

Large-Scale Measurement and Real-World Mitigation of Web Browser Fingerprinting in the Wild

Tom Ritter
Mozilla Corporation
San Francisco, USA
tritter@mozilla.com

Fatih Kilic
University of Toronto
Toronto, Canada
f.kilic@mail.utoronto.ca

Frederik Braun
Mozilla Corporation
San Francisco, USA
fbraun@mozilla.com

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

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

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

Christoph Kerschbaumer
Mozilla Corporation
San Francisco, USA
ckerschbaumer@mozilla.com

Abstract

Web browser fingerprinting has become a key tracking mechanism on the modern web, yet its real-world effectiveness and which of the existing mitigations offer value at scale remain insufficiently understood.

In this paper, we present a large-scale study of browser fingerprinting based on real-world telemetry from almost 25 million Firefox release users. This dataset enables us to quantify user uniqueness at population scale and identify which characteristics contribute most to this uniqueness.

Our analysis shows fingerprintability is largely driven by a small set of high-impact characteristics and their combinations, rather than accumulating many independent measurements. Building on these insights, we propose and evaluate fingerprinting protections based on their impact on uniqueness and cohort structure, and design data-driven mitigations.

These protections, shipped in Firefox 143, reduce the fraction of uniquely identifiable users from 75% to 46% while maintaining web compatibility, demonstrating that practical reductions in fingerprintability are achievable in a mainstream browser.

CCS Concepts

• Security and privacy → Browser security.

Keywords

Browser Fingerprinting, Large-Scale Measurement, Tracking Mitigation, Industry Deployment

ACM Reference Format:

Tom Ritter, Fatih Kilic, Frederik Braun, Elisa Luo, Geoffrey M. Voelker, Stefan Savage, and Christoph Kerschbaumer. 2026. Large-Scale Measurement

and Real-World Mitigation of Web Browser Fingerprinting in the Wild. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26)*, November 15–19, 2026, The Hague, Netherlands. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3830454.3846585>

1 Motivation

In contrast to traditional stateful cookie-based tracking [1, 2, 9, 15, 17, 19, 20, 22, 23, 26, 39, 50, 51, 80, 99], fingerprinting is a stateless technique that exploits web APIs to collect browser settings and hardware characteristics, e.g. installed fonts, preferred languages, timezone, CPU information, and construct an often-unique browser identifier, allowing adversaries to track users across sites. To make matters worse, modern fingerprinting now approaches the reliability of cookie-based tracking [47].

In response, browsers have begun deploying fingerprinting defenses, ranging from blocklists of known trackers to standardizing responses to hide underlying system differences to injecting noise into Web APIs such as Web Audio [6] and Canvas [13].

However, these approaches remain piecemeal: blocklists and targeted disruptions are susceptible to bypasses and require continuous maintenance, while API-level mitigations and standardization efforts are often developed in isolation for individual fingerprinting vectors. Both stem from the same gap: no systematic evidence of which characteristics contribute most to fingerprintability in real-world populations. Browser vendors therefore cannot assess whether deployed defenses meaningfully reduce user uniqueness or merely address low-impact signals.

Additionally, vendors face another important trade-off: reducing fingerprintability often costs usability and web compatibility [38]. Naive solutions, like gating powerful features behind prompts causes prompt fatigue, undermining their security benefits [94], while hiding system information can break legitimate functionality, such as serving downloads for the wrong operating system.

We therefore analyze fingerprint data from real-world users, simulate commercial fingerprinting services, and develop data-driven protections.



This work is licensed under a Creative Commons Attribution 4.0 International License. *CCS '26, The Hague, Netherlands*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2871-6/2026/11

<https://doi.org/10.1145/3830454.3846585>

Unlike prior studies of web user fingerprints, which rely on automated crawlers, volunteer samples, or users visiting a limited set of websites [4, 21, 33, 42], our approach captures real-world data at scale.

To the best of our knowledge, we present the first large-scale analysis of web fingerprinting using real-world telemetry from nearly 25 million Firefox users. While this dataset avoids those biases, its findings reflect Firefox’s user population and may not fully generalize to other browser ecosystems. Nevertheless, it enables the characterization of fingerprinting in the wild and identifies which APIs provide the most value to fingerprinting adversaries.

In detail, we contribute the following:

1) Large-Scale Measurement: We measure real-world, large-scale data from nearly 25 million Firefox release users about end users’ hardware and software settings, covering characteristics that are exposed to the web, thereby making users fingerprintable. We analyze the gathered dataset to measure the entropy of characteristics and how they interact when combined into a fingerprint. Based on the reverse engineering of popular fingerprinting libraries, we calculate the number of users grouped into different-sized cohorts of shared fingerprints and demonstrate how many users are unique, as well as *why* those users are unique (Section 3).

2) Real-World Mitigation: Leveraging the insights of (1), we design and implement effective fingerprinting protection techniques which have since been deployed in Firefox (version 143). Evaluated against simulated commercial fingerprinting services, our techniques reduce user uniqueness from 75% to 46% while preserving web compatibility (Section 4).

3) Design Considerations: We present design considerations for deploying fingerprinting defenses, including entropy reduction, standardization, and randomization. We discuss how these techniques complement each other and their role in practical, deployable defenses (Section 5).

These results demonstrate that fingerprinting is not an unavoidable property of the web, but a tractable problem addressable through data-driven, deployable defenses.

2 Background

In this section, we establish fundamentals of web tracking, outline our threat model, discuss limitations of our approach, and describe the three JavaScript services we simulate to evaluate our protection techniques.

2.1 Tracking Fundamentals

All web browsers adhere to the guidelines of the same-origin policy [104] which, among other goals, protects users against the possibility of extensive tracking on the web.

Browser fingerprinting is the capability of a site to observe software and hardware characteristics of an end user’s browser and re-identify it across contexts, e.g., over time or across origins. It leverages measurements such as installed fonts, preferred languages, the results of mathematical operations, timezones, and CPU information to construct an identifier of a user’s browser. While

fingerprinting can serve legitimate purposes such as fraud detection and bot mitigation, the fundamental problem remains the same: it enables invasive tracking, making users prone to cross-site tracking attacks [4, 7, 32, 33, 38, 41, 42, 44, 46, 47, 83, 87, 93, 97, 98, 105].

2.2 Balancing Privacy and Usability

Most characteristics revealed about users arise from web standards or unavoidable implementation differences, such as graphics hardware. Reducing this information often requires deviating from standardized web APIs, potentially breaking web functionality—referred to as *web compatibility*.

Optional Stricter Controls: The most comprehensive, but also most disruptive, fingerprinting defenses are implemented in Tor Browser [95] and the closely related Mullvad Browser [79]. Many have been upstreamed into Firefox [76], whose *Resist Fingerprinting (RFP)* mode [71] reproduces most of these defenses behind hidden configuration flags. **These protections can cause substantial web compatibility issues**, including broken calendaring, graphics functionality, and games; general user confusion and, in the period before our first Phase 1 protection (cf. Table 2) shipped in Firefox 115, 424 cataloged breakage reports against Firefox itself [60]. Since users tend to disable protections rather than tolerate such disruptions [69], this approach is unsuitable for a mainstream browser. However, these reports provide a valuable catalog of real-world compatibility constraints that informs the design of practical, deployable defenses.

Prompting the User: Browser vendors generally avoid prompting users for security decisions, as it often reduces rather than improves security [10]. Prompts rely on users making informed, consistent choices. However, users are unlikely to make correct decisions. Users also experience prompt fatigue: frequent requests lead to habitual approval, reducing prompts to noise rather than security checkpoints.

2.3 Threat Model

We assume the adversary can (1) execute JavaScript within web pages visited by the user, and (2) invoke browser-exposed APIs to collect information about the user’s environment. We do not assume that the adversary can exploit browser vulnerabilities or bypass existing security boundaries (e.g., the same-origin policy or iframe sandbox restrictions). Instead, **our focus is on limiting the abuse of legitimate JavaScript APIs for fingerprinting purposes**.

2.4 Scope and Limitations

The following limitations fall outside our threat model or, in the case of web compatibility, limit our available actions.

Web Compatibility Constraints: Fingerprinting defenses must preserve the user experience and avoid web compatibility issues. Prior work shows that more than half of users do not report breakage they encounter [84], so reported incidents represent only a fraction of actual breakage. Firefox’s deployment process therefore places strict constraints on candidate mitigations. Before code ships,

design review assesses each proposal using historical web compatibility reports, the RFP breakage corpus, and domain expertise on real-world API usage. Surviving proposals are deployed as *staged experiments* through Mozilla’s Experimenter platform [59], which randomly enrolls a fraction of the release population and evaluates pre-declared metrics for statistical significance. We monitor *usage metrics* for abandonment or decline and *user feedback* submitted through Bugzilla and Firefox’s feedback form. For mitigations that introduce observable differences, acceptability depends on impact severity, site popularity, available workarounds, and the likelihood of ecosystem adaptation. **Web compatibility is thus a hard constraint: strategies likely to cause unacceptable breakage are eliminated early** in the development lifecycle.

IP-Based Tracking: Fingerprinters can execute client-side JavaScript and send data to tracker-controlled servers [49], revealing the user’s IP address as an additional tracking vector. Mitigating IP-based tracking falls outside the scope of our threat model and is not addressed by the protections presented in this paper.

Website Population: Our results reflect the global user population, not the population of any individual web site. While we can identify users as unique or part of a cohort in our dataset, this may not hold for a specific site’s audience. For example, a Mandarin-localized user may be common globally, but unique on a local Icelandic website.

Bots: Autonomous programs (bots) may interact with websites and submit excessive telemetry data, leading to overcounting of certain characteristics. We assume that default telemetry settings in automated tools, together with safeguards in data collection pipelines (e.g., rate limiting and filtering), sufficiently limit this effect.

Excluding Users from Data Collection: Our study relies on Firefox telemetry [74] (cf. Section 3.1), which collects data from real-world users with minimal selection bias and includes an opt-out mechanism; we estimate that about 7% of users disable it. While this opt-out may introduce some bias, we expect its impact to be limited, as the dataset is neither self-selecting nor strongly regionally skewed.

Web Extensions: Fingerprinting libraries can detect content blockers [28, 30], introducing entropy. Roughly 40% of users install extensions [61], making them a common aspect of real-world browser configurations [40, 92]. However, incorporating extensions poses practical and ethical challenges: accurate characterization would require collecting fine-grained, highly dynamic data (e.g., installed extensions, versions, and configurations), which is difficult to measure at scale. As extensions are not uniformly available across browsers and are outside browser vendors’ control, we treat them as out of scope and part of the ambient fingerprinting surface.

