Shedding light on web privacy impact assessment: A case study of the Ambient Light Sensor API

Lukasz Olejnik
European Data Protection Supervisor, Brussels, Belgium
me@lukaszolejnik.com

***Abstract*—As modern web browsers gain new and increasingly powerful features the importance of impact assessments of the new functionality becomes crucial.**

A web privacy impact assessment of a planned web browser feature, the Ambient Light Sensor API, indicated risks arising from the exposure of overly precise information about the lighting conditions in the user environment. The analysis led to the demonstration of direct risks of leaks of user data, such as the list of visited websites or exfiltration of sensitive content across distinct browser contexts.

Our work contributed to the creation of web standards leading to decisions by browser vendors (i.e. obsolescence, non-implementation or modification to the operation of browser features). We highlight the need to consider broad risks when making reviews of new features. We offer practically-driven high-level observations lying on the intersection of web security and privacy risk engineering and modeling, and standardization. We structure our work as a case study from activities spanning over three years.

I. Introduction

Designing powerful application programming interfaces (APIs) requires care. Assessing the various impacts of new technologies, especially in the already complex environments like the web platform is a challenge. In 2020, all web browser features typically undergo various forms of scrutiny to inspect the impact on security, privacy, accessibility or even human rights. Variations of such reviews or impact assessment processes exist at every web browser vendor. They are also pursued in multi-stakeholder settings like within the IETF or W3C, at the standardization time. Standards for privacy review of emerging or novel web features are still in the process of evolution [1]–[6], essentially a work in progress. Our contribution is another step in this journey.

Modern smartphones and a selection of notebooks are equipped with light sensors. Ambient Light Sensor API [7] enable the web browser to take advantage of this device capability to access lighting condition in the user environment, similarly to what mobile applications can do. While the ability to detect the light level in the user environment potentially introduces ways of building innovative web application features [8], some concerns over misuse of the light sensor on mobile devices included the ability to infer the watched video [9], or even to stealing of bank PINs [10].

In this work, we describe several risks we identified that arise from the unconstrained access to precise raw readouts of the light sensors. We specifically focus on the potential of using the device capability to perform side-channel attacks leading to information leaks induced by attacker-controller websites. Our work had practical impacts. The constructed proof of concepts were instrumental during the standardization and feature development process, with a direct impact on the shaping of the understanding of potential dangers.

This work partly describes the activities from interacting with the wider standardization community over more than three years (2016–2020). In this sense, the work we present here is a unique privacy engineering case study.

II. Related work

Importance to consider privacy when designing internet protocols has been clarified in the seminal *Request for Comment 6973* [3]. The W3C maintains a dedicated security and privacy health-check document meant to help feature developers to consider risks at early stages [4]. In the case of privacy reviews of web standards, the earliest works are those of Dawson [1] and Doty [2]. Impact-oriented case studies validated by data followed, like with the example of the Battery Status API case [5]. The work of Das [6] offer practical insight from the study of website usage of device sensors. While the time of their work coincided with our study, they are separate; the authors link to our demonstrations as external examples.

With many risks demonstrated recently, research into side-channel leaks on the web if flourishing, with examples including keystroke leaks [11] or the information leaked via high precision timers [12]. Web browser history leaks [13]–[15] are perhaps among the most straight-forward way of demonstrating security and privacy impact arising from a new web component.

III. Ambient Light Sensor: case study

Ambient Light Sensor [7] is a web browser API that enables websites to access to the light level conditions as seen by the user device sensors. The illuminance value is expressed in the SI unit, lux. Initially, the API was on the rails of a typical W3C standardization process; the event-based API from 2015 [16] was refactored into Ambient Light Sensor. In 2017 light sensor access in the browser was available via Ambient Light Sensor API (Chrome Canary) and via the previous-generation API via device light events (Mozilla Firefox). Both worked similarly, providing precise readouts with a resolution between 0 to tens of thousands of lux.

```
a:link#tester {
background-color: black;
padding: 100%;
text-align: left;
display: inline-block; }
a:visited#tester  {
position: absolute;
background-color: white;
padding: 100%;
text-align: left;
display: inline-block; }
```

