Building a web- or forum crawler to extract (and analyze) scam/extortion emails

1st David Jungmalm
Department of Computer and Information Science
Linkoping University
Linköping, Sweden
davju247@student.liu.se

2nd Markus Loborg
Department of Computer and Information Science
Linkoping University
Linköping, Sweden
marlo975@student.liu.se

***Abstract*—Newspapers are reporting an increased amount of scam emails. The subject of scammers trying to get people to pay them with Bitcoin in exchange for not leaking personal information is a threat. In this report we built a web crawler to extract these mails from different forums in order to do further analysis of the threat types and in later reports also see patterns in where the Bitcoins disappear. The crawler that is able to collect and store emails posted on Reddit and Bitcoin Abuse, that can also do some basic analysis of these emails. The result of the analysis is that it indeed appears as the scam mails are increasing. Most commonly there was blackmail with a sub specification of sextortion. Many different email addresses and bitcoin addresses were used.**

***Index Terms*—crawler, email, scam, forum, analysis, bitcoin, abuse, Reddit, blackmail**

I. Introduction

In the news there have been reports that more and more scam mails are happening, or at the very least more scams are being reported. Many scammers are trying to threaten people in order to make them send money to the scammer and the European Commission reports that the European citizens may have lost about 24 billion EUR as a result of scams and frauds [2] in the last two years (2018-2020) making scams a substantial security problem. While the number of reports grow, people share the mails in forums to help people who have gotten similar mails to know it is a scam. Despite this, these mails mostly stay on the forums and do not get added to any public dataset that could be used by companies and programs to recognize new scam mails. This is something that we want to change, therefore we constructed a web crawler to find all these posted mails and collect them into a dataset. This paper will describe how we did this as well as some of the results we could draw from the gathered dataset. Our results and investigation will be focused around the following questions:

- Are the amount of threats in extortion/scam mails increasing?
- What percentage of the mails are using threats?
- What is the most common type of threat?
- How can a web crawler be designed for the purpose of finding scam email patterns?

The rest of this paper is organized as follows. Firstly there will be explanations of terms as well as descriptions of the different frameworks and tools that were used to create the web crawler, in section II. Then, in section III, we will describe how we went about creating the web crawler. This will be followed by the results from our analysis as well as an evaluation of the web crawler in section IV. Then we proceed to analyze and discuss our findings and sources in section V. Following that we draw our final conclusions based on the result and the analysis in section VI.

II. Background

In this section we present the necessary information needed to understand the rest of the report.

A. Bitcoin

Bitcoin is an online cryptocurrency aiming to remove the need of third party financial institutions. It is a pure peer-to-peer electronic currency that allows users to make direct payments without going through any financial institution. Many modern banking solutions offer digital currency without the need of physical money bills and with this solution often comes a debit card to make transactions. While it might be easy to think that this is the same thing, they have a massive difference in that debit cards are still tied to a central bank who controls the transactions making them centralized. There is also no way for anyone except the bank to see the transactions making them non-transparent. Bitcoin, on the other hand is decentralized meaning they are not controlled by a central institution but on the other hand they are transparent meaning anyone can see any transaction in what is called a blockchain. Any wallet can send money to any other wallet, but the wallet can not be bound to any person making them popular in email scams trying to blackmail persons for bitcoin money [3].

B. Web scraping

Web scraping is a way of extracting data from the world wide web. In its basic form, copy-pasting is a type of web scraping at a minimal scale which can be done much more efficiently by computers. This is what search engines like google.com is doing, searching the web for keywords that the user is inserting. Web crawlers, or spiders, is an intelligent program that searches and indexes web pages, follows links etc. that can be used to extract information and later export the information into useful file formats like JSON [4].

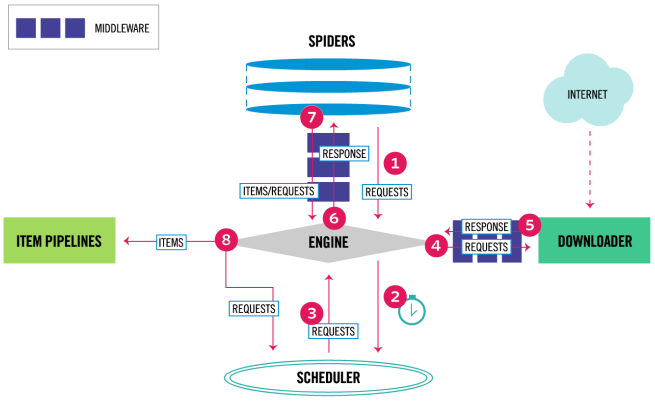

Fig. 1. Scrapy architecture.

