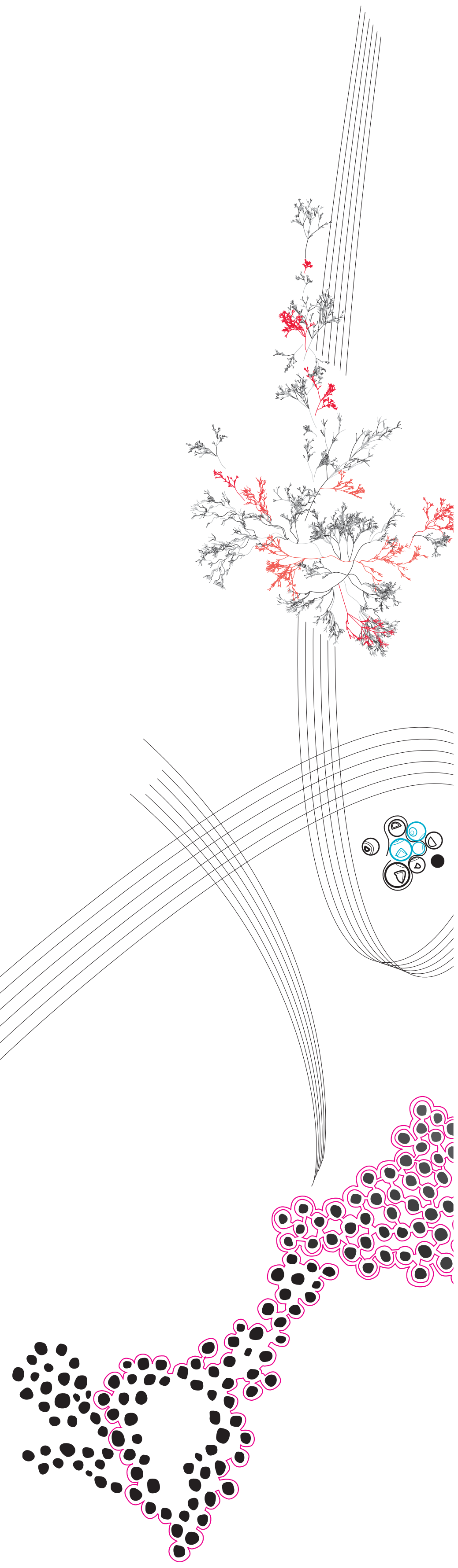
MSc Computer Science
Final Project



Web Crawl Refusals: Insights From Common Crawl

Mostafa Ansar

Supervisors:
prof.dr. Ralph Holz
prof.dr. Anna Sperotto

May , 2024

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Contents

Chapter 1

Introduction

The primary report of this thesis is presented as a research paper, following an agreement with my supervisory team. This introductory chapter provides an overview of the research process and demonstrates how this work satisfies the quality requirements for a Master thesis. While Chapter 2 delves into the technical specifics, this chapter focuses exclusively on the procedural aspects, tracing the project from its inception to the completion of the final paper.

The research paper presented in Chapter 2 is prepared for submission to the ACM Internet Measurement Conference (IMC) in 2024[1]. The paper explains the methodology and results of my research work within the 13-page limit set by the IMC. While the research outlined in Chapter 2 is a product of my independent effort, it is expected to undergo further refinements by my supervisors before the final submission to the conference.

The rest of this chapter will describe how all the key criteria[2] for meeting the requirements of a Master thesis have been fulfilled in this thesis.

Interpreting the initial problem and translating it to more concrete research questions

The project began by exploring the Common Crawl's[3] web crawl dataset to identify potential biases in data or gaps in coverage. This exploratory phase did not start with a well-defined problem statement; instead, the initial goal was to understand the structure of the dataset and assess the design and functionality of the Common Crawl project, looking out for potential drawbacks.

The research direction evolved during the *Research Topic* phase, following an extensive literature review on Internet scanning and web measurements. Initially, we considered the idea of identifying gaps in Common Crawl's data by analyzing variations in web content that a different crawler technology or vantage point than those of Common Crawl would cause. However, the dynamic nature of the web posed challenges in attributing content differences solely to the crawler or vantage point. This insight led us to focus on instances where access by Common Crawl was explicitly blocked by websites. The research questions were eventually formulated as:

1. What is the prevalence of refusals in a Common Crawl web crawl snapshot?

[1] https://conferences.sigcomm.org/imc/2024/

[2] This list is extracted from the information page of the course "192199978 - Final Project CS" on University of Twente's Osiris system.

[3] https://commoncrawl.org/

2. What types of refusals are most common?

3. How can Common Crawl avoid or mitigate these refusals?

Identifying and using relevant literature and tools

This criterion was met during the *Research Topics* course where I identified and reviewed literature from a broad spectrum of studies related to active measurements, including works on Internet-wide scanning and web measurements, with a focus on coverage and completeness.

In the section dedicated to Internet-wide scanning, I examined the fundamentals of such scans as a foundation for active measurement studies in general. I then identified and explored several key dimensions of scanning that are commonly addressed in the literature: the scope of the scan (methods to generate target address list), the order of scan, improving accuracy (factors such as timing intervals between probes), and the origin of scans, which influences overall scan coverage. The findings from related works concerning these dimensions were thoroughly analyzed and documented.

Regarding web crawling and measurements, I explored the basic mechanics of web crawlers, focusing on web content acquisition, data storage strategies, and web search techniques. Then, the Common Crawl project was introduced, detailing its operational policies, including prioritization, revisiting, politeness, and parallelisation methods. Additionally, I discussed various studies in the field of web measurements, from security-focused crawls to research assessing the effectiveness of different crawling technologies. I also examined works on website unavailability and server-side blocks to understand the methods used to identify such blocks.

With regard to the relevant tools, the most challenging aspect of this project was acquiring the necessary skills to manage and analyze big data, which was a field I was unfamiliar with. To tackle this, I dedicated several months to learning and experimenting with Apache Spark[4] and gradually improved my PySpark skills to handle and analyze terabytes of raw crawling data efficiently. These efforts resulted in creating a robust data processing pipeline that could download, parse, pre-process, filter, and analyze approximately 561 million records from 3.43 TB of compressed data on a personal physical server I set up for this task.

Furthermore, I identified and utilized several other tools to enhance the data gathered from the crawls. For instance, I conducted active DNS measurements for a subset of domains to extract and analyze their NS and PTR records, which helped me identify their hosting providers. Additionally, I employed a variety of other tools to transform HTML source codes into a textual format, extract fully qualified domains and registered domains from page addresses, and obtain autonomous system numbers of website IP addresses.

Working systematically and with an academic attitude

The project unfolded in a series of systematic steps using a rigorous academic approach. Initially exploratory, the project began with a thorough review of existing literature on web measurements. This initial phase was crucial as it laid the groundwork, highlighting key challenges and existing gaps in the field of web crawling. The insights gained from this comprehensive literature review informed the adjustments made to the initially designed research questions, allowing for a more targeted investigation into explicit web refusals encountered in web content.

[4]https://spark.apache.org/

Following this, the project transitioned into the data analysis phase, where specific methodologies were employed to gather and analyze data related to server-side blocks systematically. The use of semantic analysis to categorize refusal types was both novel and aligned with academic standards. In addition, each methodological step was carefully documented and continuously refined based on ongoing findings and feedback from academic supervisors.

Moreover, regular presentations and discussions of the findings in weekly meetings with supervisors ensured that the project maintained a high academic standard. These meetings facilitated a critical examination of the work completed and reinforced the systematic research approach.

Using skills and insights learned during the Master programme

The skills I acquired during the Master programme were instrumental in structuring this work, particularly in two key areas. First, the insights I gained from studying research papers in the domain of Internet measurement in various courses in this programme helped me design the methodology and interpret the results of this research. Second, the big data management skills I learned during this programme equipped me with the ability to manage and process large-scale data efficiently. These competencies were crucial for developing the methods, synthesizing information and drawing comprehensive conclusions from the results.

Producing work of sufficient depth and quality

The paper produced for this thesis provides valuable insights into server-side blocks within a large-scale web crawl dataset. It advances the current state-of-the-art by thoroughly investigating the frequency and nature of web refusals observed in a public web crawl data set. Furthermore, this study improves upon the server-side block detection method of previous research, which mostly relied on HTTP status codes and Cloudflare errors, by introducing a method of semantic analysis to identify and categorize refusal messages encountered by Common Crawl.

The study reveals a significant occurrence of web refusals, particularly noting a substantial cluster of websites managed by subsidiaries of a major hosting company. It also evaluates the persistence of these blocks and the effectiveness of various strategies used by Common Crawl. Based on these observations, the paper proposes recommendations to enhance Common Crawl's back-off strategies for better handling and avoidance of server-side blocks, aiming to broaden its coverage.

Working independently and goal-oriented under the guidance of a supervisor and benefiting from the guidance of the supervisor by setting the agenda for meetings and taking suggestions on board

The work on this project was carried out independently, with consistent, targeted guidance provided through weekly meetings with my supervisors. Each meeting was planned with a structured agenda to ensure that discussions were productive and focused. This structured approach enabled the effective communication of the latest methodologies, emerging results, and any challenges encountered. Additionally, it ensured that strategic decisions aligned with the overarching objectives of the project, fostering a coherent and goal-oriented development process.

Preliminary analysis and findings were routinely examined during these sessions, with the supervisors providing critical feedback that shaped the direction and execution of subsequent phases of the project. These results were visually presented through well-prepared slide decks each week to facilitate clearer and more efficient communication. This enhanced the clarity of the discussions and allowed for more dynamic and informed feedback. The iterative process of analysis, presentation, and refinement through these weekly interactions ultimately led to the development of a robust methodology. The insights gleaned from this process were instrumental in compiling the comprehensive final report, encapsulating the depth and breadth of the research conducted.

Chapter 2

Research Paper

Understanding Web Crawl Refusals: Insights from Common Crawl

Mostafa Ansar

ABSTRACT

This study investigates server-side blocks encountered by Common Crawl, a major web crawling project. Unlike previous studies that rely on HTTP status codes or Cloudflare errors to identify server-side blocks, this research utilizes semantic analysis of page contents to cover a broader range of refusals. By constructing and utilizing 147 fine-grained regular expressions crafted to identify various refusal pages precisely, we found that approximately 1.68% of websites in a Common Crawl snapshot exhibit some form of explicit refusal. Significant contributors to these refusals include large hosting providers and website builders. Our analysis categorizes the diverse forms of refusal messages, ranging from outright blocks to challenges and rate-limiting responses across multiple HTTP status codes. The study also examines the temporal dynamics of refusals, offering insights into the persistence of these blocks and the effectiveness of Common Crawl's retry strategies. Our findings highlight the diversity of server-side blocks and suggest using tailored approaches to navigate and mitigate them.