2.5 Simulated Fingerprinting Services

For our evaluation, we reverse engineer and simulate three leading commercial services that allow us to assess how fingerprintable end users are. We selected these services based on their prevalence in the most recent measurement study [48] and their coverage in prior research [24, 81, 85, 90].

Fingerprint Pro: *Fingerprint Pro* [25] is a server-side, commercial browser fingerprinting service. *Fingerprint Pro* uses more sophisticated tracking techniques than its source-available counterpart, including additional fingerprinting surfaces such as MathML and emoji [49].

Akamai: Akamai has a broad reach over websites [48] and performs browser fingerprinting as part of its Bot Manager product [3]. While Akamai’s Bot Manager uses fingerprinting techniques such as canvas fingerprinting to reduce web scraping and other malicious Web automation, we found that its privacy statement does not expressly disallow the repurposing of collected user data for other purposes (i.e., tracking).

Signifyd: *Signifyd* [91] is a prominent [48, 90] fraud prevention service for e-commerce that distinguishes legitimate shoppers from malicious actors. Notably, analysis of its script suggests it is largely derived from *CreepJS* [18], another popular source-available fingerprinting library.

Fingerprinter Simulation: For each commercial service, we reconstruct the client-side measurements collected by its fingerprinting script using static and dynamic analysis. Because the server-side matching logic is proprietary, we do not replicate the exact identification algorithm. Instead, we model the *information available* to each service by combining all measurements into a canonical fingerprint.

Specifically, we compute a deterministic hash over these measurements, representing an *idealized matching model* that uses all signals consistently and without noise. This abstraction provides an upper bound on fingerprinter distinguishability and enables consistent comparisons across services and protection techniques.

This captures each service’s *capability* to track users rather than its behavior, since its internal use of collected data cannot be externally audited. Although fingerprinting is often justified by fraud and abuse prevention, leaving it unhindered enables tracking and privacy violations.

We exclude the browser’s outer and inner dimensions because benign actions such as window resizing make them unsuitable as persistent identifiers. We confirm this empirically by probing *Fingerprint Pro* following Luo et al. [49]: perturbing these values does not affect the returned visitor ID.

Using three established, widely deployed fingerprinting services enables a real-world evaluation of our analysis and provides empirical insights into the effectiveness of our proposed protections.

3 Large-Scale Measurement

In this section, we present insights from a large-scale analysis of web fingerprinting based on **data collected from nearly 25 million Firefox release users** on desktop and mobile (Windows 12.1m, macOS 2.0m, Android 8.9m, and Linux 1.8m). This measurement study collects telemetry that enables analysis from a highly privileged vantage point with visibility into underlying hardware characteristics. In Sections 3.1 and 3.2 we outline the method and reasoning behind our data collection procedure. In Sections 3.3 and 3.4, we find that the *method* by which user uniqueness is computed can significantly affect the results of such measurement studies,

and report the uniqueness of user fingerprints at a large population scale. Finally, in Sections 3.5 and 3.6 we find that no individual measurement contributes highly to user uniqueness, but rather, the identifying power is concentrated in a small subset of measurements.

3.1 Background on Data-Gathering

The initial data collection phase was conducted in June 2025. Based on this data, we developed fingerprinting protection techniques (cf. Section 4, Phase 2) which we began deploying in Firefox 143. To reflect the current ecosystem, re-evaluate the impact of these mitigations, and assist us in developing additional mitigations, we performed a second, updated data collection for this study. Our results are based on this telemetry [74] gathered during a Firefox release cycle, from **March 25th, 2026 to April 16th, 2026**.¹

As per Mozilla’s policy, all collected **data adheres to Mozilla’s Data Privacy Principles** [66], which enforce strict privacy and ethical standards (cf. Appendix A).

3.2 Collected Characteristics and Measurements

With few exceptions (Section 2.4), we collected a comprehensive set of features known or potentially useful for browser fingerprinting. This includes features used by source-available fingerprinting libraries [18, 25, 27], prior academic work [4, 7, 16, 32, 33, 42, 43, 46, 47, 83, 85, 87, 93, 105], and all browser APIs used by the three services modeled in our evaluation: *Akamai*, *Fingerprint Pro*, and *Signifyd* (Section 2.5).

In total, we collected **133 user characteristics comprising 708 distinct measurements** per user.

We define a **characteristic** as a *conceptual signal* (e.g., default system fonts) and a **measurement** as a *concrete observation* (e.g., default serif or monospace font). Importantly, our dataset covers a **superset of characteristics actively used by real-world fingerprinting deployments**.

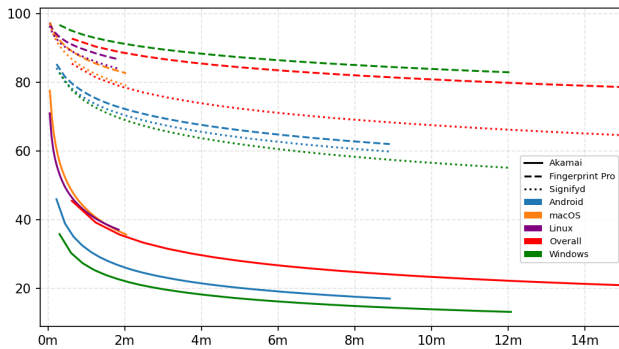


Figure 1: Effect of dataset size on fingerprint uniqueness. Y-axis shows percent of users classified as unique. Color indicates operating system, stroke style represents the evaluated fingerprinters.

¹Because data collection spanned Daylight Saving Time (DST) transitions in several regions, results were normalized to a consistent DST offset to prevent otherwise identical clients from differing due to submission date.

3.3 Impact of Dataset Size and OS Segmentation

We first assess the impact of **dataset size** and **Operating System (OS) Segmentation** on user fingerprint uniqueness, demonstrating the importance of population size and composition.

Dataset size: For each of the three commercial fingerprinting services, we compute a per-user fingerprint from the measurements it collects. Then, we compute the number of users classified as unique by each commercial fingerprinting service over increasingly large *random subsets* of our dataset.

As shown on the x-axis in Figure 1, which represents the total number of users, the likelihood of a user appearing unique declines steeply as population size increases, across all fingerprinting services and operating systems. For example, uniqueness estimates fall sharply between 0 and 3 million users, with curves stabilizing only at larger scales, typically around 4 million users or more.

OS Segmentation: One of the characteristics revealed by the browser is the user’s operating system, which browsers must report accurately for web compatibility. Misreporting the OS can cause significant issues, e.g., downloads for the wrong platform and broken website features such as non-functional shortcut keys [57]. The OS thus serves as a natural top-level differentiator for analyzing user populations. Partitioning users accordingly reveals substantial differences in both *how unique users are* and *why*.

As shown by lines with matching styles in Figure 1, the uniqueness percentages for each commercial fingerprinter can vary drastically for users of different operating systems. Specifically, for every fingerprinter there is at least a 23 percentage point difference across OSes. Only in three of the twelve scenarios does an individual OS’s uniqueness closely match the overall result: macOS and Linux for *Akamai*, and macOS for *Signifyd*.

Key Insights: Our analysis shows that first, **dataset size fundamentally affects uniqueness estimates**: convergence for high-dimensional measurements requires datasets on the order of *millions* of samples, far exceeding prior work (cf. Section 6). Second, **population segmentation reveals substantial differences in uniqueness across user groups**. Thus, **aggregating heterogeneous populations without accounting for these differences can distort uniqueness estimates and lead to systematically misleading conclusions**. These findings have direct implications for future measurement studies: results obtained from a single platform or homogeneous population may not generalize, and studies should account for population composition when interpreting and reporting uniqueness.

3.4 Fingerprinting Uniqueness and Cohorts

Now, we evaluate the uniqueness and cohort structure of the user fingerprints in our dataset through the lens of real-world fingerprinting services. Similar to the previous section, we model the behavior of *Akamai*, *Fingerprint Pro*, and *Signifyd* by combining the measurements that each fingerprinter collects, and then grouping users sharing the same fingerprint into a *cohort*.

To the best of our knowledge, prior work [4, 37, 42] only reports the percentage of users that can be uniquely identified through their browser fingerprint (i.e., the percentage of users that are in

Service	Windows (12.1m)		macOS (2.0m)		Android (8.9m)		Linux (1.8m)		Weighted Average
	% Uniq	Sim ($\times 10^{-7}$)	%	Sim	%	Sim	%	Sim	
<i>Fingerprint Pro</i>	82.9%	3.16	82.6%	11.94	62.0%	46.24	86.7%	76.05	75.7%
<i>Signifyd</i>	55.1%	31.80	79.1%	13.40	59.8%	51.12	83.9%	80.70	60.9%
<i>Akamai</i>	13.1%	1653.90	35.7%	135.48	17.0%	581.77	37.1%	478.27	18.2%

Table 1: Baseline user uniqueness and cohort score across three real-world fingerprinting services, segmented by OS.

a cohort of size 1). However, this does not tell the full story – two datasets may result in similar numbers of unique users while exhibiting substantially different cohort structures, leading to distinct privacy implications. As an extreme example, in one case all the non-unique users could share the same fingerprint, while in another the remaining non-unique users are all in cohorts of 2, meaning they share a fingerprint with only 1 other user.

Thus, we provide additional insight into the cohort structure of our dataset by reporting and comparing the overall cohort structure (i.e., the distribution of non-unique users across cohorts of varying sizes). To do this, we use the *Simpson Index*, a diversity metric that captures the concentration of users across cohorts and provides a single summary measure of the distribution. Intuitively, the *Simpson Index* measures the probability that two randomly selected users share the same fingerprint: a higher value indicates that users are concentrated into fewer, larger cohorts. As before, we partition all analysis by OS.

Baseline Uniqueness Across Fingerprinters: Table 1 summarizes the baseline uniqueness results for the three fingerprinting services, before any privacy features are applied. We observe substantial variation across services and operating systems, with *Fingerprint Pro* consistently yielding the highest proportion of unique users.

Specifically, it uniquely identifies a population-weighted average of 75.7% of users, with rates of 82.9% on Windows, 82.6% on macOS, 62.0% on Android, and 86.7% on Linux, indicating that commercial-grade fingerprinting can reliably single out a large fraction of users in practice.

