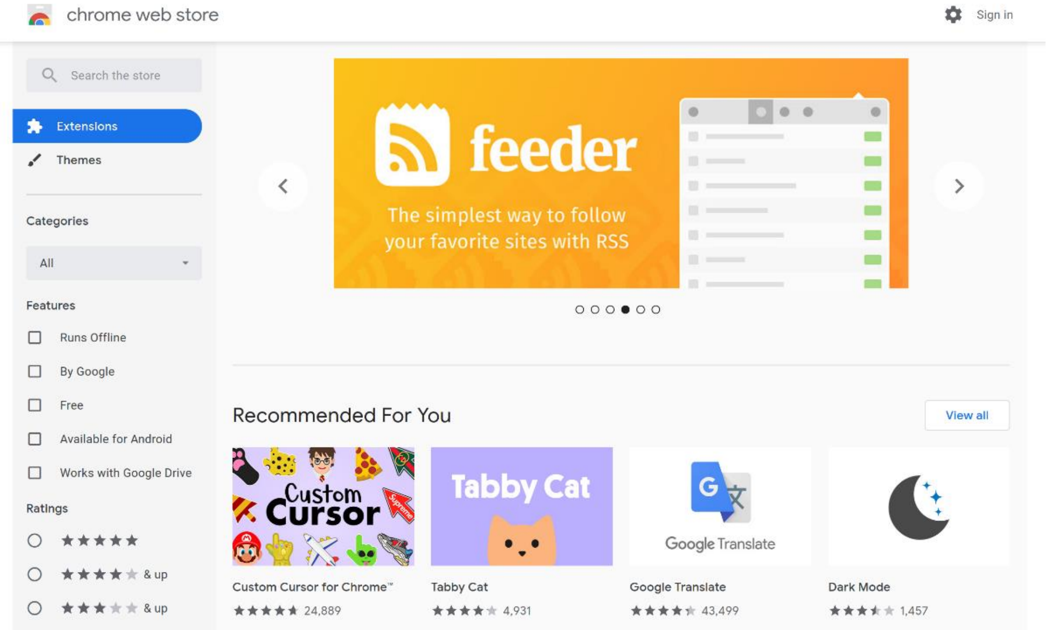
Fingerprinting in Style: Detecting Browser Extensions via Injected Style Sheets

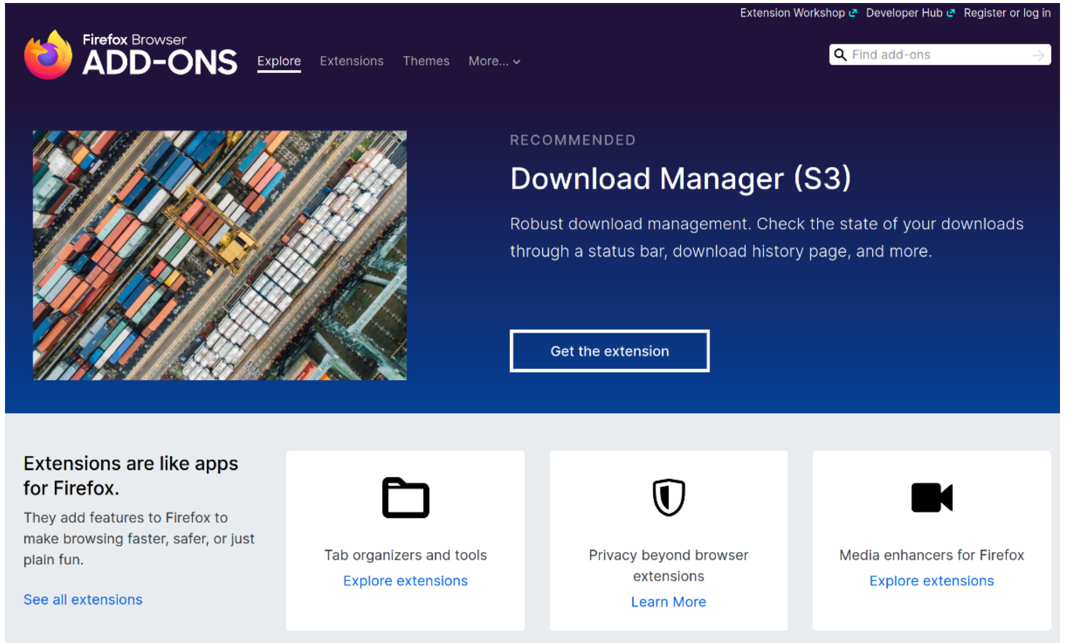
Pierre Laperdrix, Oleksii Starov, Quan Chen, Alexandros Kapravelos and Nick Nikiforakis

Browser extensions

- Small programs that extend the capabilities of a browser
- Found in official extension stores



The simplest way to follow your favorite sites with RSS



Download Manager (S3)

Robust download management. Check the state of your downloads through a status bar, download history page, and more.

Extensions are like apps for Firefox.

They add features to Firefox to make browsing faster, safer, or just plain fun.

- Some of the most popular extensions are ad blockers, password managers or download helpers.

Browser extension fingerprinting: what is it?

- Build a list of extensions installed in a user’s browser
  - No API exists to get this list.
  - Use of extension side effects to detect them.
- Who can fingerprint extensions?
  - Any website with a simple script can do it.
  - No need for any permissions.

