A tight scrape: methodological approaches to cybercrime research data collection in adversarial environments

Kieron Turk
University Of Cambridge
Cambridge, United Kingdom
kst36@cam.ac.uk

Sergio Pastrana
Computer Science and Engineering Department
Universidad Carlos III de Madrid
Leganes, Spain
Sergio.Pastrana@uc3m.es

Ben Collier
Cambridge Cybercrime Center
University of Cambridge
Cambridge, United Kingdom
ben.collier@cl.cam.ac.uk

***Abstract*—We outline in this article a study of 'adversarial scraping' for academic research, which involves the collection of data from websites that implement defences against traditional web scraping tools. Although this is primarily a research methods article, it also constitutes a valuable systematic accounting of the different defensive techniques used by the administrators of illicit online services. Some of these administrators intentionally implement functionality which attempts to prevent web scrapers from gathering data from their site, and some will unintentionally design their sites in ways that make data gathering harder. This is of particular importance for criminological research, where websites such as cryptomarkets and underground forums are publicly available (and hence there is an ethical case for data collection), but the illicit activity involved means that the administrators of these services limit scraping. We classify different anti-crawling techniques taken by websites and outline our developed countermeasures. Based on this, we evaluate which of these methods do and do not succeed at preventing data gathering from a website, as well as those which impact the scraper but do not necessarily prevent the data from being obtained. We find that there are some defences that, if used together, might thwart scraping. There are also a series of defences that are successful at slowing down scrapers, making historical scraping more difficult. On the other hand, we show that many defences are easy to work around and do not impact scraping.**

***Index Terms*—web crawling, web scraping, underground forums, chat channels, cybercrime**

1. Introduction

Empirical research often requires data to be gathered from various online sources and stored offline to provide easier access for researchers. This allows both for the archiving of historical data where sources may become unavailable, and to facilitate data analysis. Gathering online data is generally done through web crawling and scraping, where an automated program visits and retrieves data from websites without human interaction. This semi-automatic process collects and stores data from multiple online sources in a format that is easier to work with. As such, the creation and maintenance of these scrapers are fast becoming core skills in data collection for the social sciences.

Many illicit web services of interest to researchers are hosted publicly on the Internet and can be freely accessed. Scraping these sites and storing them in a research database is a time-consuming task, since it is required to develop and maintain custom crawlers, create accounts for register-only sites, and bypass technical barriers placed by site administrators. Thus, having access to such data is of great interest to researchers, which otherwise would need to spend various weeks designing and running their own crawlers. This also permits greater collaboration with other research groups and ensures the reproducibility of results as researchers can be certain that they are working from the comparable datasets.

Existing projects that provide scraped data have been used in a variety of projects. For example, data crawled from different cybercrime-related online communities has been released publicly for academic researchers, including underground forums [24], [20], [1], chan boards [18], or Darknet Markets [4], [25]. Crawled data have also been used by journalists, such as Unicorn Riot's repository of Discord channels owned by far-right groups [21], which are increasingly organising on public online platforms.

Existing available datasets run the risk of becoming outdated as new posts are added. Also, the ecosystem of online communities related to cybercrime is ephemeral, with new sites starting and ceasing operations rapidly (e.g. due to police takedowns [12]) Thus, researchers often need to develop their own crawling operation, which starts from the development of the crawler and finishes either when the required data is gathered, or when the site goes offline. However, especially for research on clandestine communities, the administrators of some websites do not want their site to be scraped, and so will implement defences to restrict access to human users only. These involve both explicit technical design decisions (such as requiring Captchas or inclusion of trap links) and human intervention (such as scrutinising logs of user activity for suspicious behaviours). Other websites may not intentionally defend against scraping but will have features in the design of the site that impact the scraping. The prevention methods are designed to impact scrapers in different ways. Some methods try to protect the site by preventing access, whereas others aim to slow-down the process. They may attempt to make data gathering difficult, or to identify and attack web scrapers on their site. These methods have varying success at stopping certain types of web scrapers or crawlers from accessing the website, or from gathering its content. Existing works have provided tips and insights into how to develop crawlers for adversarial environments [10], [3], [19]. However, these typically lack a proper analysis of the impact incurred by anti-crawling techniques and the level of difficulty to bypass them.

Contributions In this paper, we set out a comprehensive accounting of the different anti-crawling defences which we have observed 'in the wild' while developing web crawlers and scrapers, and present a catalogue of these defences. Concretely, we have been crawling various cybercrime communities for more than four years, including web forums, chat and instant messaging channels, and content sharing platforms *Citation blinded for review*. We classify the anti-crawling techniques into distinct categories and provide details of the different strategies adopted to bypass them. We then evaluate the success of each defensive measure, based on the effectiveness of the defence in preventing access to the site, delaying access to the site, and the ease of countering the defence.

We believe that our analysis will be a useful methodological resource for those designing web crawlers and scrapers for adversarial environments, which we hope will support research by cybercrime researchers of different disciplines trying to better understand online crime and harm.

Organization In section 2 of this paper, we provide background information on web scraping. In section 3, we describe our methods and discuss the ethical considerations of data gathering. Sections 4 and 5 detail the methods we have seen in use by websites and the countermeasures we have used to work around them. Section 6 then evaluates the success of these defences.

2. Background

2.1. Web crawling and scraping

Web scraping is the use of software that automatically gathers wanted information from a website, normalizes it into a format that is useful for the intended purpose, and stores in an offline database for later use [14]. An inherent component of web scraping is a web crawler, which navigates through websites, and visits all links of interest. Typically, crawling is done by either requesting the HTML page and then parsing the raw result, or by using frameworks that automatize the use of regular web browsers (e.g. Selenium [16] or Puppeteer [27]). While the latter approach is more complex and consumes more resources, it has the benefits that all the browser features can be used by the crawler, like the use of JavaScript.

Developing a web scraper typically involves four main processes:

1) **Accessing the site**. The crawler needs to create a connection to the site, bypassing potential barriers and using adequate credentials if registration is needed (see Sect. 4.1)
2) **Navigating the site**. The crawler needs to be able to access particular pages of interest, e.g. by following specific links or to scroll down a web page to load further content. In the case of adversarial settings, i.e. when administrators do not want their site to be crawled, the navigation should be done carefully to prevent detection and avoid following malicious links (see Sect. 4.2).
3) **Loading the page**. Once a particular link is visited, the contents of the page are loaded. While this is straightforward for a human user, web crawlers must deal with challenges aimed at thwarting bots, such as rate limiters or DDoS protection services (see Sect. 4.3).
4) **Gathering the data**. Once a page is loaded, a crawler will search for all links which are required to follow, and a scraper will search and fetch the specific data (e.g. looking for specific HTML tags or content).

Once the crawler accesses the site, it then cycles between navigation, loading, and data gathering as it traverses the site and downloads the data. However, adversaries (usually the administrators of the website to be scraped) can deploy anti-scraping measures to interfere with each of these stages of the scraping process. In Sect. 4, we describe in depth the anti-scraping protections relevant at each of these stages, and in Sect. 5, we describe our mitigations.