Fig. 1. Example code facilitating the distinguishing between states via light sensor

A. The permissions debate

It is often assumed that permission prompts lead to user fatigue, a phenomenon thought to be similar to the mechanics of users disregarding browser security warnings [17]. For this reason browser vendors are often inclined to favor distraction-less browsing experiences. Reasoning along this line, on 8/04/2017 Chrome developers expressed an intention to expose the Ambient Light Sensor API without gating it behind a browser permission mechanism (i.e. without a prompt) [18]. Problems with such approaches were demonstrated on 19/04/2017 via a proof of concept that suggested to reassess the threat model.

B. Stealing user data with light sensors

While capability to read the light level may result in complex risks [10], such unstructured concerns might be unconvincing. While the role of proof of concepts in security engineering is well established, this model is not always directly transferable to privacy engineering. Light sensor API proved to be the case where such a concrete demonstration was possible.

Single-origin policy (and other mechanisms meant to protect user privacy) prevent access to certain information across distinct browser contexts, e.g. when websites are prevented to read data from embedded third-party contexts like iframes. We devised proof of concepts demonstrating the exfiltration of user data, demonstrating that the inclusion of the light sensor negatively impacts on existing web platform security and privacy model (a regression). This misuse was possible solely due to the introduction of the light sensor API to the web platform.

The attack we devised was a side-channel leak, conceptually very simple, taking advantage of the optical properties of human skin and its reflective properties. Skin reflectance only accounts for the 4-7% emitted light [19] but modern display screens emit light with significant luminance. We exploited these facts of nature to craft an attack that reasoned about the website content via information encoded in the light level and conveyed via the user skin, back to the browsing context tracking the light sensor readings. Attacker website changed the background color via CSS styling, conditioned on the information to be exfiltrated. We observed that the relative differences in the level of few to a few tens of lux between light levels emitted by a screens displaying websites with a full white or full black background were measurable (e.g. a full white page would correspond to the readout of 6 lux, the black page to 0 lux, under dim conditions).

1) Stealing web browsing history with light sensor: The first step in the attack is an initial calibration where the website implements a simple heuristic to associate the detected light levels with different colors of the page content.

The first demonstration was the classic example of web browsing history stealing using the CSS :visited class to style the document background color according to the visited or unvisited state of the link. The proof of concept iterated over a list of websites and dynamically injected an <a> tag with a URL of interest to test the visited or unvisited state of the sites. The CSS styling depicted on Fig. 1 result in a full white page if the website is visited, and full black otherwise. The measurable differences in light levels as reported by the light sensor detected via the Ambient Light Sensor API allowed for a discovery of the state of the website (i.e. visited/unvisited).

This demonstrated a bypass of the standard privacy mechanism [20] responsible for the impossibility of similar leaks via standard JavaScript APIs [14]. Identical mechanics can be reused to exfiltrate other information.

2) Stealing cross-origin resources: Attacker website could fetch objects, such as images, or contents of iframes, containing sensitive information (i.e. bank balance, etc.) and displaying them pixel-by-pixel on a website styled similarly to the ways described in the previous section. In this way, Ambient Light Sensor API allowed to read cross-origin resources, bypassing the standard security model of the web, the single-origin policy.

