Fingerprinting

Contents

 [[hide](https://wiki.mozilla.org/Fingerprinting)]

* [1Overview](https://wiki.mozilla.org/Fingerprinting#Overview)
* [2Data](https://wiki.mozilla.org/Fingerprinting#Data)
  + [2.1Plugins](https://wiki.mozilla.org/Fingerprinting#Plugins)
  + [2.2Fonts](https://wiki.mozilla.org/Fingerprinting#Fonts)
  + [2.3User Agent](https://wiki.mozilla.org/Fingerprinting#User_Agent)
  + [2.4HTTP ACCEPT](https://wiki.mozilla.org/Fingerprinting#HTTP_ACCEPT)
  + [2.5Screen resolution](https://wiki.mozilla.org/Fingerprinting#Screen_resolution)
  + [2.6Timezone](https://wiki.mozilla.org/Fingerprinting#Timezone)
  + [2.7Supercookies](https://wiki.mozilla.org/Fingerprinting#Supercookies)
  + [2.8Cookies enabled](https://wiki.mozilla.org/Fingerprinting#Cookies_enabled)
* [3Extra credit](https://wiki.mozilla.org/Fingerprinting#Extra_credit)
  + [3.1Other data acquired via plugins](https://wiki.mozilla.org/Fingerprinting#Other_data_acquired_via_plugins)
  + [3.2Clock skew measurements](https://wiki.mozilla.org/Fingerprinting#Clock_skew_measurements)
  + [3.3TCP stack](https://wiki.mozilla.org/Fingerprinting#TCP_stack)
  + [3.4JS behavioral tests](https://wiki.mozilla.org/Fingerprinting#JS_behavioral_tests)
  + [3.5Recommend privacy-related addons and services](https://wiki.mozilla.org/Fingerprinting#Recommend_privacy-related_addons_and_services)
  + [3.6User interface](https://wiki.mozilla.org/Fingerprinting#User_interface)
  + [3.7HTML5 Canvas](https://wiki.mozilla.org/Fingerprinting#HTML5_Canvas)
* [4See Also](https://wiki.mozilla.org/Fingerprinting#See_Also)

Overview

The EFF published an excellent study in May, detailing some of the various methods of fingerprinting a browser. See <http://www.eff.org/deeplinks/2010/05/every-browser-unique-results-fom-panopticlick>. They found that, over their study of around 1 million visits to their study website, **83.6% of the browsers seen had a unique fingerprint; among those with Flash or Java enabled, 94.2%**. This does **not** include cookies! They ranked the various bits of information in order of importance (i.e. how useful they are in uniquely identifying a browser): things like UA string, what addons are installed, and the font list of the system. We need to go through these, one by one, and do what we can to reduce the number of bits of information (entropy) it provides. In their study, they placed a lower bound on the fingerprint distribution of 18.1 bits of entropy. (This means that, choosing a browser at random, at best one in 286,777 other browsers will share its fingerprint.)

Data

The following data is taken from the published paper, <https://panopticlick.eff.org/browser-uniqueness.pdf>:

|  |  |
| --- | --- |
| **Entropy of various pieces of browser information** | |
| **Variable** | **Entropy (bits)** |
| plugins | 15.4 |
| fonts | 13.9 |
| user agent | 10.0 |
| http accept | 6.09 |
| screen resolution | 4.83 |
| timezone | 3.04 |
| supercookies | 2.12 |
| cookies enabled | 0.353 |

In all cases, data was either collected or inferred via HTTP, or collected by JS code and posted back to the server via AJAX.

Plugins

The PluginDetect JS library was used to check for 8 common plugins on that platform, plus extra code to estimate the Acrobat Reader version. Data sent by AJAX post.

IE does not allow enumeration via navigator.plugins[]. Starting in Firefox 28 ([bug 757726](https://bugzilla.mozilla.org/show_bug.cgi?id=757726)), Firefox restricts which plugins are visible to content enumerating navigator.plugins[]. This change does not disable any plugins; it just hides some plugin names from enumeration. Websites can still check whether a particular hidden plugin is installed by directly querying navigator.plugins[] like navigator.plugins["Silverlight Plug-In"].

