Python Scrapers for Scraping Cryptomarkets on Tor

Yubao Wu[1(✉)], Fengpan Zhao[1], Xucan Chen[1], Pavel Skums[1], Eric L. Sevigny[2], David Maimon[2], Marie Ouellet[2], Monica Haavisto Swahn[3], Sheryl M. Strasser[3], Mohammad Javad Feizollahi[4], Youfang Zhang[4], and Gunjan Sekhon[4]

[1] Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA
{ywu28,pskums}@gsu.edu, {fzhao6,xchen41}@student.gsu.edu
[2] Department of Criminal Justice and Criminology, Georgia State University, Atlanta, GA 30303, USA
{esevigny,dmaimon,mouellet}@gsu.edu
[3] School of Public Health, Georgia State University, Atlanta, GA 30303, USA
{mswahn,sstrasser}@gsu.edu
[4] Institute for Insight, Georgia State University, Atlanta, GA 30303, USA
mfeizollahi@gsu.edu, {yzhang107,gsekhon1}@student.gsu.edu

Abstract. Cryptomarkets are commercial websites on the web that operate via darknet, a portion of the Internet that limits the ability to trace users' identity. Cryptomarkets have facilitated illicit product trading and transformed the methods used for illicit product transactions. The surveillence and understanding of cryptomarkets is critical for law enforcement and public health. In this paper, we design and implement Python scrapers for scraping cryptomarkets. The design of the scraper system is described with details and the source code of the scrapers is shared with the public.

Keywords: Scraper · Cryptomarket · Tor · Darknet · MySQL

1 Introduction

The Darknet is a layer or portion of the Internet that limits the ability to trace users' identity. It is considered part of the deep web, which is a portion of the Internet that is not indexed by standard web search engines. Accessing the Darknet requires specific software or network configurations, such as Tor ("The Onion Router"), the most popular anonymous network.

Cryptomarkets operate on the Darknet, much like eBay or Craigslist, as commercial websites for selling illicit products, including drugs, weapons, and pornography [1]. The first cryptomarket, Silk Road [2,3], launched in early 2011 and operated until October 2013, when the website was taken down by the Federal Bureau of Investigation (FBI) following the arrest of the site's founder,

© Springer Nature Switzerland AG 2019
G. Wang et al. (Eds.): SpaCCS 2019, LNCS 11611, pp. 244–260, 2019.
https://doi.org/10.1007/978-3-030-24907-6_19

Ross Ulbricht. However, new cryptomarkets have proliferated in the wake of Silk Road's demise [4], presenting an increasingly serious challenge to law enforcement and intelligence efforts to combat cybercrime [5]. We have documented at least 35 active cryptomarkets as of February 2019. Figure 1 shows the homepage of Dream Market, the largest cryptomarket at present. The link address ending with ".onion" indicates that it is a hidden web service in the Tor anonymous network. A hidden service in Tor means the identity (IP address or location) of any web server is hidden. From Fig. 1, we can see that Dream Market offers five categories of products including Digital Goods, Drugs, Drugs Paraphernalia, Services, and Other. Table 1 shows the subcategories and number of corresponding advertisements within each parent category. From Table 1, we can see that the illicit products include hacking tools, malware, stolen credit cards, drugs, and counterfeit products. Table 2 shows the largest seven cryptomarkets at present according to the total number of ads listed in each market. All cryptomarkets offer similar categories of products.

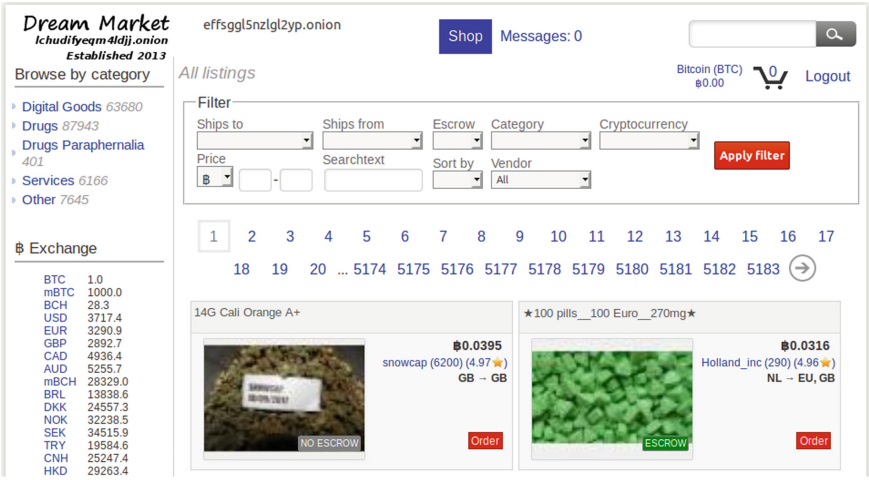


Fig. 1. The homepage of Dream market

The onion routing (Tor) system is the most popular anonymous network for accessing these cryptomarkets. Tor conceals users' activities through a series of relays called "onion routing," as shown in Fig. 2. The decentralized nature of peer-to-peer networks makes it difficult for law enforcement agencies to seize web hosting servers, since servers are potentially distributed across the globe. Payments are made using cryptocurrencies like Bitcoin. Since both cryptomarkets and cryptocurrencies are anonymous, there are minimal risks for vendors selling illicit products on the Darknet.