Notably, cohort behavior does not consistently track user uniqueness. For *Akamai*, macOS and Linux exhibit nearly identical uniqueness rates (35.7% and 37.1%), yet their Simpson Indices differ by a factor of 3.5 $\times$: 135.48($\times 10^{-7}$) on macOS versus 478.27 on Linux.

Because the Simpson Index directly measures the probability that two randomly selected users share a fingerprint, this difference corresponds to a concrete change in likelihood: approximately 1 in 73,809 on macOS versus 1 in 20,909 on Linux. In other words, two randomly selected users are 3.5 $\times$ more likely to share a fingerprint on Linux than on macOS.

This reflects a fundamental difference in cohort structure. While a similar fraction of users are unique on both platforms, the remaining non-unique users are distributed differently: on macOS they are spread across many smaller cohorts, whereas on Linux they are concentrated into fewer, larger cohorts. These results demonstrate that uniqueness alone is insufficient to characterize fingerprinting risk, as similar uniqueness rates can mask substantial differences in how users are grouped.

Key Insights: At real-world scale, a large fraction of users remain uniquely identifiable, confirming the effectiveness of modern fingerprinting techniques in practice and highlighting the impact of characteristic selection. However, uniqueness alone does not fully capture fingerprinting risk. Cohort-based analysis reveals that users deemed non-unique can be distributed very differently, either across many small cohorts or concentrated into a few large ones, leading to substantially different privacy implications. Consequently, **evaluating fingerprinting requires jointly considering both uniqueness and cohort structure**, as similar uniqueness rates can mask meaningful differences in how easily users can be re-identified in practice.

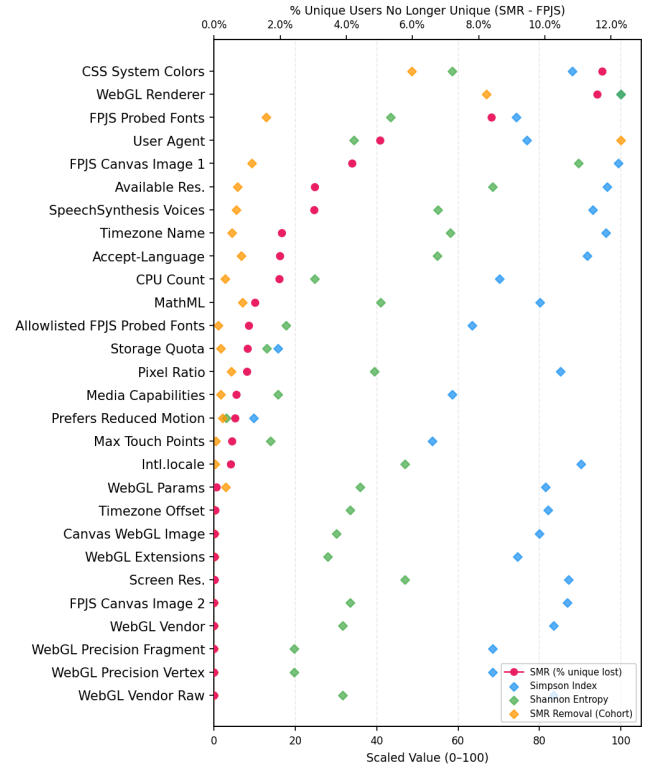


Figure 2: Individual measurement impact on fingerprint uniqueness across complementary metrics (Simpson Index, Shannon Entropy, and Single Measurement Removal (SMR)).

3.5 Measurement Contribution to Uniqueness

We analyze the data from multiple perspectives to identify high-impact mitigation targets to develop effective fingerprinting defenses.

Methodology: To evaluate the distribution of individual measurements, we analyze each using three complementary techniques: (1) **Simpson Index (Sim)**: measures concentration by estimating the probability that two users share the same value. (2) **Shannon Entropy**: captures the average information content of a measurement. (3) **Single Measurement Removal (SMR)**: quantifies the marginal contribution of a measurement by removing it and observing the impact on unique users and cohort score.

While the Simpson Index and Shannon entropy characterize value distributions, SMR captures the direct impact of a measurement on fingerprint uniqueness.

Measurement Impact Analysis: As illustrated in Figure 2, each row represents a single measurement evaluated using our complementary techniques and ranked by its Single Measurement Removal (SMR) impact under the *Fingerprint Pro* model, aggregated across all OSes.

The top x-axis shows the SMR effect, i.e., the percentage of users that cease to be unique when the measurement is removed. The bottom x-axis reports the inverted Simpson Index, Shannon entropy, and SMR cohort improvement, each normalized to a 0–100 scale relative to their maximum values. The Simpson Index is inverted ($1 - \text{val}$) so that higher values indicate greater dispersion across the population, while higher Shannon entropy reflects greater identifying information. SMR cohort improvement captures the increase in cohort score when a measurement is removed, indicating its contribution to fragmenting users. Measurements are included if their SMR impact is $\geq 0.5\%$ or if any normalized metric is ≥ 65 .

Limited Impact of Individual Measurements: Using SMR, we find that removing a single high-impact measurement when all others are retained can eliminate uniqueness for 11.7% of previously unique users. However, this effect drops off rapidly: only 8 measurements reduce uniqueness by more than 2%, indicating that no single measurement is critical to the overall effectiveness of a fingerprinter. Despite many measurements exhibiting high entropy or Simpson Index values (e.g., Screen Resolution), their individual contribution to uniqueness remains limited.

Key Insights: While removing any one of a small number of measurements reduces uniqueness by $\geq 5\%$, these are clear outliers: fingerprinting is generally not driven by any single dominant measurement. Moreover, commonly used statistical measures such as Shannon entropy or Simpson Index do not consistently predict a measurement’s impact on uniqueness when considered individually.

Instead, most measurements contribute only marginally on their own, and fingerprinting effectiveness largely arises from the interaction of multiple, partially independent measurements. **Effective defenses should therefore prioritize combinations of high-impact measurements, rather than optimizing them in isolation.**

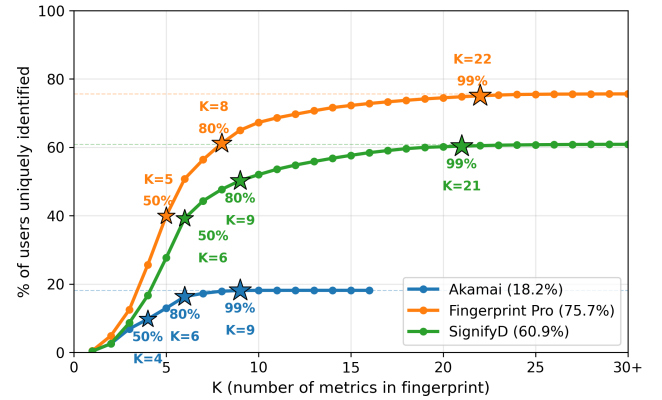


Figure 3: Cumulative coverage of unique users under incremental fingerprint construction.

3.6 Incremental Fingerprint Construction

Given that no individual measurement is crucial, we ask a complementary question: how many measurements are required in an optimal scenario to identify nearly all of the unique users that can be identified?

Methodology: Starting from an empty fingerprint, we greedily add measurements one at a time. At each step, we select the measurement whose addition maximizes the number of newly unique users, compute fingerprints using only the measurements selected so far, and record the cumulative number of unique users. This process continues until all measurements in the fingerprinter’s set have been included. We perform this analysis both overall and per operating system using the same dataset.

Individual Attributes: Figure 3 shows the cumulative coverage curves for our three modeled fingerprinters’ measurements. The curves reveal that a small number of measurements produce disproportionately large gains when added.

For example, adding Timezone Name at position 4 adds 2.55 million unique users on Windows (*Fingerprint Pro*), while User Agent adds 1.07 million on Android (*SignifyD*). These measurements are not necessarily dominant in isolation (cf. Section 3.5), but they interact strongly with previously selected measurements, creating many newly unique combinations.

It is noteworthy that the greedy ordering differs substantially across platforms. On Windows, fonts, WebGL renderer, and canvas dominate early positions; on Android, locale and CSS colors lead; on Linux and macOS, timezone and screen availability are among the top measurements.

Concentration of Fingerprinting Power: The shape of the curves directly addresses whether fingerprintability arises from aggregating many weak measurements or a concentrated set of strong ones. Rather than increasing linearly, the curves are sharply concave: across all three fingerprinters, no more than 10 measurements suffice to reach $\sim 80\%$ of unique users, while on a per-OS basis at least half of all measurements contribute less than 1%.

<i>Fingerprinting Technique</i>	<i>Technique Description</i>	<i>Fingerprinting Protection Technique</i>
FPT Phase 0 (Pre-2023 - Experience-Driven and Best Effort)		Firefox 16, 51, 87 (Deployed)
CPU Core Count (Experience-Driven)	returns the logical core count	cap the result to 128
WebGL Renderer String	returns raw renderer string of graphics driver	report sanitized string of graphic card driver
Patch Version	returns full version string of the browser	reports the patch version as .0
FPT Phase 1 (2023-2024 - Experience-Driven and Best Effort)		Firefox 118 (Deployed)
Fonts	exposes installed fonts	limit fonts to a reduced, specific list
Math Functions	math operations expose decimal differences	force all calculations inside math library
Web Audio	audio operations expose data differences	force all calculations inside math library
Canvas	allows drawing images using JS	add small bits of randomness into the image
FPT Phase 2 (2025 - Data-Driven Analysis)		Firefox 143 (Deployed)
CPU Core Count (Data-Driven)	returns the logical core count	report 4 if logical core count ≤ 4 ; else 8
Available Screen Resolution	returns available screen resolution	report screen resolution minus fixed offset
Font Updates	exposes installed fonts	exposes some fonts only in certain locales
FPT Phase 3 (2025-2026 - Data-Driven Analysis)		Firefox 157 (Experimental)
WebGL Vendor and Renderer	returns details of graphics hardware & software	reports constant values
Storage Quota	returns approximate space available to an origin	reports constant value
FPT Phase 4 (2026 and Beyond - Data-Driven Analysis)		(Under Consideration)
Software-Rendered Canvas	canvas draws images using dedicated hardware	canvas uses software backend to render
Accept-Language	exposes a list of locales a user accepts	report only first of list, and collapse locales
Timezone	returns timezone	collapse functionally identical timezones

Table 2: Fingerprinting Techniques and proposed Fingerprinting Protection Techniques (FPT).

