Canvassing the Fingerprinters: Characterizing Canvas Fingerprinting Use Across the Web

Elisa Luo
UC San Diego
La Jolla, CA, USA
e4luo@ucsd.edu

Tom Ritter
Mozilla
San Francisco, CA, USA
tom@mozilla.com

Stefan Savage
UC San Diego
La Jolla, CA, USA
ssavage@ucsd.edu

Geoffrey M. Voelker
UC San Diego
La Jolla, CA, USA
voelker@ucsd.edu

Abstract

Canvas fingerprinting is an effective technique for implicitly re-identifying visitors to a Web site based on subtle variations in the graphical rendering of specific "test canvases". Different fingerprinting actors make use of distinct canvases for this purpose and thus, as we show, it is possible to "fingerprint the fingerprinters" by grouping together identical canvases that are employed for these tests. In this paper, we document the prevalence of canvas fingerprinting (finding that 12.7% of the top 20K sites engage in it), use this grouping technique to measure and characterize the online footprint of widely-used fingerprinting services, and finally analyze the context in which these services are used to shine light on their intended purpose.

CCS Concepts

• **Information systems → World Wide Web**; • **Security and privacy → Privacy protections**.

Keywords

Canvas Fingerprinting; Browser Fingerprinting

ACM Reference Format:
Elisa Luo, Tom Ritter, Stefan Savage, and Geoffrey M. Voelker. 2025. Canvassing the Fingerprinters: Characterizing Canvas Fingerprinting Use Across the Web. In *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25), October 28–31, 2025, Madison, WI, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3730567.3764500

1 Introduction

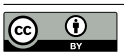
Web browser fingerprinting is a technique where a site measures inter-machine differences to probabilistically re-identify a user across visits. Dating back to at least the 2010's, researchers have identified a broad range of distinguishing characteristics that can be used together for effective fingerprinting, including both explicit features (e.g., HTTP headers) and implicit features (e.g., installed fonts, JavaScript and browser API implementation, and math functionality) [1, 14, 28]. However, among these, differences in object rendering via the HTML <canvas> element [35] provides the greatest discriminatory power [30] and consequently such *canvas fingerprinting* is almost universally incorporated into production fingerprinting schemes [1, 14, 28, 39].

Canvas fingerprinting relies on detailed, per-pixel differences in how carefully crafted "test canvases" are rendered. In principle, a single set of test canvases could be universally used. However, as we will show, in practice the considerable flexibility afforded by canvas rendering has led different fingerprinting services to innovate and create distinct test canvases for their purposes. As a result, the very *choice* of test canvas can be used as a "fingerprint" identifying the particular fingerprinting script or service being employed.

In this paper, we further develop this observation via three primary contributions:

- **Measuring canvas fingerprinting usage**. We crawl two groups of sites: a collection of "popular" sites (the top 20K as ranked by Tranco [31]) and a sample of 20K "tail" sites selected from the remainder of the top 1 million sites in the same ranking. We describe a technique for identifying potential "test canvases" by identifying Canvas API calls and filtering those whose content is explicitly extracted in JavaScript. Overall canvas fingerprinting is relatively modest, found in 12.7% of popular and 9.9% of tail sites.
- **Canvas clustering**. By grouping together identical canvases found across different sites, we document that a modest number (~500) of distinct canvases dominate the landscape and that these, in turn, correspond to particular fingerprinting services and scripts.
- **Describing fingerprinting context**. Finally, we analyze the context around the use of these different fingerprinting services in the wild and how they might relate to questions of user privacy. This includes any public representation of their operators, and how often such fingerprinting is employed in an advertising or tracking context (i.e., and thus is more likely to be used for cross-site re-identification purposes). We similarly analyze and document the range of evasion and cloaking techniques used to circumvent anti-fingerprinting defenses in some modern browsers.

Our findings underscore the complexity of modern browser fingerprinting in the wild, and provide a set of initial techniques for



IMC '25, Madison, WI, USA

ACM ISBN 979-8-4007-1860-1/2025/10
https://doi.org/10.1145/3730567.3764500

identifying and teasing apart how these capabilities are being used in practice.

2 Background and related work

The HTML <canvas> element is a powerful feature in modern Web development allowing for in-browser rendering support of dynamic and scriptable graphics within a Web page. Developers can use JavaScript to draw shapes, text, images and animations on the canvas, making it a popular tool for interactive applications, gaming, and data visualization.

However, the Canvas API can also be used for browser fingerprinting: first presented in 2012, canvas fingerprinting exploits subtle differences in the rendering of the same text or WebGL scene to track and identify users [35]. This mechanism does not require user consent, and can be done invisibly and quickly on Web pages.

These variations in canvas rendering arise from differences in the graphic cards, operating systems, display settings, browsers, and installed fonts on different machines (e.g., how anti-aliasing and sub-pixel smoothing is performed). To capture these differences, a fingerprinter can craft a "test canvas" that triggers such differing behaviors, such as rendering an emoji or using many distinct letters in a string (for example, the open-source fingerprinting library FingerprintJS uses the pangram *Cwm fjordbank gly*) [19]. Among all browser fingerprinting methods, canvas fingerprinting generates some of the highest entropy, proving its efficacy in uniquely identifying users [23, 30]. Indeed, several recent studies have also found the HTML canvas to be a highly popular fingerprinting surface among fingerprinters the wild [32, 39].

To this end, several anti-fingerprinting mechanisms have been developed to limited success. For example, browsers can fully mitigate canvas fingerprinting by blocking all Canvas API accesses. The Tor browser uses this approach, but at the cost of significantly degrading the browsing experience. Another method is to add random noise to the pixel data returned from Canvas API calls. Unless carefully implemented, though, attackers can circumvent the mitigation [37]. Finally, a frequently adopted defense blocks Web requests to domains serving scripts known to be performing canvas fingerprinting, such as Firefox's Disconnect list [13]. However, such blocklists rely on the assumption that fingerprinters do not try to obfuscate their origins (e.g., front their script as first-party). As the adoption of blocklists increases, though, fingerprinters are increasingly motivated to obfuscate their origins.