The surveillance and understanding of cryptomarkets within the context of drug abuse and overdose is critical for both law enforcement and public

Table 1. Categories of products in Dream market

Categories	Sub-categories
Digital Goods 63680	Data 2709, Drugs 587, E-Books 14918, Erotica 2819, Fraud 4726, Fraud Related 11086, Hacking 2654, Information 16206, Other 2051, Security 570, Software 1940
Drugs 87943	Barbiturates 49, Benzos 4031, Cannabis 29179, Dissociatives 3258, Ecstasy 11672, Opioids 5492, Prescription 5559, Psychedelics 6349, RCs 646, Steroids 4090, Stimulants 14296, Weight loss 220
Drugs Paraphernalia 401	Harm Reduction 65
Services 6166	Hacking 689, IDs & Passports 1545, Money 1432, Other 897, Cash out 1012
Other 7645	Counterfeits 4233, Electronics 257, Jewellery 1391, Lab Supplies 109, Miscellaneous 620, Defense 376

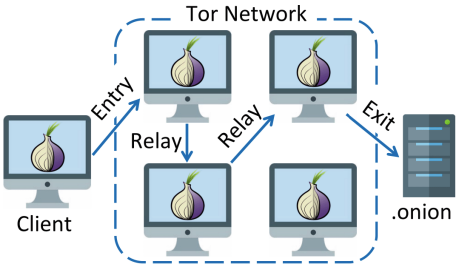


Fig. 2. The onion routing system

Table 2. Cryptomarkets

Cryptomarkets	#Ads
Dream	165,835
Berlusconi	38,270
Wall Street	16,766
Valhalla	11,023
Empire	9,499
Point Tochka	6,358
Silk Road 3.1	5,657

health [3,6–8]. Enhanced surveillance capabilities can gather information, provide actionable intelligence for law enforcement purposes, and identify emerging trends in substance transactions (both licit and illicit) that are contributing to the escalating drug crisis impacting populations on a global scale. The absence of a systematic online drug surveillance capability is the motivational catalyst for this research, which is the development of an online scraping tool to employ within cryptomarkets.

In this paper, we develop scrapers for the seven largest cryptomarkets shown in Table 2. The scraped data are stored in a MySQL database. Details surrounding the computational development and capacity used in the scraper design are described. To the best of our knowledge, this is the first Python package created specifically for scraping multiple cryptomarkets to investigate drug-related transactions. The scraper source code is publicly available upon request. (Send correspondence to scraper.crypto@gmail.com with your name, position, and affiliation. We will send you a link for downloading the source code upon verification).

2 System Overview

Figure 3 shows the system networking framework. Our Python scraper programs run in an Ubuntu operating system (OS). For the convenience of sharing, we use VirtualBox and Ubuntu virtual machine. Since VirtualBox can be installed on any OS, students can easily import our virtual machine and start using the scrapers without the need for further coding or configurations. The university security disallows the Tor connection. Therefore we use Amazon Web Service (AWS) as a proxy for visiting Tor. The scraped data is uploaded into a local database server hosted at the university data center. All data will be uploaded into the database server and no data will be stored in students' local computers. The system is designed to allow multiple students to run the scrapers simultaneously. The scraper will check whether a webpage exists in the database before scraping the webpage in order to avoid scraping duplicate webpages.

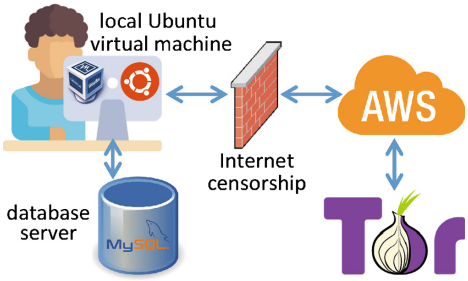


Fig. 3. The system networking framework

The scraping system consists of scraping and parsing stages. In the scraping stage, the scraper program will navigate through different webpages within a cryptomarket. The scraper uses the Selenium package to automate the Firefox browser to navigate through different webpages, download the html files, and upload them into the MySQL database. Most cryptomarkets like Dream Market require users to input CAPTCHAs after users browse a predetermined number of webpages. Cracking CAPTCHAs automatically is not an easy task and different markets utilize different types of CAPTCHAs. Therefore the scraper is delayed until human operators are able to manually input the required CAPTCHAs to extend browsing time allowance. In the parsing stage, the program will automatically parse the scraped html files and automatically insert the extracted information into structured database tables.

3 Scraping Stage

In order to scrape the cryptomarkets, the computer needs to be connected to the Tor network. Because the university security disallows Tor connections, we use AWS as a proxy to connect to Tor, as shown in Fig. 3.

AWS Setup: We register an AWS account and launch an instance of EC2 t2.micro Ubuntu 18.04 with 1 CPU, 1 GB memory, and 30 GB disk, which is free for 1 year. The download speed is about 970 Mbit/s and the upload speed is about 850 Mbit/s. In the EC2 Dashboard webpage, we add Custom TCP Rule for ports 9001 and 9003 from anywhere to the Inbound of the security group. To install Tor on the server, we use Tor Relay Configurator [9], where we select "Ubuntu Bionic Beaver (18.04 LTS)" for the Operating System, "Relay" for the Tor node type. We do not choose "Exit Node" since AWS disallows Tor exit nodes because of the potential abuse complaints [10]. We leave ORPort and DirPort as defaults and set the total monthly traffic limit to 500 GB, maximum bandwidth to 1 Mbit/s, and maximum burst bandwidth to 2 Mbit/s. After clicking on the Submit button, users will receive a command starting with "curl." Running that command in the terminal of the AWS server will install Tor. After Tor is installed, comment "SocksPort 0" in the Tor configuration file "/etc/tor/torrc" to allow SOCKS connection [11,12]. Users must then type "sudo ss -n -p state listening src 127.0.0.1" to make sure that Tor is listening to port 9050 for SOCKS connection. Restarting Tor service by "sudo service tor restart" will display the message "Self-testing indicates your ORPort is reachable from the outside. Excellent. Publishing server descriptor." in the log file "/var/log/tor/notices.log". This means Tor is successfully installed. Three hours after Tor installation, you will find it in Tor Relay Search website by searching the nickname [13].

