Browser Fingerprinting: An Introduction and the Challenges Ahead

by [gk](https://blog.torproject.org/users/gk) | September 04, 2019



*Guest post by*[*Pierre Laperdrix*](https://plaperdr.github.io/)

In the past few years, a technique called browser fingerprinting has received a lot of attention because of the risks it can pose to privacy. What is it? How is it used? What is Tor Browser doing against it? In this blog post, I’m here to answer these questions. Let’s get started!

**What Is Browser Fingerprinting?**

Since the very beginning of the web, browsers did not behave the exact same way when presented with the same webpage: some elements could be rendered improperly, they could be positioned at the wrong location or the overall page could simply be broken with an incorrect HTML tag. To remedy this problem, browsers started including the “user agent” header. This informed the server on the browser being used so that it could send the device a page that was optimized for it. In the nineties, this started the infamous era of the “Best on IE” or “Optimized for Netscape.”

In 2019, the user-agent header is still here but a lot has changed since then. The web as a platform is a lot richer in terms of features. We can listen to music, watch videos, have real-time communications or immerse ourselves in virtual reality. We can also use a very wide variety of devices from tablets, smartphones or laptops to connect to it. **To offer an experience that is optimized for every device and usage, there is still a need today to share configuration information with the server.** “Here is my timezone so that I can know the exact start time of the NBA finals. Here is my platform so that the website can give me the right version of the software I’m interested in. Here is the model of my graphic card so that the game I’m playing in my browser can chose graphic settings for me.”

All of this makes the web a truly beautiful platform as it enables us to have a comfortable experience browsing it. However, all that information that is freely available to optimize the user experience can be collected **to build a browser fingerprint**.



*Figure 1: Example of a browser fingerprint from a Linux laptop running Firefox 67*

In Figure 1, you can see a browser fingerprint taken from my Linux laptop. The information in the fingerprint was collected **via HTTP with the received HTTP headers and via JavaScript by running a small script.** The “user-agent” indicates that the user was using Firefox version 67 on the Fedora Linux distribution. The “content-language” header indicates that the user wants to receive her page in English with the “US” variant. The “-120” for the timezone refers to the GMT+2 time. Finally, the WebGL renderer gives information on the CPU of the device. Here, the laptop is using an Intel CPU with a Kaby Lake Refresh microarchitecture.

This example is a glimpse of what can be collected in a fingerprint and the exact list is evolving over time as new APIs are introduced and others are modified. If you want to see your own browser fingerprint, I invite you to visit [AmIUnique.org](https://amiunique.org/). It is a website that I launched in 2014 to study browser fingerprinting. With the data that we collected from more than a million visitors, we got invaluable insight into its inner-workings and we pushed the research in the domain forward.

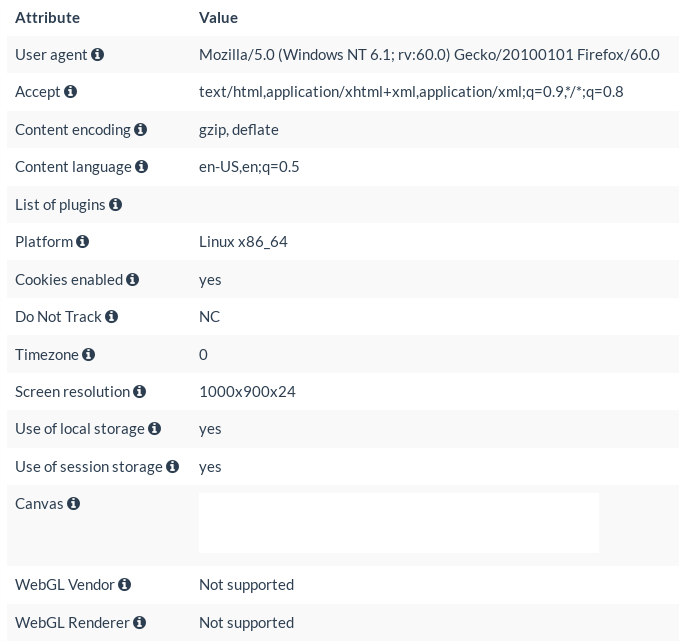
**What Makes Fingerprinting A Threat To Online Privacy?**

It is pretty simple. First, there is **no need to ask for permissions** to collect all this information. Any script running in your browser can silently build a fingerprint of your device without you even knowing about it. Second, **if one attribute of your browser fingerprint is unique or if the combination of several attributes is unique, your device can be identified and tracked online**. In that case, no need for a cookie with an ID in it, the fingerprint is enough. Hopefully, as we will see in the next sections, a lot of progress have been made to prevent users from having unique values in their fingerprint and thus, avoid tracking.

