A Machine Learning Approach to Detect Differential Treatment of Anonymous Users

Isabel Wagner(✉)

Cyber Technology Institute, De Montfort University, Leicester, UK
isabel.wagner@unibas.ch

Abstract. Anonymous Internet use is essential to safeguard against mass surveillance and to protect privacy online. Tor Browser and the Tor anonymity network provide an effective and convenient way to browse the Internet anonymously. However, many websites make browsing inconvenient for anonymous users either by blocking access entirely, by blocking access to some functionality, or by using CAPTCHAs to make access more difficult. Prior work has relied on heuristics to study the extent to which anonymous users are treated differently. However, these heuristics either underestimated blocking or required extensive manual labeling. To address these shortcomings, here we propose a machine learning approach to detect when anonymous users are treated differently. We train binary and multi-class classifiers based on six feature sets and find that they perform very well on our test data (F1 scores 100%–94% for binary and 97%–84% for multi-class classifiers). Applying these classifiers to data collected from browsing 1,000 sites, including visits to subsites and executing search functionality, via 100 Tor exit nodes we find that 16.7% of landing pages inconvenience anonymous users, compared with 15.2% of subsites and 3.8% of search result pages. In particular, we find that websites hosted by Akamai, EdgeCast, and Cloudflare have significantly higher block rates than average, and that blocking of search results is dominated by Google which displays a block page or CAPTCHA for 39.8% of search result pages.

Keywords: Tor · Anonymous communication · Tor exit blocking · Machine learning · CAPTCHA

1 Introduction

Anonymity is the property that individual users of a communication systems are not identifiable [17]. The most well-known anonymity system for the Internet is The Onion Router (Tor) [7]. Proposed in 2004, Tor is now available as Tor Browser, which makes it as easy to use as other browsers. However, people who want to use the Internet anonymously are subject to more restrictions and inconveniences than regular Internet users. For example, some websites categorically deny access to users of Tor browser, while some make Tor users solve

V. Atluri et al. (Eds.): ESORICS 2022, LNCS 13555, pp. 3–23, 2022.
https://doi.org/10.1007/978-3-031-17146-8_1

CAPTCHAs before allowing access. This differential treatment of Tor users is not always aimed specifically at Tor users. Instead, the IP addresses of Tor exit nodes can be labeled as suspicious because many Tor users share the same exit node [22]. However, regardless of the technical reason for blocking of Tor users, anonymous users are experiencing inconvenience and disadvantage as a result.

Prior work has studied the nature of Tor exit node blocking [22] and how the network and application layer contribute to the discrimination against Tor users [9]. These studies have also estimated the prevalence of Tor exit blocking based on a sample of the top 500 or 1,000 websites. However, these studies have relied on heuristics and manual labeling to identify which websites are blocking Tor users.

In this paper we use machine learning to classify when websites are blocked because (1) heuristics can be imprecise, in particular, we show in Sect. 3.1 that heuristics do not perform well when predicting which websites are blocked; (2) existing heuristics do not distinguish between blocked websites and websites that show a CAPTCHA; and (3) manual labeling does not scale well. Based on our machine learning classifiers, we then estimate the prevalence of Tor exit blocking, taking into account visits to subsites in addition to landing pages as well as searches and visits to search result pages. Specifically, we make the following contributions[1]:

- We engineer six feature sets based on paired visits to websites with Firefox and Tor Browser, including features extracted from HTTP requests and responses, HTML source code, and screenshots. Each feature set restricts the number of features it uses to allow for classification when only limited data collection can be performed.
- We train binary (blocked/unblocked) and multi-class (blocked/CAPTCHA/unblocked) classifiers for each feature set and evaluate classifier performance. We find that the classifiers perform very well on our test data, with F1 scores of 100%–94% for binary and 97%–84% for multi-class classifiers.
- We estimate the prevalence of Tor exit blocking and analyze to what extent block and CAPTCHA rates depend on the visit type, characteristics of Tor exit nodes, and characteristics of the websites, based on data collected from 998 websites and a sample of 101 exit nodes. We find average block rates of 16.7% for landing pages, 15.2% for subsites, and 3.8% for searches. The characteristics of exit nodes have no statistically significant influence on block or CAPTCHA rates, whereas significant differences can be observed depending on a website's hosting provider and category. Specifically, websites hosted on Akamai, Cloudflare, and EdgeCast have higher block rates than average, and websites from Fashion/Beauty (51%) and Finance/Banking (29%) have the highest block rates.

The remainder of this paper is organized as follows: we present related work in Sect. 2, explain the methodology for feature selection and engineering, as well as classifier training and performance evaluation in Sect. 3, and evaluate the

[1] Code and data for this paper are available at https://gitlab.com/iwagner/tor-inconvenience.

prevalence of Tor exit blocking in Sect. 4. We discuss limitations of our work in Sect. 5 and conclude in Sect. 6.

2 Related Work

The main areas of relevant related work are (1) work that studies Tor and Internet censorship, and (2) work on Internet measurement and associated methods.

Tor is an overlay network that routes user traffic over three intermediate hops or relays [7]. Routing information including source and destination is encrypted in onion layers so that none of the relays can link the identity of the user to the identity of the visited web service, thereby providing anonymity to the user. Although Tor is often portrayed as a tool that provides anonymity to criminals, most of its 8 million daily users [10] likely rely on Tor for safe, privacy-friendly, and uncensored Internet access. This point is underpinned by the fact that less than 4% of traffic on Tor visits onion services [10] which include the dark web (but also anonymous access to regular websites such as search engines). In addition, it has been shown that Wikipedia edits made by Tor users are of similar quality as edits made by non-anonymous users [25].

The characteristics and performance of Tor, as well as attacks against the Tor network, have been studied extensively. For example, researchers have proposed machine learning classifiers to distinguish Tor traffic from other Internet traffic [5,8,23], as well as traffic fingerprinting techniques to identify search terms in encrypted Tor traffic [15]. The two studies most similar to our work are [9, 22]. Both studies used heuristics to determine when Tor users are blocked from accessing a website, based on HTTP status codes [9] and perceptual hashes of screenshots [22], respectively. We describe these heuristics in detail in Sect. 3.1. The main limitations of these studies are their reliance on heuristics and their selection of websites which does not consider subsites, i.e., internal pages that are not a domain's front- or landing page.

Another group of related work studies Internet censorship [13] and geoblocking [12]. Like our work, they are concerned with detecting when access to certain websites is available and when it is not. However, they focus on differences related to the vantage point, i.e., which country a website is accessed from, whereas we focus on access via Tor. The heuristics proposed in [12,13] are nevertheless useful for detecting blocking, and we integrate some of them as features into our machine learning classifiers.

Many Internet measurement studies have been published in the last decade. We follow the commonly used method of automated website crawls followed by statistical data analysis [27]. Websites are usually selected from a toplist, such as Alexa or Tranco [18], which rank the most popular websites. Measurement studies increasingly recognize that it is important to study subsites in addition to landing pages because subsites have been found, for example, to include more advertising [2] and more trackers [26]. To the best of our knowledge, the only toplist that currently includes subsites is Hispar [2] which consists of around 2,000 websites with 50 subsites each.