Python Scraper: Part 1: Tor Network Connection: Users can now connect the local Ubuntu virtual machine to the AWS server through SOCKS via command

```
$ ssh ubuntu@serverid.amazonaws.com -i key.pem -L50000:localhost:9050 -f -N
```

Replace "serverid" and "key.pem" with your own server's information. Users can test the Tor connection by opening a Firefox browser and set the "Preferences - General - NetworkSettings" to "ManualProxyConfiguration - SocksHost:127.0.0.1 - Port:50000" and "Yes: Proxy DNS when using SOCKS v5". After that, check the status of the Tor connection by visiting the website [14] in Firefox.

In Python, we use os.system("ssh ...") command to connect to the AWS server. To setup the SOCKS connection, we first create an instance of the Selenium Webdriver by "aProfile = webdriver.FirefoxProfile()", and then set up the preferences in Table 3 through "aProfile.set_preference(Preference, Value)".

Table 3. Network configurations for connecting to Tor in Python

Preference	Value	Meaning
network.proxy.type	1	Use manual proxy configuration
network.proxy.socks	127.0.0.1	SOCKS host
network.proxy.socks_port	50000	The port used the SSH command
network.proxy.socks_remote_dns	True	Proxy DNS when using SOCKS v5
network.dns.blockDotOnion	False	Do not block .onion domains

Firefox is the best option for connecting to Tor since Tor browser is modified from Firefox. Firefox is more friendly for Linux than Windows OS. Therefore we implement the Python scrapers in Ubuntu OS.

Python Scraper: Part 2: Database Design and Connection: Our database server has CentOS 7 and MariaDB, which is a fork of MySQL. We run the command "mysql -u root -q" in the terminal to connect to the MySQL database. We first create a database for the scraping stage by command "CREATE DATABASE cryptomarket_scraping;". Our scrapers run on local Ubuntu virtual machines, which will remotely connect to the database server. To enable remote database connection, we run the command "grant all on cryptomarket_scraping.* to 'user' identified by 'passwd';" in the terminal of the database server. Table 4 shows the seven tables in the cryptomarket_scraping database.

Table 4. Tables in the cryptomarket_scraping database

Table name	Table content
cryptomarkets_list	List of cryptomarkets
product_list	List of unique products
product_desc_scraping_event	Events of scraping product descriptions
product_rating_scraping_event	Events of scraping product ratings
vendor_list	List of unique vendors
vendor_profile_scraping_event	Events of scraping vendor profiles
vendor_rating_scraping_event	Events of scraping vendor ratings

Table 5 shows the description of the "cryptomarkets_list" table. Information on the seven cryptomarkets is inserted manually. The scraper program will read the table and retrieve the market URL, username, and password information to navigate to and log into the market website.

Table 5. Description of the "cryptomarkets_list" table

Field	Type	Null	Key	Default	Extra
cryptomarket_global_ID	int(11)	NO	PRI	NULL	auto_increment
cryptomarket_name	varchar(256)	NO	UNI	NULL	
cryptomarket_name_abbr	varchar(2)	NO	UNI	NULL	
cryptomarket_url	text	NO		NULL	
my_username	text	YES		NULL	
my_password	text	YES		NULL	

Table 6 shows the description of the "product_list" table. It stores the information of products and helps avoid scraping the same product multiple times.

The fields whose names start with "my_lock" are used for concurrent writing. Table 7 shows the description of the "product_desc_scraping_event" table. It stores the events of scraping product webpages and maintains the scraping history. The scraped html files are stored in the file system and the html file paths are stored in the "product_desc_file_path_in_FS" field. Table 8 shows the description of the "vendor_list" table. It stores the information of vendors and helps avoid scraping duplicate vendors. Table 9 shows the description of the "vendor_rating_scraping_event" table. It stores the fields from scraping vendor webpages. The descriptions of the "product_rating_scraping_event" and "vendor_profile_scraping_event" tables are omitted.

Table 6. Description of the "product_list" table

Field	Type	Null	Key	Default	Extra
product_global_ID	int(11)	NO	PRI	NULL	auto_increment
cryptomarket_global_ID	int(11)	NO	MUL	NULL	
product_market_ID	varchar(256)	NO		NULL	
last_scraping_time_pr	text	YES		NULL	
my_lock_pr	tinyint(1)	NO		0	
last_scraping_time_pd	text	YES		NULL	
my_lock_pd	tinyint(1)	NO		0	

The scraped html files are saved to the disk of the database server and the full paths of the html files are stored in the table. For example, the "vendor_rating_file_path_in_FS" field in the "vendor_rating_scraping_event" table contains the full path of the html files. In the parsing stage, the program will read and parse the html files.