**Tor + Fingerprinting**

Tor Browser was the very first browser to address the problems posed by fingerprinting as soon as 2007, even before the term “browser fingerprinting” was coined. In March 2007, the changelog for the Tor button indicated the inclusion of [Javascript hooking to mask timezone for Date Object](https://gitweb.torproject.org/torbutton.git/tree/src/CHANGELOG).

In the end, the approach chosen by Tor developers is simple: **all Tor users should have the exact same fingerprint**. No matter what device or operating system you are using, your browser fingerprint should be the same as any device running Tor Browser (more details can be found in the [Tor design document](https://2019.www.torproject.org/projects/torbrowser/design/#fingerprinting-linkability)).



*Figure 2: Example of a browser fingerprint from a Linux laptop running Tor Browser 8.5.3*

In Figure 2, you can find the fingerprint of my Linux machine running version 8.5.3 of the Tor Browser.

Comparing with the one from Firefox, we can see notable differences. First, no mater on which OS Tor Browser is running, you will always have the following user-agent:

Mozilla/5.0 (Windows NT 6.1; rv:60.0) Gecko/20100101 Firefox/60.0

As Windows is the most widespread OS on the planet, TBB masks the underlying OS by claiming it is running on a Windows machine. Firefox 60 refers to the ESR version on which TBB is based on.

Other visible changes include the platform, the timezone, and the screen resolution.

Also, you may have wondered why the following message appears when you maximize the browser window (see Figure 3): “Maximizing Tor Browser can allow websites to determine your monitor size, which can be used to track you. We recommend that you leave Tor Browser windows in their original default size.”

This is because of fingerprinting. Since users have different screen sizes, one way of making sure that no differences are observable is to have everyone use the same window size. If you maximize the browser window, you may end up as being the only one using Tor Browser at this specific resolution and so comes a higher identification risk online.

Tor Browser warning 1Tor Browser warning 2 

*Figure 3: Warning from the Tor Browser when maximizing the browser window*

Under the hood, a lot more modifications have been performed to reduce differences between users. [Default fallback fonts](https://trac.torproject.org/projects/tor/ticket/18097) have been introduced to mitigate font and canvas fingerprinting. WebGL and the Canvas API are blocked by default to prevent stealthy collection of renderings. [Functions like performance.now](https://trac.torproject.org/projects/tor/ticket/1517) have also been modified to prevent timing operations in the browser that can be used for micro-architectural attacks. If you want to see all the efforts made by the Tor team behind the scenes, you can take a look at the [fingerprinting](https://trac.torproject.org/projects/tor/query?keywords=~tbb-fingerprinting&order=priority) tag in the bug tracker.  A lot of work is being done to make this a reality. As part of the effort to reduce fingerprinting, I also developed a fingerprinting website called [FP Central](https://fpcentral.tbb.torproject.org/) to help Tor developers find fingerprint regressions between different Tor builds.

Finally, more and more modifications present in TBB are making their way into Firefox as part of the [Tor Uplift program](https://wiki.mozilla.org/Security/Tor_Uplift).

**Where We Are**

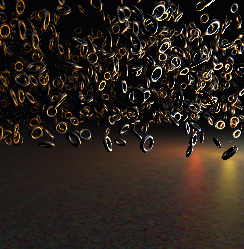
Over the past few years, research on browser fingerprinting has substantially increased and covers many aspects of the domain. Here, we will have a quick overview of the research done in academia and how fingerprinting is used in the industry.

**Academic Research**

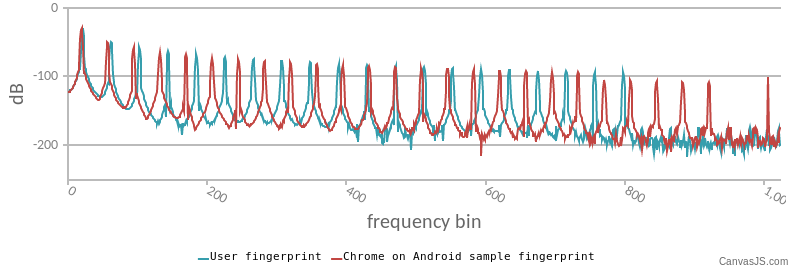
1. Tracking with fingerprinting is a reality but it cannot replace different tracking schemes based on identifiers. Different studies have been published over the years trying to assess the diversity of modern devices connected on the web [1,2]. One study that I was part of in 2018 [3] surprised us as it showed that tracking at a very large scale may not be feasible with low percentage of uniqueness. Anyhow, the one clear takeaway from these studies is the following: even though some browser vendors are working very hard to reduce as much as possible the differences between devices, it is not a perfect process. If you have that one value in your browser fingerprint (or a combination) that nobody has, you can still be tracked and that is why you should be careful about fingerprinting. There is no strong guarantee today that your device is identical to another one present on the Internet.

