Privacy, Standards and Anti-Standards

Peter Snyder, Privacy Researcher, pes@brave.com
Overview

- Standards as a privacy focused implementor

- How the standards process makes privacy difficult (and how it can be fixed)

- Bonus concerns and conclusions
Overview

- Standards as a privacy focused implementor

- How the standards process makes privacy difficult (and how it can be fixed)

- Bonus concerns and conclusions
Privacy in Brave

- Tighter Default Storage Controls
- Tor Integration
- Resource Blocking
- Web API / DOM Modifications
Privacy in Brave

- Tighter Default Storage Controls
- Tor Integration
- Resource Blocking
- Web API / DOM Modifications

Web Standards / W3C / IETF
Browser Fingerprinting: A survey

PIERRE LAPERDRIX, CISPA Helmholtz Center for Information Security, Germany
NATALIJA BIELOVA, Inria Sophia Antipolis, France
BENOIT BAUDRY, KTH Royal Institute of Technology, Sweden
GILDAS AVOINE, Univ Rennes, INSA Rennes, CNRS, IRISA, France

With this paper, we survey the research performed in the domain of browser fingerprinting, while providing an accessible entry point to newcomers in the field. We explain how this technique works and where it stems from. We analyze the related work in detail to understand the composition of modern fingerprints and see how this technique is currently used online. We systematize existing defense solutions into different categories and detail the current challenges yet to overcome.

CCS Concepts: • Security and privacy → Web application security; Browser security; Privacy protections;

Additional Key Words and Phrases: Browser fingerprinting, user privacy, web tracking

1 INTRODUCTION
The web is a beautiful platform and browsers give us our entry point into it. With the introduction of HTML5 and CSS3, the web has become richer and more dynamic than ever and it has now the foundations to support an incredible ecosystem of diverse devices from laptops to smartphones and tablets. The diversity that is part of the modern web opened the door to device fingerprinting, a simple identification technique that can be used to collect a vast list of device characteristics.
Table 4. Overview of four studies measuring adoption of browser fingerprinting on the web.

<table>
<thead>
<tr>
<th>Fingerprinting techniques detected</th>
<th>Detection of 3 known fingerprinting libraries</th>
<th>JS-based and Flash-based font probing</th>
<th>Canvas fingerprinting, canvas-based font probing, WebRTC and AudioContext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sites crawled</td>
<td>10K sites (up to 20 pages per site)</td>
<td>1M sites (homepages)</td>
<td>1.4% for canvas fingerprinting, 0.325% for canvas font probing, 0.0715% for WebRTC, 0.0067% for AudioContext</td>
</tr>
<tr>
<td>Prevalence</td>
<td>0.4%</td>
<td>0.04% (404 of 1M) for JS-based, 1.45% (145 of 10K) for Flash-based</td>
<td>5.5%</td>
</tr>
<tr>
<td>Detection method</td>
<td>Presence of JS libraries provided by BlueCava, Iovation and ThreatMetrix.</td>
<td>Logging calls of font probing methods. A script that loads more than 30 fonts or a Flash file that contains font enumeration calls is considered to perform fingerprinting.</td>
<td>Logging calls of canvas fingerprinting related methods. A script is considered to perform fingerprinting if it also checks other FP-related properties.</td>
</tr>
</tbody>
</table>

1 INTRODUCTION

The web is a beautiful platform for the development of HTML5 and CSS3, the web is the built-in foundations to support an increasing number of desktops and tablets. The diversity that arises from this identification technique is currently used on a large scale and has led to the current challenges yet to be addressed.


<table>
<thead>
<tr>
<th>Detection method</th>
<th>Presence of JS libraries provided by BlueCava, Iovation and ThreatMetrix.</th>
<th>A script that loads more than 30 fonts or a Flash file that contains font enumeration calls is considered to perform fingerprinting.</th>
<th>A script is considered to perform fingerprinting if it also checks other FP-related properties.</th>
<th>Logging calls of canvas fingerprinting related methods.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fingerprinting techniques detected</td>
<td>Detection of 3 known fingerprinting libraries</td>
<td>JS-based and Flash-based font probing</td>
<td>Canvas fingerprinting</td>
<td>Logging calls of advanced FP-related JavaScript functions.</td>
</tr>
<tr>
<td>Sites crawled</td>
<td>10K sites (up to 20 pages per site)</td>
<td>1M sites (100K sites (25 links per site) for JS</td>
<td>100K sites (homepages)</td>
<td>1.4% for canvas fingerprinting 0.325% for canvas font probing 0.0715% for WebRTC 0.0067% for AudioContext</td>
</tr>
<tr>
<td>Prevalence</td>
<td>0.4%</td>
<td>0.04% (404 of 1M) for JS-based 1.45% (145 of 10K) for Flash-based</td>
<td>5.5%</td>
<td>Logging calls of canvas fingerprinting related methods.</td>
</tr>
</tbody>
</table>