Large-scale studies of the prevalence of canvas fingerprinting on the Web show that it is being frequently adopted for both tracking and security purposes. In 2014, Acar et al. showed that 5.5% of the Alexa top 100K sites used canvas fingerprinting scripts on their homepage, with over 95% of the scripts belonging to a singular (now defunct) advertising company, addthis.com [1]. A 2016 study found that 1.6% of the top 1 million sites used canvas fingerprinting, with a much greater diversity of domains serving the script (with 98.2% of scripts being third-party). The authors noted that the ad industry seemed to be shifting away from canvas fingerprinting and found an increase in fraud detection companies (e.g., doubleverify.com) performing canvas fingerprinting [14]. Corroborating this shift to security use-cases, several more recent studies have identified security-motivated browser fingerprinting in the wild [3, 9, 38, 39]. Identifying fingerprinting *intent* is difficult, yet it remains a critical problem for answering questions of regulatory compliance [34].

While there have been several recent measurement studies on browser fingerprinting [28, 39], the most recent large-scale study to measure canvas fingerprinting in isolation is Englehardt et al.'s 2016 study. Our work presents an updated measurement of canvas fingerprinting on the Web, with measurement techniques that better capture the nuances in today's landscape of browser fingerprinting.

3 Methodology

To study the operation and operators of canvas fingerprinting on the Web today, we visited both popular and random sites, recorded their Canvas API-related activity, and identified generated canvases used for fingerprinting.

To capture behavior from popular, well-provisioned sites, we visited the top-level page of the top 20K sites on the May 2025 Tranco rankings [31]. To also capture behavior from a diverse set of sites, we visited the top-level pages of a random sample of 20K "tail" sites ranked between 20K+1 and 1M on the same Tranco list (the most popular random site had rank 20,025, and the least popular 997,854).

The Canvas API provides a mechanism to *extract* the result of a series of Canvas API calls (i.e., a generated canvas) as a Base64 encoded string using the `toDataURL` method of `HTMLCanvasElement`. In our analysis of canvas fingerprinting, we focused on detecting and recording the return value of canvas extractions precisely because it captures the result of all preceding Canvas API calls.

3.1 Crawler

To record Canvas API calls on a page, we modified DuckDuckGo's Tracker Radar Collector [22], a Puppeteer-based Web crawler. We added functionality that intercepted and recorded the arguments, return value, script source URL, and timestamp of API calls and property accesses to the interfaces `CanvasRenderingContext2D` and `HTMLCanvasElement`.

The crawler handles common anti-bot detection mechanisms, simulates basic user behavior by scrolling the page up and down and then waiting five seconds, and uses the autoconsent library [21] to opt-in to common consent banners. We performed the crawls in May 2025 using a UCSD IP address.

By definition, canvases used for fingerprinting depend on the hardware and software used to render to the canvas. As a result, we used a single Intel machine running Ubuntu 22.04.2 LTS to crawl the sites and record the canvases generated. To validate that identical canvases across sites are not spurious, we performed a second crawl of the same sites using a laptop with Apple's M1 chip. While the two machines rendered a site's canvases differently, the behavior across sites was the same: all the sites with identical canvases in the Intel crawl also had identical canvases in the M1 crawl.

3.2 Detecting Canvas Fingerprinting

While we recorded all canvas image extractions (e.g., `toDataURL` calls), not all generated canvases are used for fingerprinting. For example, Web-based image manipulation tools allow the user to extract the completed image as a data URL. However, such use

cases require explicit user action which our crawler does not perform. Other benign use cases can be related to webp and emoji compatibility checking [24].

Adapting the heuristics outlined in [14], we filtered extracted canvases according to three criteria. We excluded canvases extracted in a lossy compression format such as JPEG or webp; compression loses the subtle differences required to perform fingerprinting, and excluding webp excludes compatibility checks for it. We also excluded small canvases (smaller than 16×16 pixels); these have insufficient complexity to support effective canvas fingerprinting, and also conveniently excluded emoji compatibility tests. Lastly, we excluded canvases generated by scripts that also invoke methods associated with animation (`save`, `restore`, etc.).

After applying these filters, 83% of all extracted canvases from the popular and tail sites were *fingerprintable*. We manually inspected a random sample of 200 canvases excluded by the above heuristics and all had a benign use case. We provide examples of canvases we excluded (and the sites they appear on) in Appendix A.2. We also manually inspected a random sample of 300 unique fingerprintable canvases. We found only two false positives, and these canvases each appear only on a single domain (i.e., their existence does not affect the results of our grouping analysis). We use this set of *fingerprintable canvases* in our subsequent analyses.

Limitations. Our crawls only focused on the homepage of each domain and did not follow inner links, which may have different privacy properties [4, 8, 44]. As a result, our methodology will miss instances of fingerprinting on inner pages (e.g., login pages) and hence our prevalence results represent a lower bound on canvas fingerprinting. Furthermore, while our crawler can overcome some common bot detection mechanisms, it is possible that sites still recognize our requests as automated and modify their fingerprinting behavior. Finally, fingerprinting behavior on a page may be triggered from a specific user action (e.g., logging in), but replicating this behavior is difficult to automate at scale and we accept that we will miss some instances of canvas fingerprinting.

4 Canvas Fingerprinting Usage

Using the canvases extracted from crawling both popular and tail sites, we characterize the prevalence of canvas fingerprinting on the Web today. Based on the particular canvases used, we are able to identify the footprint specific fingerprint services have across the sites in our data sets and the inferred intent of those services.