3) Analysis: Information exfiltration via the light sensor channel extracts one bit of information at a time. This factor determines the practical speed of detection in any similar approaches. While in principle in 2017 web browser sensors could operate at $60Hz$ rate, for practical reasons this did not correspond to an actual ability of extracting 60 bits per second. The rate at which a change in screen brightness happen and can be detected by the sensor are the constraining factors. In our experimentation and demonstrations, we measured the screen brightness-to-readout latency in the range of $200-300ms$. For a practical exploit, we limited to one bit per $500ms$. In the case of web browsing history stealing, such constraints correspond to two websites scanned per second (or 1000 URLs scanned in 8 minutes and 20 seconds). For the exfiltration of information across origins, assuming 6 bits per character string rendered in a known font, the performance would correspond to the extraction of 8-character string in 24 seconds

There are also other environmental considerations affecting the lighting conditions, like as the possibility of the user moving the device such as a mobile phone around, factors such as the screen brightness (i.e. bright screen helps the attack), proximity and angle of a reflecting surface above the sensor

(i.e. a phone lying on a shelf reflecting light from a parallel surface like a mirror or an object, produces good results), the amount of ambient light (darker environments are less noisy and make detection easier), etc.

But the attack was a practical enough a demonstration of an unconsidered, unforeseen and unexpected risk, ultimately demonstrating a bypass of fundamental web security and privacy model. This *template demonstration* stimulated design reviews to continue in a structured manner, focusing on the mitigation of privacy risks in an objective, evidence-based setting.

C. Response

Risks of information leaks have been considered seriously. As of 2020, practical vendor mitigation approach vary from dropping the feature (Mozilla), non-implementation (WebKit), and sanitization of light sensor readout (Chrome).

Firefox 62 deprecated and removed the *devicelight* events which allowed to read the light level without user interaction of awareness [21], [22], previously addressing a potential fingerprinting vector due to an overly precise readout [23]).

The strategy chosen by the Chrome browser was more nuanced. The early considerations for sensors in Chrome considered capping the performance to a resolution of $4bit$ at a frequency of $10Hz$ [24]. Our practical intervention via the demonstration of unexpected privacy risks led developers to practically test the environmental impact of change in lighting conditions, in the end proposing an option of rounding off the readout precision of the sensor to the nearest multiplication of 50 (i.e. using a 50 lux threshold; for example, a true readout of 49 is rounded to 50) [25]. The threshold was chosen following manual tests involving the use of professional light sensors and using our proof of concepts as a benchmark test to determine a choice of the threshold. Version 80 of Chrome browser ship with Ambient Light Sensor with this minimization strategy. As of this writing access to Ambient Light Sensor remains gated behind flags; the sensor is not available in default settings [26].

1) Other considerations:

- One potential minimization approach is the use of finite enumeration-like states of light conditions, such as: *"dark", "dim", "light"*. Similar scheme is to be provided by *'light-level'* feature of the Media Queries Level 5 [27]. But Ambient Light Sensors API specification drafters determined that the information offered by the Media Queries API is not similar enough to use it as a full replacement mechanism for the ambient light sensor.
- A separate issue identified during the work on the standard was that no satisfactory agreement over the actual use cases for the Ambient Light sensors [28] existed, a matter that became problematic in light of the potential privacy risks. Eventually, potential use cases were identified.

IV. High level takeaways

We believe distilling of the *lessons learned* as conclusions with future in mind is important. Privacy engineering is a research and engineering art in development; in this sense, we enrich the elements of the previous works [1]–[3], [5]. Based on the aforementioned case study, we highlighting points potentially worth to consider in future data protection or privacy impact assessments concerning new technologies or features, particularly those related to web or mobile ecosystems.

A. Viewing risk in a broad sense

Interest in equipping the Web platform with functionality often already available to mobile apps may become significant in context of the Progressive Web Application paradigm. Due to the web complexity and differences to the native mobile apps model, novel web features may surface unexpected security or privacy risks. Considering platform and ecosystem specific constraints focus may be put on both typical as well and the the less conventional risks.

Some concerns [10] may be insufficient to demonstrate a convincing practical case. But more directed corner cases, like the information leaks and (out-of-band) data exfiltration described in this paper, might help in constructive ways. Outside-the-box approach to risk scenarios is beneficial.

B. Role of demonstrations to show a clear case to discuss

While privacy engineering and security engineering are not the same. But like in security domain, objective criteria helps to identify risk points, as well as to track and measure the improvement or regressions. We constructed a clear and objective edge case that clearly showed problematic issues, allowing the discussions over privacy concerns to be put in a rational and objective plane.

