Battery Status Not Included: Assessing Privacy in Web Standards

Lukasz Olejnik
University College London
lukasz.w3c@gmail.com
http://lukaszolejnik.com

Steven Englehardt
Princeton University
ste@cs.princeton.edu
http://senglehardt.com

Arvind Narayanan
Princeton University
arvindn@cs.princeton.edu
http://randomwalker.info

***Abstract*—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 such as the W3C as well as to 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 paper we systematically analyze the W3C Battery Status API to help inform future privacy assessments. We begin by reviewing its evolution — the initial specification, which only cursorily addressed privacy, the discovery of surprising privacy vulnerabilities as well as actual misuse in the wild, followed by the removal of the API from major browser engines, an unprecedented move. Next, we analyze web measurement data from late 2016 and confirm that the majority of scripts used the API for fingerprinting. Finally, we draw lessons from this affair and make recommendations for improving privacy engineering of web standards.

I. Introduction

The Battery Status API offers an interesting and unusual case study of privacy 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. In this paper, authored by several of the researchers who discovered the privacy problems, we reflect on these events, present new empirical evidence of misuse, and extract recommendations for privacy reviews in future standards.

Before 2013, there were no formal review processes at the major standards bodies to address privacy during the design and standardization of web features. This is reflected in the lack of privacy or security considerations in many past specifications. However, the standards community has recently made numerous substantial changes to address privacy during the early stages of feature development. In 2013 the IETF formally defined recommendations for privacy assessments during the development of internet protocols [1]. The W3C has also invested considerable resources into the creation of specialized methodologies for such privacy assessments [2], [3]. This includes taking public stances on privacy expectations [4] and defining new groups and processes to evaluate privacy [5], [6], [7]. Even the frameworks used during specification, development, and publishing have been updated to encourage all specification authors to include privacy and security review sections [8], [9].

These recent advances in privacy review 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. Examples include Bluetooth connectivity [10], low-level device sensors such as ambient light, acceleration, and vibration [11], [12], and even the user's interpupillary distance, in the context of Virtual Reality [13].

But why should standards consider privacy at all, rather than leave it to browser vendors? Perhaps the market will then allow each user to choose a browser that provides his or her preferred trade-off between privacy and functionality. This argument is beguiling but simplistic — standards must fix many design choices to ensure interoperability, and some of these will inevitably impact privacy [1]. Indeed, there are many examples of standards that impede efforts by browser vendors to improve privacy without causing breakage [14]. Further, standards allow setting a privacy floor across implementations. This helps avoid a race to the bottom, considering that privacy might be a "market for lemons" [15] and is susceptible to numerous behavioral biases [16].

In this paper we study the privacy engineering aspects [17] of the Battery Status API, with a focus on the standardization and implementation process. Specifically,

1) We conduct a systematic case study of the Battery Status API (Section III).
2) We provide new measurements of Battery Status API use on the web (Section IV).
3) We extract useful privacy engineering practices and provide recommendations to improve the design process (Section V).

We hope our work will be useful to standardization bodies, browser vendors, privacy engineers, researchers, and web developers. We seek to enrich Privacy Impact Assessment (PIA)

methodologies, with possible applications to other domains including IoT and mobile APIs.

II. Background and Related Work

The W3C standardization process. The W3C employs a *maturity level* model in the standardization process [18]. Specifications start as a community group Working Draft and may undergo several revisions while the scope and content is refined. Once the specification is ready for a final review by a wide audience, it will progress to a Candidate Recommendation. The W3C formally calls for implementations at this stage, although in practice they may already exist. Feedback from the Candidate Recommendation and experience gathered from implementations is used to refine the specification further. If the specification requires no substantive changes it will progress to a Proposed Recommendation. After a final set of endorsements a specification will progress to a full W3C Recommendation.

The lengthy standardization process is consensus-driven. The stakeholders of a standard are generally organized into Working Groups, typically comprised of employees of browser vendors and other technology companies. To reach consensus, all members must agree on a decision. Other Working Groups, such as those specializing in privacy, accessibility, or web architecture may give their input on aspects of the specification relevant to their mission. Additionally, the specification Working Group must provide evidence of wide review, which includes reviews by a number of external parties: the public (i.e. researchers) and NGOs (some of whom are W3C members).