2.2. Forums and Chat Channels

Scraping web forums presents a different set of challenges than scraping chat channels. The former requires visiting a large number of pages covering all posts within each thread, and within each board on the site. Admins can restrict access to the site or particular sections. Additionally, they might monitor the crawling activity alongside other information to identify and ban bots. However, doing manual inspection requires human resources. In some communities (typically large), admins are organized into teams to do this, but this is not the common case. In general, gathering data from forums is most commonly prevented by technical measures rather than manual intervention by site admins.

When scraping chat channels, the crawler connects to each chat room and collects all the information ever posted to the conversation. These applications tend to be less bot friendly, and often unintentionally create problems for bots with user-experience oriented updates. The participation of the bot accounts in conversation (or lack thereof) is much more likely to be scrutinised on chat platforms than it is on forums, as each community moderates itself. This implies that researchers should focus on bypassing detection by humans (i.e. channel/site administrators) rather than working around technical measures.

2.3. Scraping through the Tor network

While most web services are accessible by scrapers through regular web browsers, these services can also be hosted through the Tor network, a global infrastructure of servers (or ‘relays’) that bounce encrypted user signals between them in order to provide very robust security and anonymity protections. Websites hosted through the Tor network, called Onion Services, use the .onion top-level domain, and can only be accessed through Tor. As such, should we wish to scrape these sites, we must route our scraper through Tor. There are further impacts on our scraper when accessing these sites: for example, on Tor, the expectation is that all users will have JavaScript disabled, and hence websites will not use JavaScript on their site - with a common exception being an alert to tell users with JavaScript enabled to disable it. The Tor network constitutes important public infrastructure in its own right and is relied on by a range of users around the world for security and privacy online. As a result, special care should be taken when using Tor for web scraping not to put undue strain on the network. As a result, scrapers should only open a limited number of connections at a time via Tor, and researchers conducting research on Onion Services should consider running a Tor node themselves at their institution to contribute capacity to the network (or donating to Tor relay operator organisations). For planned research which makes extensive or high-traffic use of the Tor network, we advise contacting the Tor Research Safety board at: https://research.torproject.org/safetyboard . While Tor can be useful for scraping sites hosted on Onion Services, it is facile for forums to block access from the Tor network, and so, as shown in Table 1, it is not recommended as a first port of call for scraping clearnet sites.

2.4. Related works

Several systems have been proposed for web scraping. There are systems which attempt to intelligently learn a site's structure [14], [22] and others which rely on manual configuration [2]. Some systems are designed for adversarial environments, such as CARONTE: a semi-automatic tool that maintains a low profile while navigating websites, acting in a similar fashion to a human to avoid being identified as a scraper [6].

We find some sources advising on methods that can be used by a website to defend against web scrapers. Fu et al. provided insights on how to crawl web forums that are not easily accessible through regular search engines [10]. Benjamin et al. present a list of particular challenges to crawl hacker forums [3]. These include gaining accessibility (solving Captcha, bypassing DDoS protection and registering to the community), avoiding detection (prevent IP blacklisting and user-agent check) and accessing private content (paying fees and gaining reputation within these communities). In our work, we discuss similar defences and also present new ones not observed by previous works.

Also, non-academic work in the form of white papers or online blogs have addressed this problem. These describe systems which attempt to defend a generic website [13], as well as specific defences that can be employed by a given site [7], [15]. Other sources discuss classes of countermeasures without specifying concrete versions that could be added to a given website [30]. Furthermore, some sources advise on how to avoid being detected as a scraper [28], [11]. These provide some valuable methods to avoid detection, helping scraper writers to avoid being banned. These sources generally focus on theoretically possible defences against scraping, and few discuss ways to work around the defences. In this paper, we account for the methods actively being used to defend against scrapers, as well as the countermeasures we have implemented to defeat them.

3. Methods and ethics

We have collected this data through the process of designing our own scraping systems for use in three main projects: *Content blinded for review*. We discuss the methods which we see in use on the sites we scrape and provide the countermeasures that we found to be effective in combating these defences. We have scraped 26 forums (described in Table 1), around 300 chat channels across Discord and Telegram, and an archive of files.

There are assorted ethical issues with gathering and analysing data for research from underground forums, especially in the case where preventative measures are in place. In the remainder of this section, we analyse the potential ethical and legal issues arising from this practice. Web scraping creates a record of data that is generally publicly visible. These public channels and forums are important parts of the broader cybercrime economy, and several high-profile developments in cybercrime activities can be traced back to beginnings in public underground communities [19]. Thus, data from these sites constitutes an important resource for academic researchers and security practitioners seeking to better understand online crime, and therefore there is a clear social benefit in (cautiously and ethically) collecting these data. As these data are already public, copying this data to a research database does not increase the amount of data in the public realm. Also, given the public nature of these forums, it is unlikely that people providing this data did so in any expectation of privacy.

However, the collection, processing, and analysis of these data may reveal new relationships, behaviours, and characteristics about groups and individuals which are not evident in the 'raw' data (and this is often the explicit purpose of qualitative or quantitative analysis). Hence, care should be taken to ensure that the outputs of these processes do not reveal or draw attention to hidden aspects of these communities, where there is a clear risk of harm to members. Equally, referring to particular communities or individuals in outputs, or using easily-identifiable quotes may bring undue attention to particular communities or individuals, and hence make them targets for law enforcement or competitors. Finally, simply writing about a community or type of activity may well artificially increase the perceived seriousness of the crime problem which it poses to an unwarranted degree [5]. As a result, we strongly advise both robust anonymisation of individual outputs and quotes, and careful consideration of the broader salience of the research, who will read it, and how it will be read.

The subjects of this data cannot explicitly consent to their data being collected. In particular, although this data is publicly available, the adversarial environment means that we can also safely assume that site administrators actively do not want researchers scraping their websites, which presents additional ethical challenges to simply collecting publicly available data. In the ethics statement of the British Society of Criminology [17], it is stated that "Covert research may be allowed where the ends might be thought to justify the means". The statement further mentions that informed consent may not be required when the dataset is collected from the internet and is publicly available. From a legal perspective, these collections are generally compliant with relevant legislation (such as GDPR in Europe) as there is a clear public interest in their collection.

The Menlo Report [9] and its companion [8] are the primary reference on ethical practice in ICT research. In the report, it is recommended that the Research Ethics Board (REB) must protect the interests of individuals in cases where gaining consent is not possible. Research involving data of illicit origin would need to have a clear benefit to society, and simply because data is public does not exempt research using such data from obtaining REB approval as it may contain personally identifiable information [23]. Our REB has granted us approval for the collection of data, however additional approval is required for each specific subsequent project which analyses or processes the gathered data. We argue, in accordance with existing research [29], that collection and sharing of this data, if carefully handled and considered, poses a sufficient social benefit to outweigh the costs of collection.

