Putting users in charge: Introduction to the browser privacy UI

Lukasz Olejnik

W3C Invited Expert
Email: lukasz.w3c@gmail.com
Site: http://privatics.inrialpes.fr/ lukasz/
7.11.2015

Abstract. A description of a session for MozFest 2015. `https://github.com/mozilla/mozfest-program/issues/221`

1 Start with an introduction and problem statement: focus on the user

Goal: ensure participants have an intuitive idea about the basic terminology we will use.

People are still having problems with adapting to the constantly changing networked world. Privacy issues are one of the side effects. But their origin is technology.

1.1 Definitions

Useful terms:

- Privacy as cluster of concepts related with data management.
- Anonymity: hiding the sender/receiver/both in communications.
- Transparency: making the user informed about all the things concerning him and the use of his/her data.
- Information Source: “a magic tap” for various types of data. Geolocation API is an example of information source. Example: a website asks your browser “where are you now?” and receives an answer.
- Identification: how to identify the users on the Web?
  - Need to associate identifiers with the users’ browsers to e.g. enable e-commers (“shopping basket”).
  - Can be active (something is set, then read, i.e. cookies) and passive (some browser attributes are being read).
  - Can be relatively easy to manage (like: delete), or very difficult.
    - Example of an identifier with a relatively civilized management: cookies. Yes, cookies.

- Fingerprinting: the art of associating features with the user or his/her browser. Usually passive.
    - Usually difficult to manage (no easy “removal”).
    - Transparency suffers (user don’t know that is even happening).
    - That’s why cookies are actually good (because you can manage them).

Think how Web browsers currently work.

- Useful questions:
    - Do we know what is happening when we visit a Web site?
    - Example: do we understand what data is being sent/received? Who receives them?
    - Who can see the way we browse the Web?
    - Are you aware which information about you are exposed by your browser?

2 Short introduction to privacy and transparency in browsers and web systems

Goal: ensure participants understand the basic information flows during web browsing.

Long topic. And we are short with time. So just one good example.

- When we visit a site, a lot of things happen. Lots of resources are requested. The server can set/read the cookies, etc.
- Some of the requested resources are images, html files.
- Others may be scripts (think: “dynamic programs”). They can do various stuff, e.g.: access some information sources, send the data to some place.
- Example of data: location (GPS), installed plugins, screen resolution, battery level... The actual temperature?

Points for consideration:

- Are you aware of this?
- Should you be aware of this?
- In what way should you be made aware of this?

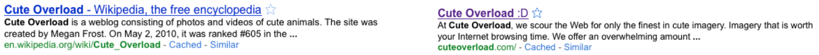
3 Description of real privacy/transparency issues; with questions and discussion with participants

Goal: show examples of real privacy risks. Ensure participants have an understanding of possible consequences.

Let’s start by recalling some of the past privacy issues!

3.1 History hijack: who knows which sites do you visit?

- Browsers apply additional styles to links which the user had visited



Cute Overload - Wikipedia, the free encyclopedia ☆
Cute Overload is a weblog consisting of photos and videos of cute animals. The site was created by Megan Frost. On May 2, 2010, it was ranked #605 in the ...
en.wikipedia.org/wiki/**Cute_Overload** - Cached - Similar

(a) Example links

Cute Overload :D ☆
At **Cute Overload**, we scour the Web for only the finest in cute imagery. Imagery that is worth your Internet browsing time. We offer an overwhelming amount ...
cuteoverload.com/ - Cached - Similar

(b) Example visited links. Is this information intended for you?

Fig. 1: Visited vs Unvisited links

- Example JavaScript. Simple. Effective. This issue is fixed. Other techniques exist.

```
<script>
var r1 = 'a␣{color:green;}';
var r2 = 'a:visited␣{color:red;}';

document.styleSheets[0].insertRule(r1, 0);
document.styleSheets[0].insertRule(r2, 1);

var a_el = document.createElement('a');
a_el.href = "http://foo.org";

var a_style=document.defaultView.getComputedStyle(a_el,"");

if (a_style.getPropertyValue("color") == 'red')
        // link was visited
</script>
```

- Web sites ask your browser: "did you visit this site previously"?
- Technically: retrieve the Web sites browsed by you.

This was used in wild by numerous Web sites.

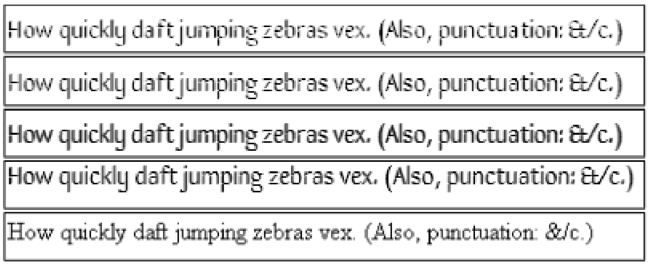
3.2 Identifier assignment

- Browsers expose *canvas elements*, so Web masters can render graphics dynamically.
- Use standard JavaScript/HTML5 Effective.
- Quite simply:

```
<script type="text/javascript">
var canvas = document.getElementById("drawing");
var context = canvas.getContext("2d");
context.font = "18pt␣Arial";
context.textBaseline = "top";
context.fillText("Hello,␣user.", 2, 2);
</script>
```

Fig. 2: Taken From Mowery's paper ("Pixel Perfect").