However, this concentration is not extreme. While a few measurements have relatively disproportionate impact, as noted above, even the most powerful individual measurement identifies fewer than 1.2% of users, and reaching 99% coverage requires 7–21 measurements depending on OS. Thus, identifying power is concentrated in a small set of interacting measurements rather than a few dominant attributes. This result highlights that fingerprinting effectiveness emerges from combinations of measurements rather than any single dominant high-entropy feature.

Key Insights: Fingerprint uniqueness is concentrated in a small subset of measurements: no more than 10 suffice to identify 80% of unique users, while a majority contribute negligible additional signal.

3.7 From Measurement to Defenses

The preceding analyses provide a direct basis for prioritizing fingerprinting defenses. Sections 3.5 and 3.6 show that fingerprintability is driven by a concentrated set of interacting measurements, and that high entropy alone does not reliably predict a measurement’s contribution to uniqueness. Accordingly, rather than attempting to eliminate all sources of entropy, we prioritize measurements whose reduction or standardization has the greatest impact on uniqueness, while accounting for interactions between measurements and differences across platforms.

We use these findings to guide the mitigations evaluated in Section 4, subject to the web compatibility constraints described in Section 2.4. Section 4 presents these protections according to Firefox’s deployment timeline: Phases 0–1 capture earlier experience-driven mitigations, while Phases 2–4 apply the data-driven approach developed from our measurement study. This distinction allows us to

evaluate how progressively targeting the measurements identified by our analysis reduces fingerprintability in practice.

4 Real-World Mitigation

Section 3 showed that fingerprintability is driven by a relatively small set of interacting measurements whose importance varies across platforms. We now translate these findings into practical defenses by targeting high-impact measurements, while balancing the resulting privacy gains against web compatibility constraints.

This allows us to address the central question of our analysis: whether population-scale measurements can identify practical mitigation targets whose combined application materially reduces fingerprintability. Evaluated against simulated commercial fingerprinting services, these techniques reduce user uniqueness from 75% to 46% while preserving web compatibility.

We first situate our data-driven mitigations within Firefox’s existing fingerprinting protections and deployment history, before evaluating their impact on user uniqueness and cohort structure.

Background on Fingerprinting Protection in Firefox: Firefox has introduced multiple mechanisms over time to limit information exposure and tracking on the web. Many of these mechanisms fall under the umbrella of *Enhanced Tracking Protection (ETP)* [58], which encompasses privacy protections including script blocking, storage isolation, and fingerprinting defenses. Some of these protections are enabled by default, while others are activated only in *Private Browsing Mode (PBM)* [70]. PBM applies behaviors associated with *Strict Enhanced Tracking Protection (ETP Strict)*. Users may also enable *ETP Strict* globally to apply these protections to regular browsing. The Fingerprinting Protection Techniques (FPT)

in this paper are part of the Fingerprinting Protection feature (FPP) which is enabled in PBM and ETP Strict.

4.1 Fingerprinting Protection Techniques

Section 3 showed that fingerprinting is driven by a relatively small set of high-impact, interacting measurements that account for a disproportionate share of user uniqueness at scale. Building on this insight, we use our simulation platform and large-scale dataset to systematically evaluate fingerprinting protection techniques, quantifying each technique’s impact on user uniqueness and cohort score across five phases (cf. Table 2):

Phase 0 consists of ad hoc and best-effort mitigations, without a unified anti-fingerprinting strategy. **Phase 1** represents the first coordinated effort, guided by experience-driven tech leads rather than data. **Phase 2** introduces data-driven techniques derived from our initial measurement study, tempered by web compatibility reports from Phase 1 deployments (cf. Section 2.4); the resulting design changes are noted with each technique below. **Phase 3** expands on this approach with more aggressive mitigations informed by our second round of analysis as well as generative AI-assisted reverse engineering of sites using the affected APIs. These protections are under evaluation. **Phase 4** outlines further, forward-looking mitigations that push the boundary of entropy reduction while carefully considering their feasibility in a production browser.

FPT Phase 0 (Pre-2023)

CPU Core Count (Experience-Driven): Modern CPUs expose multiple logical processors, reported to websites via `navigator.hardwareConcurrency` [67] which has been identified as a fingerprinting vector as early as 2018 [83, 89]. Our data confirms this, as we observed 99 distinct values ranging from 1 to 768. Since 2016, Firefox has capped this value: 16 until 2021, then 128, disguising long-tail users.

WebGL Renderer String: WebGL (Web Graphics Library) is a JavaScript API for rendering 2D and 3D graphics [78] that, via `WEBGL_debug_renderer_info`, can reveal graphics hardware details [22]. Since 2021, Firefox mitigates this by returning a sanitized renderer string (e.g., “Apple M1, or similar”) to reduce entropy.

Patch Version: The browser version string follows the format `major.minor.patch` and is exposed via the `User-Agent` header and through `navigator.userAgent`. Since patch updates introduce no web-visible changes, Firefox has omitted the patch number since 2012 (e.g., `143.0.1` → `143.0`) to reduce fingerprinting entropy.

FPT Phase 1 (2023-2024)

Fonts: Fonts are a major source of fingerprinting entropy [46, 106], as scripts can detect installed fonts via rendering differences. Phase 1 mitigates this by restricting websites to standardized, platform-specific font lists. This approach introduced user-reported regressions [53], which we addressed by restoring some affected fonts to the allowlist.

Math Functions: OS-specific implementations of trigonometric functions (e.g., `Math.sin`, `cos`, `tan` [64]) introduce fingerprinting

entropy [88]. Firefox mitigates this by using the `fdlibm` library [31] to standardize results.

Web Audio: The Web Audio API [77] can be used for fingerprinting via system-specific processing differences [22]. Firefox mitigates this by standardizing computations using `fdlibm`, producing uniform outputs across users.

Canvas: Canvas fingerprinting uses the HTML5 `<canvas>` element [75] to derive device identifiers by rendering hidden content and exploiting subtle output differences across systems, extracted via APIs such as `.toDataURL()` [21]. Firefox mitigates this by deploying Canvas Randomization, injecting deterministic, context-dependent noise into outputs. This approach preserves stability within a domain and session while preventing cross-site reuse. Although limited, the injected noise did still result in reports of unexpected behavior and broken websites [54]; for example joining Google Meet calls failed until we exempted uniformly colored canvases from randomization [52]. Unlike other mitigations discussed in this paper, canvas randomization does not reduce exposed entropy but instead disrupts fingerprinting attempts. As a result, its effectiveness cannot be directly quantified through uniqueness metrics; we discuss its benefits in Section 5.

FPT Phase 2 (2025)

CPU Core Count (Data-Driven): Our empirical analysis showed that Phase 0 did not sufficiently reduce entropy from CPU core counts. In Phase 2, we cap values at 8 (≥ 8 cores) or 4 (< 8) to balance privacy and compatibility. Originally a constant value of 4 was chosen, but this degraded video encoding quality on Google Meet [55], motivating the two-value design above.

Available Screen Resolution: The two read-only screen interfaces `screen.availHeight`, `availWidth` [72] reveal screen resolution minus OS UI elements. We standardize these values using fixed offsets to prevent custom interface configurations from contributing to fingerprinting. On Windows, 48px is the dominant value at 51% of submissions (P96); on macOS, 76px is the 4th most common at 4.7% (P50); on Linux, 0px dominates at 58% (vs 10% for the next-most-common 40px).

Font Updates: We analyzed missing fonts from the Phase 1 allowlist and identified rarely installed ones, selecting nine also used by *Fingerprint Pro*. As these fonts are primarily used in specific locales, we proposed exposing them only when the corresponding locale is set, preserving compatibility while reducing entropy.

FPT Phase 3 (2025-2026)

WebGL Vendor and Renderer: Building on the WebGL renderer sanitization from Phase 0, both the renderer and the associated vendor string (identifying the graphics manufacturer) can be replaced with standardized values, e.g., `Mozilla`. Graphics-heavy sites that select rendering paths from these values are exempted, following static analysis of website JavaScript [56].

Storage Quota: The standard async method `estimate()` of the `StorageManager` [73] exposes the amount of storage available to

a website for local data. Because websites should only use this as an estimate rather than a guarantee, we can normalize it to 10GB, which is reported by 91.82% of users.

FPT Phase 4 (2026 and Beyond)

Software-Rendered Canvas: Some canvas entropy arises from graphics hardware differences [21]. Using software rendering reduces this variability, offering privacy gains with low compatibility risk.

Accept-Language: The Accept-Language header exposes user language preferences, contributing significant entropy while remaining essential for compatibility. We evaluate reducing it by sending only the primary locale and collapsing regional variants (e.g., en-US → en). However, this approach remains under consideration due to compatibility risks, such as serving content in unintended languages.

Timezone: Web APIs expose UTC offset as well as timezone name (e.g., America/New_York), with the latter adding significant entropy (e.g., near-identical zones having distinct names). We evaluate collapsing identifiers per UTC offset, reducing entropy but introducing compatibility risks (e.g., DST discrepancies).

4.2 Effectiveness of Protections in Practice

Now, we evaluate the cumulative impact of our fingerprinting protection techniques by analyzing their impact on user uniqueness through the lens of three real-world fingerprinting services (*Fingerprint Pro*, *Signifyd*, and *Akamai*) over our dataset. Figure 4 reports the fraction of users classified as unique (solid lines) and the corresponding cohort score (Simpson Index, dashed lines) for each protection technique.

Following the initial “experience-driven” phases, the data-driven mitigations introduced in Phases 2 and 3 steadily reduce the fraction of unique users for *Fingerprint Pro* from approximately 75% to 44%, while increasing the Simpson Index by an order of magnitude (weighted average across operating systems for *Fingerprint Pro*). Similar trends hold for *Signifyd*, where uniqueness falls from 60.9% to 35.4% over the same phases with a comparable rise in the Simpson Index. Furthermore, ongoing work in Phase 4 aims to further reduce uniqueness by an additional ~10 percentage points.

These improvements come unevenly: only 6 out of 64 scenarios (8 protection techniques from Phase 2 onward $\times$ 4 OSes $\times$ 2 fingerprinters²) reduce uniqueness by more than 15% relative to the prior protection. Among these, Software Canvas accounts for three (both *Fingerprint Pro* and *Signifyd* for Windows, and *Signifyd* for Linux), Available Screen Resolution for two (macOS for both fingerprinters), and Timezone for one (*Signifyd* for Windows). While only half of scenarios exceed 5% relative improvement, every protection technique reduces uniqueness by at least 5% for some OS/fingerprinter combination.

