Daniel Kats*, David Luz Silva*, and Johann Roturier*

Who Knows I Like Jelly Beans? An Investigation Into Search Privacy

Abstract: Internal site search is an integral part of how users navigate modern sites, from restaurant reservations to house hunting to searching for medical solutions. Search terms on these sites may contain sensitive information such as location, medical information, or sexual preferences; when further coupled with a user's IP address or a browser's user agent string, this information can become very specific, and in some cases possibly identifying.

In this paper, we measure the various ways by which search terms are sent to third parties when a user submits a search query. We developed a methodology for identifying and interacting with search components, which we implemented on top of an instrumented headless browser. We used this crawler to visit the Tranco top one million websites and analyzed search term leakage across three vectors: URL query parameters, payloads, and the *Referer* HTTP header. Our crawler found that 512,701 of the top 1 million sites had internal site search. We found that 81.3% of websites containing internal site search sent (or *leaked* from a user's perspective) our search terms to third parties in some form. We then compared our results to the expected results based on a natural language analysis of the privacy policies of those leaking websites (where available) and found that about 87% of those privacy policies do not mention search terms explicitly. However, about 75% of these privacy policies seem to mention the sharing of some information with third-parties in a generic manner. We then present a few countermeasures, including a browser extension to warn users about imminent search term leakage to third parties. We conclude this paper by making recommendations on clarifying the privacy implications of internal site search to end users.

Keywords: privacy policy, web tracking, measurement

DOI 10.2478/popets-2022-0053
Received 2021-08-31; revised 2021-12-15; accepted 2021-12-16.

Daniel Kats*: NortonLifeLock Research Group, E-mail: daniel.kats@nortonlifelock.com
David Luz Silva*: NortonLifeLock Research Group, E-mail: david.silva@nortonlifelock.com
Johann Roturier*: NortonLifeLock Research Group, E-mail: johann.roturier@nortonlifelock.com

1 Introduction

Internal site search is integral to how users discover and interact with a wide range of web content including shopping, travel planning, medical information, and social network features. Internal search is increasingly being used by businesses to drive revenue, since 30% of e-commerce website visitors use internal site search which contributes to 14% of all revenue [71].

Users may use these search boxes to type in highly personal terms expressing racial identity, sexual or religious preferences, and medical conditions. Prior work has shown how easy it is to de-anonymize users based on their search terms [30, 75]. Alarmingly, Guha et al. found that advertisers collect and use sensitive data such as sexual preference for ad personalization: for example, nursing college degree ads appeared to target gay users exclusively [27]. In 2006, after AOL published a large amount of anonymized search queries, several participants were completely de-anonymized based on the uniqueness and specificity of their search terms [5]. Therefore, the secrecy of search terms is paramount to any reasonable definition of privacy on the web.

A parallel trend shows the increasing proliferation of third-party scripts and fingerprinting techniques (including persistent cookies, tracking pixels, and other technologies - collectively called *trackers*) on the web, mostly for advertising purposes [80]. A 2014 crawl of the top 1 million websites[1] showed ubiquitous user tracking: 62.9% of sites spawned third-party cookies, 83.0% loaded third-party JavaScript libraries (potentially for browser fingerprinting), and 88% of all sites made a third-party network request in some form [38]. A follow-up study in 2019 found the percentage of sites which track users increased to 90%, despite the introduction of privacy-protecting legislation in the European Union at that time [67].

In this paper, we explore the question of search privacy in a novel way, by investigating search term leak-

* All three authors contributed equally to this work as first authors.

1 using the Alexa top 1 million list [2]

ages to third-parties via internal site search (whereas previous work had focused on user interests being leaked via links provided by generic search engines [37]). Our concern is indeed not with man-in-the-middle actors but with third-parties with resources on a first-party web page. Another novel contribution of this work is the analysis of the wording of such a practice in privacy policies, which extends previous work that had simply focused on determining whether third-parties were mentioned in privacy policies [39]. Specifically, we visited the Tranco top 1 million websites [36] and performed an automated internal site search. We developed a crawler on top of a headless browser which can handle dynamically generated websites, identify search inputs, and capture outgoing requests. We then analyzed the content of outgoing requests to determine whether our search terms appeared in the data or metadata. In particular, we examine search term leakage along three vectors:

1. **Leakage via *Referer*[2]**. Presence of search terms in the HTTP *Referer* request header sent to third parties.
2. **Leakage via URL**. Presence of search terms in third-party URLs and query parameters.
3. **Leakage via payload**. Presence of search terms in HTTP request payloads sent to third parties.

We discover that 81.3% of visited websites send search terms to third parties in some form, representing a potential privacy threat to users, without being the equivalent to leakage from web-wide search. Additionally, despite many websites having privacy policies, we found the topic of search term processing was not explicitly covered by the privacy policies (about 87% of the time when we were able to recover a privacy policy).

The contributions of this paper are as follows:

1. We present the first large-scale measurement study on the leakage of search terms via internal web search, visiting one million websites, intercepting over 68 million network requests, and discovering 512,701 unique domains that seem to include an internal search functionality.
2. We build a crawling methodology (validated using human annotators) to identify search inputs across a range of natural languages during a large-scale crawl, collecting search input and search-related element selectors for each domain. We then open-source the crawler and this labeled dataset of discovered search inputs per domain to the community for further study[3].
3. We perform an analysis of natural-language privacy policies to determine if these privacy policies mention processing search terms or sharing some information with third parties.
4. We provide a framework to characterize the various vectors for leaking search terms to third parties. Using this framework, we found that 81.3% of sites send search terms in some form to third parties. By breaking this down according to our vectors, we find that 75.8% of websites send data via the *Referer* header (72.5% directly and 10.6% indirectly); 71% via URL query parameters; and 21.2% via the payload. Additionally, 87.4% of websites had the potential to send sensitive query terms to a new third-party if a link on the page was clicked by the user. Finally, this framework allowed us to identify those third parties that are actively involved in providing search results.
5. We develop a browser extension to inform users about potential privacy leaks via internal search, based on our findings, which we release [4].

2 Background

This section provides additional information on the leakage vectors defined in the previous section, focusing on the leakages via the *Referer* since it is the most subtle. It also provides an explanation of what is considered as a third-party in this study.

2.1 *Referer* HTTP Header

The *Referer* HTTP request header is automatically set by the client (browser) to indicate to a website how a user found that website. The *Referer* header has been a feature in HTTP since at least HTTP 1.0 [56]. This first RFC stipulates that the *Referer* header may be useful in debugging broken links, cache optimization, and logging. It also contains a warning about the privacy implications, and suggests that users should have the option to control this value. This recommendation is also made in the RFC for HTTP 1.1 [57]. To our knowl-

2 *Referer* is the original spelling

3 https://jellybeans-paper.com/

4 under review, which should be complete by the time of the symposium edge, no major browser allows users to easily configure this value as suggested.

2.2 *Referrer-Policy* HTTP Header

The *Referrer-Policy* response header is set by the website owner, and specifies how the *Referer* header will be set upon requests originating from that site. The *Referrer-Policy* header has several acceptable values, which are shown in Table 1. The referrer policy can be set either in the HTTP header, in a `<meta>` tag in the header section of the document, or as a `noreferrer` attribute for a particular link on the site. It may also be set implicitly via inheritance.

As can be seen from this table, a significant amount of attention has been paid to not sending *Referer* values over insecure (non-HTTPS) channels. It was observed by the early writers of the specification that the *Referer* values may contain sensitive information that man-in-the-middle attackers may want to steal. Our concern in this study is not with man-in-the-middle actors but with third-parties with resources on a first-party web page.

2.3 Leaking Search Terms via the *Referer*

The *Referer* header can therefore accidentally (or intentionally) leak search terms to third parties. This attack has been previously documented by Libert [37], Krishnamurthy [34], Malandrino [43], and several others. A real-world attack scenario is as follows: A user performs a search for "pancreatic cancer" on the WebMD website.[5] The WebMD server returns a webpage with the results, which then loads a variety of third-party JavaScript tracking scripts, including one from Google Ad Services. When the request is made to fetch that script from the Google server, the default behaviour (and the one observed in the wild) is sending the URL together with the query string, which contains the sensitive search term. The raw request header values can be seen below[6]:

```
GET /pagead/conversion_async.js HTTP/2
Host: www.googleadservices.com
User-Agent: Mozilla/5.0 XXX Firefox/80.0
Referer: https://www.webmd.com/search/search_results/
default.aspx?query=pancreatic%20cancer
```

A similar attack scenario involves a user browsing the WebMD website and after following a few links, landing on a page whose URL contains revealing words (e.g., such as "what-to-do-when-getting-pancreatic-cancer"). The main difference between these two scenarios, however, is that the user may be unaware of the words present in the URLs when clicking on links, so some of the pages visited may be irrelevant. However, when a user enters a search term in a box, there is an explicit intent to obtain results about a specific topic.

In this paper, we use the term *search term leakage* to refer to the transmission of search terms to third parties. We note this transmission might be either intentional or accidental with respect to the website owner. It may be accidental because the website owner may rely on default settings that are not privacy-oriented or because they place too much trust on third-parties, which may be collecting and/or sharing more information than they should. It may be intentional if the website owner has put in place some contractual agreements with third parties to handle such information (e.g., to provide the actual search service, to record requests for security or analytics purposes, or to serve personalized ads). When search term leakages are intentional, users are extremely unlikely to be aware of them, with the exception of potential visual cues being placed in the search box (e.g., "powered by ServiceX"). However, a previous study found that most sites tend to prefer seamless integrations with third-party services (for instance, when embedding forms) [72].

2.4 What is a Third Party?

What constitutes a first party and a third party can be difficult to define. In this study, we rely on effective Top-Level Domains as defined in the Public Suffix List [53], specifically the *eTLD+1*, so we consider domains that do not share the same eTLD+1 third parties.

