryption and then, ‘guaziclientuc’ add the above things formd5 encryption.



Figure 3. View variables in the console

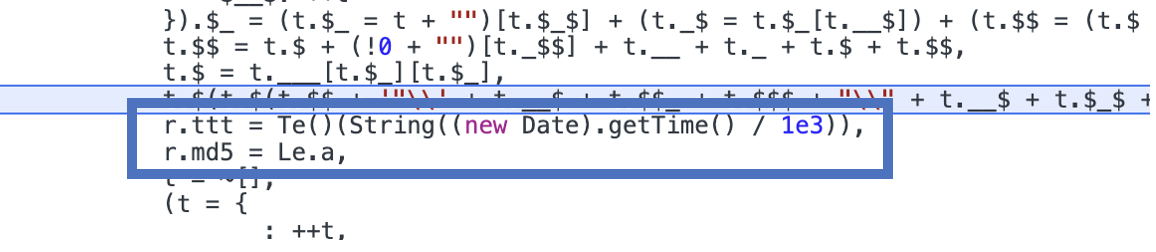


Figure 4. **ttt** is a timestamp in milliseconds

And we can view variables in the console and find that **ttt** is a timestamp in milliseconds and **stringifyParams** is URL query params. Finally, manually find a 32-bit md5 encryption tool and test it with postman to verify.

* **Web Scraping**

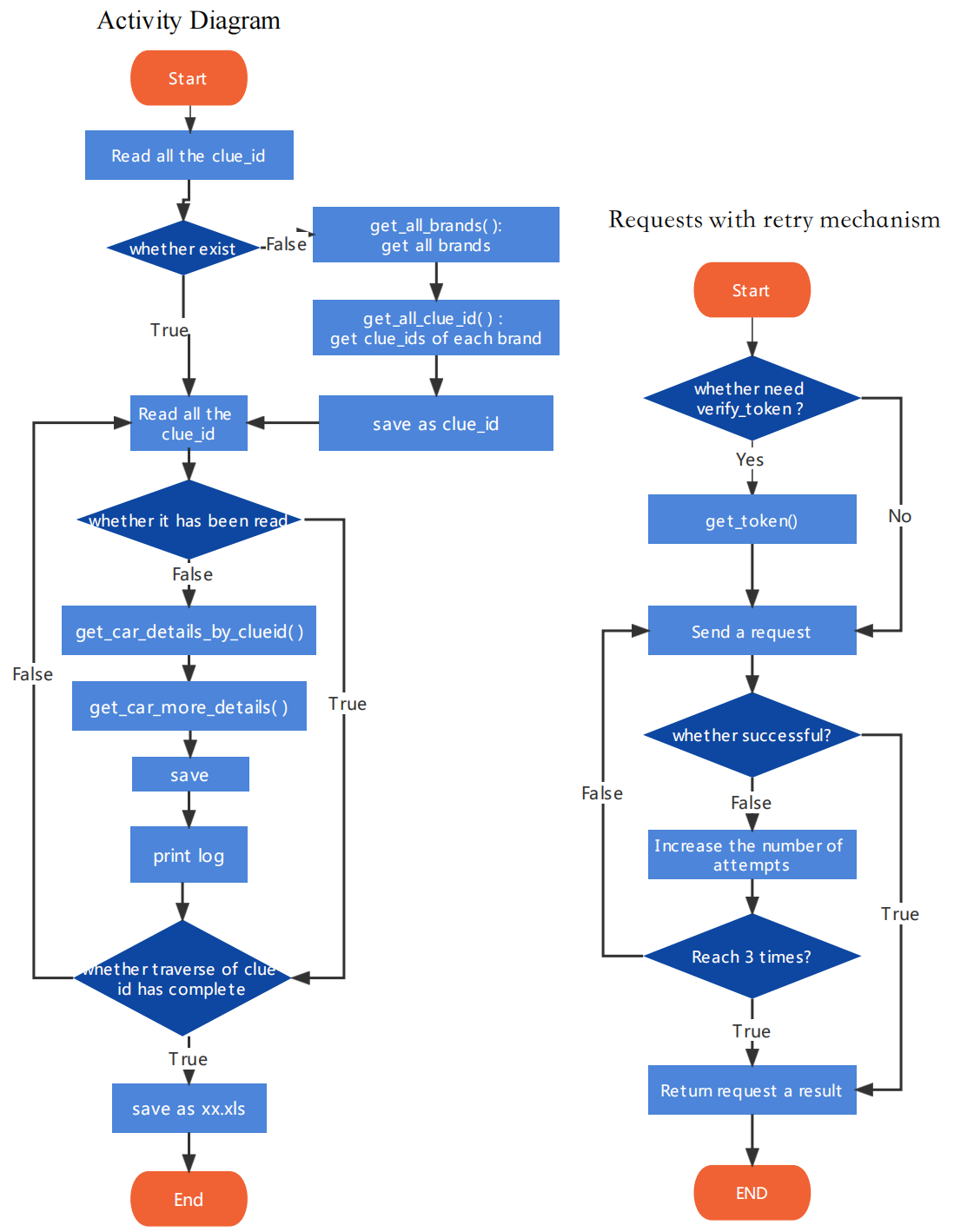
****

Figure 5.Activity Diagram

The above is an activity diagram of the crawler code, containing four requests, two of which require tokens which call the get\_token() function. For each request, a retry mechanism is used due to verification code risk control or IP lock.