Measuring the blocking of AN.ON users by popular websites through web scraping

Jurgen Mulder[1] , **Stefanie Roos**[1]
[1]TU Delft

Abstract

Users of anonymity networks face differential treatment and sometimes get blocked by websites, it is currently unclear how common this blocking is. This research aims to provide an overview of how common this blocking is while utilizing the AN.ON anonymity network. The analysis is accomplished by utilizing automated web scraping and processing to recognise and classify blocks by comparing them to a control connection. This process and software can be used and extended to analyze and compare any two connections. The scope is limited to the one thousand most popular websites according to the Alexa rating.

Different kinds of blocks were identified and automatically recognised in processing, though manual verification is still required. Evidence is found and presented that there is a significant amount of blocking, occurring on approximately 23% of the analyzed domains. There is also a significant difference in blocking between using different cascades.

1 Introduction

When using the internet, one leaves identifiable data that gets collected by various instances and services [1][2]. Some internet users feel the need to prevent their data from being collected [3], and one component which can be used to protect this data and online identity is the use of an anonymity network, such as AN.ON[1]. AN.ON routes traffic from the user through a service to assist in preventing professional data collectors from collecting your data [4]. These anonymity networks can be used by legitimate users for various reasons, but also by criminals to facilitate protection for online crimes [5]. Due to these different potentially malicious users some websites may impose restrictions on users of such anonymity networks. Some content might be blocked, some content might be hidden behind CAPTCHAs, and some websites might even completely lock out such a user [6][7]. There has been some research in a related field about censorship and how to measure it via various methods [8][9][10][11][12][13], and it is clear that there are instances of discrimination against requests originating from anonymity networks [11]. There are also two studies on website blocking while using TOR specifically that acknowledge the fact that knowledge about this issue is sparse [6][7], but there is no known research into the blocking of users of AN.ON. Blocking can be performed by various parties, mainly the ones giving the internet access (E.G. ISPs and governments) and the ones receiving the traffic (E.G. websites). There is some research into the blocking of accessing AN.ON and how to circumvent those blocks [14], but that is not the type of blocking this research is concerned with. It is currently unclear how extensive this blocking by websites is for most anonymity networks, the exploration of the severity and commonality of this blocking while using AN.ON is the goal of this research. This research set out to explicitly answer the question of how frequent blocking of AN.ON users is while visiting popular websites.

This research shows that there is a significant amount of blocking faced by AN.ON users imposed by websites. There are still some unknown factors and uncertainties, but an AN.ON user can expect to see some kind of blocking on approximately 12-23% of popular websites visited.

1.1 AN.ON

Project AN.ON, Anonymity.Online, was created around 2004. The goal of the project is to offer a protection layer between the user and data collectors. This layer is provided by routing all internet traffic through multiple servers, called mixes, in encrypted one kilobyte packets that get interlaced with traffic of other users and dummy packets [15]. The servers that handle AN.ON traffic are all operated by independent entities committed to protecting the data of the users. The main difference between Tor[2] and AN.ON is this inherent trust of routing through independent organisations [16], rather than unknown volunteers with no legal limitations or obligations [17]. AN.ON also consistently sends all traffic through the same route while in use, which is accomplished by having the user choose from a limited number of statically defined routes. These possible routes through mixes are referred to as cascades.

The AN.ON project first took shape in JAP, Java Anon Proxy [2], the most recently released version of which is what this research uses for its experiments. The commercial ver-

[1]https://anon.inf.tu-dresden.de/index_en.html

[2]https://www.torproject.org/

sion of JAP was renamed JonDo/JonDonym, and is managed by JonDos GmbH. JonDonym has announced to be shutting down, with an unknown future for JAP, more details in section 6.1. The terms AN.ON, ANON, JonDo, JAP, and even anonymous-proxy-servers are used by the creators and users to denote the same thing, the anonymity service and the software to access it. The rest of this document will use ANON to denote the AN.ON project and the use of JAP software for the sake of consistency and clarity.

2 Methodology

The core principle used for detecting blocks caused by ANON consists of requesting pages with and without ANON enabled, and then comparing the results. The request made without ANON enabled is referred to as the control or baseline, details about the used baseline network can be found in section 3.5. These requests are made in parallel and always get started at the same time per URL, more information about the used process of which can be found in section 3.1. Multiple web pages are requested per domain, to get an impression of the overall availability of the website. All data received from each request is stored independently in a database. This data is then later processed and compared against their counterparts to reach a final verdict per request, and then summarized together with other pages of the same domain to reach a conclusion. More details about this database, the processing, and the classification of pages and domains that was used can be found in section 3.7.

2.1 Popular websites

This research aims to measure the prevalence of blocks while accessing popular websites using ANON. For this task a list of representative popular websites is needed to perform tests on. Various lists of popular websites exist, with different methods used for the compiling these lists, such as Alexa[3], Majestic[4], Cisco Umbrella[5], Tranco[6], and Quantcast[7]. Many papers use the Amazon Alexa top sites on the web list [18][19][20][21][22][23], being by far the most popular list used in security research according to Le Pochat et al [24], and will therefore be used in this research. Although popular, there are some issues with the Alexa list that allows a malicious actor with few resources to maintain a rank in the top 100k domains [25]. However, only utilizing the top 10k or less should be representative of the most popular websites, as long as no value is bound to the individual rankings [26]. The Alexa list also appears to not take traffic from the EU into account [24], though this does not strictly matter for the popular website criterion. Due to practical limitations in available bandwidth and time, further explored in section 3.3, this research will only use the top one thousand domains that pass the filter described below.

[3]https://www.alexa.com/topsites

[4]https://majestic.com/reports/majestic-million

[5]https://umbrella.cisco.com/blog/cisco-umbrella-1-million

[6]https://tranco-list.eu/

[7]https://www.quantcast.com/top-sites/

Domain Filters

Some websites were filtered out of this list and ignored for practical reasons. These domains took longer than 120 seconds to load the main page on the control connection (see section 3.5), did not load at all, or are categorized as Asian according to the definition below. The slow or not loading domains were removed as they are likely blocking traffic of bots, the geographical area, or are just broken. Examples of this can be found in section 4.5. Predominantly Asian websites were removed mainly due to them being generally extremely slow to load, and often changing a lot of content between requests, making them relatively difficult to compare. Furthermore Chinese users might not even be able to use ANON due to the great firewall attempting to restrict access to anonymity networks [27][28][29]. With ANON being located mainly in the EU with no endpoints near Asia [30] at the time of writing, it would be even slower and less usable for that audience. This filtering out of Asian websites is performed by counting the amount of characters Java marks as Chinese, Japanese, Korean, or Vietnamese (CJKV), using `Character#isIdeographic` [31]. Sites are excluded if more than 300 of these characters are visible on the homepage, or more than three in the title. The limit for CJKV characters on the homepage was set to this high amount to avoid false-positives due to language selectors.

3 Scraping and Processing

As described in the previous section, data from website requests gets collected, stored, and processed. This chapter aims to clarify and explain the exact workings and decisions made in these steps.