Previous studies have tried to refine this definition by considering a third-party as "a subdomain of the main domain whose IP address is registered to a different autonomous system (AS) than the IP address of the first-party website." [72] While this approach would catch those tracking entities that do not want to be detected, we accept that in some cases our third-party results might be slightly lower that what they might be in reality. However, we think that this might be balanced

5 WebMD's privacy policy includes language suggesting that search terms are collected by WebMD and that some information may be collected by, or transmitted to, third parties

6 some values were redacted to protect the author's privacy

Value	Description	Prevents leakage via referrer to third party
`no-referrer`	*Referer* will be empty	Yes
`no-referrer-when-downgrade` (default for most browsers)	sends full *Referer* header value if both this and the target page are HTTPS otherwise sends nothing	No when both pages are HTTPS
`same-origin`	same origin requests contain full *Referer* information while no *Referer* is sent for cross-origin requests	Yes
`origin`	only the origin, not path or query information, are sent	Yes
`strict-origin`	same as above, but only for HTTPS pages. Non-HTTPS pages result in no *Referer* being sent	Yes
`origin-when-cross-origin`	cross-origin requests behave as if $origin$ was the specified policy while same-origin requests behave as if $same - origin$ was specified	Yes
`strict-origin-when-cross-origin` (default for Safari and Chrome v85+)	same as above, but only for HTTPS pages. Non-HTTPS pages result in no *Referer* being sent	Yes
`unsafe-url`	Full *Referer* is sent in all contexts. This is generally advised against	No

Table 1. *Referrer-Policy* header valid values, derived from the W3C specification [78]. `no-referrer-when-downgrade` is the default policy for Firefox, Chrome versions 84 and below, and Edge. `strict-origin-when-cross-origin` is the default policy for Safari and Chrome versions 85 and higher [55].

by the fact that in other cases, two domains that do not share the same eTLD+1 might actually be owned by the same entity. As mentioned by editors from the W3C Privacy Community Group's *First-Party Sets* work group, "the website the user is interacting with may be deployed across multiple domain names" [64]. So ideally, when identifying third-party requests, we would like to exclude those domains that may be owned by the entity that also owns the site that is being analyzed (e.g. *datahug.com* and *sap.com* are part of the same entity after SAP's acquisition of DataHug but nothing in their respective WHOIS records can link one to the other). While this violates the *Origin* concept, it is closer to a user's mental model of what a website is and its associated trust relationship. However, we may not be able to reliably identify such relationships, since domain sets which claim to provide this information are not completely reliable (e.g. WHOIS) or exhaustive (e.g. Disconnect's Tracker Protection lists [17]). So for the purpose of this study, we rely solely on eTLD+1 and define **search term leakage** as any transmission of the search string between different eTLD+1 entities.

2.5 Privacy Policies

To determine whether website operators explicitly inform users about sharing their search queries with third-parties, we examine the privacy policies of leaking websites (where possible). While some jurisdictions do not mandate the posting of privacy policies, "many individual [US] states do have this type of legislation. Additionally, the countries within the European Union (EU) have also enacted similar legislation that requires websites that collect personal or identifiable data to post privacy policies regarding its use." [63] A recent study focusing on a set of 6,579 popular websites found that 84.5% of these had privacy policies [16]. This number should be interpreted as an upper bound since this study used a hybrid data collection approach (automated, followed by manual validation), which does not scale too well for larger data sets. Besides, one would expect popular sites to include privacy policies whereas less known sites (such as personal pages) are less likely to do so.

3 Data Collection

In this section, we describe in detail the crawling environment used to simulate user searches over the Tranco top 1M domains[7] [36]. We also provide a high-level overview of our privacy policy collection methodology. In order to complete this task, we instrumented a headless Chromium-based browser to visit each domain and detect the presence of all search inputs contained on the landing webpage. For each detected search input, the crawler simulates a real user search by typing the

7 We used the list created on April 27, 2020. Available at https://tranco-list.eu/list/NYJW.

keyword "JELLYBEANS" into the search input box one character at a time. While doing so, the crawler simultaneously intercepts all inbound and outbound network traffic, request redirection chains, payloads, and HTML content. We can then search for our dummy search string in the intercepted outbound network request data and metadata to detect search string sharing with third-party entities.

3.1 Web Crawler

Considering the trade-offs from the crawler categories minutely discussed in [1], we developed a User Layer based crawling system built on top of the Puppeteer headless browser [11], written in TypeScript, which we call `SlicedPie`. This architecture was chosen to maximize the overall accuracy, coverage, and completeness levels of the crawling process while allowing for some moderate extensibility. `SlicedPie`'s design facilitated simulating real users' interactions with the browser and enabled (i) driving Chromium as a full-fledged browser capable of handling dynamically generated content; (ii) launching on-demand JavaScript execution tasks; and still (iii) benefiting from lower level access to the Chrome Development Tools (CDP) [10] API (through CDP sessions [12] instantiation).

In terms of computational power, the underlying system was configured with 64 CPU cores running 128 threads and 800 GB RAM running Ubuntu 18.04LTS and Linux kernel version `4.15.0-72-generic`. This setup was used to handle more than 100 concurrent worker-thread [58] instances. Each worker-thread was initially assigned to process a single input domain URL, and during its execution process it could run multiple parallelized browser instances.

3.2 Crawling Stages

The crawling process encompasses four interdependent crawling stages. Each stage contains additional intermediate steps where data validation and information extraction tasks are performed in order to reduce error rates, improve performance, and maximize results coverage. Crawls are scheduled and subsequently parsed by a crawler manager (main thread).

Preflight Request (*Stage I*) In this stage we perform a *preflight* request using Node `fetch` [49] to confirm the URL's availability. Common system errors such as DNS failures are captured at this stage and therefore prevent further stages from executing. Additionally, we implement a mechanism to handle timeouts to circumvent the `fetch` API's inability to perform as expected [73].

Extracting HTML (*Stage II*) During this stage the HTML text is extracted and analysed in order to:

- **Detect additional natural languages.** The HTML language attribute [51] is extracted from the HTML text body in an attempt to identify additional languages present in the page structure. Each additional language found (apart from English) is used in *Stage III* to generate additional selectors.
- **Detect embedded search functionalities.** Websites may contain embedded search capabilities that are defined using the Schema.org vocabulary [79] or for a small number of sites the OpenSearch description format [52]. For instance, the Schema.org *SearchAction* type defines a common URL structure that can be detected using a regular expression and parsed into a search URL.[8]. This is the pattern that search engine services rely on to offer search boxes directly in their search results pages. [26] Thus, as an additional step of this crawling stage we try to identify and extract the search endpoint URL from the HTML string. Using this method, we discover that 8.5% of crawled sites rely solely on the search functionality defined using the Schema.org vocabulary.
- **Collect all hyperlinks found in the page**, including those pointing to privacy policies, whose identification is described in Section 3.5.

Detecting Search Functionality (*Stage III*) In order to detect search functionalities that users interact with on a site (either providing by the site itself or by a third party), we first rely on a list of query selectors [48] to identify all elements on a Web page that contain the keyword *search* as part of their attributes. While English is the default language used, the keyword varies according to any additional language detected during stage I. In order to use quality translations for the term *search*, we leveraged translations from Mozilla's Firefox Translation Project [50].

The list of elements is subsequently filtered and bucketed into two unique query selector lists of search inputs and search-related elements. Search inputs represent all elements into which it is possible to enter text. In the search-related inputs category fall all other HTML

8 For instance, http://example.com/?s={search-term-string}

elements (for example: div, li, span, etc.) that cannot be used to perform a search but indicate via their attributes that they are somehow related to search functionality.

Simulating User Search (*Stage IV*) All information gathered during previous stages is used here to define a strategy to simulate a search. There are two base approaches to do so:

1. **Search Inputs.** If the element is of type input, the approach is to visit the page, locate the element, simulate the user typing the keyword "JELLYBEANS" followed by the return key.
2. **Embedded Search URL.** If an embedded search URL has been identified and extracted during *Stage II*, it is used to conduct a search. In this case the keyword is added as an input parameter while the page is visited using a URL in the format of http://example.com?s=JELLYBEANS.

3.3 Crawl Summary

Table 2 outlines the results of our crawl. In summary, we tried to crawl 1 million sites using a crawling node located in the US-West region. However, 15.7% of those sites generated errors preventing us from determining whether the site might contain a search component. For 32.9% of the remaining sites, we were unable to detect any search component. This left us with 512,701 sites, where our search simulations seem to succeed 92.1% of the time. However, this number must be interpreted as an upper bound because for 1.57% of the 472,333 sites labelled as success, we were unable to detect any network traffic. Our methodology may also have identified some fields that were not actual search inputs, so we present some validation tests in Section 3.4.

The SlicedPie crawler can deal with a number of challenging situations that arise in the wild including dynamically-generated element attributes, interstitials, hidden search inputs, and search results displayed in another tab. The mechanics of our solutions for these are detailed in Appendix 1.

3.4 Crawl Validation

In order to validate our data collection approach, we considered two approaches: using an existing ground truth dataset or performing some manual annotation to build our own validation dataset. Since we are not aware of any dataset providing a list of sites containing search input boxes (on the site's main page), the only dataset that could be considered was a list of sites using either schema.org's action search [79] or OpenSearch [52]. As described in Section 3.2, our detection methodology considers schema.org's action search inputs, so to avoid any bias, we turned our focus on OpenSearch. We were able to detect 17,321 sites that included a reference to OpenSearch, and 84.2% of these sites were labelled as *success* in our crawl results. Examining manually a few sites using such technology, however, revealed two issues: first, a site may make use of OpenSearch without providing users a visual component to search the site from their main page. For instance, the HTML of nationalgeographic.org contains a link to https://www.nationalgeographic.org/opensearch.xml. In this particular example, a search mechanism is made available to users after clicking on an "Education" tab. As mentioned in Section 3.2, navigating away from the site's main page to locate search components is not something that our crawler currently supports. The second problem we discovered is that the search URL that may be provided using the OpenSearch's format may be stale. Based on this analysis, we concluded that it would not be possible to assume that the presence of *OpenSearch* on a site would guarantee the successful execution of a search using a visual component.