This code change will reduce browser uniqueness by "cloaking" uncommon plugin names from navigator.plugins[] enumeration. If a website does not use the "Adobe Acrobat NPAPI Plug-in, Version 11.0.02" plugin, why does it need to know that the "Adobe Acrobat NPAPI Plug-in, Version 11.0.02" plugin is installed? If a website does need to know whether the plugin is installed or meets minimum version requirements, it can still check navigator.plugins["Adobe Acrobat NPAPI Plug-in, Version 11.0.02"] or navigator.mimeTypes["application/vnd.fdf"].enabledPlugin (to workaround problem plugins that short-sightedly include version numbers in their names, thus allow only individual plugin versions to be queried).

For example, the following JavaScript reveals my installed plugins:

for (plugin of navigator.plugins) { console.log(plugin.name); }

"Shockwave Flash"

"QuickTime Plug-in 7.7.3"

"Default Browser Helper"

"Unity Player"

"Google Earth Plug-in"

"Silverlight Plug-In"

"Java Applet Plug-in"

"Adobe Acrobat NPAPI Plug-in, Version 11.0.02"

"WacomTabletPlugin"

navigator.plugins["Unity Player"].name // get cloaked plugin by name

"Unity Player"

But with plugin cloaking, the same JavaScript will not reveal as much personally-identifying information about my browser because all plugin names except Flash, Shockwave (Director), Java, and QuickTime are hidden from navigator.plugins[] enumeration:

for (plugin of navigator.plugins) { console.log(plugin.name); }

"Shockwave Flash"

"QuickTime Plug-in 7.7.3"

"Java Applet Plug-in"

In theory, all plugin names could be cloaked because web content can query navigator.plugins[] by plugin name. Unfortunately, we could not cloak all plugin names because many popular websites check for Flash or QuickTime by enumerating navigator.plugins[] and comparing plugin names one by one, instead of just asking for navigator.plugins["Shockwave Flash"] by name. These websites should be fixed.

The policy of which plugin names are uncloaked can be changed in the about:config pref plugins.enumerable\_names. The pref’s value is a comma-separated list of plugin name prefixes (so the prefix "QuickTime" will match both "QuickTime Plug-in 6.4" and "QuickTime Plug-in 7.7.3"). The default pref cloaks all plugin names except Flash, Shockwave (Director), Java, and QuickTime. To cloak all plugin names, set the pref to the empty string "" (without quotes). To cloak no plugin names, set the pref to magic value "\*" (without quotes).

Fonts

System fonts collected by Flash or Java applet, if installed, and sent via AJAX post. Font list was not sorted, which provides a bit or two of additional entropy. We can ask Adobe to either limit this list by default; or ask them to implement an API such that we can provide the list to them; or (made possible by OOPP) replace the OS API calls they use to get the font list, and give them our own. None of these things are easy, but given that this is #1, we should definitely do something here. The fastest option is probably to hack the OS API calls ourselves.

Font lists can also be determined by CSS introspection. We could perhaps reduce the available set to a smaller number of common fonts; and back off (exponentially?) if script attempts to brute-force the list. Could require that sites provide unusual fonts via WOFF?

User Agent

Detected from HTTP header. Pretty simple fix, but has the potential for breakage (as with any UA change!). For instance: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.7) Gecko/20100106 Ubuntu/9.10 (karmic) Firefox/3.5.7. Remedies: remove the last point digit in the Firefox and Gecko versions, and the Gecko build date; for Linux, remove distribution and version; possibly remove CPU. Windows is actually the least unique since the OS version string only identifies the major version (e.g. XP), and by far the majority of users are on it.

Remove language and "Firefox" as well?

Boris Zbarsky points out that most parts of the UA lead to bad sniffing. Irish "ga-IE" and "Minef**ie**ld" get detected as IE. Sites incorrectly sniff based on OS. Sites sniff for Gecko years rather than Gecko versions. Going from 3.0.9 to 3.0.10 probably breaks things. And quite a few sites sniff for "Firefox", which is a threat to the continued freedom of the web. So removing things from the UA string has a long-term positive effect on compatibility as well as privacy.