In Python, we import the mysql and mysql.connector packages for MySQL connections. Specifically, we call the "aDB = mysql.connector.connect(host, user, passwd, database, port, buffered)" function to connect to the database server. The database cursor can thus be obtained by "aDBCursor = aDB.cursor(dictionary=True)". We can execute any SQL commands by calling the "aDBCursor.execute(aSQLStatement)" function, where "aSQLStatement"

Table 7. Description of the "product_desc_scraping_event" table

Field	Type	Null	Key	Default	Extra
scraping_event_ID_product	int(11)	NO	PRI	NULL	auto_increment
product_global_ID	int(11)	NO	MUL	NULL	
scraping_time	text	NO		NULL	
product_desc_file_path_in_FS	text	YES	MUL	NULL	

Table 8. Description of the "vendor_list" table

Field	Type	Null	Key	Default	Extra
vendor_global_ID	int(11)	NO	PRI	NULL	auto_increment
cryptomarket_global_ID	int(11)	NO	MUL	NULL	
vendor_market_ID	varchar(256)	NO		NULL	
last_scraping_time_vr	text	YES		NULL	
my_lock_vr	tinyint(1)	NO		0	
last_scraping_time_vp	text	YES		NULL	
my_lock_vp	tinyint(1)	NO		0	

Table 9. Description of the "vendor_rating_scraping_event" table

Field	Type	Null	Key	Default	Extra
scraping_event_ID_vendor	int(11)	NO	PRI	NULL	auto_increment
vendor_global_ID	int(11)	NO	MUL	NULL	
scraping_time	text	NO		NULL	
vendor_rating_file_path_in_FS	text	YES	MUL	NULL	

represents a SQL statement. In the scraper program, we execute the SELECT, INSERT, and UPDATE statements. To fetch the data records, we call the "aDBCursor.fetchone()" or "aDBCursor.fetchall()" function. After we finish the operation, we always call the "aDB.close()" function to close the connection. Please refer to the source code for more details.

The "cryptomarket_scraping" database stores the data scraped from all seven cryptomarkets since all markets contain products, vendors, and ratings. Therefore, in Python, we design a class containing the MySQL functions, which is independent of the scraper classes of different cryptomarkets. Each scraper class will call the MySQL functions to interact with the database.

Python Scraper: Part 3: Scraper Design

The seven cryptomarkets in Table 2 can be categorized into two groups. Dream, Berlusconi, Valhalla, Empire, Point Tokcha, and Silk Road 3.1 belong to the first group. Wall Street itself belongs to the second group. The two market groups differ in how the webpages are navigated. In the first group, changing the URL will navigate to different pages. For example, in Dream Market, the following link is the URL of page 2 of products.

http://effsggl5nzlgl2yp.onion/?page=2&category=103

We can change the page value to navigate to different pages. However, in Wall Street, the URL does not contain page information. We always get the same link:

http://wallstyizjhkrvmj.onion/index

Table 10. Properties of cryptomarkets

cryptomarkets	login	CAPTCHA
Dream	Yes	Yes
Berlusconi	Yes	No
Wall Street	Yes	Yes
Valhalla	No	No
Empire	Yes	Yes
Point Tochka	Yes	No
Silk Road 3.1	No	Yes

This URL will not change when we click on the "Next (page)" button. Based on the above observations, we design two scraping strategies: 1. Scraping the webpages of products and vendors on one product-list page first, and then navigating to the next product-list page; 2. Navigating multiple product-list pages first, and then scraping the webpages of products and vendors listed in those product-list pages. Strategy 1 is used for the cryptomarkets in group 1. Strategy 2 is only used for Wall Street (group 2). Following these strategies, we design a Python scraper program for each cryptomarket.

CAPTCHA is an acronym for "completely automated public Turing test to tell computers and humans apart". It is a challenge-response test used in computing to determine whether or not the user is human. Different cryptomarkets require different types of CAPTCHAs. The CAPTCHAs are the major obstacle in scraping the websites. In our scrapers, we rely on humans to input those CAPTCHAs. The scraping program will stall whenever it encounters a webpage requiring CAPTCHAs. We use the explicit wait method provided in the Selenium package. More specifically, we call the "aWait=WebDriverWait(aBrowerDriver, nSecondsToWait)" and "aWait.until (EC. element_to_be_clickable(. . .))" functions. The program will wait until some element that never appears in the webpage containing CAPTCHAs appears in the new webpage and is clickable. Since the speed of loading an .onion webpage is slow, waiting for a short time period like 2 s before extracting the product and vendor information will help reduce program errors. During the experiments, we find that Dream, Wall Street, Empire, and Silk Road 3.1 markets require CAPTCHAs, but Berlusconi, Point Tochka, and Valhalla markets do not require CAPTCHAs. We also find that Dream, Wall Street, Empire, Berlusconi, and Point Tochka markets require logins, but Silk Road 3.1 and Valhalla do not require logins. Table 10 summarizes these properties.

4 Parsing Stage

In the parsing stage, we implemented the Python parser programs to read the data stored in the "cryptomarket_scraping" database, parse various information from the html files, and store the parsed data into the "cryptomarket_parsed" database.

Python Parser: Part 1: Database Design: All cryptomarkets contain the webpages for products and vendors. In the product webpages, product title, description, vendor, price, shipping, and rating information are shown. In the vendor webpages, vendor name, profile, sales records, and rating information are shown. Therefore we create the "product_descriptions" table and the "vendor_profile" table to store the product and vendor information respectively. We notice that the rating information is usually shown in tables in the product or vendor webpages since vendors have an incentive to maintain their reputation. Therefore, we created two more tables for storing ratings. An additional table is created for memorizing the progress of the parser. Table 11 shows the five tables created in the "cryptomarket_parsed" database.

Table 11. Tables in the "cryptomarket_parsed" database

Table name	Table content
parser_progress	The progress of parsing
product_descriptions	Product descriptions
product_ratings	Ratings on products' webpages
vendor_profiles	Vendor profiles
vendor_ratings	Ratings on vendors' webpages

Table 12 shows the "parser_progress" table. It is used for memorizing the progress of the parser. If the program is interrupted by unexpected issues, the parser can continue parsing the htmls file from where it stopped.