3.1 Library

There are several ways of requesting information from a website, but not all are as realistic as a normal user requesting a web page in a browser, and as such may add biases to the experiment. The simplest method is a simple GET request, this can be done in most languages with relative ease. However due to its simplicity this method does not run any JavaScript code of the website, and can therefore theoretically get treated differently, not load certain content, or get blocked by simple checks and bot prevention systems, such as those of Cloudflare[8] [32]. More sophisticated options such as Selenium[9] automate entire browsers, and libraries such as Puppeteer[10] provide APIs to full headless browsers. Headless browsers work similar to normal browsers, only without the GUI being launched. Options such as hiring real people to manually gather data exist [33], but are both out of budget and can introduce further biases [34]. The OpenWPM[11] framework, built upon Selenium, is an often used choice to simulate browsers and extract data from websites, being used in over 76 crawl-based studies as of October 2020 [35]. As there should be no difference for websites between OpenWPM and Selenium, and as OpenWPM uses Python whereas Selenium can be used

[8]https://cloudflare.com/

[9]https://www.selenium.dev/

[10]https://pptr.dev/

[11]https://github.com/mozilla/OpenWPM

with Java, the scraping and crawling in this research is performed utilizing Selenium. It should be noted that all of these options, including Selenium and OpenWPM, do differ from human web browsing [34], and their use can be detected by websites [36].

3.2 Cookies

Most sophisticated scraping libraries support cookie storage, and using them to have a more stateful crawler may be desired by some to better emulate users [34]. Cookies were cleared to keep measurements independent, as having old cookies stored could allow websites to present different behaviour on subsequent requests. Cookies were not cleared between pages of the same domain, to behave more like a normal first time user of a site.

3.3 Rates of querying

The bottleneck in this setup is the ANON connection, as outlined below in the practical limitations. Due to this slow connection, doing multiple requests at the same time in parallel is not an option. Therefore the fastest crawling possible is to load every page sequentially back to back. This assumes that no website is slower than the ANON connection, which, as long as no website is having issues and timing out due to technical issues, should be the case with any large website. This is performed in parallel with the requests done without ANON to get the most up to date comparisons between content. This simultaneously paces both crawlers to behave somewhat like a realistic user, only going to visit another page after the previous one has finished loading. This speed would be possible for a normal user pressing links on a web page, though it is still faster than a user that reads every page in its entirety before loading another.

Practical page load speed limitation

The main limiting factor in how fast pages can be collected is the latency and bandwidth of ANON. From some preliminary testing it was clear that loading basic pages with some images and light JavaScript can take over a minute, see table A for some initial small scale page load time test results. An adblocker sped up the loading of pages significantly, especially with JAP enabled, and is therefore used. This makes sense as ads can use up a lot of bandwidth while browsing websites [37]. Websites can block users for using adblockers, but since this is relatively uncommon [21], and since it is the same in both the base and the experiment, this should not introduce additional bias. The biggest page load time improvement appears to be possible using uBlock Origin [38], and is therefore the adblocker of choice for this experiment, with default configuration settings. An upper limit of two minutes per page is given with uBlock Origin, and as a result only thirty pages can be loaded per hour in a worst-case scenario. Due to this extremely slow connection, pages are given time to load until they fire a document `readyState` of `complete`, or time out after a maximum of 120 seconds. After the `readyState` is set correctly an extra two seconds is provided to the browser to properly render everything and make final adjustments if needed, to allow pages with loading symbols to properly display the main content. With this sped up strategy the six sites tested in table A would allow for well over a hundred pages being loaded per hour. A speed of roughly a hundred pages per hour was reached in real world scraping with all further optimizations listed below.

Caching

For a significant speed improvement in subsequent crawls, a decision was made to enable caching between scraping sessions. These caches are created and handled completely separately per experiment, as to avoid an ANON scrape from using data retrieved by a regular control scrape. The cache serves already downloaded content to the browser if present, avoiding a slower request to the original web server [39]. With these separated caches per browser, any content shown or used on an ANON scrape will have been retrieved by a request utilizing ANON. This adds a small bias against temporary resource-specific outages and blocks, as theoretically a cached result could (temporarily) be getting blocked in the present, but show not to be blocked as it is received from cache. However any intermittent blocking of partial content on only a month worth of scraping data would be hard to distinguish from connection issues or temporary website outages, thus they would likely have gotten discredited either way.

Caching on ANON

ANON likely makes use of caching on the exit node side. In old designs of JAP and the theory behind it, the principle of the `cache-proxy` is discussed, explaining how it should precache requested web pages and their content [40]. As caching is also named in further work, with them labeled as separate proxies [15], it is presumed that this caching is present in ANON. Mentions of this Cache-Proxy is also still present in the source code of Mixes, though the exact implementation was not found, and JonDos could not provide clarification or confirmation about the `cache-proxy` implementations on cascades. This caching could, in theory, trigger bot protection mechanisms on websites, and this could explain some ANON blocks. This caching could also make temporal blocking hard or impossible to detect, as the `cache-proxy` could serve old content. However, as mentioned earlier, such temporal blocking would have likely not been recognised either way.

Lazy-Loading

Lazy loading defers the loading of images until they are either in the viewpoint or near it [41]. With this enabled fewer unnecessary images outside of the observed part of the page are downloaded, saving on bandwidth. As the images below the fold are never compared in this experiment this should not have any impacts on checking blocks. To force lazy loading to work on all pages an extension named LazyLoadify[12] was used, created by Gildas. This ensures that even pages that have not manually set the `loading` attribute to `lazy` will still be lazy loaded. Enabling lazy loading and LazyLoadify sped up the scrape time by about 9-14% and reduced timeouts by 26% in a small scale test of 430 links, repeated twice. Although there are some other variables that could explain this difference, such as the varying speed of ANON and caching,

[12] https://github.com/gildas-lormeau/LazyLoadify

it makes it worthwhile to enable. There were no visual disadvantages found to using this approach during manual verification, with the overall amount of missing images visible being less common than in earlier tests.

3.4 Pages and functionality to test

To get a proper view of the blocking of content on a site, one should ideally exhaustively test every single page and every single functionality that can be used. Unfortunately, this would require a relatively large amount of traffic and engineering work per website, and would simply not be feasible within a reasonable amount of time without overloading all ANON mixes. To attempt to get a proper overview of a site, an initial crawl has selected a small set of three randomly picked URLs linked from the homepage, the amount of pages was chosen arbitrarily. These pages along with the homepage then served as the test set for that website for the remainder of the experiment. Websites also often have different functionalities outside of static content, such as logins, registers, comments, and other such interactions. Due to every website having implemented this in different ways, and automatically entering content could easily get flagged as being a bot and further impose biases, this research will not attempt to interact with any of this sort of functionality. Furthermore automatically inputting and sending data to servers takes up ANON bandwidth and could be seen as malicious bot use by the websites, and is therefore ethically questionable. Only web-content retrieved from a directly linked URL presented without external user-input is compared.

3.5 Compared against

All scrapes are compared against a standard connection to produce results, this standard connection is referred to as the control or baseline connection throughout this document. Furthermore to make sure that blocks identified by the crawler are actually due to ANON, and not due to faults in the crawler or due to the behaviour of the crawler, all block data is compared against this control. As this research aims to explore the blocking of ANON users compared to normal users, all comparisons of blocking are made against a residential connection, and not that of a cloud provider, VPN provider, or other anonymity network. Due to availability, this baseline will therefore be a single Dutch Ziggo[13] provided residential connection.

3.6 Types of blocks

Websites can choose to block or restrict access in various ways, this section functions as a glossary for the types discussed further used. This list is not exhaustive and does not list every single possible block on the internet.