There is another issue with UA spoofing. For some reason, Components.classes and Components.interfaces exist in the content-window javascript namespace. Gregory Fleischer used this to test for the existence of ephemeral interfaces to [fingerprint both OS and Firefox version](http://pseudo-flaw.net/tor/torbutton/fingerprint-firefox.html), down to the minor revision (FF3.5.3 was the latest release at the time). He has a [number of other fingerprinting demos](http://pseudo-flaw.net/content/defcon/dc-17-demos/) you should investigate as well. -- [mikeperry](https://wiki.mozilla.org/index.php?title=User:Mikeperry&action=edit&redlink=1)

Filed [bugs](https://bugzilla.mozilla.org/show_bug.cgi?id=http-fingerprint). [Hsivonen](https://wiki.mozilla.org/index.php?title=User:Hsivonen&action=edit&redlink=1) 09:33, 18 June 2010 (UTC)

HTTP ACCEPT

Example: text/html, \*/\* ISO-8859-1,utf-8;q=0.7,\*;q=0.7 gzip,deﬂate en- us,en;q=0.5. Not sure we can do much here?

Screen resolution

Example: 1280x800x24. Can't mess with this, except perhaps to always report "24" for the color depth -- of dubious value.

Mapping "32" to "24" or vice versa in the color depth would reduce entropy by ~0.9 bits. May be worthwhile.

Torbutton takes two countermeasures with respect to screen resolution: quantising AvailWidth and AvailHeight, and setting Width and Height to the values of AvailWidth and AvailHeight. Torbutton currently errs in not doing this if the window is maximised. These measures might be appropriate in private browsing mode. -- [Pde](https://wiki.mozilla.org/index.php?title=User:Pde&action=edit&redlink=1) 03:12, 15 June 2010 (UTC)

Timezone

Too useful to break.

Supercookies

The reported entropy includes **only** whether the following were enabled: DOM localStorage, DOM sessionStorage, and (for IE) userData. It did **not** test Flash LSOs, Silverlight cookies, HTML5 databases, or DOM globalStorage. We can't do anything to prevent testing whether these are enabled, but we can lock them down for third parties, as we will with cookies.

For Flash and Silverlight we need to pressure them to implement better APIs for controlling and clearing stored data. This is undoubtedly **more important** than anything else on this list, though it was ignored in this study since it does not fit within their definition of fingerprinting. We could be aggressive here by using the new Flash API for private browsing mode very liberally; or do something with the OS APIs as mentioned above.

Cookies enabled

Irrelevant due to low amount of entropy.

Extra credit

Other fingerprinting methods were mentioned, but not included, in the study. A Gartner report on fingerprinting services was referenced in the study, which will undoubtedly be interesting to read.

Examples:

Other data acquired via plugins

Undoubtedly Flash and Java provide other interesting tidbits. ActiveX and Silverlight, for example, allow querying the "CPU type and many other details". More study needed here.

Clock skew measurements

"41st Parameter looks at more than 100 parameters, and at the core of its algorithm is a time differential parameter that measures the time difference between a user’s PC (down to the millisecond) and a server’s PC." We can't break the millisecond resolution of Date.now, but we could try adding a small (< 100ms) offset to it. This would be generated per-origin, and would last for some relatively short time: life of session, life of tab, etc. Would have to be careful that it can't be reversed.

Clock skew measurement isn't really a browser issue; it tends to be exposed by the operating system at the TCP level. It would be appropriate to assume that an attacker can obtain 4-6 bits of information about the identity of a host by this method. -- [Pde](https://wiki.mozilla.org/index.php?title=User:Pde&action=edit&redlink=1) 02:55, 15 June 2010 (UTC)

This is not 100% correct. According to [RFC 1323](http://www.faqs.org/rfcs/rfc1323.html) sections 3.2 and 4.2.2, timestamps may only be used if the initial syn packet (not syn+ack) contains a timestamp field. This is a property of the client OS, and may be controllable on some platforms. The timestamp value is also not absolute, but is typically some arbitrary number of milliseconds with no specific reference point. TLS also has a timestamp, but this value is fully controlled by Firefox. -- [mikeperry](https://wiki.mozilla.org/index.php?title=User:Mikeperry&action=edit&redlink=1)