Therefore we decided to manually inspect some sites in order to build our own ground truth dataset by checking for the presence of search inputs. To do so, we randomly selected from our initial crawl dataset 200 sites that had been labelled as "success" by our crawler, 100 sites that had been labelled as "not found" and 100 sites that had been labelled as "invalid" (as described

Code	Description	Domains
PREFLIGHT ERR	Preflight request failures due to DNS, invalid HTTP Status Code (4xx or 5xx), and timeouts.	157,357
NO INPUTS FOUND	No elements suggesting the presence of search functionality on the page were detected.	329,942
INVALID INPUTS	Search was attempted but discovered elements were not interactive (could not input text and simulate user interaction).	40,368
SUCCESS	At least one interactive element was found and used to attempt to complete the search simulation.	472,333
Total		1,000,000

Table 2. Crawl Summary.

in Table 2). To select these 400 sites, we used a stratification approach to make sure that sites were evenly represented across 20 site categories and 10 Tranco ranking buckets. In order to perform validation on these sites, we first had to re-crawl them (since their status could have changed since they had initially been crawled), and could do so for 387 of them (13 of them generating preflight errors). In order to annotate them, we decided to use 3 human annotators, who were given each 155-157 sites to validate, 40 of these sites being common to all three annotators.

The annotation task took place in a custom desktop application (whose interface is shown in Appendix 2. The task given to the annotator was to look for the presence of search input that they could interact with (and perform a search). Once the site was successfully loaded using a Webview, annotators could select one of the following three possible values: “Search input present”, “No search input present”, or “Unsure”. “Unsure” was to be selected in those potentially ambiguous cases, where a search form contained a specific text input field to search for (e.g., an IP address), which is much more specific than internal site search, or when the user had to navigate away from the main page to find the search input box. The three annotators strongly agreed during the annotation task, reaching a Fleiss kappa score of 0.64 [15]. Most of the disagreements occurred because of the “Unsure” category. The annotation results are shown in Table 3, broken down by the labels of our crawler.

We then used this manually annotated dataset to validate the performance of our crawler. Excluding sites annotated as “Unsure” or “Not loading”, this validation dataset contained 214 sites where a search input was present and 127 sites where a search input was not present. Based on this sample of 337 sites, the accuracy of our crawler is 75.7% (with a precision of 88.6% and a recall of 69.1%).

3.5 Collecting Privacy Policies

To determine whether users are notified about the leakage of their search terms to third parties, we also collected the natural-language privacy policies (where possible). To collect natural-language policies, we followed a two-step approach: First, privacy policy links are identified (by checking page links in reverse order, and using English terms or detected language terms if required). Second, privacy policy documents are retrieved (by performing a simple GET request or using a headless browser if the simple GET request fails to retrieve enough content, and perform PDF to text conversion if required). Our first step follows the approach proposed in [39]. However, our keywords are not limited to English. While English is the default language used, the keywords vary according to any additional language detected during the HTML extraction step, once again leveraging translations from Mozilla’s Firefox Translation Project [50].

4 Results

In Section 3, we successfully executed a search for our dummy query string, “JELLYBEANS”, on 472,333 websites and captured the outgoing network requests with our crawler. In this section, we analyze the collected data and categorize the observed privacy leakages using the ontology presented in Figure 1, inspired by Starov et al. [72].

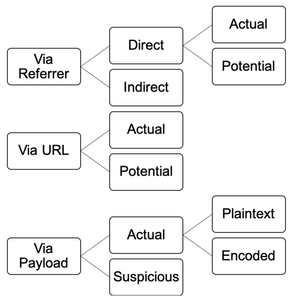


Fig. 1. Types of privacy leakages analyzed

While Starov et al. categorized privacy leakages as intentional or accidental, we think it is quite difficult to be confident whether a leakage is one or the other. Instead, we analyze leakages using several criteria. First, we focus on the mechanism by which the search query was leaked, be it via the *Referer* HTTP request header, via a *URL query parameter* or via the actual *payload* of the HTTP request. We further classify *Referer*-based leakages depending on whether our dummy query string appears in a *Referer* header that was either directly sent by the analyzed site or indirectly by one of the site’s third-parties.

Finally, we classify leakages depending on whether they actually occurred during our crawl, or whether they could potentially occur if the crawl’s scope was expanded. Since we analyze the document containing the search results, we inspect all links present on that

Crawl Status	Number	Search input present	No search input present	Unsure	Not Loading
success	176	147 [83.52%]	19 [10.80%]	9 [5.11%]	1 [0.57%]
not_found	98	10 [10.20%]	82 [83.67%]	4 [4.08%]	2 [2.04%]
invalid	113	57 [50.44%]	37 [32.74%]	13 [11.50%]	6 [5.31]%
unreachable	13	0	0	0	0
total	400	214	138	26	9

Table 3. Distribution of annotations per crawl status

page to determine whether further leakages might occur if these links were followed. Also, we break down the results in the *payload* category using additional criteria. Since we observed some large request payloads in our dataset, we suspect some of them may contain our search query, but that is not necessarily easy to confirm since some of these payloads are encoded.

All of our results are based on the number of sites where our crawler was able to execute a search without encountering any error. As presented in Table 2, this number is 472,333. It should be noted that two or more sites may redirect to the same domain. While the final number of distinct response domains is 447,428, we report all of our results based on a total of 472,333.

4.1 Leakage via *Referer*

Our dummy query was leaked via the *Referer* header by 75.8% of the sites, be it directly or indirectly. The example in Figure 2 shows both scenarios when conducting a search on the *www.autoguide.com* site.

```
{'url': 'https://ib.adnxs.com/ut/v3/prebid',
 ↪  'resourceType': 'xhr', 'headers':
 ↪  {'sec-fetch-mode': 'cors', 'referer':
 ↪  'https://www.autoguide.com/search.html?q=JEL⌋
 ↪  LYBEANS', 'payload':
 ↪  '...'}}
...
{'url': 'https://sync.placelocal.com/openadid=864⌋
 ↪  616440172932000;redirect=https%3A%2F%2Fsync.⌋
 ↪  placelocal.com%2Fsyncdatapartnersimg%3F0bypa⌋
 ↪  sssync%3D1%26blob%3Dcdc4c4f81e1770430aea8361⌋
 ↪  0fd967e8%253Aef9dc7a0939e092d667bc27fbf9dc57⌋
 ↪  060a97d39dfb6f590dd1cb76e2dbfa57e6fc69c7ec6c⌋
 ↪  fd67ef623c6836288b87e', 'resourceType':
 ↪  'image', 'headers': {'referer':
 ↪  'https://tag.placelocal.com/ad/iframe?...auc⌋
 ↪  tionid=1789060303328245535&refurl=https%253A⌋
 ↪  %252F%252Fwww.autoguide.com%252Fsearch.html%⌋
 ↪  253Fq%253DJELLYBEANS...'}}
```

Fig. 2. Example (shortened) of referrer-based leakages.

The first part of the example shows a request to a third party domain (*ib.adnxs.com*) with our search query present in the *Referer* header. Subsequently, our query string appears in the *Referer* header of a request to *sync.placelocal.com* even though this third party domain was never contacted directly by *www.autoguide.com*. This flow is visualized in Figure 3.

Direct Leakage 72.5% of sites directly send our query string to at least one third party domain via the *Referer* header. Unsurprisingly one of Google's domains is almost always present whenever this phenomenon is observed. The other entities (and associated domains) that are contacted the most frequently reveal of mix of advertizers, analytics solutions, CDNs and Consent Management Platforms. This suggests that some of the leakages observed are probably accidental (e.g., CDNs do not need access to query strings to deliver their assets successfully).

Direct leakage via referrers is due to the fact that most sites do not set any page-level *Referrer-Policy*, be it using a response header or an HTML directive (using a `meta` element). The most frequently observed pairs of policy values are shown in Table 4.

Rank	Header Policy	Meta Policy	Freq.
1	None	None	93.6%
2	`no-referrer-when-downgrade`	None	3.2%
3	(empty string)	None	1.4%
4	None	`always`	0.6%
5	None	`unsafe-url`	0.4%
6	`unsafe-url`	None	0.2%

Table 4. Most frequent page-level *Referrer-Policies*, as specified in the HTTP response header and the HTML `meta` element.

It is interesting to note that the most frequently used values are ineffective at preserving users' privacy. `no-referrer-when-downgrade` is often never relevant, as most of today's traffic is relying on HTTPS and downgrades are uncommon. `always` is not a valid policy

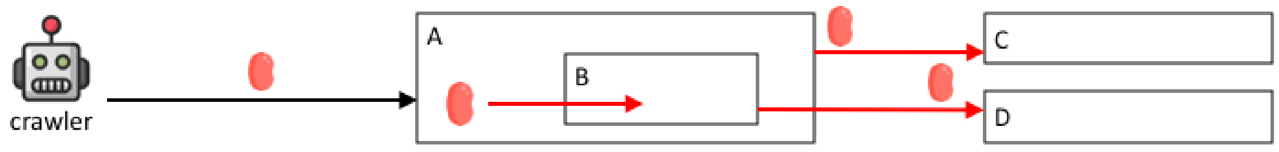


Fig. 3. Crawler searches for "JELLYBEANS" on site A. A then passes this search query to A's child `iframe` B. A then makes a network request to C and "JELLYBEANS" is leaked to C via the *Referer* header. B makes a request to D, where B inserts "JELLYBEANS" into the URL. Thus in this example "JELLYBEANS" is leaked to three domains (B, C, D) via two different methods (*Referer* header and URL).

and `unsafe-url` is discouraged except in specific use-cases, as shown in Table 1.

Indirect Leakage As shown in Figure 2, leaks via *Referer* can also occur indirectly when a third-party domain is contacted by another third-party domain. This situation was observed for 10.6% of the sites, with the most prevalent entities (based on the Disconnect list [17] of domains/entities) shown in Table 5. While entities involved in direct leakage were quite diverse, those that are involved in indirect leakages appear to be linked most often to the advertizing industry, suggesting that this type of query sharing is intentional.

Entity	Frequency
Google	7.8%
Moat	1.8%
Quantcast	1.6%
Facebook	1.3%
The Trade Desk	1.1%
VerizonMedia	0.9%
Integral Ad Science	0.7%
AK	0.7%
DoubleVerify	0.7%
Flashtalking	0.6%

Table 5. Most frequent entities whose domains are present in indirect referrers

This table shows that many of these domains are classified as tracking entities according to the Disconnect list. However, some domains, such as those associated with Google, may be more ambiguous as they might deliver actual search results via a custom search engine (*cse.google.com*) [25]. This possibility is explored in Section 4.2.