Browser extension fingerprinting: Implications

Finding the list of installed extensions can:

- Complement existing browser fingerprinting techniques. If a list is unique or highly unusual, it can lead to user identification online.
- Reveal some personal information like the use of a specific software or service.

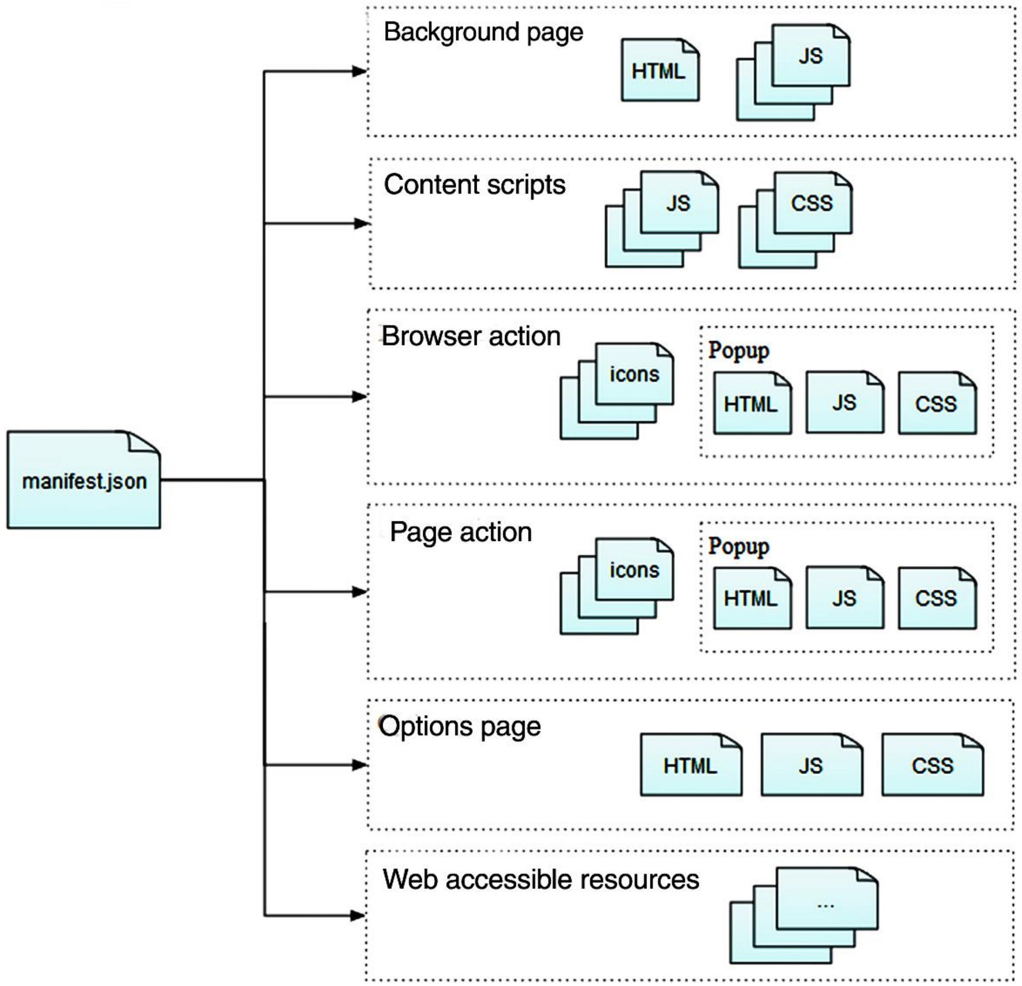
VPNs

Password managers

Countdown

Religion

Architecture of a browser extension



Source: MDN Web Docs

Structure of a browser extension

- Manifest.json is a mandatory file that provides metadata information on how the extension works.
- Background page implements long-term logic.
- Content scripts are scripts that are injected into visited webpages.
- Web accessible resources are files like JS libraries or icons that can be accessed by the extension or any webpage.

Browser extension fingerprinting: Related work

1st method: Web Accessible Resources (WAR) fingerprinting (Codaspy’17)

Discovering Browser Extensions via Web Accessible Resources

Alexander Sjösten
Chalmers University of Technology
Gothenburg, Sweden
sjosten@chalmers.se

Steven Van Acker
Chalmers University of Technology
Gothenburg, Sweden
acker@chalmers.se