1 INTRODUCTION

Web crawlers are recognized as valuable tools for systematic data collection for various research purposes. One challenge for web crawlers that could affect their coverage and accuracy is server-side blocking. Websites may block web crawlers for various reasons, such as protecting against malicious bots, enforcing geo-blocking, or preventing unauthorized information extraction and excessive resource usage.

Previous research has often concentrated on server-side blocking stemming from geo-blocking or compliance with regulations, primarily by assessing the availability of websites from different geographic locations [1, 9, 12]. However, to the best of our knowledge, no research has specifically examined server-side blocks encountered by web crawlers in large-scale web crawl datasets such as Common Crawl[1]. Given its comprehensive nature as one of the largest publicly accessible web crawl archives, Common Crawl provides a valuable resource for characterizing server-side blocks encountered by web crawlers.

This research utilizes the Common Crawl dataset to explore the extent and nature of server-side blocks and aims to enhance understanding of how these blocks impact web crawlers. Unlike prior work that relies on HTTP status codes or errors generated by Cloudflare to identify blocks [12], our approach includes a semantic analysis of page contents. This method allows for identifying a broader range of server-side blocks, providing a more detailed understanding of the reasons behind server-side blocks and enhancing the granularity of our analysis.

We have identified 3.4 million explicit refusal pages using 147 fine-grained regular expressions that we built to capture various types of refusal pages in the entire set of fetching failures encountered by Common Crawl. Our analysis indicates that about 1.68% of websites present in a Common Crawl snapshot explicitly reject the Common Crawl's bot at least once, with major hosting providers and website builder platforms being the primary sources of server-side blocks. Later, we put forward some recommendations for treating refusals, namely adapting back-off patterns based on the status codes received and also distributing requests to individual domains across all fetcher nodes.

The paper is organized as follows: Section 2 provides background on Common Crawl. In Section 3, we discuss related works, followed by a detailed description of our methodology in Section 4. Our main findings are presented in Section 5, and Section 6 delves into the implications of these findings, offering recommendations based on our analysis. The paper concludes by acknowledging the current study's limitations and suggesting directions for future research in Section 7 before drawing final conclusions in Section 8.

2 COMMON CRAWL

The Common Crawl project is a non-profit initiative that compiles and publishes public web crawl datasets every few months. Common Crawl uses a custom version of Apache Nutch[2], which means it does not process dynamic content, such as non-HTML elements like images, CSS stylesheets, and JavaScript files.

Common Crawl is designed to be polite and adheres to robots.txt files and other web directives, such as *nofollow* tags. It waits for at least five seconds between requests to the same host and employs an exponential back-off strategy when encountering fetching errors. Additionally, it identifies itself clearly to web servers by sending a unique user-agent string ("CCBot") and contact information with each request.

[1]https://commoncrawl.org/

[2]https://nutch.apache.org

The fetching process is structured into 100 segments, processed sequentially over 14 days. These segments are subdivided into 40 partitions, each assigned to a single fetcher task. All URLs from a single host are assigned to the same partition, and hosts from the same domain are typically grouped in the same partition to reduce DNS resolution and robot.txt processing load. Fetching is then handled by twenty AWS EC2 spot nodes located in N. Virginia [10].

Common Crawl provides datasets in the Web Archive (WARC) format [6]. If the fetching process is successful, the record is added to the *warc* subset. However, if the fetching process fails, the record is added to the *non-200 responses* subset. All records are indexed in a columnar index, and each entry points to a specific WARC file path, offset, and record length. Every WARC file contains a *warcinfo* record that includes metadata about the file, such as the hostname of the machine that created the WARC resource.

3 RELATED WORK

There is a large body of work on coverage of active measurements. In the context of Internet-wide scanning, Leonard and Loguinov utilized a multi-origin scanning approach to distribute the scanning load and improve coverage [8]. Another study demonstrated geographic biases and the impact of scanning origin on host visibility [13]. With respect to coverage and visibility of web measurements, studies typically discuss variations in website contents when obtained by different crawling technologies and vantage points. One study showed how cloaking services deceive web crawlers [7]. Other researchers compared what real users experience on high-traffic websites and compared it with what those websites present to automated crawlers and found significant discrepancies in tracking and fingerprinting activities[15]. They also found the type of IP address used for crawling (e.g. cloud-based or residential) a key factor in this regard. Another work classified web crawler tools into three categories based on their capability to process dynamic content or mimic real browsers for circumventing anti-bot measures[2].

A group of studies specifically investigated the reasons behind website unavailability, most of which focused on censorship [5, 11]. Server-side blocking has mostly been studied in relation to geo-blocking [1, 9]. One relevant study, however, explores other reasons behind website unavailability as well by loading websites from various locations[12]. Using the status codes returned by Cloudflare as a proxy for the reason for the block, Tschantz et al. identified three causes of server-side blocking: GDPR compliance, geo-blocking, and ones returned due to security reasons. Our work is different from theirs since our work involves a web-scale dataset, while theirs was limited to websites that used Cloudflare. Secondly, we do not rely on HTTP status codes to identify blocks and their reason. Instead, we perform a semantic analysis of the page contents, which allows us to identify a much wider range of refusals. Finally, we provide a more detailed breakdown of the reasons for blocking based on the information provided within the page contents.

4 METHODOLOGY

4.1 Datasets

This work employs the CC-MAIN-2023-50 snapshot, which was compiled between the 28th of November and the 12th of December 2023 [4]. We use two pieces of this snapshot: *Columnar URL index* and *non-200 responses*. The non-200 responses archive contains all unsuccessful fetching attempts and contains 90000 WARC files and 3.43 TB of compressed data. The columnar index (cc-index-table) contains 900 parquet files and has a size of 0.28 TB.

Since our research focuses on analyzing and identifying refusals, we will only consider *response* records. Although cc-index-table provides an index for non-200 responses, we chose to download and parse all the WARC files within the non-200 responses set instead of locating and fetching individual records. We chose this because we are interested in page contents, and parsing the files sequentially provides better performance than iteratively locating and fetching individual WARC files. The index, however, is still used to identify successful fetching attempts for combining the success/failure records for each host.

4.2 Processing WARC files

We process records of type *response* and extract *hostname*, *WARC-Date*, *WARC-IP-Address*, *WARC-Target-URI*, and the raw response received from the web server. The raw response encompasses HTTP status codes, HTTP headers, and the page contents. We use the BeautifulSoup library to extract the page's textual content (lowercase) and PyASN (using BGP data from RouteViews project[3][4]) to obtain the ASN number of *WARC-IP-Address*. Finally, we use the tldextract library to extract the registered domain and also the Fully Qualified Domain Names (FQDN) from *WARC-Target-URI*. The resulting schema of the records is given in Table A1 of the Appendix.

4.3 Preliminary analysis and data pruning

Parsing the entire set of non-200 responses yields as many as 561 million records. Table 1 gives this set's breakdown of status codes. The preliminary exploratory analysis shows

[3]https://www.routeviews.org/routeviews/

[4]For Simplicity, we used the latest snapshot of the RIB archive of RouteViews when this stage of analysis was conducted. That was the snapshot with the date 2024-01-12, approximately a month after the last crawl day by Common Crawl for the respective snapshot.

Table 1: No. of status codes before and after pruning

Code	Description	All non200s	%	Pruned	%
301	Moved Permanently	235,772,654	41.97%	-	-
404	Not Found	175,006,242	31.16%	-	-
302	Found*	114,777,210	20.44%	-	-
307	Temporary Redirect	6,003,819	1.07%	-	-
403	Forbidden	5,855,588	1.04%	5,855,588	26.87%
308	Permanent Redirect	4,905,497	0.87%	-	-
410	Gone	4,338,335	0.77%	4,338,335	19.91%
500	Internal Server Error	3,379,744	0.60%	3,379,744	15.51%
303	See Other	3,344,468	0.60%	-	-
406	Not Acceptable	1,676,200	0.30%	1,676,200	7.69%
400	Bad Request	1,528,691	0.27%	1,528,691	7.02%
401	Unauthorized	1,190,637	0.21%	1,190,637	5.46%
429	Too Many Requests	708,933	0.13%	708,933	3.25%
Other	-	3,143,206	0.56%	2,854,781	13.11%
Total	-	561,631,224	100.00%	21,790,009	100.00%

* Previously named 302 Moved temporarily

us that certain status codes are unlikely to contain refusal content. Therefore, we prune the dataset in the first step to make the subsequent analysis less computationally intensive.

Table 1 indicates the most common status codes are among 3XX, 4XX or 5XX ranges and that the top three codes account for 93.57% of the total. Given that 3XX status codes generally signify redirection, which inherently does not imply a refusal but facilitates automatic redirection to a new location, we opted to exclude all entries with 3XX status codes.

The second most common status code among the non-200 response dataset is 404 (Not Found). Although this status code is not generally associated with refusals, we investigate potential refusals in this set to ensure that by discarding all records with status code 404, we will not miss a significant number of refusals. To do so, we manually inspect the records that contain keywords such as *blocked* or *banned* and find that HTTP status code 403 is the most common code used on pages containing some block. Next, we analyze the most common N-grams of length four among all records with code 403 and then manually sift through these N-grams to create a list of regular expressions (REs) that includes frequently used bigrams in 403 pages containing blocking content. After that, we search for these REs in 404 records and find about 4K records out of 25M. We argue that if 404 had been widely used for web refusals with human-readable messages, we would have expected the numbers to be significantly higher. Thus, we exclude those records from this analysis and are left with 21.7M records as shown in Table 1. Table A2 of the Appendix provides a more extensive breakdown of the pruned set.

4.4 Defining labels

We define *refusals* as any content returned by a website specifically intended to prevent the crawler from accessing the actual content. We specify three groups of labels to characterize the refusal content: *type*, *who*, and *reason*. Table 2 provides the first three labels, their descriptions and their values. The label *tag* will specify a provider's name (content delivery network, web application firewall, website builder, WordPress plugin, etc.) such as Cloudflare.