The sharing of these data sets is beneficial for cybercrime research and data science, however a key consideration is privacy protection, as it is likely that data collected by scrapers is not intended for research purposes and may not be anonymised. We make all the data collected by our scrapers available to academic researchers under legal agreements, however (in the interests of preserving our implementations of the mitigations we describe in this paper and preventing websites from breaking our scrapers), we do not provide the source code for our scrapers open-source, rather, we make this available to researchers on request.

4. Techniques to prevent crawling and scraping

In this section, we detail the defences used by websites to prevent automated data gathering from their websites. Using the four-stage process model of web scraping described in

TABLE 1. DESCRIPTION OF THE CRAWLED FORUMS

Site	#Threads	#Posts	#Members	Language	Toolkit	Registration?	*VIP* content?	Captchas?	Onion?	Blocks TOR?	Bans
F1	4.0M	41.7M	623.6K	English	MyBB	✓	✗	✓	✗	✓	✓
F2	3.4K	25.8K	1.5K	English	phpBB	✗	✗	✗	✗	✗	✗
F3	11.3K	88.8K	8.2K	English	SMF	✓	✗	✓	✓	✗	✗
F4	119.3K	161.5K	11.5K	English	MyBB	✗	✓	✗	✗	✗	✗
F5	643.6K	8.1M	258.5K	English	XenForo	✗	✗	✗	✗	✗	✗
F6	2.1K	7.7K	872	English	vBulletin	✗	✗	✗	✗	✗	✗
F7	34.8K	214.9K	44.3K	English	MyBB	✓	✓	✓	✗	✗	✗
F8	767.3K	9.4M	477.5K	English	vBulletin	✓	✓	✓	✗	✗	✓
F9	708	7.1K	764	English	MyBB	✗	✗	✗	✗	✗	✗
F10	456.3K	2.5M	50.8K	English	MyBB	✓	✓	✓	✗	✓	✗
F11	1.6K	9.2K	728	English	MyBB	✓	✗	✗	✗	✓	✗
F12	243.2K	2.4M	77.9K	Russian	XenForo	✗	✗	?	✗	✗	✗
F13	155.5K	3.5M	508.7K	English	?	✓	✓	✓	✗	✗	✗
F14	244.8K	3.6M	39.2K	English	MyBB	✓	✓	✗	✗	✗	✓
F15	577.6K	6.2M	126.9K	Russian	XenForo	✗	✗	✗	✗	✗	✗
F16	1.6K	6.2K	1.1K	Russian	vBulletin	✓	✗	?	✓	✗	✗
F17	13.0K	27.0K	7.4K	English	vBulletin	✗	✓	✗	✗	✗	✗
F18	419.7K	8.4M	120.5K	Russian	?	✗	✗	✗	✗	✗	✗
F19	75.1K	294.6K	47.5K	English	?	✗	✓	✓	✓	✗	✗
F20	4.3K	28.5K	3.8K	English	phpBB	✓	✗	✓	✓	✗	✗
F21	454	2.2K	361	English	SMF	✓	✗	✓	✓	✗	✗
F22	10.7K	59.7K	8.3K	English	MyBB	✓	✗	✗	✓	✗	✗
F23	3.6K	15.6K	1.7K	German	MyBB	✓	✗	✗	✓	✗	✗
F24	16.9K	240.6K	17.2K	Russian	?	✓	✗	✗	✓	✗	✗
F25	78.1K	276.7K	81.5K	English	?	✓	✓	✓	✗	✓	✗
F26	120.5K	1.9M	169.6K	English	vBulletin	✓	✗	✓	✗	✗	✗

Sect. 2, we have identified a range of anti-scraping defences that can interfere with each of the stages of this process.

4.1. Accessing The Site

Websites might restrict access to the bulk of the site for users who are not logged in, thus requiring crawlers to use registered accounts and to log in. In particular, 16 out of 26 of the forums we crawled (61.5%) required to register accounts. Some will not restrict all access; instead, they will hide parts of the contents (e.g. attachments or links) to encourage registration, and cause scrapers to gather incomplete information. Even if registered, some sites contain private content, only available for upgraded or *VIP* members (i.e., requiring users to have a certain reputation or to pay a fee). While this is typically reduced proportion of the entire site, 9 forums (34.6%) in our dataset contain private content.

Access Restrictions During registration, many websites will attempt to prevent robots from registering for their site so that they cannot log in. To prevent this, it is often required to provide contact information such as email address or phone number which must then be validated to complete registration. Furthermore, sites may prevent or dissuade unwanted users by making their site invite-only, or by requiring payment to register each account. A special case is a closed Russian forum which gives the option of a paid account for $100 or a free account which requires an application demonstrating extensive experience in hacking technologies or services.

Login To log in to the website, web crawlers may fill out and submit the login form and then store the cookies returned. To prevent automated login, some sites will require a Captcha to be filled out in the login form. Others may look for robot-like behaviours when filling in the login form, and either prevent access to suspected robots or force them to complete a Captcha due to their behaviour (see discussion about Captchas below). The actions observed which can trigger these sites include submitting a form too fast after loading it, completing different fields of a form too fast, or typing too quickly. Additionally, sites might provide optional or enforced Two-factor authentication (2FA). This requires multiple methods for authentication like sending a one time code to the user's known email address, phone number, authenticated app, or requiring another active session to authorize the new login. Some darknet sites which are focused on anonymity will create their own versions of 2FA that do not require personal information (e.g. one forum provides a set of 20 numbered codes on registration, and will ask for a random one on each new login). This causes problems when attempting to automate the login process, as the additional verification must also be automated.

Some websites will prevent logins that do not use a JavaScript-enabled browser by requiring the main page to be loaded through the execution of an embedded JavaScript from the login page. This implies that scrapers not using regular browsers (i.e. with JavaScript enabled) cannot access their site.

Alternatively, web crawlers can use the cookies from a prior browsing session, for example from a session where a human logged in. To prevent reuse of cookies, some web pages will force cookies to expire after a short time, such that the user must log in again frequently. However, this deteriorates the user experience, and in our experience we have observed sites with expiration times ranging from one hour to one week. Many sites have cookies that never expire.

Captcha Many sites make use of Captchas as a method for determining if a user is a robot (as shown in Table 1, 40% of the forums crawled use it). The vast majority of websites will have a Captcha present on the registration page, and many will include one on login as well. Nearly all sites on the surface web employ Google's reCaptcha, which has a very high reputation and is good at preventing robots from accessing a website, although pay-to-solve services also exist. On the dark web, it is standard for JavaScript to be disabled, and hence dark web sites will never use reCaptcha - instead, many sites will create their own Captchas, with varying success. The majority of these Captchas such as those in Figure 1 ask to transcribe the letters or numbers in an image. Some less common Captchas are used in other sites (see Figure 2). In this forum, a simple image is overlaid with three offset rows of shapes, some of which are filled in with the remainder simple outlines. The user then selects which shapes are filled in on the 3x3 grid opposite.