Privacy reviews often happen during the draft stage, although the depth of reviews can vary. As of 2017, the official W3C Process Document [18] does not require a privacy review. In practice, reviews are often performed prior to a draft entering the Candidate Recommendation level. A privacy consideration section can be *normative*, in which the statements included are requirements that an implementation must follow to be compliant with the specification. Alternatively, the section can be *non-normative*, which is used to provide extra information, context, or recommendations that an implementation is not required to adhere to.

The W3C's Technical Architecture Group (TAG), which aims to build consensus on principles of web architecture, authored the Security and Privacy Self-Review Questionnaire [7]. The questionnaire exists to help authors, TAG, and others assess the privacy and security impacts of a specification. It recommends that authors review their specifications under several different threat models and asks a series of questions related to data access and quality. The W3C also has the Privacy Interest Group (PING), which provides guidance and advice for addressing privacy in standards [5].

W3C privacy assessment practices and requirements. Privacy reviews in specifications often focus on how the proposed design impacts web tracking. Past studies have shown that trackers frequently use many browser technologies to track users: by using stateful mechanisms like cookies, localStorage, and Flash storage to respawn cleared identifiers [19], [20], [21], [22], using "enriched" headers that contain tracking IDs injected by ISPs [23], and by identifying a device solely by its properties, i.e. device fingerprinting [24], [25], [26], [27]. The W3C's TAG has identified these advanced tracking behaviours as "actively harmful to the Web, because [they are] not under the control of users and not transparent" [4].

In this case study, we examine how the Battery Status API can be used to fingerprint devices (Section III-C). Device fingerprinting is the process of identifying a device by its set of features, rather than by setting a persistent identifier on the device. The effectiveness of tracking increases as a device's feature set is more unique. Past studies have demonstrated that a majority of both desktop and mobile users have a unique fingerprint [28], [29]. In response to fingerprinting concerns, the W3C's PING released a Working Group Note to provide guidance to specification authors on how to address and mitigate fingerprinting in their specifications [30].

When data is identified as potentially sensitive, such as that which relates to the user's device, behavior, location, or environment, various W3C specifications have applied different restrictions on access to that data. Some specifications have made the data available only in the top level browsing context (i.e. where access from third-party scripts is limited) [31], and others provide data only in a secure context (i.e. among other restrictions, requiring TLS) [32]. This type of data access also frequently requires user permission before any potentially sensitive information is made available. The Web Permissions API is a draft specification of a mechanism that allows users to manage these types of permissions in a user-friendly way [33].

Past privacy assessment research. Several studies have examined how privacy assessments are conducted as part of the specification process. Nick Doty identifies and addressees the challenges of privacy reviews in standardization bodies [2]. Doty describes the history of security and privacy consideration sections in Request for Comments (RFC), IETF specifications, and W3C specifications. RFC 6973 describes how design choices in internet protocols may impact privacy, and provides guidelines for drafting of Privacy Considerations sections in RFC documents [1]. Similarly, Frank Dawson describes a methodology for drafting the privacy considerations sections of W3C standards [3]. Dawson highlights the importance of privacy assessments during each stage of a draft specification and the need for an open process to incorporate the findings of external research.

III. Case study: Battery Status API

A. API specification

The Battery Status API is a browser mechanism that provides access to the power management information of a device [34]. An example use case listed in the W3C specification is responding to low battery levels. For instance, an online word processor might auto-save more frequently when the battery is low. The API provides the following information: the battery charge `level` (e.g. 0.43 when the battery has 43%