Prudence is needed to keep in mind the special considerations related to cases that may be challenging to quantify, such as some aspects of some data protection, privacy or human rights considerations. Accepting that it might not always be possible to craft such a universal, clear and demonstrable problem point, may be justified.

C. Interact between research, standardization, developers

Close cooperation and direct interactions between researchers, security and privacy engineers, analysts, standardization and feature developers help in effective identification of undesirable aspects that could lead in privacy risks, allowing for satisfactory resolution; in this sense, we reaffirm some observations from [5]. The consensus-driven approach to web standards development might have helped to drive the debate towards seriously considering the risk scenarios, by involving a diverse group of stakeholders.

Such an interactive process may be time-consuming. Raised concerns may result in a sizable number of conversations. We are aware of more than 10 different discussion threads taking place over different channels (i.e. GitHub, per-vendor bug repository systems, mailing lists, and other places) about the Ambient Light Sensor API problems. While each such thread may contain unique insights, some participants are unable to take part in all of them. We perceive it as a feature of the web platform development nature. Standardization fora may be the contact hub.

D. Risk-benefit considerations for new features

Some APIs providing sensitive information deserve special attention and protection. This is the case with numerous APIs gated behind permissions (e.g. access to the camera where explicit permission is needed), picker-like mechanics (e.g. access to files via Filesystem API is based on individual files; similar exists behavior in case of pairing with wireless devices via the Bluetooth API). In a reality where no objective requirements exist that indicate when a feature should be subject to the protective mechanism, it is solely at the designer discretion, and subject to reviews. During this study, we witnessed that participants taking part in consensus-driven standardization work may face difficulties to convince others when 1) unconstrained access to a potentially sensitive API is or is not justified, 2) that it is or is not a security or privacy issue. But prior to an expansion of the risk surface, importance of use cases and motivations should be clear and evident. Identified early in the development, before deciding about the exposure of the new API, in line with the risk-benefit assessment.

V. Conclusion

Stealing web browsing history with a light sensor is not expected. Perhaps even difficult to believe. Website's ability to steal the list of user visited websites or exfiltrating data with a light sensor is highly counter-intuitive. We performed a deep and out of the box privacy risk assessment of the ambient light sensor API, and in an ultimate result, **risks were identified, and averted**. Our initial motivations were grounded in concerns arising from the potentially inconsiderate desire of exposing a powerful feature. Our intervention led to a healthy discussion within the consensus-based standardisation and browser vendor communities, which ultimately led to a safer part of the web ecosystem. No web browser did decide to ship the API, at least not without special considerations.

That said, we point out that in general, the introduction of the system of explicit user permissions do not fundamentally change the nature of the sensitive data.

Even small changes or innocuous features have a potential of back-firing in unexpected ways. Privacy engineering is still a nascent field. Privacy remains a complex problem set spanning technology, regulation, human rights, and more – including local or cultural considerations. Due to this complexity, not all problems can be objectified in a clear technical way. This does not mean it is always impossible. We believe the case in this paper offers insight in a broader light.

References

[1] F. Dawson, "Specification Privacy Assessment (SPA) ," https://yrlesru.github.io/SPA/, 2013.

[2] N. Doty, "Reviewing for privacy in internet and web standard-setting," in *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, 2015, pp. 185–192.

[3] A. Cooper, H. Tschofenig, B. Aboba, J. Peterson, J. Morris, M. Hansen, and R. Smith, "Rfc 6973 — privacy considerations for internet protocols," IETF, Tech. Rep., 2013.

[4] L. Olejnik and J. Novak, "Self-Review Questionnaire: Security and Privacy," https://w3ctag.github.io/security-questionnaire/, 2019.

[5] L. Olejnik, S. Englehardt, and A. Narayanan, "Battery status not included: Assessing privacy in web standards." in *In 3rd International Workshop on Privacy Engineering (IWPE'17). San Jose, United States*, 2017.