4.1 Prevalence

Our results report a modest increase of browser fingerprinting over the past decade. Of the 16,276 popular sites crawled successfully, 2,067 (12.7%) of them extracted at least one fingerprintable canvas. Fingerprinting is slightly less prevalent among tail sites. Of the 17,260 tail sites that crawled successfully, 1,715 (9.9%) of them extracted at least one canvas. These results are in contrast to the 5.5% of the top 100K sites reported by Acar et al. in 2014 [1]. Furthermore, a 2016 study by Englehardt and Narayanan [14] found that only 1.6% of the top 1 million sites performed canvas fingerprinting, but the effect of including more less-popular sites is unclear. Our findings more closely match more recent studies: a 2021 study by Iqbal et al. [28] found 10.18% of sites perform general fingerprinting techniques.[1] Most recently in 2024, Senol et al. [39] found that 8.9% of homepages (from the CrUX top 100K [10]) performed general fingerprinting, while almost all (93.1%) of these sites also engaged in canvas fingerprinting.

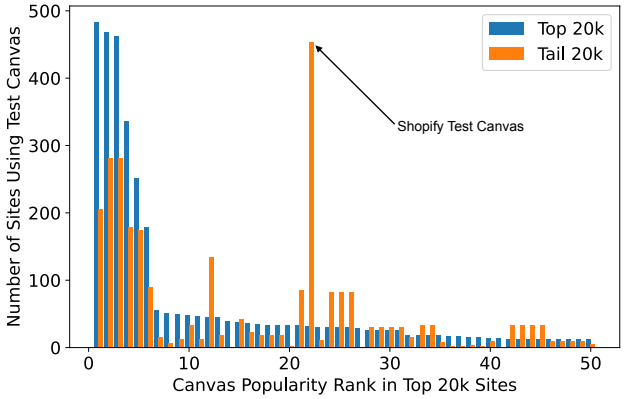


Figure 1: Number of sites using the top 50 most-frequently encountered test canvases across the top 20K sites. The orange bars shows the frequency of the test canvas appearing on a tail 20K site. The outlier orange bar represents Shopify's test canvas.

At a site granularity, our crawler found on average 3.31 fingerprintable canvases per site (with a max of 60 and median of 2). There are multiple reasons why a site may render multiple fingerprintable test canvases. For instance, increasing the number of test canvases could theoretically increase entropy by testing additional distinguishing characteristics of GPUs. Rendering multiple canvases can also detect browser evasion mechanisms (Section 5.3).

4.2 Reach

To capture the reach of particular fingerprinting services, we group together sites that generate identical test canvases. Since fingerprinting algorithms are deterministic and our crawler visited all of the sites using the same browser and machine, every site that uses a particular fingerprinting script will have exactly the same output from the `toDataURL` method of `HTMLCanvasElement`. In theory, a single set of test canvases could be universally used across all sites. However, in practice the diversity among canvas fingerprinting scripts is relatively high. Popular sites generated a total of 504 unique fingerprinting canvases, and tail sites generated 288.

Sharing of canvases across sites is a long-tailed distribution. Figure 1 shows the head of this distribution: for the 50 most common test canvases in the top 20K sites, it shows the number of popular sites (blue) and tail sites (orange) a particular canvas appears on. For example, the most frequently encountered canvas among popular sites appeared on 483 of them (3% of the popular sites crawled successfully).

The six most-frequent canvases account for much of the sharing on both popular and tail sites, accounting for 70.1% of the top 20K sites generating a test canvas and 47.1% of tail sites. There

[1]These techniques include but are not exclusively canvas fingerprinting. Concurrent studies have found that the vast majority of sites performing general fingerprinting also engage in canvas fingerprinting [32].

is, however, one noticeable outlier among the tail sites: the most frequently used test canvas among tail sites (used by 454 tail sites), only appears on 32 of the top 20K sites. Upon inspection, this canvas fingerprint is used for Shopify storefront "performance" monitoring [40]. This difference in prevalence can be explained by Shopify storefronts being far more common among less popular sites.

Overlap of test canvases between the tail and top sites. To a significant extent, tail sites and popular sites are using similar test canvases: 91.4% of tail sites that perform canvas fingerprinting generated a canvas that was also used by a popular site. Furthermore, there were no significant groups of tail sites that share a test canvas not seen in the popular sites: the largest group of tail-only test canvases appeared on 15 tail sites, and the next-largest appeared on only 3 sites.

Implication on cross-site tracking and fingerprinter reach across sites. In practice, not a single organization owns all scripts that perform canvas fingerprinting across all sites. Thus, the groups of sites that render the same set of test canvases provide an *upper bound* on the reach of any particular organization. The measured reach is an upper bound because our methodology assumes that different organizations will render unique test canvases (which is not always the case). However, to attribute users across sites, at a minimum the test canvas must be the same for those sites.

When compared to a 2016 study that found Google had trackers embedded on over 42% of all visited pages in Germany [45], the scope of cross-site tracking currently feasible through canvas fingerprinting is small: a reach of at most of 3% of the top 20K sites.

4.3 Attribution

We can use the diversity of test canvases being rendered to analyze the footprint specific fingerprint services have across the Web. Many fingerprinting services render a set of canvases distinct from other fingerprinters, making identification straightforward. There are also a few cases where a set of test canvases is used by several different entities (e.g., FingerprintJS), requiring more care.

Treating unique canvases generated by a specific fingerprinter as a fingerprint itself, we can use our sample of popular and tail sites to gauge their relative use. We gather the ground truth of the test canvases used by each fingerprinting service as follows: If a demo of the service was publicly available, we crawled the demo page and recorded the test canvases used. Otherwise, we crawled sites of customers of the fingerprinting service and recorded the test canvases used. We determined customers of commercial fingerprinting services through a combination of identifying customers on the service's Web site and pattern matching on the fingerprinting script source URL. We provide additional details in Appendix A.3.

We repeated the process for twelve prominent fingerprinting services that account for 73% of popular and 71% of tail sites that generate fingerprinting test canvases, as shown in Table 1. Note that some sites may use multiple fingerprinting services. We created an initial list of fingerprinting services by searching for commercial fingerprinting services advertised for security and marketing/attribution, and adding popular vendors identified by prior work (e.g., [28, 39]). To improve coverage of the remaining commonly-appearing test canvases, we also manually inspected script source code.