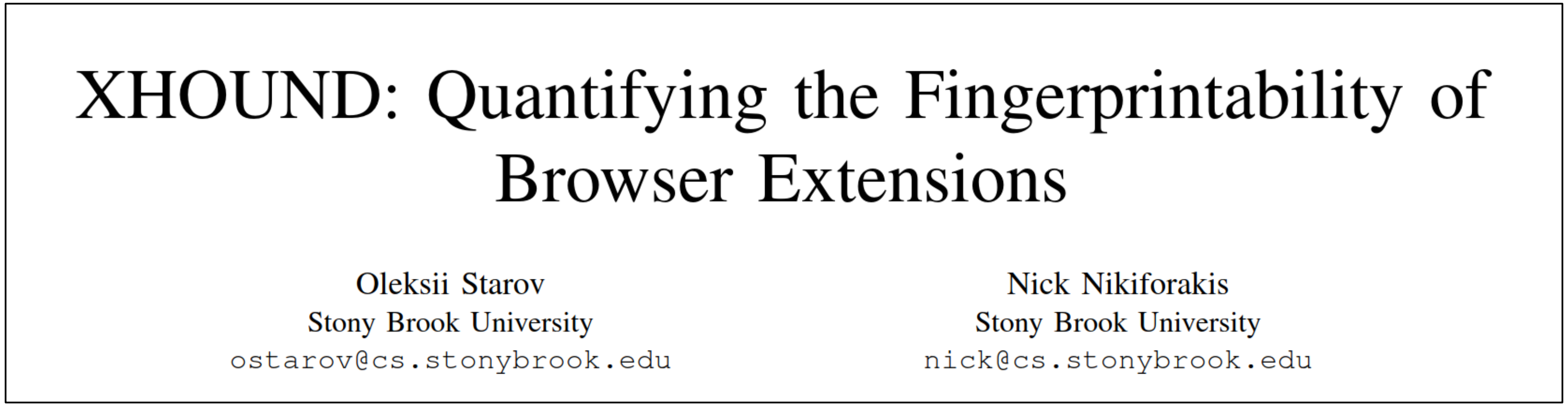
Andrei Sabelfeld
Chalmers University of Technology
Gothenburg, Sweden
andrei@chalmers.se

- Probes specific WARs in the browser to identify an extension.
- Requires knowledge beforehand of extension IDs and paths of WAR files.
- In the future: Manifest V3 in Chrome will provide finer-grained access control for WARs along with the introduction of dynamic URLs

developer.chrome.com/docs/extensions/mv3/manifest/web_accessible_resources/

Browser extension fingerprinting: Related work

2nd method: Behavioral fingerprinting (S&P'17)



XHOUND: Quantifying the Fingerprintability of Browser Extensions

Oleksii Starov
Stony Brook University
ostarov@cs.stonybrook.edu

Nick Nikiforakis
Stony Brook University
nick@cs.stonybrook.edu

- Identifies an extension organic activity on a page (DOM modifications made by an extension).
- Requires knowledge beforehand of the modifications made by an extension on a page.

Browser extension fingerprinting: Related work

3^{rd} method: postMessages (NDSS’20)

Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting

Soroush Karami, Panagiotis Ilia, Konstantinos Solomos, Jason Polakis
University of Illinois at Chicago, USA
{skaram5, pilia, ksolom6, polakis}@uic.edu

- Exchanges messages with an extension through the postMessages API.

Detecting extensions through style modifications

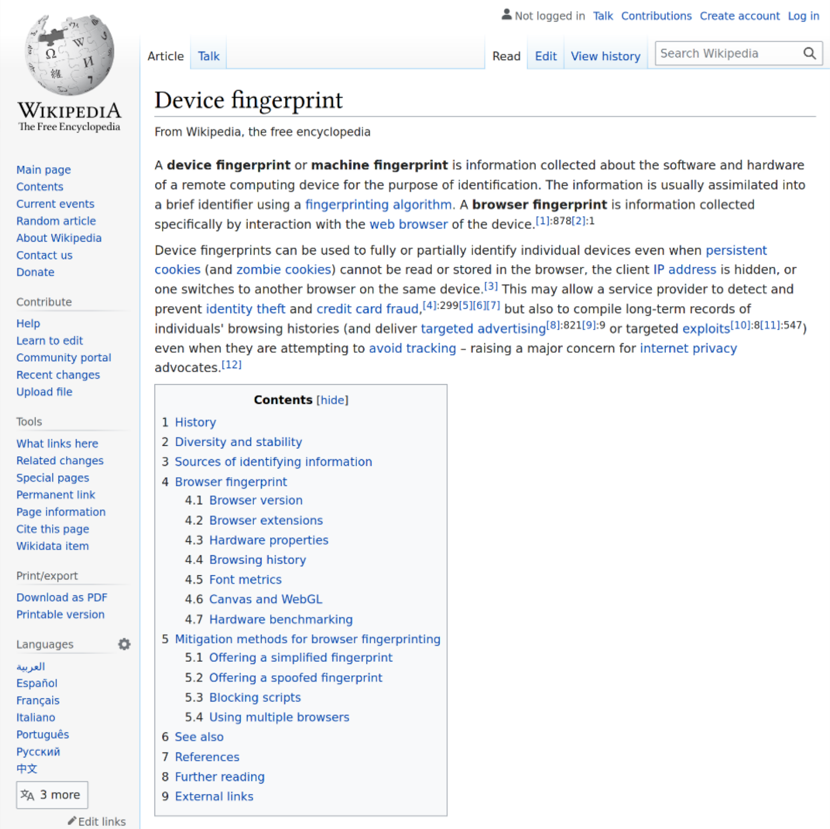
- A HTML webpage is a simple file mixing HTML elements, JavaScript code and CSS directives.

- Extensions can customize the way the page looks by inserting CSS rules as a **content script**.

- While some rules are active on very specific URLs, other rules are active on all webpages visited by the user, rendering them fingerprintable by any website.

Example: Fingerprinting the Wikiwand extension

- Wikiwand is a website that optimizes the reading experience on Wikipedia.