Fig. 2. Typical HTML structure.

C. Scrapy

Scrapy is an application framework that can be used to crawl websites and extract structural data. It is a Python library and it also supports extraction of data using APIs, like the Reddit API. This can be useful in the case where CSS selectors are hard to use, like Reddit which has nested comments i.e. every comment can have a reply which has a reply. The architecture of Scrapy is shown in Fig. 1. All information is processed in the spiders, and these are implemented by the user. The spiders define start URLs and for each URL sends that request to the engine which will schedule a request from the downloader. After the response is received it is sent to the spiders to be handled. The exact details of the spiders will be covered in section III. After the spider has done what it needs, it can send a new request if there are links to follow etc., or output items through pipelines. These items are what the user gets back after running the crawler meaning the information one wants to extract from a given website [5].

D. CSS selectors

Cascading Styling Sheets (CSS) is a language that describes the style of HTML documents which build up websites. A way to refer to these HTML elements is using CSS selectors. In Fig. 2 you can see the typical structure of HTML, and let us say one want to extract the link in <a href>. Then the CSS selector would be ".mb-3:nth-child(1) a" where .mb-3 is the class name of the div in which <a>is stored, :nth-child(1) means the first child with the class name "mb-3" and lastly chose a tag in this div tag. The CSS selectors are useful for defining what in a web page one wants to extract. Some useful selectors are shown in table I.

E. SelectorGadget for Chrome

A useful tool for getting the correct CSS Selector for a specific element or group is the SelectorGadget extension for Chrome browser. It lets you click an element and it will show a CSS Selector and also display which other elements are included in the current selector. You can also choose if some elements should not be included with the selector [8].

TABLE I
USEFUL CSS SELECTORS

Selector	Example	Example description
.class	.container	Selects all elements with class="container"
element.class	ul.pagination	Selects all elements with class="pagination"
#id	#table	Select all elements with id="table"
element	a	Select all <a>elements
:nth-child(n)	td:nth-child(4)	Selects every <td> element that is the fourth child of its parent
:nth-last-child(n)	li:nth-last-child(1)	Selects every element that is the first element of its parent, counting from the last child

F. Reddit

Reddit is a website made for posting about different subjects. Each subject has its own page called a subreddit. In these pages users can post things about that subject and comment on posts or on other comments [9].

G. PRAW

PRAW stands for Python Reddit API Wrapper and is as it says an API for Reddit written in Python. With PRAW you can fetch any subreddit, comment, user, post etc. from Reddit. You can also extract any of the available data from these such as timestamps, content, etc [11].

III. METHOD

In order to build the crawlers, the first thing needed was websites containing some sort of information that interests our study. Mostly websites containing mails with bitcoin references in them. Although raw email archives were very hard to find, the best effort was made to ensure that the bitcoin addresses found were truly used in scam mails. The websites

Fig. 3. HTML structure of reports on main page.

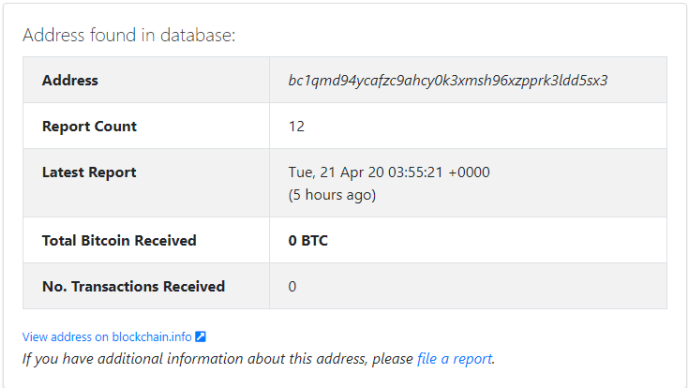

Fig. 4. Graphical view of a report.

decided to crawl were Reddit and Bitcoin Abuse [10]. A website which contained some example emails were also used, however these were considered a test run since the dataset was so small.