Potential Leakage To conclude this analysis of *Referer*-based leakages, we are interested in examining the links present on the pages obtained after submitting the search (i.e. most likely search results pages). This link analysis is not limited to search results links: all links present on the resulting page are inspected. We want to find out whether any of these links could leak the search query to a third-party domain if they were clicked by the user (if the query was not previously leaked to any of these third-party domains). To determine whether a link could leak or not, we analyzed the page-level referrer policies (as shown in Table 1 but also element-level policies that may be present. We found that 87.4% of sites with search functionality had links that had potentially leaking referrer policies.

Domain	Entity	Frequency
twitter.com	Twitter	44.8%
facebook.com	Facebook	38.3%
instagram.com	Facebook	34.2%
youtube.com	Google	29.2%
linkedin.com	LinkedIn	15.8%
pinterest.com	Pinterest	9.0%
google.com	Google	8.8%
wordpress.org	Wordpress	5.7%
apple.com	Apple	4.0%
vk.com	VKontakte	2.0%
t.me	Telegram	1.9%
bit.ly	Bitly	1.8%
vimeo.com	Vimeo	1.5%
flickr.com	Yahoo!	1.5%

Table 6. Most frequent domains from links on resulting pages.

The results in Table 6 show that the most prevalent domains in links on resulting pages are linked with social media services (e.g. Facebook, Twitter, Instagram, Youtube, Linkedin and Pinterest). For instance, 44% of sites included at least one link pointing to *twitter.com*, possibly in the page footer or via a widget. A generic link shortening service is also present in that list (*bit.ly*).

4.2 Leakage via URL

71% of sites send our search query as part of URL query parameters. As shown in Figure 2, however, a URL query parameter may contain the full search page

URL rather than the actual query. Distinguishing these two scenarios seems important because some of our observations (that could be qualified as *leakages*) may be absolutely necessary for the search functionality to be delivered to the user. For instance, if a website is relying on a third-party search engine provider (be it *Google Custom Search Engine* or any other service), our search query may be passed on directly to that third-party using technologies such as AJAX. In order to address this scenario, we try to identify those domains that receive specific query parameters (such as *q* or *query*) with our query string (or parts of it) as the value. By observing whether URL parameter values contain substrings of "JELLYBEANS", such as *q=J* or *q=JE* (which is the case when services provide auto-complete suggestions), we can reliably identify some actual third-party search engine providers. Some examples are provided in Appendix 5.

4.3 Leakage via Payload

Finally we are interested in examining whether sites leak search queries with third parties via the HTTP request payload. We find that 21.2% of sites actually do, the most prevalent third parties being listed in Table 7.

Domain	Entity	Frequency
facebook.com	Facebook	7.1%
hotjar.com	Hotjar	3.8%
adnxs.com	AppNexus	2.6%
criteo.com	Criteo	1.5%
pubmatic.com	PubMatic	1.5%
twitter.com	Twitter	1.0%
lijit.com	Federated Media	0.6%
sumo.com	None	0.6%
google-analytics.com	Google	0.6%
flashtalking.com	Flashtalking	0.6%
algolia.net	None	0.5%

Table 7. Most frequent domains receiving query via payload.

This list is quite varied since it contains social media entities (e.g. Facebook, Twitter), advertizers (Criteo), analytics services (sumo.com, Google Analytics) and actual search service providers (algolia.net) identified using the methodology described in the previous section. It is also worth highlighting the presence of hotjar.com, which provides session replay capabilities to site owners as reported in [20]. A deeper analysis of the request payloads (as well as the responses) should allow us to identify specific entities (e.g. search providers) as we did in the previous section using specific query parameters.

To conclude this analysis, we report that 0.9% of sites make use of third parties that rely on Base 64-encoded payloads that we were able to decode in order to detect our search query. We also report that some requests contain unusually large payloads (over 10kb per request), most often sent to entities that offer session replay services, like yandex.ru, hotjar.com, fullstory.com, smartlook.cloud, clicktale.net, quantummetric.com, sessioncam.com or inspectlet.com.

4.4 First Party Categorization

We used WebPulse [74] to group the domains according to their content category to determine if privacy leakage varies across website categories. These results can be seen in Figure 4. By and large, the worst offenders were "Personal Sites" (92.2% Any), which is consistent with the findings of [3]. The other categories of bad actors were "Restaurants/Dining/Food" (88.1%) and "Shopping" (86.7%), which might be consistent with sites that are most likely to have the most advertising. Interestingly, some of the most well-behaved sites were sites categorized as "Piracy/Copyright Concerns" (77.6%), "Suspicious" (77.5%), and "Pornography" (77.7%). This may be because these sites exist outside the usual ad ecosystem that powers the conventional web, as documented by [76]. Finally, the "Search Engines/Portals" category ironically had the lowest value for Direct Referrer leaks (55.3%), possibly because these tend to be large sites whose central focus is search, and are no doubt aware of the sensitivity of search terms and are careful against this data leaking into the hands of competitors. These results also confirm that some sites leak search queries across multiple vectors. As shown in Tables 5, 6 and 7, some entities like Google or Facebook operate across all vectors.

4.5 Analysis of Privacy Policies

Now that we have observed how sites leak search queries to third parties, we want to determine whether this is a practice that website operators explicitly mention to their users. In order to measure how often website operators in fact include this type of data leakage in privacy policy documents, we analyze natural-language privacy policies.

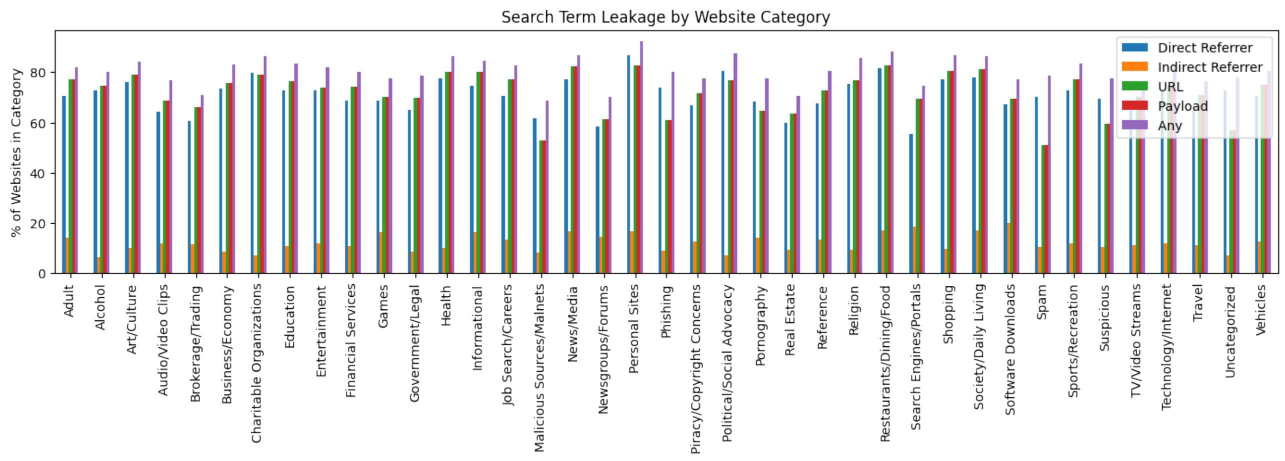


Fig. 4. Leakage breakdown by category and leakage type

Identifying Privacy Policy Documents Using our approach from Section 3.5, we were able to find privacy policy links on 50.5% of input domains (we only considered those sites leaking our search query to at least one entity). While this number may seem quite low, it is not unexpected given we relied on a fully automatic link identification approach. Previous work has shown that a word-list approach can work for specific site types (such as news) but miss links on other sites, thus requiring a complementary manual identification step [16]. Using such a manual step was not feasible based on the size of our dataset.

Using the identified links, we were able to retrieve some content 96.5% of the time. However, while the keywords used in our first step are generally a good indicator that the linked content is an actual privacy policy, there is no guarantee that this is always the case. For example, some sites rely on intermediate pages to guide users towards a relevant privacy policy if they have more than one (e.g. one per product, or one per platform). So while the link identified might point to a page containing the word *privacy*, there may actually not be any actual privacy policy content in it.

Previous studies have used some heuristics such as keyword spotting to justify their scraping [39], but we think this is too prone to errors. We therefore decided to train a classifier to filter out any content that does not appear to contain any privacy policy content. To do so, we used the sanitized OPP-115 corpus [82] as training data and relied on a one-class SVM classifier (RBF kernel with *gamma* set to *auto*) to perform outlier detection using the Scikit-learn framework [61] whose implementation is based on Libsvm [8]. Using a very small *nu* value (0.05) to reduce the number of training errors to a minimum, we extracted TF-IDF features (n-grams in the range 1-3) after removing English stopwords from the content. The resulting model detected 4 documents from our training set as outliers, and when applied to all of the 127,996 English policies retrieved detects 37.9% of them as outliers. This might be very conservative but ensures that we are looking for search-related mentions in the appropriate content.

To validate our model, we manually sampled 100 of the 79,515 English documents labelled as privacy policies by our model and found that all of them were indeed privacy policy documents. Our pipeline is almost directly comparable to the one described in [40] which achieved a privacy policy link candidate detection rate of about 43% (ours is 50.5% with duplicates and 44.7% after removing duplicates) and an actual privacy policy document detection rate of about 15% using a CNN (ours is 20.7%). Our results are summarized in Table 8. We also make available all discovered privacy policy links to future researchers [9].