Agree that one could turn off the TCP RTTM option at the OS layer. My naive intuition is that all modern OSes have this turned on, and turning it off would be a radical intervention bad for congestion avoidance and possibly fingerprintable itself. Note that clock skew is a function of how fast a clock ticks, not of what time the clock has. An arbitrary reference point is sufficient for measuring clock skew. -- [Pde](https://wiki.mozilla.org/index.php?title=User:Pde&action=edit&redlink=1) 08:23, 9 December 2010 (PST)

Note also that it's not just clock skew, but also clock precision that can allow for fingerprinting - both in terms of how long certain operations take on a system and in terms of user action. For example, [Scout Analytics](http://www.scoutanalytics.com/) provides software to fingerprint users based on [typing cadence](http://arstechnica.com/tech-policy/news/2010/02/firm-uses-typing-cadence-to-finger-unauthorized-users.ars). One can also imagine tight loops of timed javascript that fingerprint users based on certain resource-intensive calls. One possibility might be to quantize Date values to the second, and then add random, monotonically increasing amounts of milliseconds to subsequent calls during private browsing mode. -- [mikeperry](https://wiki.mozilla.org/index.php?title=User:Mikeperry&action=edit&redlink=1)

TCP stack

"ThreatMetrix claims that it can detect irregularities in the TCP/IP stack and can pierce through proxy servers". Not sure what this means yet.

nmap's host fingerprinting options (and source code) are the first place to start for understanding the TCP/IP stack issues. Again, there's not much the browser can do about that.

As for "pierce through proxy servers", my best guess is that they use the raw socket infrastructure provided by Flash, which does not respect the browser's proxy settings, in order to learn the client's IP. Not sure if Java and Silverlight have similar problems. -- [Pde](https://wiki.mozilla.org/index.php?title=User:Pde&action=edit&redlink=1) 02:58, 15 June 2010 (UTC)

JS behavioral tests

Can be used to gather information about whether certain addons are installed, exact browser version, etc. Probably nothing we can do here.

Recommend privacy-related addons and services

"TorButton has evolved to give considerable thought to fingerprint resistance [19] and may be receiving the levels of scrutiny necessary to succeed in that project [15]. NoScript is a useful privacy enhancing technology that seems to reduce fingerprintability."

"We identified only three groups of browser with comparatively good resistance to fingerprinting: those that block JavaScript, those that use TorButton, and certain types of smartphone."

We should study what TorButton does, and see if we can integrate some of its features. We can also recommend it, NoScript, and Flashblock to users. We could suggest improvements to relevant addons, such as providing options for blocking third party but not first party content. (This doesn't strictly solve anything, but makes gathering the data more difficult, since the third party now relies on the first party to collect it.)

Unfortunately Flashblock does not appear to prevent Flash from reading and writing LSOs, so it's doubtful it can be relied upon to protect against fingerprinting. -- [Pde](https://wiki.mozilla.org/index.php?title=User:Pde&action=edit&redlink=1) 03:00, 15 June 2010 (UTC)

User interface

Things like geolocation, database access and such require the user to grant permission for a given site. For geolocation, this is done with an infobar. We should do everything we can to make it clear to users what they're providing, and give them centralized control of those permissions in the privacy panel. This is what the UX privacy proposals seek to do.

HTML5 Canvas

"After plugins and plugin-provided information, we believe that the HTML5 Canvas is the single largest fingerprinting threat browsers face today." - Tor Project. Original research: [Pixel Perfect: Fingerprinting Canvas in HTML5](http://cseweb.ucsd.edu/~hovav/dist/canvas.pdf), demo: [HTML5 Canvas Fingerprinting](https://www.browserleaks.com/canvas).

See Also

* [Chromium's list of client identification mechanisms](http://www.chromium.org/Home/chromium-security/client-identification-mechanisms)