Table 2: Description and Possible Values for Labels

Label	Description	Values
Type	The type of refusal content	block, challenge, checking, require_js, other429s, none
Who	If type is *block*, indicates who is blocked	ip, you, request, none
Reason	The reason for refusal provided in the contents	security/malicious, excessive/suspicious, geo-block ip/asn reputation, none

If a website requests a user to complete a challenge like a captcha, we label the record as *challenge*. When the contents indicate a block with no option provided for getting unblocked, it will be labelled as *block*. *Checking* is assigned to cases where a website responds with a page that asks the user to wait while it checks the browser or connection, and *require_js* is when a website asks the visitor to enable JavaScript or disable ad-blockers. Finally, *other429s* is a type of block where the status code (429 Too many requests) signals that the access request is refused due to excessive requests or traffic.

Regarding *reason*, *security/malicious* is assigned to a record if page content implies that a security solution has blocked a request or if preventing harmful activities is cited as the reason for refusal. The *excessive/suspicious* label is used when content mentions spamming, scraping or information extraction or when a website rejects due to excessive hits. The *geo-blocking* label is allocated to contents where access denials are caused by the geographical location of the visitor. *IP/ASN reputation* is used for contents that mention negative IP history or its presence on a block list. Also, if page contents contain a reference to proxies, VPNs, or cloud IP addresses as the source of block, the record will receive an *IP/ASN reputation* label.

4.5 Extracting Regular Expressions

In this step, we build a set of long and precise regular expressions (REs), each capturing a specific cluster of similar refusal messages. Each regular expression will be manually assigned the *type*, *reason*, *tag* labels, if available.

We follow a two-stage process to create REs for capturing blocking and refusal messages. In the first stage, we begin by searching for common keywords like "blocked," "banned," "restricted," "limited," "denied," and "your ip" that are typically associated with such messages. We then manually inspect the records found and build loose REs that can capture various forms of refusals. This list is then supplemented in multiple cycles by adding new loose REs or modifying previously built ones to capture similar or slightly different strings.

In the second stage, we create more specific and longer REs using the ones built in stage one. If a secondary RE captures less than 50 records, we discard it and filter out the records found. Otherwise, we determine its labels (type, who, reason, tag) and filter the matching records out. We repeat this step until our manual inspection does not yield large clusters (more than 50 records). Finally, we add some less-fine-grained REs to the list to capture less common but diverse refusal messages.

We only create REs for contents that explicitly indicate server-side blocks. This means that pages without such a message or those that imply a path-specific refusal will not be included. However, there is one exception to this rule, which is a case with status code 406. In this case, more than a million records with a "Not acceptable" message generated by ModSecurity are considered a block in our method despite not containing semantically explicit refusal.

We built 147 REs, 141 fine-grained and specific to a certain class of refusal messages, and six loose REs. A breakdown of the labels (type and reason) of these REs is given in Tables A3 and A4 of the Appendix.

4.6 Analysis of the remaining records

We follow a two-step process to ensure that we do not miss a significant number of explicit refusals. First, we identify and label the most common non-refusal content. Then, we take a sample of the remaining records to estimate the number of uncaptured refusals.

First, we conduct an N-gram analysis to identify high-frequency word combinations that do not occur with refusals and then create REs for each combination to capture similar contents, e.g. those containing internal server errors. As this work focuses on refusals, we stop our manual analysis when the clusters found have less than 50 thousand occurrences. Eventually, we built 176 REs for non-refusal content, each annotated with a label. The details of these labels, their descriptions, and the number of REs in each set are provided in Table A9 of the Appendix.

In step two, we take a small sample from the remaining set. The sample is small enough for us to semantically analyse the records manually. The analysis focuses on language and content type. This allows us to estimate any explicit refusals that our method may have missed. The output of this process is an estimation of the number of explicit refusals that were not identified by our method.

4.7 Labelling records

While the REs we built for capturing refusals and non-refusals do not have considerable overlap, we process them in an ordered way to find refusals before capturing non-refusals. Moreover, the refusal REs are also applied in a certain order, with more fine-grained ones given higher priority for better precision. Once all refusals that fit the previous description are captured, we apply non-refusal REs and finally identify blank pages. The order of process is provided in Table A5 of the Appendix.

4.8 Augmenting tags by header analysis

We noticed that some refusal contents can be associated with a specific provider, such as Cloudflare or a particular web application firewall, even if the provider's name is not explicitly mentioned in the contents. To cover these cases, we analyse the `Server` headers of records captured by each regular expression. If we find an association between a refusal RE and a `Server` header, we assign a tag to that RE based on what the `Server` header implies. This association is only made if two conditions are both met: if 99% of all records captured by that RE have the `Server` header in question, and if there are more than a thousand FQDNs associated with those records. We use online sources to infer the provider's name from the `Server` header when it is not self-explanatory. For example, if we find `Pepyaka` in the `Server` header, we tag it as Wix[5].

4.9 Joining refusals with successful fetches

Once we have identified the records containing refusals, we analyze the mixture of successes and failures for each website. We made two choices to ensure high precision of analysis. First, we identify unique websites using FQDNs rather than registered domains since a single registered domain could encompass multiple subdomains hosted at multiple providers or running distinct web applications, resulting in divergent behaviours. Second, we source successful fetches for each FQDN from the cc-index because we are only interested in the occurrence of successful fetches and not what was fetched. The timestamp and the URI of a successful fetch can easily obtained from the index without fetching the full records. Finally, we employ a left join operation between the refusals and cc-index-table on equal FQDNs. This means that if an FQDN does not have success records, it will still be included in the joined set.

4.10 Failure Rate Analysis

After compiling the set of successes and failures for all FQDNs with at least a refusal, we proceed to calculate the percentage of failures for each FQDN. This metric is chosen because it

[5]https://webtechsurvey.com/technology/pepyaka

provides a consistent measure of failure frequency across all Uniform Resource Identifiers (URIs) and throughout the entire duration of the data set for each FQDN.

When the failure percentage is obtained for all FQDNs, we create failure percentage distributions to determine the proportion of websites that always reject Common Crawl's bot. Then, we refine our analysis by obtaining the same distribution per each status code, *type*, *reason* and *tag*.

A crucial requirement for the above analysis is assigning each FQDN a distinct status code, type, reason, and tag. Initially, we confirm that for almost all FQDNs, refusal records are associated with no more than one status code, *type*, *reason*, and *tag*. Then, we exclude FQDNs with multiple associations from the analysis so the failure percentage of an FQDN can be attributed to the corresponding category.

4.11 Temporal analysis

In the next phase, we quantify the length of time in which Common Crawl's access is denied. To do this, we define a metric called *first failure to next success*. This metric measures the time between a failed attempt to access a website (after a success) and the next successful attempt. Common Crawl uses an exponential back-off strategy to deal with access denials [10], but the details of this strategy are not documented. Using this metric, we can understand Common Crawl's back-off and retry patterns and determine how many back-offs (roughly corresponding to time spans) are required before access is granted following an initial denial. Since an FQDN may experience multiple instances of access refusal, we calculate this metric for all occurrences and then find the minimum time span for each FQDN.

We conduct the analysis for a time period of 0 to 3600 seconds. The reason behind choosing this time frame is twofold. Firstly, it covers the time length in which the majority of FQDNs see their next success record. Secondly, it ensures that all requests made to each FQDN are routed through a single IP address (see Section 5.6).

4.12 Analysis of Common Crawl's public IP

In this part of our analysis, we analyze the pattern of public IP addresses that are used to crawl websites. The goal is to determine how many public IP addresses were used in total and whether IP rotation was used to visit single FQDNs. To this end, we search for visitors' IP addresses reflected in page contents in the entire non-200 responses archive consisting of 561 million records. We first discard IPs that do not belong to Amazon (AS14618) and then accept only those IPs that were reflected in the highest number of FQDNs. Finally, we manually check the top IPs found to ensure that they are indeed visitor IP addresses reflected in page contents. Finally, we analyze one-hour windows of consecutive hits to each FQDN to determine whether the visitor's IP changes over that one hour.

5 RESULTS

5.1 General statistics

Table 3 summarizes the number of labelled and unlabeled records after the pruning stage (see Sections 4.3 and Section 4.7). In terms of the number of FQDNs, out of approximately 47.5 million FQDNs present in the Common Crawl's tabular index, about 31 million FQDNs have at least one record in the non-200 responses set. Upon pruning, the remaining FQDNs amount to 3.5 million, of which about 800k FQDNs were found to have at least one refusal.

Table 3: General Statistics of labels

Description	Count	Percentage (%)
Refusal	3,430,207	15.74
Non-refusal	9,817,878	45.06
Empty	4,222,824	19.38
Unlabeled	4,319,100	19.82
Total	21,790,009	100.00

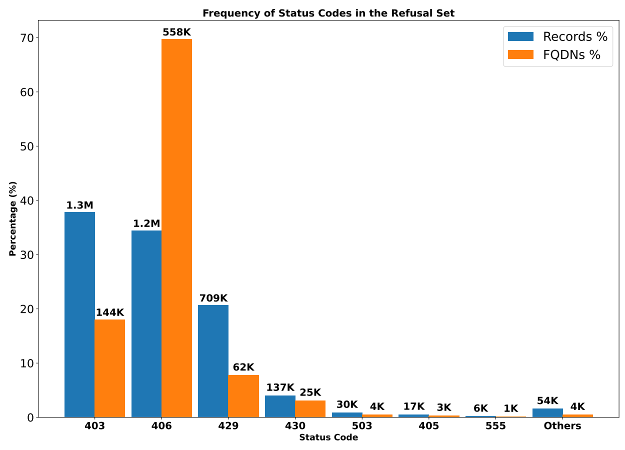
5.2 Breakdown of refusals

5.2.1 By status code. Figure 1 outlines the distribution of top status codes used for refusals, broken down by the number of records and FQDNs linked to each status code. Status code 403 (Forbidden) is the most frequently used code by the number of records. However, 406 (Not acceptable) is almost as prevalent in record count yet used in significantly more FQDNs, representing about 70% of all FQDNs with refusals. Further analysis in later sections will reveal that 406 (Not acceptable) is predominantly returned by ModSecurity (a web application firewall), with a single regular expression capturing the majority of these records.

Furthermore, the data shows that the third most common status code used to deny access is 429. It is important to note that our method interprets all records with status code 429 as refusals, which may explain the high record count for this code. It is also interesting to note the long tail of status codes employed for refusals (see Table A6 of the Appendix), indicating a wide range of codes used to deny Common Crawl's access.