Parsing Privacy Policies To investigate whether websites inform users of the fact that their search queries may be sent to third parties, we must parse the extracted privacy policies and find sections relevant to processing search terms. Since search queries are instances of personal information (rather than personally identifiable information), we would expect privacy policies to mention them in sections pertaining to personal data collection or handling. In order to verify this hypothesis, we manually check the annotations from the OPP-115 corpus to find occurrences of phrases

9 https://jellybeans-paper.com

Description	Count
Input domains	384101
Domains with policy link identified	193815
Link found but scraping error	6703
Link found and scraping success	187112
Uniq. links with document	165139
Uniq. documents detected as English	127996
Uniq. English docs labelled as privacy policies	79515
Uniq. English docs including search-related terms	10914
Uniq. English docs including generic info. sharing	60212

Table 8. Analysis of privacy policies statistics. Only domains with observed search term leakage were used as input domains.

containing the token *search*. While we find such mentions in the corpus (as shown in Table 9), we notice that search queries collection is explicitly mentioned in segments labelled as "First Party Collection/Use" by the OPP corpus annotators more frequently than segments labelled as "Third Party Sharing/Collection" (24 vs. 11). A closer inspection of these "Third Party Sharing/Collection" segments reveals that some vague or generic phrases are used to inform the reader that some of their information (without necessarily specifying which) may be shared with third parties. Examples of such generic phrasing is also shown in Table 9.

Based on this discovery, we decided to manually review all 672 segments (originating from 112 distinct privacy policies of the OPP-115 corpus) to identify those that seem to mention (either explicitly or implicitly) the collection by a third party of information (such as search queries) or the sharing of such information with a third party when the user interacts with the site. The results of this manual review are as follows: 116 segments seem to mention such a practice (labelled as "yes") while 462 do not (labelled as "no") and for 94 segments, we are unable to decide (labelled as "unsure").

We further examined these 116 "yes" segments to extract the sentences (or parts of sentences) that refer to the sharing of information with a 3rd party. Once these parts were identified, we defined patterns using morphological and syntactic information with a view to create detection rules of such mentions in newly collected privacy policies. We use rules instead of a machine learning approach due to the small amount of labelled data (116 sentence parts). These rules were implemented using Spacy's entity ruler [29] so that a sentence was deemed to mention information sharing with a third party if it contained the following elements: a third party (which may also be described using words such as *vendor* or *partner*, a verb describing the act of sharing or collecting, such as *disclose* or *gather*, and a reference to *information* or *data*. In order to avoid some false positives in specific contexts (such as information sharing with law enforcement entities or information transfer in the event of a corporate merger), we penalized sentences containing specific terms. These rules showed an accuracy of 89% when run on the annotated segments from the OPP-115 corpus. We then used these rules on the 79,515 privacy policies we collected and found that 60,212 of them (about 75%) contained at least one mention of information sharing with a third party. An example of detection of a generic mention is: "These third-party ad servers or ad networks can use technology which displays the advertisements that appear on our site directly to your browser, enabling them to collect information."[10] 10,914 of these policies (about 13%) did mention search terms explicitly, an example being: "We and our service providers and third-party advertising partners also use these technologies to track your activities on our Site and other websites, including the websites and web pages that you visited, the ads or content that you clicked on, any items you may have purchased and the search terms you used, in order to deliver tailored advertising to you."[11] These results must be considered as upper bounds since our detection rules can still be prone to false positives. To validate our detection results, we read 12 of the 100 validation documents manually and confirmed our automated approach did not find any mention of users' search terms in this sample. However, 2 of these 12 documents contained some mention of user interaction with some services and 8 other documents in the sample did mention users' *usage* of the site/services, which could encompass the use of the search functionality.

5 Discussion

In this section, we summarize our findings, provide a discussion on how privacy policies handle search term processing, introduce some countermeasures, and list the limitations of our study.

10 http://www.miicharacters.com/privacy.php
11 https://www.audi-forums.com/help/privacy

Source	Category	Text	Match type
1703_sports-reference.com.csv	First Party Collection and Use	SRL may record information identifying the visitor or linking the visitor to the **search perform**ed.	explicit
1300_bankofamerica.com.csv	Third Party Sharing and Collection	certain information about your activities on our Sites, such as pages visited and **search key** words entered	explicit
1070_wnep.com.csv	Third Party Sharing and Collection	We may share your information with third-party advertisers and advertising networks or others with which we have a contractual relationship	generic
175_mlb.mlb.com.csv	Third Party Sharing and Collection	A permitted network advertiser may use cookies, web beacons or similar technologies to collect information about your interaction with our Services	generic

Table 9. Examples of relevant OPP-115 corpus fragments.

5.1 Findings

As mentioned in Section 4, we executed a search for our dummy query string, "JELLYBEANS", on 472,333 websites. 81.3% of these leak search terms to third parties in some form, according to our definition of leakage in Section 2.4. 75.8% of websites leaked data via the *Referer* header (72.5% directly and 10.6% indirectly); 71% via URL query parameters; 21.2% via the payload. Additionally 87.4% of websites had the potential to leak sensitive query terms to a new third-party if a link on the page was clicked.

The high incidence of leakage via the *Referer* header is likely related to the fact that over 93.6% of sites did not explicitly specify a *Referrer-Policy*, and when they did it was either at the same level or worse than the default. After we completed our crawl, the Google Chrome team announced that starting with Chrome version 85, the default *Referrer-Policy* will be `strict-origin-when-cross-origin` [54]. This change should have a significant positive change on the privacy of users on the web, especially if adopted by the other major browsers.

Most sites (~80%) leaked search terms through more than one vector, with ~20% of sites leaking queries via at least three vectors. This is extremely concerning, as often times these search terms refer to sensitive topics such as medical conditions, sexual preferences, racial identity, or financial situation. Even for seemingly innocuous cases such as shopping, products can reveal sensitive information about users - consider products such as pregnancy tests, wedding bands, or anti-anxiety medication.

5.2 Privacy Policies

The concept of consumers receiving clear notice about privacy is key to many privacy frameworks, such as the OECD's 1980 privacy guidelines [59], the U.S. Federal Trade Commission's Fair Information Practice Principles [13] and the European parliament's latest regulation (GDPR) [60]. However, most of these framework do not mandate explicitly what should or should not be included in privacy policy, which can lead to wide differences, even within the same industry. For instance, a previous study evaluated 6,191 U.S. financial institutions' Web privacy notices and found large variance in stated practices, even among institutions of the same type [14]. In 2016 another study found that "the average policy complies with 39% of the FTC guidelines issued in 2012" [45]. While recent legislation such as the California Consumer Privacy Act (CCPA) [7] has been attempting to provide further guidance on how consumers should be informed of personal information collection, it mentions informing "consumers as to the categories of personal information to be collected and the purposes for which the categories of personal information shall be used", without listing what these categories are. This situation provides some context for the results presented in Section 4.5. Our results suggest that privacy policies tend to be worded in such a way that *search query handling* by a third party is very often described in a very generic manner. This means that users who are looking for an answer to the question *are my search terms shared with or collected by a third party?* will have to carefully read the entire privacy policy (or specific sections of it) to find such information (instead of simply searching for a keyword like *search* using a browser shortcut). In practice, this means that most users will not have the time to perform such a careful review and will be unaware that their search terms won't be kept private by the site they are interacting with. This is something we decided to address by designing a novel browser extension, as described in Section 5.3 on countermeasures.

5.3 Countermeasures

A few options are available to address the privacy leakages identified in this paper (with a full comparison included in Appendix 3.). First, some browsers such as Chrome v85+ and Safari have changed their default Referrer-Policy to the more secure "strict-origin-when-cross-origin" value, which provides a solution to referrer-based leakages. We would like other popular browsers to adopt this default, and have standards bodies such as W3C change the guidance for browsers to make this the default value. We would also like to see tracking protection tools (such as those built into Firefox) flag sites that downgrade the *Referrer-Policy* using a server-side header or HTML, and prevent that behavior when tracking protection has been turned on by the user. In the meantime, we developed a browser extension which displays our findings for each crawled site (two screenshots are included in Appendix 4). This extension also links to the privacy policy where possible, providing privacy-savvy users with a way to quickly identify which third parties they are about to share their search queries with. Besides providing some awareness to users, the extension could also be responsible for dropping (or replacing with dummy placeholder terms) search queries whenever these queries are found in Referrer values, URL parameters or even payload fields that are about to be sent to hosts that are not providing search functionality. In Section 4.2, we showed that we were able to identify some third-party search providers so an Allowed List could be used and refined over-time. Modifying leaking requests (by tweaking a specific parameter for example) seems less risky than blindly blocking requests as these requests may provide additional functionality that could prevent the site from operating normally. None of these countermeasures, however, solves the situation where third-party JavaScript gets included into a first-party context, as it can access all first-party data, perform arbitrary obfuscations to it, and export it. To prevent this scenario, one could encourage website owners to make sure that non-search third-party JavaScript does not get included in first-party contexts, by leveraging iframes for example. However, it is not clear that all website owners would have the skills (or time) to deploy such a solution (as suggested by the currently low deployment of Referrer policies, which are quick and easy to roll out). Alternatively, the browser extension could also be tuned to re-implement native JavaScript APIs methods (leveraging Proxies and Reflection) to prevent access from third-parties found within first-party contexts (this is possible by throwing deliberate exceptions and performing domain look-ups in the stack traces). While this approach requires substantial implementation efforts it would guarantee fine-grained controls over which entities are able to read values from search input fields. Some of these techniques have been covered in previous works, such as [42], [47], [77] and [19].

5.4 Limitations

Crawling Limitations. During the crawl process, we failed to visit 15.7% of domains, which is roughly in line with the expected results in [1]. Instead of relying on a single crawling node (in the US West region), distributing the crawling to multiple regions may have improved the coverage. However, we made few efforts to circumvent IP-blocking or solve CAPTCHAs during our crawl. Some other teams use Tor to evade IP-blocking, while others employ anti-crawling-detection stealth toolkits. We do not believe such approaches are consistent with the spirit of the web, and allowed for such failures. However, it may be the case that these 15.7% of websites differ in some key characteristics from the ones that we successfully visited.

We made every effort to locate the correct internal search inputs on every website regardless of language. However, our approach was not perfect. Our validation tests show that our crawler was unable to detect any search input for less than 5% of the sites that actually contained a search input.[12] For example, we only visited the landing page of each webpage, and did not navigate the site to search for the internal search feature on other pages. In other cases, our detection approach proved too aggressive so we executed some search using some detected inputs that were not actual visible search inputs. Our validation tests show that it happened for 14% of the sites that did not contain any search input. We also made some effort to interact successfully with the search inputs we detected. However, in some cases the interaction was too complex and we failed to execute the search and obtain search results. Our validation

12 This number is derived from the 214 times annotators used the label *Search input present*. 10 of out these 214 sites were labeled as *not_found* by the crawler.

tests show that we were unable to interact successfully with a detected search input for 26% of the sites that actually contained a search input. We anticipate future work will improve on our approach and be able to find and interact with more search inputs in a more reliable manner. For this reason, we will release the labeled corpus of detected search input selectors for each domain under open source for other researchers to use.