Table 12. Description of the "parser_progress" table

Field	Type	Null	Key	Default
last_parsed_scraping_event_ID_pd	int(11)	YES		NULL
last_parsed_scraping_event_ID_pr	int(11)	YES		NULL
last_parsed_scraping_event_ID_pd_4pr	int(11)	YES		NULL
last_parsed_scraping_event_ID_vp	int(11)	YES		NULL
last_parsed_scraping_event_ID_vr	int(11)	YES		NULL
last_parsed_scraping_event_ID_vp_4vr	int(11)	YES		NULL

Tables 13 and 14 show the partial descriptions of the "product_descriptions" table and the "vendor_profiles" table respectively. Some fields are omitted. Table 15 shows the description of the "product_ratings" or "vendor_ratings" table. These two tables both contain ratings and have the same description.

Table 13. Description of the "product_descriptions" table

Field	Type	Null	Key	Default	Extra
index_pd	int(11)	NO	PRI	NULL	auto_increment
scraping_time	varchar(256)	NO		NULL	
cryptomarket_global_ID	int(11)	NO		NULL	
product_global_ID	int(11)	NO		NULL	
vendor_global_ID	int(11)	NO		NULL	
vendor_market_ID	varchar(256)	NO		NULL	
vendor_market_name	varchar(256)	NO		NULL	
product_title	varchar(256)	NO		NULL	
product_desc	text	YES		NULL	
price_bitcoin	float	YES		0	
price_usd	float	YES		0	
price_eur	float	YES		0	
ships_to	varchar(256)	NO		NULL	
ships_from	varchar(256)	NO		NULL	
escrow	varchar(256)	NO		NULL	
category	varchar(256)	NO		NULL	
num_sales	int(11)	NO		0	
num_stock	int(11)	NO		0	

Table 14. Partial description of the "vendor_profiles" table

Field	Type	Null	Key	Default	Extra
index_vp	int(11)	NO	PRI	NULL	auto_increment
scraping_time	varchar(256)	NO		NULL	
cryptomarket_global_ID	int(11)	NO		NULL	
vendor_global_ID	int(11)	NO		NULL	
vendor_market_ID	varchar(256)	NO		0	
vendor_profile	text	YES		NULL	
terms_conditions	text	YES		NULL	
join_date_member_since	varchar(256)	YES		NULL	
last_active_date	varchar(256)	YES		NULL	
pgp	text	YES		NULL	
num_orders_completed	int(11)	YES		0	
num_orders_open	int(11)	YES		0	

Table 15. Description of the "product_ratings" or "vendor_ratings" table

Field	Type	Null	Key	Default	Extra
index_pr	int(11)	NO	PRI	NULL	auto_increment
cryptomarket_global_ID	int(11)	NO		NULL	
vendor_global_ID	int(11)	NO		NULL	
buyer_market_ID	varchar(256)	NO		NULL	
rating_stars	float	YES		0	
text_comments	text	YES		NULL	
post_date	varchar(256)	YES		NULL	
money_bitcoin	float	YES		0	
buyer_total_num_of_orders	int(11)	YES		0	
buyer_total_value_of_orders	int(11)	YES		0	
product_global_ID	int(11)	YES		NULL	

Python Parser: Part 2: Parser Design: We use polymorphism to design the parsers. We first create a base class called "parser_base". In the base class, we create a member variable for each field of the tables in the "cryptomarket_parsed" database. In the Python source code, the member variables always start with "m_". In the base class, we implement all database related functions. For example, the "insert_one_product_rating()" function will insert a product rating into the "product_ratings" table. In the base class, we also design four abstract functions:

1. def parse_one_html_product_descriptions(self): pass
2. def parse_one_html_product_ratings(self): pass
3. def parse_one_html_vendor_profiles(self): pass
4. def parse_one_html_vendor_ratings(self): pass

Each cryptomarket website has its own design. Therefore, we need to design a parser for each cryptomarket. For each cryptomarket, we design a child class inheriting from the base class. For example, the parser class for Dream Market is defined as "class parser_dream(parser_base):". In each of the seven child classes, we only implement the above four abstract parsing functions.

When parsing an html file, we use the BeautifulSoup package. Once a html file is read by "aFile = open(filename, encoding='utf-8')", we directly convert it into a BeautifulSoup instance by "aBS = BeautifulSoup(aFile, features='html.parser')". Then, we call "aBS.findChild(...)" to find a single element or "aBS.findChildren(...)" to find a set of elements in the html file, and then obtain data for different columns of the tables. Regular expressions are useful for finding elements satisfying certain conditions. For example, we call "aBS.findChildren('a', {'herf': re.compile('http://.*')})" to find all 'a' elements whose 'href' attributes start with "http://". Please see the Python source code for more examples.

After we implement the scrapers for the cryptomarkets, we can start parsing product descriptions, product ratings, vendor profiles, and vendor ratings. We create a Python file for each task. In each file, the program reads the records one by one from a table, identifies the cryptomarket, and calls the corresponding parsing function to parse the html file. The parsing results will be inserted into one of the four tables: product_descriptions, product_ratings, vendor_profiles, and vendor_ratings.