A. Page information

To decide the format of the crawler, the specific website structure needed to be analyzed. Since we used the Reddit API for that crawler, the analysis was mostly done for Bitcoin Abuse. Firstly the page was looked over for information that might be useful. For Bitcoin Abuse, one could visit the page /reports where every single report is listed. Inside each of those links, one can see all the common reports for that specific address. It also contains info like total reports counts, Bitcoin transactions received and descriptions. Decisions were made to extract the Bitcoin address, together with report count and all the descriptions, where every description also had a time stamp, abuse type and a name/email address of the abuser who sent the mail. One important note that was taken here is that since all reports are listed in the /reports page, there would be duplicates if no countermeasures were taken. The the needs of the crawler were the following:

1) Extract all links to reported addresses.
2) Visit links and extract address, report count and descriptions if address is not a duplicate.
3) Visit next reports page and repeat until all pages are visited (or set threshold reached).

Using the inspection tool in Chrome, as shown in Fig. 3, one can see that reports are stored in a div called row and that the specific reports are stored in a div called col-x1-4 col-md-6 mb-3 and the link wanted is stored in an a tag. The appropriate CSS selector would therefore be ”.mb-3 a” and since we want the link in href and not the text of the tag we use ”a::attr(href)”. Now that we have a link to follow we move on to the page containing the information we need. The structure of these report pages are very useful since they all follow the same standard and looks like Fig. 4. Using the Chrome inspector tool once again shows the HTML structure of the table, as seen in Fig. 5. By either looking at the HTML code or using SelectorGadget we see that the CSS selector for the address is ”#summary-table i”. Moving on we need to extract the report count field. Same strategy as before gives us

Fig. 5. HTML structure of address table shown in Fig4.

the selector ”#summary-table tr:nth-child(2) td”. The two CSS selectors above will give us the actual HTML element and to extract the text inside the element we simply add ::text right after the selector. Lastly looking at the table containing all the descriptions for the reported address we see the structure shown in Fig. 6 And as seen in the Chrome inspector tool we see that the class name for the table is rather complicated, however using the SelectorGadget we see that a possible selector is ”td:nth-child(4)” meaning that the elements in the table have a <td>field and is happens to be the only 4th element with that tag meaning it can be used as selector. As before, we add ::text to extract the text.

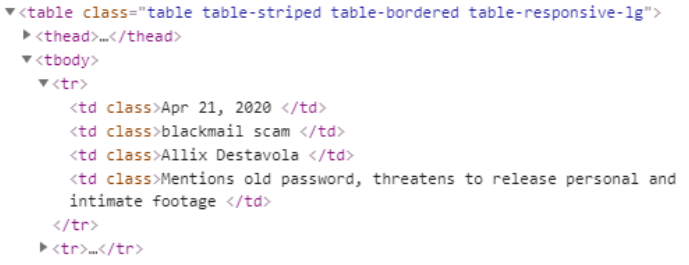

Fig. 6. HTML structure of report descriptions.

B. Presenting data

Scrapy has the functionality to output scraped data as JSON files. The way this is done is by using yield keyword, almost like a return but it keeps enough state to resume execution after the yield. There are two main ways to return data. Either you yield JSON data directly in the form **yield** {**"message": message, "address": address**} or you use Scrapy items. This is a class that extends the scrapy.Item class and assigns variables as **var = scrapy.Field()** and then you can import the item class and use it as a list. Ie if you have an **item = ScrapyItem()** with field **address = scrapy.Field()** you edit the list with **item['address'] = localvar** and then simply yield the item. Both these ways work the same, but using items is a more clean way of doing it and that is also how we chose to do it in this project.

C. Picking forums

Generally there is a lot of information on the internet, you just have to look in the right place. The definition of the problem stated that emails that contain bitcoin references were desired. Finding email was kind of hard in the first place and adding in the need for bitcoin references made it even harder. Without going completely out of scope for this project a forum that contained many mails in the same thread/site was needed since making a crawler that scrapes full sites is kind if hard and would be very time consuming to get working properly. Generally it would be kind of a time waste to search a forum with only a few emails per forum thread/site.

IV. Result

The crawler developed in this project will be described in the following section.