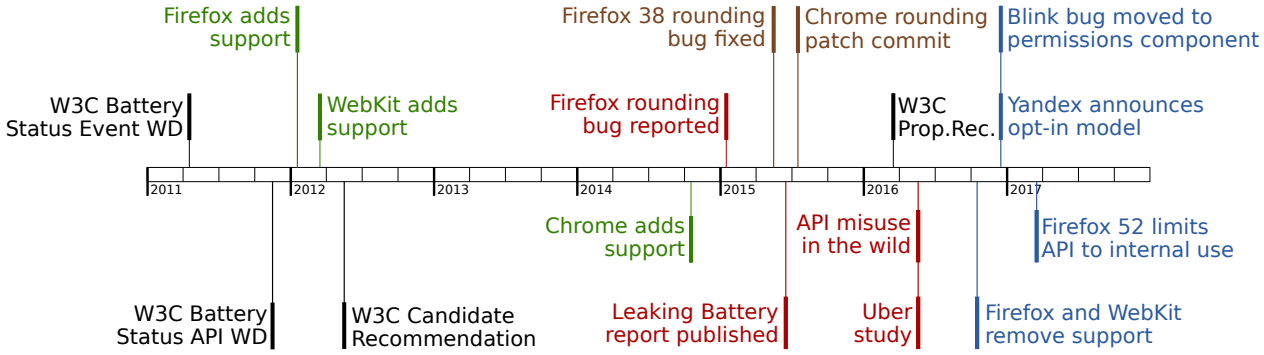

Fig. 1: Timeline of events.

remaining power), the `charging` status (whether or not the device is charging), the time in seconds to a full discharge (`dischargingTime`; when discharging) or a full charge (`chargingTime`; when charging).

In April 2011, the Battery Status Event Working Draft first described how a website can access battery information [35]. The specification was reworked into the Battery Status API in November 2011 [36] and progressed to a Candidate Recommendation in May 2012 [37]. This version identified a "minimal impact on privacy or fingerprinting", but suggested "the user agent can obfuscate the exposed value in a way that [websites] cannot directly know if a hosting device has no battery, is charging or is exposing fake values." [38]. This shows that although the specification authors felt the privacy impact was minimal, they felt there was enough of a risk that user agents (browsers) may want to hide a user's true battery status.

B. Adoption by browser vendors

The first implementations of the Battery Status API were in 2012 by Firefox [39] (mobile and desktop) and WebKit [40]. Chrome[1] also started an implementation in 2012 that was never completed, citing a general lack of interest and convincing use cases [41]. Chrome later removed and re-implemented it in 2014 on both mobile and desktop [41]. Other browsers based on Chrome's Blink engine supported the API soon after, such as Opera in 2014 [42]. This level of implementation fulfills W3C requirements of having at at least two independent implementations prior to finalization of a specification [18]. For a summary of support by additional browsers, see Table I.

C. Discovery of privacy vulnerabilities

In 2015 Olejnik et al. [43] examined the W3C specification and the browser implementations of the API and found several privacy vulnerabilities. They 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 on certain platforms.

[1]Chromium and Chrome are both based on the Blink rendering engine and expose the same Battery Status API. We refer exclusively to Chrome for the remainder of the paper but our analysis is applicable to both browsers.

Battery status readouts made users of certain platforms vulnerable to tracking across sites by a third-party. The API's charge `level` returns a value between 0 and 1 which represents the ratio of the current charge remaining to the maximum capacity. The researchers found that Firefox on Linux returned a double-precision floating-point value for charge level, the result of returning the operating system's value directly without truncating the result. This means that there is a large number of possible values for the charge level. Thus, if a tracker were to see the same charge level readout on two different sites at the same instant (or within a short time window), it gives the tracker evidence that the page visits were from the same device. This is true even if the user cleared cookies between the two visits or used different browser contexts for the two visits, such as regular versus private browsing.

The researchers further pointed out that short-term tracking was possible even on platforms which didn't expose the high-precision charge level, although it was less effective. On platforms other than Firefox on Linux, the battery charge level was just two significant digits. However, by combining the charge level with `dischargeTime` and `chargeTime` the researchers estimated the possible number of states to be on the order of millions under normal operating conditions. Thus, a tracker could still conclude that two page visits with the same status readout is likely the same device, particularly if it coupled that measurement with the device's IP address.

Finally, the researchers showed that the double-precision readouts for Firefox on Linux enabled a more sophisticated attack in which a site could recover the battery capacity. The attack works by reading the device's current charge level and calculating which possible battery capacities could result in that charge level based on how the underlying operating system battery library calculates charge level. As the tracker makes more readings, it decreases the number of possible battery capacity values that could result in the observed sequence of charge levels. In the end, a tracker could recover the actual battery capacity, and use that as a static identifier for the device. This capacity could also be included alongside other device properties in a broader device fingerprint.