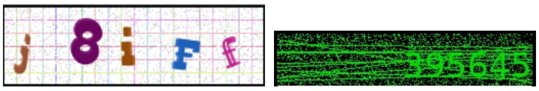


Figure 1. Examples of text Captchas being used by underground forums

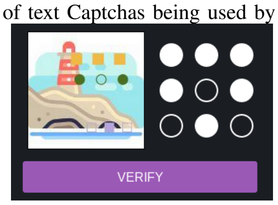


Figure 2. Example of unique Captcha used by one of the forums

4.2. Navigating the site

In order to gather all the historical data of a community, it is needed to visit all the pages, including old content. This involves suspicious behaviour that can be easily detected by site admins, which can ban the account or IP being used, halting the scraper progress. Additionally, there are other measures that thwart crawling, even if they are not directly implemented for such purpose.

Redirects Websites can present multiple variants of redirects. Those affecting web scraping allow the web page to load initially (such that the scraper starts to gather data) but suddenly redirect away from the page. This will cause errors as the scraper no longer has access to the data it is expecting to be available.

Malicious Links On some sites, there are links which are not intended to be clicked by humans, but will be accessed by automated crawlers. The most common behaviour on one such site is to redirect to an otherwise unused page on the site, apply measures intended for DDoS protection, and then redirect back to the homepage of the site. The homepage will now run a script, which performs some unknown checks that appear to determine if the user is a robot. If it determines that the user is a robot, a function is called which will endlessly call itself, causing the call stack to overflow after a short time and crashing the browser.

We have observed a similar feature in the content delivered by other websites, which are intended to detect and remove bots. An example is shown in Figure 3, where the website shows a message at the beginning of the HTML with a link that directly removes the account being used. An unsophisticated crawler which simply follows every link it encounters will fall into the trap, and the account used will be banned or deleted.

Another observed behaviour upon following a crawler-only link is to attempt to trap the scraper in a loop between a pair of pages, each redirecting to the other. The malicious link will send the browser to one of these pages, which then causes a redirect loop that some browsers will follow endlessly.

Endless Text Generation When attempting to access a specific directory on one site, a page is returned that runs a simple JavaScript program:

```
while (1) { document.write("lolwhy "); }
```

which will endlessly append text to the web page until the browser crashes. This is hard to stop once it is running, potentially preventing the browser from closing, and as such it is extremely successful at crashing the browser.

Massive Files to overload memory In a particular case, the crawler loaded a CSV file hosted on the website, containing a single line that is incredibly long. Attempting to load this in the browser causes it to crash after it has loaded.

Banning user accounts Websites that are attempting to find bot-like users will often look at their browsing patterns to find suspicious behaviour: a user who rapidly navigates many pages without pause is likely to not be interacting with the content on any of the pages, and hence is likely to be a robot. Furthermore, a user who navigates all pages of the site in a predictable order is likely to be a robot. Users that move between pages without any link are suspicious and if they frequent this behaviour may get banned. In our experience, we have observed a higher ratio of detection while our crawlers were scraping old content (e.g. posted more than 5 or 6 years ago). This is reasonable since probably few or no users are accessing such old information, and a single IP or user account suddenly accessing it constitutes anomalous behaviour. Concretely, 3 of the forums we crawled banned at least one of the accounts we were using.

During our crawling, we have observed discussions of our own data collection (see Table 2). From these discussions, we conclude that the main suspicious patterns are: i) accounts being online 24/7; ii) accounts connecting from different IP addresses; and iii) accounts not actively interacting within the forum. Moreover, forum websites operate trust services providing information related to each account. This trust information can be viewed by all the moderators and certain upgraded accounts. Figure 5 shows an example of the information provided by these services.

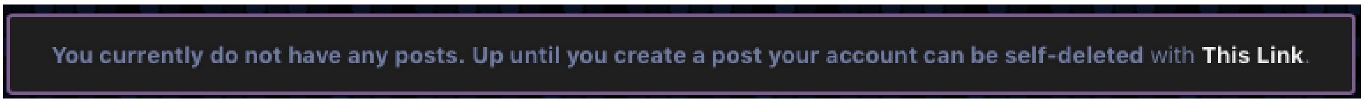
Banning IP addresses Some websites may take a more extreme approach to banning to prevent users from registering new accounts: they can ban the IP address a user used to connect to the website, preventing them from accessing it again (see Figure 4).

Technical Preventions Websites may implement various technical restrictions to prevent robots accessing their site. They may provide various challenges to the browser, which attempt to identify non-human users and restrict their access. These are usually Javascript or cookie-setting requests, which will be automatically handled by any standard browser without user interaction. Websites may also inspect the "User-Agent" header from the HTTP request, which includes information like the version of the browser and the operating system. Most web crawlers will include information here that identifies them as being a crawler, such as the name of the software that is scraping or having "bot" somewhere in the user-agent, giving websites a simple way to prevent access to web scrapers and crawlers.

4.3. Page Loading

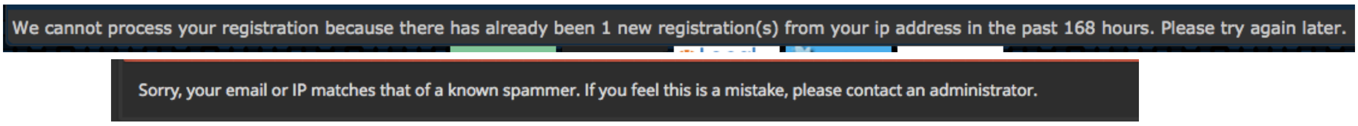
Many websites will implement anti-bot measures when the user attempts to load a page from the site. These often go unnoticed by users, but cause problems for scrapers trying to access the page.

DDoS Protection Services DDoS protection services provide an easy defence against malicious users attempting to overload a web server. The majority of services use commercial services for this, but we note that some websites choose to implement their own versions. We also see some websites that force the user to view multiple protection pages to access a page of the website, attempting to slow down or break scrapers. At the user level, this will appear as a simple web page, a short wait, and then a redirect to the requested content. For a simple web scraper, being shown this page as the response to their request may lead to incorrect data being saved. For a scraper which realises they are looking at a delay page, this will cause a slowdown to the scraper.



You currently do not have any posts. Up until you create a post your account can be self-deleted with This Link.

Figure 3. A trap found in an underground forum, causing ‘dummy’ crawlers to follow a link which will automatically remove the account being used



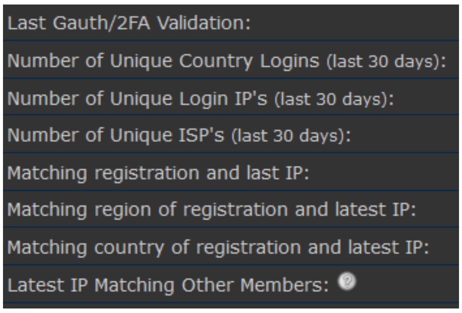
We cannot process your registration because there has already been 1 new registration(s) from your ip address in the past 168 hours. Please try again later.