Service	Top 20K	Tail 20K
Akamai	485 (23%)	205 (12%)
FingerprintJS	462 (22%)	298 (17%)
mail.ru	242 (12%)	173 (10%)
FingerprintJS (legacy)	179 (9%)	90 (5%)
Imperva*	49 (2%)	13 (1%)
AWS Firewall	48 (2%)	14 (1%)
InsurAds	40 (2%)	1 (0%)
Signifyd	39 (2%)	18 (1%)
PerimeterX	35 (1%)	2 (0%)
Sift Science	31 (1%)	8 (0%)
Shopify	32 (2%)	457 (27%)
Adscore	25 (1%)	30 (2%)
GeeTest	1 (0%)	0 (0%)
Total Sites	1,513 (73%)	1,222 (71%)

Table 1: Number of sites linked to each fingerprinting vendor. Bold services are associated with security applications. *Imperva was identified based on script URL.

4.3.1 Highest Reach. These vendors have the potential to track users across the largest number of sites.

Akamai has the greatest reach of any one fingerprinting service. Its unique test canvas was generated on 485 (23%) popular sites and 205 (12%) tail sites that engage in canvas fingerprinting. As one of the largest hosting services, Akamai estimates it serves 15–30% of global Web traffic [2]. The test canvas we observe is associated with Akamai's bot detection service, giving Akamai potential user visibility across sites spanning a wide range of sites and markets (from shopping to news, banking, etc.).

FingerprintJS is a unique case as they control both a highly popular open-source browser fingerprinting library and offer a (paid) commercial fingerprinting service [18]. Both the open-source and commercial services render the same set of test canvases. However, we can distinguish between open-source and commercial customers by the URLs for the scripts and the script content.[2] In all, 462 (22%) popular sites and 298 (17%) tail sites that perform canvas fingerprinting use the FingerprintJS set of test canvases. Of these, only 23 top sites and 10 tail sites use the paid commercial service. While most of the use cases advertised for the commercial service are targeted towards improving security (e.g., account sharing prevention), FingerprintJS commercial has the potential to be mixed-use: its dashboard exposes the fingerprint identifier to the customer, and the customer is free to use that identifier as they wish. Furthermore, their site advertises providing "personalization to anonymous users" as a potential use case of their service [16].

Interestingly, many commercial advertisers and analytics companies utilize the open-source version of FingerprintJS to perform browser fingerprinting: among scripts rendering the FingerprintJS set of test canvases, we see ones belonging to advertising companies AIdata (40 top sites, 10 tail), adskeeper (10 top, 6 tail), trafficjunky (7 top, 1 tail) and MGID (23 top, 17 tail), and Russian analytics company acint.net (18 top, 29 tail). Further, an additional 179 top

[2]The commercial version uses additional fingerprint surfaces (e.g., the mathML library) that the open-source version does not use.

and 90 tail sites use a test canvas associated with an older (~2020) version of FingerprintJS. Updates to a single fingerprinting vendor's script can lead to changes in the way the test canvas is rendered, potentially breaking re-identification.

mail.ru has broad reach across .ru domains. In our sample of sites, the mail.ru test canvas set appeared on 242 popular sites and 173 tail sites. These sites encompass one-third of all .ru domains among the top 20k sites, showing significant potential for re-identifying users across .ru domains.

InsurAds provides ad analytics data to customers and uses browser fingerprinting to generate "real-time unique-user attention optimization" [27]. However, their reach is limited across sites.

4.3.2 Security Applications. We identified a diversity of vendors providing security applications performing canvas fingerprinting in our dataset. **Sift Science** and **Signfyd** provide fraud detection, **Adscore** focuses on ad fraud, and **PerimeterX** and the **AWS Application Firewall** are bot detection services. While we were able to identify an assortment of fingerprinting vendors for security applications, their prevalence on the Web is overshadowed by FingerprintJS and mail.ru, and they have a limited reach across sites.

Imperva is another special case. Although it also uses canvas fingerprinting for bot detection, it employs a unique test canvas for *each* site it is deployed on. As a result, Imperva fundamentally does not have the ability to track users across sites using canvas fingerprinting. While we were unable to group canvases to identify sites using Imperva's services, we instead were able to identify their customer sites by pattern matching the script URL that generates their test canvas (Appendix A.4).

5 Tracking, Advertising, and Evasion

Previously we identified fingerprinting services and categorized them based upon how they advertise their services. This approach highlights positive uses of fingerprinting, in particular commercial fingerprinting services that advertise various security uses (e.g., detecting bots, streamlining authentication, etc.). However, browser fingerprinting is well known for privacy-sensitive purposes, such as tracking and advertising; indeed, a recent study [33] found early evidence of its use to influence advertisement bids. In this section we use blocklists to characterize the use of canvas fingerprinting for tracking and advertising, and then explore how fingerprinters try to evade ad blockers and browser defenses.

5.1 Tracking and Advertising

To provide a first glance at fingerprinting scripts associated with known privacy-sensitive Web resources, we turn to popular crowd-sourced anti-advertising and anti-tracking blocklists. We check for a script's inclusion in three such lists: EasyList [11], EasyPrivacy [12], and Disconnect [13].[3] For EasyList and EasyPrivacy, we check if any of their rules apply to the URL of the script rendering a test canvas using adblockparser [29], and do not consider the dynamic context of the script (as dynamic rules are often in place in such lists to prevent breaking sites). We configure adblockparser to use the resource type `script` as we focus on analyzing fingerprinting scripts. The Disconnect list is domain-based, so we simply check if the domain of the script's URL is included in the list.

Our results suggest that many fingerprinting scripts are perceived to be privacy-sensitive (at least in the context of crowd-sourced blocklists). Nearly half (45%) of test canvases in the top 20k sites and over one-third (37%) in the tail 20k sites are generated by a script that has been included in one of the three lists. Indeed, 15% of canvases are generated by a script that matches all three blocklists, indicating clear tracking or advertising intent (Appendix A.4 has a full breakdown). However, as we show in the following section these numbers represent an *upper* bound on the number of privacy-sensitive fingerprinting scripts blocked in practice.

