Mitigating Browser Fingerprinting in Web Specifications

By Nick Doty

Published: 2025-09-25 · Archived: 2026-04-05 16:58:25 UTC

Jump to Table of Contents Collapse Sidebar

Abstract

Exposure of settings and characteristics of browsers can harm user privacy by allowing for browser fingerprinting. This document defines different types of fingerprinting, considers distinct levels of mitigation for the related privacy risks and provides guidance for Web specification authors on how to balance these concerns when designing new Web features.

Status of This Document

This section describes the status of this document at the time of its publication. A list of current W3C publications and the latest revision of this technical report can be found in the W3C standards and drafts index.

This document provide guidance to Web specification authors on mitigating the privacy impacts of browser fingerprinting.

The Privacy Working Group is collaborating with the Technical Architecture Group (TAG) on this guidance.

This document was published by the Privacy Working Group as a Group Note using the Note track.

This Group Note is endorsed by the Privacy Working Group, but is not endorsed by W3C itself nor its Members.

The W3C Patent Policy does not carry any licensing requirements or commitments on this document.

This document is governed by the 18 August 2025 W3C Process Document.

Table of Contents

In short, *browser fingerprinting* is the capability of a site to identify or re-identify a visiting user, user agent, or device via configuration settings or other observable characteristics.

A similar definition is provided by [*RFC6973*]. A more detailed list of types of fingerprinting is included below. This document does not attempt to catalog all features currently used or usable for browser fingerprinting; however, A. Research provides links to browser vendor pages and academic findings.

Browser fingerprinting can be used as a security measure (e.g. as means of authenticating the user). However, fingerprinting is also a threat to users' privacy on the Web. This document does not attempt to provide a single unifying definition of "privacy" or "personal data", but we highlight how browser fingerprinting might impact users' privacy. For example, browser fingerprinting can be used to:

- identify a user
- correlate a user’s browsing activity within and across sessions
- track users without transparency or control

The privacy implications associated with each use case are discussed below. Following from the practice of security threat model analysis, we note that there are distinct models of privacy threats for fingerprinting. Defenses against these threats differ, depending on the particular privacy implication and the threat model of the user.

There are many reasons why users might wish to remain anonymous or unidentified online, including: concerns about surveillance, personal physical safety, and concerns about discrimination against them based on what they read or write when using the Web. When a browser fingerprint is correlated with identifying information (like an email address, a recognized given and sur-name, or a government-issued identifier), an application or service provider may be able to identify an otherwise pseudonymous user. The adversary and consequences of this threat will vary by the particular user and use case, but can include nation-state intelligence agencies and threats of violence or imprisonment.

Browser fingerprinting raises privacy concerns even when offline identities are not implicated. Some users may be surprised or concerned that an online party can correlate multiple visits (on the same or different sites) to develop a profile or history of the user. This concern may be heightened because (see below) it may occur without the user's knowledge or consent and tools such as clearing cookies or using a VPN do not prevent further correlation.

Browser fingerprinting also allows for tracking across origins [*RFC6454*]: different sites may be able to combine information about a single user even where a cookie policy would block accessing of cookies between origins, because the fingerprint is relatively unique and the same for all origins.

In contrast to other mechanisms defined by Web standards for maintaining state (e.g. cookies), browser fingerprinting allows for collection of data about user activity without clear indications that such collection is happening. Transparency can be important for end users, to understand how ongoing collection is happening, but it also enables researchers, policymakers and others to document or regulate privacy-sensitive activity. Browser fingerprinting also allows for tracking of activity without clear or effective user controls: a browser fingerprint typically cannot be cleared or re-set. (See the finding on unsanctioned tracking [*TAG-UNSANCTIONED*].)

Advances in techniques for browser fingerprinting (see A. Research, below), particularly in active fingerprinting, suggest that complete elimination of the capability of browser fingerprinting by a determined adversary through solely technical means that are widely deployed is implausible. However, mitigations in our technical specifications are possible, as described below (6. Mitigations), and may achieve different levels of success (4. Feasibility).

Mitigations recommended here are simply mitigations, not solutions. Users of the Web cannot confidently rely on sites being completely unable to correlate traffic, especially when executing client-side code. A fingerprinting surface extends across all implemented Web features for a particular user agent, and even to other layers of the stack; for example, differences in TCP connections. For example, a user might employ an onion routing system such as Tor to limit network-level linkability, but still face the risk of correlating Web-based activity through browser fingerprinting. In order to mitigate these privacy risks as a whole, fingerprinting must be considered during the design and development of all specifications.