2. As the web is getting richer, new APIs make their way into browsers and new fingerprinting techniques are discovered. The most recent techniques include WebGL [4,5], Web Audio [6] and extension fingerprinting [7,8]. To provide protection for users, it is important to keep a close watch on any new advances in the field to fix any issues that may arise.

One lesson learned from the past concerns the BatteryStatus API. It was added to provide information about the state of the battery to developers so that they could develop energy-efficient applications. Drafted as early as 2011, it was not until 2015 that researchers discovered that this API could be misused to create a short-term identifier [9,10]. In the end, this was a reminder that we have to be very thoughtful when introducing a new API in a browser. A deep analysis must be conducted to remove or mitigate as much as possible hidden fingerprinting vectors before they are deployed to end-users. To provide guidance for Web specification authors, the W3C has written [a document](https://www.w3.org/TR/fingerprinting-guidance/) on how best to design an API while considering fingerprinting risks



*Figure 4: Example of a WebGL rendering as tested on*[*http://uniquemachine.org/*](http://uniquemachine.org/)



*Figure 5: Example of an audio fingerprint as tested on*[*https://audiofingerprint.openwpm.com/*](https://audiofingerprint.openwpm.com/)

3. Today, there is no ultimate solution to fix browser fingerprinting. As its origin is rooted in the beginning of the internet, there is no single patch that can fix it for good. And as such, designing defenses is hard. A lot of approaches have been tried and evaluated over the years with each their strengths and weakness. Examples include blocking attributes, introducing noise, modifying values, or increasing fingerprint diversity. However, one important observation that has been made is that sometimes having no specific defense is better than having one. Some solutions, because of the way they were designed or coded, remove some fingerprinting vectors but introduce some artifacts or inconsistencies in the collected fingerprints.

For example, imagine a browser extension that changes the value of fingerprints before they are sent.

Everything works perfectly except the fact that the developer forgot to override the navigator.platform value. Because of this, the user-agent may say that the browser is running on Windows whereas the platform still indicates it is on a Linux system. This creates a fingerprint that is not supposed to exist in reality and, as such, make the user more visible online. It is what Eckersley [1] called the “Paradox of Fingerprintable Privacy Enhancing Technologies.” By wanting to increase online privacy, you install extensions that in the end make you even more visible than before.

**Industry**

1. To identify websites who use browser fingerprinting, one can simply turn to privacy policies. Most of the time, you will never see the term “fingerprinting” in it but sentences along the lines of “we collect device-specific information to improve our services.” The exact list of collected attributes is often imprecise and the exact use of that information can be very opaque ranging from analytics to security to marketing or advertising.

Another way of identifying websites using fingerprinting is to look directly at the scripts that run in the browser. The problem here is that it can be challenging to differentiate a benign script that is here to improve the user experience from a fingerprinting one. For example, if a site accesses your screen resolution, is it to adjust the size of HTML elements to your screen or is it the first step in building a fingerprint of your device? The line between the two can be very thin and identifying fingerprinting scripts with precision is still a subject that has not been properly studied yet.

2. One use of fingerprinting that is lesser known is for bot detection. To secure their websites, some companies rely on online services to assess the risk associated with external connections. In the past, most decisions to block or accept a connection was purely based on IP reputation. Now, browser fingerprinting is used to go further to detect tampering or identify signs of automation. Examples of companies that use fingerprinting for this purpose include ThreatMetrix, Distil Networks, MaxMind, PerimeterX, and DataDome.

3. On the defensive side, more and more browser vendors are adding fingerprinting protection directly in their browser. As mentioned previously in this blog post, Tor and Firefox are at the forefront of these efforts by limiting passive fingerprinting and blocking active fingerprinting vectors.

Since its initial release, the Brave browser also includes [built-in protection](https://github.com/brave/browser-laptop/wiki/Fingerprinting-Protection-Mode) against it.

Apple made changes to Safari in 2018 [to limit it](https://gizmodo.com/apple-declares-war-on-browser-fingerprinting-the-sneak-1826549108) and Google announced in May 2019 its intention to [do the same for Chrome](https://blog.chromium.org/2019/05/improving-privacy-and-security-on-web.html).