D. Initial privacy improvements by browser vendors

In response to the 2015 report [43], Firefox fixed the precision of the battery level readout to two significant digits for all platforms [44]. Chrome did not return high precision readouts on Linux because of an implementation difference. However, had Chrome used a high precision source it would have exhibited the same behavior. To prevent this, Chrome preemptively capped the precision to no more than 2 digits on all platforms [45]. The W3C specification was amended [46] with non-normative privacy recommendations. Notable additions are:

1) *data minimization* — avoiding the exposition of high precision readings of battery status information
2) *user control* — optionally making the API subject to browser permissions that may require prompting the user prior to the use of the API
3) *incognito support* — disabling the API in private browsing modes
4) *transparency* — informing the user when the API is and has been in use

This response prevents the battery capacity from being recoverable and lessens the usefulness of status readouts as a short-term identifiers. However, even with reduced precision the battery status output will still provide additional identifying information that can be used to fingerprint and re-identify a device over short time intervals.

E. Discovery of misuse on the web

Englehardt and Narayanan measured the prevalence of tracking and fingerprinting techniques on the Alexa top 1 million websites in January 2016 [27]. During manual examination of automatically identified fingerprinting scripts, they found two scripts, together present on 22 sites, that used battery status as part of a device fingerprint. This finding confirmed that battery status was being used to fingerprint devices in the wild. In Section IV we present an automated analysis of the Alexa top 50,000 to audit the use of the API in late 2016.

One script, served by lynxbroker.de, retrieves only the current charge level of the device. The other script, served by adscore.com, queries all properties of the `BatteryManager` interface, including the current charging status, the charge level, and the time remaining to discharge or recharge. Both scripts combine the retrieved battery information with other identifying properties of the device, such as the canvas fingerprint[2] or system fonts, to create a device fingerprint.

F. Discovery of second-order privacy concerns

Until May 2016, the primary privacy concern of the API was its usefulness for online tracking. In May 2016, Uber disclosed its finding that users with a low battery are more willing to pay higher prices for a ride (i.e. more likely to book a ride during surge pricing) [47]. This sparked concerns that Uber or other companies could misuse battery status information to increase prices for users with a low battery level [48], [49]. Mozilla cited these second order privacy concerns on W3C mailing lists during their initial discussions of whether to remove or restrict the API [50]. Materials obtained in a Subject Access Request confirm that Uber indeed collects the device battery level for use in fraud detection [51].

G. Removal of the API from browsers

As summarized in Table I, support for the Battery Status API has tumbled in response to privacy concerns. At its peak in 2016, the API was implemented in the engines supporting Firefox, Chrome, and Safari (though it was disabled in Safari). In addition, other browsers built on the major engines, such as Opera and the Yandex Browser, also exposed the API to the web.

In October 2016 Mozilla announced it would remove access to API by web content in Firefox 52 [59]. The API is now restricted to internal browser code, and may eventually be exposed to WebExtensions-based browser extensions [52]. In March 2017 Firefox 52 was released with the API removed [53]. Following Mozilla's decision, WebKit, the underlying engine behind Safari browser, removed the Battery Status API from its source tree [54].

As of March 2017, Chrome developers have not provided an official stance on whether or how they will change the API. It is possible they are considering placing it behind a user permission, as evidenced by the choice of bug component[3] for the relevant bug [55]. Microsoft Edge continues to have the API on its feature wishlist [56]; as of March 2017 there is no indication of a change.

Other browsers based on these engines could also restrict access to the API if desired. One such example is the Yandex Browser (built on the Blink engine), which now spoofs a fully charged status until the user explicitly enables the API using the appropriate browser setting [58]. Yandex has limited market share, but we include it in the table to show the versatility of responses by browser vendors.

Browser vendors regularly make privacy-related changes and continually deprecate unused and insecure features. However, the removal of an entire API in response to privacy concerns is unprecedented. We verified that this type of feature removal has not happened before by checking a website which tracks browser changes for compatibility purposes[4].