The TAG finding on Unsanctioned Web Tracking, including browser fingerprinting, includes description of the limitations of technical measures and encourages minimizing and documenting new fingerprinting surface [*TAG-UNSANCTIONED*]. The best practices below detail common actions that authors of specifications for Web features can take to mitigate the privacy impacts of browser fingerprinting. The Self-Review Questionnaire documents mitigations of privacy impacts in Web features more generally that may complement these practices [*security-privacy-questionnaire*].

- Best Practice 1: Avoid unnecessary or severe increases to fingerprinting surface, especially for passive fingerprinting.
- Best Practice 2: Narrow the scope and availability of a feature with fingerprinting surface to what is functionally necessary.
- Best Practice 3: Mark features that contribute to fingerprintability.
- Best Practice 4: Specify orderings and non-functional differences.
- Best Practice 5: Limit the fingerprinting surface to only the entropy necessary.
- Best Practice 6: Enable graceful degradation for privacy-conscious users or implementers.
- Best Practice 7: Design APIs to access only the entropy necessary.
- Best Practice 8: Require servers to advertise or opt in to access data.
- Best Practice 9: Avoid unnecessary new local state mechanisms.
- Best Practice 10: Highlight any local state mechanisms to enable simultaneous clearing.
- Best Practice 11: Avoid permanent or persistent state.

Passive fingerprinting is browser fingerprinting based on characteristics observable in the contents of Web requests, without the use of any code executed on the client.

Passive fingerprinting would trivially include cookies (often unique identifiers sent in HTTP requests), the set of HTTP request headers and the IP address and other network-level information. The User-Agent string [*RFC9110*], for example, is an HTTP request header that typically identifies the browser, renderer, version and operating system. For some populations, the User-Agent and IP address will often uniquely identify a particular user's browser [*NDSS-FINGERPRINTING*].

For *active fingerprinting*, we also consider techniques where a site runs JavaScript or other code on the local client to observe additional characteristics about the browser, user, device or other context. In recent years numerous techniques have ab(used) CSS features to perform fingerprinting on par with JavaScript.

Techniques for active fingerprinting might include accessing the window size, enumerating fonts or connected devices, evaluating performance characteristics, reading from device sensors, and rendering graphical patterns. Key to this distinction is that active fingerprinting takes place in a way that is potentially detectable on the client.

Note that in some types of active fingerprinting, characteristics are combined on the client to produce a fingerprint. In most cases; however, the characteristics are sent en masse to a server, which can combine them in unobservable ways. The latter mechanism may be detectable, but the efficacy of fingerprinting mitigation techniques is much harder to measure in this scenario.

Transient Event Correlation is a technique to associate separate simultaneous sessions on a device with one another using observations of events that occur near simultaneously on multiple origins [*EPHEMERAL-FINGERPRINTING*]. These events are typically fired as a result of a change in hardware or environment, such as when a device's posture changes or when the set of available media devices changes.

Transient event correlation is not typically a concern except in certain threat models - it is only useful when an attacker is unable to link two sessions via passive or active fingerprinting techniques, which would typically include considering the sessions' IP address. In uncommon situations, those techniques can fail, but event

correlation can still be used to link sessions between e.g. two entirely different browser applications or two tabs that are sent over different network connections.

Transient event correlation may be possible with complex CSS, but typically requires JavaScript and it can be done in a reactive manner where JavaScript merely observes events, or it can be done in a proactive manner by heavily utilizing resources such as the CPU or GPU that another origin can observe. This type of attack between cooperating origins is typically referred to as a "covert channel" and there have been many papers about them using different techniques, for example [*RENDERING-CONTENTION*].

Users, user agents and devices may also be re-identified by a site that first sets and later retrieves state stored by a user agent or device. This *cookie-like fingerprinting* allows re-identification of a user or inferences about a user in the same way that HTTP cookies allow state management for the stateless HTTP protocol [*RFC6265*].

Cookie-like fingerprinting can also circumvent user attempts to limit or clear cookies stored by the user agent, as demonstrated by the "evercookie" implementation [*EVERCOOKIE*]. Where state is maintained across user agents (as in the case of common plugins with local storage), across devices (as in the case of certain browser syncing mechanisms) or across software upgrades, cookie-like fingerprinting can allow re-identification of users, user agents or devices where active and passive fingerprinting might not. The Security and Privacy Self-Review Questionnaire also considers this threat in origin state that persists across browsing sessions [*security-privacy-questionnaire*].