5.2 Evading Blocklists

While nearly half of canvas fingerprinting scripts have been added to a crowd-sourced blocklist, indicating that a significant amount of canvas fingerprinting is indeed coming from privacy-sensitive Web resources, statically-applied blocklist rules do not tell the complete story. These high inclusion rates, combined with the use of these blocklists by ad blockers with hundreds of millions of users, gives fingerprinters strong incentive to evade detection.

To explore the impact of such evasions, we revisited the top 20k and tail 20k sites with our crawler twice more, with two different popular ad blocker extensions installed: AdblockPlus [15] and UBlock Origin [26], both of which use EasyList's rules. Using this methodology we can determine which canvases are not rendered and extracted as a result of being blocked by one of these extensions. Revealingly, under both crawls with ad blockers, the number of test canvases generated, and the number of sites that generate at least one test canvas, only decreased by about 5% (Table 2).

There are several reasons for this significant difference in effectiveness in practice. First, due to the context in which the blocklist's rules are applied, ad blockers in practice miss many canvas fingerprinting scripts, whether from poor rule design[4] or from their precautions to avoid breaking site functionality.

Next, ad blockers commonly make exceptions for content served first-party. Despite perceptions that most tracking is done through third parties — and thus their Web resources are served third-party — 49% of the top 20k and 52% of the tail 20k sites engaging in canvas fingerprinting have at least one test canvas rendered by a script that is served first-party, falling under this exception.[5] The prevalence of first-party use suggests that fingerprinters are benefiting from, if not actively exploiting, such first-party exceptions.

Indeed, fingerprinting services actively encourage their customers to take advantage of such first-party exceptions to ensure their fingerprinting scripts are not blocked. By analyzing the documentation of the services we identified in Section 4.3, we discovered several methods commercial fingerprinters are actively using to mask their origins. The most popular in our data is **bundling** the fingerprinting library into the site's first-party JavaScript (common in single-page applications), a practice that circumvents all URL- or DNS-based fingerprinting detection techniques. More advanced

[3] EasyList targets advertising content, and EasyPrivacy and Disconnect target trackers.

[4] Our results are consistent with a 2020 study that found that up to 90% of EasyList's rules provide no benefit in real browsing scenarios [41]. See Appendix A.6 for an example of such a rule that fails in practice.

[5] For example, Akamai's script is missed by ad blockers due to this first-party exception, even though EasyList includes a rule matching Akamai's fingerprinting script's URL.

	# Test Canvases		# Sites	
	Top 20K	Tail 20K	Top 20K	Tail 20K
Control	6,037	4,422	2,067	1,715
Adblock Plus	5,834	4,228	1,948	1,656
UBlock Origin	5,776	4,175	1,976	1,651

Table 2: Using ad blockers only slightly reduces the number of test canvases generated and the number of sites that generate at least one test canvas.

techniques include **CNAME cloaking**, where fingerprinters use CNAME records to make their third-party scripts appear as if they are being served from the first-party domain [7]. Another option is **subdomain routing**, where services instruct their customers to use subdomains of their site to serve the fingerprinting scripts: 9.5% of the top 20K and 2.1% of the tail 20K sites performing canvas fingerprinting have at least one test canvas rendered by a script served by a subdomain of the site. Finally, fingerprinters often use **popular CDNs** to serve their scripts.[6] We found 2.1% of the top and 1.9% of the tail 20K sites performing canvas fingerprinting to have at least one test canvas served through a popular CDN, providing a lower bound on the number of sites using this technique (Appendix A.5 lists the CDNs we considered). Since CDNs are widely used for legitimate purposes, ad blockers avoid blocking them.

In summary, blocklist-based defenses are ineffective for a large fraction of fingerprinting scripts, and the domain that a fingerprinting script is served from is no longer a reliable method to identify fingerprinting vendors (as previously done in [1, 14, 28]). Both fingerprinters for security and tracking commonly mask their origins, making it increasingly difficult to disentangle fingerprinting intent.

5.3 Evading Browsers

Finally, in addition to using first-party exceptions to avoid ad blockers, fingerprinters also try to evade common browser fingerprinting mitigations. In particular, fingerprinters have developed relatively simple methods to detect and bypass canvas randomization techniques. Canvas randomization thwarts fingerprinting by injecting random noise into the canvas rendering process, and several browsers [36, 42] and privacy tools [46] use it.

In turn, fingerprinters have learned to detect canvas randomization using multiple renderings: fingerprinting scripts render the same canvas multiple times and compare the results. If the results are inconsistent, they infer the presence of canvas randomization[7] and adjust their fingerprinting strategy accordingly. Appendix A.7 provides pseudo-code for performing this check.

Across all the sites that perform canvas fingerprinting we visited, nearly half of them perform this canvas inconsistency check: 45% of them have at least one test canvas that was generated and extracted twice. Indeed, the most popular open-source fingerprinting library performs this inconsistency check, and disregards the canvas as part of the full browser fingerprint if it finds a discrepancy [20]. Commercial fingerprinters can also adapt their re-identification algorithms based on the knowledge that a user may be using such privacy-enhancing features [6], and usage of canvas randomization could also be a fingerprintable signal itself.

[6]For example, the commercial version of FingerprintJS offers their services routed through a Cloudflare worker [17].

[7]Note that this detection technique only works if *different* noise is added to each canvas rendering, rather than *persistent* noise added to all renderings across a given browsing session (e.g., as done by Firefox) [36].

6 Conclusion

Browser fingerprinting is a powerful technique because it is covert and involuntary: it is not part of any standard protocol and requires no explicit consent from the user. As a result, it may not be clear when such fingerprinting is taking place and thus, how prevalent the practice is.