Table 4. Overview of four studies measuring adoption of browser fingerprinting on the web.
Web API Modifications

Fingerprinting methods blocked in Fingerprinting Protection Mode

- Canvas fingerprinting: it should report a fixed value on tests like panopticlick
- WebGL fingerprinting: it should report as undefined on tests like panopticlick
- AudioContext fingerprinting
- WebRTC IP leakage
- SVG fingerprinting (specifically, the
  SVGTextContentElement.prototype.getComputedTextLength and
  SVGPathElement.prototype.getTotalLength methods)
- HSTS fingerprinting

Privacy protection enabled regardless of whether Fingerprinting Protection Mode is on

This list is not complete. See https://github.com/brave/brave-browser/wiki/Deviations-from-Chromium-(features-we-disable-or-remove) for other things which are disabled in Brave but not in Chrome.
Web API Modifications

Fingerprinting methods blocked in Fingerprinting Protection Mode

- **Canvas fingerprinting**: it should report a fixed value on tests like panopticlick
- **WebGL fingerprinting**: it should report as undefined on tests like panopticlick
- **AudioContext fingerprinting**
- **WebRTC IP leakage**
- **SVG fingerprinting** (specifically, the
  `SVGTextContentElement.prototype.getComputedTextLength` and
  `SVGPathElement.prototype.getTotalLength` methods)
- **HSTS fingerprinting**

Privacy protection enabled regardless of whether Fingerprinting Protection Mode is on

This list is not complete. See [https://github.com/brave/brave-browser/wiki/Deviations-from-Chromium-(features-we-disable-or-remove)](https://github.com/brave/brave-browser/wiki/Deviations-from-Chromium-(features-we-disable-or-remove)) for other things which are disabled in Brave but not in Chrome.
Web Audio Fingerprinting

- Standard says websites can query hardware
- Hardware is pseudo-identifying
- Enough pseudo-identifiers yield a real identifier
- So Brave breaks the standard…
Breaking Standards for Privacy

- Hardware Detection:
  - Web Audio
  - WebGL
  - WebUSB
  - Battery API

- Network Information
  - WebRTC

- Font Enumeration:
  - Canvas
  - SVG

- Display Information:
  - Client Hints

- Browsing History:
  - Referrer Policy
Overview

- Standards as a privacy focused implementor

- How the standards process makes privacy difficult (and how it can be fixed)

- Bonus concerns and conclusions
Three Standards
Privacy Anti-Patterns
There’s no trick to it. It’s just a simple trick!
1. Defined Functionality, Non-Normative Mitigations
Privacy Risk w/ Non-Normative Mitigations

- Privacy-harming / risky functionality

- "Privacy considerations" section, but non-standardized mitigation

- The Web assumes the dominant implementation, instead of the standard

- **Result:** Harm is “locked in” / out of control of the standards process
Abstract
§ 1. Introduction

This section is not normative.

Requests made from a document, and for navigations away from that document are associated with a Referer header. While the header can be suppressed for links with the noreferrer link type, authors might wish to control the Referer header more directly for a number of reasons:

§ 1.1. Privacy

A social networking site has a profile page for each of its users, and users add hyperlinks from their profile page to their favorite bands. The social networking site might not wish to leak the user’s profile URL to the band websites when other users follow those hyperlinks (because the profile URLs might reveal the identity of the owner of the profile).

Some social networking sites, however, might wish to inform the band websites that the links originated from the social networking site but not reveal which specific user’s profile contained the links.