5.2.2 By type. Table 4 shows the distribution of refusal types by the total number of records and the FQDNs associated with each type. The most common type of refusal is "block", which includes around 2.7M records and 740K

Figure 1: Frequency of top status codes in refusals



FQDNs. More than half of these refusals are related to ModSecurity. However, with ModSecurity refusals excluded, "block" would still be the most prevalent form of refusal.

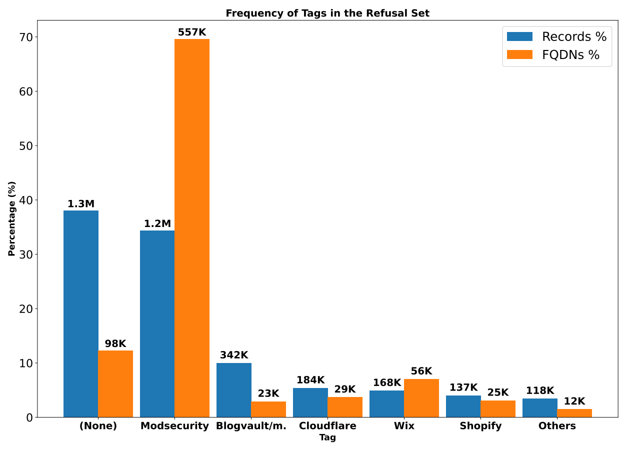
5.2.3 By reason. Table 5 categorizes the reasons for website refusals based on the number of records and FQDNs. The majority of the records do not specify a reason for denial. This is particularly common for refusals with a 406 status code from ModSecurity, which typically does not provide a detailed reason. The next most frequent reason is excessive/suspicious, which covers several cases, including contents indicating a server-side block due to too many requests or bot-like behaviour, as well as records with a 429 status code. The security/malicious category follows, covering refusals generated for security reasons or to prevent malicious activities. Other reasons like IP/ASN reputation and geo-blocking are less common, each representing less than one per cent of the records.

Table 4: Frequency of types by records and FQDNs

Type	Records		FQDNs	
	Count	Percentage (%)	Count	Percentage (%)
Block	2,717,399	79.22	739,602	92.48
Other 429s	428,031	12.48	30,043	3.76
Checking	143,194	4.17	22,969	2.87
Challenge	78,678	2.29	4,996	0.62
Require JS	59,772	1.74	1,888	0.24
(None)	3,133	0.09	276	0.03
Total	3,430,207	100.00	799,774*	100.00

* FQDNs with more than one type have been excluded.

Figure 2: Frequency of top tags in refusals



5.2.4 By tag. The distribution of top tags is shown in Figure 2. ModSecurity is the most common tag with respect to the number of records and FQDNs, indicating that many domains exhibit ModSecurity-generated refusals. The second most common tag, in terms of the number of records, is Blogvault/Malcare, a WordPress plugin. However, when it comes to the number of domains, Wix, Cloudflare, and Shopify have higher numbers than Blogvault/Malcare. The complete list of tags and their frequencies is given in Table A7 of the Appendix.

5.3 Case of Modsecurity

A regular expression targeting ModSecurity's refusal message identified over a million refusal records from 556K FQDNs and 488K registered domains, all returning a 406 status code. Analysis of the *WARC-Target-IP* indicates that these sites predominantly reside on four ASNs, pointing to the involvement of a few major hosting providers. Furthermore, examination of the `Server` headers shows that 93% of these domains were hosted on Apache, with the remainder using Cloudflare, further suggesting a possible connection between them in terms of their hosting provider.

Table 5: Frequency of reasons by records and FQDNs

Reason	Records		FQDNs	
	Count	Percentage (%)	Count	Percentage (%)
(None)	1,741,041	50.76	651,251	81.43
Excessive/suspicious	859,309	25.05	89,145	11.15
Security/malicious	803,538	23.43	56,180	7.02
Ip/asn reputation	19,047	0.56	2,248	0.28
Geo-block	7,272	0.21	917	0.11
Total	3,430,207	100.00	799,741	100.00

Table 6: Top ASNs hosting ModSecurity-labeled FQDNs and major hosting companies operating on them

ASN	Name	FQDN No.	Percentage
46606	Unified Layer *Major hosters:* 55.78% Bluehost, 18.47% Hostgator, 6.13% Hostmonster 4.27% ResellerClub, 3.88% Justhost, 1.63% Domain.com, 1.00% Site5	351,390	63.11%
19871	Network Solutions *Major hosters:* 83.72% Hostgator	143,041	25.69%
13335	Cloudflare	40,681	7.31%
394695	PDR *Major hosters:* 55.02% ResellerClub, 12.05% HostGator, 11.12% BlueHost, 9.26% BigRock	16,815	3.02%
Others	-	4,096	0.88%
Total	-	556,833	100.00%

To investigate this matter further, we set out to identify the hosting providers responsible for this group of refusals. We used the following sources for this purpose: Caida AS-to-organization data set [3] to find the organization associated with each ASN, whois information from Hurricane Electric BGP toolkit[6], and DNS records of a sample of ModSecurity-labelled FQDNs.

Table 6 lists the four autonomous system numbers and their names. From the Caida AS-to-organization data set, we found that Unified Layer is owned by a company called BlueHost[7]. Whois information for the other two (Network Solutions and PDR) indicates they are both associated with a company called Newfold[8], formerly known as Endurance International Group, a conglomerate hosting company with many subsidiaries[9] including BlueHost, HostGator, Host-Monster[10], ResellerClub[11], Domain.com, Network Solutions, BigRock, JustHost[12], Site5[13], etc.

Aiming to determine whether they were indeed hosted by companies associated with Newfold, we conducted a retroactive analysis of DNS records of a sample of FQDNs. Random samples were from each ASN as follows: 4096 FQDNs from Unified Layer, 4028 from Network Solutions, and 3325 from PDR. As this analysis was done four months after Common Crawl's snapshot was compiled, some domains could have expired or moved to other hosting companies. Thus, we only selected FQDNs whose first *A record*[14] still pointed to the respective ASN on which it was hosted before. Then, we extracted NS records of FQDNs and DNS pointer records (PTR) of their first *A records*. If either contained the name of a company of interest, we considered the FQDNs to be hosted at that hosting company.

Table 6 presents the results of this analysis. At least 92% of FQDNs sampled for Unified Layer, 83% of FQDNs from Network Solutions, and at least 87% of FQDNs on PDR were found to be hosted with companies associated with Newfold. This suggests that the ModSecurity configuration is possibly consistent across all these domains and has been centrally adopted by the parent company as an anti-bot security measure.

[6]https://bgp.he.net/

[7]www.bluehost.com

[8]https://newfold.com

[9]https://newfold.com/brands

[10]Endurance International Group is mentioned in https://www.hostmonster.com/privacy_policy

[11]webhostbox.net used as name server[14]

[12]Endurance I.G. is mentioned in https://www.justhost.com/terms/user-agreement

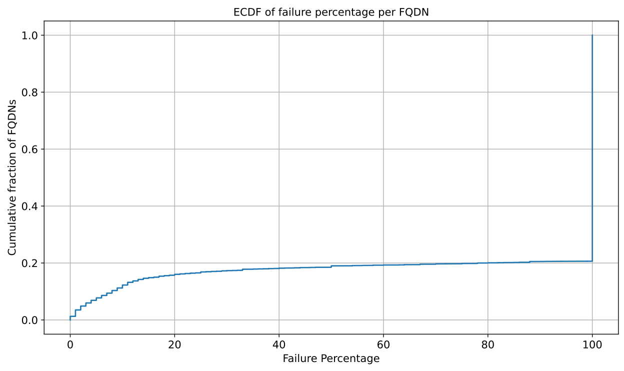
[13]site5.com, owned by web.com

[14]By first *A record* we mean the first record obtained from domain resolution. While the order may change with every attempt, what mattered to us was selecting domains that still were using the same hosting provider.

5.4 Failure percentage

5.4.1 All FQDNs. The graph illustrated in Figure 3 represents the ECDF of failure percentages across all FQDNs. It shows that nearly 80% of FQDNs always block Common Crawl as their failure percentage is recorded at 100%. This indicates that for this group of FQDNs, all the requests were rejected immediately, suggesting that crawling behaviour had not been a factor in the refusal decision.

Figure 3: ECDF of failure percentage per FQDN



5.4.2 By status code. Table 7, provides the average failure rate and standard deviation for commonly encountered status codes, sorted by FQDN frequency. The table highlights that status codes 406, 405, 444, 418, 510, and 451 have failure rates close to 100%, indicating a non-transient nature, while status codes 430, 555, 500, and 403 have the lowest average failure rates, indicating a transient refusal pattern for these codes.

Figure 4 presents an ECDF of failure rate categorized by status codes. Only status codes with over 1000 FQDNs and less than 99% failure are included for clarity. Clearly, refusals with status code 430 (as also seen in the table) have the most transient refusal behaviour. Nearly 80% of the FQDNs with

Figure 4: ECDF of failure % per FQDN by status code

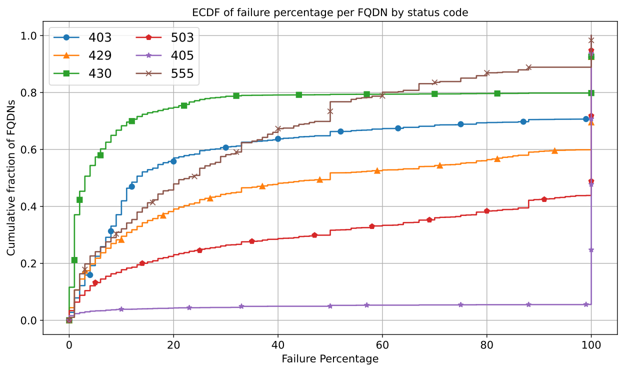
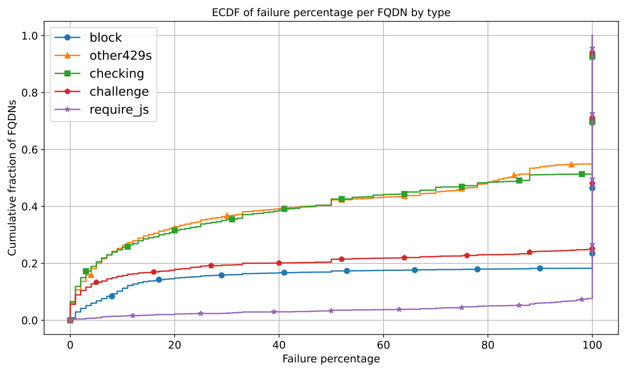


Figure 5: ECDF of failure % per FQDN by type



this status code have about 30% of failure or less. Status code 430 is seemingly an unofficial client error specific to Shopify and is returned by the server to indicate that too many HTTP requests are being made[15]. Further analysis confirmed this as this status error was predominantly associated with Shopify, and the description of the error also matches the pattern seen in the graph (transient refusal).