Sorry, your email or IP matches that of a known spammer. If you feel this is a mistake, please contact an administrator.

Figure 4. Examples of IP banning during registration due to the IP being limited (above) or blacklisted (below)

TABLE 2. EXCERPT OF DISCUSSIONS OBSERVED IN DIFFERENT FORUMS RELATED TO THE ACTIVITIES OF CRAWLERS AND BOTS. DESCRIPTIONS ARE NOT PROVIDED VERBATIM TO PREVENT DIRECT LINKAGE TO THE ORIGINAL CONTENT.

I found something pretty strange, an account that was registered 3 days ago and has not gone online since then. Every time I refresh the profile, it’s accessing a different page/post/profile. I think it’s a crawler that’s archiving everything it stumbles over
This account has been roaming some sections for 24/7, has 0 posts and [...] the account is connected to many different IP addresses. [...] I’m almost sure it’s a bot



Last Gauth/2FA Validation:
Number of Unique Country Logins (last 30 days):
Number of Unique Login IP's (last 30 days):
Number of Unique ISP's (last 30 days):
Matching registration and last IP:
Matching region of registration and latest IP:
Matching country of registration and latest IP:
Latest IP Matching Other Members:

Figure 5. Information used to create the trust report in an underground forum

Rate Limiting Web servers can be configured to track the number of requests made by any specific client over time, and choose to deny access to users who make too many requests in a specified time period by returning 429 - Too Many Requests errors. Many surface web forums use services such as Cloudflare DDoS protection which will also rate limit scrapers.

Loading Content Some websites make use of animations while their site is loading in the background, which may cause problems as unaware scrapers will get a frame of animation instead of the loaded web page and cannot gather data from the page. Web pages frequently show part of the page instantly and then have other content load afterwards, which can lead to scrapers gathering incomplete data. Some content will only be loaded on certain events, such as clicking a button on the page or scrolling past what is currently loaded, which forces the scraper to interact with the web page.

4.4. Data Gathering

Whether intentionally or not, some web pages will implement features that make getting data off of a web page much harder than grabbing the text from an HTML element.

Visibility Gathering data is easier when the content of interest is visible on the page, however there are multiple common features of websites preventing this. Some websites will have pop-ups appear on screen either on load or shortly after load, which occlude large portions of the web page. Some content will only appear when hovered over, such as drop-down menus or hidden additional information. Finally, due to some sites being designed around large displays and then being viewed on smaller displays - or in the case of most web scrapers, no display - there are websites where content will overlap other content due to scaling. Sidebars and headers are common causes of overlap.

Obfuscation Obfuscation is the practice of replacing text with an image, CSS sprite or other formats, so text-based scrapers are unable to retrieve the data. Using this technique explicitly is rarely used due to accessibility and usability issues, since it restricts some functionalities of the site such as content search or reduces the user experience. However, we have observed various users posting images instead of text for advertising their goods or services, in the form of publicity banners. While this is probably not intended to prevent scraping, it is a technique making the gathering of the actual content harder.

Changing Page Layout Web scrapers look at specific places in the structure of a website to find the wanted information. This requires a knowledge of the path to the HTML element. Thus, websites can hinder scrapers by frequently changing the layout of the page. This may involve changing the HTML elements used, the CSS classes of elements, or the layout of the website overall, which is a significant effort from the website creators. However, it is effective since most web scrapers need to be updated to deal with the changes. In our experience, we observed that most changes are due to updates in the software toolkits being used (e.g. MyBB or phpBB for web forums) rather than explicit changes to thwart scraping.

User Attacks on Scrapers User discussion on various forums and channels may impact a scraper attempting to process their posts and messages. These interactions are unlikely to be intentional but may cause problems for scrapers. For example, if our scraper is inserting content into an SQL database without proper input sanitisation, we may find that a discussion on SQL injection techniques successfully interferes with the scraper database.

5. Countermeasures

5.1. Accessing The Site

Access Restrictions For sites that use techniques to prevent automated registration (i.e. bots), we register accounts manually. This way we can bypass most of the issues faced, such as Captchas. In cases where an email address must be provided, we either provide a fake address (if no confirmation is required) or use disposable addresses.

In some cases, such disposable emails are banned, and so we set up email addresses from regular providers (e.g. Google or Yahoo). Creating these accounts is time-consuming since it involves a manual process. However, the emails can be reused for registration across different online sites.

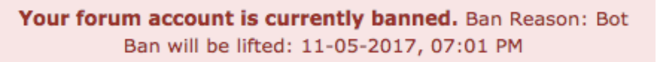
Login We can then automate the login process through the browser, by submitting the login form. If the browser looks for unusual behaviour in filling out this form, we add delays between each of the interactions, and if needed, we split up the typing to add a small delay between keys. Mimicking human typing is an effective mechanism to bypass these defences. After logging in, we should check on each page load that the login session remains by checking for the presence of tokens indicating that the user is logged in (e.g. the *log out* button or the *personal settings* menu). If we have been logged out we repeat the automated login, then continue scraping. If 2FA is required, we must work around it on a case by case basis. In the case of the forum requesting one of the 20 verification codes given on registration, we store the codes offline and scrape the index of the code required.

Captcha Some login forms require solving Captchas. In these cases, as with the registration, we either use a Captcha solver (if possible) or log in once manually, storing and reusing session cookies for later (in those cases, it is needed to enable the *'Remember me'* option commonly featured in login-based systems). The scraper then loads these cookies during subsequent crawls, and can login without filling in the login form again. When using this method, we must replace the cookies anytime they expire, which requires human interaction. If this is too frequent, we should try to automate the login process instead. However, it is unlikely that sites have short expiration times since this decreases the users' experience when navigating the site. In our experience, the worst cases have one hour of expiration time. However, in this case, the site was not asking for Captcha solving, so we were able to automatically fill the logging form and get the session cookie. The worst-case scenario was a forum where cookies' expiration times were of just one week, and it also required solving Captcha. In that case, we needed to manually solve the Captcha once a week. However, the resulting session cookie for Captcha clearance could be reused across all the different accounts that were crawling the site in parallel.[1]

If Captchas are provided during the regular navigation of a site (i.e., not during registration or login), they need to be dealt with on a case by case basis: some potential methods we found useful are to reload the page, navigate away and then come back, or use a new browser instance to access the page. Moreover, if we are aware of the actions that lead to Captchas being served (e.g. high network traffic), we can adapt the crawling to avoid the problem. During our research, these cases were very few, and indeed they occurred while the site was being targeted by a DDoS attack.

5.2. Navigating the site

Redirects If we find we are being redirected several times, i.e. reaching many intermediate pages after, we download each page source before redirection occurs, so the information is available from the saved copy of the data for offline processing. This requires an additional scraper that operates on the raw HTML offline.