Our work helps address this question by exploiting implementation artifacts in canvas fingerprinting, a widely used and highly discriminating browser fingerprinting technique. We have established that the precise choice of test canvas is a simple feature that can not only identify if a site is using canvas fingerprinting, but also which *particular* fingerprinting service they are employing. We found that canvas fingerprinting is moderately widespread: between 10–13% of sites employ this technique, dominated by a few major players. Furthermore, any changes in the exact series of Canvas API calls used to render a test canvas (e.g., due to updates to fingerprinting scripts, or changes to the Canvas API itself) can impact user re-identification over time.

However, fingerprinting can be used for a variety of purposes. At one extreme, it may be focused on user tracking (e.g., cross-site re-identification), while at the other, it may be used for anti-fraud (e.g., bot identification or same-site user re-identification on login pages). Distinguishing between these two can be challenging since intent and use are not readily apparent. However, the distinction is critical to our normative understanding of the practice as well as for questions of regulatory compliance (e.g., under the GDPR fingerprinting use associated with marketing/attribution purposes requires user consent, while security purposes do not [43]). We make progress on this question via two proxies: the public representation of companies (since we are able to tie canvases back to their services) and the presence of the domains serving such canvases on privacy blocklists (e.g., as used by ad blockers). We find that the use of fingerprinting services clearly associated with security represent the minority use case (and a *small* minority among less popular sites) and that between 37% and 45% of sites serving fingerprinting canvases are listed on privacy blocklists. Together, these suggest that there remain reasons to be concerned about the privacy implications of browser fingerprinting in practice.

7 Acknowledgements

We thank our anonymous reviewers and shepherd for their insightful suggestions and feedback. We are also grateful to Alisha Ukani and Tim Huang for providing feedback on our work, Seoyoung Kweon and Liam Arzola for their help on a precursor project to this paper, and Paul Chung for his insights on anti-advertising blocklists. Many thanks also to Cindy Moore and Jennifer Folkestad for operational and administrative support of our research. Funding for this work was provided in part by NSF grant CNS-2152644, the Irwin Mark and Joan Klein Jacobs Chair in Information and Computer Science, the CSE Professorship in Internet Privacy and/or Internet Data Security, a generous gift from Mozilla, and operational support from the UCSD Center for Networked Systems.

References

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. 2014. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS'14)*. Association for Computing Machinery, New York, NY, USA, 674–689.

[2] Akamai. 2018. Akamai Internet Observatory. https://community.akamai.com/customers/s/article/Akamai-Internet-Observatory?language=en_US#:~:text=Since%20Akamai%20is%20the%20global,on%20the%20Akamai%27s%20Internet%20Observatory!

[3] Babak Amin Azad, Oleksii Starov, Pierre Laperdrix, and Nick Nikiforakis. 2020. Web Runner 2049: Evaluating Third-Party Anti-bot Services. In *Proceedings of the 17th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (Lisbon, Portugal) *(DIMVA'20)*. Springer-Verlag, Berlin, Heidelberg, 135–159.

[4] Waqar Aqeel, Balakrishnan Chandrasekaran, Anja Feldmann, and Bruce M. Maggs. 2020. On Landing and Internal Web Pages: The Strange Case of Jekyll and Hyde in Web Performance Measurement. In *Proceedings of the 2020 ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC'20)*. Association for Computing Machinery, New York, NY, USA, 680–695.

[5] dieki (asker) and Pointy (commenter). 2011. Detecting WebP support. https://stackoverflow.com/questions/5573096/detecting-webp-support

[6] Martin Bajanik. 2022. Why Anti-Fingerprinting Techniques Don't Work in Browsers. https://fingerprint.com/blog/browser-anti-fingerprinting-techniques

[7] Ha Dao, Johan Mazel, and Kensuke Fukuda. 2021. CNAME Cloaking-Based Tracking on the Web: Characterization, Detection, and Protection. *IEEE Transactions on Network and Service Management* 18, 3 (2021), 3873–3888.

[8] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann. 2022. Reproducibility and Replicability of Web Measurement Studies. In *Proceedings of the 2022 ACM Web Conference* (Virtual Event, Lyon, France) *(WWW'22)*. Association for Computing Machinery, New York, NY, USA, 533–544.

[9] Antonin Durey, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. 2021. FP-Redemption: Studying Browser Fingerprinting Adoption for the Sake of Web Security. In *Proceedings of 18th Conference on the Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'21)*. Springer-Verlag, Berlin, Heidelberg, 237–257.

[10] Zakir Durumeric. 2025. crux-top-lists: Chrome (CrUX) Top Million Websites. https://github.com/zakird/crux-top-lists.

[11] EasyList. 2025. EasyList Filter Lists. https://easylist.to/.

[12] EasyList. 2025. EasyPrivacy Filter List. https://easylist.to/easylist/easyprivacy.txt.

[13] Steven Englehardt. 2020. Firefox 72 blocks third-party fingerprinting resources. https://blog.mozilla.org/security/2020/01/07/firefox-72-fingerprinting.

[14] Steven Englehardt and Arvind Narayanan. 2016. Online Tracking: A 1-million-site Measurement and Analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS'16)*. Association for Computing Machinery, New York, NY, USA, 1388–1401.

[15] eyeo GmbH. 2025. AdblockPlus: Surf the web with no annoying ads. https://adblockplus.org/.

[16] Fingerprint. 2023. How to Provide Personalization to Anonymous Users. https://fingerprint.com/blog/providing-personalization-to-anonymous-users

[17] Fingerprint. 2025. Cloudflare Proxy Integration. https://dev.fingerprint.com/docs/cloudflare-integration.

[18] Fingerprint. 2025. Fingerprint: Identify Every Visitor. https://fingerprint.com

[19] Fingerprint. 2025. FingerprintJS. https://github.com/fingerprintjs/fingerprintjs.

[20] FingerprintJS. 2025. canvas.ts. https://github.com/fingerprintjs/fingerprintjs/blob/9a6c283cdeaee8905d5f085aef180c6ebbbf0e04/src/sources/canvas.ts#L79

[21] Duck Duck Go. 2025. Autoconsent. https://github.com/duckduckgo/autoconsent.