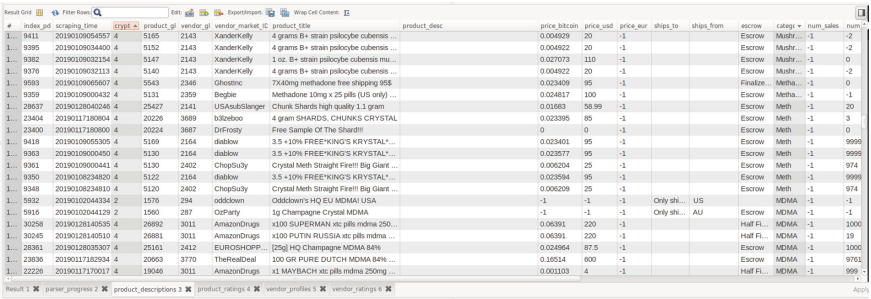


Fig. 4. The screenshot of the "product_descriptions" table in MySQL Workbench

5 Experimental Results

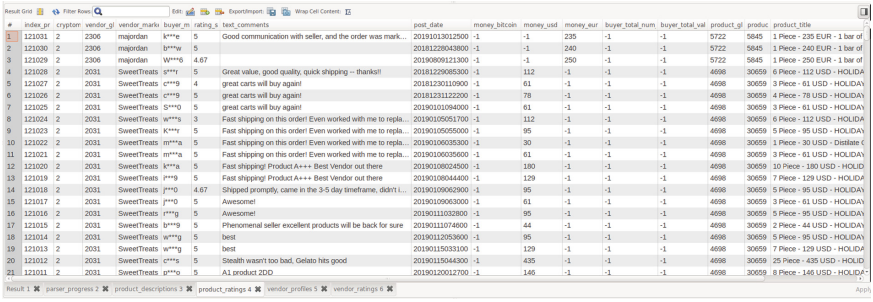
One student runs the scrapers for two weeks and inputs many CAPTCHAs. The total number of hours for scraping the data is about 70 h. The parsing stage only costs several hours and is automatic without human intervention. Table 16 shows the numbers of data records in the tables in the "cryptomarket_parsed" database. From Table 16, we can see that the student has scraped 26,190 products, 3,950 vendors, 119,934 product ratings, and 626,850 vendor ratings. The dataset can be easily shared with other researchers through MySQL Workbench.

Table 16. Statistics of the data

Table name	#records
product_descriptions	26,190
product_ratings	119,934
vendor_profiles	3,950
vendor_ratings	626,850

Figure 4 shows a screenshot of the "product_descriptions" table. In Fig. 4, each row represents a product and each column represents a property of the product. Empty cells indicate missing values. The "product_desc" column usually contains a long text description of a product.

Figure 5 shows a screenshot of the "product_ratings" table. In Fig. 5, each row represents a product rating and each column represents a property of the rating. From Fig. 5, we can see that each rating contains the vendor, buyer, date, rating stars, comment, money, and product information. Each rating actually represents one transaction. The buyer IDs are masked for privacy protection.



#	index_pr	cryptom	vendor_gl	vendor_marke	buyer_m	rating_s	text_comments	post_date	money_bitcoin	money_usd	money_eur	buyer_total_num	buyer_total_val	product_gl	produc	product_title
1	121031	2	2306	majordan	k***e	5	Good communication with seller, and the order was mark...	20191013012500	-1	-1	235	-1	-1	5722	5845	1 Piece - 235 EUR - 1 bar of
2	121030	2	2306	majordan	b***w	5		20181228043800	-1	-1	240	-1	-1	5722	5845	1 Piece - 240 EUR - 1 bar of
3	121029	2	2306	majordan	W***6	4.67		20190809121300	-1	-1	250	-1	-1	5722	5845	1 Piece - 250 EUR - 1 bar of
4	121028	2	2031	SweetTreats	s***r	5	Great value, good quality, quick shipping -- thanks!!	20181229085300	-1	112	-1	-1	-1	4698	30659	6 Piece - 112 USD - HOLIDA
5	121027	2	2031	SweetTreats	c***9	4	great carts will buy again!	20181230110900	-1	61	-1	-1	-1	4698	30659	3 Piece - 61 USD - HOLIDAY
6	121026	2	2031	SweetTreats	c***9	5	great carts will buy again!	20181231122200	-1	78	-1	-1	-1	4698	30659	4 Piece - 78 USD - HOLIDAY
7	121025	2	2031	SweetTreats	S***0	5	great carts will buy again!	20190101094000	-1	61	-1	-1	-1	4698	30659	3 Piece - 61 USD - HOLIDAY
8	121024	2	2031	SweetTreats	w***s	3	Fast shipping on this order! Even worked with me to repla...	20190105051700	-1	112	-1	-1	-1	4698	30659	6 Piece - 112 USD - HOLIDA
9	121023	2	2031	SweetTreats	K***r	5	Fast shipping on this order! Even worked with me to repla...	20190105055000	-1	95	-1	-1	-1	4698	30659	5 Piece - 95 USD - HOLIDAY
10	121022	2	2031	SweetTreats	m***a	5	Fast shipping on this order! Even worked with me to repla...	20190106035300	-1	30	-1	-1	-1	4698	30659	1 Piece - 30 USD - Distilate C
11	121021	2	2031	SweetTreats	m***a	5	Fast shipping on this order! Even worked with me to repla...	20190106035600	-1	61	-1	-1	-1	4698	30659	3 Piece - 61 USD - HOLIDAY
12	121020	2	2031	SweetTreats	k***a	5	Fast shipping! Product A+++ Best Vendor out there	20190108024500	-1	180	-1	-1	-1	4698	30659	10 Piece - 180 USD - HOLID
13	121019	2	2031	SweetTreats	r***9	5	Fast shipping! Product A+++ Best Vendor out there	20190108044400	-1	129	-1	-1	-1	4698	30659	7 Piece - 129 USD - HOLIDA
14	121018	2	2031	SweetTreats	j***0	4.67	Shipped promptly, came in the 3-5 day timeframe, didn't i...	20190109062900	-1	95	-1	-1	-1	4698	30659	5 Piece - 95 USD - HOLIDAY
15	121017	2	2031	SweetTreats	j***0	5	Awesome!	20190109063000	-1	61	-1	-1	-1	4698	30659	3 Piece - 61 USD - HOLIDAY
16	121016	2	2031	SweetTreats	r***g	5	Awesome!	20190111032800	-1	95	-1	-1	-1	4698	30659	5 Piece - 95 USD - HOLIDAY
17	121015	2	2031	SweetTreats	b***9	5	Phenomenal seller excellent products will be back for sure	20190111074600	-1	44	-1	-1	-1	4698	30659	2 Piece - 44 USD - HOLIDAY
18	121014	2	2031	SweetTreats	w***g	5	best	20190112053600	-1	95	-1	-1	-1	4698	30659	5 Piece - 95 USD - HOLIDAY
19	121013	2	2031	SweetTreats	w***g	5	best	20190115033100	-1	129	-1	-1	-1	4698	30659	7 Piece - 129 USD - HOLIDA
20	121012	2	2031	SweetTreats	c***s	5	Stealth wasn't too bad, Gelato hits good	20190115044300	-1	435	-1	-1	-1	4698	30659	25 Piece - 435 USD - HOLID
21	121011	2	2031	SweetTreats	p***o	5	A1 product 2DD	20190120012700	-1	146	-1	-1	-1	4698	30659	8 Piece - 146 USD - HOLIDA