Machine learning is often used in the context of measurement studies to overcome the limitations of heuristics. For example, machine learning has been proposed to discover privacy leaks in mobile network traffic [19], to automate the detection of trackers [20], or to categorize the advertisements served to users [21]. To the best of our knowledge, machine learning has not yet been used to detect when access to websites is restricted.

3 Methodology

To create classifiers that can detect when Tor users are treated differently from non-anonymous users, we follow three steps: (1) we collect and label training data; (2) we define and select features; and (3) we train and tune the classifiers.

3.1 Collection and Labeling of Training Data

Our training data consists of paired website visits, once with Tor Browser and once with Firefox. We automate Firefox and Tor Browser with Selenium and tor-browser-selenium [1], respectively. For each website visit, we record the HTML source, a screenshot, and a HAR archive of HTTP requests and responses using the browser add-on HAR Export Trigger [14]. We select two Tor exit nodes for half of the visits so that we acquire two pairs of visits for every three website loads. In addition, before data collection we manually verify that traffic from the two exit nodes results in some blocking behavior. Specifically, we verify that we obtain CAPTCHAs when searching on google.com and when visiting zillow.com. By doing this, we increase the number of block pages in our data set and thereby reduce the amount of training data we have to label manually.

We visit a total of 600 websites: the top 150 sites from the Tranco toplist [18], 100 sites sampled uniformly at random from ranks 150–500, 50 sites sampled from ranks 1000–1500 (two exit nodes each), and 300 sites sampled from the Hispar toplist (one exit node) [2]. In addition, we visit three subsites for each website. For sites from the Tranco list, we sample three links from the website's landing page, and for sites from the Hispar list, we sample three subsites from the published list. The subsites are sampled prior to data collection to ensure that they are consistent between all visits. The rationale for including subsite visits is that they may trigger more blocking behaviors than landing pages, similar to them including more advertising and tracking [2,26]. We use the Tranco and Hispar lists instead of Alexa to improve reproducibility. In particular, the Alexa list does not allow to cite specific list versions, does not include subsites, and will be discontinued in 2022. Tranco aggregates rankings from different sources which makes the ranking more consistent and less open to manipulation [18].

To trigger additional blocking behaviors, similar to [22], we detect search boxes on each website and submit one search query per detected search box. However, in contrast to prior work, we did not attempt to log into websites: creating a fresh account for each website would have been too time-consuming, and the method of authenticating with Google OAuth credentials used by Singh et al. in 2017 [22] did not work anymore because Google now blocks logins when the browser is controlled by automated software.

In total, we collect 3,409 paired visits, covering 509 topsites, 1,350 subsites, and 535 searches. The number of successfully visited topsites is smaller than 600 because some domains, such as highly-ranked windowsupdate.com, do not serve HTML pages. The final distribution of ranks in our training data, according to Tranco, is shown in Fig. 1.

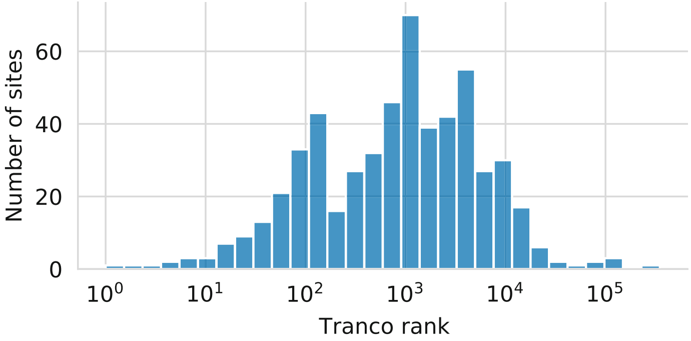


Fig. 1. Rank distribution of 509 sites in training data

Label	Count
same	2837
block	257
captcha	44
missing	44
slow	110
nocookie	18
cookie	4
other	95

Fig. 2. Label distribution in training data set.

Labeling. We created a Python GUI for labeling the collected data (see Fig. 8 in Appendix B). The GUI presents the two screenshots from a paired visit and allows the creation of eight labels, indicating whether the visit from Tor Browser shows *no substantial differences*, a *block page*, a *captcha*, missing elements such as login buttons or search functions, missing elements due to timeouts, a cookie banner where Firefox shows none, no cookie banner where Firefox shows one, or other substantial differences such as localization. To clean our data set for training, we include only visits with the first three labels and disregard visits with the other labels. Figure 2 shows the number of visits for each label.

Heuristics vs Manual Labels. We compare three previously proposed heuristics[2] to our manual labels. These heuristics classify paired visits into *blocked* and *not blocked*. We use our *same* label for the *not blocked* case, and compare two options for the *blocked* case: (1) using only our *block* label, and (2) combining the visits with *block* and *captcha* labels.

The first heuristic, labeled *Khattak1* in Table 1, classifies visits as blocked if Firefox receives a 200 (OK) status code, but Tor Browser receives a non-200 level status code [9]. The *Khattak2* heuristic classifies visits as blocked if Firefox is unblocked while Tor Browser is blocked, where a visit counts as blocked if the request timed out, was rejected, or received an HTTP response status code 400 or

[2] For convenience of referring to them, we name each heuristic after the first author of the corresponding paper.

higher [9]. Finally, the *Singh* heuristic relies on perceptual hashing of screenshots [22]. Perceptual hashing maps images to hashes such that the hashes are similar if the images are similar. Visits are classified as blocked if the difference between perceptual hashes is higher than a threshold (0.75 in [22]), or when Firefox receives a 200 status code but Tor Browser receives a status code indicating an error (400 or higher). Visits are classified as not blocked if the perceptual hash difference is below a threshold of 0.4. The remaining visits remain unclassified; [22] manually label these cases.

Table 1. Performance of three heuristics on our manually labeled data set, showing the F1 scores for each heuristic predicting *blocked* and *not blocked* labels.

Heuristic	"Blocked" = *block*			"Blocked" = *block+captcha*		
	Khattak1	Khattak2	Singh	Khattak1	Khattak2	Singh
Blocked	0.253	0.374	0.729	0.249	0.371	0.744
Not blocked	0.934	0.938	0.935	0.927	0.932	0.934
Support	3094	3094	724	3138	3138	735

Table 1 shows the performance of these heuristics in terms of their F1 scores, assuming our manual labels as ground truth. The performance of the *Singh* heuristic depends on the perceptual hashing algorithm and its parameters. Here, we use the Python ImageHash library[3] [4]. The number of cases for the *Singh* heuristic is lower because we omit all cases that, according to the heuristic, would have to be labeled manually. As a result, we expect that the Singh heuristic combined with a manual labeling step would perform better than what Table 1 indicates. However, manual labeling is costly, which is why a machine learning approach is desirable.

Table 1 shows that all heuristics are very good at predicting the *not blocked* label (F1 scores of 93–94%), but have significant weaknesses in predicting the *blocked* label (F1 scores of 25–74%). In practice, this means that the heuristics are likely to underestimate the number of blocked cases. Overall, this comparison indicates that if a machine learning approach can improve on the prediction of the *blocked* label, it can enable more accurate and fully automated labeling of the differential treatment of Tor users.

3.2 Feature Selection

We use three main sources of features for our classifiers: the HTML source code, HTTP requests and responses, and screenshots. Based on these feature groups, we construct six feature sets: unrestricted (F1); all feature groups without features based on differences between Firefox and Tor Browser (F2); only features from HTML source code and screenshots (F3); only features from HTTP requests and responses (F4); only features from HTML and screenshots without differences (F5); and only HTTP features without differences (F6).