H. The future of the API

As of March 2017, it is unclear how the specification and remaining implementations will progress. Since the API was implemented in both Chrome and Firefox in 2016, it fulfilled the W3C requirement of two interoperable implementations [18]. Thus, despite only having one current implementation in 2017 (i.e. Chrome), the specification could progress to a W3C Recommendation. If there isn't sufficient interest by the authors to continue the specification, it can be published as a W3C Note, which would signify the end of active

[2]For a description of canvas fingerprinting see [26]

[3]`Blink > Permissions API`

[4]www.fxsitecompat.com

Browser	Engine	API Support (2016)	Current API Support (2017)
Firefox	Gecko	Initial support since version 10 [39]	Inaccessible to web content [52] [53]
Safari	WebKit	Not enabled. WebKit support since 2012 [40]	WebKit removed [54]
Chrome	Blink	Supported since version 38 [41]	Permissions under consideration [55]
Edge	EdgeHTML	Not supported. Considered [56]	Not supported. Considered [56]
Yandex	Blink	Supported	Spoofing by default, opt-in [57], [58]

TABLE I: API support in various browsers and privacy strategies employed.

development by the W3C. The specification authors have suggested restricting access to the API to secure, top-level browsing contexts as an additional step to the privacy risks associated with the API [60].

IV. Use and misuse of the API in the wild

A. Statistics on usage in the wild

We measured the use of the Battery Status API on the homepages of the Alexa top 50,000 sites in November 2016[5] using OpenWPM [27]. We found that in total, the API was used by 56 distinct parties on 841 sites. The majority of this usage was by third parties — 33 third parties on a total of 815 sites.

We manually classified these 33 scripts, all distinct, to determine how the feature was being used around the time of its removal. We build on the fingerprinting classification methodology outlined in [27]. We classify a script as *benign* if it uses the battery status to do things we feel the API designers intended to support, such as performance and diagnostic measurements. We classify a script as *tracking* if it uses the API for device identification, whether for fingerprinting, analytics, or fraud detection.

We found 16 third parties using the API for tracking, 11 of which use it as part of a device fingerprint. An additional 8 third parties use the API for benign purposes. For the remaining 9 third parties, we were unable to classify the usage, either due to script obfuscation or vagueness. Scripts from the 16 trackers are present on 347 sites (or around 48% of sites on which we classified API use). The benign uses of the API were primarily from two third parties: YouTube, where the API was used in performance metrics for embedded videos, and Boomerang[6], a performance measurement library.

B. A representative example of misuse

A representative third-party script using the Battery Status API for tracking is a *beacon* script[7] served by riskified.com, a supplier of fraud prevention solutions for e-commerce systems. Riskified's marketing material maintains that the company takes advantage of user behavioral monitoring models.

Riskified scripts collect a number of behavioral properties of a user's device. From the Battery Status API, they collect the charging state, the charge level, and time to discharge (or charge). In addition to battery information the scripts collect the number of CPU cores, the JS console heap size, the in-use console heap size, and the total console heap size.

We found Riskified scripts utilitizing the Battery Status API on 27 of the top 50,000 sites measured. Among the notable sites embedding Riskified were burberry.com (a popular clothing fashion brand). Additionally, we queried the Princeton Web Census data for November 2016 [27] and found that 205 of the top 1 million sites request the same Riskified script. Riskified receives the data with requests to *https://c.riskified.com/client_infos.json*.

C. A retrospective look at vendor response

Nearly half of the Battery Status API use we classified was for user profiling or identification — a use case the API designers did not intend to support. When measured in terms of distinct scripts rather than distinct sites, that fraction rises to two-thirds. Note that our measurement is conservative; parties which collect battery status information through performance or feature detection libraries can also use that information to track users, but we do not make this assumption.

At the time of Firefox's and WebKit's decisions to remove the API, the preliminary evidence suggested that it was being misused in the majority of cases [61]. Mozilla's discussion thread cites two specific uses: the research described in Section III and the Boomerang performance library [61]. The empirical data presented here suggest that use is indeed split between gathering performance metrics and tracking users. We found no additional categories of use, and confirm that the examples presented in the discussion thread reflect the broader usage on the web.

V. Lessons Learned & Recommendations

Based on insights from our case study, 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 in the draft phase, as well as a review and tests of experimental web browser features.

A. Information exchange between vendors and researchers is essential

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. Standards-based platforms such as

[5]Since Mozilla and WebKit announced their intent to remove the API in October 2016, it is possible some sites or scripts could have changed their use of it in response to that news. We believe a 1 month time window is small enough that this is unlikely to have a significant effect on our results.