§ 1.2. Security

A web application uses HTTPS and a URL-based session identifier. The web application might wish to link to HTTPS resources on other web sites without leaking the user’s session identifier in the URL.

Alternatively, a web application may use URLs which themselves grant some capability. Controlling the referrer can help prevent these capability URLs from leaking via referrer headers. [CAPABILITY-URLS]

Note that there are other ways for capability URLs to leak, and controlling the referrer is not enough to control all those potential leaks.

§ 1.3. Trackback

A blog hosted over HTTPS might wish to link to a blog hosted over HTTP and receive trackback links.
3. Set `url's username` to the empty string.
4. Set `url's password` to null.
5. Set `url's fragment` to null.
6. If the `origin-only flag` is true, then:
   1. Set `url's path` to null.
   2. Set `url's query` to null.
7. Return `url`.

§ 9. Privacy Considerations

§ 9.1. User Controls

Nothing in this specification should be interpreted as preventing user agents from offering options to users which would change the information sent out via a `Referer` header. For instance, user agents MAY allow users to suppress the referrer header entirely, regardless of the active referrer policy on a page.

§ 10. Security Considerations

§ 10.1. Information Leakage

The referrer policies "origin", "origin-when-cross-origin" and "unsafe-url" might leak the origin and the URL of a secure site respectively via insecure transport.

Those three policies are included in the spec nevertheless to lower the friction of sites adopting secure transport.

Authors wanting to ensure that they do not leak any more information than the default policy should instead use the policy states "same-origin", "strict-origin", "strict-origin-when-cross-origin" or "no-referrer".
Result

- Well described functionality
- Vaguely / undefined / unclear mitigations
- Web assumes the defined functionality, privacy-harm gets locked in

**Solution:** Make mitigations normative and standardized!
1. Defined Functionality, Non-Normative Mitigations

2. Uncommon Use Case, Common Availability
Uncommon Use Case, Common Availability

- Genuinely useful functionality, for niche scenarios

- Functionality is made widely available (first-party, third-party, frames, etc.)

- Co-opted by tracking, code-paths assume availability

- **Result:** can't be removed, even from irrelevant sites
4.12.5 The canvas element

4.12.5.1 The 2D rendering context
  4.12.5.1.1 Implementation notes
  4.12.5.1.2 The canvas state
  4.12.5.1.3 Line styles
  4.12.5.1.4 Text styles
  4.12.5.1.5 Building paths
  4.12.5.1.6 Path2D objects
  4.12.5.1.7 Transformations
  4.12.5.1.8 Image sources for 2D rendering contexts
  4.12.5.1.9 Fill and stroke styles
  4.12.5.1.10 Drawing rectangles to the bitmap
  4.12.5.1.11 Drawing text to the bitmap
  4.12.5.1.12 Drawing paths to the canvas
  4.12.5.1.13 Drawing focus rings and scrolling paths into view
  4.12.5.1.14 Drawing images
  4.12.5.1.15 Pixel manipulation
  4.12.5.1.16 Compositing
  4.12.5.1.17 Image smoothing
  4.12.5.1.18 Shadows
  4.12.5.1.19 Filters
  4.12.5.1.20 Working with externally-defined SVG filters
  4.12.5.1.21 Drawing model
The `toDataURL(type, quality)` method, when invoked, must run these steps:

1. If this `canvas` element’s bitmap’s `origin-clean` flag is set to false, then throw a "SecurityError" DOMException.

2. If this `canvas` element’s bitmap has no pixels (i.e. either its horizontal dimension or its vertical dimension is zero) then return the string `"data:;"` (This is the shortest `data:` URL; it represents the empty string in a `text/plain` resource.)

3. Let `file` be a serialization of this `canvas` element’s bitmap as a file, passing type and quality if given.

4. If `file` is null then return `"data:;"`.

5. Return a `data:` URL representing `file`. [RFC2397]

The `toBlob(callback, type, quality)` method, when invoked, must run these steps:

1. If this `canvas` element’s bitmap’s `origin-clean` flag is set to false, then throw a "SecurityError" DOMException.

2. Let `result` be null.

3. If this `canvas` element’s bitmap has pixels (i.e., neither its horizontal dimension nor its vertical dimension is zero), then set `result` to a copy of this `canvas` element’s bitmap.