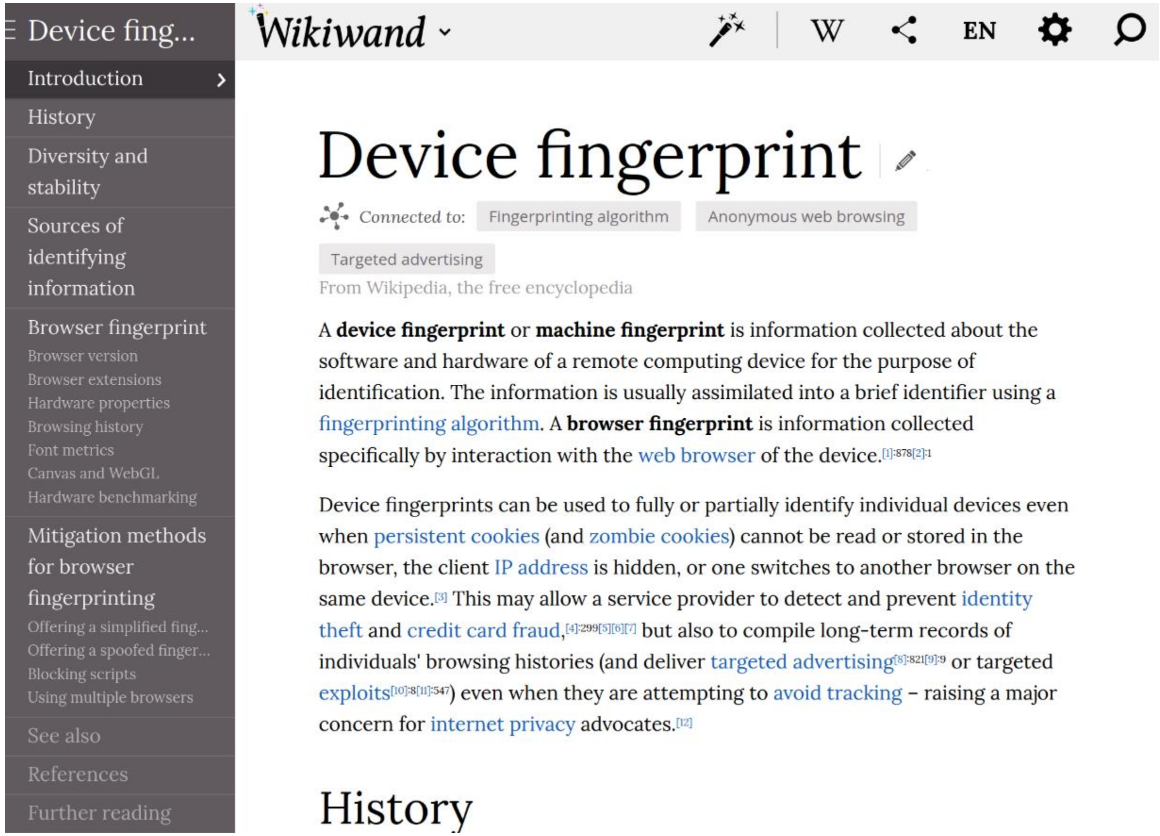
Device fingerprint

From Wikipedia, the free encyclopedia

A **device fingerprint** or **machine fingerprint** is information collected about the software and hardware of a remote computing device for the purpose of identification. The information is usually assimilated into a brief identifier using a fingerprinting algorithm. A **browser fingerprint** is information collected specifically by interaction with the web browser of the device.[1]:878[2]:1

Device fingerprints can be used to fully or partially identify individual devices even when persistent cookies (and zombie cookies) cannot be read or stored in the browser, the client IP address is hidden, or one switches to another browser on the same device.[3] This may allow a service provider to detect and prevent identity theft and credit card fraud,[4]:299[5][6][7] but also to compile long-term records of individuals' browsing histories (and deliver targeted advertising[8]:821[9]:9 or targeted exploits[10]:8[11]:547) even when they are attempting to avoid tracking – raising a major concern for internet privacy advocates.[12]

Contents [hide]



Device fingerprint

Connected to: Fingerprinting algorithm Anonymous web browsing Targeted advertising