[3] Parameter settings: hash size = 32, high frequency factor = 10.

Screenshot-Based Features. For screenshot-based features, we take inspiration from [22] and compute perceptual hashes[4] and wavelet hashes[5] for each screenshot. We also compute the differences between hashes for paired visits. Because we use machine learning, we do not have to select manual thresholds for these hashes. In total, we use 14 screenshot-based features.

HTML-Based Features. We extract four groups of features from the HTML source. First, we focus on the size of the HTML source. Because block pages are often shorter than non-blocked websites [12], we compute the byte size of the HTML source and the absolute and relative differences between sources sizes. In addition, we compute a unified diff between the sources of paired visits and count the number of differences.

Second, we focus on HTML tags. Differences in tag frequencies can indicate differences in page structure, which can indicate presence of a block page or CAPTCHA. We therefore extract HTML tag frequencies for all tags, specifically the number of and difference between tags present in HTML sources [12].

Third, we focus on JavaScript inclusions. This can help detect CAPTCHAs that are served from the same included JavaScript. Accordingly, we extract the sources, domains, and paths of included JavaScript as well as their differences. We then convert each set of JavaScript information (i.e., sources, domains, paths, and their differences) into a bag-of-words representation using the CountVectorizer from scikit-learn. The bag-of-words representation results in a matrix of token counts which indicates which pieces of JS information were present for each visit.

Fourth, we focus on the website text. We extract the website's text from the HTML source and compute the textual similarity for paired visits [13]. To do this, we construct a list of the words contained in each visit and estimate the Jaccard similarity between the two lists using the MinHash algorithm [3] implemented in the Python datasketch library [28]. We then compute the difference in website text as the list of words that appear in the Tor Browser visit, but not in Firefox. We use website text and text differences to construct page signatures [12], i.e., term frequency–inverse document frequency [24] feature vectors with 1- and 2-grams. In total, we use 4 features based on HTML source size, 1,410 features based on HTML tags, 18,159 features from JavaScript information in bag-of-words representation, and 478,945 features from website text in Tf-Idf representation.

HTTP-Based Features. For features from HTTP requests and responses, we first focus on status codes and extract the first received status code [9], the number and percent difference in received 200-level, 300-level, and 500-level status codes, and the frequencies of all received status codes as well as their differences. Next, we extract the number of and difference in sent requests and received responses, the number of requests that contain a (previously set) cookie

[4] Parameter settings for perceptual hash sizes and high frequency factors: (8,4), (16,8), (32,10), and (32,16).

[5] Parameter settings for wavelet hash sizes: 8, 16, 32.

value, and the number of responses that set a cookie. This results in a total of 73 HTTP-based features.

3.3 Classifier Training and Tuning

We pursue three goals when training our classifiers: (1) we expect our binary classifiers to exceed the performance of the three heuristics; (2) we expect our multi-class classifiers to accurately separate block pages from CAPTCHAs; and (3), we expect the classifiers to perform well on restricted feature sets, in particular: on HTML-only features (F3) because these features are easier to record than HTTP requests and responses; on features that do not include differences (F2, F5, F6) because this reduces the reliance on paired samples; and on HTTP-only features (F4) because this would enable classification of paired samples from the Open Observatory of Network Interference (OONI) [16].

A challenge for the training phase is that our training data is imbalanced: we have 11x more non-blocked samples than samples of block pages, and 64x more non-blocked samples than CAPTCHAs. To address this challenge, we select algorithms that are insensitive to imbalanced training data, such as decision trees and ensemble methods. Specifically, we use most of the algorithms implemented in the sklearn.tree and sklearn.ensemble modules: DecisionTreeClassifier, ExtraTreesClassifier, RandomForestClassifier, AdaBoostClassifier, and GradientBoostingClassifier. In addition, we apply appropriate metrics to evaluate the performance of the classifiers, i.e., F1 scores instead of accuracy. We report macro F1 scores to give equal weight to each label, irrespective of the number of samples.

To select the parameter settings for each classifier, we perform a grid search with 10-fold cross-validation for each algorithm. In addition, we use recursive feature elimination and tree-based feature elimination to train classifiers that rely on a smaller number of features. For each feature set, we then select the algorithm, hyperparameter settings, and feature elimination method that results in the best classifier performance. We train classifiers on a stratified sample of 70% of our data set and reserve 30% as a test set to evaluate classifier performance.

Tables 2 and 3 show the classifier performance for binary and multi-class classification, respectively (detailed performance results are in Appendix A). Overall, classifier performance is very satisfactory and clearly meets the three goals set out above. We can see that binary classification always performs better than multi-class classification, and that classification performance decreases with increasingly restricted feature sets. In addition, we note that performance for prediction of the *same* label is almost always better than for the *block* and *captcha* labels. This may be caused by the comparatively small number of training samples with the *block* and *captcha* labels. All of our classifiers perform better than the three heuristics introduced in Sect. 3.1.

4 Results: Differential Treatment of Tor Users

We now apply these classifiers to a large data set of paired samples collected in October–December 2021 (Sect. 4.1). For binary classification, we present results

Table 2. Performance of best classifiers for binary classification on *test* data set

Feature set	F1	F2	F3	F4	F5	F6
classifier	Random Forest	Decision Tree	Random Forest	Ada Boost	Gradient Boosting	Extra Trees
feature elimination	Recursive	Tree	recursive	–	recursive	tree
number of features	10	4573	131	81	45	10
block+captcha	1.000	0.979	1.000	0.971	0.978	0.882
same	1.000	0.998	1.000	0.997	0.998	0.988
macro F1	1.000	0.988	1.000	0.984	0.988	0.935

Table 3. Performance of best classifiers for multi-class classification on *test* data set

Feature set	F1	F2	F3	F4	F5	F6
classifier	Ada Boost	Decision Tree	Ada Boost	Gradient Boosting	Ada Boost	Gradient Boosting
feature elimination	tree	tree	recursive	–	recursive	–
number of features	3549	4624	131	81	683645	41
block	1.000	0.941	1.000	0.967	0.876	0.894
captcha	0.900	0.900	0.900	0.842	0.842	0.625
same	0.998	0.993	0.998	0.996	0.988	0.988
macro F1	0.966	0.945	0.966	0.935	0.902	0.836

using feature set F1 (all features) with recursive feature elimination (Sects. 4.2, 4.3 and 4.4). For multi-class classification, we present results using feature set F5 (HTML features and screenshots, no features based on differences) because the classification results matched best with a sample of manually labeled instances (Sect. 4.5). Importantly, the statistically significant effects described below are present in classifiers based on all six feature sets, even though the precise block rates predicted by each classifier vary slightly.

4.1 Data Collection

To collect this data set, we use the same methods as described in Sect. 3.1, with the exception of the selection of sites and the selection of Tor exit nodes.

To make our selection of subsites reproducible, we use the Hispar toplist [2] and select 1,000 sites with five subsites each. In addition, we attempt to find search boxes following the heuristics in [22] and execute one search for each search box. Overall, we successfully collected data for 998 topsites, a total of 3,992 subsites, and 1,574 searches executed on 587 different topsites.