Together, these protection techniques drive down uniqueness nearly 30 percentage points, but examined individually (Figure 5) the picture is different. For example, consider the discrepancy in the

WebGL Vendor and Renderer protection. It reports an 8.8 percentage point improvement when applied individually, but only a 1.1 point improvement when applied cumulatively. This is explained by redundancy in the techniques: the earlier WebGL Renderer protection (which sanitizes the Renderer string) subsumes most of the gains of the later one, which sets the string to a constant value.

However, the opposite can also be true: on Windows for *Fingerprint Pro*, Software Canvas reports only a 1.3 percentage point improvement individually, but a 5.8 point improvement cumulatively. Its cohort score tells a similar story: individually, Software Canvas barely moves the score (a 0.3×10^{-7} increase from baseline, cf. Table 1). Cumulatively, it jumps from 318.2 to 811.5×10^{-7} (a $2.6\times$ increase), driven by prior protections that have already concentrated non-unique users into smaller cohorts.

Indeed, browser features can be correlated [8], and our work confirms that interaction effects between protections amplify modest individual contributions and reinforces that OS segmentation is essential for designing protections and evaluating their impact. Importantly, these gains are achieved while maintaining web compatibility.

We deliberately avoid simulating adaptive adversaries, as doing so would require subjective assumptions about feature selection and matching strategies, and risk inadvertently designing stronger fingerprinting techniques. Instead, we evaluate robustness across feature subsets and optimal selection bounds, which provide attacker-agnostic guarantees.

Overall, these results demonstrate that data-driven, deployable defenses significantly weaken real-world fingerprinting systems at scale, reducing user uniqueness from 75% to 46%.

The reduction we report is specific to Firefox and its user population, and the specific numbers would not transfer directly to other browsers—particularly those built on different engines. Other engines expose Web APIs that Firefox does not, enlarging the fingerprinting surface: Chromium, for example, implements the read-only property `navigator.deviceMemory` as well as the API method `navigator.userAgentData.getHighEntropyValues()`. Internal implementations of APIs such as canvas and Web Audio also differ, producing different values and therefore different baseline entropy, and uniqueness depends further on population composition, which differs across browsers.

The mitigations themselves, however, are not Firefox-specific: each targets a standardized Web API, reduces the information that API exposes, and was designed against real web compatibility constraints. We therefore expect the same techniques to apply to other major browsers (e.g., Chrome, Safari, Edge, Brave; cf. Section 6.3), though the magnitude of the benefit would need to be measured in each.

5 Design Considerations

Building on the deployed protections of Section 4, we step back to situate our approach among other anti-fingerprinting strategies and outline remaining challenges.

5.1 Measurement-Driven Design

Our results yield five principles that depart from prior work.

²Because *Akamai* collects fewer metrics, it is more affected by individual protection techniques, so we focus on the two fingerprinters with larger measurement sets.

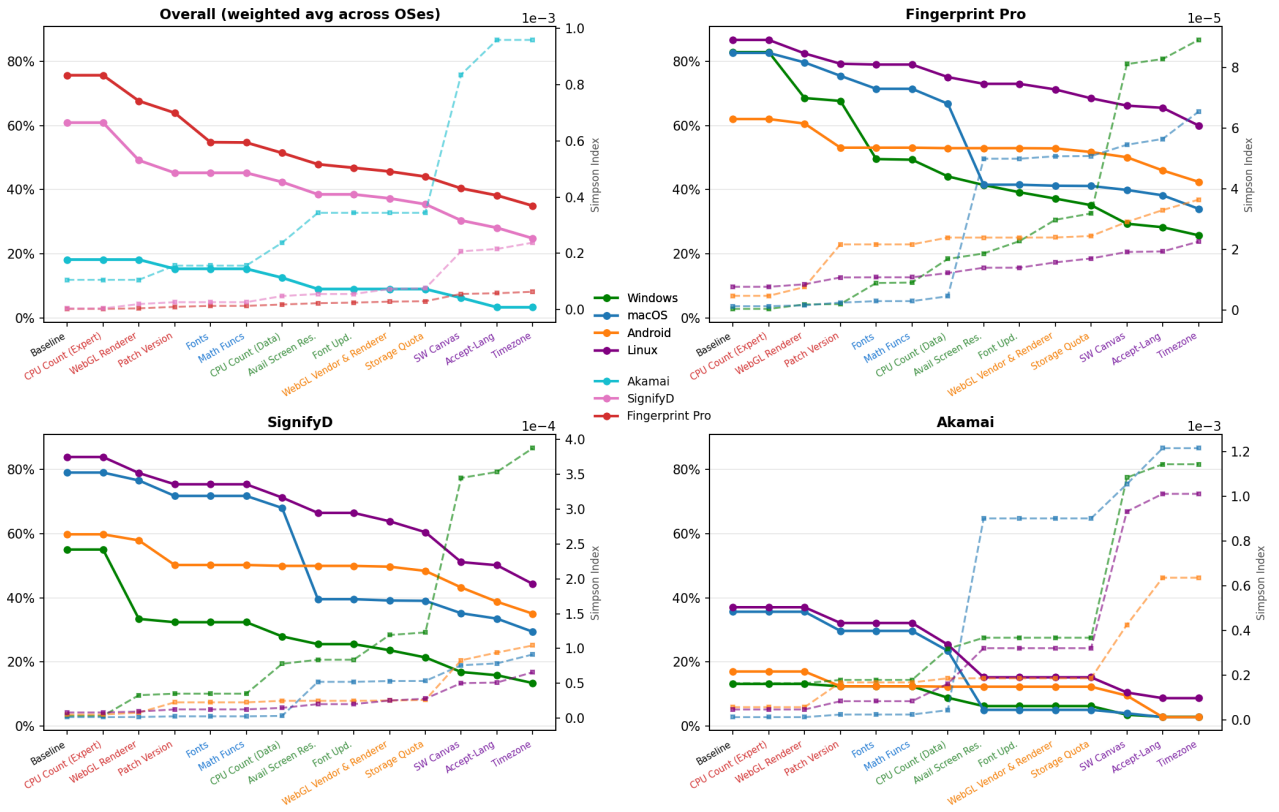


Figure 4: Impact of fingerprinting protections on user uniqueness and cohort score.

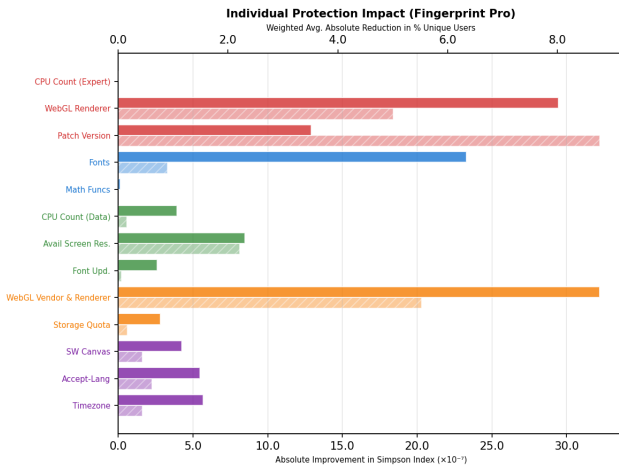


Figure 5: Impact of each fingerprinting protection technique applied in isolation against the baseline.

(1) **Select targets by removal, not entropy:** Shannon entropy and the Simpson Index do not reliably predict a measurement’s contribution to uniqueness (Figure 2). (2) **Report cohort structure, not uniqueness alone:** For *Akamai*, macOS and Linux differ

by 1.4 percentage points in uniqueness but 3.5× in Simpson Index (Section 3.4): two random Linux users share a fingerprint far more readily. (3) **Spend compatibility risk on the head, not the tail:** ten measurements suffice to identify 80% of unique users (Section 3.6); the remainder rarely justify breakage risk. (4) **Evaluate mitigations jointly:** software-rendered canvas gains 1.3 percentage points alone but 5.8 once other protections have concentrated cohorts (Windows, *Fingerprint Pro*). Assessed in isolation, such mitigations appear not to warrant their cost. (5) **Segment by operating system:** high-impact measurements vary by platform, so a single global policy is mis-targeted for most users.

5.2 Alternative Anti-Fingerprinting Approaches

Our work demonstrates that standardizing the values web APIs return can substantially reduce browser fingerprintability at scale. We briefly contrast this approach with two alternatives — randomization and targeted disruptions — and discuss where each fits.

Randomization: Fingerprinting techniques rely on web APIs returning stable and consistent values across time and contexts. Introducing randomness violates this assumption, thereby disrupting fingerprinting algorithms and reducing their ability to reliably re-identify users. Randomization-based defenses have been widely studied and are often detected or bypassed [82], yet they remain valuable; Brave applies them the most broadly of any major browser

(cf. Section 6.3). While their efficacy is difficult to measure without access to proprietary algorithms, they still provide benefits. Fingerprinting providers adapt at different speeds, and randomization increases the cost of recovering underlying signals, either through added engineering effort or performance overhead. It is also possible that sophisticated systems may detect and discard randomized measurements altogether [29, 49]. Consequently, randomization can provide both short- and long-term privacy benefits, though it may introduce compatibility issues and can be bypassed by determined adversaries.

Targeted Disruptions: Another approach is to identify fingerprinting attempts and selectively block or disrupt them. One method is to block network requests retrieving fingerprinting scripts by matching known URLs (e.g., through crowd-sourced blocklists). However, recent work [48] has shown that blocklist-based defenses are ineffective for a large fraction of fingerprinting scripts as trackers can adopt evasion techniques such as CNAME cloaking [20] or simply rotate the script URL. Another method is to disrupt only the fingerprinting script on a webpage, without disrupting the rest of the page’s operation. By detecting the fingerprinting script using heuristics (typically static patterns), specific operations can be disrupted (e.g. by adding noise only to the fingerprinting script’s canvas operations). Safari deploys this approach, applying canvas noise outside Private Browsing only to scripts on a list distributed to the browser at runtime (cf. Section 6.3). Both these approaches are similarly prone to evasion, as they rely on static patterns, and as with randomization, fingerprinting providers can adapt to such defenses with varying effort and speed. Further, both script blocking and targeted API restriction can cause significant website breakage [38].