There are different levels of success in mitigating browser fingerprinting:

Decreased fingerprinting surface
: Removing the source of entropy or available attributes that can be used for fingerprinting, or limiting the data to not be accessible without some form of privilege being granted explicitly or implicitly.

Increased anonymity set
: By standardization, convention or common implementation, increasing the commonality of particular configurations to decrease the likelihood of unique fingerprintability.

Detectable fingerprinting
: Making fingerprinting observable to others, so that the user agent might block it or researchers can determine that it's happening.

Clearable local state
: Helping users respond to fingerprinting by making state mechanisms clearable.

Research has shown feasible improvement in privacy protection in all of these areas. Collected data on Web users has shown mobile devices to have substantially larger anonymity sets than desktop browsers [*HIDING-CROWD*]. Research on forms of active fingerprinting has documented its use and demonstrated changes in use of those techniques as an apparent result of increased awareness [*WPM-MILLION*]. Respawning of cookies has continued, with an increasing variety of techniques, but awareness and technical responses to the issue has made the practice less widespread [*FLASHCOOKIES-2*].

This document works under the expectation that mitigations with different levels of success are feasible under different circumstances, for different threat models and against different types of fingerprinting. In general, active

fingerprinting may be made detectable; we can minimize increases to the surface of passive fingerprinting; and cookie-like mechanisms can be made clearable.

Some implementers and some users may be willing to accept reduced functionality or decreased performance in order to minimize browser fingerprinting. Documenting which features have fingerprinting risk eases the work of implementers building modes for these users; minimizing fingerprinting even in cases where common implementations will have easy active fingerprintability allows such users to reduce the functionality trade-offs necessary. Making browser fingerprinting more detectable also contributes to mitigations outside the standardization process; for example, though regulatory or policy means [*TAG-UNSANCTIONED*].

To mitigate browser fingerprinting in your specification:

1. identify features that can be used for browser fingerprinting;
2. evaluate the severity of the fingerprinting surface based on these five factors; and,
3. apply mitigations described in the best practices below (6. Mitigations), focused on limiting the severity of that fingerprinting surface.

The *fingerprinting surface* of a user agent is the set of observable characteristics that can be used in concert to identify a user, user agent or device or correlate its activity.

Data sources that may be used for browser fingerprinting include:

- user configuration (of the browser or operating system)
- device characteristics
- environmental characteristics *(e.g. sensor readings)*
- operating system characteristics
- user behavior
- browser characteristics

These data sources may be accessed directly for some features, but in many other cases they are inferred through some other observation. Timing channels, in particular, are commonly used to infer details of hardware (exactly how quickly different operations are completed may provide information on GPU capability, say), network information (via the latency or speed in loading a particular resource) or even user configuration (what items have been previously cached or what resources are not loaded). Consider the side effects of feature and how those side effects would allow inferences of any of these characteristics.

The Tor Browser design document [*TOR-DESIGN*] has more details on these sources and their relative priorities; this document adds environmental characteristics in that sensor readings or data access may distinguish a user, user agent or device by information about the environment (location, for example).

For each identified feature, consider the severity for the privacy impacts described above (1.2 Privacy impacts and threat models) based on the following factors:

entropy
: How distinguishing is this new surface? Consider both the possible variations and the likely distribution of values. Adding 1-bit of entropy is typically of less concern; 30-some bits of entropy would be enough to

uniquely identify every individual person. Different data sources may provide different distributions of variation; for example, even 1 bit of entropy can unique identify a user if they are the only one for whom it is true.

detectability

Will use of this feature for browser fingerprinting be observable to the user agent or likely to be discoverable by researchers? Because detectability is an important — and perhaps the most feasible — mitigation, increases to the surface for passive fingerprinting are of particular concern and should be avoided.

persistence

How long will the characteristics of this fingerprinting surface stay unchanged? Can users control or re-set these values to prevent long-lived identification? While short-lived characteristics may still enable unexpected correlation of activity (for example, between two browser profiles on the same device), persistent or permanent identifiers are particularly concerning for the lack of user control.

availability

Will this surface be available to the "drive-by Web" or only in certain contexts such as when a document is visible or where a user has interacted with the document or granted a particular permission? While browser fingerprinting is still something to mitigate in the permissioned context, the concern that a feature will end up used primarily for fingerprinting is reduced.

scope

Is this surface consistent across origins or only within a single origin? In general, characteristics or identifiers that are tied to a particular origin are of less concern and can be handled with the same tools as HTTP cookies.