Analysis Limitations. During our analysis, we found some cases of large payloads sent to third party sites, which appeared to be encrypted or encoded, and which we suspected contained the "JELLYBEANS" search string. We utilized Ciphey to try to decode all large payloads, but sometimes were not successful [6]. Therefore, the payload leakage may in fact be higher than reported in this paper. Using our approach from Section 3.5, we were able to find (potential) privacy policy links on 50.5% of input domains. Using the identified links, we were able to automatically retrieve some content 96.5% of the time. It is obvious that a manual inspection of the sites and links would have helped us recovered more privacy policy links and content. However, using such a manual step was not feasible based on the size of our dataset. We then parsed the extracted privacy policies to find sections relevant to processing search terms, using rules derived from the OPP-115 corpus. It is possible that the rules performed less effectively on newly collected policies, so future work would be required to improve these rules. In addition to privacy policies, some sites also have a cookie policy document or a cookie banner. While we looked for mentions of search term handling and information sharing in privacy policies, we did not analyze these cookie-related documents. However, we do not believe that this would substantially change our results.

6 Related Work

Crawling for Privacy Leaks. The use of web crawlers to measure privacy leakages is ubiquitous in the literature: a recent survey paper by Ahmad et. al. [1] found that 350 papers published at top-tier venues from 2015-2020 relied on data gathered from web crawlers. These web crawlers were most often implemented as headless browsers on top of the PhantomJS [62], Selenium [70], OpenWPM [22], or Puppeteer [11] frameworks. Such crawlers were often deployed in large-scale scans to discover privacy leakages around the web, often using either the Tranco or the Alexa [2] top domain datasets as inputs. The resulting studies measured privacy leakages found in email tracking [21], link tracking [66], third-party cookies [67] (used on 70% of sites for tracking purposes), the ad ecosystem [23], HTTP headers [35], contact forms [72] (where 2.5% of the total number of contact forms leaked the user's PII through the URL query string), and registration forms [9]. Many other crawls characterized privacy leakages (often by looking at domains of third-party requests) more broadly on the web [3, 34, 38, 39, 69], or in specific sectors such as adult websites [44, 76], health websites [37], online collaboration services [31], and the financial sector [65]. Ahmad et al. noted that challenges exist in crawling the web in an automated way: only approximately 77-93% of the top 500 domains can be successfully reached by a crawler [13], including blocking due to regional restrictions, IP-based blocking, and encountering CAPTCHAs.

Privacy of the *Referer* HTTP header. This header has been known to be a potential source of privacy leakage since at least 2011 [18, 34]. The *Referrer-Policy* HTTP header was specifically created for website owners to control the leakage of the referrer [18]. The issue of potentially sensitive queries being leaked in referrer values was showcased by Krishnamurthy et al. using the example of searching for pancreatic cancer on a health information website, and having that search string transmitted to third parties [34]. Similar examples were shown in work done by Libert in 2014 [37]. Lavrenovs et al. demonstrated in 2018 that the *Referrer-Policy* HTTP header was only explicitly set in 0.05% of HTTP responses and 0.33% of HTTPS responses [35], though this work looked at HTTP headers only, while the policy may also be set inside HTML. In 2017, Dolnak characterized the *Referrer-Policy* HTTP header status of 7 million websites and suggested 56% of those websites might leak sensitive queries via the *Referer* HTTP header[18].

Search Privacy. Search queries, especially over a longer time period and associated with a specific IP address, are well-known to lead to significant loss of user privacy. [24] investigated the network properties of the third party referral structures used for the delivery of personalized ads. Jones et al. showed that search queries can be used to efficiently de-anonymize users [30]. White et al. showed that it was possible to use search queries to diagnose neurodegenerative disorders [81]. Libert showed that health terms in medical search

13 depending on the crawler used

engines were leaked in 2014 [37]. However, Libert started his analysis by searching for likely medical terms using popular search engines, and then analyzed the resulting 80,142 pages for privacy leakage. Note that this set of pages corresponds to a much smaller set of domains, as Libert includes multiple pages per domain in his analysis. In contrast to this approach, we focus on performing an internal site search directly on each domain, which requires a more complex crawler to correctly locate the search input fields. We also scan a much larger set of domains: 1 million in all, and do not restrict ourselves to the health and medical field.

Privacy Policies. Privacy policies provide another way to measure the intended privacy practices of a company [4], but previous work has shown that their usefulness can be limited due to their complexity [46]. The adoption of privacy policies by companies has been measured in numerous studies [16, 41, 68, 83]. Some of these studies have focused on specific platforms (such as mobile apps [84] or specific categories of web sites, such as adult sites [76] [44]. For instance, Maris et al. used the `webXray` tool to crawl and analyze 22,484 adult websites for privacy leakages.[44] They developed a tool called `policyXray` to locate privacy policy links on the pages, looking for links with text such as "privacy" and "privacy policy". The privacy policy text was then extracted using Google Chrome's `Readability.js` library, and its reading difficulty and time to read was assessed. Maris et al. also extracted any explicitly disclosed third parties in the privacy policy. In this work, Maris et al. did not try to perform a more in-depth linguistic analysis of the extracted privacy policies. Finally, privacy policies have also been recently analyzed automatically using deep learning so that they can be queried by users using natural language questions [28].

Users' Perception of Search Privacy. Many studies have been conducted to analyze users' perception of privacy. For instance, standardized privacy policy information formats were designed to determine whether users would find them more useful than traditional policies [33]. However, very few have specifically analyzed the users' mental models of online search activity. A recent study focusing on a tracking visualization tool [80] did find that a majority of users did not want to have their search activity tracked, while a previous study found that lay people had simpler mental models than technical people - their models omitting concepts such as Internet levels and entities [32] (suggesting that a very large number of users does not realize that their search queries are shared with third parties).

7 Conclusions and Future Work

We showed how terms entered into the internal site search feature of the Tranco top 1 million websites are leaked to third parties, compromising user privacy, but without necessarily allowing these third parties to have a long-term or comprehensive picture of a user's activity. We developed a crawler which can interact with modern dynamic websites, identify search inputs, and capture outgoing network requests. Our crawler was able to locate an internal site search feature on 512,701 websites, achieving 75.7% accuracy in executing actual search queries on these sites. We analyzed privacy leakage across three vectors: (i) *Referer* HTTP header; (ii) URL including query parameters; (iii) request payload. We detected 81.3% of these websites leaking the "JELLYBEANS" string to third-party domains through at least one of these three vectors. We released a browser extension to inform users about potential privacy leaks via internal search, based on our findings. We also developed a novel technique to analyze natural language privacy policies and determine whether they mention sharing search terms with third parties. We extracted privacy policies from about 50% of the websites leaking search queries, and found that only 13% of those privacy policies explicitly mentioned search terms. In most privacy policies (about 75%), the sharing of information (which may include search terms) is also described using generic wording.

Future Work The present work has identified a few areas that would require further investigation. Specifically, we would like to improve the coverage of our crawler to (i) detect those search inputs that require more complex interaction and (ii) analyze payloads that are not in plain text. We would also like to refine our crawling framework to tackle other types of user input (e.g., chat widgets). Finally, we would like to contrast our results with users' expectations of search privacy. Previous work [80] has suggested that most users do not want to have their search activity tracked. We would like to know how they react when they are shown the amount/type of entities receiving their search queries.

Acknowledgements

This research was funded by NortonLifeLock and the authors would like to thank Leyla Bilge, Kevin Roundy, Ethan Schatz, Kirsten McMullen and Jane Christie for providing comments on earlier versions of their paper.

References

[1] 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

[2] Alexa Internet Inc. 2020. Alexa Top Sites. https://www.alexa.com/topsites.

[3] Amirhossein Aleyasen, Oleksii Starov, Alyssa Phung Au, Allan Schiffman, and Jeff Shrager. 2015. On the Privacy Practices of Just Plain Sites. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society (WPES '15)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2808138.2808140

[4] Ryan Amos, Gunes Acar, Elena Lucherini, Mihir Kshirsagar, Arvind Narayanan, and Jonathan Mayer. 2020. Privacy Policies over Time: Curation and Analysis of a Million-Document Dataset. arXiv:cs.CY/2008.09159

[5] Michael Barbaro and Tom Zeller Jr. 2006. A Face Is Exposed for AOL Searcher No. 4417749. https://www.nytimes.com/2006/08/09/technology/09aol.html.

[6] Bee and Ciphey collaborators. 2008. Ciphey. https://github.com/Ciphey/Ciphey.

[7] CCPA. 2018. California Consumer Privacy Act of 2018. https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5.

[8] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (2011), 27:1–27:27. Issue 3.

[9] Manolis Chatzimpyrros, Konstantinos Solomos, and Sotiris Ioannidis. 2020. You Shall Not Register! Detecting Privacy Leaks Across Registration Forms. In *Computer Security*. Springer International Publishing, Cham, 91–104.

[10] Chrome DevTools Team. 2020. Chrome DevTools Protocol. https://chromedevtools.github.io/devtools-protocol/.

[11] Chrome DevTools Team. 2020. Puppeteer. https://github.com/GoogleChrome/puppeteer.

[12] Chrome DevTools Team. 2020. Puppeteer Chrome DevTools Protocol Session. https://devdocs.io/puppeteer/index#class-cdpsession/.

[13] Federal Trade Commission. 1998. Privacy online: A report to Congress. https://www.ftc.gov/sites/default/files/documents/reports/privacy-online-report-congress/priv-23a.pdf.

[14] Lorrie Faith Cranor, Pedro Giovanni Leon, and Blase Ur. 2016. A Large-Scale Evaluation of U.S. Financial Institutions' Standardized Privacy Notices. *ACM Trans. Web* 10, 3, Article 17 (Aug. 2016), 33 pages. https://doi.org/10.1145/2911988

[15] Mark Davies and Joseph L. Fleiss. 1982. Measuring Agreement for Multinomial Data. *Biometrics* 38, 4 (1982), 1047–1051.