Combining the Three: Ultimately, these approaches are complementary rather than mutually exclusive. Standardization provides significant and measurable improvements, but is limited by both web compatibility constraints and when users have uncommon hardware (e.g., extremely large monitors) or unusual settings (e.g., the Mandarin-localized visitor to a local Icelandic website from before). Randomization fills that gap, accepting some compatibility risk and the possibility of bypass in exchange for frustrating tracking attempts by at least some fingerprinting services for a period of time. Targeted disruptions then add a final layer by blocking scripts known to perform fingerprinting outright, before they get to execute at all. Firefox’s configuration reflects this layering: standardization across Phases 0–4, Canvas Randomization for irreducible canvas output, and Enhanced Tracking Protection blocklists for known fingerprinting URLs. An effective defense strategy should combine all three.

5.3 Outlook

Our results show that standardization informed by real-world population measurements can substantially reduce browser fingerprintability while preserving web compatibility. By aligning exposed characteristics with common values, browsers concentrate users into larger cohorts and mask atypical configurations, achieving meaningful privacy gains with fewer compatibility regressions

than more comprehensive approaches such as Tor Browser or RFP mode. Since Firefox 115, RFP has generated 153 breakage reports compared to FPP’s 59 over the same period, despite being used by at least three orders of magnitude fewer users [60].

Sustaining these gains, however, requires continued measurement. Even within the span of this study, we observed *Fingerprint Pro* and *Akamai* expand the measurements they collect, illustrating the continuing adaptation between fingerprinters and browser defenses. Our measurement-driven approach provides a basis for responding to this evolution: re-evaluating which measurements drive uniqueness as fingerprinting techniques change, and using these results to prioritize subsequent protections without unnecessarily sacrificing functionality.

6 Related Work

Since Eckersley’s 2010 demonstration of browser fingerprinting [21], the technique has become a widespread privacy threat. Numerous studies have documented both cookie-based [1, 2, 9, 15, 17, 19, 20, 22, 23, 26, 39, 50, 51, 80, 99] and fingerprinting-based tracking.

6.1 Fingerprinting-Based Tracking

We organize prior fingerprinting research around measurement, characteristics, and effectiveness.

a) Large-Scale Measurement Studies: Prior work has measured in-the-wild fingerprinting for tracking [1, 22, 38, 48] and authentication [90]. Notably, Iqbal et al. [38] used learning-based techniques to detect previously unknown fingerprinting scripts without relying on known fingerprinting APIs. Other studies examine user fingerprints themselves: Wu et al. [105] analyze attacker-crafted fingerprints from commercial sites, while Pau et al. [85] provide a small open dataset.

b) High-Entropy Fingerprinting Characteristics: Prior work identifies signals that drive fingerprintability. Su et al. [93] and Nikkha Bahrami et al. [83] detect emerging fingerprinting APIs using program analysis and machine learning. Other work demonstrates the strength of specific measurements, including CSS-based probes [46] and features such as fonts, plugins, and Canvas [42], while entropy analyses highlight window size, canvas rendering, and the speech engine [43]. Bacis et al. [8] further studied the entropy of *correlated* browser APIs. Together, these findings show that a small set of measurements contributes disproportionately to user uniqueness.

c) Fingerprinting Effectiveness: Prior work evaluates fingerprinting effectiveness in practice. Liu et al. [47] provide evidence of its use in tracking and advertising, while Gómez-Boix et al. [33] show that uniqueness decreases at scale. Longitudinal and cross-platform studies demonstrate persistence over time [87] and cross-browser tracking via OS- and hardware-level features [16]. Andriamilanto et al. [4] further show that fingerprints exhibit high uniqueness and stability. Together, these results highlight both the effectiveness and limitations of fingerprinting at scale.

6.2 Distinction from Prior Work

The most comprehensive prior studies rely on volunteer-based or website-specific datasets and are limited in scale, typically ranging from 120,000 to 2 million users. In contrast, we measure fingerprinting *in the wild* using data from nearly 25 million clients, representing a **12× to 208× increase in scale** over prior work. This enables a more representative and statistically robust assessment of fingerprint uniqueness and cohort score. Moreover, rather than relying on synthetic or researcher-curated datasets, our analysis uses **real-world telemetry data**, capturing actual user configurations at population scale. By modeling commercial fingerprinting services and deploying mitigations in Firefox, a production browser, we **bridge the gap between measurement and practical defense**.

6.3 Fingerprinting Defenses in Other Browsers

Other browsers have accumulated fingerprinting defenses over time, generally targeting individual APIs without publicly measuring their effect on user uniqueness. Chrome abandoned its Privacy Budget proposal [36], which sought to bound the entropy observable by a site, while proposals to reduce Accept-Language entropy [35] and randomize canvas readbacks [34] remain unshipped as of August 2026. Safari limits font and User-Agent entropy [103] and randomizes canvas, WebGL, and Web Audio outputs [101, 102]. Brave extensively employs randomization (“furling”) across canvas, Web Audio, WebGL, hardwareConcurrency, and deviceMemory [13, 14], and has further standardized WebGL identifiers while randomizing extension lists [12].

The scope of these defenses also varies by browsing mode and configuration. Safari applies canvas randomization in Private Browsing, but otherwise restricts it to selected scripts. Brave previously offered a stricter opt-in mode [11]. Tor Browser [95, 96] instead emphasizes uniformity across APIs [86], supplemented by randomized canvas output [63]; Mullvad Browser provides similar hardening without the Tor network [79]. Firefox derivatives likewise adapt these mechanisms: LibreWolf has long enabled Resist Fingerprinting and may migrate toward Firefox’s fingerprinting-protection framework for per-domain overrides [45].

Consequently, the same API can receive different treatment across browsers; for example, the navigator property hardwareConcurrency is capped by Firefox, randomized by Safari and Brave, and unmodified by Chrome. While guidance for specification authors exists [100], it does not establish which characteristics to prioritize or quantify their contribution to user uniqueness. Our work complements these defenses by providing population-scale measurements for prioritizing high-impact fingerprinting surfaces.

7 Conclusion

To the best of our knowledge, we present the first large-scale, real-world study of browser fingerprinting based on telemetry data from nearly 25 million Firefox users. At this scale, we show that fingerprint uniqueness is highly sensitive to dataset size and remains structurally significant: approximately 75% of users are uniquely identifiable by commercial-grade fingerprinting services without protections.

Overall, our findings demonstrate that browser fingerprinting is not an unavoidable property of the web, but can be substantially

reduced at scale. Identifiability is driven by a relatively small set of high-impact, interacting features, rather than the accumulation of many independent signals. This concentration enables effective mitigation through a data-driven, privacy-through-conformity strategy that enlarges anonymity sets while preserving functionality. By simulating real-world fingerprinting services, we evaluate concrete browser engineering decisions and quantify their privacy impact. The resulting protections, shipped in Firefox 143, reduce uniqueness from 75% to 46% while maintaining web compatibility.

References

- [1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juárez, Arvind Narayanan, and Claudia Diaz. 2014. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Conference on Computer and Communications Security*. ACM, New York, NY, USA.
- [2] Zubair Ahmad, Samuele Casarin, and Stefano Calzavara. 2023. An Empirical Analysis of Web Storage and Its Applications to Web Tracking. In *Transactions on the Web*. ACM, New York, NY, USA.
- [3] Akamai. 2026. Bot Manager. <https://www.akamai.com/products/bot-manager> (accessed: September, 2026).
- [4] Nampoina Andriamilanto, T. Allard, and Gaëtan Le Guelvouit. 2020. “Guess Who?” Large-Scale Data-Centric Study of the Adequacy of Browser Fingerprints for Web Authentication. In *Transactions on the Web*. Springer, Cham, Switzerland.
- [5] Anthropic. 2026. The best AI for developers. <https://claude.com/solutions/coding> (accessed: September, 2026).
- [6] Apple. 2026. Apple announces powerful new privacy and security features. <https://www.apple.com/au/newsroom/2023/06/apple-announces-powerful-new-privacy-and-security-features/> (accessed: September, 2026).
- [7] Mohammadreza Ashouri. 2019. A Large-Scale Analysis of Browser Fingerprinting via Chrome Instrumentation. In *Conference on Internet Monitoring and Protection*. IARIA, Nice, France.
- [8] Enrico Baci, Igor Bilogrevic, Robert Busa-Fekete, Asanka Herath, Antonio Sartori, and Umar Syed. 2024. Assessing Web Fingerprinting Risk. In *World Wide Web Conference*. ACM, New York, NY, USA.
- [9] Nataliia Bielova. 2017. Web Tracking Technologies and Protection Mechanisms. In *Conference on Computer and Communications Security*. ACM, New York, NY, USA.
- [10] Igor Bilogrevic, Balazs Engedy, Judson L. Porter III, Nina Taft, Kamila Hasanebeg, Andrew Paseltiner, Hwi Kyoung Lee, Edward Jung, Meggyn Watkins, PJ McLachlan, and Jason James. 2021. “Shhh...be quiet!” Reducing the Unwanted Interruptions of Notification Permission Prompts on Chrome. In *USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA.
- [11] Brave. 2026. Brave browser simplifies its fingerprinting protections. <https://brave.com/privacy-updates/28-sunsetting-strict-fingerprinting-mode/> (accessed: September, 2026).
- [12] Brave. 2026. Brave improves protections against GPU fingerprinting. <https://brave.com/privacy-updates/38-webgl-webgpu-fingerprinting-protections/> (accessed: September, 2026).
- [13] Brave. 2026. Fingerprinting defenses 2.0. <https://brave.com/privacy-updates/4-fingerprinting-defenses-2.0/> (accessed: September, 2026).
- [14] Brave. 2026. Fingerprinting Protections. <https://github.com/brave-browser/wiki/Fingerprinting-Protections> (accessed: September, 2026).
- [15] Tomasz Bujlow, Valentin Carela-Español, Josep Solé-Pareta, and Pere Barlet-Ros. 2017. A Survey on Web Tracking: Mechanisms, Implications, and Defenses. In *Proceedings of the IEEE*. IEEE, Piscataway, NJ, USA.
- [16] Yinzi Cao, Song Li, and Erik Wijmans. 2017. (Cross-)Browser Fingerprinting via OS and Hardware Level Features. In *Network and Distributed System Security Symposium*. Internet Society, Reston, VA, USA.
- [17] Quan Chen, Panagiotis Ilia, Michalis Polychronakis, and Alexandros Kapravelos. 2021. Cookie Swap Party: Abusing First-Party Cookies for Web Tracking. In *World Wide Web Conference*. ACM, New York, NY, USA.
- [18] CreepJS. 2026. CreepJS - Real-time analysis of your browser’s unique characteristics. <https://creepsjs.org/> (accessed: September, 2026).
- [19] Savino Dambra, Iskander Sánchez-Rola, Leyla Bilge, and Davide Balzarotti. 2022. When Sally Met Trackers: Web Tracking From the Users’ Perspective. In *USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA.
- [20] Ha Dao and Kensuke Fukuda. 2020. Characterizing CNAME Cloaking-based Tracking on the Web. In *Traffic Monitoring and Analysis*. Springer, Cham, Switzerland.
- [21] Peter Eckersley. 2010. How Unique is Your Web Browser?. In *Privacy Enhancing Technologies*. De Gruyter, Berlin, Germany.
- [22] Steven Englehardt and Arvind Narayanan. 2016. Online Tracking: A 1-million-site Measurement and Analysis. In *Conference on Computer and Communications Security*.