Table 7 shows that code 555 has the second-lowest failure rate at 33.58%. However, the graph reveals that code 403 has more FQDNs with less than 33% failure. This is because a third of all FQDNs with code 403 always refuse, while the same figure for code 555 is just over 10%. Therefore, excluding FQDNs with 100% failure, code 403 is more transient.

Another observation is that status code 429 had a 100% failure rate in 40% of FQDNs, which was unexpected. A closer look at the data revealed that code 429 was not exclusively used for rate limiting. In many FQDNs, it was returned right away with the first request, which contradicts the behaviour expected for rate limiting.

[15]https://http.dev/430

Table 7: Average and standard deviation of failure percentage for top status codes

Code	Avg. Failure %	Std. Dev.	FQDN Count
406	99.76	4.22	557,618
403	39.89	41.70	144,347
429	53.16	43.16	62,418
430	24.38	38.83	24,697
503	70.02	39.86	3,919
405	95.18	20.48	2,558
444	99.24	7.47	1,562
555	33.58	32.31	1,071
401	87.57	27.32	342
500	34.58	44.12	198
418	97.83	13.07	173
510	98.45	9.66	141
451	99.24	8.73	131
400	65.25	46.12	125

5.4.3 By type. Figure 5 shows an ECDF of failure rate for different refusal types. The "Require JS" category has the highest failure rate. We attribute it to Common Crawl's inability to process JavaScript. The "Block" category has the second-highest failure rate, largely influenced by FQDNs associated with the 406 status code from ModSecurity. Notably, "Checking" and "Other 429s" exhibit a similar failure pattern across FQDNs and are more transient than other types. The average and standard deviation for failure rates can be found in Table A10 of the Appendix.

5.4.4 By reason. Table 8 shows the average failure rate for each reason for refusal. Geo-blocking has the highest average failure rate but is not as high as expected. Further investigation revealed that for certain websites, geo-blocking messages appear for specific URIs, while other URIs within the same website successfully load, which leads to a seemingly transient refusal behaviour. The second highest average failure rate is associated with refusals labelled as IP/ASN reputation. Further investigation into these instances showed these refusals sometimes resolve when Common Crawl visits the same FQDN days later, suggesting these refusals could be triggered by specific public IPs used by Common Crawl, and changing the IP would potentially resolve them. Finally, security/malicious refusals are seemingly less persistent than excessive/suspicious ones, which might be unexpected. Figure A1 of the Appendix provides an ECDF of failure rate broken down by refusal reason.

5.4.5 By tag. Table 9 shows the failure rates for different categories of tags. It seems that FQDNs tagged as ModSecurity tend to deny Common Crawl's access almost all the time, indicating that the blockage may not be caused by the

Table 8: Average and standard deviation of failure percentage for reasons

Reason	Avg. F. %	Std. Dev.	FQDN No.
Excessive/suspicious	45.65	44.10	89,643
Security/malicious	33.64	40.32	56,295
IP/ASN reputation	56.83	39.43	2,258
Geo-block	87.37	29.45	917

Table 9: Average and standard deviation of failure percentage for tags

Tag	Avg. F. %	Std. Dev.	FQDN No.
Modsecurity	99.76	4.19	556,751
Wix	34.10	38.26	56,176
Cloudflare	51.89	43.26	29,411
Shopify	24.38	38.83	24,696
Blogvault/Malcare	30.16	38.75	23,228
Cleantalk	74.92	33.12	3,951
Cloudfront	81.58	37.43	2,869
Wordfence	60.32	43.51	2,215

behaviour of the crawler. To determine whether the user agent or Amazon's IP address is triggering the block, we accessed some of these FQDNs from the University of Twente's campus network using "CCBot" as the user agent. In all cases, the websites returned the same refusal message generated by ModSecurity. We then retried accessing the same FQDNs, this time with a real browser-like user agent [16], and all of them loaded successfully. Further, we tried fetching the same websites using cURL, and all opened. This suggests that either a bot-like user agent is triggering the block or "CCBot" is specifically blocked.

Except for ModSecurity, other tags seem to evaluate visitor behaviours before refusing, as their failure rates are not close to 100%. As mentioned earlier, the figures for Shopify match those of status code 430 (refer to Table A6), as Shopify exclusively uses code 430. The ECDF of the failure percentage by tags can be found in Figure A2 in the Appendix.

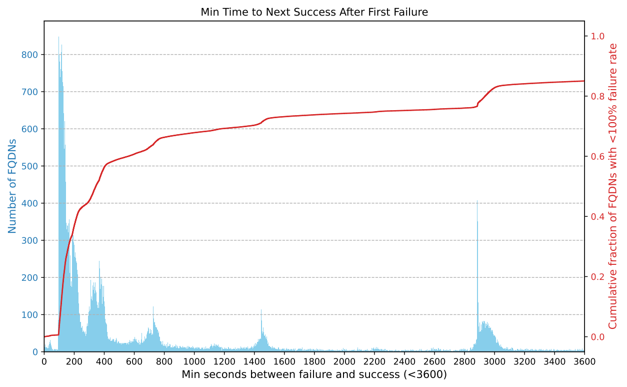
5.5 Temporal analysis

In this section, we will analyze how quickly a refusal disappears. It is important to note that since we are only dealing with transient failures in this section, all figures regarding the fraction of FQDNs refer to the fraction of FQDNs with a non-100% failure rate, not the entirety of them.

Figure 6 presents the minimum time from the first failure to the next success exclusively for FQDNs with a failure rate below 100%, as noted before. For this graph, we calculated

[16]Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36

Figure 6: Minimum T from first failure to next success (<3600 seconds)



the minimum duration from an initial failure to the first subsequent success for each FQDN. These FQDNs are then grouped together based on this minimum duration. Only instanced with a duration of less than 3600 seconds are shown.

The graph displays peaks at certain intervals, indicating that there are certain times when large groups of FQDNs allow access to Common Crawl again. The size of a spike seen at time T indicates the number of FQDNs that allowed Common Crawl's access after T seconds had elapsed since a refusal. The timing of the spikes is likely related to the back-off and retry pattern used by Common Crawl. Moreover, the vertical dispersion of data points around peaks on the graph suggests that Common Crawl does not reattempt at exactly the same intervals after seeing a failure.

The graph shows that the largest peaks occur at around 100 and 2900 seconds, meaning that a larger fraction of domains allow Common Crawl access after these intervals. It is interesting to note that Common Crawl appears to wait for approximately 100 seconds and makes four retries before backing off and retrying again after around 700 seconds, which clearly shows an exponential pattern.

The graph also displays an ECDF, where the right X-axis represents the cumulative fraction of FQDNs with transient behaviour. From the graph, we can see that around 85% of FQDNs with transient refusals are covered within a 3600-second period. Furthermore, the spikes on the bar graph are reflected by the jumps in the ECDF.

5.5.1 By status code. The graph displayed in Figure 7 shows the ECDF of the shortest time duration between the first failure and the next success by refusal status code having less than 90% failure rate. The graph shows that status code 430, if transient, is the quickest to disappear. Over half of FQDNs with status code 430 (excluding the 100% failure ones)

Figure 7: ECDF of minimum T from first failure to next success (<3600 seconds) by status code

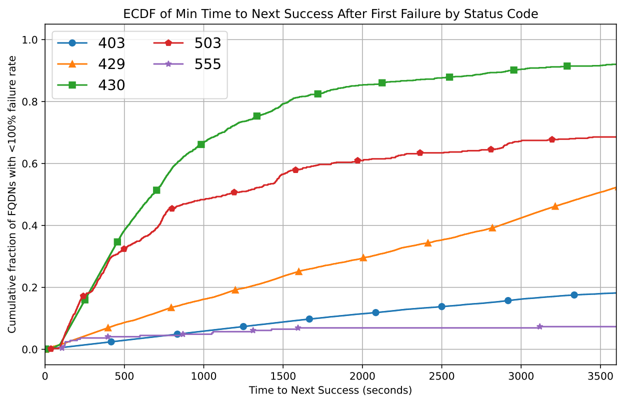
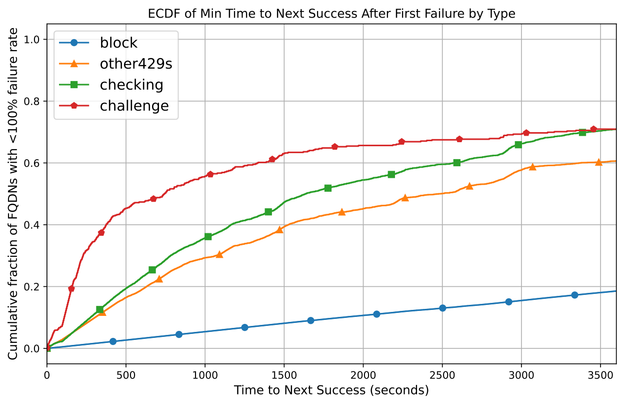


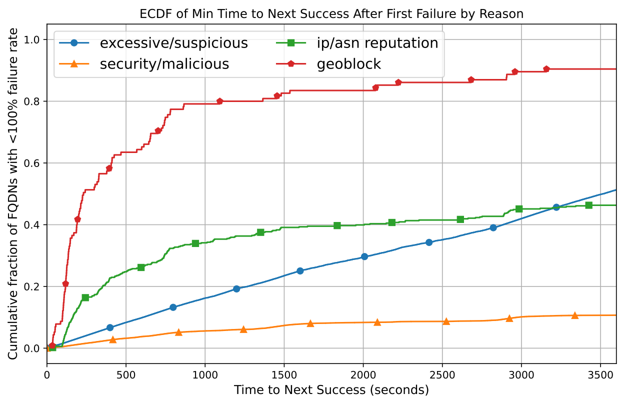
Figure 8: ECDF of minimum T from first failure to next success (<3600 seconds) by type



are accessible within 15 minutes from the first failure. Status code 503 is the second quickest to resolve, and refusals with status code 555 are found to be the slowest status code to resolve, where less than 10% of FQDNs with this status code become accessible after an hour.