Fig. 5. The screenshot of the "product_ratings" table in MySQL Workbench

Figure 6 shows a screenshot of the "vendor_profiles" table. In Fig. 6, each row represents a vendor and each column represents a property of the vendor. The "vendor_profile" and "term_conditions" are two similar properties. Some cryptomarkets call it "vendor_profile" while others call it "term_conditions". They both contain the long text descriptions of a vendor, which usually contains rich information about the vendor.

Figure 7 shows a screenshot of the "vendor_ratings" table. In Fig. 7, each row represents a vendor rating and each column represents a property of the rating. Each rating contains the vendor, buyer, date, rating stars, and money information. Similar to the ratings in the "product_rating" table, each rating here still represents one transaction. But the product information is usually not provided in the "vendor_ratings" table in most cryptomarkets. The buyer IDs are also masked for privacy protection.

We further perform preliminary analysis and visualization on the parsed data. We first count the number of ads in each country. We use the "ship from" information to determine the country of an ad. Note that some vendors may provide fake information, e.g., they are in France but they claim Germany. In this work, we did not analyze the authenticity but it is an interesting problem. Figure 8 shows the global distribution of ads across the largest seven cryptomarkets.

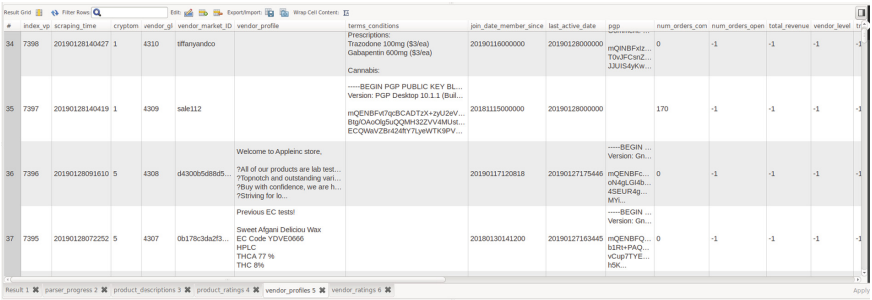


Fig. 6. The screenshot of the "vendor_profiles" table in MySQL Workbench

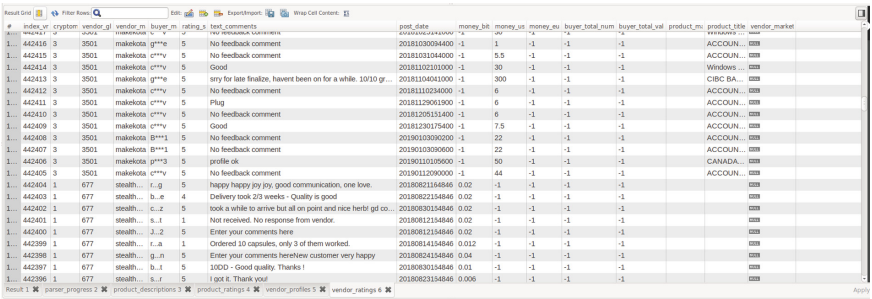


Fig. 7. The screenshot of the "vendor_ratings" table in MySQL Workbench

The hue is proportional to the number of ads. We can see that a large number of ads are from USA, UK, Australia, Germany, and Canada. A small number of ads are from Russia because we did not scrape the Russian cryptomarket called the Black Market.

We further construct a social network among buyers and vendors. Each node represents a buyer or a vendor, and each edge represents that a buyer orders products from a vendor. The ordering information is collected from the feedback ratings. Figure 9 shows a subgraph of the social network in the Dream Market. The purple nodes represent vendors and the white nodes represent buyers. The buyer IDs are masked and only the initial and last letters are visible. The ten vendors all sell digital goods. From Fig. 9, we can observe the community structures among buyers. Most buyers only buy from one vendor, and only a few buyers buy from more than one vendor. We will analyze the collected data further and discover more patterns.