Your forum account is currently banned. Ban Reason: Bot
Ban will be lifted: 11-05-2017, 07:01 PM

Figure 6. Message received in one of the forums due to our account being banned

Malicious Links and Files Some sites purposely provide content aimed at thwarting bots by attacking the browser. If we expect such behaviour, like the exposure of malicious links, the scrapers must be designed to be resilient such that they recover from a successful attack. We do not want to visit pages that attack the browser, but we cannot know which pages do so unless we have visited them before. As such, we should track previously seen "dangerous links", and avoid navigating to them.

Identifying these dangerous links can be done by looking for impacts to scraping. In the extreme case, some attacks observed will crash the browser, and so if the browser generates an exception or otherwise fails at any time we can log the link that caused the failure as dangerous. If we are aware of more specific attacks, the system can identify each attack individually to become more resistant to attacks. For example, with the script shown before that dynamically generates infinite content until it crashes the browser, we analyse the page size before and after an interval to check if it has grown unexpectedly. Other attacks can be identified through similarly simple checks, such as ensuring that a redirect link has not been visited recently, as this may imply it is a loop. These heuristic checks are tailored to specific attacks, and investigating each of the dangerous links discovered by the scraper is the optimal method to identify the new attack and mitigate it in the future. If manual intervention is to be avoided, being able to detect and recover from an attack - for example, by creating a new browser instance - is sufficient.

If it is not needed, we disable JavaScript, as this is the best defence we can use against malicious scripts. However, when crawling sites out of the Tor network, JavaScript typically must be enabled, as it is needed to solve challenges or to use any other dependant feature.

Banning Accounts and IP addresses Bot-based behaviours typically exhibit patterns that lead to its detection, which results in the account being banned, either temporarily or permanently (see Figure 6). For this purpose, we create different human-like profiles which can be adapted for the different bots. These profiles are configurable and include techniques such as adding random delays between requests, navigating around the site using the links on the site (as opposed to jumping directly to a link which is not present in the currently loaded page) or adapting the delays to the size of textual content shown in the page. The different configurations of these profiles result in different trade-offs between crawling speed and risk of being detected: stealthier crawlers are slower and vice-versa. For example, our initial strategy for forum F1 was to change the configuration of all the running crawlers whenever one of them was banned, so the rest became stealthier. However, our experience showed that the most suspicious activity related to bots is not due to navigation patterns, but due to the content accessed being too old (indeed, our crawlers were more frequently banned whenever they were accessing content dating back 5 or 6 years). We believe that this is due to the server using monitoring tools that highlight whenever a substantial amount of traffic is requesting old content, which is rarely accessed otherwise. Thus, our strategy for that site was to

[1] We note that cookie reuse could be potentially harmful due to tracking of the different accounts. In such a case, the detection and banning of one of them could result in the others being banned. While this is an actual risk, during our crawling operations we have reused cookies several times and only in one case were various accounts banned within a few hours of each other. However, as we discuss in Sect. 5.2, we believe this is was due to theses crawlers visiting old content frequently.

set extremely slow and stealthy crawlers for this old content (around 1 request/minute), while using faster crawlers for recent content (published less than 1 year ago).

Additionally, we keep the HTTP *User-Agent* header used during registration of an account, and always use those related to commonly used browsers (e.g. Google Chrome, Mozilla Firefox, or Safari). If we find out that one account has been banned (even temporarily), we cease its operations and create a new one (using a different email and IP). Banning IPs is less common, and often related to the use of IPs that are blacklisted, typically due to them being Tor exit nodes. We use our own proxies distributed across the globe to conduct the crawling.

5.3. Page Loading

The use of a browser-based scraper prevents many problems related to page loading, since many websites will aim to filter out other non-browser tools such as the *curl* and *wget* commands.

Challenges and DDoS Protection In particular, JavaScript challenges and cookie challenges will have no impact on any user or scraper with a browser (provided it supports JavaScript and this is enabled). For example, sites protected by Cloudflare might conduct an up-to 5 second browser check requiring the execution of a JavaScript function to prevent DDoS attacks [26]. In these cases, the scrapers check for particular text in the loaded page after fetching an URL, and keeps waiting for the actual page being rendered (i.e. checking when that particular text is no longer present).

Rate Limiting Another issue is when our IP is rate limited. In these cases, we automatically reduce the rate of the crawlers such that we do not reach the limit in use. This slows down the overall process, which is partially compensated by using various proxies and conduct parallel crawls.

Loading Content When a page renders content a while after the main page is loaded, the scrapers must wait for this content to appear on the page. This is done by frequently checking if the content is present (to prevent halting, whenever a timeout is reached an error message is triggered). If we must interact with the page for specific content to load, we will need to use the browser to provide the required interactions and cause the content to load.

5.4. Data Gathering

Visibility To handle issues with content visibility, we download the page source after loading and then create a scraper that gathers data from the HTML content, which will be able to access data that is not visible or that is occluded on the screen. Alternatively, we can call JavaScript through our browser to scrape the raw HTML, for example using query selectors.

Obfuscation If a site uses obfuscation on data that we wish to gather, we can either download the image or take a screenshot of the data, and then run it through OCR software to gather the data from it.

Changing Page Layout Whenever the website changes its layout, we will have to modify our scraper to handle these changes. We can try to design our scraper to be robust to these changes. For example, we can look for some label text and gather the required information from the rest of this block of text. However, this design is more complex and not always possible, and so regular changes to the site structure might be a good approach to interfere with web scrapers.

User attacks User attacks are often unintentional exploitation of vulnerabilities in the scraping software, such as SQL injection. It is recommended to follow security best practices for any such attack that could apply: for example, if a SQL database is used, all queries should be prepared statements to mitigate the possibility of SQL injection.

6. Discussion

So far we have revisited the different techniques to prevent web crawling and scraping, and the countermeasures that researchers can take to thwart these. Table 3 summarizes the techniques discussed so far in terms of their intent and the impact on crawling. We can observe that some countermeasures are harder to apply than others, and the impact of the defences vary. In this section we discuss and evaluate the effectiveness and robustness of the defences against web scrapers.

Ineffective Defences Methods aimed at preventing bot access typically rely on human interaction (e.g. solving Captchas), after which automated crawling could begin. While these methods prevent large scale crawling, they are ineffective for targeted crawlers where a human operator can spend a small amount of time registering or logging into the site. Even though this process is manual, the human operator can still benefit from an automated program that assists in the process. Some websites, notably *.onion* sites, will use Captchas on their registration forms but nowhere else on the website. This prevents robots from registering but does nothing to prevent a scraper once an account has been made. Creating an automated login for a web scraper requires a small amount of development time, and does not slow down the scraper enough to be effective. The sites we scraped have been observed to log out users at most once an hour, having very little impact overall. Sites that enforce a form of two-factor authentication to be used tend to be sites with easy to automate 2FA, and hence it is often an ineffective defence. Blocking specific user-agents from accessing the website is an ineffective defence against scraping, as browsers allow custom user-agent strings to be used. The attempt at trapping the scraper's browser by sending it into a redirect loop is ineffective with any modern browser, as the browser will detect this behaviour and stop following the loop. Both JavaScript and cookie challenges are ineffective so long as we use a regular browser to access the web pages being scraped, as the browser will handle them automatically.

