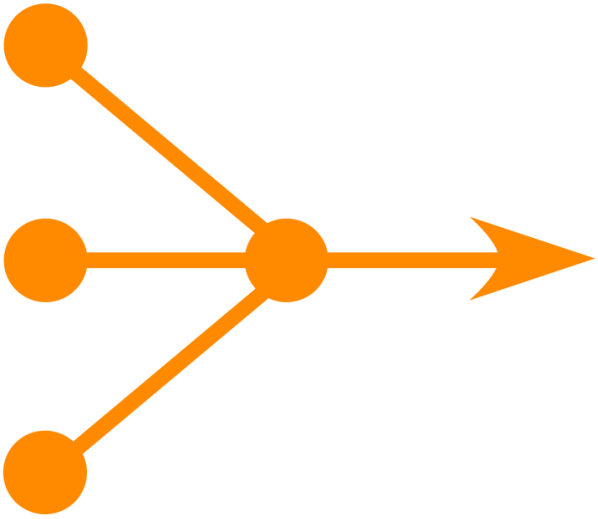
Etherscan labelcloud scraper

Developed by: https://github.com/evilgenius786

Project link: **https://github.com/evilgenius786/Etherscan-Scraping**

Delivered to: **Chainlabs.AI**



Technical details

Language used: Python 3

Libraries and dependencies:

1. Requests
2. Bs4
3. Lxml
4. Selenium
5. Webdriver-manager
6. Fake-useragent

Subscriptions required:

1. Proxies (**32 GB** volume or **314k** requests)
2. 2captcha (required only once for logging in)

Resources (cost and time) analysis:

Total accounts: 297496

Total tokens: 13112

Total listings (token + account): 310608

Total pages (100 listings per page): 3106

Total requests required: **314k**

Price of 100k requests: **$49**

Price of 400k requests: **$200**

but we won’t be using whole requests count so we can reuse the remaining 85k requests for next run.

So, if we follow it smartly, we can run 5 iterations at the cost of 4 iterations.

So old cost of 5 iterations per month would be 200*5 = **$1000**

If we use it smartly, we can bring the cost of 5 iterations to **$800** per month.

And 2captcha usage is only 1 time so we don’t need to count its cost since once its logged in, it'll stay logged in forever unless we delete the "ChromeProfile" folder from there.

Summary: Worst case scenario cost per month will be $200.

Time calculation:

Total accounts: 297496

Total tokens: 13112

Total listings (token + account): **310608**

If time taken per listing is 1 second then it'll take **87 hours** from 0 to 100. But we have multiple threads (50) working at same time so, it'll be faster than that. Also, according to tests, it took around 3 days to scrape all data from 0 to 100 since the data of bigger labels (uniswap, eth2-depositor, contract deployer, token contract, gnosis safe multisig) takes lots of time to load.

Total pages (100 listings per page): **3106**

So, it'll take around **1.5 hour** to go from 0 to 100 scanning all pages (tokens + accounts) of all labels if all addresses are scraped (we run it to verify if everything is scraped).

Notes:

1. To verify if all addresses are scraped, delete the **ScrapedLabels.txt** and re-run the script. It'll take around 1.5-2 hours.
2. To scrape fresh data, all from 0. Move **all** the files to some other folder except the script itself (**etherscan.py**).
3. To skip bigger labels, add them to **blocked.txt**.
4. 2captcha key is saved in **2captcha.txt**.
5. Rotating proxy endpoint is saved in **proxy.txt.**
6. **ChromeProfile** folder contains user profile data so it doesn't need to login every time.
7. **CSVs** folder contains individual CSV files of labels (tokens and accounts).
8. **Labelcloud** contains CSV of scraped table data of label's category (account or token) and subcategory (eg main or others) for caching and resuming from the latest page if script is stopped in between.
9. The number of threads in queue are limited so the script won't crash. So, if threads in queue exceeds the limit, it'll wait for some time, like 10 seconds, before spawning new threads in queue.
10. **AccountsMaster.csv** and **TokensMaster.csv** are the master files that contain all the data.
11. **Error-Account.txt** and **Error-Token.txt** contains address that had any sort of error while scraping. You can test them individually

using test functions (**checkToken** and **checkAccount**) provided in the end of script.

12. **Summary.json** contains the summary, it is updated after each label scraped.
13. **Logs** directory contains all the logs of each run, separated by time stamp. You can delete them if they take too much space.
14. **ScrapedAccount.txt** and **ScrapedToken.txt** contains addresses of scraped tokens and accounts respectively.
15. **ScrapedLabels.txt** contains labels that are scraped.