5.5.2 By type. The ECDF of the minimum time length between the first failure and the next success, broken down by refusal type (those with <90% failure rate), is shown in Figure 8. Refusals labelled as "block" are seen to take the longest time to resolve. After one hour, only 20% of domains with this label allow access. In contrast, "Challenge" FQDNs are the quickest group to unblock, with half of the domains successfully fetched after only 15 minutes.

Figure 9: ECDF of minimum T from first failure to next success (<3600 seconds) by reason



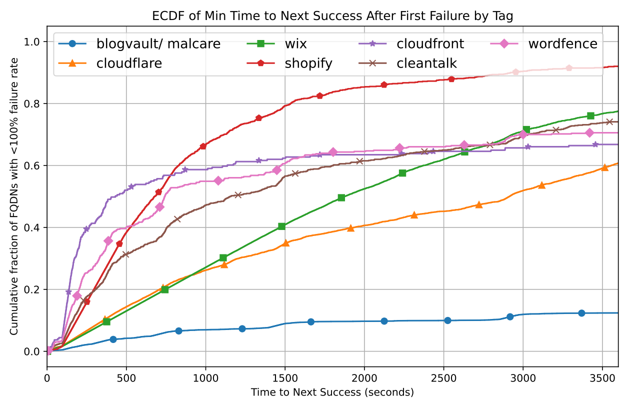
5.5.3 By reason. Figure 9 shows that transient geo-block refusals are often resolved quickly, and nearly 80% of them are resolved within 1000 seconds. This could be due to geo-blocks specific to certain website paths, as discussed in Section 5.4. A rather unexpected observation from Figure A1 is that close to half of FQDNs with transient IP/ASN reputation refusals become accessible after one hour of the initial block. This is surprising because, as we will see in Section 5.6, Common Crawl does not visit a single FQDN with more than one IP over an hour span. There are two possible explanations. First, the IP/ASN reputation messages may not be generated solely on the basis of the IP used, but rather excessive hits may also contribute to their occurrence. Second, refusals that cite IP/ASN/VPN/Proxy in their refusal message could be path-specific, where only certain paths of websites are protected. In such cases, subsequent hits might fetch unprotected paths and result in successful fetches.

5.5.4 By tag. Figure 10 shows that CloudFront is the quickest to permit access, with more than half of FQDNs allowing access after only 500 seconds. Blogvault/Malcare is the slowest, with less than one-fifth of domains allowing Common Crawl access after one hour of the first refusal. Shopify is not the fastest until 500 seconds, yet after one hour, Shopify has the highest percentage of FQDNs that unblocked Common Crawl. Cloudflare is the second slowest, with just over 60% of domains becoming accessible after an hour. It is important to note that all the FQDNs considered for this plot eventually allow access again, although it may take longer, anywhere between one hour and 14 days.

5.6 Public IP addresses used for crawling

Out of 561M records extracted from the non-200 responses archive, a total of 1.2M records within 154K FQDNs contained

Figure 10: ECDF of minimum T from first failure to next success (<3600 seconds) by tag



IP addresses belonging to Amazon. As described in Section 4.12, we determined the most frequent IP addresses in terms of the number of reflecting FQDNs to distinguish between visitor IPs and other strings, such as version numbers which still could be mistaken for Amazon IPs. Figure 11 shows the top candidate IPs (labelled with letters for anonymity) reflected in page contents. Out of a total of 662 Amazon IPs found, eighteen have significantly higher frequencies. The IP with the lowest frequency (3.5.21.16 labelled as U) was ruled out as it was found to be a version number, while the rest (twenty items) were actual IP addresses reflected in page contents. The total number of reflecting FQDNs for each IP and their prefixes is presented in Table A12 of the Appendix.

According to Common Crawl's documentation, twenty AWS EC2 spot instances are used as fetching nodes [10], which may correspond to the public IP addresses identified. At first, we assumed that the hostname field (refer to Section 4.2) could be used as a unique identifier for each fetcher host.

Figure 11: Top extracted IPs by number of reflecting FQDNs

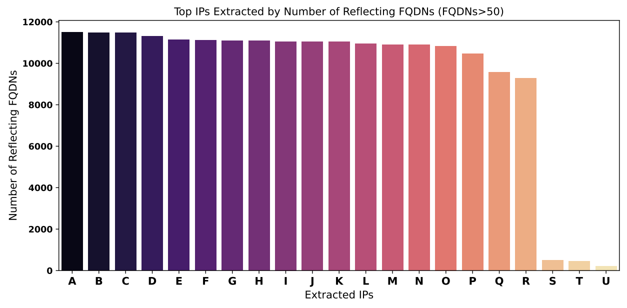
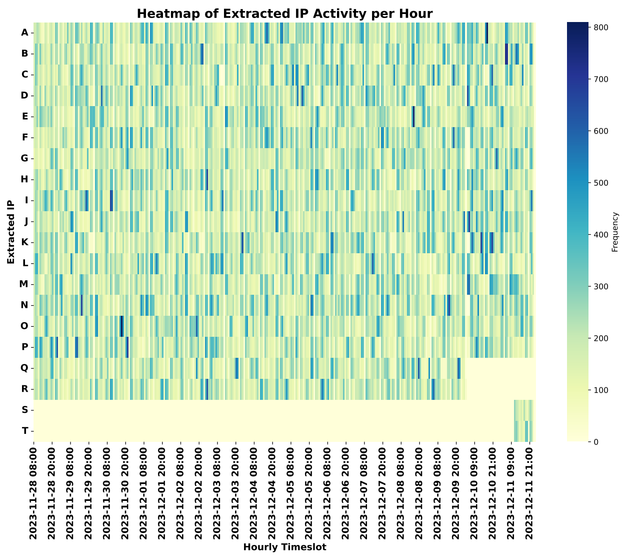


Figure 12: Heatmap of extracted IPs



However, we found no one-to-one relationship between the hostname and the reflected public IP addresses. Therefore, to understand IP usage patterns, we generated a heatmap, shown in Figure 12, that shows the frequency of encountering these addresses in one-hour time slots over 14 days. The heat map reveals that the sixteen IPs were consistently used throughout fourteen days. However, two IPs stopped appearing on pages at a certain point in time and were replaced by two other IPs the following day. This pattern suggests that each public IP may correspond to a single fetcher node, as most are used continuously for the entire 14-day period.

Finally, we examined whether IP rotation is used to access a single FQDN. We found that public IP address changes occur only in around 0.6% of FQDNs within an hour. Thus, Common Crawl does not seem to switch its public IP address for accessing an FQDN within one-hour windows.

5.7 Rest of the records

5.7.1 Non-refusal records. In total, 9.8M records, which account for 45% of the pruned set, were found to contain non-refusal contents. A breakdown of the status codes and labels assigned to the non-refusal records set are given in Tables A8 and A9 of the Appendix, respectively.

5.7.2 Unlabeled records. We took a random sample of 385 records to estimate the number of missed refusals in the 4.3M unlabeled records. The frequency of status codes seen in the entire unlabeled and sampled sets are shown in Figure A3 of the Appendix.

We first analyzed the language of the contents in our sample set and then determined the category of page contents. Table 10 shows the distribution of content categories broken down by English and non-English pages. For a detailed breakdown of content language, see Table A11 of the Appendix.

Table 10: Frequency Distribution of Content Categories in Sampled Unlabeled Records

Category	English	Non-English	Total	Missed
Standard 403 Forbidden	144	0	144	0
Item not found	15	43	58	0
Internal Error	25	27	52	0
General Access Denial	20	20	40	0
Business-specific text	9	16	25	0
Log-in Required	11	11	2	0
Site down/Maintenance	4	7	11	0
Unknown	2	8	10	0
Invalid request format	2	5	7	0
Not Acceptable	0	3	3	0
Explicit Bot refusal	2	1	3	2
Expired Subscription	0	2	2	0
Require JS	2	0	2	2
Challenge	1	1	2	1
CDN Errors	1	0	1	0
Rate-limiting	0	1	1	0
Geo-block	0	1	1	0
Implicit Bot refusal	1	0	1	0
Total	**239**	**146**	**385**	**5**

Table 10 indicates that the most common pages found in the sample set were standard 403 forbidden pages in English. Upon manual inspection, all general access denials were found to be not exclusively targeting Common Crawl. Also, none of the URLs containing "Not acceptable" messages was server-side blocks intended for crawlers.

However, we found three bot refusals in our sample set that our method had missed: two in English and one in Russian. A closer inspection revealed that one was present in 514 records with only one FQDN, while the other was found in 4,744 records across 17 FQDNs. This suggests that although there might be missed refusals, they are likely specific to particular websites and do not occur frequently enough to be detected by our method.

We also found two "Require JS" instances in the sample set. Identifying this type of refusal can be challenging as many pages contain text about enabling JavaScript as a recommendation and it is hard to distinguish between recommendations and mandatory requirements for accessing website contents. In addition, our method missed two "Challenge" pages, one in English and one in Russian. The English cases were found to be using unusual keywords that our method did not cover. Furthermore, our approach missed cases such as one "Rate-limiting" case in Chinese and a "Geo-block" in French. We also observed an implicit bot refusal where the website responded with the same format of response when visited by curl, indicating a user-agent-based blocking being in place, which was impossible to detect using our method.

In summary, out of 385 records sampled, five (1.3%) could have been detected assuming an exhaustive Latin alphabet-based method. Extrapolating the findings to the complete dataset suggests that about 56,000 refusal records may have been missed, which would have increased the refusal set by about 1.63%. Considering refusals in other languages, a total of nine records (2.33%) were not identified, projecting to about 101k records and a 3% increase in the size of refusal records. Although these figures are not insignificant, they fall within an acceptable margin of error for this scale of data analysis.

6 DISCUSSION

Our findings indicate that 1.68% of all examined FQDNs display refusal behaviours. ModSecurity accounts for 1.17% of these cases, where a few sister hosting companies adopt aggressive anti-bot measures. This observation showcases the large impact of centralized security measures implemented by hosting providers and platforms such as Wix, Cloudflare, and Shopify on crawler visibility.

In addition, the results indicate that a wide range of HTTP status codes are used to reject crawlers. This variety poses a challenge for Common Crawl and similar web crawlers, as there is no standard way to signal refusals consistently across the web. Notably, the misuse of status code 429 and the employment of an unofficial 430 code by Shopify exemplify the need for clearer, universally accepted standards for crawler interactions. Such a standard could formulate status codes for certain signals, such as asking bots to stop or proceed at a lower rate.