- Errored: No response, blocking all contact and not responding at all (likely via a firewall). Or responding with a HTTP code other than 2xx or 3xx.
- Block page: Returns a valid HTTP 200 response, but with clearly other content including keywords often found on block pages.
- CAPTCHA: Anti-Bot, a Captcha or Test page, the same as a block page, but with an option to pass it after manual input.
- Action block: Allowing access to static content but blocking actions such as logging in, registering, commenting, or otherwise interacting with content.
- Content block: Allowing access to some content, but not all. For example getting everything but heavy to load content such as videos.

As mentioned in section 3.4, not all functionality is tested. Due to the limited amount of available data from this, the action blocks can not be analyzed or recognised in any meaningful way. Due to the low bandwidth of ANON, as seen in section 3.3, some pages simply take too long to load images or videos and time out on them due to this slow speed, rather than potential blocking. Furthermore ANON imposes a maximum download of 2MByte per file [42] which can block further large assets. As it is unclear whether this missing content is from websites performing content blocking, or simply ANON being too slow, no meaningful analysis can be performed on content blocking, aside from entire pages not loading.

3.7 Recognising blocks

Data gets stored as-is during scraping, this section explains how this data is later retrieved and processed. All data of scrapes gets stored in a MariaDB (SQL) database, and the screenshots get saved in png format in a directory. The specifics of the database schema and the specifics of the types of stored data can be found in appendix B. This stored data is then processed, and reprocessed in case of an algorithm change, to produce the final verdicts. This processing is done by first determining characteristics and setting flags based on limits. A summary of the important flags and their design can be found in appendix D. After these flags are set, a final verdict for the scrape is determined by comparing the flags according to the flow scheme provided in appendix C. In this flow scheme, green verdicts signify a confirmed non-blocked page, red verdicts signify confirmed block pages, and yellow verdicts should be manually checked to find unknown block pages. Confirmed blocks always get manually verified to reduce the amount of false positives, as the limits are purposefully skewed to produce these more rather than false negatives for blocking. These final verdicts of scrapes are then grouped together to form a final verdict for the entire website, this final verdict can be one of the following:

- `FULL_BLOCK`, blocks all ANON traffic, or always presents CAPTCHAs. Requires at least 95% of the comparisons to be blocks or CAPTCHAs. The 5% is a margin of error for the comparisons, these sites are deemed unusable.
- `SOME_BLOCKING`, sometimes blocks traffic, sometimes shows CAPTCHAs, sometimes allows users with no issues. Requires at least one comparison to be classified as a block of CAPTCHA.
- `NO_BLOCKING`, always allows ANON users and does not treat them measurably differently.

[13] https://www.ziggo.nl/

Limits

All limits for setting flags based on characteristics were initially set by an educated guess, and were then refined to have as few false negatives for blocking as possible through manual verification. The focus was set on reducing the false negatives, as there were relatively few block pages detected. This does have a negative consequence that there are a significant amount of false positives in the block categories, that all have to be manually verified before making conclusions. The manual verification was performed by manually looking at the screenshots, flags, and underlying statistics of groups of scrape results. Half of the scrapes of randomly chosen recent scrape time slots were verified on their accuracy. This was done until at the end manual verification sessions of over 300 comparisons did not result in any false negatives being identified. The exact final parameters and limits can be found in the source code provided, with no specific descriptions about what limits do what. This source code also includes an easy to use GUI to speed up manual verification, and a command based system to help verifying and adjusting parameters.

3.8 Avoiding biases

To get a perfect comparison, all variables other than the use of ANON should be equalized for both the control and experiment. Obvious variables such as the browser used, the user agent, the operating system, and the resolution of the monitor were of course identical over all the requests. However, it is impossible to get rid of all possible external variables. A few more complicated external variables are highlighted in this section.

Availability/timing/outages

To avoid biases related to timing, such as temporary website availability issues, local internet issues, or changing content, all scraping of the control and the experiments are started at the same time. This way if a website is offline at a certain moment, both should likely have the same failed result. In the case of a local internet issue anywhere along the line an anomaly in the frequency of timeouts should be noticeable. Multiple crawls at different times with comparisons in between were also used to further mitigate this issue. Furthermore after a series of outages on the JAP side and recognising that these blocks always seem to come with multiple in a row, a small system was created to automatically recognise these outages. This system checks all scrapes in order of execution, and if two or more different domains had timeouts in a row they get flagged as having `NETWORK_ISSUES`, this flag will ensure the comparison will always get the `NETWORK_ISSUES` verdict. Results with the `NETWORK_ISSUES` verdict get ignored and discarded from the results, and are not counted towards any final domain verdict.

Net bias

Users from different countries tend to get treated differently, either by their ISP or the website that receives the traffic [43][44]. This is unfortunately not something that can be completely removed without risking further biases as explained in section 3.5.

Country blocking/GDPR

Websites can decide to restrict access from certain countries [20], sometimes for legal compliance [45]. The comparisons are made between different European countries, and can therefore encounter different restrictions. The impact of this should be minimal due to the proximity of the countries being compared, all being in the EU (so receiving the same GDPR treatment), as well as manual verification of block pages.

Bandwidth

As the ANON connection has a relatively high latency and low bandwidth compared to the control connection it can time out on requests more easily. This external variable could be mitigated by somehow limiting the bandwidth and artificially increasing the latency of the control connection. However, as the scraping already takes a long time, and as this slow connection would introduce an extra source of randomness in the data due to more timeouts and unloaded data, it was chosen not to throttle the control connection in this manner. Instead, it is simply seen as a consequence of using ANON and part of the experience, and most false positives due to congestion or timeouts are filtered out manually.

To confirm that this limited bandwidth is not the leading factor in getting blocked, a slow control connection was created. This slow control connection was created by routing the traffic through a local limited proxy, created with the BrowserUp Proxy library[14]. To create a roughly similar speed to ANON, the bandwidht of the BrowserUp proxy was limited to 100Kbit/s and 600ms ping, roughly in line with the speeds claimed by ANON [46]. Then, an experiment was ran comparing the two control connections, the results of this experiment can be found in section 4.4, and are related to ANON in section 4.1. The average scraping speed of this slow connection was roughly in line with that of ANON scrapes based on the average scrape times. It is approximately 17% slower than the Dresden cascade, and approximately 9% faster than the SpeedPartner-Cyrax cascade. A statistical T-Test was attempted for the load times between cascades, but showed that these times could not even be compared for the same cascade at different times. The distributions of the load times were significantly different even at a mere 90% confidence interval. This might be due to jitter in the ANON connection, which was not verified, quantified, or tested.

Software versions

Theoretically any update to software can make the behaviour change, therefore the versions of any software and libraries used during scraping was kept the same between runs on the client side. This was ensured by running everything on a Ubuntu[15] 20.04 LTS virtual machine, and refusing to install any updates after the first scrape was started. Libraries were kept the same using Maven[16] dependency management and not updating any versions of libraries used during scraping.

[14] https://github.com/browserup/browserup-proxy

[15] https://ubuntu.com/download/desktop

[16] https://maven.apache.org/

4 Results

This section will discuss the results obtained from scraping and categorizing the results of 1000 domains. These are categorized and simplified according to the specification described in section 3.7. The experiment was performed on the two freely available ANON cascades [46] in great detail, the results of which are discussed both separately and combined in this section. The results are comparing data gathered during different time frames, this could have caused variances in the data that were not further investigated due to practical limitations.