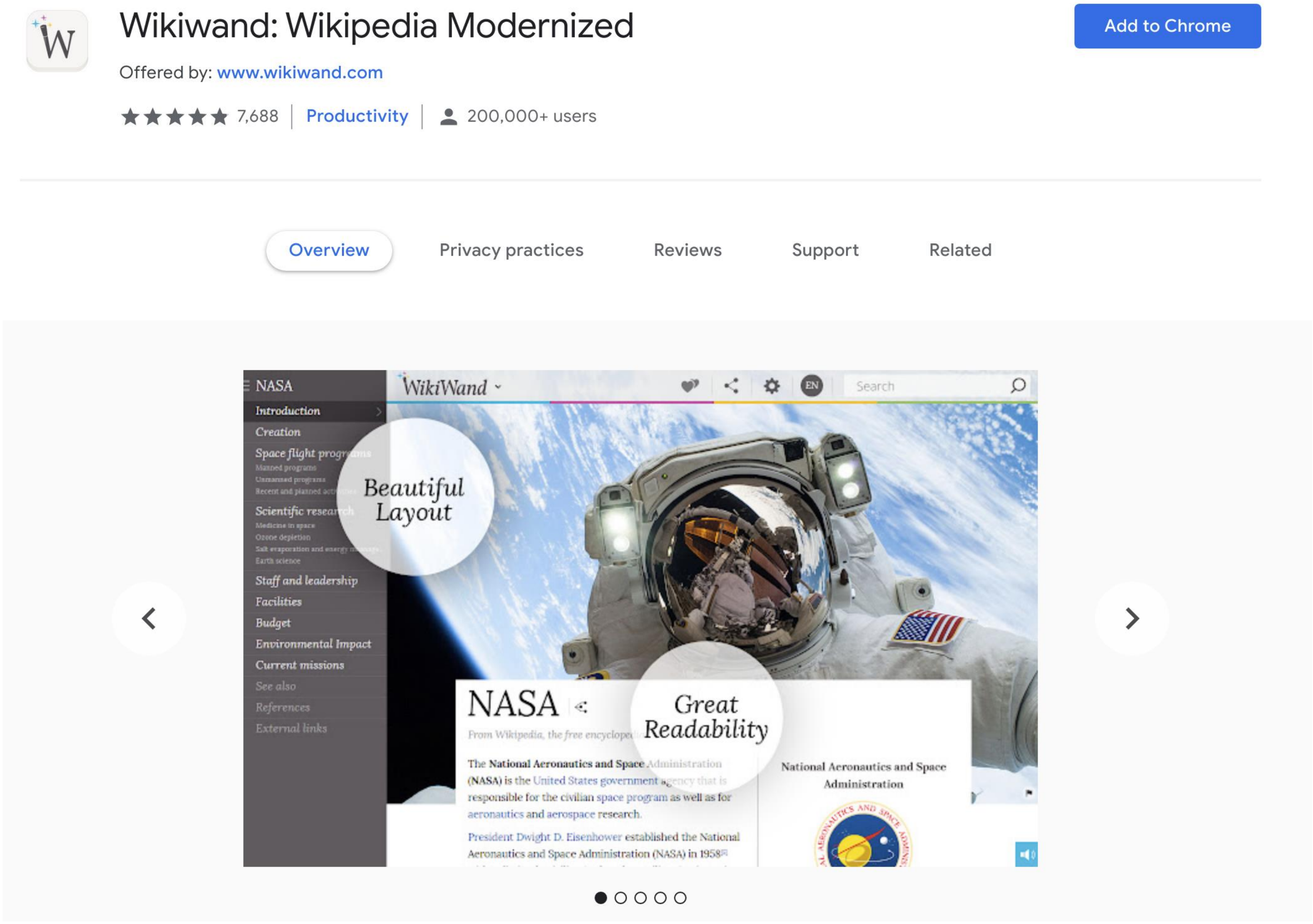
From Wikipedia, the free encyclopedia

A **device fingerprint** or **machine fingerprint** is information collected about the software and hardware of a remote computing device for the purpose of identification. The information is usually assimilated into a brief identifier using a fingerprinting algorithm. A **browser fingerprint** is information collected specifically by interaction with the web browser of the device.[1]:878[2]:1

Device fingerprints can be used to fully or partially identify individual devices even when persistent cookies (and zombie cookies) cannot be read or stored in the browser, the client IP address is hidden, or one switches to another browser on the same device.[3] This may allow a service provider to detect and prevent identity theft and credit card fraud,[4]:299[5][6][7] but also to compile long-term records of individuals' browsing histories (and deliver targeted advertising[8]:821[9]:9 or targeted exploits[10]:8[11]:547) even when they are attempting to avoid tracking – raising a major concern for internet privacy advocates.[12]