[6]https://soasta.github.io/boomerang/doc/

[7]Supplied from the URL of the form http[s]://beacon.riskified.com/?shop=...

the web are more conducive to research, and therefore attract more of it, compared to proprietary platforms. Further, when implementations are open-source, knowledge propagates not just from researchers to vendors but also among vendors.

Our case study illustrates these themes and their beneficial effects on privacy. After Firefox was reported to return high-precision values on Linux (Section III-C), both Firefox and Chrome fixed the bug. Note that although Chrome did not exhibit the vulnerability, it chose to preemptively restrict the precision of the readout to prevent the possibility. This illustrates knowledge propagation between vendors, sparked by research results. Similarly, the specification was updated to include a recommendation to avoid high-precision readouts (Section III-D).

Still, 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.

B. The specification process should include a privacy review of implementations

On the modern web, proposed features often get deployed rapidly. At the time a specification is drafted, initial implementations are typically available in the development versions of browsers. By the time the spec is finalized, several vendors may 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.

In contrast to the Battery Status API, consider the Ambient Light Events API, which provides access to the light level of the device's surroundings. The specification was examined at the API design level, the implementation was tested, and the source code was reviewed[8] — all as part of the review process. Issues identified at the implementation level led both Chrome and Firefox to address a rounding issue related to the light level data [62], [63].

C. 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 [61]. 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 [27], [26]. 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 [27], [26], [24], [25]. 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.

D. 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 [38]. 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. Privacy assessment methodologies must evolve to keep abreast of the growing technical complexity of web APIs.

The case study shows the importance of assessing the data quality requirements of APIs, as unnecessarily precise data may expose users to unforeseen privacy risks. It also shows the importance of data minimization as a precaution against unexpected future misuses. Indeed, the Battery Status vulnerability served as a motivating example of these strategies in a W3C draft recommendation on browser fingerprint minimization [30].

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 [64].

E. Avoiding over-specification supports innovative privacy solutions

W3C specifications are expected to be well defined, but do allow implementers significant leeway. We recommend that standards exploit this flexibility to set a privacy floor yet leave room for innovative privacy solutions by implementers. Over-specification may have the unintended effect of rendering some privacy solutions uninteroperable with the standard or with other web features.

Indeed, implementers have employed novel strategies to mitigate the privacy risks of the Battery Status API (Section III-G). Yandex, which reports that the battery is fully

[8]Lukasz Olejnik participated in this effort as a W3C Invited Expert.

charged, effectively disables the API by default. A user may choose to offer websites battery information in an explicit opt-in manner [57]. This opt-in mechanism is also used for the Vibration API [12], possibly addressing potential misuses [65], and giving users a consistent privacy control experience across APIs. In contrast, Firefox removed access to the entire API from web content, but allowed the API to be accessible internally and to addon-sdk extensions. This allows the API to still be used by privileged code while preventing abuse from untrusted code.

F. Specifications should provide guidance for web developers, not just browser vendors

Specifications should not only identify points of privacy concern for browser vendors and other implementers, but should also provide useful guidance for web application developers when possible. 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.

The Battery Status API specification currently contains guidance describing the possible risks and suggesting mitigation strategies (Section III-D). It is one of the sensor APIs maintained by W3C's Device and Sensors Working Group [66]. Other sensor APIs have richer data sources and may pose more complex privacy threats, making it crucial to provide guidance for developers. For example, draft specifications of the Generic Sensors API [11] and the Web Bluetooth API [10] recommend that web developers perform a privacy impact assessment prior to deploying applications which make use of these APIs. We commend these authors for attending to privacy, but call for such specifications to include more detailed recommendations.

VI. Conclusion

The removal of an entire browser feature by multiple vendors in response to privacy concerns is an unprecedented decision. It underscores the importance of engineering for privacy throughout the specification, implementation, and web development stages.

Our work highlights how privacy research can influence standards and implementations. We hope that our case study and recommendations will prove useful to standards bodies, browser vendors, and web developers. We also hope that privacy impact assessment and other sound privacy engineering practices will make inroads into nascent domains such as the Internet of Things.

Acknowledgments

We would like to thank Hadley Beeman (W3C TAG), Marcos Caceres (Mozilla) and Anssi Konstiainen (Intel) for help and useful feedback. Englehardt and Narayanan are supported by NSF award CNS 1526353. Measurements were funded with an AWS Cloud Credits for Research grant.