While we do not recommend specific trade-offs, these factors can be used to weigh increases to that surface (6.1 Weighing increased fingerprinting surface) and suggest appropriate mitigations. Although each factor may suggest specific mitigations, in weighing whether to add fingerprinting surface they should be considered in concert. For example, access to a new set of characteristics about the user may be high entropy, but be of less concern because it has limited availability and is easily detectable. A cross-origin, drive-by-available, permanent, passive unique identifier is incompatible with our expectations for privacy on the Web.

In conducting this analysis, it may be tempting to dismiss certain fingerprinting surface in a specification because of a comparison to fingerprinting surface exposed by other parts of the Web platform or other layers of the stack. Be cautious about making such claims. First, while similar information may be available through other means, similar is not identical: information disclosures may not be exactly the same and fingerprintability is made even more effective by combining these distinct sources. Second, where identical entropy is present, other factors of severity or availability may differ and those factors are important for feasible mitigation. Third, the platform is neither monolithic nor static; not all other features are implemented in all cases and may change (or be removed) in the future. Fourth, circular dependencies are a danger when so many new features are under development; two specifications sometimes refer to one another in arguing that fingerprinting surface already exists. It is more useful to reviewers and implementers to consider the fingerprinting surface provided by the particular Web feature itself, with specific references where surface may be available through other features as well.

Web specification authors regularly attempt to strike a balance between new functionality and fingerprinting surface. For example, feature detection functionality allows for progressive enhancement with a small addition to fingerprinting surface; detailed enumerations of fonts or connected devices may provide a large fingerprinting surface with minimal functional support.

Authors and Working Groups determine the appropriate balance between these properties on a case-by-case basis, given their understanding of the functionality, its implementations and the severity of increased fingerprinting surface. However, given the distinct privacy impacts described above and in order to improve consistency across specifications, these practices provide some guidance:

Best Practice 1

: Avoid unnecessary or severe increases to fingerprinting surface, especially for passive fingerprinting.

Consider each of the severity factors described above and whether that functionality is necessary and whether comparable functionality is feasible with less severe increases to the fingerprinting surface.

In particular, unless a feature cannot reasonably be designed in any other way, increased passive fingerprintability should be avoided. Passive fingerprinting allows for easier and widely-available identification, without opportunities for external detection or control by users or third parties.

Best Practice 2

: Narrow the scope and availability of a feature with fingerprinting surface to what is functionally necessary.

What browsing contexts, resources and requests need access to a particular feature? Identifiers can often be scoped to have a different value in different origins. Some configuration may only be necessary in top-level browsing contexts.

If an event is to be fired in response to an environmental or hardware change, only fire that event when the `Window` 's associated document's visibility state is "`visible`", or in `Worker` s whose owner set includes such a Document. If background pages need to learn of the event when they're focused, also fire the event while updating the visibility state. Consider whether it should be restricted by an iframe sandbox.

Should access to this functionality be limited to where users have granted a particular permission? While excessive permissions can create confusion and fatigue, limiting highly granular data to situations where a user has already granted permission to access sensitive data widely mitigates the risk of that feature being used primarily for browser fingerprinting in "drive-by" contexts. For example, Media Capture and Streams [*mediacapture-streams*] limits access to attached microphone and camera device labels to when the user has granted permission to access a camera or microphone (while still allowing access to the number and configuration of attached cameras and microphones in all contexts, a noted increase in drive-by fingerprinting surface).

Some implementations may also limit the entropy of fingerprinting surface by not exposing different capabilities for different devices or installations of a user agent. Font lists, for example, can be limited to a list commonly available on all devices that run a particular browser or operating system (as implemented in Tor Browser, Firefox and Safari).

Best Practice 3

: Mark features that contribute to fingerprintability.

 Where a feature does contribute to the fingerprinting surface, indicate that impact, by explaining the effect (and any known implementer mitigations) and marking the relevant section with a fingerprinting icon, as this paragraph is.

The following code can be used to mark a paragraph with the fingerprint icon.

```
<img src="https://www.w3.org/Icons/fingerprint.png"
    class="fingerprint"
    alt="This feature may contribute to browser fingerprintability.">
```

Specifications can mitigate against fingerprintability through standardization; by defining a consistent behavior, conformant implementations won't have variations that can be used for browser fingerprinting.