- You then use a JavaScript *ToDataUrl* method to transform the bitmap representation to base64 encoding.
- What matters is: you retrieve a browser identifier.
- Hardware-dependant. Was used by a third-party company, whose scripts were included on the White House site (among 5% of other popular sites...).



How quickly daft jumping zebras vex. (Also, punctuation: &/c.)

How quickly daft jumping zebras vex. (Also, punctuation: &/c.)

How quickly daft jumping zebras vex. (Also, punctuation: &/c.)

How quickly daft jumping zebras vex. (Also, punctuation: &/c.)

How quickly daft jumping zebras vex. (Also, punctuation: &/c.)

Fig. 3: Taken From Mowery's paper ("Pixel Perfect"). Browsers render a specific image differently. This identifies them.

Examples show certain browser characteristics can be exploited in creative manner to the advantage of *some actors.*

3.3 Discussion of how some of the past issues were adressed in practice (5-10m)

- History hijack fix: it is not possible to programatically read the color associated with a link (But: other techniques still work).
- Canvas fingerprinting is not "fixed". That's how it works...

The general take away for us should be: Guard the information sources. Identify them, analyze consequences. Inform the user (transparency).

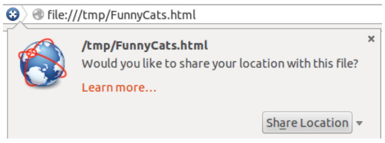
4 Touch upon the role of the W3C, explain how to read web standards even before they're implemented

Goal: familiarize the participants with browser APIs, what those provide and what are the possible side effects.

World Wide Web Consortium (W3C) standardizes the Web. Browsers implement those standards. Recent developments (including HTML5) provided new APIs – new information sources. Is there a need to revise other parts of the browsers?

4.1 Example: API

- Some APIs ("information sources") provide sensitive data.
- It is a good idea to protect them. Whenever used, a prompt pops out ("allow", "deny"). That's how browser permissions work.



file:///tmp/FunnyCats.html

/tmp/FunnyCats.html

Would you like to share your location with this file?

Fig. 4: Standard permission prompt. Can you spot any problems? ["file" can be "site"]

- Why not make permissions for everything? But...
  - Would you like to see *tens of permissions prompts* whenever you visit a web site? No.
- There is a need to find a good solution here.

Task: You are the user. Try to design an interface which you yourself could use. And which would not irritate your grandma. At the same time. Focus: how to track the use of data sources?

Focus on APIs like geolocation and the user controls (permissions). Discuss the pros and cons. Then focus on the user interface and experience (i.e. "how should the browser display that"?). Attempt to identify risks, flaws, inadequacies and possible improvements.

4.2 Discussion of specific browser functionalities

```
<script>
function show(position) {
    alert("Lat:␣" + position.coords.latitude +
    "␣Lon:␣" + position.coords.longitude);
}
navigator.geolocation.getCurrentPosition(show);
</script>
```

Fig. 5: That's how browser APIs work in practice. We get latitude/longitude. Others work similarly.

Points of discussion For your convenience we list some examples of APIs ("information sources").

- APIs: geolocation (coordinates), battery (readout: time to charge, discharge, levels: 0.0-1.0), sensors allowing the readout of temperature, humidity, lighting (in lux units), proximity (closeness to the user) ...
- Example (positive) use case of lighting: Web site changes its appearance on per night/day.
  - It's for you to fill the rest. I'll provide something for the beginning:
    - Possibility of profiling (think: e.g. in advertising/profiling): how frequently do you browse in dark conditions? What does this tell about you? And is it a problem if someone learns this?
- You can try to define: the risks, the expectations of privacy, the transparency requirements.
- Then come up with a solution.

Helpful items. Assume browser APIs for the listed sensors can provide data with the following accuracy:

- Light: [0,100000] lux
- Temperature: [-40,150] C
- Proximity: [0, 152.4] cm

Helpful questions:

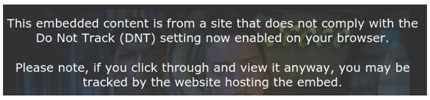
- Is the user aware what is going on?
- Is there a consent involved?

5 Appendix: DNT

5.1 Example: Do Not Track

W3C standardizes the use of Do Not Track header. It expresses a preference: "do not track this user". Currently browsers do not clearly inform if a Web site chooses not to comply. Is it possible to change this?

Interesting case is a Web site - Medium (medium.com):



This embedded content is from a site that does not comply with the Do Not Track (DNT) setting now enabled on your browser.

Please note, if you click through and view it anyway, you may be tracked by the website hosting the embed.

Fig. 6: Medium does inform if the link on their site takes the user to a Web site not respecting DNT

If Medium can show this, why your browser can't?