Partially effective defences Websites employing rate limiting will force the crawler to slowdown the navigation. Similarly, any website which bans users behaving like robots will force the crawler to run slower to mimic human behaviour. In both cases, performance can be improved by running multiple parallel scrapers on the website, each using a different proxy and account. Methods which make data gathering difficult required customized techniques, but are often easy to automate or design around. Handling visibility issues and redirects after load require working on saved copies of HTML increasing the complexity of the scraper. If obfuscation is used, we can make use of OCR technology to retrieve the text, which might affect the completeness of the content. When scraping a website that uses malicious redirects, for example those which will direct to a page that attempts to crash the browser, we need to actively check for redirects and handle them accordingly. Provided we can do so before any malicious script runs, or if we can recover from an attack, we can continue scraping after the attack.

Successful Slowdowns If we have any animations on the website or data that loads over time, and that are of interest, we have no choice but to wait for these resources to load. Similarly, if DDoS protection is being used, the crawlers must wait for it to pass. Any attack on the browser that we are not previously aware of will cause a slowdown since the browser will crash and have to be restarted. This requires tracking these attacks and updating the scraper to handle them, which incurs additional costs. In some cases, we may not be able to automate the login process, for example due to Captchas that we do not have a solver for, or forms of 2FA that cannot be performed through a browser. We will be forced to use manual logins and reuse cookies from this. In some cases, websites will also force these cookies to have short expiry times, such that the scraper will need human interaction frequently to be able to continue working. As such, there is no easy way to keep the scraper logged in, and keeping the scraper running is difficult. When websites change their layout, scrapers need to be updated, which is costly and time-consuming. While sometimes is it possible to design around this, this is a good way to thwart web scrapers. However, as it can be observed from Table 1, many of the sites use similar toolkits. Thus, learning the new layout and updating crawlers is typically transferable across sites. Finally, automated anomaly detection tools can be designed to detect bot-related patterns. In those cases, we were forced to reduce the speed of the crawlers, thus significantly slowing down the process.

6.1. A Note On Onions and chat channels

When accessing websites with Tor - specifically .onion websites - the expectation is that JavaScript is disabled in the browser. As such, onion websites design their site to function without it. This will prevent a large number of the defences from working, as they depend on JavaScript to function, and hence we see that some defences are replaced with older systems and some are removed entirely. Google reCaptcha and JavaScript challenges have to be replaced with older style image Captchas and cookie challenges, despite both of these systems no longer being used elsewhere. DDoS protection pages and JavaScript attacks on the browser will not function, and so neither are not present on onions. Web pages are forced to be static, and so we face no problems with animations, visibility, or asynchronously loaded content. Overall, we find that Onion Services are far easier to scrape than surface sites.

We have additionally been scraping chat channels on two platforms as part of our research. On both platforms, we find that there are very few technical measures employed to stop scrapers - the few that we encountered such as content loading and visibility issues are not intended for this purpose. Instead, we find that the main problem is getting banned from individual channels on each platform due to moderator intervention. Where platforms themselves ban scraping accounts (as scrapers will often be members of a range of illicit services and hence become swept up in clean-ups), developers of these platforms may well have a more sympathetic sensibility towards researchers than the administrators of underground forums. Equally, while Telegram's developer API means that robust scrapers are simple to develop and need little maintenance, scrapers for platforms such as Discord, which lack such a pro-scraping API and have regular UX updates can be easily broken where the page markup changes, and hence need regular maintenance.

7. Conclusions

The collection of research data in 'adversarial' environments is inherently challenging. At each stage within the crawling process, the administrators of these websites have a range of options available to inhibit data collection. While there are ways in which scrapers can be designed to mitigate these problems, there are no easy solutions; where administrators manually comb through the data, they will generally be able to spot the suspicious activity patterns which crawlers generate. Conversely, researchers can always collect data from public sites through manual labour. The general effectiveness of our solutions implies that most websites lack the resources or incentive to engage in much of this manual administration: crawlers are generally banned when they engage in easily-spotted patterned behaviour, such as accessing every post on a website beginning with the oldest. What manual detection exists is generally sporadic - well-designed scrapers will still occasionally be banned, but only at irregular intervals, and only on the larger sites which can afford sizeable admin teams.

Defences which attempt to automatically detect and stop scraping altogether are generally ineffective, however the more effective defences are those which force a slowdown or require human interaction to solve. This is especially true where defences are used in combination: if we have Captchas on login as well as a short session expiry time, we force large amounts of human interaction to scrape the website, potentially making the time to scrape the site infeasible. On the other hand, having one of these without the other can be combated with relative ease. Hence, on both sides of scraper attack/defence, the limiting factor is the level of manual work required, and productive moves generally revolve around undermining the ability of the other side to automate detection or scraping processes effectively.

Finally, we encourage collaboration between research centres and the ethical sharing of collected data. This would provide a means to carry out this cybercrime research without needing to develop scrapers and accomplish the tedious crawling task. For researchers who do have this facility, parsimony around data collection helps the research community in general as a smaller number of scrapers implies fewer defences being put in place. In this spirit, we make our data collections of forum and chat channel data available to other researchers by agreement.

References

[1] Luca Allodi. Economic factors of vulnerability trade and exploitation. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1483–1499. ACM, 2017.

[2] Bissan Audeh, Michel Beigbeder, Antoine Zimmermann, Philippe Jaillon, and Cédric Bousquet. Vigi4med scraper: A framework for web forum structured data extraction and semantic representation. *PloS one*, 12(1), 2017.

[3] Victor Benjamin, Sagar Samtani, and Hsinchun Chen. Conducting large-scale analyses of underground hacker communities. *Cybercrime Through an Interdisciplinary Lens*, 26:56, 2016.

[4] Gwern Branwen, Nicolas Christin, David Dcary-Htu, Rasmus Munksgaard, Andersen, StExo, El Presidente, Anonymous, Daryl Lau, Sohhlz, Delyan Kratunov, Vince Cakic, Van Buskirk, Whom, Michael McKenna, and Sigi Goode. Dark net market archives, 2011-2015. https://www.gwern.net/DNM-archives, July 2015.

[5] Finn Brunton and Gabriella Coleman. Closer to the metal. *Media technologies: Essays on communication, Materiality, and society*, pages 77–97, 2014.

[6] Michele Campobasso, Pavlo Burda, and Luca Allodi. Caronte: crawling adversarial resources over non-trusted, high-profile environments. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 433–442. IEEE, 2019.