Randomization of certain browser characteristics has been proposed as a way to combat browser fingerprinting. While this strategy may be pursued by some implementations, we expect in general it will be more effective for us to standardize or null values rather than setting a range over which they can vary. The Tor Browser design [*TOR-DESIGN*] provides more detailed information, but in short: it's difficult to measure how well randomization will work as a mitigation and it can be costly to implement in terms of usability (varying functionality or design in unwanted ways), processing (generating random numbers) and development (including the cost of introducing new security vulnerabilities). Standardization provides the benefit of an increased anonymity set for conformant browsers with the same configuration: that is, an individual can look the same as a larger group of people rather than trying to look like a number of different individuals.

Best Practice 4

: Specify orderings and non-functional differences.

To reduce unnecessary entropy, specify aspects of API return values and behavior that don't contribute to functional differences. For example, if the ordering of return values in a list has no semantic value, specify a particular ordering (alphabetical order by a defined algorithm, for example) so that incidental differences don't expose fingerprinting surface.

Even within a single implementation, variation can occur unexpectedly due to differences in processor architecture or operating system configuration. Access to a list of system fonts via Flash or Java plugins notably returned the list sorted not in a standard alphabetical order, but in an unspecified order specific to the system. This ordering added to the entropy available from that plugin in a way that provided no functional advantage.

Best Practice 5

: Limit the fingerprinting surface to only the entropy necessary.

Following the basic principle of data minimization [*RFC6973*], design your APIs such that a site can access only the entropy necessary for particular functionality. This can take the form of:

- **Clamping decimal points:** Instead of returning a high-precision floating-point value such as `2.735928`, consider clamping to fewer decimal places or returning a coarser value like `2.7` if that satisfies the functional requirements. For example, a battery API might return `0.8` (80%) instead of `0.81234567`.
- **Using an enumerated list of options:** Replace fine-grained numeric outputs with a constrained set of meaningful categories. For instance, instead of exposing a percentage of the battery remaining, return an enum like `"low"`, `"medium"`, `"high"`, or `"charging"`, which can guide performance decisions without leaking precise details.
- **Using a boolean:** In some cases, a yes/no answer may be sufficient and much less fingerprintable. For example, instead of returning detailed information about a user's battery, encode it as simply `Low Battery` or not.

If your specification exposes some fingerprinting surface (whether it's active or passive), some implementers (e.g. Tor Browser) are going to be compelled to disable those features for certain privacy-conscious users.

Best Practice 6

: Enable graceful degradation for privacy-conscious users or implementers.

Following the principle of progressive enhancement, and to avoid further divergence (which might itself expose variation in users), consider whether some functionality in your specification is still possible if fingerprinting surface features are disabled.

Explicit hooks or API flags may be used so that browser extensions or certain user agents can easily disable specific features or aspects of a feature. For example, an event defined in a feature might specify that certain properties that describe the hardware device that triggered it may be blank.

Standardization does *not* need to attempt to hide all differences between different browsers (e.g. Edge and Chrome); implemented functionality and behavior differences will always exist between different implementations. For that reason, removing `User-Agent` headers altogether is not a goal. However, variation in the `User-Agent` string that reveals additional information about the user or device has been shown to provide substantial fingerprinting surface [*BEAUTY-BEAST*].

Where a client-side API provides some fingerprinting surface, authors can still assist User Agents via detectability. If client-side fingerprinting activity is to some extent distinguishable from functional use of APIs, user agent implementations may have an opportunity to prevent ongoing fingerprinting or make it observable to users and external researchers (including academics or relevant regulators) who may be able to detect and investigate the use of fingerprinting.

Best Practice 7

: Design APIs to access only the entropy necessary.

Following the basic principle of data minimization [*RFC6973*], design your APIs such that by default, they expose only the entropy necessary for particular functionality, and requires more explicit parameters to receive more expansive data.

Making an API asynchronous and returning a promise gives User Agents the option, but not the requirement, to interpose a permission dialog in whatever circumstances they may deem desirable.

Authors might design an API to allow for querying of a particular value, rather than returning an enumeration of all values. User agents and researchers can then more easily distinguish between sites that query for one or two particular values (gaining minimal entropy) and those that query for all values (more likely attempting to fingerprint the browser); or implementations can cap the number of different values.

For more information, see:

- Device API Privacy Requirements [*dap-privacy-reqs*], DAP Working Group Note, June 2010.
- Data Minimization in Web APIs [*TAG-MINIMIZATION*], W3C TAG, September 2011.
- Generic Sensor API: Security and privacy considerations [*generic-sensor*], March 2018.
- The leaking battery: A privacy analysis of the HTML5 Battery Status API [*LEAKING-BATTERY*], 2015.