Similar to prior work [22], we sample 101 Tor exit nodes (˜8%). However, in contrast [22], we add a purposive element to our sampling. Specifically, we analyze the countries where exit nodes are located and sample one exit node per country (52 countries corresponding to 52 nodes). Then we sample one additional node from every country with at least 5 exits (23 additional nodes), 15 exits (14 nodes), 25 exits (6 nodes), 50 exits (4 nodes), and 100 exits (2 nodes). As a result, the two countries with more than 100 exit nodes (US and the Netherlands) are each represented with 6 nodes. We ensure that all sampled exit nodes come from a unique Autonomous System (AS).

Two exit nodes stopped offering exit node services during the data collection. One was replaced with another node from the same country, while the other was removed because it was the only node from its country.

4.2 Block Rates by Visit Type

We distinguish three types of visits to websites: a visit to the front page or landing page, a visit to a subsite, and a visit to a search result page after having submitted a search query. To analyze differential treatment of Tor users, we focus on the *block rate*, i.e., the fraction of visits that appear blocked to the user. On average across all visits to landing pages, subsites, and searches, we observe a block rate of 12.9%.

Landing Pages. For landing pages, we find that 16.7% of visits are blocked. This is less than the 20% reported by Singh et al. in 2017 [22], but more than the 6.8% reported by Khattak et al. in 2016 [9]. It is likely that the heuristic used in [9] underestimates the block rate, as has already been argued by [22]. Singh's heuristic, due to its reliance on manual labeling, is less likely to under- or overestimate the block rate. Our lower estimate may instead be due to a change in block rates over the last 5 years, a difference in which sites were sampled ([22] used the top 500 from Alexa), or a difference in the sampled exit nodes.

Figure 3 shows the 100 landing pages that block the most Tor exit nodes. In 2016, the site with the highest block rate blocked ˜70% of exit nodes, and 1.6% of sites blocked more than 60% of exit nodes [9]. In contrast, we find that the highest block rate is 99%, indicating that sites have become better at blocking all visits from Tor. In addition, 7.2% of sites in our sample block more than 60% of exit nodes.

Subsites. Compared to landing pages, we observe a slightly lower block rate for visits to subsites (15.2%). This is contrary to our expectation that subsites would be blocked at higher rates. However, we note that our data collection method is slightly different to how a real user would browse because we visited subsites directly, from a prior list of subsites, instead of clicking on a link on the landing page. This means that our crawler may be able to visit subsites that ordinarily would be hidden behind a blocked landing page. As a result, the block rates for subsites experienced by real users would be at least as high as the block rate for landing pages. This may indicate that some sites have only implemented blocking for visitors to their landing page, but have not rolled out the blocking logic on direct visits to subsites.

Searches. We observed a very low block rate for search result pages (3.8%). It is important to note that our crawler could only find search boxes and submit queries on pages that were not blocked. As a result, the real block rate for searches is the block rate for landing pages plus the observed 3.8% block rate for searches. This increment of 3.8% in blocking for search pages is almost exactly what [22] found in 2017.

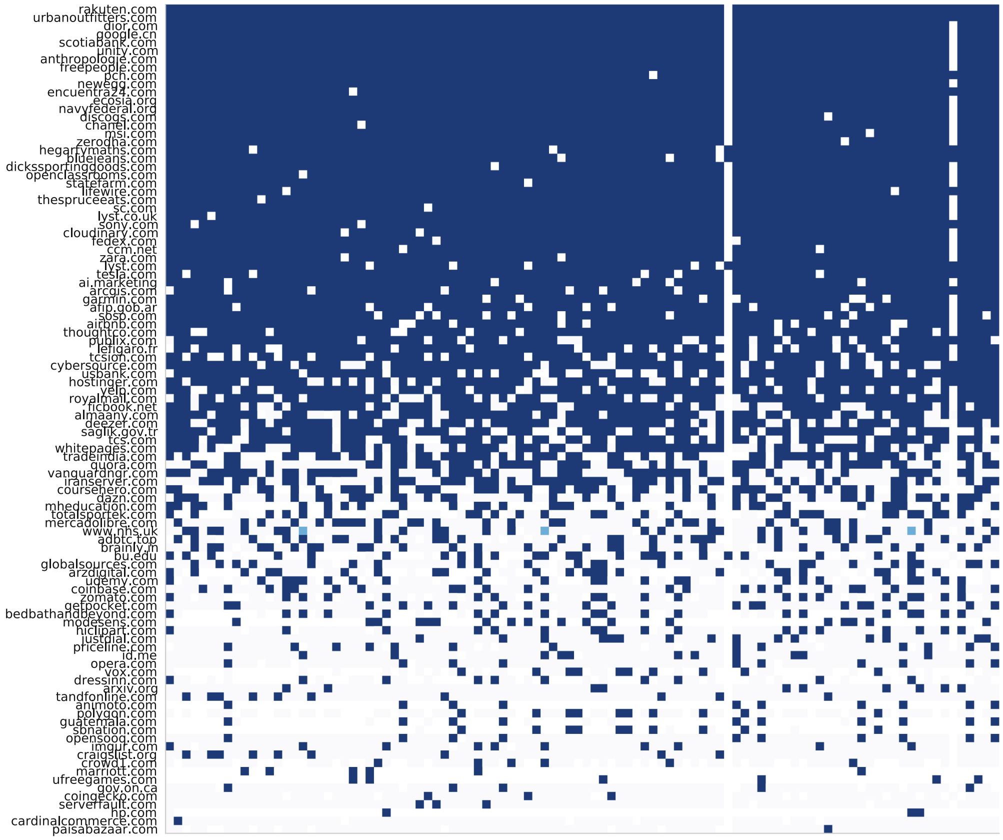


Fig. 3. 100 *landing pages* that block the most Tor exit nodes. Each column represents one exit node. Blue squares indicate when a landing page has blocked an exit node. The white vertical lines represent the two exit nodes that disappeared during data collection. Heat maps for subsites and searches are in Appendix C.

4.3 Block Rates by Characteristics of Tor Exit Nodes

We analyze whether block rates differ depending on characteristics of Tor exit nodes, in particular the country, exit probability, age, bandwidth, and number of open ports. We compute correlation coefficients using the Pearson correlation coefficient r and find that all correlations with exit probability ($\mathrm{r} = 0.16$, $\mathrm{p} = 0.10$), age ($\mathrm{r} = -0.016$, $\mathrm{p} = 0.88$), bandwidth ($\mathrm{r} = 0.10$, $\mathrm{p} = 0.31$), and number of open ports ($\mathrm{r} = 0.06$, $\mathrm{p} = 0.55$) are statistically not significant. This finding is consistent with results reported in 2017 [22].

Analyzing these correlations separately depending on a website's hosting provider, focusing on the four large hosters Akamai, Amazon, Cloudfront, and Cloudflare, we do not find any significant correlations. This is in contrast to prior work which found that block rates for Amazon-hosted sites are higher when the exit node has higher bandwidth [22].