We also found that about 80% of the refusal cases were not triggered by crawling behaviour, suggesting that dynamic crawling behaviour has a less pronounced impact on Common Crawl's visibility than its static characteristics, such as user agent. At least 556K websites (tagged with ModSecurity) would not have blocked Common Crawl had it used a browser-like user agent, although this would negatively affect Common Crawl's good Internet citizenship in identifying itself. An alternative could be for Common Crawl to negotiate with the hosting companies to allow its bot access.

Our analysis also highlights the potential benefits of employing IP rotation strategies. The current configuration of Common Crawl does not support extensive IP rotation within short intervals for single FQDNs, limiting its ability to circumvent blocks applied to IP addresses. An alternative approach could involve distributing requests across different fetchers to minimize the probability of being blocked by leveraging all IP addresses simultaneously for each FQDN.

Based on our findings, one enhancement for web crawlers, in general, could be using an adaptive back-off strategy based on encountered status codes. For instance, immediate halting of attempts upon encountering status codes prone to persistent blocks (such as 406, 444, 510, or 451) would avoid unnecessary resource allocation. Moreover, our analysis of refusal temporal patterns indicates that domains typically grant access either shortly after initial denial or after a significant delay, around 2900 seconds later. This suggests that retrying after 100 seconds and then resuming attempts after a 30-minute interval could be effective. Tailoring this approach based on specific status codes could potentially further optimize performance. For instance, shorter back-off periods could be applied for codes like 430 and 503, while longer waits may be more suitable for codes like 555 and 403. Additionally, reattempting requests after encountering transient but slow-disappearing status codes could involve assigning tasks to different fetcher machines to incorporate IP rotation into the back-off mechanism. This optimization maximises resource utilisation and reduces unnecessary load and retries.

7 LIMITATIONS AND FUTURE RESEARCH

Uneachability level. Web crawlers may fail to access a website due to network-level or application-level unreachability. Our research focuses on application-level unreachability as Common Crawl does not record network failures.

Sampling Bias in Common Crawl. Common Crawl's algorithm for prioritizing domains (Harmonic centrality [10]) may introduce bias in the analysis due to over-representation of high-ranking domains. Additionally, central domains may have more records, making it difficult to accurately identify failure rates for FQDNs with fewer records.

Methodological Constraints. Our method of identifying refusals requires a manual analysis that concentrates mainly on messages written in English or related languages. This approach is not comprehensive as it does not detect refusals written in other languages, as discussed in Section 5.7.

Content-Based Analysis Limitations. Our analysis was limited to explicit refusals. However, we came across instances where the content was blank, but HTTP headers indicated some form of block. Since our method relied exclusively on explicit content, we did not consider these cases. There were also instances where it was difficult to distinguish between standard access denials and explicit refusals. However, since verification of these refusals would need live interaction with the website, we did not take these cases as refusals.

Path-specific refusals. Our method assumes that all paths of an FQDN exhibit the same refusal behaviour. However, in some instances, refusals may only happen at a specific path and not in others. Due to the limited number of records available for most FQDNs, we were unable to distinguish paths, as some paths had only one record. Therefore, we assumed that all paths of an FQDN exhibit a consistent refusal pattern and do not differ, which may not always be the case.

Reasons for Refusals. We assigned a *reason* tag to each RE used for refusal to identify the reason behind the refusal clearly. However, it is possible that the actual reason for the refusal may differ from what was stated on the website.

In future, employing automated approaches for content analysis could be advantageous. One approach could be using Large Language Models (LLMs) or other natural language processing methods to analyze page contents' semantics across various languages. Additionally, incorporating HTTP header examination in refusal identification could be an improvement. Moreover, in cases where the contents or other indicators, such as HTTP headers, are insufficient to determine if an access denial is targeted at Common Crawl, conducting active scanning of the URL could be useful to verify that the refusals Common Crawl faced would not happen with a crawler that mimics a real browser.

8 CONCLUSION

This study aimed to examine the refusals that the Common Crawl project encountered while crawling the web. Our work revealed that 1.68% of Fully Qualified Domain Names (FQDNs) displayed refusal behaviours, with some large hosting providers being a significant contributor. It also revealed the different shapes and forms of refusal responses used by websites to deny crawler access. Understanding these patterns, such as the relationship between status codes and blocking persistence, can improve crawler efficiency. Additionally, the findings show that in the majority of cases, blocking Common Crawl happens irrespective of its crawling behaviour, suggesting that refusals are predominantly triggered by the properties of the crawler.

However, the study has its limitations, particularly in the methodology adopted for refusal identification, which primarily focuses on analyzing page content semantics. Future research could leverage natural language processing techniques and consider a broader array of features, including HTTP headers, to enrich the analysis and understanding of web access refusals.

Overall, the study provides a deeper understanding of the refusals faced by web crawlers and highlights the importance of this knowledge in enhancing the tools essential for internet research and measurement. This study serves as a step towards optimizing web crawling practices in a constantly evolving web landscape.

REFERENCES

[1] Sadia Afroz, Michael Carl Tschantz, Shaarif Sajid, Shoaib Asif Qazi, Mobin Javed, and Vern Paxson. 2018. Exploring Server-side Blocking of Regions. *ArXiv* abs/1805.11606 (2018). https://api.semanticscholar.org/CorpusID:44131334

[2] Syed Suleman Ahmad, Muhammad Daniyal Dar, Muhammad Fareed Zaffar, Narseo Vallina-Rodriguez, and Rishab Nithyanand. 2020. Apophanies or Epiphanies? How Crawlers Impact Our Understanding of the Web. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA, 271–280. https://doi.org/10.1145/3366423.3380113

[3] Center for Applied Internet Data Analysis (CAIDA). 2024. AS Organizations Dataset. https://catalog.caida.org/dataset/as_organizations. (2024). Accessed: 2024-04-04.

[4] Common Crawl. [n. d.]. November/December 2023 Crawl Archive Now Available. https://www.commoncrawl.org/blog/november-december-2023-crawl-archive-now-available. ([n. d.]). Accessed: 2023-03-12.

[5] Alexander Darer, Oliver Farnan, and Joss Wright. 2018. Automated Discovery of Internet Censorship by Web Crawling. In *Proceedings of the 10th ACM Conference on Web Science (WebSci '18)*. Association for Computing Machinery, New York, NY, USA, 195–204. https://doi.org/10.1145/3201064.3201091

[6] International Internet Preservation Consortium. 2024. WARC, Web ARChive file format. Web Page. (2024). https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.0/

[7] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean-Michel Picod, and Elie Bursztein. 2016. Cloak of Visibility: Detecting When Machines Browse a Different Web. In *2016 IEEE Symposium on Security and Privacy (SP)*. 743–758. https://doi.org/10.1109/SP.2016.50

[8] Derek Leonard and Dmitri Loguinov. 2010. Demystifying Service Discovery: Implementing an Internet-Wide Scanner. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*. Association for Computing Machinery, New York, NY, USA, 109–122. https://doi.org/10.1145/1879141.1879156

[9] Allison McDonald, Matthew Bernhard, Luke Valenta, Benjamin VanderSloot, Will Scott, Nick Sullivan, J. Alex Halderman, and Roya Ensafi. 2018. 403 Forbidden: A Global View of CDN Geoblocking. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. Association for Computing Machinery, New York, NY, USA, 218–230. https://doi.org/10.1145/3278532.3278552

[10] Sebastian Nagel. 2023. Common Crawl: Data Collection and Use Cases for NLP. http://nlpl.eu/skeikampen23/nagel.230206.pdf. (2023). Accessed: 2023-11-07.

[11] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. 2020. ICLab: A Global, Longitudinal Internet Censorship Measurement Platform. In *2020 IEEE Symposium on Security and Privacy (SP)*. 135–151. https://doi.org/10.1109/SP40000.2020.00014

[12] Michael Carl Tschantz, Sadia Afroz, Shaarif Sajid, Shoaib Asif Qazi, Mobin Javed, and Vern Paxson. 2018. 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)*. USENIX Association, Baltimore, MD. https://www.usenix.org/conference/foci18/presentation/tschantz

[13] Gerry Wan, Liz Izhikevich, David Adrian, Katsunari Yoshioka, Ralph Holz, Christian Rossow, and Zakir Durumeric. 2020. On the Origin of Scanning: The Impact of Location on Internet-Wide Scans. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. Association for Computing Machinery, New York, NY, USA, 662–679. https://doi.org/10.1145/3419394.3424214

[14] Yamini. 2012. Now Live: Geo-Located Hosting & New Single-Domain Shared Hosting Architecture. (18 December 2012). https://blog.resellerclub.com/now-live-geo-located-hosting-new-single-domain-shared-hosting-architecture/ Accessed: 2023-04-20.

[15] David Zeber, Sarah Bird, Camila Oliveira, Walter Rudametkin, Ilana Segall, Fredrik Wollsén, and Martin Lopatka. 2020. The Representativeness of Automated Web Crawls as a Surrogate for Human Browsing. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA, 167–178. https://doi.org/10.1145/3366423.3380104

APPENDIX

Table A1: Description of constructed fields in parsing

Field Name	Field Description
fetch_time	Record's timestamp extracted from WARC-Date
hostname	Hostname of machine which generated the WARC file
WARC-IP-Address	IP address of target webserver
WARC-Target-URI	URL of the recorded page
status_code	HTTP code extracted from raw payload
http_headers	HTTP headers extracted from raw payload
html_text	Textual contents of the received page
asn	AS number of WARC-IP-Address
registered_domain	Registered domain extracted from WARC-Target-URI
fqdn	FQDN extracted from WARC-Target-URI.