4.1 Cascade: Dresden

This cascade goes through a single mix, and is marked as a test or experimental service [46]. This would not be a good cascade to use with the goal of anonymization, as the single mix could theoretically de-anonymize the user with ease, although the signed clauses of mix operators prohibit the storage of de-anonymized data or doing anything with it [42]. The endpoint of this cascade is an IP on the network of the TU-Dresden. Scrapes were significantly faster when running on this cascade than on the SpeedPartner-Cyrax cascade, and as such it was ran more, resulting in it providing more data points on the same domains as the other cascade. This cascade saw blocking on approximately 12% of the Alexa top 1000 as defined in section 2.1. According to the Chi-Square test of independence, this is a significantly different higher result than the results from the slow control connection in section 4.4. $X^2(2, N = 2000) = 58.3, p < 0.0001$. Therefore it can be concluded that there is a significant amount of blocking of ANON users on the Dresden cascade, as compared to regular users with an equally slow connection. From this it can be concluded that the blocking is an actual result of the usage of ANON, and not of the limited bandwidth. The composition of the results and the amount of blocking can be found in figure 1(a).

4.2 Cascade: SpeedPartner-Cyrax

This cascade goes through two mixes, one in Germany and one in France, and is the only cascade marked as free while not being experimental [46]. This would be the better free to use cascade for anonymity, as the dual mix system ensures no single party can de-anonymize the user [4]. The endpoint of this cascade is 178.33.255.188 [47], an IP registered to OVH SAS[17] according to the RIPE database [48]. OVH is a large and easily accessible hosting provider, and as such it could make sense for some websites weary of attackers and scrapers to block OVH IP addresses by default. Due to this datacenter IP on the exit mix, it is expected that this cascade sees more blocking than the Dresden cascade mentioned earlier. As can be seen in figure 1(b), around 18% of the Alexa top 1000 saw some sort of blocking on this cascade. This relatively high amount of blocking is likely due to the endpoint IP being that of a commercial datacenter rather than that of a university. According to the Chi-Square test of independence, this is a significantly different higher result than the results from the Dresden cascade in section 4.1. $X^2(2, N = 2000) = 40.1, p < 0.0001$, thus it can be concluded that blocking is more likely on the SpeedPartner-Cyrax cascade than on the Dresden cascade.

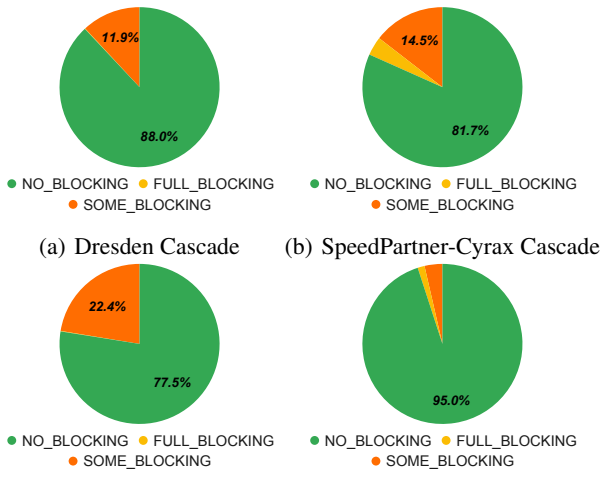
[17] https://ovh.com

4.3 ANON Combined

This data is composed of all scraped data while using ANON, where some cascades had more data points than others. Scrapes were performed over a long time period, with almost 30,000 data points. As defined earlier, if a single instance of blocking on either cascade in all of these spread out data points got categorized as blocking, the entire domain is categorized as `SOME_BLOCKING`. As such, these results might be blown out of proportion and unrealistic for real world usage, and should not be compared with the other statistics mentioned in this section. However, what the results in figure 1(c) do show is that around 23% of the analyzed websites exhibited some form of blocking of ANON users.

4.4 Slow control

To verify these results are not just anomalies or due to the low bandwidth of ANON, an experiment was ran with a slowed down control network instead of an ANON cascade. This experiment was performed on the same list of web pages as the experiments on cascades, with all the same parameters, as often as the SpeedPartner-Cyrax cascade. The workings of this connection were designed to be similar to the specifications of an ANON cascade, and is further described in section 3.8. The results from this experiment are presented in figure 1(d). Notably there is an amount of websites that somehow get classified as being blocked, these were all manually verified, and were mainly websites that presented CAPTCHAs to the slowed down connection. It is not clear why sites presented different pages to these connections, it could perhaps be due to plain random checking, or due to the low bandwidth and slow speed also seen while using ANON, or due to differences in requests while they go through proxies.



(a) Dresden Cascade (b) SpeedPartner-Cyrax Cascade

(c) All Cascades (d) Slow Control Connection

Figure 1: The categorized results of 1000 domains

4.5 Exceptions

Some websites had some idiosyncrasies limited to few domains, such as not accepting https, timing out randomly, crashing selenium with JavaScript, or blocking selenium altogether. These sites, of which there were only 19 in total, were handled slightly differently than the rest or excluded from the list of websites where applicable. A complete list of domains that got different treatment from the others and the exact different treatment is provided in appendix E.

5 Responsible Research

5.1 Reproducibility

Any competent developer with enough time, a stable network connection, and a normal computer to dedicate to this should be able to reproduce any and all experiments performed during this research. However, it would have to be on a different anonymity network, as ANON is about to be closed down as described in section 6.1. Following section 3 should make it possible to create a program that resembles the same functionality as the one produced for this research. The exact source code produced for and used by this research can be found on `https://gitlab.com/simgar98/rpcrawler`, such that anyone can verify the functionality and run it for themselves if desired. The exact list of links used during experimentation is also on this repository under `libs/links-used.txt`. With this source code and enough time even larger tests could be conducted. However, this code has next to no comments in it, little instructions on how to use it, and comes with no warranties whatsoever.

Licensing

To ensure anyone can use this software freely, any code created for this research is licensed under the GNU General Public License v3.0. This licensing provides anyone the freedom to use, change, share, and share changes made to the software however they seem fit [49].

5.2 Congestion

This project uses up some bandwidth of the ANON project. Theoretically, this could use up bandwidth that other users could have made use of, and could slow their connection down by using too much. However, the free usage of ANON is limited to 100 kbit/s [46], with about one or two hundred users per cascade, with the highest seen during this experiment being at most 250. Even if there were 300 concurrent users on the same cascade, and if they would all use their maximum bandwidth, that would be around 30 Mbit/s of bandwidth. Bandwidth is not infinite or free, but even this extreme usage is likely to be fraction of the available bandwidth to the mix operators used. The three owning parties of the mixes present in the two used cascades, SpeedPartner, Cyrax, and TU-Dresden, either use datacenter space or have their own uplinks that can be assumed to be over a gigabit judging by their ASNs[18]. Uplinks have to have over 100Mbit/s available to become a mix operator [50], thus the maximum sustained load of this project would be a thousandth of their minimum resources. The load produced by this research on the ANON network seems insignificant, and should therefore not be an issue.