[16] Martin Degeling, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz. 2019. We Value Your Privacy ... Now Take Some Cookies - Measuring the GDPR's Impact on Web Privacy. *Inform. Spektrum* 42, 5 (2019), 345–346. https://doi.org/10.1007/s00287-019-01201-1

[17] Disconnect. 2020. The Tracker Protection lists. https://github.com/disconnectme/disconnect-tracking-protection.

[18] Ivan Dolnák. 2017. Implementation of referrer policy in order to control HTTP Referer header privacy. In *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. IEEE, USA, 1–4.

[19] Xinshu Dong, Minh Tran, Zhenkai Liang, and Xuxian Jiang. 2011. AdSentry: Comprehensive and Flexible Confinement of JavaScript-Based Advertisements. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*. Association for Computing Machinery, New York, NY, USA, 297–306. https://doi.org/10.1145/2076732.2076774

[20] Steven Englehardt. 2017. No boundaries: Exfiltration of personal data by session-replay scripts. https://freedom-to-tinker.com/2017/11/15/no-boundaries-exfiltration-of-personal-data-by-session-replay-scripts/.

[21] Steven Englehardt, Jeffrey Han, and Arvind Narayanan. 01 Jan. 2018. I never signed up for this! Privacy implications of email tracking. *Proceedings on Privacy Enhancing Technologies* 2018, 1 (01 Jan. 2018), 109 – 126. https://doi.org/10.1515/popets-2018-0006

[22] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of ACM CCS 2016*. Association for Computing Machinery, New York, NY, USA, 1388–1401.

[23] Kiran Garimella, Orestis Kostakis, and Michael Mathioudakis. 2017. Ad-blocking: A study on performance, privacy and counter-measures. In *Proceedings of the 2017 ACM on Web Science Conference*. Association for Computing Machinery, Troy, New York, USA, 259–262.

[24] Richard Gomer, Eduarda Mendes Rodrigues, Natasa Milic-Frayling, and M.C. Schraefel. 2013. Network Analysis of Third Party Tracking: User Exposure to Tracking Cookies through Search. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Vol. 1. IEEE, USA, 549–556. https://doi.org/10.1109/WI-IAT.2013.77

[25] Google. 2020. Google's Programmable Search Engine. https://developers.google.com/custom-search/docs/overview.

[26] Google. 2021. Sitelinks search box. https://developers.google.com/search/docs/data-types/sitelinks-searchbox.

[27] Saikat Guha, Bin Cheng, and Paul Francis. 2010. Challenges in measuring online advertising systems. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. Association for Computing Machinery, New York, NY, USA, 81–87.

[28] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. 2018. Polisis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning. arXiv:cs.CL/1802.02561

[29] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. https://doi.org/10.5281/zenodo.1212303

[30] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2007. " I know what you did last summer" query logs and

user privacy. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. Association for Computing Machinery, New York, NY, USA, 909–914.

[31] Beliz Kaleli, Manuel Egele, and Gianluca Stringhini. 2019. On the Perils of Leaking Referrers in Online Collaboration Services. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, Germany, 67–85.

[32] Ruogu Kang, Laura Dabbish, Nathaniel Fruchter, and Sara Kiesler. 2015. "My Data Just Goes Everywhere": User Mental Models of the Internet and Implications for Privacy and Security. In *Proceedings of the Eleventh USENIX Conference on Usable Privacy and Security (SOUPS '15)*. USENIX Association, USA, 39–52.

[33] Patrick Gage Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor. 2010. Standardizing Privacy Notices: An Online Study of the Nutrition Label Approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. Association for Computing Machinery, New York, NY, USA, 1573–1582. https://doi.org/10.1145/1753326.1753561

[34] Balachander Krishnamurthy, Konstantin Naryshkin, and Craig Wills. 2011. Privacy leakage vs. protection measures: the growing disconnect. In *In Web 2.0 Workshop on Security and Privacy*, Vol. 2. IEEE, USA, 1–10.

[35] Arturs Lavrenovs and F Jesús Rubio Melón. 2018. Http security headers analysis of top one million websites. In *2018 10th International Conference on Cyber Conflict (CyCon)*. IEEE, USA, 345–370.

[36] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. TRANCO: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. https://doi.org/10.14722/ndss.2019.23386

[37] Tim Libert. 2014. Privacy implications of health information seeking on the web.

[38] Timothy Libert. 2015. Exposing the hidden web: An analysis of third-party HTTP requests on 1 million websites.

[39] Timothy Libert. 2018. An automated approach to auditing disclosure of third-party data collection in website privacy policies. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 207–216.

[40] Thomas Linden, Rishabh Khandelwal, Hamza Harkous, and Kassem Fawaz. 01 Jan. 2020. The Privacy Policy Landscape After the GDPR. *Proceedings on Privacy Enhancing Technologies* 2020, 1 (01 Jan. 2020), 47 – 64. https://doi.org/10.2478/popets-2020-0004

[41] Chang Liu and Kirk P. Arnett. 2002. Raising a Red Flag on Global WWW Privacy Policies. *Journal of Computer Information Systems* 43, 1 (2002), 117–127. https://doi.org/10.1080/08874417.2002.11647076

[42] Mike Ter Louw, Karthik Thotta Ganesh, and V.N. Venkatakrishnan. 2010. AdJail: Practical Enforcement of Confidentiality and Integrity Policies on Web Advertisements. https://www.usenix.org/conference/usenixsecurity10/adjail-practical-enforcement-confidentiality-and-integrity-policies-web

[43] Delfina Malandrino, Andrea Petta, Vittorio Scarano, Luigi Serra, Raffaele Spinelli, and Balachander Krishnamurthy. 2013. Privacy awareness about information leakage: Who knows what about me?. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. Association for Computing Machinery, New York, USA, 279–284.

[44] Elena Maris, Timothy Libert, and Jennifer R Henrichsen. 2020. Tracking sex: The implications of widespread sexual data leakage and tracking on porn websites. *New Media & Society* 22, 11 (2020), 2018–2038.

[45] Florencia Marotta-Wurgler. 2016. Understanding Privacy Policies: Content, Self-Regulation, and Markets. , 43 pages. https://doi.org/10.2139/ssrn.2736513

[46] Aleecia M. McDonald, Robert W. Reeder, Patrick Gage Kelley, and Lorrie Faith Cranor. 2009. A Comparative Study of Online Privacy Policies and Formats. In *Privacy Enhancing Technologies*, Ian Goldberg and Mikhail J. Atallah (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 37–55.

[47] Leo A. Meyerovich and Benjamin Livshits. 2010. ConScript: Specifying and Enforcing Fine-Grained Security Policies for JavaScript in the Browser. In *2010 IEEE Symposium on Security and Privacy*. IEEE, USA, 481–496. https://doi.org/10.1109/SP.2010.36

[48] Mozilla. 2020. Query selector. https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector/.

[49] Mozilla. 2021. The fetch API. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API.

[50] Mozilla. 2021. Firefox Translation Project on Pontoon. https://pontoon.mozilla.org/projects/firefox.

[51] Mozilla. 2021. The language global attribute. https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/lang.

[52] Mozilla. 2021. OpenSearch description format. https://developer.mozilla.org/en-US/docs/Web/OpenSearch.

[53] Mozilla Foundation. 2021. The Public Suffix List. https://publicsuffix.org/.

[54] Maud Nalpas. 2020. A new default Referrer-Policy for Chrome: strict-origin-when-cross-origin. https://developers.google.com/web/updates/2020/07/referrer-policy-new-chrome-default.

[55] Maud Nalpas. 2020. Referer and Referrer-Policy best practices. https://web.dev/referrer-best-practices/.

[56] Henrik Nielsen, Roy T. Fielding, and Tim Berners-Lee. 1996. Hypertext Transfer Protocol – HTTP/1.0. RFC 1945. https://doi.org/10.17487/RFC1945

[57] Henrik Nielsen, Jeffrey Mogul, Larry M Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. 1999. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616. https://doi.org/10.17487/RFC2616

[58] Node.js core collaborators. 2021. Node Worker Threads. https://nodejs.org/docs/latest-v12.x/api/worker_threads.html.

[59] OECD. 2013. Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. https://www.oecd.org/sti/ieconomy/oecd_privacy_framework.pdf.

[60] European Parliament. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons (General Data Protection Regulation). https://eur-lex.europa.eu/legal-

content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[62] PhantomJS contributors. 2018. PhantomJS. https://github.com/ariya/phantomjs.

[63] Privacy Alliance. 2020. Guidelines for Online Privacy Policies. http://www.privacyalliance.org/resources/ppguidelines.

[64] Privacy Community Group. 2020. The First-Party Sets. https://github.com/privacycg/first-party-sets.

[65] Agus Purwanto and Andi Wahju Rahardjo Emanuel. 2020. The State of Website Security Response Headers in Indonesia Banking.

[66] Iskander Sanchez-Rola, Davide Balzarotti, Christopher Kruegel, Giovanni Vigna, and Igor Santos. 2020. Dirty Clicks: A Study of the Usability and Security Implications of Click-related Behaviors on the Web. In *Proceedings of The Web Conference 2020*. Association for Computing Machinery, New York, NY, USA, 395–406.

[67] Iskander Sanchez-Rola, Matteo Dell'Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos. 2019. Can I Opt Out Yet? GDPR and the Global Illusion of Cookie Control. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 340–351.

[68] Kanthashree Mysore Sathyendra, Florian Schaub, Shomir Wilson, and Norman Sadeh. 2016. Automatic extraction of opt-out choices from privacy policies. In *FS-16-01 (AAAI Fall Symposium - Technical Report)*. AI Access Foundation, United States, 270–275.

[69] Sebastian Schelter and Jérôme Kunegis. 2016. On the ubiquity of web tracking: Insights from a billion-page web crawl.

[70] Selenium contributors. 2020. Selenium. https://github.com/SeleniumHQ/selenium.

[71] Rebecca Sentance. 2020. 24 best practice tips for ecommerce site search. https://econsultancy.com/24-best-practice-tips-for-ecommerce-site-search.