[22] Duck Duck Go. 2025. DuckDuckGo Tracker Radar Collector. https://github.com/duckduckgo/tracker-radar-collector.

[23] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. 2018. Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *Proceedings of the 2018 ACM Web Conference* (Lyon, France) *(WWW'18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 309–318.

[24] Conner Gurney. 2017. How can I detect rendering support for emoji in JavaScript? — Stack Overflow. https://stackoverflow.com/questions/45576748/how-can-i-detect-rendering-support-for-emoji-in-javascript.

[25] Raymond Hill. 2024. uBlock Origin Static Filter Syntax. https://github.com/gorhill/uBlock/wiki/Static-filter-syntax

[26] Raymond Hill and Nik Rolls. 2025. uBlock Origin — Free, open-source ad content blocker. https://ublockorigin.com/.

[27] InsurAds. 2025. InsurAds: Revolutionize Digital Advertising. https://insurads.com/.

[28] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. 2021. Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy* (San Francisco, CA, USA) *(S&P'21)*. IEEE, New York, NY, USA, 1143–1161.

[29] Mikhail Korobov. 2016. adblockparser. https://pypi.org/project/adblockparser.

[30] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. 2016. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy (S&P'16)*. IEEE, New York, NY, USA, 878–894.

[31] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Network and Distributed System Security Symposium* (San Diego, CA, USA) *(NDSS'19)*. Internet Society, Fredericksburg, VA, USA, 15 pages.

[32] Xu Lin, Panagiotis Ilia, Saumya Solanki, and Jason Polakis. 2022. Phish in Sheep's Clothing: Exploring the Authentication Pitfalls of Browser Fingerprinting. In *Proceedings of the 31st USENIX Security Symposium* (Boston, MA) *(USENIX Security'22)*. USENIX Association, Berkeley, CA, USA, 1651–1668.

[33] Zengrui Liu, Jimmy Dani, Yinzhi Cao, Shujiang Wu, and Nitesh Saxena. 2025. The First Early Evidence of the Use of Browser Fingerprinting for Online Tracking. In *Proceedings of the 2025 ACM Web Conference* (Sydney, NSW, Australia) *(WWW'25)*. Association for Computing Machinery, New York, NY, USA, 4980–4995.

[34] J. Martínez Llamas, K. Vranckaert, D. Preuveneers, and W. Joosen. 2025. Balancing Security and Privacy: Web Bot Detection, Privacy Challenges, and Regulatory Compliance under the GDPR and AI Act. *Open Research Europe* 5 (2025), 76.

[35] Keaton Mowery and Hovav Shacham. 2012. Pixel Perfect: Fingerprinting Canvas in HTML5. In *Proceedings of the Sixth Web 2.0 Security & Privacy Workshop* (San Francisco, CA, USA) *(W2SP'12)*. IEEE, New York, NY, USA, 12 pages.

[36] Mozilla. 2025. Firefox's protection against fingerprinting. https://support.mozilla.org/en-US/kb/firefox-protection-against-fingerprinting.

[37] Hoang Dai Nguyen and Phani Vadrevu. 2025. Breaking the Shield: Analyzing and Attacking Canvas Fingerprinting Defenses in the Wild. In *Proceedings of the 2025 ACM Web Conference* (Sydney, NSW, Australia) *(WWW'25)*. Association for Computing Machinery, New York, NY, USA, 4336–4345.

[38] Phillipe Raschke and Axel Küpper. 2018. Uncovering Canvas Fingerprinting in Real-Time and Analyzing Its Usage for Web Tracking. In *Proceedings of the 2018 INFORMATIK Conference* (Berlin, Germany). Gesellschaft für Informatik e.V., Bonn, Germany, 12 pages.

[39] Asuman Senol, Alisha Ukani, Dylan Cutler, and Igor Bilogrevic. 2024. The Double Edged Sword: Identifying Authentication Pages and their Fingerprinting Behavior. In *Proceedings of the 2024 ACM Web Conference* (Singapore, Singapore) *(WWW'24)*. Association for Computing Machinery, New York, NY, USA, 1690–1701.

[40] Shopify. 2024. About performance optimization. https://shopify.dev/docs/apps/build/performance.

[41] Peter Snyder, Antoine Vastel, and Ben Livshits. 2020. Who Filters the Filters: Understanding the Growth, Usefulness and Efficiency of Crowdsourced Ad Blocking. In *Proceedings of the 2020 ACM Conference on Measurement and Analysis of Computing Systems (SIGMETRICS'20)*. Association for Computing Machinery, New York, NY, USA, 24 pages.

[42] Brave Software. 2025. Fingerprinting Protections. https://github.com/brave/brave-browser/wiki/Fingerprinting-Protections.

[43] European Union. 2018. Art. 21 GDPR — Right to object. https://gdpr-info.eu/art-21-gdpr/.

[44] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2020. Beyond the Front Page: Measuring Third Party Dynamics in the Field. In *Proceedings of the 2020 ACM Web Conference* (Taipei, Taiwan) *(WWW'20)*. Association for Computing Machinery, New York, NY, USA, 1275–1286.

[45] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M. Pujol. 2016. Tracking the Trackers. In *Proceedings of the 2016 ACM Web Conference* (Montréal, Québec, Canada) *(WWW'16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 121–132.

[46] Yubi. 2025. Canvas Fingerprint Defender. https://chromewebstore.google.com/detail/canvas-fingerprint-defend/lanfdkkpgfjfdikkncbnojekcppdebfp.

A Appendix

A.1 Ethics

We believe our work has very low ethical risk. Our study draws from data that is publicly available on the Web, and does not contain any personal information.

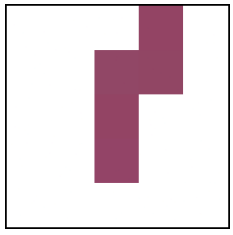
A.2 Excluded Canvases

As discussed in Section 3.2, not all canvases a site extracts using `toDataURL` are test canvases used for fingerprinting. Here we provide more details about the canvases that our heuristics exclude as fingerprintable test canvases, including examples of excluded canvases and the sites that use them.