History

Example: Fingerprinting the Wikiwand extension



Wikiwand: Wikipedia Modernized

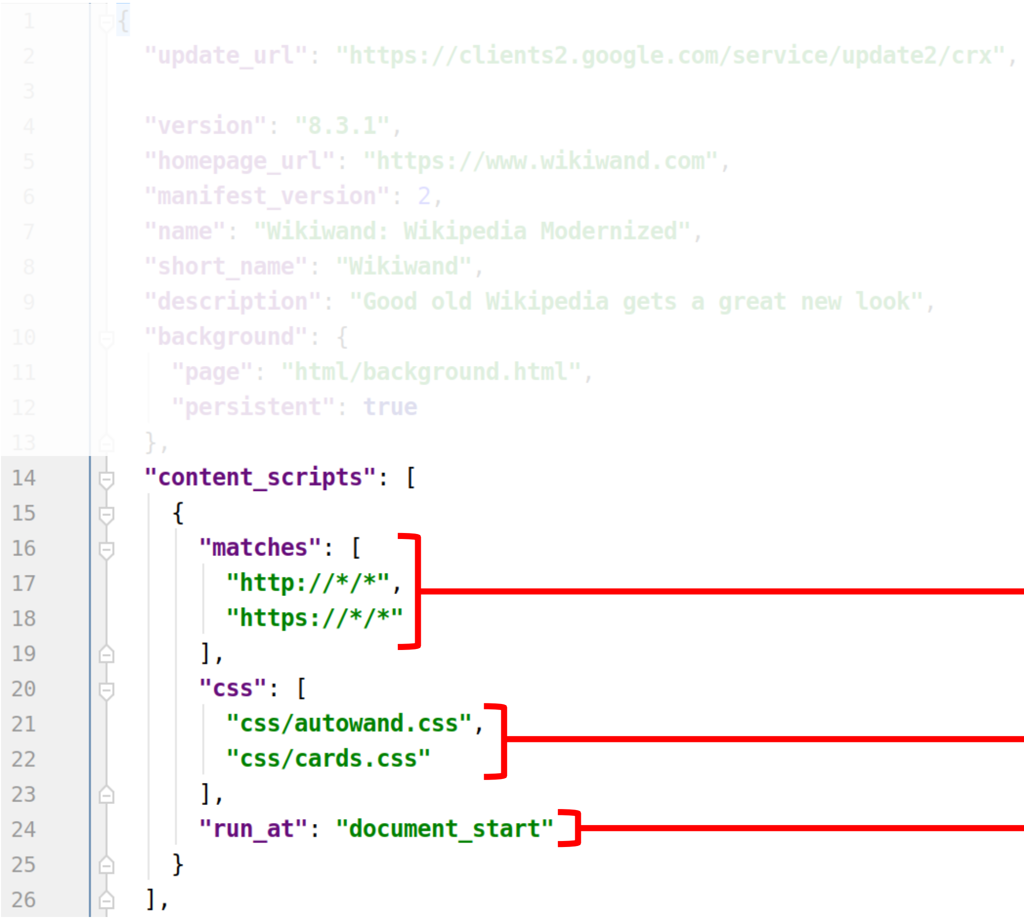
Offered by: www.wikiwand.com

★★★★★ 7,688 | Productivity | 200,000+ users

- The Wikiwand extension replaces Wikipedia links with Wikiwand ones.
- 200,000+ users

Example: Fingerprinting the Wikiwand extension

Manifest.json file



```
{
  "update_url": "https://clients2.google.com/service/update2/crx",

  "version": "8.3.1",
  "homepage_url": "https://www.wikiwand.com",
  "manifest_version": 2,
  "name": "Wikiwand: Wikipedia Modernized",
  "short_name": "Wikiwand",
  "description": "Good old Wikipedia gets a great new look",
  "background": {
    "page": "html/background.html",
    "persistent": true
  },
  "content_scripts": [
    {
      "matches": [
        "http://*/*",
        "https://*/*"
      ],
      "css": [
        "css/autowand.css",
        "css/cards.css"
      ],
      "run_at": "document_start"
    }
  ],
```

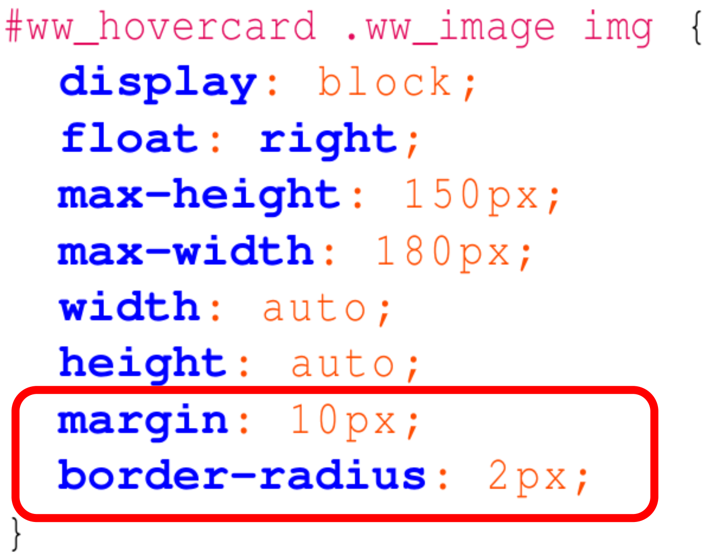
The two CSS files will be injected on all HTTP and HTTPS webpages visited by the user.

- ➢ autowand.css has 18 CSS rules
- ➢ cards.css has 20 CSS rules

The stylesheets are injected as soon as the DOM is loading.

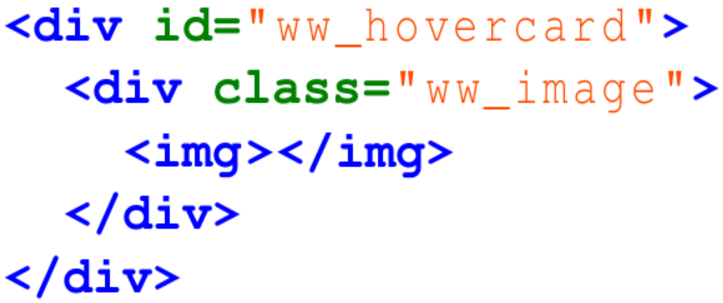
Example: Fingerprinting the Wikiwand extension

One rule in the *cards.css* file



```
#ww_hovercard .ww_image img {
  display: block;
  float: right;
  max-height: 150px;
  max-width: 180px;
  width: auto;
  height: auto;
  margin: 10px;
  border-radius: 2px;
}
```