- Security. ACM, New York, NY, USA.
- [23] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. 2015. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In *World Wide Web Conference*. ACM, New York, NY, USA.
 - [24] Julian Fietkau, Kashyap Thimmaraju, Felix Kybranz, Sebastian Neef, and Jean-Pierre Seifert. 2021. The Elephant in the Background: A Quantitative Approach to Empower Users Against Web Browser Fingerprinting. In *Workshop on Privacy in the Electronic Society*. ACM, New York, NY, USA.
 - [25] Fingerprint. 2026. Fingerprint Pro. <https://fingerprint.com/> (accessed: September, 2026).
 - [26] Gertjan Franken, Tom van Goethem, and Wouter Joosen. 2018. Who Left Open the Cookie Jar? A Comprehensive Evaluation of Third-Party Cookie Policies. In *USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA.
 - [27] GitHub account: arkenfox. 2026. TorZillaPrint - A comprehensive, all-in-one, fingerprinting test suite. <https://github.com/arkenfox/TZP> (accessed: September, 2026).
 - [28] GitHub account: fingerprintjs. 2026. Content blockers. https://github.com/fingerprintjs/fingerprintjs/blob/30361297159ef130fe18635dae2a595a16083159/docs/content_blockers.md (accessed: September, 2026).
 - [29] GitHub account: fingerprintjs. 2026. FingerprintJS - Source Code Comment: 'Some browsers add a noise to the canvas'. <https://github.com/fingerprintjs/fingerprintjs/blob/30361297159ef130fe18635dae2a595a16083159/src/sources/canvas.ts#L83> (accessed: September, 2026).
 - [30] GitHub account: fingerprintjs. 2026. Source Code: dom_blockers.ts. https://github.com/fingerprintjs/fingerprintjs/blob/master/src/sources/dom_blockers.ts (accessed: September, 2026).
 - [31] GitHub account: freemint. 2026. fdlibm Math Library. <https://github.com/freemint/fdlibm> (accessed: September, 2026).
 - [32] Alejandro Gómez-Boix, Davide Frey, Yérom-David Bromberg, and Benoit Baudry. 2019. A Collaborative Strategy for Mitigating Tracking through Browser Fingerprinting. In *Workshop on Moving Target Defense*. ACM, New York, NY, USA.
 - [33] 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 *World Wide Web Conference*. ACM, New York, NY, USA.
 - [34] Google. 2026. Noising canvas readbacks in Incognito. <https://chromestatus.com/feature/5589949602332672> (accessed: September, 2026).
 - [35] Google. 2026. Reduce fingerprinting in the Accept-Language header. <https://github.com/explainers-by-googlers/reduce-accept-language> (accessed: September, 2026).
 - [36] Google. 2026. The Privacy Budget explainer (no longer active). <https://github.com/bslassey/privacy-budget> (accessed: September, 2026).
 - [37] Inria France (Team of Researchers). 2026. AmIUnique - Inform users about their position within the web ecosystem. <https://amiunique.org/> (accessed: September, 2026).
 - [38] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. 2020. Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors. In *Symposium on Security and Privacy*. IEEE, Piscataway, NJ, USA.
 - [39] Umar Iqbal, Peter Snyder, Shitong Zhu, Benjamin Livshits, Zhiyun Qian, and Zubair Shafiq. 2018. AdGraph: A Graph-Based Approach to Ad and Tracker Blocking. In *Symposium on Security and Privacy*. IEEE, Piscataway, NJ, USA.
 - [40] Soroush Karami, Panagiotis Ilia, Konstantinos Solomos, and Jason Polakis. 2020. Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting. In *Network and Distributed System Security Symposium*. Internet Society, Reston, VA, USA.
 - [41] Pierre Laperdrix, Natalia Bielova, Benoit Baudry, and Gildas Avoine. 2019. Browser Fingerprinting: A survey. In *Transactions on the Web*. ACM, New York, NY, USA.
 - [42] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. 2016. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. In *Symposium on Security and Privacy*. IEEE, Piscataway, NJ, USA.
 - [43] Vicki Wei Qi Lee, Shih Yin Ooi, and Ying Han Pang. 2023. Assessing the Importance of Browser Fingerprint Attributes towards User Profiling through Clustering Algorithms. In *Symposium on Computer Applications & Industrial Electronics*. IEEE, Piscataway, NJ, USA.
 - [44] Song Li and Yinzhi Cao. 2020. Who Touched My Browser Fingerprint?: A Large-scale Measurement Study and Classification of Fingerprint Dynamics. In *Internet Measurement Conference*. ACM, New York, NY, USA.
 - [45] LibreWolf. 2026. Issue 2121: Move from RFP to FPP. <https://codeberg.org/librewolf/issues/issues/2121> (accessed: September, 2026).
 - [46] Xu Lin, Frederico Araujo, Teryl Taylor, Jiyong Jang, and Jason Polakis. 2023. Fashion Faux Pas: Implicit Stylistic Fingerprints for Bypassing Browsers' Anti-Fingerprinting Defenses. In *Symposium on Security and Privacy*. IEEE, Piscataway, NJ, USA.
 - [47] Zengrui Liu, Jimmy Dani, Yinzhi Cao, Shuijiang Wu, and Nitesh Saxena. 2025. The First Early Evidence of the Use of Browser Fingerprinting for Online Tracking. In *World Wide Web Conference*. ACM, New York, NY, USA.
 - [48] Elisa Luo, Tom Ritter, Stefan Savage, and Geoffrey M. Voelker. 2025. Canvassing the Fingerprinters: Characterizing Canvas Fingerprinting Use Across the Web. In *Internet Measurement Conference*. ACM, New York, NY, USA.
 - [49] Elisa Luo, Tom Ritter, Stefan Savage, and Geoffrey M. Voelker. 2026. Understanding Server-side Commercial Fingerprinting. In *World Wide Web Conference*. ACM, New York, NY, USA.
 - [50] Maryam Mehrzad, Kovila P. L. Coopamootoo, and Ehsan Toreini. 2021. How Can and Would People Protect From Online Tracking?. In *Privacy Enhancing Technologies*. De Gruyter, Berlin, Germany.
 - [51] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar R. Weippl. 2017. Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools. In *European Symposium on Security and Privacy*. IEEE, Piscataway, NJ, USA.
 - [52] Mozilla. 2023. Bug 1852541: meet.google.com - Cannot join meetings when Canvas Randomization is enabled. https://bugzilla.mozilla.org/show_bug.cgi?id=1852541 (accessed: September, 2026).
 - [53] Mozilla. 2026. Bug 1827475: Tracking bug for font visibility restrictions causing web compatibility issues. https://bugzilla.mozilla.org/show_bug.cgi?id=1827475 (accessed: September, 2026).
 - [54] Mozilla. 2026. Bug 1957426: Tracking bug for canvas randomization causing web compatibility issues. https://bugzilla.mozilla.org/show_bug.cgi?id=1957426 (accessed: September, 2026).
 - [55] Mozilla. 2026. Bug 1984132: Sent video in Google Meet is low quality. https://bugzilla.mozilla.org/show_bug.cgi?id=1984132 (accessed: September, 2026).
 - [56] Mozilla. 2026. Bug 2036879: Address Graphics-Heavy Website Vendor/Renderer detection. https://bugzilla.mozilla.org/show_bug.cgi?id=2036879 (accessed: September, 2026).
 - [57] Mozilla. 2026. Bug List - Breakeage from Incorrectly Reporting the Operating System. https://bugzilla.mozilla.org/buglist.cgi?quicksearch=1509829%2C1610762%2C1755832%2C1826098%2C1840385%2C1519122%2C1405810%2C1404608%2C1409001%2C1441003%2C1621988%2C1696669%2C1509829&list_id=17938923 (accessed: September, 2026).
 - [58] Mozilla. 2026. Enhanced Tracking Protection in Firefox. <https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop/> (accessed: September, 2026).
 - [59] Mozilla. 2026. Experimenter: Mozilla's Experimentation Platform. <https://experimenter.info/> (accessed: September, 2026).
 - [60] Mozilla. 2026. Fingerprinting/Breakage. <https://wiki.mozilla.org/Fingerprinting/Breakage> (accessed: September, 2026).
 - [61] Mozilla. 2026. Firefox Public Data Report. <https://data.firefox.com/dashboard/usage-behavior> (accessed: September, 2026).
 - [62] Mozilla. 2026. Firefox Source Tree Documentation - Code quality. <https://firefox-source-docs.mozilla.org/code-quality/index.html> (accessed: September, 2026).
 - [63] Mozilla. 2026. GeneratePlaceholderCanvasData.h. <https://searchfox.org/firefox-main/source/dom/canvas/GeneratePlaceholderCanvasData.h> (accessed: September, 2026).
 - [64] Mozilla. 2026. Math.sin(). https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/sin (accessed: September, 2026).
 - [65] Mozilla. 2026. Mozilla Data Collection. https://wiki.mozilla.org/Data_Collection (accessed: September, 2026).
 - [66] Mozilla. 2026. Mozilla Data Documentation. <https://docs.telemetry.mozilla.org/> (accessed: September, 2026).
 - [67] Mozilla. 2026. Navigator: hardwareConcurrency property. <https://developer.mozilla.org/en-US/docs/Web/API/Navigator/hardwareConcurrency> (accessed: September, 2026).
 - [68] Mozilla. 2026. Obnoxious HTTP (OHTTP) explained. <https://support.mozilla.org/en-US/kb/ohttp-explained> (accessed: September, 2026).
 - [69] Mozilla. 2026. Privacy Settings Breakage Study. https://docs.google.com/presentation/d/1OVtXAnyeBLX2N1yyZoTMP9AV_6HnI3mnXwIFOL7yOA/edit?slide=id.g2c198d8539_0_0#slide=id.g2c198d8539_0_0 (accessed: September, 2026).
 - [70] Mozilla. 2026. Private Browsing - Use Firefox without saving history. <https://support.mozilla.org/en-US/kb/private-browsing-use-firefox-without-history> (accessed: September, 2026).
 - [71] Mozilla. 2026. Resist Fingerprinting. <https://support.mozilla.org/en-US/kb/resist-fingerprinting> (accessed: September, 2026).
 - [72] Mozilla. 2026. Screen: availHeight property. <https://developer.mozilla.org/en-US/docs/Web/API/Screen/availHeight> (accessed: September, 2026).
 - [73] Mozilla. 2026. StorageManager: estimate() method. <https://developer.mozilla.org/en-US/docs/Web/API/StorageManager/estimate> (accessed: September, 2026).
 - [74] Mozilla. 2026. Telemetry Portal. <https://telemetry.mozilla.org/> (accessed: September, 2026).
 - [75] Mozilla. 2026. The Graphics Canvas element. <https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/canvas> (accessed: September, 2026).
 - [76] Mozilla. 2026. Tor Uplift: Fingerprinting Resistance. https://bugzilla.mozilla.org/show_bug.cgi?id=uplift_tor_fingerprinting (accessed: September, 2026).
 - [77] Mozilla. 2026. Web Audio API. https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API (accessed: September, 2026).