[18] http://infoservice.inf.tu-dresden.de/cascades

5.3 Reputation

The behaviour of the software used in this experiment is that of a robot, which websites could detect and link to the IP addresses used. This could result in the IP addresses of ANON getting a worse reputation because of the actions performed in this research. However, as ANON is shutting down its service, this should not be an issue for the long term. This could also have had short term impact for current users of ANON, which there are not a lot of, who may possibly have experienced a heightened frequency of differential treatment while this experiment was running. The results of this are hard or impossible to quantify.

5.4 Legality

There are three main parties involved in this experiment, outside of the web sites, that could have issues with this crawling and scraping. These three parties are JonDos, Ziggo, and the government of The Netherlands. Firstly JonDos could take issue in abuse of their services, however their terms of service agreement does not prohibit this use case [51]. Secondly Ziggo, the ISP that provided the connection for the control queries, could also take issue in abuse of their services. However Ziggo does not explicitly prohibit, condemn, or even mention this usage of their services in their terms and conditions [52]. So JonDos and Ziggo do not seem to have any issues with this scraping, but the Dutch Government theoretically could. There does not seem to be any law prohibiting scraping directly, it is a rather gray area in the Netherlands with certain instances being forbidden by other laws. For example, in the Netherlands a website can legally prohibit this scraping via their privacy policy, as Ryanair did in 2015, winning a court battle over it [53]. There are more instances of such lawsuits against scrapers in the United States, that rely on various laws that have similar counterparts in EU member states [32]. Due to this being a legally gray area, and there being no significant costs or damage incurred by any parties involved, this should not be an issue, however it should definitely be considered by anyone that wants to do a lot of scraping.

5.5 Robot exclusion protocol

The robot exclusion protocol, REP, was designed back in 1994 [54], to keep out unwanted automated traffic. Currently it is not an official internet standard backed by a standards body, however a draft to formalize the REP as such is now being worked on [55]. Still, at the time of writing, it is simply an unofficial widely used guideline. It was designed to automatically tell robots, including scrapers, to ignore certain pages or entire websites via the `robots.txt` document. There can be various valid reasons to not want robots on a website, and a well-mannered scraper should probably take the REP into account while requesting data, but this project does not. Very few requests are made to very large, as defined in section 2.1, websites. These few requests to pages linked to from the main pages likely do not represent any form of

significant load on such websites, and as such should not be seen as problematic for ignoring the REP. In a larger scale study with more requests for longer periods of time, or with more end points, it would be courteous to honor the REP of websites.

6 Conclusions and Future Work

It is clear that ANON users face significant amounts of blocking while browsing popular websites. The exact amount is unclear, and varies on the cascade used and the definition of facing blocking, but the proportion was found to be between 12% and 18% of the most popular 1k websites per cascade. However, this number might be skewed towards the high side, as a control connection with similarly limited bandwidth also faced blocking on about 5% of websites. Furthermore this number may also be too low due to the limited functionalities and web pages tested per website. This research set out to quantitatively measure the prevalence of blocking, but due to there being many untested variables and interpretations or definitions, no single percentage can be concluded upon. This section aims to highlight some limitations that were identified while this experiment was running. If these limitations were to be solved, much more meaningful statistics could be generated with a similar experimental setup.

6.1 AN.ON Defunct

The current JonDonym website[19] now reads that *"Unfortunately, we have to close our service until 2021-08-31. Until then, our operators will ensure you that you can consume your existing plans. Purchasing new plans is not possible any more."* on a small banner at the top of the screen. The German version clarifies that it will close on instead of until 2021-08-31. From this it is clear that ANON will cease operations. After inquiry it was clarified that the free Dresden cascade will remain online, with all others scheduled to shut down. The creators even use `www.jondonym.net` in promotional material [56] while it is not even registered at the time of writing, having been used by the JonDos company for displaying their portfolio between 2015 and 2019 [57]. However, while this research was performed and this paper was written specifically for use with ANON, the methods used should still apply to any other anonymity network.

6.2 Limitations

There are some shortcomings to the methods and software used, this section aims to list the ones currently known of. These could be improved in further work and should be taken into account when using the presented results for anything.

- Time frame: Measurements were not performed at the same time, several samples exist from different dates, with no distinctions made in the results.
- Tested functionality: As mentioned before, only limited functionality is tested.
- Webdriver detection: Selenium sets a `webdriver` property, websites can easily use this to detect they are getting visited by an automated program.
- Amount of websites analyzed: The subset of 1000 tested domains should give a decent overview of popular websites, but analyzing more would be better.
- Amount of pages tested per website: As mentioned previously, ideally all pages of a site should be checked, but this exhaustive testing is impossible.
- Use of caching: Cache could have masked some temporal blocking of some resources.
- One time measurement: As this is a short experiment, it only captures a one time measurement over a short period of time.
- Inconsistent manual verification: Manual verification was done at different times, with likely some different ideas as to what constitutes as a block or simply a timeout, which possibly caused some inconsistencies.
- Selenium issue with switching: Sometimes upon switching queries the page content would not change. This resulted in some discarded data points upon manual verification, these were marked as having network issues as to not be included in the results.
- Loading pages: Some pages load a small document with a loading symbol quickly, and then begin loading the actual content. With the current setup, this does not get detected, no extra time gets allotted, and the decisions have to be made on a loading page.
- Viewport: Nothing was compared beyond the fold, while theoretically a page could block content lower down.

6.3 Further work

Further work could attempt to improve upon the limitations listed in the previous section, and could also work on expanding the scope, some options for which include:

- Categorizing the websites and seeing if there are correlations in blocking amounts between categories.
- Exploring whether analyzing more pages of websites even matters at all.
- Exploring whether other anonymity networks have similar blocking characteristics. This could easily be performed for any service provided in HTTP proxy form with little modification to code.

6.4 Acknowledgements

Special thanks goes to Anant Pingle, Paula Iacoban, Francine Biazin do Nascimento, and Willemijn Tutuarima, who were all working on similar projects as the one in this report. Resources were shared, ideas were discussed, and problems were discussed by all members of this group.

References

[1] N. Thompson, A. Ahmad, and S. Maynard, "Do privacy concerns determine online information disclosure? the case of internet addiction," *Information & Computer Security*, 2021.

[19] https://anonymous-proxy-servers.net/

[2] O. Berthold, H. Federrath, and S. Köpsell, "Web mixes: A system for anonymous and unobservable internet access," in *Designing privacy enhancing technologies*, pp. 115–129, Springer, 2001.

[3] A. I. Standlee, "Under the watchful eye: Users'perceptions of online privacy and surveillance," *AoIR Selected Papers of Internet Research*, 2020.

[4] ANON, "Technischer hintergrund von jap," *Technische Universität Dresden*, 2001.

[5] B. Li, E. Erdin, M. H. Gunes, G. Bebis, and T. Shipley, "An overview of anonymity technology usage," *Computer Communications*, vol. 36, no. 12, pp. 1269–1283, 2013.

[6] S. Khattak, D. Fifield, S. Afroz, M. Javed, S. Sundaresan, V. Paxson, S. J. Murdoch, and D. McCoy, "Do you see what i see? differential treatment of anonymous users," in *NDSS Symposium'16*, Internet Society, 2016.

[7] R. Singh, R. Nithyanand, S. Afroz, P. Pearce, M. C. Tschantz, P. Gill, and V. Paxson, "Characterizing the nature and dynamics of tor exit blocking," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 325–341, 2017.