Related, detectability is improved even with data sent in HTTP headers (what we would typically consider passive fingerprinting) if sites are required to request access (or "opt in") to information before it's sent.

Best Practice 8

: Require servers to advertise or opt in to access data.

Even for data sent in HTTP request headers, requiring servers to advertise use of particular data, publicly document a policy, or "opt in" before clients send configuration data provides the possibility of detection by user agents or researchers.

For example, Client Hints [*client-hints-infrastructure*] proposes an `Accept-CH` response header for services to indicate that specific hints can be used for content negotiation, rather than all supporting clients sending all hints in all requests.

Note

This is a relatively new approach; we're still evaluating whether this provides meaningful and useful detectability.

Implementers can facilitate detectability by providing or enabling instrumentation so that users or third parties are able to calculate when fingerprinting surface is being accessed. Of particular importance for instrumentation are: access to all the different sources of fingerprinting surface; identification of the originating script; avoiding exposure that instrumentation is taking place. Beyond the minimization practice described above, these are largely implementation-specific (rather than Web specification) features.

Features which enable storage of data on the client and functionality for client- or server-side querying of that data can increase the ease of cookie-like fingerprinting. Storage can vary between large amounts of data (for example,

the Web Storage API) or just a binary flag (has or has not provided a certain permission; has or has not cached a single resource).

Best Practice 9

: Avoid unnecessary new local state mechanisms.

If functionality does not require maintaining client-side state in a way that is subsequently queryable (or otherwise observable), avoid creating a new cookie-like feature. Can the functionality be accomplished with existing HTTP cookies or an existing JavaScript local storage API?

For example, the Flash plugin's Local Shared Objects (LSOs) used to be abused to duplicate and re-spawn HTTP cookies cleared by the user [*FLASHCOOKIES*], and the single bit that indicates if the Strict-TransportSecurity Header has been set for a domain has been abused in the same way [*HSTS-SUPERCOOKIE*].

Where features do require setting and retrieving local state, there are ways to mitigate the privacy impacts related to unexpected cookie-like behavior; in particular, you can help implementers prevent "permanent", "zombie", "super" or "evercookies".

Best Practice 10

: Highlight any local state mechanisms to enable simultaneous clearing.

Clearly note where state is being maintained and could be queried and provide guidance to implementers on enabling simultaneous deletion of local state for users. Such functionality can mitigate the threat of "evercookies" because the presence of state in one such storage mechanism can't be used to persist and re-create an identifier.

Permanent or persistent data (including any identifiers) are of particular risk because they undermine the ability for a user to clear or re-set the state of their device or to maintain different identities.

Best Practice 11

: Avoid permanent or persistent state.

Permanent identifiers or other state (for example, identifiers or keys set in hardware) should typically not be used. Where necessary, access to such identifiers would require user permission and limitation to a particular origin. However even heavy-weight mitigations are imperfect: explaining the implications of such permission to users may be difficult and server-side collusion between origins is typically impossible to detect. As a result, your design should not rely on saving and later querying data on the client and expecting it to persist beyond a user clearing cookies or other local state. That is, you should not expect any local state information to be permanent or to persist longer than other local state.

Though not strictly browser fingerprinting, there are other privacy concerns regarding user tracking for features that provide local storage of data. Mitigations suggested in the Web Storage API specification include: safe-listing, block-listing, expiration and secure deletion [HTML#user-tracking].

Expressions of, and compliance with, a Do Not Track signal does not inhibit the capability of browser fingerprinting, but may mitigate some user concerns about fingerprinting, specifically around tracking as defined in those specifications [*TRACKING-DNT*] [*TRACKING-COMPLIANCE*] and as implemented by services that comply with those user preferences. That is, DNT can mitigate concerns with cooperative sites.

The use of DNT in this way typically does not require changes to other functional specifications. If your specification expects a particular behavior upon receiving a particular DNT signal, indicate that with a reference to [*TRACKING-DNT*]. If your specification introduces a new communication channel that could be used for tracking, you might wish to define how a DNT signal should be communicated.

Some browser developers maintain pages on browser fingerprinting, including: potential mitigations or modifications necessary to decrease the surface of that browser engine; different vectors that can be used for fingerprinting; potential future work. These are not cheery, optimistic documents.