We did find statistically significant differences in block rates depending on the exit node's country. However, all differences have small effect sizes according to Cohen's d. For example, the difference between the two countries with highest

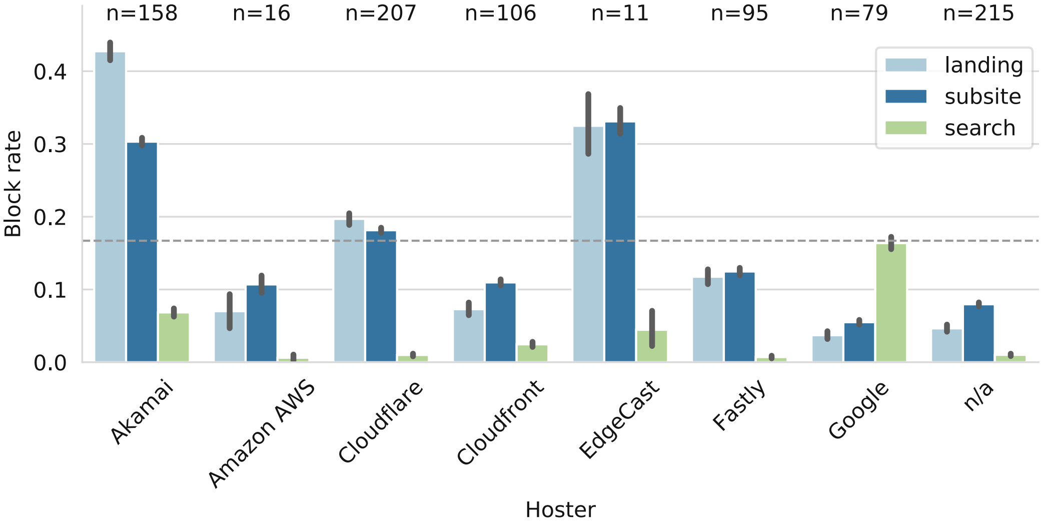


Fig. 4. Block rate by hoster (only hosters with more than 10 sites in our sample are shown). The dashed line indicates the average block rate for landing pages (16.7%).

and lowest block rates (New Zealand with 14.5% average and Azerbaijan with 11.4% average) is significant, but has a very small effect size ($p < 0.001$, $d = 0.093$).

4.4 Block Rates by Characteristics of Web Sites

Finally, we analyze whether block rates differ depending on characteristics of the visited websites, specifically the hosting provider, website category, and rank. To identify the hosting provider for each visited site, we use the findCDN library [6]. We use McAfee's URL categorization service [11] to identify website categories, and the Tranco toplist [18] from 2021-05-11 for website ranks. For all our results on statistical significance, we apply the Bonferroni correction to offset the problem of multiple comparisons.

Block Rates by Hosting Provider. Figure 4 shows the block rates we observe for the seven most common hosters in our data set, split by block rates for landing pages, subsites, and searches. We find that Akamai (28%, $d = 0.46$), EdgeCast (31%, $d = 0.45$), and Cloudflare (15%, $d = 0.089$) have block rates that are significantly higher than average (all $p < 0.001$). Especially Akamai and Cloudflare, because they are used by many sites, drive up the overall block rate. Block rates for the other hosters are significantly lower than the average. The lowest block rate is observed for websites with an unidentified hoster, which only block 5.8% of visits (n/a in the figure, $d = 0.3$).

In addition, we find that Google-hosted sites block significantly more searches than any other hosting provider (16% compared to the average of 3.8%, $d = 0.51$).

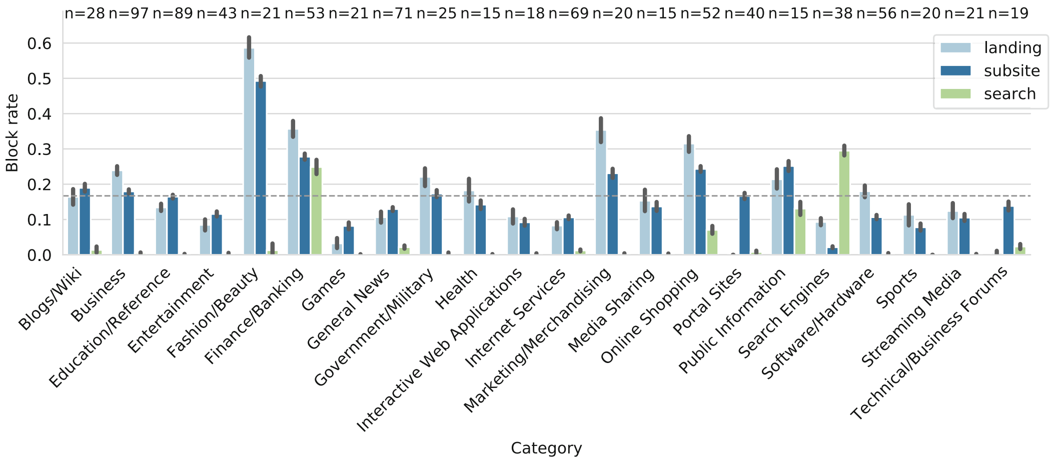


Fig. 5. Block rate by website category (only categories with at least 15 sites in our sample are shown). The dashed line indicates the average block rate for landing pages (16.7%).

Block Rates by Category. Figure 5 shows the block rate by website category. Overall, five categories of websites block landing pages and subsites with a significantly higher than average rate and at least a small effect size (all $p < 0.001$, $d > 0.2$): *Fashion/Beauty* (51%, $d = 0.85$), *Finance/Banking* (35%, $d = 0.35$), *Marketing/Merchandising* (25%, $d = 0.25$), *Online Shopping* (25%, $d = 0.27$), and *Public Information* (24%, $d = 0.23$). Conversely, three categories have significantly lower block rates: *Games* (7.4%, $d = 0.25$), *Search Engines* (3.6%, $d = 0.43$), and *Sports* (8.2%, $d = 0.22$).

Analyzing whether block rates on landing pages and subsites differ by category, we find only few cases with statistically significant differences and at least small effect sizes: websites in *Marketing/Merchandising*, *Search Engines*, and *Software/Hardware* block landing pages at higher rates, where as *Games*, *Portal* sites and *Technical/Business Forums* block subsites at higher rates. These findings indicate that our expectation that subsites are blocked at a much higher rate than landing pages is likely incorrect.

Finally, we observe that searches are blocked with a high rate by websites in the *Search Engines* category. This block rate is caused almost entirely by Google's "unusual traffic" CAPTCHA: Google blocks 39.7% of searches by Tor users, whereas other search engines only block 1%.

Block Rates by Rank. Analyzing block rates by website rank, we find that the correlation coefficients for landing pages, subsites, and searches are all negative but very small (r between -0.02 and -0.05), indicating a slightly lower block rate for lower-ranked websites. However, none of the correlations are statistically significant.