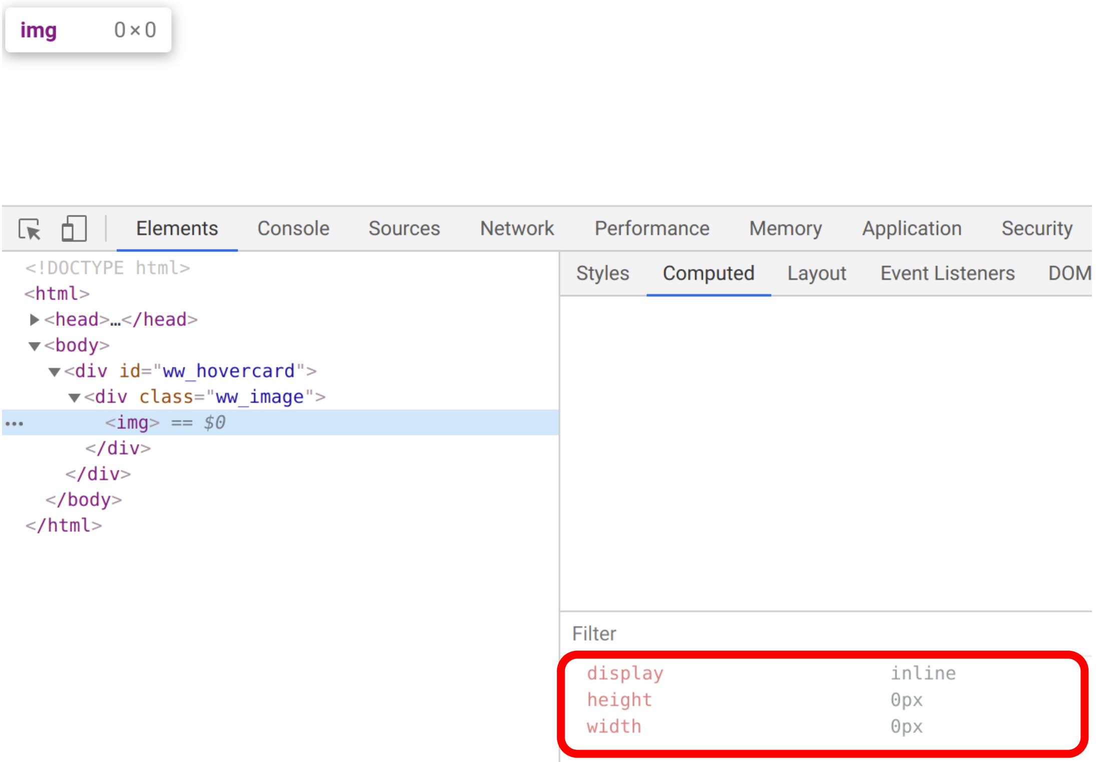
HTML structure associated
with this rule



```
<div id="ww_hovercard">
  <div class="ww_image">
    <img></img>
  </div>
</div>
```

Example: Fingerprinting the Wikiwand extension

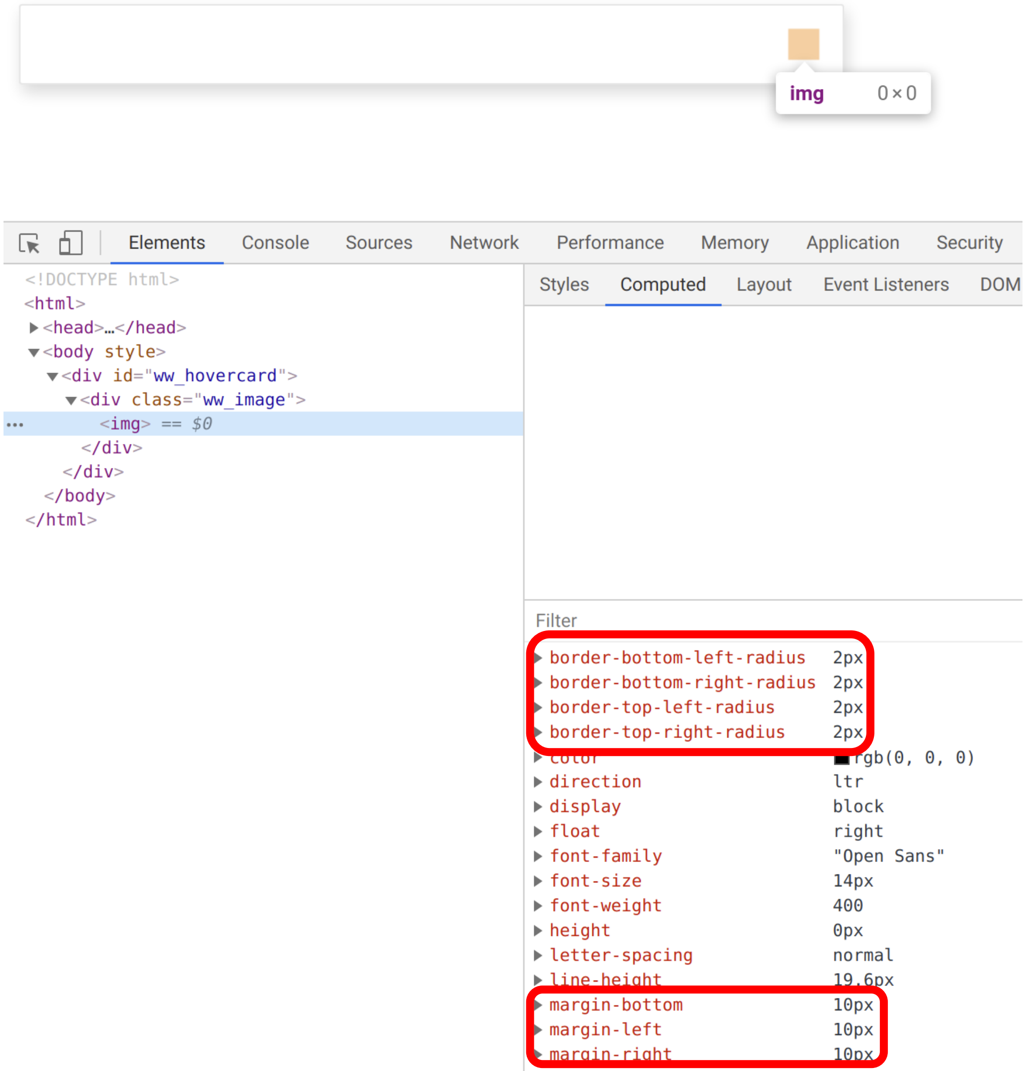
Without the extension installed



```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body>
  ▼<div id="ww_hovercard">
    ▼<div class="ww_image">
...     <img> == $0
      </div>
    </div>
  </body>
</html>
```

```
display                 inline
height                  0px
width                   0px
```

