# Battery Status Not Included: Assessing Privacy in Web Standards

Lukasz Olejnik
University College London
lukasz.w3c@gmail.com

Steven Englehardt*
Princeton University
ste@cs.princeton.edu

Arvind Narayanan
Princeton University
arvindn@cs.princeton.edu

## 1. INTRODUCTION

The standardization process is core to the development of the open web. Until 2013, the process rarely included privacy review and had no formal privacy requirements. But today the importance of privacy engineering has become apparent to standards bodies and browser vendors. Standards groups now have guidelines for privacy assessments, and are including privacy reviews in many new specifications. However, the standards community does not yet have much practical experience in assessing privacy.

In this talk we provide a case study of the W3C Battery Status API — reviewing its evolution from the initial specification to the eventual removal of the API from major browser engines following privacy concerns, an unprecedented move. We provide context for these decisions, analyze the standards processes that led to them, and make recommendations for improving the privacy review process for web standards.

This talk builds on a research paper published at the 2017 International Workshop on Privacy Engineering[1]. The talk extends beyond the paper to address new insights, including those from discussions held at the Dagstuhl Seminar on Online Privacy and Web Transparency.

## 2. THE BATTERY STATUS CASE STUDY

Since 2013 the standards community has made numerous substantial changes to the review process to address privacy throughout feature development, which includes the creation of specialized methodologies and working groups for such assessments [3, 5, 7]. These advances are timely, as new and proposed web features will provide websites with much deeper access to the user's device and environment, especially on smartphones and Internet-of-Things (IoT) devices. In this talk we analyze these processes through a case study of the Battery Status API, present new empirical evidence to provide additional context for the case study, and extract recommendations for privacy reviews in future standards.

The Battery Status API offers an interesting and unusual case study of privacy impact assessment in the web standardization process. The specification started with a typical progression: it went through a few iterations as a draft, was quickly implemented by multiple browser vendors, and then progressed to a candidate recommendation — one which characterized the privacy impact as "minimal". Several years later, after it was implemented in all major browser engines and was nearing finalization, researchers discovered several privacy vulnerabilities as well as misuse in the wild. In an unprecedented move, two of the three major browser engines removed support for the API and another browser moved to an opt-in model. Figure 1 provides an overview of developments related to the API.

In the talk, we focus our discussion on the impact of privacy research on the specification itself and adoption of the API by vendors. Privacy research by Olejnik et al. showed that the API can be used to track users, both by using the status readouts as short-term identifiers and by using repeated readouts to reconstruct the battery capacity [6]. In response to this analysis, the specification was updated – changing the privacy considerations section from identifying a "minimal" risk to detailing specific protection recommendations. At the same time, both Firefox and Chrome shipped patches to reduce the effectiveness of the attacks. Despite these changes, Englehardt and Narayanan found two scripts, together present on 22 sites, that used battery status as part of a device fingerprint [4]. In response to these findings, as well as concerns of second-order privacy impacts (e.g. dynamic pricing based on a device's battery level), both Mozilla and Webkit removed support for the API.

Was this the right decision for the web? What process failures exist that allowed the API to be designed, implemented, and deployed, only to be recalled several years later due to privacy concerns? These are the questions we analyze in our talk. We hope to spur a discussion between standardization bodies, browser vendors, privacy engineers, researchers, and web developers around these questions with the goal of improving the privacy review process.

## 3. RECOMMENDATIONS & DISCUSSION

We extract a set of good privacy engineering practices and make several concrete recommendations on how standards bodies can improve the standardization process. We draw from our research, our participation in standardization efforts, our assessments of specifications. While we hope that our recommendations improve the privacy review process, we are confident the issues we've identified will benefit from a larger discussion within the privacy research community. The remainder of our talk summarizes our recommendations and points to open questions and directions for future research.

**Information exchange between vendors and researchers is essential to privacy assessments.** Research can reveal theoretical privacy risks and their exploitation in the wild; it can also provide data on the usage of features on the web. As our case study illustrates, the specification process can benefit from a deeper connection to research. Deliberate attempts to break the privacy assumptions of specifications should be actively incentivized — perhaps by funding attack research, or by organizing a forum for academics and researchers to publish their privacy reviews.

*This author will be the presenting author at HotPETs.