References

[1] 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.

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

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

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

[6] "Tracking Protection Working Group Charter," https://www.w3.org/2011/tracking-protection/charter.html, 2011.

[7] M. West, "Self-Review Questionnaire: Security and Privacy," https://w3ctag.github.io/security-questionnaire/, 2015, accessed: 25.10.15.

[8] ——, "Bikeshed should require security and privacy considerations sections," https://github.com/tabatkins/bikeshed/issues/513, 2015.

[9] N. Doty, "Nudging/warnings/prompts for Privacy and Security Considerations sections," https://github.com/w3c/respec/issues/539, 2015.

[10] Web Bluetooth Community Group, "Web Bluetooth API," https://webbluetoothcg.github.io/web-bluetooth/, 2017.

[11] T. Langel and R. Waldron, "Generic Sensors API," https://www.w3.org/TR/generic-sensor/, 2017.

[12] A. Kostiainen, "Vibration API," https://www.w3.org/TR/vibration/, 2016.

[13] W3C, "WebVR," https://w3c.github.io/webvr/spec/1.1/, 2017.

[14] Y. Zhu, "Clarify section 3.14 and add Third Party Tracking as an opt-in threat model," https://github.com/w3ctag/security-questionnaire/issues/7, 2015.

[15] T. Vila, R. Greenstadt, and D. Molnar, "Why we can't be bothered to read privacy policies models of privacy economics as a lemons market," in *Proceedings of the 5th international conference on Electronic commerce*. ACM, 2003, pp. 403–407.

[16] A. Acquisti and J. Grossklags, "What can behavioral economics teach us about privacy," *Digital Privacy: Theory, Technologies and Practices*, vol. 18, pp. 363–377, 2007.

[17] S. Gürses and J. M. del Alamo, "Privacy engineering: Shaping an emerging field of research and practice," *IEEE Security & Privacy*, vol. 14, no. 2, pp. 40–46, 2016.

[18] "W3C Technical Report Development Process ," https://www.w3.org/2017/Process-20170301/#Reports, accessed 10.02.16.

[19] H. C. Altaweel I, Good N, "Web privacy census," *Technology Science*, 2015. [Online]. Available: http://techscience.org/a/2015121502

[20] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash cookies and privacy." in *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

[21] M. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle, "Flash cookies and privacy II: Now with HTML5 and ETag respawning," *World Wide Web Internet And Web Information Systems*, 2011.

[22] A. M. McDonald and L. F. Cranor, "Survey of the use of Adobe Flash Local Shared Objects to respawn HTTP cookies, a," *ISJLP*, vol. 7, 2011.

[23] J. Mayer, "The Turn-Verizon Zombie Cookie," http://webpolicy.org/2015/01/14/turn-verizon-zombie-cookie/, 2015.

[24] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Security and Privacy (S&P)*. IEEE, 2013.

[25] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "FPDetective: dusting the web for fingerprinters," in *Proceedings of CCS*. ACM, 2013.

[26] 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.

[27] 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*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1388–1401. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978313

[28] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*. Springer, 2010, pp. 1–18.

[29] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints," in *37th IEEE Symposium on Security and Privacy (S&P 2016)*, 2016.

[30] N. Doty, "Mitigating Browser Fingerprinting in Web Specifications," https://github.com/w3c/fingerprinting-guidance/issues/3, 2015.

[31] "HTML 5.1 W3C Recommendation, 1 November 2016," https://www.w3.org/TR/html51/browsers.html#top-level-browsing-context, 2016.

[32] Y. Zhu and M. West, "Secure Contexts," https://www.w3.org/TR/secure-contexts/, 2016.

[33] M. Lamouri, M. Cceres, and J. Yaskin, "Permissions API," https://w3c.github.io/permissions/, 2016, accessed: 15.02.17.

[34] A. Kostiainen and M. Lamouri, "Battery Status API," https://www.w3.org/TR/2016/PR-battery-status-20160329, March 2016.

[35] A. Kostiainen, "Battery Status Event Specification," https://www.w3.org/TR/2011/WD-battery-status-20110426/, April 2011.

[36] ——, "Battery Status API," https://www.w3.org/TR/2011/WD-battery-status-20111129/, November 2011.