[6] A. Das, G. Acar, N. Borisov, and A. Pradeep, "The web's sixth sense: A study of scripts accessing smartphone sensors," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1515–1532.

[7] A. Kostiainen, "Ambient Light Sensor," http://www.w3.org/TR/ambient-light/, 2019.

[8] R. H. Venkatnarayan and M. Shahzad, "Gesture recognition using ambient light," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–28, 2018.

[9] L. Schwittmann, V. Matkovic, T. Weis *et al.*, "Video recognition using ambient light sensors," in *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2016, pp. 1–9.

[10] R. Spreitzer, "Pin skimming: Exploiting the ambient-light sensor in mobile devices," in *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, 2014, pp. 51–62.

[11] M. Lipp, D. Gruss, M. Schwarz, D. Bidner, C. Maurice, and S. Mangard, "Practical keystroke timing attacks in sandboxed javascript," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 191–209.

[12] M. Schwarz, C. Maurice, D. Gruss, and S. Mangard, "Fantastic timers and where to find them: high-resolution microarchitectural attacks in javascript," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 247–267.

[13] M. Smith, C. Disselkoen, S. Narayan, F. Brown, and D. Stefan, "Browser history re: visited," in *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[14] A. Janc and L. Olejnik, "Web browser history detection as a real-world privacy threat," in *European Symposium on Research in Computer Security*. Springer, 2010, pp. 215–231.

[15] Z. Weinberg, E. Y. Chen, P. R. Jayaraman, and C. Jackson, "I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks," in *2011 IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 147–161.

[16] D. Turner and A. Kostiainen, "Device and Sensors Working Group," https://www.w3.org/TR/2015/WD-ambient-light-20150903/, 2015.

[17] D. Akhawe and A. P. Felt, "Alice in warningland: A large-scale field study of browser security warning effectiveness," in *Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13)*, 2013, pp. 257–272.

[18] pozdnyakov, "Relax requirements for asking permissions in sensors," https://github.com/w3c/sensors/issues/174, 2017.

[19] R. R. Anderson, J. A. Parrish *et al.*, "The optics of human skin." *Journal of investigative dermatology*, vol. 77, no. 1, pp. 13–19, 1981.

[20] L. D. Baron, "Preventing attacks on a user's history through css: visited selectors," *Tillgänglig på Internet: http://dbaron. org/mozilla/visited-privacy [Hämtad 15.03. 29]*, 2010.

[21] Firefox Site Compatibility, "Various device sensor APIs are now deprecated," https://www.fxsitecompat.dev/en-CA/docs/2018/various-device-sensor-apis-are-now-deprecated/, 2018.

[22] J. Kingston, "Disable devicelight, deviceproximity and userproximity events ," https://bugzilla.mozilla.org/show_bug.cgi?id=1359076, xxx.

[23] "Bug 1299454: Round Off Ambient Light Sensor event.value," https://bugzilla.mozilla.org/show_bug.cgi?id=1299454, 2016.

[24] "Sensor api permissions ux (public)," https://docs.google.com/document/d/1XThujZ2VJm0z0Gon1zbFkYhYo6K8nMxJjxNJ3wk9KHo/edit, 2019.

[25] A. Shalamov, "Security and Privacy considerations for ALS," https://github.com/w3c/ambient-light/issues/13#issuecomment-302393458, 2018.

[26] L. Olejnik, "Issue 642731: Round Off Ambient Light Sensor event.value," https://bugs.chromium.org/p/chromium/issues/detail?id=642731, 2016.

[27] D. Jackson, F. Rivoal, and T. A. Jr., "Media Queries Level 5 ," https://drafts.csswg.org/mediaqueries-5/, 2020.

[28] A. Konstiainen, "Provide "Use Cases and Requirements" section," https://github.com/w3c/ambient-light/pull/26, 2017.