This is the abstract for your specification.

What is fingerprinting?

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 more detailed list of types of fingerprinting is included below. A similar definition is provided by [[RFC6973]].

Privacy impacts and threat models

Browser fingerprinting can be used as a security measure (e.g. as means of authenticating the user). However, fingerprinting is also a potential 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
- track and correlate a user’s browsing activity within and across sessions
- collect information from which inferences can be drawn about a user

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.

Identify a user

There are many reasons why users might wish to remain anonymous or unidentified online, including: concerns about surveillance, personal physical safety, 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 a real name), an application or service provider may be able to identify an otherwise pseudonymous user.

Users concerned about physical safety from, for example, a governmental adversary might employ onion routing systems such as Tor to limit network-level linkability but still face the danger of browser fingerprinting to correlate their Web-based activity.

Unexpected correlation of browsing activity

Fingerprinting provides privacy concerns even when real-world 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 it may occur without the user's knowledge or consent and tools such as clearing cookies do not prevent or “re-set” correlation done via browser fingerprinting.

Inferences about a user

The observable characteristics used for browser fingerprinting can themselves reveal information from which inferences can be drawn about a user. For example, the OS version and CPU information might be used to draw inferences about a user’s purchasing power or proclivity (for example). Users may consider this an unwelcome intrusion into their privacy even if they remain unidentified. Additionally, decisions might be made based on these inferences (e.g. which offers to display and at what price) that users perceive as discriminatory and an instance of being singled out and treated differently. This intrusion is compounded when browser fingerprints are correlated with user credentials, purchasing histories and other information about the user (e.g. cross-site browsing histories).

Editorial question: Is this in scope for browser fingerprinting? Or is this just a general privacy concern about leakage of information via observable characteristics?

Fingerprinting mitigation levels of success

There are also different levels of success in addressing browser fingerprinting:

Decreased fingerprinting surface
: Removing the source of entropy or accessible attributes that can be used for fingerprinting.

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 (in particular, client-side) fingerprinting observable to the user agent or some other party, so that the user agent might block it or a crawler can determine that it's happening.

Types of fingerprinting

Passive

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

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

Active

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. Techniques for active fingerprinting might include accessing the window size, enumerating fonts or plug-ins, evaluating performance characteristics, or rendering graphical patterns.

Cookie-like (setting/retrieving local state)

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.

Mitigations

Weighing increased 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. Web specification authors regularly attempt to strike a balance between new functionality and fingerprinting surface. For example, feature detection functionality allows for progressive enhancement, but detailed granularity in feature detection increases the fingerprinting surface of a user agent. (An attacker can test for many features on every visitor's browser and might uniquely identify a user by the exact set of features enabled.)

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

- Specifications MUST NOT increase the surface for passive fingerprinting unless a feature cannot reasonably be designed in another way.
- Specifications SHOULD NOT increase the surface for active fingerprinting if comparable functionality could be accomplished without increasing the surface.

The difference between these requirements recognizes that passive fingerprinting surface has lesser options for mitigation (lacking external detectability and client-side preventability) and greater feasibility for reduction.

A standardized profile?

Detectability

Where a client-side API provides some fingerprinting surface, authors can still mitigate the privacy concerns 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) may be able to detect and investigate the use of fingerprinting.

Following the basic principle of data minimization, authors SHOULD design APIs such that a site can access (and does access by default) only the entropy necessary for particular functionality.

TODO: An example would probably be very useful here.

Clearing all local state

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). If functionality does not require maintaining client-side state in a way that is subsequently queryable (or otherwise observable), a specification SHOULD NOT include this cookie-like feature.

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, specification authors SHOULD clearly note where state is being maintained and could be queried and SHOULD 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. Authors SHOULD NOT design functionality whose utility depends on saving and later querying data on the client beyond a user's clearing their 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: white-listing, black-listing, expiration and secure deletion [[WEBSTORAGE-user-tracking]].

Do Not Track: a cooperative approach

Expressions of, and compliance with, a Do Not Track signal may not prevent or inhibit 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 engaged in fingerprinting that voluntarily comply with those user preferences. This mitigation is included here for completeness; DNT standardization and adoption is ongoing.

Research

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. panopticlick.eff.org
- WebKit Wiki: Fingerprinting
- Yen, Ting-Fang, et al. "Host fingerprinting and tracking on the web:

Privacy and security implications." *Proceedings of NDSS*. 2012.

- Mowery, Keaton, and Hovav Shacham. "Pixel perfect: Fingerprinting canvas in HTML5." *Proceedings of W2SP* (2012).
- Mowery, Keaton, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. "Fingerprinting Information in JavaScript Implementations." In Web 2.0 Security and Privacy, 2011.
- 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 Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.
- Nikiforakis, Nick, et al. "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting." IEEE Symposium on Security and Privacy. 2013.

Acknowledgements

Many thanks to Robin Berjon for ReSpec and to Tobie Langel for Github advice; to the Privacy Interest Group for review; and to Christine Runnegar for contributions.