[1]Available at: `senglehardt.com/papers/iwpe17_battery_status_case_study.pdf`

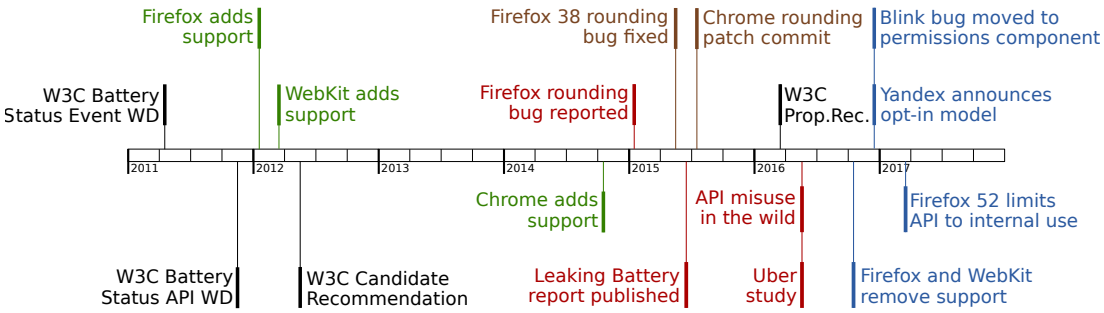


**Figure 1: Timeline of events.**

**The specification process should include a privacy review of implementations.** On the modern web, proposed features often get deployed rapidly. By the time a spec is finalized, it is common for several vendors to already fully support a feature; in fact, the W3C requires at least two implementations to exist before official recommendation. We recommend that specification authors study implementations to prepare higher quality privacy assessments. Implementations enable field testing of theoretical attacks and can be examined for potential API misuses.

With the Battery Status API, the privacy risk stemmed from a difficult-to-predict interconnection of software layers, namely the browser acquiring information from the operating system in order to supply it to a web script. Such a risk is difficult to predict during the design phase, but becomes much easier to identify with access to an implementation.

**API use in the wild should be audited after implementation.** In removing the Battery Status API from Firefox, Mozilla was influenced by the paucity of legitimate uses of the API in the wild [2]. This underscores the importance of analyzing the early use of an API after deployment. Measurement studies have continually shown that fingerprinting scripts are often early adopters of a new API [4, 1]. The benefits of doing an early audit are two-fold: misuses of the API that weren't found during the privacy assessment may be discovered, and any uncovered vulnerability can be fixed at the specification level before web compatibility and breakage become a concern.

In the past, fingerprinting abuse in the wild has been primarily measured by the academic research community [4, 1]. As research on fingerprinting starts to lose its novelty, academic researchers may lose the incentive for frequent measurements of fingerprinting abuse. As a replacement, we suggest measurement through built-in browser probes or a dedicated web crawling infrastructure run by browser vendors or privacy advocacy groups.

**Browser vendors have not given up on fingerprinting — specifications should continue to address the concern.** Many new APIs increase the fingerprinting surface of the browser. Privacy discussions of those APIs will inevitably arrive at the argument that the browser's fingerprinting surface is already large enough that any increase in that surface is inconsequential. This contradicts the W3C's own stance on device fingerprinting [5], which considers it harmful to the web and suggests authors take steps to minimize the fingerprinting surface of any new feature. The decision by Mozilla and WebKit to remove the Battery Status API due to privacy concerns shows that browser vendors are also continuing to support this stance.

**Specification authors should carry out privacy assessments with multiple threat models.** Our case study shows how a seemingly innocuous mechanism can introduce privacy risks. The original 2012 specification of Battery Status API characterized the fingerprinting risk as "minimal", but did not include any analysis of that risk. An enumeration of the possible fingerprinting approaches, even if minimal in expected effectiveness, may have helped avoid the blind spot. We recommend that if any privacy vulnerability is identified, possible exploitation should be modeled and analyzed in detail.

Specification authors must also enumerate and analyze all relevant threat models. Some implementers, such as the Tor Browser, operate under much stricter threat models. For example, most implementers may find it acceptable to reveal the user's operating system through a new API. But not the Tor Browser, as it attempts to maintain a uniform fingerprint across all devices.

**In addition to providing guidance to browser vendors, specifications should include advice for web application developers.** Web developers are ultimately the end consumers of new features and are responsible for complying with local data protection regulations. To assist these developers, specifications should highlight if a particular feature provides sensitive data. Including this information in a specification will also assist browser vendors in properly documenting the APIs.

# 4. REFERENCES


[1] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of CCS*, 2014.

[2] Chris Peterson. Removing the Battery Status API? `https://groups.google.com/forum/#!msg/mozilla.dev.platform/5U8NHoUY-1k/9ybyzQIYCAAJ`, 2016.

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

[4] S. Englehardt and A. Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1388–1401, New York, NY, USA, 2016. ACM.

[5] M. Nottingham. Unsanctioned Web Tracking. `https://www.w3.org/2001/tag/doc/unsanctioned-tracking/`, 2015.

[6] L. Olejnik, G. Acar, C. Castelluccia, and C. Diaz. The leaking battery. In *Data Privacy Management*, 2015.

[7] W3C. Privacy Interest Group Charter. `https://www.w3.org/2011/07/privacy-ig-charter`, 2011.