- The Chromium Projects: Technical analysis of client identification mechanisms
- WebKit Wiki: Fingerprinting
- Mozilla Wiki: Fingerprinting
- The Design and Implementation of the Tor Browser: Cross-Origin Fingerprinting Unlinkability

What are the key papers to read here, historically or to give the latest on fingerprinting techniques? What are some areas of open research that might be relevant?

- Eckersley, Peter. "How unique is your web browser?" *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2010.
- Mowery, Keaton, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. “Fingerprinting Information in JavaScript Implementations.” In *Web 2.0 Security and Privacy*, 2011.
- Yen, Ting-Fang, et al. "Host fingerprinting and tracking on the web: Privacy and security implications." *Proceedings of NDSS*. 2012. [*NDSS-FINGERPRINTING*]
- Mowery, Keaton, and Hovav Shacham. "Pixel perfect: Fingerprinting canvas in HTML5." *Web 2.0 Security and Privacy*, 2012.
- Mattioli, Dana. "On Orbitz, Mac Users Steered to Pricier Hotels". *Wall Street Journal*, August 23, 2012.
- Gunes Acar et al. "FPDetective: dusting the web for fingerprinters." In *CCS '13*.
- Nikiforakis, Nick, et al. "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting." *IEEE Symposium on Security and Privacy (S&P 2013)*, 2013.
- G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, C. Diaz. "The Web never forgets: Persistent tracking mechanisms in the wild." In *Proceedings of CCS 2014*, Nov. 2014.
- Steven Englehardt, Arvind Narayanan. "Online tracking: A 1-million-site measurement and analysis." May 2016. [*WPM-MILLION*]
- Pierre Laperdrix, Walter Rudametkin, Benoit Baudry. "Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints." *IEEE Symposium on Security and Privacy (S&P 2016)*, May 2016.
- "Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale." *WWW2018 - TheWebConf 2018: 27th International World Wide Web Conference*, April 2018. [*HIDING-CROWD*]

A non-exhaustive list of sites that allow the visitor to test their configuration for fingerprintability.

- amiunique.org (INRIA)
- panopticlick.eff.org (EFF)
- BrowserSPY.dk
- pet-portal cross-browser fingerprinting test
- p0f v3 (purely passive fingerprinting)

Many thanks to Robin Berjon for ReSpec and to Tobie Langel for Github advice; to the Privacy Interest Group and the Technical Architecture Group for review; to the Tor Browser designers for references and recommendations; and to Christine Runnegar for contributions.

[BEAUTY-BEAST]
Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints. Pierre Laperdrix; Walter Rudametkin; Benoit Baudry. IEEE Symposium on Security and Privacy (S&P 2016). May 2016. URL: https://inria.hal.science/hal-01285470v2/

[client-hints-infrastructure]
Client Hints Infrastructure. W3C. Draft Community Group Report. URL: https://wicg.github.io/client-hints-infrastructure/

[dap-privacy-reqs]
Device API Privacy Requirements. Alissa Cooper; Frederick Hirsch; John Morris. W3C. 29 June 2010. W3C Working Group Note. URL: https://www.w3.org/TR/dap-privacy-reqs/

[device-posture]
Device Posture API. Kenneth Christiansen; Alexis Menard; Diego Gonzalez-Zuniga. W3C. 26 November 2024. W3C Candidate Recommendation. URL: https://www.w3.org/TR/device-posture/

[dom]
DOM Standard. Anne van Kesteren. WHATWG. Living Standard. URL: https://dom.spec.whatwg.org/

[EPHEMERAL-FINGERPRINTING]
Ephemeral Fingerprinting On The Web. Asanka Herath. 1 September 2020. URL: https://github.com/asankah/ephemeral-fingerprinting

[EVERCOOKIE]
evercookie - virtually irrevocable persistent cookies. Samy Kamkar. September 2010. URL: https://samy.pl/evercookie/

[FLASHCOOKIES]
Flash Cookies and Privacy. Ashkan Soltani; Shannon Canty; Quentin Mayo; Lauren Thomas; Chris Jay Hoofnagle. 10 August 2009. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1446862

[FLASHCOOKIES-2]
Flash cookies and privacy II: Now with HTML5 and ETag respawning. Mika Ayenson; Dietrich Wambach; Ashkan Soltani; Nathan Good; Chris Hoofnagle. URL: https://ptolemy.berkeley.edu/projects/truststc/education/reu/11/Posters/AyensonMWambachDpaper.pdf