TABLE 3. SUMMARY OF THE DEFENCES, THE DEVELOPED COUNTERMEASURES, AND THE IMPACT INCURRED ON THE SCRAPING PROCESS. COLOURS REPRESENT THE SEVERITY OF IMPACT THAT A SUCCESSFUL DEFENCE WOULD HAVE IN CRAWLING (HIGH , MEDIUM , LOW) AND THE EASINESS TO IMPLEMENT COUNTERMEASURES FOR THE DEFENCE (HARD , MEDIUM , EASY)

Defence	Intent	Countermeasure	Impact to crawling
Category: SITE ACCESS			
Registration w/ Captcha	Restrict accounts	Manual Registration	None
Login w/o Captcha	Restrict access	Automated login	None
Login w/ Captcha	Restrict access	Manual login, cookie reuse	Manual intervention required
2FA	Improve Security	Disable if possible ; Workaround if not	Potential intervention
Category: NAVIGATION			
Redirects after load	Advertising	Scrape HTML after initial load	Increase complexity
Malicious redirects	Crash scrapers	Detect redirect and restart browser	Moderate slowdown
Redirect loops	Trap scrapers	Browser handles automatically	None
Honeytokens and Traps	Ban scrapers	Avoid known traps; Adapt to new	Minor updates of scrapers
Endless text generation	Crash browser	Restart browser; Disable JS	Moderate slowdown; JS not featured
Malicious files	Crash browser	Detect actions and restart browser	Moderate slowdown
Rapid navigation	Prevent scraping	Rate limit scraper; Parallel scraping	Moderate slowdown. Operational cost
Indirect navigation	Prevent scraping	Follow links provided on site	Small slowdown
Scraping old content	Prevent scraping	Rate reduction when scraping historical data	Large slowdown
Inhuman page interaction	Prevent scraping	Delay between actions; Reduce interactions	Moderate slowdown
User-agent blocking	Block known crawlers	Set common user-agent header	None
IP Banning	Block known crawlers	Use another proxy	Operational cost
Captcha on page load	Prevent automated traffic	Captcha solver; Avoid triggering Captchas	Increase complexity
Category: PAGE LOADING			
DDoS Protection Services	Prevent high traffic	Recognise pages and wait	Small slowdown
Rate Limiting	Prevent high traffic	Slow down scrapers; Run multiple in parallel	Moderate slowdown. Operational cost
JavaScript/cookie challenges	Prevent automated traffic	Use a browser-based scraper	None
Loading Content	UI/UX, performance	Recognise and wait; Interact where required	Small slowdown.
Category: DATA GATHERING			
Visibility	—	Scrape HTML where possible	Increase complexity
Obfuscation	Protect sensitive data	OCR Software	None
Changes in page layout	Break existing scrapers	Modify scraper to match	Major updates of scrapers
User attacks	—	Follow standard security practices	None

[7] Shaumik Daityari. Protect your site against web scraping. [Online. Last accessed: February 20, 2020], 2017.

[8] Michael Bailey David Dittrich and Erin Kenneally. Applying ethical principles to information and communication technology research: A companion to the menlo report. Technical report, U.S. Department of Homeland Security, 2013.

[9] David Dittrich and Erin Kenneally. The menlo report: Ethical principles guiding information and communication technology research. Technical report, U.S. Department of Homeland Security, 2012.

[10] Tianjun Fu, Ahmed Abbasi, and Hsinchun Chen. A focused crawler for dark web forums. *Journal of the American Society for Information Science and Technology*, 61(6):1213–1231, 2010.

[11] Scrape Hero. How to prevent getting blacklisted while scraping. [Online. Last accessed February 20, 2020], 2014.

[12] Alice Hutchings, Richard Clayton, and Ross Anderson. Taking down websites to prevent crime. In *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–10. IEEE, 2016.

[13] Imperva. Detecting and blocking site scraping attacks. Technical report, Imperva, 2014.

[14] Jingtian Jiang, Xinying Song, Nenghai Yu, and Chin-Yew Lin. Focus: learning to crawl web forums. *IEEE Transactions on knowledge and Data Engineering*, 25(6):1293–1306, 2012.

[15] JonasCz. A guide to preventing webscraping. https://github.com/JonasCz/How-To-Prevent-Scraping, 2016.

[16] Baiju Muthukadan. Selenium with python. http://selenium-python.readthedocs.io.

[17] British Society of Criminology. Statement of ethics. Accessed Oct 6, 2019 from http://www.britsoccrim.org/ethics/, https://perma.cc/K3MY-UG5U, 2015.

[18] A Papasavva, S Zannettou, E De Cristofaro, G Stringhini, and J Blackburn. Raiders of the lost kek: 3.5 years of augmented 4chan posts from the politically incorrect board. In *14th International AAAI Conference on Web and Social Media (ICWSM 2020)*. ICWSM, 2020.

[19] Sergio Pastrana, Alice Hutchings, Andrew Caines, and Paula Buttery. Characterizing eve: Analysing cybercrime actors in a large underground forum. In *Research in Attacks, Intrusions, and Defenses (RAID)*, pages 207–227, Heraklion, Crete, Greece, 2018. Springer.

[20] Rebecca S. Portnoff, Sadia Afroz, Greg Durrett, Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, Damon McCoy, Kirill Levchenko, and Vern Paxson. Tools for automated analysis of cybercriminal markets. In *Proceedings of 26th International World Wide Web conference (WWW)*, 2017.

[21] Unicorn Riot. Discord leaks. https://discordleaks.unicornriot.ninja/discord/, 2017.

[22] Wei Lai Yida Wang Rui Cai, Jiang-Ming Yang and Lei Zhang. irobot: An intelligent crawler for web forums. In *In Proceedings of the 17th international conference on World Wide Web (WWW). ACM, 447456.*, 2008.

[23] Sonia Chiasson David Dittrich Serge Egelman, Joseph Bonneau and Stuart Schechter. Its not stealing if you need it: A panel on the ethics of performing research using public data of illicit origin. In *International Conference on Financial Cryptography and Data Security (FC)*, 2012.

[24] Alice Hutchings Sergio Pastrana, Daniel R. Thomas and Richard Clayton. Crimebb: Enabling cybercrime research on underground forums at scale. In *Proceedings of The Web Conference 2018 (WWW 2018), Lyon, France. ACM, New York, NY, USA*, 2018.

[25] Kyle Soska and Nicolas Christin. Measuring the longitudinal evolution of the online anonymous marketplace ecosystem. In *24th USENIX Security Symposium*, pages 33–48, 2015.

[26] CloudFare Support. Understanding cloudflare under attack mode (advanced ddos protection), February 2020.

[27] Chrome DevTools team. Puppeteer chromium automation library. https://pptr.dev/, 2017.

[28] The Scraper API Team. 5 tips for web scraping without getting blocked or blacklisted. https://www.scraperapi.com/blog/5-tips-for-web-scraping, 2019.

[29] Daniel R Thomas, Sergio Pastrana, Alice Hutchings, Richard Clayton, and Alastair R Beresford. Ethical issues in research using datasets of illicit origin. In *Proceedings of the 2017 Internet Measurement Conference*, pages 445–462, 2017.

[30] Colin Watson and Tin Zaw. Owasp automated threat handbook web applications. Technical report, OWASP, 2018.