[8] S. Burnett and N. Feamster, "Encore: Lightweight measurement of web censorship with cross-origin requests," in *Proceedings of the 2015 ACM conference on special interest group on data communication*, pp. 653–667, 2015.

[9] R. Sundara Raman, P. Shenoy, K. Kohls, and R. Ensafi, "Censored planet: An internet-wide, longitudinal censorship observatory," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 49–66, 2020.

[10] G. Aceto, A. Botta, A. Pescapè, N. Feamster, M. F. Awan, T. Ahmad, and S. Qaisar, "Monitoring internet censorship with ubica," in *International Workshop on Traffic Monitoring and Analysis*, pp. 143–157, Springer, 2015.

[11] A. A. Niaki, S. Cho, Z. Weinberg, N. P. Hoang, A. Razaghpanah, N. Christin, and P. Gill, "Iclab: a global, longitudinal internet censorship measurement platform," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 135–151, IEEE, 2020.

[12] A. Filasto and J. Appelbaum, "Ooni: Open observatory of network interference.," in *FOCI*, 2012.

[13] A. Sfakianakis, E. Athanasopoulos, and S. Ioannidis, "Censmon: A web censorship monitor," in *USENIX Workshop on Free and Open Communication on the Internet (FOCI)*, p. 2, 2011.

[14] S. Köpsell and U. Hillig, "How to achieve blocking resistance for existing systems enabling anonymous web surfing," in *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pp. 47–58, 2004.

[15] S. köpsell, "Anondienst - design and implementierung," *Documentation for the anonymity protocol*, 2004.

[16] J. GmbH, "Benefits of jondonym." Retrieved from https://anonymous-proxy-servers.net/en/benefits.html.

[17] i. Tor Project, "Tor project — relay requirements." Retrieved from https://community.torproject.org/relay/relays-requirements/.

[18] Q. Jacquemart, C. Pigout, and G. Urvoy-Keller, "Inferring the deployment of top domains over public clouds using dns data," in *TMA 2019 - Proceedings of the 3rd Network Traffic Measurement and Analysis Conference*, pp. 57–64, 2019.

[19] A. Kaizer and M. Gupta, "Characterizing website behaviors across logged-in and not-logged-in users," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, vol. 14-16-November-2016, pp. 111–117, 2016.

[20] A. McDonald, B. VanderSloot, M. Bernhard, W. Scott, L. Valenta, N. Sullivan, J. Alex Halderman, and R. Ensafi, "403 forbidden: A global view of cdn geoblocking," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pp. 218–230, 2018.

[21] M. H. Mughees and Z. Qian, "Detecting anti adblockers in the wild," in *Privacy Enhancing Technologies Symposium (PETS)*, 2017.

[22] N. Samarasinghe and M. Mannan, "Towards a global perspective on web tracking," *Computers and Security*, vol. 87, 2019.

[23] S. Van Acker, D. Hausknecht, and A. Sabelfeld, "Data exfiltration in the face of csp," in *ASIA CCS 2016 - Proceedings of the 11th ACM Asia Conference on Computer and Communications security*, pp. 853–864, 2016.

[24] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

[25] W. Rweyemamu, T. Lauinger, C. Wilson, W. Robertson, and E. Kirda, *Getting Under Alexa's Umbrella: Infiltration Attacks Against Internet Top Domain Lists*, vol. 11723 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Nature, 2019.

[26] W. Rweyemamu, T. Lauinger, C. Wilson, W. Robertson, and E. Kirda, *Clustering and the Weekend Effect: Recommendations for the Use of Top Domain Lists in Security Research*, vol. 11419 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Nature, 2019.

[27] S. Chandel, Z. Jingji, Y. Yunnan, S. Jingyao, and Z. Zhipeng, "The golden shield project of china: A decade later—an in-depth study of the great firewall," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 111–119, IEEE Computer Society, 2019.

[28] R. Ensafi, P. Winter, A. Mueen, and J. R. Crandall, "Analyzing the great firewall of china over space and time," *Proceedings on privacy enhancing technologies*, vol. 2015, no. 1, pp. 61–76, 2015.

[29] D. Normile, "Science suffers as china plugs holes in great firewall," 2017.

[30] J. GmbH, "Jondonym mix operators." Retrieved from https://anonymous-proxy-servers.net/en/operators.html.

[31] Oracle, "Character javadoc," Jun 2020. Retrieved from https://docs.oracle.com/javase/7/docs/api/java/lang/Character.html.

[32] S. Vanden Broucke and B. Baesens, *Practical Web scraping for data science*. Springer, 2018.

[33] W. Meng, R. Ding, S. P. Chung, S. Han, and W. Lee, "The price of free: Privacy leakage in personalized mobile in-apps ads.," in *NDSS*, 2016.

[34] D. Zeber, S. Bird, C. Oliveira, W. Rudametkin, I. Segall, F. Wollsén, and M. Lopatka, "The representativeness of automated web crawls as a surrogate for human browsing," in *The Web Conference 2020 - Proceedings of the World Wide Web Conference, WWW 2020*, pp. 167–178, 2020. Cited By :2.

[35] P. U. W. research group, "Studies using openwpm," Oct 2020.

[36] Sesamestrong, "Live fingerprinting methods to evade · issue 239 · berstend/puppeteer-extra," Jul 2020. Retrieved from https://github.com/berstend/puppeteer-extra/issues/239.

[37] A. A. Albasir and K. Naik, "Smow: An energy-bandwidth aware web browsing technique for smartphones," *IEEE Access*, vol. 2, pp. 1427–1441, 2014.

[38] J. M. Pearce, "Energy conservation with open source ad blockers," *Technologies*, vol. 8, no. 2, p. 18, 2020.

[39] M. Contributors, "Http caching - mozilla mdn web docs," May 2021. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching.

[40] O. Berthold, H. Federrath, and S. Köpsell, "Web mixes: A system for anonymous and unobservable internet access," in *Designing privacy enhancing technologies*, pp. 115–129, Springer, 2001.

[41] M. Contributors, "The image embed element - html: Mdn," Apr 2021. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img.

[42] J. GmbH, "Jondonym - operational agreement." Retrieved from https://anonymous-proxy-servers.net/en/downloads/01OperationalAgreement.pdf.

[43] T. Karr, "Net neutrality violations: A brief history," Jan 2018.

[44] M. C. Tschantz, S. Afroz, S. Sajid, S. A. Qazi, M. Javed, and V. Paxson, "A bestiary of blocking: The motivations and modes behind website unavailability," in *8th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 18)*, 2018.

[45] M. Trimble, "Geoblocking, technical standards and the law," *Scholarly Works*, 2016.

[46] J. GmbH, "Jondonym - state of the anonymization services." Retrieved from https://anonymous-proxy-servers.net/en/status/index.php.

[47] J. GmbH, "Infoservice - cascades." Retrieved from http://infoservice.inf.tu-dresden.de/cascades.

[48] RIPE-NCC, "Ripe database query." Retrieved from https://apps.db.ripe.net/db-web-ui/query.

[49] F. S. Foundation, "Gnu general public license."

[50] J. GmbH, "Jondonym - steps to become a mix operator." Retrieved from https://anonymous-proxy-servers.net/wiki/index.php/Become_a_Mix_Operator.