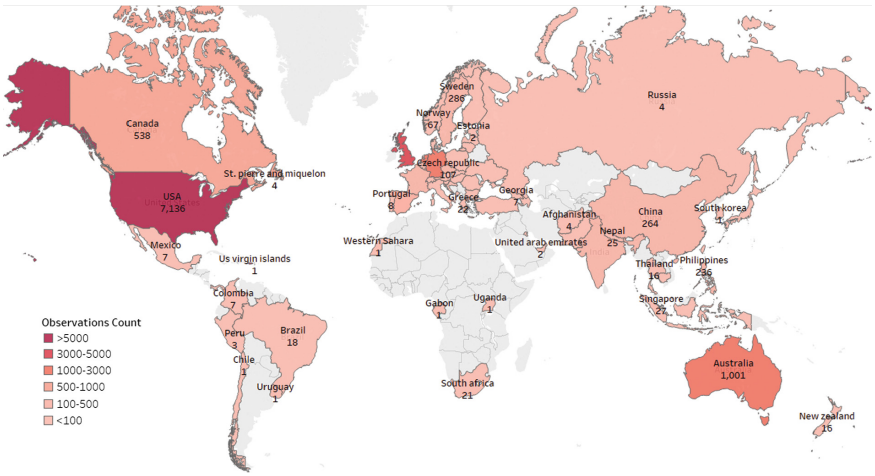


Fig. 8. The global distribution of ads across the largest seven cryptomarkets

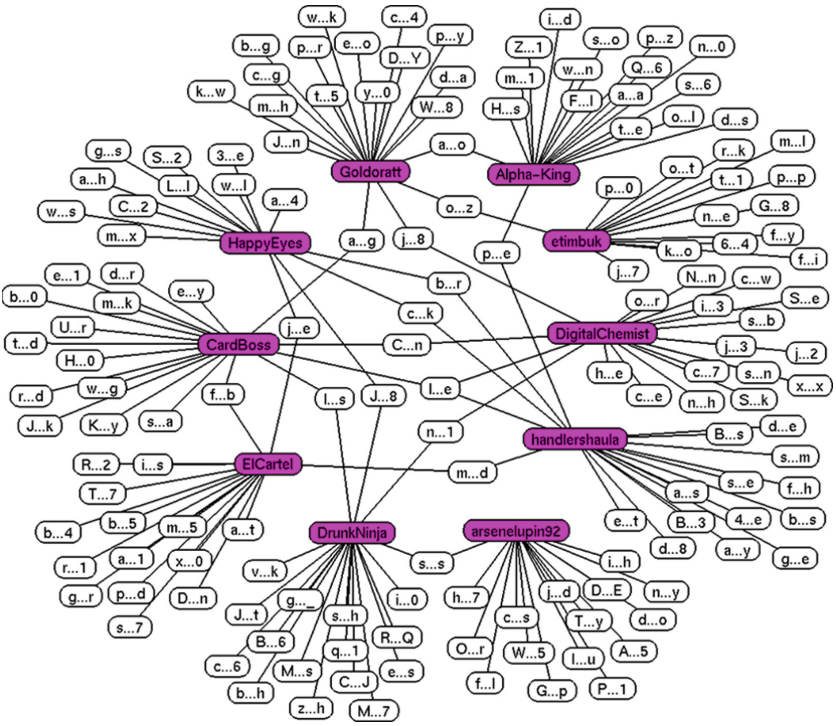


Fig. 9. The social network between buyers and vendors in the Dream Market

6 Conclusion

Cryptomarket websites contain rich information about illicit criminal activities. It is urgent to develop scrapers to collect data from cryptomarkets and then develop AI algorithms for assembling intelligence from the data. In this paper, we design and implement Python scrapers for scraping the seven largest cryptomarkets. This work demonstrates the effectiveness and efficiency of the developed scrapers and provides the foundation for the next stage of data analysis. The source code of the scrapers is publicly available.

References

1. Martin, J.: Drugs on the Dark Net: How Cryptomarkets are Transforming the Global Trade in Illicit Drugs (2014)
2. Aldridge, J., Décary-Hétu, D.: Not an 'Ebay for Drugs': the Cryptomarket 'Silk Road' as aparadigm shifting criminal innovation. Available at SSRN 2436643 (2014)
3. Christin, N.: Traveling the silk road: a measurement analysis of a large anonymous online marketplace. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 213–224. ACM (2013)
4. EMCDDA: Europol: DarkNet markets ecosystem – lifetimes and reasons for closure of over 100 global darknet markets offering drugs, sorted by date (2018)
5. European Monitoring Centre for Drugs and Drug Addiction and Europol: Drugs and the DarkNet: perspectives for enforcement, research and policy (2017)
6. DarkNet Market Archives (2013–2015). https://www.gwern.net/DNM-archives. Accessed 12 Feb 2019
7. Lawrence, H., Hughes, A., Tonic, R., Zou, C.: D-miner: a framework for mining, searching, visualizing, and alerting on darknet events. In: 2017 IEEE Conference on Communications and Network Security (CNS), pp. 1–9. IEEE (2017)
8. Hayes, D., Cappa, F., Cardon, J.: A framework for more effective dark web marketplace investigations. Information **9**(8), 186 (2018)
9. Tor Relay Configurator. https://tor-relay.co/. Accessed 15 Feb 2019
10. Tor Good Bad ISPs. https://trac.torproject.org/projects/tor/wiki/doc/GoodBadISPs. Accessed 15 Feb 2019
11. Tor Relay Guide. https://trac.torproject.org/projects/tor/wiki/TorRelayGuide. Accessed 15 Feb 2019
12. Tor Manual. https://www.torproject.org/docs/tor-manual.html.en. Accessed 15 Feb 2019
13. Tor Relay Search. https://metrics.torproject.org/rs.html. Accessed 15 Feb 2019
14. Check Tor Connection. https://check.torproject.org. Accessed 15 Feb 2019