4. Run these steps in parallel:
   1. If `result` is non-null, then set `result` to a serialization of `result` as a file with type and quality if given.

   2. Queue a task to run these steps:
      1. If `result` is non-null, then set `result` to a new `Blob` object, created in the relevant Realm of this `canvas` element, representing `result`. [FILEAPI]

      2. Invoke `callback` with « `result` ».

The `task source` for this task is the `canvas blob serialization task source`. 
var getCanvasFP = function (options) {
    var result = [];

    // Very simple now, need to make it more complex (geo shapes etc)
    var canvas = document.createElement('canvas');
    canvas.width = 2000;
    canvas.height = 200;
    canvas.style.display = 'inline';

    var ctx = canvas.getContext('2d');

    // detect browser support of canvas winding
    // https://github.com/Modernizr/Modernizr/blob/master/feature-detects/canvas/winding
    ctx.rect(0, 0, 10, 10);
    ctx.strokeRect(2, 2, 6, 6);
    result.push('canvas winding: ' + ((ctx.isPointInPath(5, 5, 'evenodd') === false) ? 'false' : 'true'));

    ctx.textBaseline = 'alphabetic';
    ctx.fillStyle = '#f60';
    ctx.fillRect(127, 1, 20, 20);

    // https://github.com/Valve/fingerprintjs2/issues/66
    // click to follow link

    if (options.dontUseFakeFontInCanvas) {
        ctx.font = '11pt Arial';
    } else {
        ctx.font = '11pt no-real-font-123';
    }

    ctx.fillText('Cwm fjordbank glyphs vext quiz, \ud83d\ude03', 2, 15);
    ctx.fillStyle = 'rgba(102, 204, 0, 0.2)';
    ctx.font = '18pt Arial';
    ctx.fillText('Cwm fjordbank glyphs vext quiz, \ud83d\ude03', 4, 45);

    // canvas blending
    // http://jsfiddle.net/NDYV8/16/
    ctx.globalCompositeOperation = 'multiply';
    ctx.fillStyle = 'rgb(255,0,255)';
    ctx.beginPath();
    ctx.arc(50, 50, 50, 0, Math.PI * 2, true);
    ctx.closePath();
    ctx.fill();

    ctx.fillStyle = 'rgb(0,255,255)';
    ctx.beginPath();
    ctx.arc(100, 50, 50, 0, Math.PI * 2, true);
    ctx.closePath();
    ctx.fill();

    ctx.fillStyle = 'rgb(255,255,0)';
    ctx.beginPath();
    ctx.arc(75, 100, 50, 0, Math.PI * 2, true);
    ctx.closePath();
    ctx.fill();

    ctx.fillStyle = 'rgb(255,0,255)';
    ctx.beginPath();
    ctx.arc(50, 50, 50, 0, Math.PI * 2, true);
    ctx.closePath();
    ctx.fill();

    return result;
};
<table>
<thead>
<tr>
<th>Browser Characteristic</th>
<th>bits of identifying information</th>
<th>one in x browsers have this value</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Agent</td>
<td>13.54</td>
<td>11932.41</td>
<td>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.91 Safari/537.36</td>
</tr>
<tr>
<td>HTTP_ACCEPT Headers</td>
<td>3.15</td>
<td>8.87</td>
<td>text/html, &quot;;&quot;; v=0.01 gzip, deflate, br en-US, en;q=0.9</td>
</tr>
<tr>
<td>Browser Plugin Details</td>
<td>0.91</td>
<td>1.88</td>
<td>undefined</td>
</tr>
<tr>
<td>Time Zone</td>
<td>4.22</td>
<td>18.66</td>
<td>420</td>
</tr>
<tr>
<td>Screen Size and Color Depth</td>
<td>5.49</td>
<td>44.81</td>
<td>1680x1050x24</td>
</tr>
<tr>
<td>Are Cookies Enabled?</td>
<td>0.27</td>
<td>1.21</td>
<td>Yes</td>
</tr>
<tr>
<td>Limited supercookie test</td>
<td>0.4</td>
<td>1.32</td>
<td>DOM localStorage: Yes, DOM sessionStorage: Yes, IE userData: No</td>
</tr>
<tr>
<td>Hash of canvas fingerprint</td>
<td>5.68</td>
<td>51.1</td>
<td>c0f4c1dcbb26af77f05764e6cc2df0e711</td>
</tr>
<tr>
<td>Hash of WebGL fingerprint</td>
<td>3.89</td>
<td>14.78</td>
<td>undetermined</td>
</tr>
<tr>
<td>DNT Header Enabled?</td>
<td>1.24</td>
<td>2.37</td>
<td>False</td>
</tr>
<tr>
<td>Language</td>
<td>1.0</td>
<td>1.99</td>
<td>en-US</td>
</tr>
<tr>
<td>Platform</td>
<td>3.26</td>
<td>9.59</td>
<td>MacIntel</td>
</tr>
<tr>
<td>Touch Support</td>
<td>0.76</td>
<td>1.7</td>
<td>Max touchpoints: 0; TouchEvent supported: false; onTouchStart supported: false</td>
</tr>
</tbody>
</table>