[51] J. GmbH, "Standard terms and conditions for billing the jondonym service." Retrieved from https://anonymous-proxy-servers.net/en/downloads/TaC.pdf.

[52] VodafoneZiggo, "Algemene voorwaarden ziggo," Dec 2019. Retrieved from https://www.ziggo.nl/voorwaarden.

[53] J. Kraan, "Vergelijkingssite mag geen gegevens van ryanair verzamelen," Jan 2015. Retrieved from https://www.nu.nl/internet/3973127/.

[54] "A standard for robot exclusion," Jun 1994. Retrieved from http://www.robotstxt.org/orig.html.

[55] M. Koster, G. Illyes, H. Zeller, and L. Harvey, "Robots Exclusion Protocol," Internet-Draft draft-koster-rep-04, Internet Engineering Task Force, Dec. 2020. Work in Progress.

[56] J. GmbH, "Support for jondonym." Retrieved from https://anonymous-proxy-servers.net/en/support.html.

[57] J. GmbH, Aug 2015. Retrieved from http://web.archive.org/web/20150801175022/http://jondonym.net/.

A Preliminary speed test

Preliminary speed tests done manually, with likely biases due to caching, even though ”disable cache” was checked and page was refreshed with ctrl+f5 if visited earlier. Time taken from FireFox network timings, last item to load on a page. JAP was set to use a two hop mix, going through SpeedPartner in Germany and exiting through Cyrax in France. There are major biases and likely errors in this data.

Page	Regular (s)	Regular + Adblock (s)	JAP (s)	JAP + Adblock (s)
google.com	2.01	0.89	34.98	18.73
ipchicken.com	6.29	1.10	31.86	4.19
youtube.com (cookie consent page)	3.64	3.86	16.33	18.11
tmall.com (excl rotating banners)	19.02	15.29	124	84
yahoo.com (cookie consent page)	0.69	0.88	4.11	3.70
en.wikipedia.org	0.94	1.10	19.05	14.39