A. Crawler

The crawler created is able to crawl all pages on the website http://www.bitcoinabuse.com/reports. In the crawler there are two constants for deciding how many pages to crawl and what addresses to include. PAGE_THRESHOLD is the number of main pages on the website that the crawler should crawl. There is also a constant named REPORT_COUNT_THRESHOLD which is a constant that can be used to filter addresses that only have a few reports. The number defined in this constant will prevent output of any addresses that have less reports than the given number. For Bitcoin Abuse the crawler will output every address specified by the constants together with the number of reports for the given address where every report is saved as an item with four fields: date, abuse type, abuser and the description.

The crawler can also crawl any subreddit in Reddit simply by adding the subreddit name to an array of subreddits. This crawl looks into the latest posts on the subreddit and crawls each post and the comments on said posts. Apart from this it crawls one subreddit and three posts specifically since they are made to post scam emails on. The subreddit being r/Scams and the posts are old posts on that subreddit. Since the information available on Reddit is not exactly the same as on Bitcoin Abuse, the output from this part of the crawler is somewhat different. The fields outputted here are the message, the extracted bitcoin address is there exists one, if it is marked as suspicious and the time stamp.

B. Limitations

The crawler that was created has some limitations. First of all is the fact that when it crawls Reddit it is using Reddits own API for fetching the data. This API has a built in limitation of only fetching around 1000 objects. Meaning if you fetch the hottest posts of cars you will only get around the 1000 hottest posts, while if you fetch the newest posts you will get around the 1000 newest posts. This means that if any post has more than 1000 comments, we can not get them. We still get around the first 1000 but not more.

Secondly there are problems with some of the bitcoin addresses. Some mails have removed the bitcoin addresses and some have them obfuscated. This is mentioned at a larger scale in our analysis but even with the tools we used to minimize it some will still slip through our grasp.

Thirdly there is a limitation of correctness. Sadly there are both false positives and false negatives in the crawler. This is due to the fact that there is no real way to say this is a mail and this is a comment. The way we did it is by looking for suspiciousness or bitcoin addresses in the text. If it contained either of that, then we added it to the list. By suspiciousness we are referring to the obfuscation talked about above for the bitcoin addresses which could also be found in more places than just the addresses. Sadly this means that we get false negatives when the scammer has not added this obfuscation which is a minority but they do exist and the user has removed the bitcoin addresses for anonymization. It also means that if someone simply adds a comment "is this xxxx a trustworthy address" it will be counted since it contains a bitcoin address.

C. Analysis

With the crawler made for Reddit and Bitcoin Abuse a dataset of nearly 44,000 unique addresses were collected and over 155,000 different reports/emails were found. With these addresses we could do some analysis. According to Bitcoin Abuse there have been a huge spike in reports in April [10]. Same site also tells us that the number of reports for 2020 are likely going to be more than for 2019 if the average number of reports for this year keeps the same rate. Fig. 7 and Fig. 8 show graphs from Bitcoin Abuse. During the project a rudimentary analysis was made on the collected dataset. This analysis gave the following results. The most common threat was blackmail with around 65000 reports with an additional 50000 reports of sextortion which is a subgroup of blackmail. Other than that there were 36000 ransomwares and then around 5000 other reports. The bitcoin addresses used varied with a total of 44123. The usage of every single address was also varied. Some addresses were only used once while the most common one was used 928 times. The mails came from 704 unique days and the most that was reported in a single day was 8221 reports on 16th April 2020. The average emails per day was

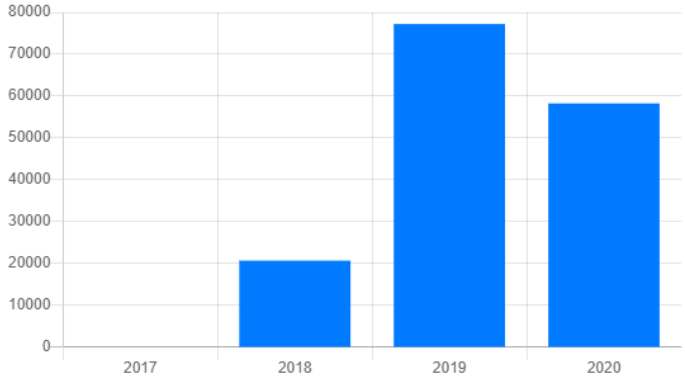

Fig. 7. Reports per year as of April 30th 2020..