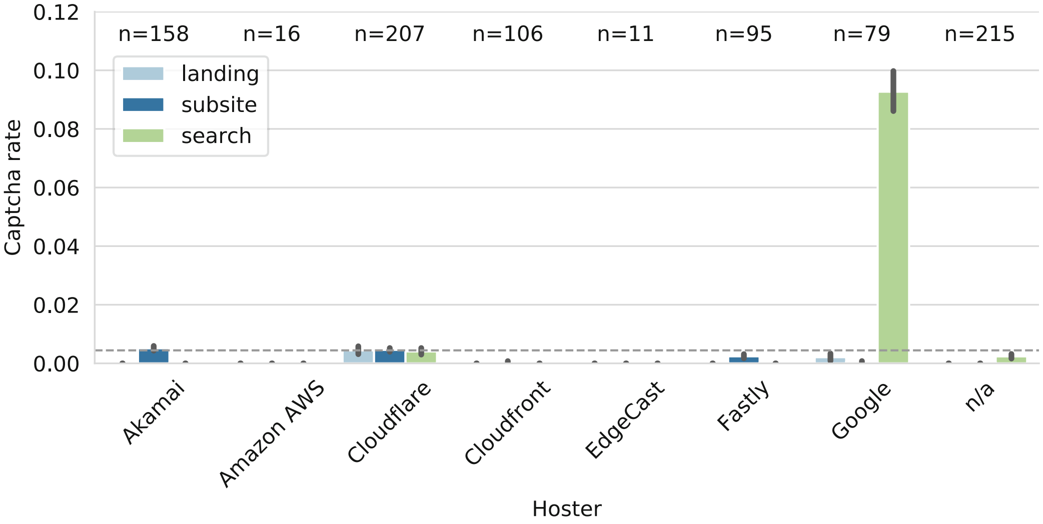


Fig. 6. Captcha rate by hoster. The dashed line indicates the average captcha rate (0.4%).

4.5 CAPTCHA Rates

To analyze the results from our multi-class classifiers, we analyze CAPTCHA rates in addition to block rates, i.e., the fraction of website visits that asked the user to solve a CAPTCHA before permitting access. We only evaluate the presence of a CAPTCHA, but not its difficulty[6].

We observe that CAPTCHA rates on average are much lower than block rates: 0.15% for landing pages, 0.23% for subsites, and 1.2% for searches. Similar to the results for block rates, we did not find significant correlations with exit node characteristics including age, bandwidth, open ports, and exit probability. CAPTCHA rates differ significantly between the countries with highest and lowest rates (Italy with 0.08% average and the Netherlands with 1% average), but the effect size is very small ($d = 0.13$).

Figure 6 shows the CAPTCHA rates by hosting provider. We can see that the overall largest contributor to CAPTCHA rates is searches performed on Google-hosted sites. The CAPTCHA rates for all hosters except Cloudflare and Akamai differ significantly from the average rate: Google has a significantly higher rate ($d = 0.18$), whereas all others have lower rates.

Analyzing CAPTCHA rates by website category (Fig. 7), we find that four categories have significantly higher than average CAPTCHA rates: *Blogs/Wiki* (1.3%, $d = 0.1$), *Finance/Banking* (1%, $d = 0.08$), *Games* (1.4%, $d = 0.1$) and *Search Engines* (3.6%, $d = 0.25$). We find no significant correlations between CAPTCHA rate and website rank.

[6] Anecdotally, CAPTCHAs given to anonymous users are more difficult than for other users, however, we leave the measurement of this effect and its analysis for future work.

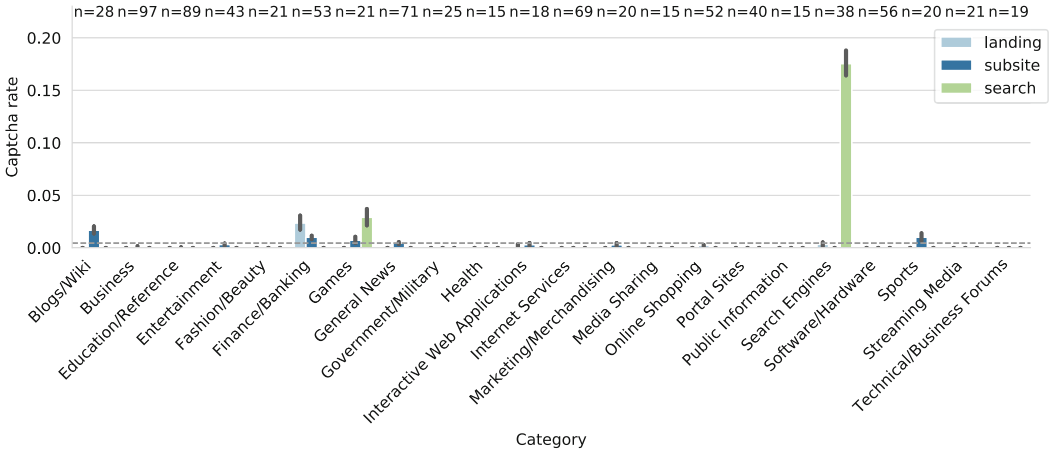


Fig. 7. Captcha rate by website category. The dashed line indicates the average CAPTCHA rate (0.4%).

In the analyses by hoster and by category, Google stands out because it shows CAPTCHAs to anonymous users who perform a search, but not on the landing page or other subsites. However, we note that Google is over-represented in our data set because it is present with several of its localized domains (76 Google domains among our 998 topsites, or 7.6%). This slightly skews the results for block and CAPTCHA rates for searches and for Google-hosted sites. However, the finding that Google serves block pages or CAPTCHAs for searches from anonymous users in almost 40% of cases remains. To an individual user, this means that 2 out of 5 Google searches may fail, which may be frustrating enough to discourage Tor users from using Google.

5 Limitations

The main limitations of our study relate to the collection and labeling of training data. First, our training data is unbalanced. In particular, the number of CAPTCHA samples is very small compared to samples of block pages, which negatively impacts multi-class classification. We have addressed this limitation mainly on the algorithm-level. Other promising ways to address this limitation would be to include more samples from known CAPTCHA-serving websites in the training data, to use data-level approaches such as over-/under-sampling or SMOTE, and to use other metrics including the receiver operating characteristic curve (ROC).

Second, our set of labels does not account for cases when both Firefox and Tor Browser show a CAPTCHA to the user (they have been labeled as *same*). This makes it more difficult for classifiers to correctly label CAPTCHA cases. It may be better to exclude these cases from the training data. Third, CAPTCHAs are often slow to load. For example, they are often hidden behind a timed "checking your browser" page. In cases where these pages did not proceed to show the CAPTCHA before our crawler's page load timeout, they were labeled as block pages instead of CAPTCHAs. This may explain the low CAPTCHA rates observed in our results. This issue may be fixed by increasing the page load timeout, however, we already used a generous timeout of 120 s.

To illustrate the extent of these limitations, we manually label 0.1% of our collected data (661 samples), focusing on cases where classifiers disagree about the label of a visit as block page or CAPTCHA. Keeping in mind that this is a very small sample that *only contains difficult cases*, we obtain classification accuracies of 54%–90% for the binary classifiers, and 63%–79% for multi-class classifiers. This is an encouraging, although not perfect, result, which indicates that the bulk of our analyses are likely correct. This is supported by the fact that the statistically significant results presented above remained significant regardless of which classifier was applied.

We use a limited set of features for JavaScript inclusions and screenshots, namely, the domain names, paths, and files names of the JS inclusions and the perceptual hashes of screenshots. It is possible that including features based on JS code and full image classification could improve the performance of our classifiers. In addition, we only evaluate the effect of feature groups on classifier performance, but not the effect of individual features.

Our study does not investigate *why* websites discriminate against Tor users. This would be an interesting extension that ultimately may help websites to better distinguish between malicious and benign traffic, and thereby reduce discrimination.