- **Widely Available**
- **Sites / benign code expects**
- **Removing / blocking breaks benign sites**
Lots of rare-use-case functionality

- Brightness sensors
- WebVR
- Machine Learning APIs
- High Resolution Timers
- Vibration
- WebGL operations
- Tracing APIs
- Many many many more…
Lesson Learned

- Assume people will find bad uses for your functionality

- General access -> difficult to remove / modify

**Solution:** Restrict access to the use cases you care about

- User gestures
- Permission prompts
- Not-in-frames
1. Defined Functionality, Non-Normative Mitigations

2. Uncommon Use Case, Common Availability

3. “No worse than the status quo”
“No worse than the status quo”

- Privacy-harming / risky functionality

- “Information is available elsewhere, so no additional harm”

- **Result:** Web compat difficulty expands…
HTTP Client Hints
draft-ietf-httpbis-client-hints-07

Abstract
HTTP defines proactive content negotiation to allow servers to select the appropriate response for a given request, based upon the user agent's characteristics, as expressed in request headers. In practice, clients are often unwilling to send those request headers, because it is not clear whether they will be used, and sending them impacts both performance and privacy.

This document defines two response headers, Accept-CH and Accept-CH-Lifetime, that servers can use to advertise their use of request headers for proactive content negotiation, along with a set of guidelines for the creation of such headers, colloquially known as "Client Hints."

Note to Readers
Discussion of this draft takes place on the HTTP working group mailing list (ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/.

Working Group information can be found at http://httpwg.github.io/; source code and issues list for this draft can be found at https://github.com/httpwg/http-extensions/labels/client-hints.

Status of this Memo
This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF).
Client

GET /index.html

Server
Client

GET /index.html
Accept-CH: DPR
Accept-CH: Viewport-Width

Server
Client

GET /index.html

Accept-CH: DPR
Accept-CH: Viewport-Width

Server

DPR: 2
Viewport-Width: 1434
Values in Client Hints are Identifying

- Eckersley, Peter. "How unique is your web browser?." PETS 2010
  Viewport height and width

  Device color depth

  The above are being used often!
Client Hints Authors’ Current Position

- This information is already available
- No further exposure / no marginal harm

Brave’s Concerns with the Client-Hints Proposal
WE’LL DIG OUR WAY OUT!
Lesson Learned

- “Horizontal” privacy risk is technological debt

- Same data in more places entrenches the risk

- **Solution:** Treat all additional privacy risk as equally problematic
Overview

- Standards as a privacy focused implementor
- How the standards process makes privacy difficult (and how it can be fixed)
- Bonus concerns and conclusions
Bonus anti-patterns

““This privacy concern is addressed by an upcoming standard…””

“This just formalizes existing bad practice…”

"Site owners want it, users like sites, so by the transitive property…”"
Bonus suggestions / concerns / worries / rants

- Pump the breaks on everything

- Complexity is a privacy risk

- Amount of “standards” work that is shipped-than-standardized
Overview

- Standards as a privacy focused implementor

- How the standards process makes privacy difficult (and how it can be fixed)

- Bonus concerns and conclusions
Conclusion

- Privacy preserving standards are important to improving the Web.

- Weak standards make it difficult for privacy-interested parties to improve things.

- A few small changes to privacy criteria in standards would make a huge difference.

Pete Snyder
Privacy Researcher
pes@brave.com