With the extension installed

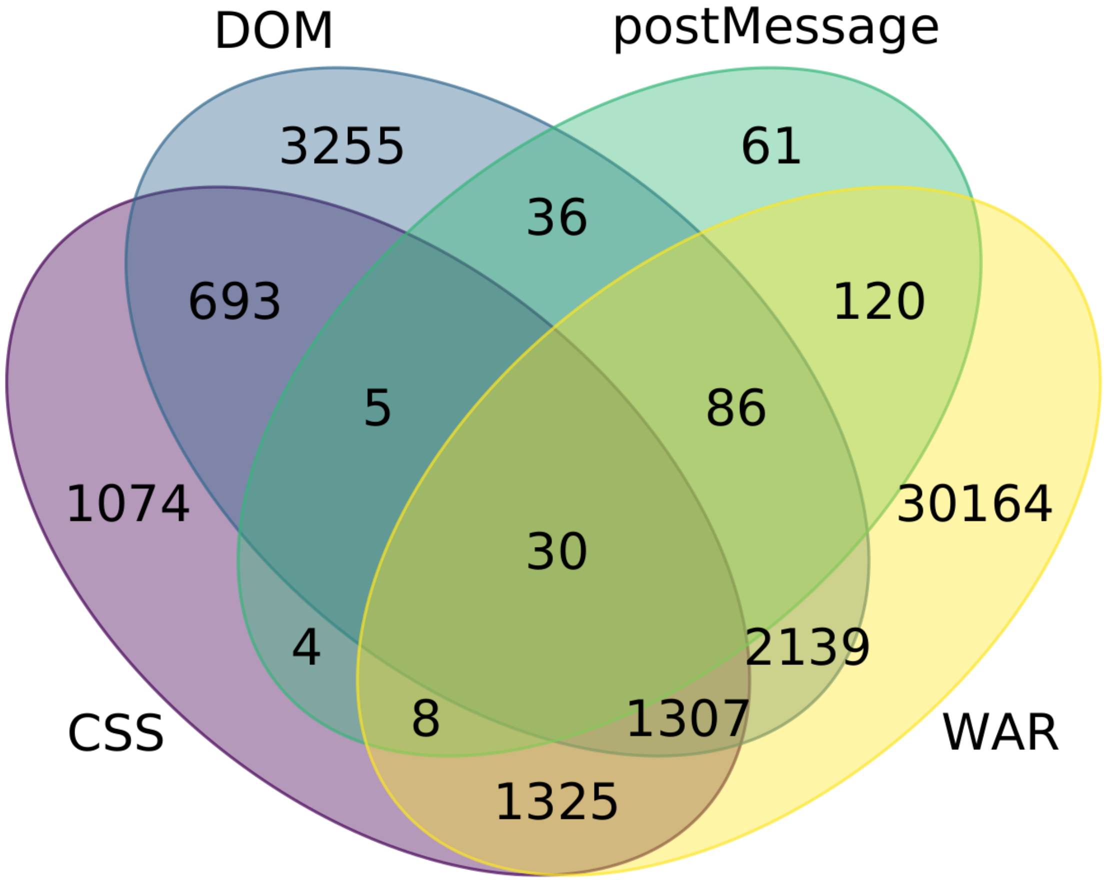


```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body style>
  ▼<div id="ww_hovercard">
    ▼<div class="ww_image">
...     <img> == $0
      </div>
    </div>
  </body>
</html>
```

```
▶ border-bottom-left-radius    2px
▶ border-bottom-right-radius   2px
▶ border-top-left-radius       2px
▶ border-top-right-radius      2px
▶ color                        ■rgb(0, 0, 0)
▶ direction                    ltr
▶ display                      block
▶ float                        right
▶ font-family                  "Open Sans"
▶ font-size                    14px
▶ font-weight                  400
▶ height                       0px
▶ letter-spacing               normal
▶ line-height                  19.6px
▶ margin-bottom                10px
▶ margin-left                  10px
▶ margin-right                 10px
```

Summary of browser extension fingerprinting methods

Number of detectable extensions
(out of 116,485 extensions)



- Each method is complimentary to another one.
- 4,446 extensions are detectable through CSS fingerprinting.
- 1,074 extensions are only identifiable through CSS fingerprinting.

What can you find in the rest of the paper?

- Details on the framework we built to detect fingerprintable extensions
- Longitudinal analysis of the fingerprintability of extensions over time
- Reasons behind the collisions of style fingerprints
- Performance benchmarks
- Details of two defense strategies to protect against style fingerprinting (one with a browser extension and another at the browser level)

Article

https://www.usenix.org/system/files/sec21fall-laperdrix.pdf

Artefact (demo, defense prototype, dataset)

https://github.com/plaperdr/fingerprinting-in-style

Contact

✉ pierre.laperdrix@univ-lille.fr

🐦 @RockPartridge