[37] A. Kostiainen and M. Lamouri, "Battery Status API," https://www.w3.org/TR/2012/CR-battery-status-20120508/, May 2012.

[38] ——, "Battery Status API," https://www.w3.org/TR/2014/CR-battery-status-20141209, 2014.

[39] Mozilla, "Firefox 10 for developers (release notes) ," https://developer.mozilla.org/en-US/Firefox/Releases/10, 2012.

[40] WebKit, "Changeset 110991. Support for Battery Status API." https://trac.webkit.org/changeset/110991, 2012.

[41] "Battery Status API," https://www.chromestatus.com/feature/4537134732017664, 2015.

[42] Opera, "Opera 26 Release Notes," https://www.opera.com/docs/changelogs/unified/2600/, 2014.

[43] L. Olejnik, G. Acar, C. Castelluccia, and C. Diaz, *The Leaking Battery*. Data Privacy Management, 2015.

[44] L. Olejnik, "Bug 1124127 - Round Off Navigator Battery Level on Linux," https://bugzilla.mozilla.org/show_bug.cgi?id=1124127, 2015.

[45] T. Volodine, "Issue 1229143006: Enforce restricted precision of the Battery Status API level attribute (Closed) ," https://codereview.chromium.org/1229143006, 2015.

[46] A. Kostiainen and M. Lamouri, "Battery Status API," https://www.w3.org/TR/battery-status/, July 2016.

[47] K. Chen, "This Is Your Brain On Uber," http://www.npr.org/2016/05/17/478266839/this-is-your-brain-on-uber, 2016.

[48] B. Carson, "You're more likely to order a pricey Uber ride if your phone is about to die," http://uk.businessinsider.com/people-with-low-phone-batteries-more-likely-to-accept-uber-surge-pricing-2016-5, 2016.

[49] J. Golson, "Uber knows you'll probably pay surge pricing if your battery is about to die," http://www.theverge.com/2016/5/20/11721890/uber-surge-pricing-low-battery, 2016.

[50] M. Caceres, "Re: Notes of June 30 teleconference," https://lists.w3.org/Archives/Public/public-device-apis/2016Jul/0000.html, 2016.

[51] P.-O. Dehaye, "Battery Status API," https://github.com/pdehaye/BigOther/blob/master/uber/uber_first_response.xlsx, November 2016.

[52] C. Peterson, "Bug 1313580 - Remove web content access to Battery API," https://bugzilla.mozilla.org/show_bug.cgi?id=1313580, 2016.

[53] Mozilla, "Firefox 52 Release Notes," https://www.mozilla.org/en-US/firefox/52.0/releasenotes/, 2017.

[54] B. Eidson, "Bug 164213 - Remove Battery Status API from the tree ," https://bugs.webkit.org/show_bug.cgi?id=164213, 2016.

[55] L. Olejnik, "Issue 661792 - Battery API raises privacy issues," https://bugs.chromium.org/p/chromium/issues/detail?id=661792, 2016.

[56] Microsoft, "Platform Status Suggestions," https://wpdev.uservoice.com/forums/257854-microsoft-edge-developer/suggestions/6263689-battery-status-api, 2012.

[57] Yandex Browser Blog, " Beware Evil APIs ," https://browser.yandex.com/blog/beware-evil-apis, 2016.

[58] L. Olejnik, "Issue 661792 - Battery API raises privacy issues," https://bugs.chromium.org/p/chromium/issues/detail?id=661792#c4, 2016.

[59] Mozilla, "Firefox 52 for developers (release notes) ," https://developer.mozilla.org/en-US/Firefox/Releases/52#Others, 2017.

[60] A. Konstiainen, "Allow use from within secure context and top-level browsing context only," https://github.com/w3c/battery/issues/10, 2017.

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

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

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

[64] M. Perry, E. Clark, and S. Murdoch, "The Design and Implementation of the Tor Browser [DRAFT]," https://www.torproject.org/projects/torbrowser/design/, 2015.

[65] " Issue 507703. Ads using navigator.vibrate," https://bugs.chromium.org/p/chromium/issues/detail?id=507703, 2015.

[66] "Device and Sensors Working Group," http://www.w3.org/2009/dap/, 2009, accessed: 15.02.17.