6 Conclusion

In this paper, we have presented machine learning classifiers to detect when websites block anonymous visitors from Tor Browser, and we have evaluated the prevalence of Tor exit blocking based on a large-scale data collection. We find that on average, one in six websites (16.7%) are inaccessible when browsing with Tor. In addition, Google's high block/CAPTCHA rate for anonymous users who perform searches is notable. Compared to studies published in 2016/17 [9,22], we find that the prevalence of blocking has not changed much, but that websites have become better at blocking all visits coming from Tor if they so wish (indicated by several websites with near 100% block rates in our sample).

Future work may improve on our results, especially for CAPTCHA rates, by expanding the training data collection with more CAPTCHA samples and fine-tuning the labeling approach. Promising avenues for future work include the evaluation of CAPTCHA difficulty (e.g., based on the CAPTCHA type, the number of sets that users have to complete, or the server-side time delays in the CAPTCHA solving process), and the measurement of the time cost imposed on Tor users for solving CAPTCHAs.

Acknowledgements. We thank the anonymous reviewers and our shepherd Kui Ren for their valuable feedback and Matthew Ford for contributions to an early prototype of our crawler and data analysis.

A Classifier Performance

(See Tables 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15).

Table 4. Performance of binary classifier for feature set F1

	Precision	Recall	F1-score	Support
block+captcha	1.000	1.000	1.000	70.000
same	1.000	1.000	1.000	646.000
accuracy	1.000	1.000	1.000	1.000
macro avg	1.000	1.000	1.000	716.000

Table 5. Performance of binary classifier for feature set F2

	Precision	Recall	F1-score	Support
block+captcha	0.972	0.986	0.979	70.000
same	0.998	0.997	0.998	646.000
accuracy	0.996	0.996	0.996	0.996
macro avg	0.985	0.991	0.988	716.000

Table 6. Performance of binary classifier for feature set F3

	Precision	Recall	F1-score	Support
block+captcha	1.000	1.000	1.000	70.000
same	1.000	1.000	1.000	646.000
accuracy	1.000	1.000	1.000	1.000
macro avg	1.000	1.000	1.000	716.000

Table 7. Performance of binary classifier for feature set F4

	Precision	Recall	F1-score	Support
block+captcha	0.971	0.971	0.971	70.000
same	0.997	0.997	0.997	646.000
accuracy	0.994	0.994	0.994	0.994
macro avg	0.984	0.984	0.984	716.000

Table 8. Performance of binary classifier for feature set F5

	Precision	Recall	F1-score	Support
block+captcha	0.985	0.957	0.971	70.000
same	0.995	0.998	0.997	646.000
accuracy	0.994	0.994	0.994	0.994
macro avg	0.990	0.978	0.984	716.000

Table 9. Performance of binary classifier for feature set F6

	Precision	Recall	F1-score	Support
block+captcha	0.909	0.857	0.882	70.000
same	0.985	0.991	0.988	646.000
accuracy	0.978	0.978	0.978	0.978
macro avg	0.947	0.924	0.935	716.000

Table 10. Performance of multi-label classifier for feature set F1

	Precision	Recall	F1-score	Support
BLOCK	1.000	1.000	1.000	59.000
CAPTCHA	1.000	0.818	0.900	11.000
SAME	0.997	1.000	0.998	646.000
accuracy	0.997	0.997	0.997	0.997
macro avg	0.999	0.939	0.966	716.000

Table 11. Performance of multi-label classifier for feature set F2

	Precision	Recall	F1-score	Support
BLOCK	0.933	0.949	0.941	59.000
CAPTCHA	1.000	0.818	0.900	11.000
SAME	0.992	0.994	0.993	646.000
accuracy	0.987	0.987	0.987	0.987
macro avg	0.975	0.920	0.945	716.000

Table 12. Performance of multi-label classifier for feature set F3

	Precision	Recall	F1-score	Support
BLOCK	0.983	1.000	0.992	59.000
CAPTCHA	1.000	0.818	0.900	11.000
SAME	0.997	0.998	0.998	646.000
accuracy	0.996	0.996	0.996	0.996
macro avg	0.993	0.939	0.963	716.000

Table 13. Performance of multi-label classifier for feature set F4

	Precision	Recall	F1-score	Support
BLOCK	0.951	0.983	0.967	59.000
CAPTCHA	1.000	0.727	0.842	11.000
SAME	0.995	0.997	0.996	646.000
accuracy	0.992	0.992	0.992	0.992
macro avg	0.982	0.902	0.935	716.000

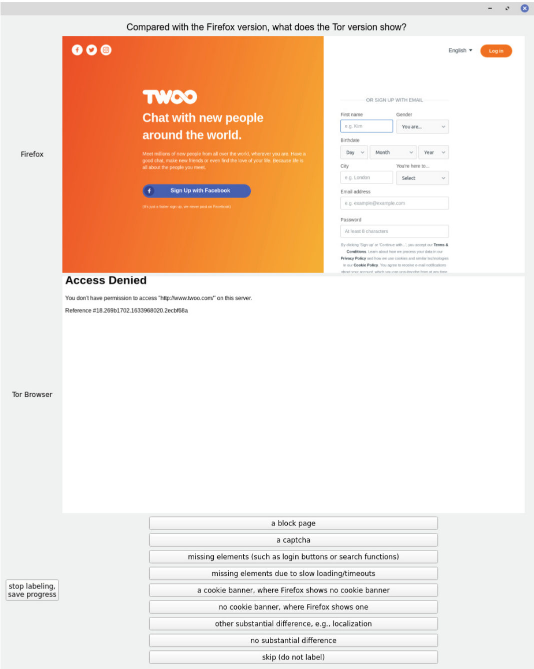
Table 14. Performance of multi-label classifier for feature set F5

	Precision	Recall	F1-score	Support
BLOCK	0.851	0.966	0.905	59.000
CAPTCHA	1.000	0.727	0.842	11.000
SAME	0.994	0.986	0.990	646.000
accuracy	0.980	0.980	0.980	0.980
macro avg	0.948	0.893	0.912	716.000

Table 15. Performance of multi-label classifier for feature set F6

	Precision	Recall	F1-score	Support
BLOCK	0.859	0.932	0.894	59.000
CAPTCHA	1.000	0.455	0.625	11.000
SAME	0.988	0.989	0.988	646.000
accuracy	0.976	0.976	0.976	0.976
macro avg	0.949	0.792	0.836	716.000

B Labeling



Compared with the Firefox version, what does the Tor version show?

Meet millions of new people from all over the world, wherever you are. Have a good chat, make new friends or even find the love of your life. Because life is all about the people you meet.

By clicking 'Sign up' or 'Continue with...', you accept our **Terms & Conditions**. Learn about how we process your data in our **Privacy Policy** and how we use cookies and similar technologies in our **Cookie Policy**. You agree to receive e-mail notifications about your account, which you can unsubscribe from at any time.

Access Denied

You don't have permission to access "http://www.twoo.com/" on this server.

Reference #18.269b1702.1633968020.2ecbf68a

stop labeling, save progress
a block page
a captcha
missing elements (such as login buttons or search functions)
missing elements due to slow loading/timeouts
a cookie banner, where Firefox shows no cookie banner
no cookie banner, where Firefox shows one
other substantial difference, e.g., localization
no substantial difference
skip (do not label)

Fig. 8. Screenshot of GUI for manual labeling of training data

C Block Rates for Subsites and Searches

(See Figs. 9 and 10).