Figure A1: ECDF of failure percentage per FQDN by reason

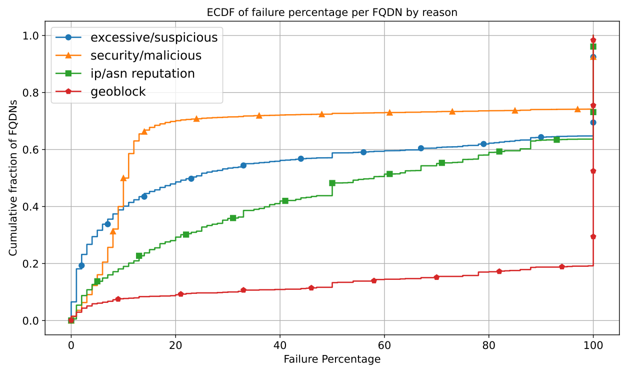


Figure A2: ECDF of failure percentage per FQDN by tag

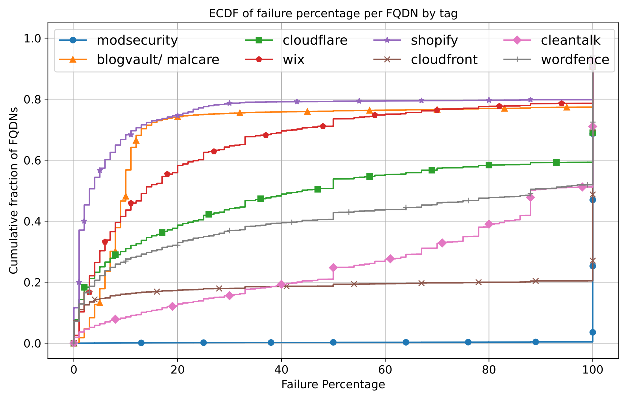


Table A2: Frequency of status codes in pruned set

Status Code	Record Count	Percentage (%)
403	5,855,588	26.87
410	4,338,335	19.91
500	3,379,744	15.51
503	1,812,300	8.32
406	1,676,200	7.69
400	1,528,691	7.02
401	1,190,637	5.46
429	708,933	3.25
502	343,984	1.58
522	152,643	0.70
430	137,609	0.63
504	116,847	0.54
520	112,555	0.52
402	97,097	0.45
444	41,874	0.19
405	39,872	0.18
409	30,404	0.14
202	30,097	0.14
423	28,483	0.13
521	24,316	0.11
508	14,296	0.07
204	12,850	0.06
412	12,168	0.06
418	7,772	0.04
555	7,506	0.03
Others	89,208	0.41
Total	21,790,009	100.00

Table A3: Freq. of Types

Type	Freq.
block	123
checking	11
challenge	8
require_js	3
(No Type)	2

Table A4: Freq. of Reasons

Reason	Freq.
(No Reason)	49
excessive/suspicious	49
security/malicious	33
ip/asn reputation	9
geoblock	7

Table A5: Order of Processing of refusal REs

Order	Type	Who	Reason
1	Block	IP	Provided
2	Block	You	Provided
3	Block	Request	Provided
4	Block	IP	Not provided
5	Block	Not provided	Provided
6	Challenge	-	-
7	Checking	-	-
8	Block	Not provided	Not provided
9	Loose REs	-	-
10	Require_js	-	-
11	Other 429s	-	-
12	Non-refusals	-	-
13	Empty text	-	-

Figure A3: Frequency of status codes in the unlabeled set

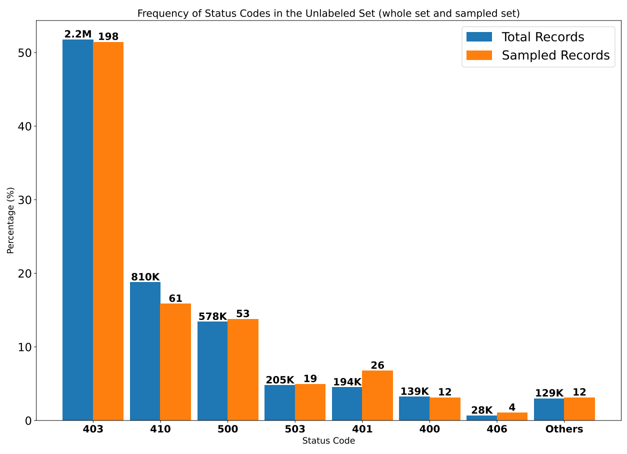


Table A6: Frequency of status codes in refusals

Type	Records		FQDNs	
	Count	Percentage (%)	Count	Percentage (%)
403	1,296,614	37.80	144,247	18.03
406	1,181,016	34.43	557,610	69.72
429	708,933	20.67	61,911	7.74
430	137,389	4.01	24,696	3.09
503	29,731	0.87	3,907	0.49
202	23,721	0.69	964	0.12
405	17,448	0.51	2,557	0.32
401	10,131	0.30	342	0.04
410	6,433	0.19	67	0.01
400	5,858	0.17	125	0.02
555	5,544	0.16	1,070	0.13
444	3,878	0.11	1,547	0.19
418	998	0.03	173	0.02
493	497	0.01	14	0.00
451	390	0.01	131	0.02
500	354	0.01	197	0.02
510	352	0.01	141	0.02
501	294	0.01	3	0.00
201	216	0.01	40	0.01
424	165	0.00	53	0.01
509	93	0.00	6	0.00
422	83	0.00	2	0.00
508	21	0.00	16	0.00
484	12	0.00	2	0.00
456	11	0.00	2	0.00
420	8	0.00	2	0.00
455	7	0.00	2	0.00
409	5	0.00	2	0.00
502	2	0.00	2	0.00
203	2	0.00	1	0.00
522	1	0.00	1	0.00
Total	3,430,207	100.00	799,833	100.00

Table A7: Frequency of tags in refusals

Type	Records		FQDNs	
	Count	Percentage (%)	Count	Percentage (%)
(None)	1,302,997	37.99	98,124	12.26
Modsecurity	1,177,945	34.34	556,751	69.58
Blogvault/malcare	342,060	9.97	23,228	2.90
Cloudflare	184,387	5.38	29,411	3.68
Wix	167,808	4.89	56,176	7.02
Shopify	137,389	4.01	24,696	3.09
Cloudfront	38,115	1.11	2,869	0.36
Cleantalk	30,089	0.88	3,951	0.49
Wordfence	16,955	0.49	2,215	0.28
Shieldpro	8,634	0.25	878	0.11
Deflect.ca	8,338	0.24	285	0.04
Google	4,550	0.13	76	0.01
Defender	2,932	0.09	320	0.04
Bitninja	2,482	0.07	583	0.07
Wexbo	2,440	0.07	184	0.02
Crowdsec	1,276	0.04	90	0.01
Tiger protect	944	0.03	105	0.01
Zb block	239	0.01	81	0.01
Virusdie	126	0.00	19	0.00
Cachewall	115	0.00	15	0.00
Ninjafirewall	98	0.00	25	0.00
Spamfirewall	89	0.00	25	0.00
Link11	42	0.00	4	0.00
Cidram	36	0.00	8	0.00
Zero spam	35	0.00	6	0.00
Blockscript	34	0.00	7	0.00
Security pro	27	0.00	3	0.00
Sysadminok	14	0.00	0	0.00
Aapenal	11	0.00	4	0.00
Total	3,430,207	100.00	800,139	100.00

Table A8: Frequency of status codes in non-refusals

Status Code	Record Count	Percentage (%)
410	2,533,661	25.81
500	1,815,199	18.49
503	1,388,313	14.14
400	1,237,856	12.61
403	932,536	9.50
401	674,589	6.87
406	381,823	3.89
502	309,285	3.15
522	152,342	1.55
520	112,210	1.14
504	89,158	0.91
402	74,727	0.76
409	23,467	0.24
423	23,135	0.24
521	22,566	0.23
508	13,837	0.14
523	6,877	0.07
525	4,785	0.05
526	3,564	0.04
405	2,998	0.03
424	2,306	0.02
414	1,631	0.02
905	1,359	0.01
530	1,060	0.01
451	1,009	0.01
Others	7,585	0.08
Total	9,817,878	100.00

Table A9: Frequency of Non-refusal Labels with Descriptions

Description	RE count	Record Count	Percentage
Internal/application/server errors	45	2,200,717	22.42
Not existing page/resource/etc.	11	1,914,100	19.50
High-frequency keywords irrelevant to refusal such as “skip to contents”, “forgot your password”, “please renew your subscription”	22	1,359,222	13.84
Invalid request, invalid link, bad request, etc.	7	944,237	9.62
Page or post being removed or deleted, etc.	15	877,183	8.93
Need for authentication, login, etc.	15	711,936	7.25
Path- or resource-specific forbidden messages	16	512,973	5.22
Service unavailable, maintenance, etc.	7	437,080	4.45
Cloudflare error pages	18	338,878	3.45
Bad gateway errors	3	291,976	2.97
Not acceptable errors	3	151,088	1.54
CDN-related errors such as "unable to reach the origin server"	5	58,972	0.60
Payment required errors	1	17,614	0.18
Censorship-related (such as Russian censorship)	6	1,158	0.01
Bandwidth limit exceeded errors	1	744	0.01
Total	175	9,817,878	100.00

Table A10: Average and standard deviation of failure percentage for types

Type	Avg. Failure %	Std. Dev.	FQDN Count
Block	84.26	34.14	739,728
Other 429s	60.16	42.8	30,525
Checking	60.71	42.88	23,062
Challenge	79.80	37.73	5,006
Require Js	96.48	15.68	1,888

Table A11: Frequency Distribution of Content Languages in Sampled Unlabeled Records

Language	Count Each	Count Total
English	239	239
German	28	28
Chinese	18	18
French, Russian	15	30
Spanish	12	12
Italian	8	8
Japanese, Dutch	7	14
Korean, Czech, Unknown	4	12
Turkish	3	3
Arabic, Catalan, Estonian, Vietnamese, Romanian, Polish, Bulgarian	2	14
Portuguese, Swedish, Thai, Basque, Persian, Croatian, Lithuanian	1	7
Total	-	385

Table A12: Candidates for Common Crawl's public IPs, their prefixes and number of reflecting FQDNs

IP Label	Prefix	No. of reflecting FQDNs
A	3.224.0.0/12	11495
B	44.192.0.0/11	11474
C	35.168.0.0/13	11471
D	3.224.0.0/12	11310
E	3.80.0.0/12	11132
F	3.224.0.0/12	11109
G	34.224.0.0/12	11084
H	3.224.0.0/12	11082
I	44.192.0.0/11	11055
J	35.168.0.0/13	11041
K	44.192.0.0/11	11037
L	18.204.0.0/14	10946
M	34.224.0.0/12	10901
N	18.204.0.0/14	10892
O	18.204.0.0/14	10836
P	18.204.0.0/14	10459
Q	18.204.0.0/14	9561
R	44.192.0.0/11	9272
S	35.168.0.0/13	507
T	44.192.0.0/11	464
U	3.5.21.0/24	212