- [78] Mozilla. 2026. WebGL constants. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Constants/webgl_debug_renderer_info (accessed: September, 2026).
- [79] Mullvad and The Tor Project. 2026. Mullvad Browser. <https://mullvad.net/en/browser> (accessed: September, 2026).
- [80] Shaor Munir, Sandra Deepthy Siby, Umar Iqbal, Steven Englehardt, Zubair Shafiq, and Carmela Troncoso. 2022. CookieGraph: Understanding and Detecting First-Party Tracking Cookies. In *Computer and Communications Security*. ACM, New York, NY, USA.
- [81] Sebastian Neef. 2026. Uncovering Fingerprinting Networks. An Analysis of In-Browser Tracking using a Behavior-based Approach. <https://arxiv.org/abs/2210.11300> (accessed: September, 2026).
- [82] Hoang Dai Nguyen and Phani Vadrevu. 2025. Breaking the Shield: Analyzing and Attacking Canvas Fingerprinting Defenses in the Wild. In *World Wide Web Conference*. ACM, New York, NY, USA.
- [83] Pouneh Nikkhab Bahrami, Umar Iqbal, and Zubair Shafiq. 2021. FP-Radar: Longitudinal Measurement and Early Detection of Browser Fingerprinting. In *Privacy Enhancing Technologies*. De Gruyter, Berlin, Germany.
- [84] Nisenoff, Alexandra and Borem, Arthur and Pickering, Madison and Nakanishi, Grant and Thumpasery, Maya and Ur, Blase. 2023. Defining “Broken”: User Experiences and Remediation Tactics When Ad-Blocking or Tracking-Protection Tools Break a Website’s User Experience. In *USENIX Security Symposium*. USENIX Association, Anaheim, CA, USA.
- [85] Kiu Pau, Vicki Lee, Shih Yin Ooi, and Ying Pang. 2023. The Development of a Data Collection and Browser Fingerprinting System. *Sensors* (2023).
- [86] Mike Perry, Erinn Clark, Steven Murdoch, and Georg Koppen. 2018. The Design and Implementation of the Tor Browser [DRAFT]. <https://2019.www.torproject.org/projects/torbrowser/design/> (accessed: September, 2026).
- [87] Gaston Pugliese, Christian Riess, Freya Gassmann, and Zinaida Benenson. 2020. Long-Term Observation on Browser Fingerprinting: Users’ Trackability and Perspective. In *Privacy Enhancing Technologies*. De Gruyter, Berlin, Germany.
- [88] Takamichi Saito, Takafumi Noda, Ryohei Hosoya, Kazuhisa Tanabe, and Yuta Saito. 2019. On Estimating Platforms of Web User with JavaScript Math Object. In *Advances in Network-Based Information Systems*. ACM, New York, NY, USA.
- [89] Saito, Takamichi and Yasuda, Koki and Kazuhisa, Tanabe and Takahashi, Kazushi. 2018. Web Browser Tampering: Inspecting CPU Features from Side-Channel Information. In *Advances on Broad-Band Wireless Computing, Communication and Applications*. Springer International Publishing.
- [90] Asuman Senol, Alisha Ukani, Dylan Cutler, and Igor Bilogrevic. 2024. The Double Edged Sword: Identifying Authentication Pages and their Fingerprinting Behavior. In *World Wide Web Conference*. ACM, New York, NY, USA.
- [91] Signifyd. 2026. E-commerce fraud protection platform. <https://www.signifyd.com/> (accessed: September, 2026).
- [92] Konstantinos Solomos, Panagiotis Ilia, Soroush Karami, Nick Nikiforakis, and Jason Polakis. 2022. The Dangers of Human Touch: Fingerprinting Browser Extensions through User Actions. In *USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA.
- [93] Junhua Su and Alexandros Kapravelos. 2023. Automatic Discovery of Emerging Browser Fingerprinting Techniques. In *World Wide Web Conference*. ACM, New York, NY, USA.
- [94] Joshua Sunshine, Serge Egelman, Hazim Almuhtedi, Neha Atri, and Lorrie Faith Cranor. 2009. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In *USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA.
- [95] The Tor Project. 2026. The Tor Browser and Network. <https://www.torproject.org/> (accessed: September, 2026).
- [96] The Tor Project. 2026. Tor at the Heart: Firefox. <https://blog.torproject.org/tor-heart-firefox/> (accessed: September, 2026).
- [97] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. 2018. FP-Scanner: The Privacy Implications of Browser Fingerprint Inconsistencies. In *USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA.
- [98] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. 2018. FP-STALKER: Tracking Browser Fingerprint Evolutions. In *Symposium on Security and Privacy*. IEEE, Piscataway, NJ, USA.
- [99] Tanvi Vyas, Andrea Marchesini, and Christoph Kerschbaumer. 2017. Extending the Same Origin Policy with Origin Attributes. In *Conference on Information Systems Security and Privacy*. SciTePress, Setúbal, Portugal.
- [100] W3C Privacy Working Group. 2026. Mitigating Browser Fingerprinting in Web Specifications. <https://w3c.github.io/fingerprinting-guidance/> (accessed: September, 2026).
- [101] WebKit. 2023. Bug 260842: Add general quirk for canvas2d fingerprinting. https://bugs.webkit.org/show_bug.cgi?id=260842 (accessed: September, 2026).
- [102] WebKit. 2024. Private Browsing 2.0. <https://webkit.org/blog/15697/private-browsing-2-0/> (accessed: September, 2026).
- [103] WebKit. 2026. Tracking Prevention in WebKit. <https://webkit.org/tracking-prevention/> (accessed: September, 2026).
- [104] World Wide Web Consortium (W3C). 2026. Same-Origin Policy (SOP). https://www.w3.org/Security/wiki/Same-Origin_Policy (accessed: September, 2026).
- [105] Shuang Wu, Pengfei Sun, Yao Zhao, and Yinzhi Cao. 2023. Him of Many Faces: Characterizing Billion-scale Adversarial and Benign Browser Fingerprints on Commercial Websites. In *Network and Distributed System Security Symposium*. Internet Society, Reston, VA, USA.
- [106] Desheng Zhang, Jianhui Zhang, Youjun Bu, Bo Chen, Chongxin Sun, and Tianyu Wang. 2022. A Survey of Browser Fingerprint Research and Application. *Wireless Communications and Mobile Computing* (2022).

A Ethical Considerations

The benefits of this work depend on analyzing browser and system configurations from a large population. Although these data do not describe user behavior, they are sensitive because they could facilitate browser fingerprinting.

Mozilla collected the data under its established privacy and data collection policies [65], which emphasize transparency, accountability, and necessity. Collection was limited to approximately 25 million Firefox users, with individual characteristics submitted separately via OHTTP [68] to decouple client identity, including IP addresses, from request contents. Access was restricted to authorized Mozilla employees, and data were retained for at most 90 days. Users could opt out and have prior submissions deleted.

We further reduced the risk of misuse by constraining our analyses and disclosures. Rather than developing more powerful fingerprinting methods, we modeled existing strategies and released aggregate information about signals already collected by fingerprinting systems. These safeguards balance the research benefits against the privacy risks.

Fingerprint stability is also important in practice. However, reducing stability ultimately converges toward randomization, which we discuss in Section 5.2. We do not analyze it because we do not expect such analysis to yield actionable improvements beyond existing defenses.

B Generative AI Usage

Claude Code [5] assisted with developing data collection and analysis code, analyzing fingerprinting scripts and web compatibility, and generating figures and tables. All Firefox data collection code underwent Mozilla’s standard expert review [62] and was validated using several hundred thousand Firefox Beta submissions before deployment.

Fingerprinting scripts were examined through de-obfuscation and static analysis, controlled execution and payload capture, and Firefox JS Tracing of DOM API calls.

The original analysis code, developed without generative AI in June 2025, was later refactored with AI assistance to improve performance using multiprocessing and free-threaded Python. Outputs from both implementations were compared for consistency.

Generative AI also assisted with analyses such as incremental fingerprint construction and with generating figures and $\LaTeX$ tables. All values were verified against source data. AI was not used to generate citations, interpret results, or substantially author the paper beyond minor language corrections.

C Open Science

We provide our analysis code, selected privacy-preserving aggregate datasets, and the results of reverse engineering fingerprinting

scripts.³ These artifacts document the characteristics collected by the examined services and enable evaluation and reproduction of our methodology.

Raw telemetry cannot be released because of its sensitivity. Furthermore, the scale of the complete dataset, which covers millions

of users and hundreds of measurements, exceeds the practical limits of anonymous code-hosting platforms. We therefore release aggregate datasets designed to support independent validation and future research without exposing individual user configurations.

³https://github.com/tomrittervg/fingerprinting_paper_artifacts