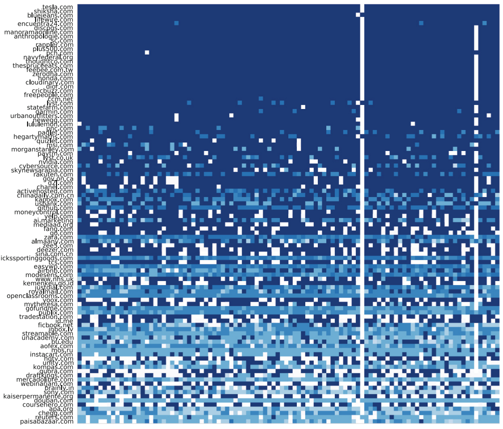


Fig. 9. Subsites that block the most Tor exit nodes. Each column represents one exit node.

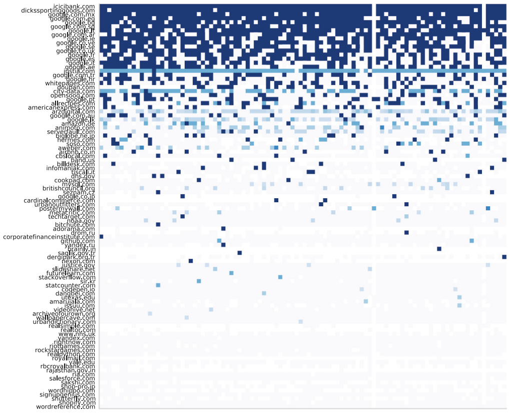


Fig. 10. Sites that block the most Tor exit nodes when performing a search.

References

1. Acar, G., Juarez, M.: Individual contributors: Tor-browser-selenium - Tor Browser automation with Selenium (2020). https://github.com/webfp/tor-browser-selenium
2. Aqeel, W., Chandrasekaran, B., Feldmann, A., Maggs, B.M.: On landing and internal web pages: the strange case of Jekyll and Hyde in web performance measurement. In: Proceedings of the ACM Internet Measurement Conference, IMC 2020, pp. 680–695. Association for Computing Machinery, Pittsburgh, PA, USA, October 2020
3. Broder, A.Z.: Identifying and Filtering Near-Duplicate Documents. In: Giancarlo, R., Sankoff, D. (eds.) Combinatorial Pattern Matching. pp. 1–10. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2000)
4. Buchner, J.: ImageHash, January 2022. https://github.com/JohannesBuchner/imagehash
5. Cuzzocrea, A., Martinelli, F., Mercaldo, F., Vercelli, G.: Tor traffic analysis and detection via machine learning techniques. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 4474–4480, December 2017
6. Cybersecurity and Infrastructure Security Agency: FindCDN. Cybersecurity and Infrastructure Security Agency, January 2022. https://github.com/cisagov/findcdn
7. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: 13th USENIX Security Symposium (USENIX Security 2004). USENIX Association, San Diego, CA, USA, August 2004
8. Gurunarayanan, A., Agrawal, A., Bhatia, A., Vishwakarma, D.K.: Improving the performance of machine learning algorithms for Tor detection. In: 2021 International Conference on Information Networking (ICOIN), pp. 439–444, January 2021

9. Khattak, S., et al.: Do You See What I See? Differential treatment of anonymous users. In: Proceedings 2016 Network and Distributed System Security Symposium. Internet Society, San Diego, CA (2016)
10. Mani, A., Wilson-Brown, T., Jansen, R., Johnson, A., Sherr, M.: Understanding Tor usage with privacy-preserving measurement. In: Proceedings of the Internet Measurement Conference 2018, IMC 2018, pp. 175–187. Association for Computing Machinery, Boston, MA, USA, October 2018
11. McAfee, LLC: Customer URL Ticketing System for McAfee Web Gateway (2022). https://www.trustedsource.org/
12. McDonald, A., et al.: 403 forbidden: a global view of CDN geoblocking. In: Proceedings of the Internet Measurement Conference 2018, IMC 2018, pp. 218–230. ACM, Boston, MA, USA (2018)
13. Niaki, A.A., et al.: ICLab: a global, longitudinal internet censorship measurement platform. In: 2020 IEEE Symposium on Security and Privacy (S&P), pp. 214–230. IEEE, San Francisco, CA, USA, May 2020
14. Odvarko, J.: HAR Export Trigger, May 2018. https://addons.mozilla.org/en-US/firefox/addon/har-export-trigger/
15. Oh, S.E., Li, S., Hopper, N.: Fingerprinting keywords in search queries over Tor. Proc. Privacy Enhancing Technol. **2017**(4), 251–270 (2017)
16. OONI: Open Observatory of Network Interference (2022). https://ooni.org/
17. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization: anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. Technical report, August 2010. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf
18. Pochat, V.L., van Goethem, T., Joosen, W.: Rigging research results by manipulating top websites rankings. In: 26th Annual Network and Distributed System Security Symposium. Internet Society, San Diego, CA, USA, February 2019
19. Ren, J., Rao, A., Lindorfer, M., Legout, A., Choffnes, D.: ReCon: revealing and controlling PII leaks in mobile network traffic. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2016, pp. 361–374. ACM, Singapore (2016)
20. Shuba, A., Markopoulou, A.: NoMoATS: towards automatic detection of mobile tracking. Proc. Privacy Enhancing Technol. **2020**(2), 45–66 (2020)
21. Silva, M., Santos de Oliveira, L., Andreou, A., Vaz de Melo, P.O., Goga, O., Benevenuto, F.: Facebook ads monitor: an independent auditing system for political ads on Facebook. In: Proceedings of The Web Conference 2020, WWW 2020, pp. 224–234. ACM, Taipei, Taiwan, April 2020
22. Singh, R., et al.: Characterizing the nature and dynamics of Tor exit blocking. In: 26th USENIX Security Symposium (USENIX Security 2017), pp. 325–341. USENIX Association, Vancouver, BC, August 2017
23. Soleimani, M.H.M., Mansoorizadeh, M., Nassiri, M.: Real-time identification of three Tor pluggable transports using machine learning techniques. J. Supercomputing **74**(10), 4910–4927 (2018). https://doi.org/10.1007/s11227-018-2268-y
24. Spärck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation **28**(1), 11–21 (1972)
25. Tran, C., Champion, K., Forte, A., Hill, B.M., Greenstadt, R.: Are anonymity-seekers just like everybody else? An analysis of contributions to Wikipedia from Tor. In: 2020 IEEE Symposium on Security and Privacy (S&P), pp. 974–990. IEEE, San Francisco, CA, USA, May 2020

26. Urban, T., Degeling, M., Holz, T., Pohlmann, N.: Beyond the front page: measuring third party dynamics in the field. In: Proceedings of The Web Conference 2020, WWW 2020, pp. 1275–1286. ACM, Taipei, Taiwan, April 2020
27. Wagner, I.: Auditing Corporate Surveillance Systems: Research Methods for Greater Transparency. Cambridge University Press, Cambridge (2022)
28. Zhu, E.: Datasketch: Big Data Looks Small, January 2022. https://github.com/ekzhu/datasketch