**Conclusion: What Lies Ahead**

Browser fingerprinting has grown a lot over the past few years. As this technique is closely tied to browser technology, its evolution is hard to predict but its usage is currently shifting. What we once thought could replace cookies as the ultimate tracking technique is simply not true. Recent studies show that, while it can be used to identify some devices, it cannot track the mass of users browsing the web daily. Instead, fingerprinting is now being used to improve security. More and more companies find value in it to go beyond traditional IP analysis. They analyze the content of fingerprints to identify bots or attackers and block unwanted access to online systems and accounts.

One big challenge surrounding fingerprinting that is yet to be solved is around the regulation of its usage. For cookies, it is simple to check if a cookie was set by a specific website. Anyone can go in the browser preferences and check the cookie storage. For fingerprinting, it is a different story. There is no straightforward way to detect fingerprinting attempts and there are no mechanisms in a browser to completely block its usage. From a legal perspective, this is very problematic as regulators will need to find new ways to cooperate with companies to make sure that the privacy of users is respected.

Finally, to finish this post, is fingerprinting here to stay? In the near future at least, yes. This technique is so rooted in mechanisms that exist since the beginning of the web that it is very complex to get rid of it. It is one thing to remove differences between users as much as possible. It is a completely different one to remove device-specific information altogether. Only time will tell how fingerprinting will change in the coming years but its evolution is something to watch closely as the frantic pace of web development will surely bring a lot of surprises along the way.

Thanks a lot for reading this post all the way through! If you want to dive even deeper in the subject, I invite you to read [the survey](https://arxiv.org/pdf/1905.01051) [11] on the topic that we recently made available online. If by any chance you find any new fingerprinting vectors in Tor Browser, I strongly suggest that you open a ticket on [the Tor bug tracker](https://trac.torproject.org/projects/tor) to help the fantastic efforts made by the Tor dev team to better protect users’ online anonymity!

\*\*Pierre Laperdrix\*\*    
<https://plaperdr.github.io/>  
Twitter: <https://twitter.com/RockPartridge>

**References**

**[1]** P. Eckersley. “How unique is your web browser?”. In International Symposium on Privacy Enhancing Technologies Symposium (PETS’10). [[PDF]](<https://panopticlick.eff.org/static/browser-uniqueness.pdf>)

**[2]** P. Laperdrix, W. Rudametkin and B. Baudry. “Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints”. In IEEE Symposium on Security and Privacy (S&P’16).  
[[PDF]](<https://hal.inria.fr/hal-01285470v2/document>)

**[3]** A. Gómez-Boix, P. Laperdrix, and B. Baudry. “Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale”. In The Web Conference 2018 (WWW’18). [[PDF](https://hal.inria.fr/hal-01718234v2/document)]

**[4]** K. Mowery, and H. Shacham. “Pixel perfect: Fingerprinting canvas in HTML5”. In Web 2.0 Security & Privacy (W2SP’12). [[PDF](https://www.ieee-security.org/TC/W2SP/2012/papers/w2sp12-final4.pdf)]

**[5]** Y. Cao, S. Li, and E. Wijmans. “(Cross-) Browser Fingerprinting via OS and Hardware Level Features”. In Network and Distributed System Security Symposium (NDSS’17). [[PDF](https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_02B-3_Cao_paper.pdf)]

**[6]** S. Englehardt, and A. Narayanan. “Online tracking: A 1-million-site measurement and analysis”. In ACM SIGSAC Conference on Computer and Communications Security (CCS’16). [[PDF](http://randomwalker.info/publications/OpenWPM_1_million_site_tracking_measurement.pdf)]

**[7]** A. Sjösten, S. Van Acker, and A. Sabelfeld. “Discovering Browser Extensions via Web Accessible  
Resources”. In ACM on Conference on Data and Application Security and Privacy (CODASPY’17). [[PDF](https://www.cse.chalmers.se/~andrei/codaspy17.pdf)]

**[8]**O. Starov, and N. Nikiforakis. “XHOUND: Quantifying the Fingerprintability of Browser Extensions”. In IEEE Symposium on Security and Privacy (S&P’17). [[PDF](https://securitee.org/files/xhound-oakland17.pdf)]

**[9]**Ł. Olejnik, G. Acar, C. Castelluccia, and C. Diaz. “The Leaking Battery”. In International Workshop on Data Privacy Management (DPM’15). [[PDF](https://eprint.iacr.org/2015/616.pdf)]

**[10]** Ł. Olejnik, S. Englehardt, and A. Narayanan. “Battery Status Not Included: Assessing Privacy in  
Web Standards”. In International Workshop on Privacy Engineering (IWPE’17). [[PDF](http://randomwalker.info/publications/battery-status-case-study.pdf)]

**[11]** P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine. “Browser Fingerprinting: A survey”. [[PDF - Preprint](https://arxiv.org/pdf/1905.01051)]