Underground Scraper

3 Collection Methodology

Now we turn to describing how we collected the data including underground market places as well as social networks like Telegram. First we will briefly introduce the main idea and methods, and our achievement measure, then we will write down some troubles and challenges we met and how we address or bypass it.

3.1 Results Overview

We mainly dive into 3 different marketplaces (AccsMarket, EZKIFY Services, BlackHatWorld) and 1 social media (Telegram), yielding millions of rows (tuples), with a $\sim$100 GB data set size. The scraper runs one to two times every day to keep the track of various data indicators.

The whole scraper project is written in pure Rust benefit from its excellent error handling and recovering, to keep it safely run on the background. We have successively used many techniques including Tors[1], IP pools, Chrome drivers, Headless chrome (puppeteer) during the whole process.

As of Mid-May, the number of tuples (rows) in each marketplace is shown in Table 1.

Marketplace	Tuples
AccsMarket	63786
EZKIFY	164 672
BlackHatWorld	581 103
Telegram	268 678 097

Table 1: # of tuples in each marketplace

3.2 Scraping Mechanism

Since viewing posts and contents in many forums of BlackHatWorld does not require authentication, we can do our scraping sneakingly. To keep our efficiency and not be noticed by Website administrators, we use proxy IP pools with 100 different IP with each one walking at a random 2.5 s $\sim$ 3 s/page rate, and we finally get a quite good rate of 30 $\sim$ 40 pages per second.

Also, BlackHatWorld built a defense of Cloudflare, to resolve it we use Chrome driver and puppeteers to temporary open a Chrome, and manually solve the annoying puzzle to get Cookie, then the Cookie will store to local automatically, then this authentication cookie will keep valid for at least 2 $\sim$ 3 days, which offers a lot.

To coordinate all "workers" that using different IPs, we use a local server technique —— We built a local server to handle the functionality of result uploading, it plays a role as gateway, which means that we can write different kinds of scrapers (in different ways), and all of the data will send to our local server in a simple (and unified) format. These scrapers can written in various languages and we avoid sticking lower into databases. The server also has the functions like "load balancing", different scrapers (workers) first request to it to get the disjoint work, and do them themselves, finally upload to server, thus ensures the efficient use of resources.

We use PostgreSQL as data storage and management system for its various features including JSON storage and a beautifully efficient speed.

3.2.1 Data freshness check and update

For AccsMarket and EZKIFY, the website itself does not keep the history of data, we can scrape it regularly with checking and comparing (post-processing), because one round of whole-scraping does not cost much. For BlackHatWorld, we can force the post order by the "post date" (a.k.a. ID order, which are some) by using `?order=post_date&direction=desc`, to prevent the out-of-order caused by irregular replies.

3.2.2 Snowball Scraping

For Telegram, we use the technique of snowball scraping. Suppose we already have a set of channels in our database,

3.3 Challenges

3.3.1 Verification/Authentication

3.3.2 TLS Fingerprints

3.3.3 Error Handling

References

[1] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, page 21, USA, 2004. USENIX Association.