[generic-sensor]
Generic Sensor API. Rick Waldron. W3C. 22 February 2024. CRD. URL: https://www.w3.org/TR/generic-sensor/

[HIDING-CROWD]

Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. Alejandro Gómez-Boix; Pierre Laperdrix; Benoit Baudry. WWW2018 - TheWebConf2018: 27th International World Wide Web Conference. April 2018. URL: https://inria.hal.science/hal-01718234v2

[HSTS-SUPERCOOKIE]

HSTS Super Cookie. Ben Friedland. Jan 2016. URL: https://github.com/ben174/hsts-cookie

[HTML]

HTML Standard. Anne van Kesteren; Domenic Denicola; Dominic Farolino; Ian Hickson; Philip Jägenstedt; Simon Pieters. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[LEAKING-BATTERY]

The leaking battery: A privacy analysis of the HTML5 Battery Status API. Łukasz Olejnik; Gunes Acar; Claude Castelluccia; Claudia Diaz. 2015. URL: https://eprint.iacr.org/2015/616.pdf

[mediacapture-streams]

Media Capture and Streams. Cullen Jennings; Jan-Ivar Bruaroey; Henrik Boström; youenn fablet. W3C. 25 September 2025. CRD. URL: https://www.w3.org/TR/mediacapture-streams/

[NDSS-FINGERPRINTING]

Host Fingerprinting and Tracking on the Web: Privacy and Security Implications. Ting-Fang Yen; Yinglian Xie; Fang Yu; Roger Peng Yu; Martin Abadi. In Proceedings of the Network and Distributed System Security Symposium (NDSS). February 2012. URL: https://www.microsoft.com/en-us/research/publication/host-fingerprinting-and-tracking-on-the-webprivacy-and-security-implications/

[RENDERING-CONTENTION]

Rendering Contention Channel Made Practical in Web Browsers. Shujiang Wu; Jianjia Yu; Min Yang; Yinzhi Cao. August 2022. URL: https://www.usenix.org/system/files/sec22summer_wu.pdf

[RFC6265]

HTTP State Management Mechanism. A. Barth. IETF. April 2011. Proposed Standard. URL: https://httpwg.org/specs/rfc6265.html

[RFC6454]

The Web Origin Concept. A. Barth. IETF. December 2011. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc6454

[RFC6973]

Privacy Considerations for Internet Protocols. A. Cooper; H. Tschofenig; B. Aboba; J. Peterson; J. Morris; M. Hansen; R. Smith. IETF. July 2013. Informational. URL: https://www.rfc-editor.org/rfc/rfc6973

[RFC9110]

HTTP Semantics. R. Fielding, Ed.; M. Nottingham, Ed.; J. Reschke, Ed. IETF. June 2022. Internet Standard. URL: https://httpwg.org/specs/rfc9110.html

[security-privacy-questionnaire]

Self-Review Questionnaire: Security and Privacy. Theresa O'Connor; Peter Snyder; Simone Onofri. W3C. 18 April 2025. W3C Working Group Note. URL: https://www.w3.org/TR/security-privacy-questionnaire/

[TAG-MINIMIZATION]

Data Minimization in Web APIs. Daniel Appelquist. W3C Technical Architecture Group. 12 September 2011. URL: https://www.w3.org/2001/tag/doc/APIMinimization

[TAG-UNSANCTIONED]

Unsanctioned Web Tracking. Mark Nottingham. W3C Technical Architecture Group. 17 July 2015. URL: https://w3ctag.github.io/unsanctioned-tracking/

[TOR-DESIGN]

The Design and Implementation of the Tor Browser. Mike Perry; Erinn Clark; Steven Murdoch; Georg Koppen. 15 June 2018. URL: https://spec.torproject.org/torbrowser-design

[TRACKING-COMPLIANCE]

Tracking Compliance and Scope. Nick Doty; Heather West; Justin Brookman; Sean Harvey; Erica Newland. W3C. 22 January 2019. W3C Working Group Note. URL: https://www.w3.org/TR/tracking-compliance/

[TRACKING-DNT]

Tracking Preference Expression (DNT). Roy Fielding; David Singer. W3C. 17 January 2019. W3C Working Group Note. URL: https://www.w3.org/TR/tracking-dnt/

[WPM-MILLION]

Online tracking: A 1-million-site measurement and analysis. Steven Englehardt; Arvind Narayanan. May 2016. URL: https://webtransparency.cs.princeton.edu/webcensus/