Table 1: Preliminary speed test

B Database design

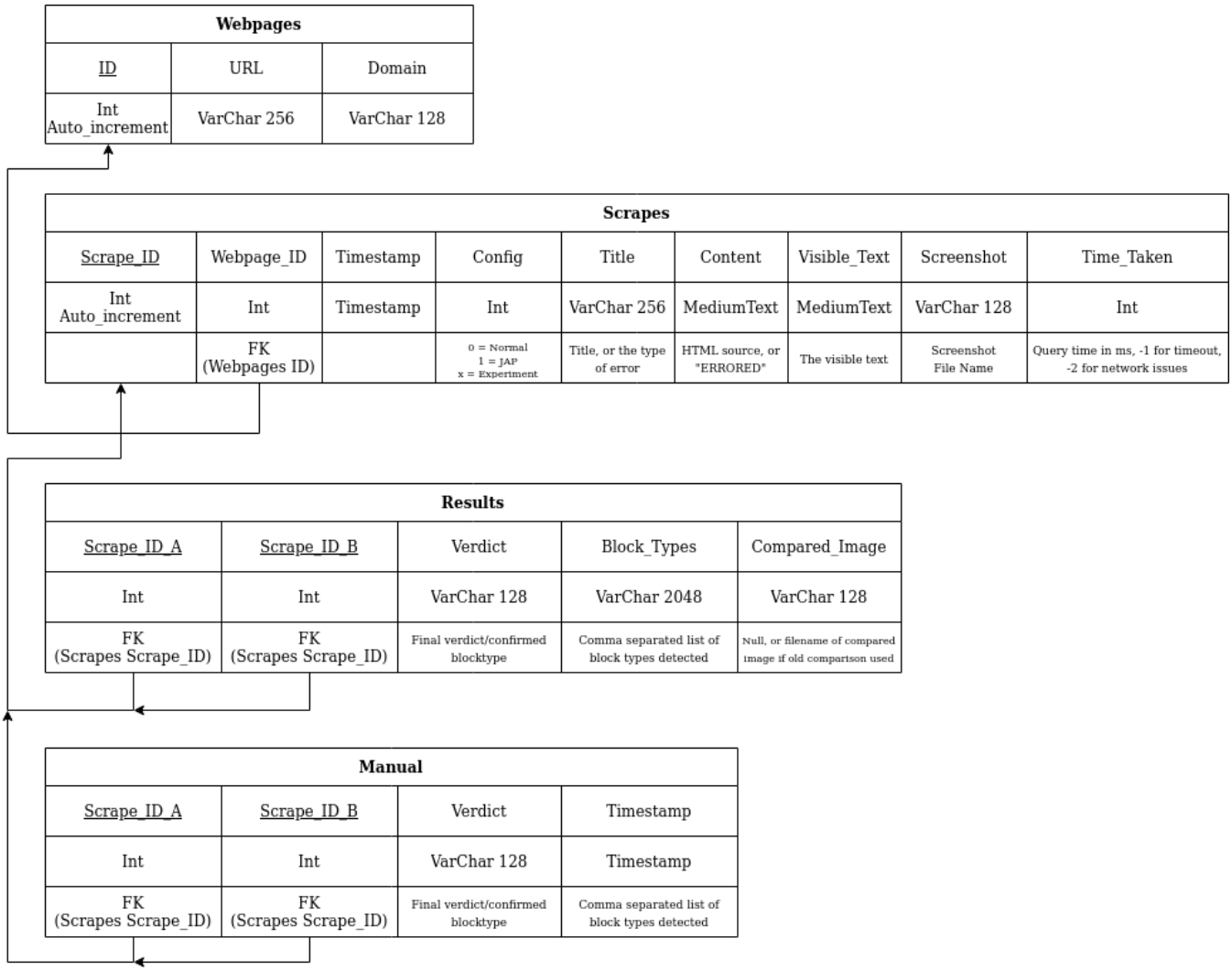


Figure 2: Crawling database schema

C Block flag final verdict flow

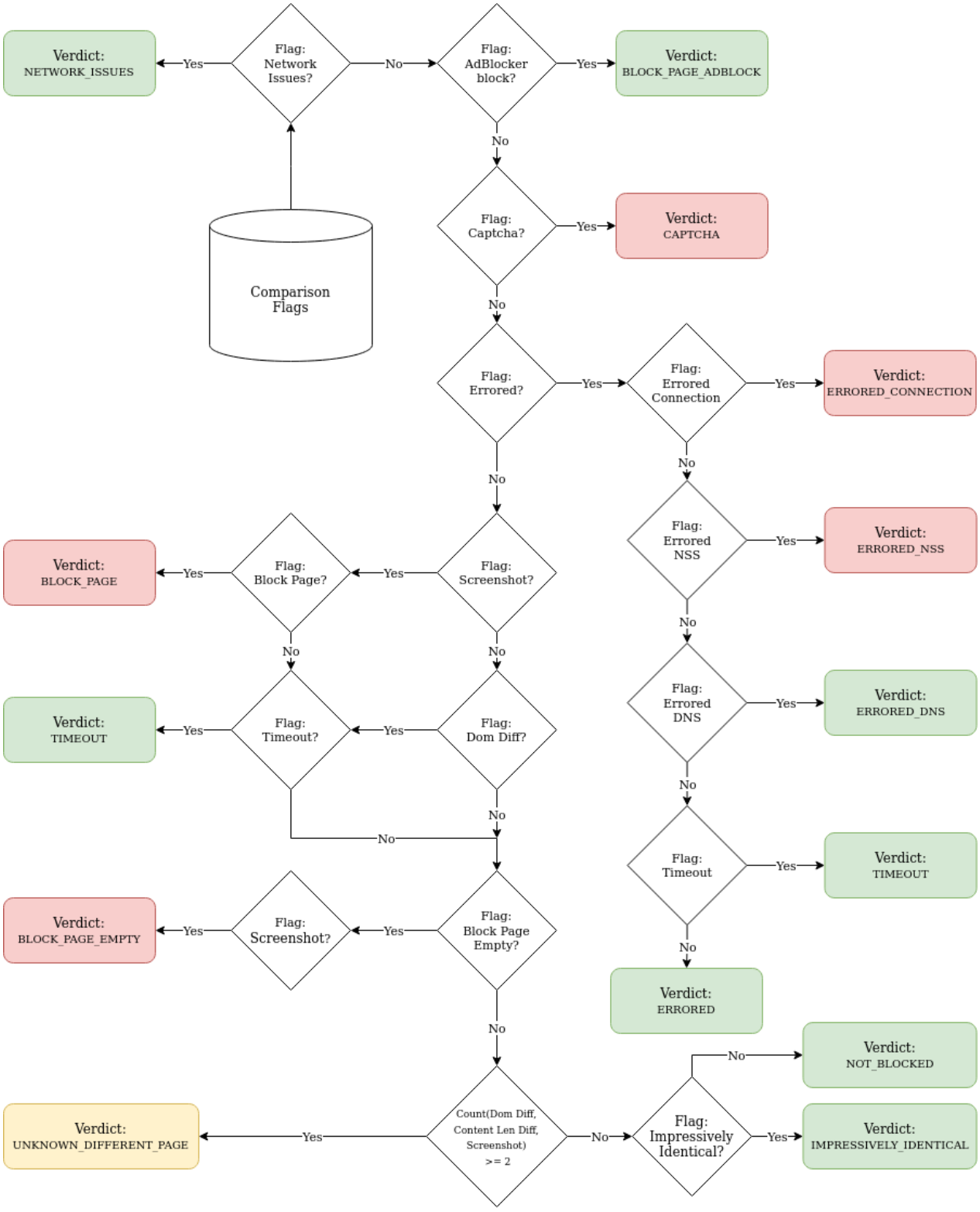


Figure 3: Final verdict determination flow

D Block flags

D.1 Flag: SCREENSHOT_DIFFERENCE

Perhaps the most important check is the one that compares how the page looks. The scraper takes a screenshot of everything that is visible when the page has loaded, this step compares these screenshots between the crawl with and without JAP. This comparison is done using the JImageHash[20] library created by Kilian Brachtendorf. The screenshots of the control and the experiment get hashed using the same 64-bit perceptual hashing hashing algorithm, and the hashes then get compared. Visually similar images get hashes which differ less from each other than non-similar images. When the normalized hamming distance of the hashes is over a threshold, the comparison gets marked as having a `SCREENSHOT_DIFFERENCE` flag. For reference, a completely identical web page with a big banner picture swapped out will have a normalized hamming distance of over 0.5. Web pages which differ by a small banner on the top being present or missing will have a normalized hamming distance in the 0.2-0.4 region.

D.2 Flag: CONTENT_LENGTH_DIFFERENCE

A simple check to verify and check if a page is not completely different. This flag compares the length of the entire HTML document of the pages through the following formula, where a is the length of one document, and b is the length of another:

$$\frac{|a-b|}{\max(a,b)}$$

This flag gets set rather often, as websites often include different amounts of hidden HTML items and differently obfuscated or compressed scripts, or simply due to rotating content for different countries or visitors.

D.3 Flag: DOM_DIFFERENCE

A simple check to see how similar the DOM structure of pages is. The HTML saved is parsed using JSOUP [21], the occurrences of each tag are counted, and compared. The difference between the different tags are added up and divided by the total amount of tags, and the flag is set when a threshold is passed. This flag gets set rather quickly, even with visually identical pages, and is therefore always used in conjunction with other flags.

D.4 Flag: BLOCK_PAGE

This scans the text of the two scrapes for different terms which can often be found on blocking pages. This flag gets raised if the experiment has one or more extra occurrences of a term more than the control. Any time a block page was found while browsing normally or during manual verification, as many distinct words or numbers were added to the terms as possible. The current selection of terms can be found in the source code in the `BlockKeyWord` enum.

D.5 Flag: BLOCK_PAGE_EMPTY

This flag is raised if the pages did not time out, but the experiment result was significantly shorter in length and domain structure, to the point where it is likely a next to completely blank page. This was added to flag websites which serve an empty page instead of a page with words explaining why they do not get the regular web page. This flag can, quite often, provide false positives due to loading pages which set the document `readyState` to `complete` before loading the actual content of the page. These false positives are filtered out manually, but some might be misidentified even by humans.

D.6 Flag: CAPTCHA

This flags when the experiment has a CAPTCHA on the page, while the control does not. This is done by scanning through the document parsed with JSOUP and checking for known CAPTCHA patterns. Further flags exist for detecting CAPTCHAs on the control instead of the experiment, or on both, which do not classify as blocks.

D.7 Flag: ERRORED

There are a few different `ERRORED` flags, which flag when the scrape raises different types of errors. This includes flags like `ERRORED_DNS` which flags in case the experiment results in a DNS error, and `ERRORED_CONNECTION` which flags in case no connection could be made, which is usually if a websites revokes all traffic of the IP. These are all considered to be blocking.

D.8 Flag: TIMEOUT

Set when loading the page took so long that it went over the maximum allotted time of 120 seconds.

D.9 Flag: NETWORK_ISSUES

Set on sraped data that was identified to have been performed during prolonged connectivity issues. See section 3.8 for more details on how these are recognised.

[20] https://github.com/KilianB/JImageHash

[21] https://jsoup.org/

D.10 Flag: IMPRESSIVELY_INDENTICAL

This flag and final verdict is a simple extra category for results that would normally get a final verdict of not being blocked. It only gets set if the screenshots are visually indistinguishable, or if the screenshots have a normalized hamming distance of less than 0.001 with 99.9% similar HTML and text length. This category is purely used to avoid manually checking the most obviously not-blocked web pages.

E Domains treated differently

Domain(s)	Reason	Treatment	Used
bet365.com	Crashes selenium with an infinite JavaScript loop while loading.	Blacklisted	No
ikea.com	Was incredibly unstable. Sometimes timed out in regular browser without ANON.	Blacklisted	No
geeksforgeeks.org, tencent.com	Specifically wants https://www. at the start or does not work.	Blacklisted	No
myworkdayjobs.com, twimg.com, cloudfront.net, bbcollab.com, banvenez.com, akamaized.net, 9384.com	Need some specific subdomain to work, do not respond to requests on the domain directly at all.	Blacklisted	No
amazonaws.com skype.com	Blocks selenium requests.	Blacklisted	No
go.com royalbank.com addthis.com cambridge.org	Requires http instead of https	http	Yes
thepiratebay.org	Dutch ISPs have to block the PirateBay	Blacklisted	No
laroza.net	Redirects to port 2053, which is not supported by ANON	Blacklisted	No

Table 2: Domains treated differently