[72] Oleksii Starov, Phillipa Gill, and Nick Nikiforakis. 01 Jan. 2016. Are You Sure You Want to Contact Us? Quantifying the Leakage of PII via Website Contact Forms. *Proceedings on Privacy Enhancing Technologies* 2016, 1 (01 Jan. 2016), 20 – 33. https://doi.org/10.1515/popets-2015-0028

[73] Ian Storm Taylor. 2019. Add a timeout option to prevent hanging. https://github.com/whatwg/fetch/issues/951.

[74] Symantec. 2020. Webpulse. https://sitereview.bluecoat.com/#/category-descriptions.

[75] Bill Tancer. 2008. *Click: What millions of people are doing online and why it matters.* Hachette Books, London, UK.

[76] Pelayo Vallina, Álvaro Feal, Julien Gamba, Narseo Vallina-Rodriguez, and Antonio Fernández Anta. 2019. Tales from the porn: A comprehensive privacy analysis of the web porn ecosystem. In *Proceedings of the Internet Measurement Conference*. Association for Computing Machinery, New York, NY, USA, 245–258.

[77] Steven Van Acker, Philippe De Ryck, Lieven Desmet, Frank Piessens, and Wouter Joosen. 2011. WebJail: Least-Privilege Integration of Third-Party Components in Web Mashups. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*. Association for Computing Machinery, New York, NY, USA, 307–316. https://doi.org/10.1145/2076732.2076775

[78] W3C. 2019. Referrer Policy Editor's Draft, 4 December 2019. https://w3c.github.io/webappsec-referrer-policy/#referrer-policies.

[79] W3C Schema.org Community Group. 2015. Search Action. https://schema.org/SearchAction.

[80] Ben Weinshel, Miranda Wei, Mainack Mondal, Euirim Choi, Shawn Shan, Claire Dolin, Michelle L. Mazurek, and Blase Ur. 2019. Oh, the Places You've Been! User Reactions to Longitudinal Transparency About Third-Party Web Tracking and Inferencing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. Association for Computing Machinery, New York, NY, USA, 149–166. https://doi.org/10.1145/3319535.3363200

[81] Ryen W White, P Murali Doraiswamy, and Eric Horvitz. 2018. Detecting neurodegenerative disorders from web search signals. *NPJ digital medicine* 1, 1 (2018), 1–4.

[82] Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, Thomas B. Norton, Eduard Hovy, Joel Reidenberg, and Norman Sadeh. 2016. The Creation and Analysis of a Website Privacy Policy Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1330–1340. https://doi.org/10.18653/v1/P16-1126

[83] Razieh Nokhbeh Zaeem and K. Suzanne Barber. 2017. A study of web privacy policies across industries. *Journal of Information Privacy and Security* 13, 4 (2017), 169–185. https://doi.org/10.1080/15536548.2017.1394064

[84] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel Reidenberg, N. Cameron Russell, and Norman Sadeh. 01 Jul. 2019. MAPS: Scaling Privacy Compliance Analysis to a Million Apps. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (01 Jul. 2019), 66 – 86. https://doi.org/10.2478/popets-2019-0037

Appendix 1: Crawler Challenges and Edge Cases

Interacting with the Right Inputs

As part of *Stage III* results, inputs not related to search are mistakenly matched because some of their attributes erroneously trigger our input detection query selectors. To anticipate and minimize the impact of triggering non-search functionalities, we interact with each input individually (one per headless browser visit) by following these steps: (i) detect the presence of a single input element via provided query selector; (ii) focus on the input element; (iii) type "JELLYBEANS"; (iv) wait for 500ms; and (v) type the return key. By doing so, we avoid interacting with non-search-related functionalities. For example, login forms require multiple fields to be filled out before triggering a useful action when the enter key is hit.

Dynamically Generated Attributes

Some websites dynamically generate random attribute names for their elements for each page visit. We detect these cases when *Stage IV* yields exceptions indicating that none of the elements collected during *Stage III* are found. To address the problem, we merge both stages in a single fallback task.

Hidden Search Inputs

Other websites do not explicitly display the search functionality to visitors, in some cases input elements are not even detectable using JavaScript. To work around this problem, we rely on the elements previously categorized as search-related (captured in *Stage III*). We can then infer from its attributes values that a search box is likely to be made available upon click. We validate the hypothesis by instrumenting the crawler to simulate a click and waiting for navigation until any elements of type `input` are displayed or a timeout period is met.

Search Results Displayed Within Another Tab

Some websites deliberately display search results within a new browser tab. This situation requires the crawler to specifically listen and intercept all new tab creation events and capture any links associated with the ongoing search simulation. Whenever this happens a new visit to the captured URL is independently performed.

Appendix 2: Annotation Tool

Figure 5 shows a website loaded in the custom annotation environment that was specifically developed to create our validation dataset. In this example, we manually inserted a red rectangle to show what the annotators had to look for (e.g., a magnifying glass that would bring up an embedded search form after being clicked). While the annotators did not have access to such visual components in the tool itself, we made annotated screenshots (taken during the crawling step) available to them to avoid missing search inputs. In some cases, search inputs can be quite difficult to find in a page, so having access to the screenshots when the crawler was able to detect and interact with a search input proved extremely useful.

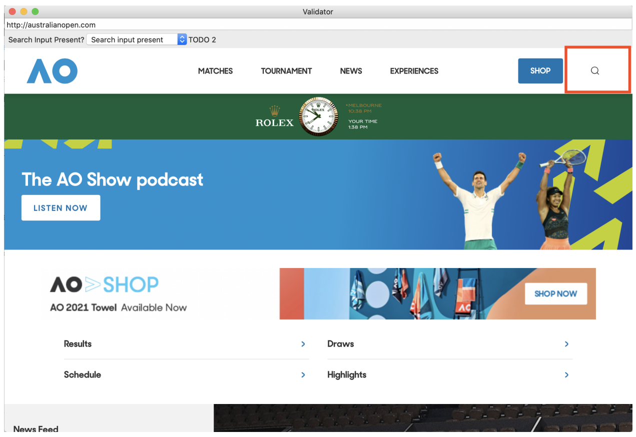


Fig. 5. Example of a web site to be annotated based on the presence or absence of a search input.

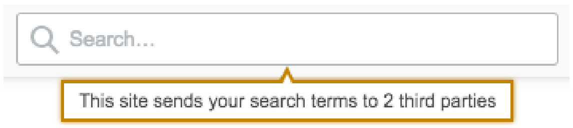
The three annotators were the authors of this paper, and their disagreements were resolved by consensus after discussion. In these cases, the authors manually inspected the sites to achieve a high degree of confidence.

Appendix 3: Comparison of Countermeasures

Countermeasure	Pros	Cons
Raise awareness via browser extension	Educates unaware users.	Decision to proceed with search is left to the user.
Re-implement native JavaScript APIs methods (leveraging Proxies and Reflection)	Provides fine-grained access controls to search inputs; Can effectively prevent 3-party leakage via payload.	Complex implementation. No guarantee that first parties will not be the ones passing on the search query.
Rewrite requests	Easy to implement.	Eventually we could miss some parameter or change the wrong thing. Ineffective against payload leakage.
Block requests	Effectively stops all kinds of third-party leakage.	Dependent on a list of domains in order to avoid breaking user experience.
Implement a default Referrer policy in the browser	Follows the Web standard. Easy to enforce.	Server sides can overwrite the policy value.
Fit all search components within isolated iframes	Leverages native browser protection mechanisms (Same origin policy) to protect against all kinds of leakage	Requires advanced knowledge and admin control access to the Website's internal structure.

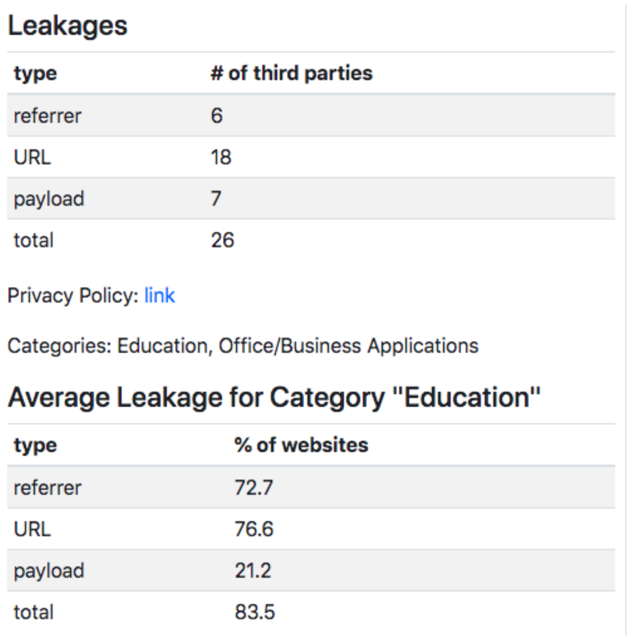
Table 10. Countermeasures to prevent search terms leakage.

Appendix 4: Browser Extension Screenshots



This site sends your search terms to 2 third parties

Fig. 6. JellyBeans browser plugin warning a user when they attempt to use internal search.



Leakages

type	# of third parties
referrer	6
URL	18
payload	7
total	26

Privacy Policy: link

Categories: Education, Office/Business Applications

Average Leakage for Category "Education"

type	% of websites
referrer	72.7
URL	76.6
payload	21.2
total	83.5

Fig. 7. Extension popup on a popular business tool website.

Appendix 5: Search Providers

Domain	Sites	Frequency
google.com	5063	0.01%
yandex.ru	692	0.0014%
searchanise.com	609	0.0012%
wikipedia.org	453	0.0009%
wiktionary.org	444	0.0009%
ksearchnet.com	284	0.0006%
doofinder.com	260	0.0005%
mybcapps.com	241	0.0005%
nextopiasoftware.com	237	0.0005%
searchspring.net	236	0.0004%
addsearch.com	223	0.0004%
cludo.com	207	0.0004%
searchspring.io	166	0.0003%
swiftype.com	156	0.0003%
resultspage.com	151	0.0003%
...	...	...
algolia.net	18	0.00003%

Table 11. Most frequent domains identified as search results providers after receiving query parts via specific URL query parameter