WebP Compatibility Testing. Many sites (306 in the top 20K) extract a 1x1 pixel or 300x150px (the `HTMLcanvasElement`'s default size) transparent WebP image using `toDataURL`. This use of `toDataURL` is likely to test dynamically whether the user's browser has WebP support [5]. We exclude such canvases in our analysis as they are not used for fingerprinting. Some examples of sites that we found to perform WebP compatibility testing through the HTML canvas are `dailynews.com`, `smule.com`, `tinder.com`, and `nj.gov`.

Small Canvases. After excluding non-png image types, we also excluded small canvases under 16x16 pixels. An additional 216 of the top 20K sites included at least one such small canvas. Figures 2(a) and 2(b) show (zoomed in) examples of such small canvases excluded from our analysis. Typically, they are uniform in color. While it is unclear what the exact purpose of extracting such canvases are, they likely do not contain sufficient complexity to perform canvas fingerprinting.

Overall, there were 155 of the top 20K sites and 138 of the tail 20K sites that were fully excluded from our analysis (i.e., they did not render a fingerprintable canvas in addition to an excluded canvas).



(a) `lacounty.gov` (12x12 pix)

(b) `betus.com.pa` (5x5 pix)

Figure 2: Example small canvases excluded from our analysis: a 12x12 pixel canvas extracted by `lacounty.gov` and a 5x5 pixel canvas extracted by `betus.com.pa`.

A.3 Identifying Fingerprinting Vendors

As discussed in Section 4.3, we used two methods for attributing test canvases to fingerprinting services. Table 3 shows how we identified the test canvases a particular fingerprinting vendor uses across its customer sites, in order of precedence (e.g., a demo is the most reliable indicator).

Demo indicates the service provided a free or public demo of their fingerprinting product. We crawled the demo site to extract the test canvases used by such a service.

Service	Demo	Customer	Script Pattern
Akamai		✓	/akam/
FingerprintJS	✓	✓	fpnpmcdn.net
mail.ru			privacy-cs.mail.ru
FingerprintJS (legacy)	✓		fpnpmcdn.net
Imperva*			*(See caption)*
AWS Firewall			awswaf.com
InsurAds		✓	insurads.com
Signifyd		✓	signifyd.com
PerimeterX		✓	px-cloud.net
Sift Science		✓	sift.com
Shopify		✓	shopifycloud
Adscore	✓		adsco.re
GeeTest		✓	geetest.com

Table 3: How we attributed the test canvases fingerprinting vendors use. For Imperva, we used the following regular expression: `https?://(?:www\.)?[^/]+/([A-Za-z\-]+)`

Known Customer indicates that we crawled sites that are known customers (e.g., advertised on the service's Web site) of a particular service. For these sites, we always confirmed our crawls with the *Script Pattern* heuristic (e.g., in case the service offers several products, and only one of which use canvas fingerprinting).

Finally, we provide a **Script Pattern** associated with each fingerprinting provider. To identify services through the script pattern, we manually inspected the URLs of scripts rendering an identical canvas. Then, to attribute the script pattern to a particular vendor, we looked for copyright statements within the scripts and/or searched the potential vendors' documentation for the specific script pattern. We note that in Table 3, we only provide one example of a pattern found in a particular vendor's fingerprinting script, and not all sites using the vendor's script will have a script with such a pattern (hence the need for improving coverage through grouping together sites using the generated canvas itself).

Blocklist	Top 20K	Tail 20K
EasyList	1,869 (31%)	1,179 (27%)
EasyPrivacy	2,157 (36%)	1,340 (30%)
Disconnect	1,251 (21%)	833 (19%)
Any	2,696 (45%)	1,635 (37%)
All	942 (16%)	670 (15%)

Table 4: Number of test canvases generated by scripts included in popular crowdsourced anti-advertising and tracking blocklists.

A.4 Scripts Covered Under Crowdsourced Blocklists

Table 4 shows how many test canvases were generated by scripts that were included in EasyList, EasyPrivacy, and the Disconnect Tracker Protection List amongst the top 20K and tail 20K sites. A significant number of scripts are included in all three crowdsourced blocklists, indicating strong tracking and/or advertising intent.

A.5 Popular CDN List

Section 5.2 noted that fingerprinting services can use popular CDNs to serve fingerprinting scripts to bypass blocklists. We used the following domains to identify scripts loaded from popular CDNs:

`akamai.net`	`cloudflare.com`
`azureedge.net`	`cloudfront.net`
`b-cdn.net`	`fastly.net`
`bootstrapcdn.com`	`gstatic.com`
`cdn.jsdelivr.net`	`googleusercontent.com`
`cdnjs.cloudflare.com`	`googleapis.com`

A.6 Easylist Rule Design

As discussed in Section 5.2, in practice ad blockers are much less effective at blocking fingerprinting scripts due to the context in which the rules are applied. For example, advertiser mgid.com's canvas fingerprinting script is not blocked in practice even though EasyList has a rule for mgid.com:

```
||mgid.com^$document
```

The latter part of the rule ("`$document`") is a modifier that specifies the rule should apply to documents (i.e., the HTML site content) and not to other site resources like scripts [25]. In all, at the time of our analysis EasyList had 828 rules that use the same modifier.

A.7 Canvas Randomization Detection

The following pseudo-code shows how a fingerprinter might detect and circumvent the addition of random noise to canvas renderings (e.g., as done by [46]).

Algorithm 1: Canvas Randomization Detection

```
Render one test canvas, store as Canvas1;
Generate the same canvas again, store as Canvas2;
if Canvas1 ≠ Canvas2 then
    // The browser adds random noise – canvas
       fingerprint is unstable
    Disregard canvas component of the browser fingerprint;
else
    // The canvas is stable, proceed with
       fingerprinting
    Continue with fingerprinting logic;
```