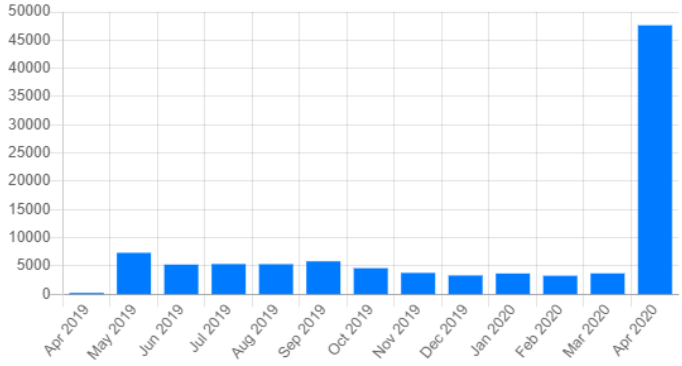

Fig. 8. Reports per month as of April 30th 2020.

around 222 but as seen above a huge spike occurred in April 2020. In fact the average before April 2020 was only 162 mails per day but the average for only April was 1637 mails per day. The mails were somewhat diverse in what they said. The most common mail was found 98 times. In total there were 1000s of unique mails. Another thing was the email address that the mails were sent from. They also varied from 86 uses down to 1.

V. Analysis

During the creation of the webcrawler we learned many things and encountered many obstacles. One of the biggest obstacles was the fact that many of the mails had characters in them that are not visible when you read the mail normally. For example you read ”Give me 100 bitcoins” but what it actually says is ”Gi\xa0 ve me 10\udcf70 bi\ufe0ftc\ufe0foi\ufe0fn”. These characters are characters that are not supposed to be used in this situation. The \u characters you see here are not visible normally because they are used for the computer to understand things about the string it reads. For example some of them are used to notify the computer that the string should be read from the left because it is a Hebrew string. Others are emoji identifiers like ”following emoji should be black”. But since no emoji follows, nothing happens and you do not see the character.

Then there were similar cases where they used characters that looked similar to normal letters. For example ”Give me 100 bitcọin”. As you see there is a small dot below the o which separates it from a normal o. You can still read bitcoin but for the computer it will see ”Give me bitc\uxxxxin” where the xxxx represents a hex number that is the code for that character. These characters are used all over the mail swapping any letter. They use anything from scientific formula notations to Russian. Any character that is in the Unicode set of characters is used.

This becomes a problem since if you ask the computer if it contains the word bitcoin it will say no. This made finding bitcoin references and analyzing the messages quite a bit harder. But there is another problem with this, and that is that the computer can have problems with encoding this. Now this is not a problem you see but when we attempted to save this to file we found lots of errors claiming it can not encode character \uxxxx. We solved both of these by creating functions that removed those that were invisible and swapped the characters to their plain text alternatives. We also used functions that simply ignored characters that could not be encoded because even with our functions there were sets of these characters that avoided us. Even with that there were still some addresses that were obfuscated since some of the scammers used tricks that the functions and scripts just could not deal with.

Then when the crawler was created we ran into another issue. That was that some of the mails were anonymized. The user had masked either bitcoin address of the mail address or both. This was done by xxxx or by replacing it with ”(bitcoin address)” and such. This made it hard since we could not just search for addresses. We never really found a good solution to this which can be seen if you look in the actual files saved that we got a bit of false positives due to not being able to search for the bitcoin addresses explicitly. We also lost some to false negatives due to this.

Then another thing we noticed was that a lot of the messages were very similar. As said in the result the most common message had 98 occurrences and that there were 100s of unique mails. This is somewhat false due to this similarity. There are a lot of emails where a few words differ. Could be down to the amount of money or changing ”last week” to ”last Sunday” and such things. This made it register as unique for us but is not really. So in actuality there are probably only about 100s of unique mails with a bit of higher occurrences than we could report.

Another thing is our own dataset. While it feels big to say that we collected tens of thousands of addresses and hundred thousands of mails in reality it is just a very small subset of all of them floating out there in someone’s inbox.The amount of addresses we have is around $10^{-52}\%$ of all possible addresses. Of course not all of these are active or even used in these scams but it still shows just how little of an amount we have collected.

Then we reach our analysis of our analysis. There was really only one interesting part there and that is the spike in mails in

April 2020. Why is it there and what happened? Sadly there is little we can do to find the answer to this, other than speculate. Could have been because of the corona virus making everyone stay home meaning the scammers do not have anything better to do and more targets are going to be home at their computer. It could also be something to do with April fools though then the trend should have been seen in previous years as well which it did not.

Finally we will analyze our sources. Did we use reliable ones and why do we think they are reliable? It depends. Are the sources reliable for our purpose in the report? Yes. Are they for the dataset? Partly. The sources as far as the report goes are reliable due to the fact that it is mostly the homepages of the websites we talk about and the actual documentation of the APIs used. Sure what web scraping is comes from a blog but it is supported by many more. W3schools is a quite famous site to help you understand different classes and how to use them. They would not be nearly as popular or famous if what they taught was not accurate.

But for the dataset we used Reddit and Bitcoin Abuse. Now Bitcoin Abuse is quite a good source since it is an accumulation of reports from people all over the world. Sure their categorizing might be a bit ambiguous since you do not know how well a user knows what sextortion is or if they just picked a category that sounded good. But it is still metadata based on multiple reports.

But Reddit on the other hand had much less actual emails and more comments around the mails. There was also rules saying that posts should be anonymized which makes the validity of the addresses collected suspicious even though the mails are still good.

VI. Conclusions

Are the amount of threats in extortion/scam mails increasing?
According to our analysis together with the data given by Bitcoin Abuse the reports, and mails, seems to be increasing.

What percentage of the mails are using threats?
Based on our analysis we did not collect any mails that do not contain threats since this was the main purpose of this report. Therefore the results of 100% are inconclusive.

What is the most common type of threat?
When it comes to scam mails with bitcoin addresses the most common threat seems to be leakage of personal and sensitive data although in many cases these threats are taken out of thin air based on trying to message people with information that the receiver identifies to be correct. For example mentioning a common password that the receiver might be using, making the receiver think that their information is leaked or mentioning a website that the receiver might have visited. The scammer is trying to make the receiver think that their information is leaked even though it might not be.

How can a web crawler be designed for the purpose of finding scam email patterns?
The report found that picking a forum to crawl and design a crawler for that page can be done using Scrapy and different APIs. The crawler developed in this project could be made to collect a lot of data by making use of the standard way that Bitcoin Abuse present their data on reported addresses and also collect data from Reddit using their API.

Acknowledgment

We would like to thank Niklas Carlsson for the support and dedication in this project even though the course had to be restructured for distance mode during second period of spring semester.

References

[1] Jack Schofield. *"I got a phishing email that tried to blackmail me – what should I do?"* URL: https://www.theguardian.com/technology/askjack/2019/jan/17/phishing-email-blackmail-sextortion-webcam [Visited 2020-04-01]

[2] Ipsos. *"SURVEY ON "SCAMS AND FRAUD EXPERIENCED BY CONSUMERS" Final Report, p.18"*, URL: https://ec.europa.eu/info/sites/info/files/aid_development_cooperation_fundamental_rights/ensuring_aid_effectiveness/documents/survey_on_scams_and_fraud_experienced_by_consumers_-_final_report.pdf [Visited 2020-04-30]

[3] Satoshi Nakamoto. *"Bitcoin: A Peer-to-Peer Electronic Cash System"* URL: https://bitcoin.org/bitcoin.pdf [Visited 2020-04-20]

[4] Scrapinghub. *"What is web scraping?"* URL: https://scrapinghub.com/what-is-web-scraping [Visited 2020-04-20]

[5] Scrapy developers. *"Scrapy at a glance"* URL: https://docs.scrapy.org/en/latest/intro/overview.html [Visited 2020-04-20]

[6] Scrapy developers. *"Architecture overview"* URL: https://docs.scrapy.org/en/latest/topics/architecture.html [Visited 2020-04-27]

[7] W3Schools. *"CSS Selector Reference"* URL: https://www.w3schools.com/cssref/css_selectors.asp [Visited 2020-04-20]

[8] Andrew Cantino et al. *"SelectorGadget: point and click CSS selectors"* URL: https://selectorgadget.com/ [Visited 2020-04-21]

[9] Reddit. URL: https://www.reddit.com/ [Visited 2020-04-21]

[10] Bitcoin Abuse. *"Bitcoin Abuse Database"* URL: https://www.bitcoinabuse.com/ [Visited 2020-04-30]

[11] Reddit. *"Reddit API documentation"* URL: https://www.reddit.com/dev/api/ [Visited 2020-04-21]