Understanding the Performance Costs and Benefits of Privacy-focused Browser Extensions

Kevin Borgolte
borgolte@cs.princeton.edu
Princeton University

Nick Feamster
feamster@uchicago.edu
University of Chicago

Abstract

Advertisements and behavioral tracking have become an invasive nuisance on the Internet in recent years. Indeed, privacy advocates and expert users consider the invasion significant enough to warrant the use of ad blockers and anti-tracking browser extensions. At the same time, one of the largest advertisement companies in the world, Google, is developing the most popular browser, Google Chrome. This conflict of interest, that is developing a browser (a user agent) and being financially motivated to track users' online behavior, possibly violating their privacy expectations, while claiming to be a "user agent," did not remain unnoticed. As a matter of fact, Google recently sparked an outrage when proposing changes to Chrome how extensions can inspect and modify requests to "improve extension performance and privacy," which would render existing privacy-focused extensions inoperable.

In this paper, we analyze how eight popular privacy-focused browser extensions for Google Chrome and Mozilla Firefox, the two desktop browsers with the highest market share, affect browser performance. We measure browser performance through several metrics focused on user experience, such as page-load times, number of fetched resources, as well as response sizes. To address potential regional differences in advertisements or tracking, such as influenced by the European General Data Protection Regulation (GDPR), we perform our study from two vantage points, the United States of America and Germany. Moreover, we also analyze how these extensions affect system performance, in particular CPU time, which serves as a proxy indicator for battery runtime of mobile devices. Contrary to Google's claims that extensions which inspect and block requests negatively affect browser performance, we find that a browser with privacy-focused request-modifying extensions performs similar or better on our metrics compared to a browser without extensions. In fact, even a combination of such extensions performs no worse than a browser without any extensions. Our results highlight that privacy-focused extensions not only improve users' privacy, but can also increase users' browsing experience.

ACM Reference Format:
Kevin Borgolte and Nick Feamster. 2020. Understanding the Performance Costs and Benefits of Privacy-focused Browser Extensions. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3366423.3380292

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380292

1 Introduction

The online advertisement (ad) industry has been one of the fastest growing industries in the recent years, growing from over 108 billion US dollars in 2018 to an estimated 129 billion US dollars in 2019 in the United States [33], and to estimated 333 billion US dollars in 2019 globally [8]. This growth has sparked numerous academic studies of various angles of online ads, from understanding the underlying economics, the overall scale of it, mechanisms and techniques used, as well as how miscreants are abusing it to profit.

Academic and industry studies on the techniques that online ads are relying on have led to privacy advocates and expert users arguing that the ad companies' tracking and data collection, which provides the foundation for the most prevalent online ads today, that is, behavioral tracking and targeting, is excessive and invasive to users' privacy [42]. Indeed, in 2012 already, the retailer Target could predict customers' pregnancies' due dates with high accuracy, based on browsing and online shopping patterns, and used the information for targeted ads by physical mail (and inadvertently disclosed a daughter's pregnancy to her father) [13].

Concerns about excessive data collection as well as data breaches have since led to new regulation, such as the European General Data Protection Regulation (GDPR) [9], which went into effect in May 2018 in the European Union, and the California Consumer Privacy Act (CCPA), which will go into full effect January 2020 [40]. Such regulation is an important first step to ensuring users' privacy, and, considering the large fines and penalties that GDPR allows to impose for violations (up 20 million Euro or 4% of the worldwide annual revenue, whichever is greater), it has already forced companies to rethink their approach to data handling, even though the exact interpretation of GDPR's requirements and enforcement remain largely uncharted territory. After introduction of GDPR, the New York Times switched from ads based on behavioral tracking to context-based ads, and it did not experience a loss in ad revenue [3]. This brings into question whether behavioral tracking is even necessary from an economical point of view for content and service providers to allow free access to their content and services.

Regulation is however, by its nature, a local matter. Online tracking, on the other hand, is a global phenomena and online tracking remains pervasive on a global level. There may less or no tracking in the European Union due to GDPR, but there is still substantial behavioral tracking and targeting in the United States of America, China, India, and most other countries. Therefore, users who want to protect their privacy and not be subject to tracking need to rely on other means, with anti-tracking or ad-blocking browser extensions [4, 10, 11, 14] being the most popular solution.

Browser extensions, however, had their own fair share of security and privacy issues, because of their powerful capabilities, which

can allow them to inspect and modify all of a user's browsing activity. For example, extensions designed with malicious intent have been observed stealing user credentials, cookie stuffing to generate affiliate revenue, and ad injection. Using a browser extension has proven to be so profitable for miscreants that they even approached extension developers to buy extensions from them, with the goal of then pushing a malicious update to the extension's users through the browser's update mechanism [1, 31].

The discovery of these security problems led to a debate among browser vendors to remove some capabilities for extensions, with the most dangerous capability being the ability to inspect and modify or prevent any request that the browser makes, to any website. Removing the functionality would clearly prevent malicious extensions from misusing it, and, hence, protect users' privacy. However, removing this capability also affects benign extensions. In particular, it renders existing privacy-focused extensions that rely on this functionality to protect users' privacy inoperable, like anti-tracking extensions and ad-blockers. A particularly interesting angle on this argument is that extensions requiring these capabilities have been preventing security and privacy issues that the browser vendors have not tackled (in some cases because they have no financial incentive to do so). For example, by blocking all ads, malicious ads cannot become a security issue. Similarly, a user may be less willing to accept that an ad company is tracking her than her extension being compromised, as she trusts the extension developer more than the ad company. As such, preventing extensions from privacy-sensitive capabilities or allowing such extensions represents a security and privacy trade-off, and removing the functionality would patronize users by restricting their choice.

Independent of the security argument, browser vendors have argued that the way this capability is used also reduces performance and, in turn, negatively affects user experience, which affects user engagement and profit [28, 48]. However, intuitively, privacy-focused extensions could provide a performance benefit to the user: By preventing unwanted content from being loaded, less content needs to be loaded, which should be faster. But, to do so, these extensions need to inspect and understand the browser's request, and then make a decision to allow or prevent a request, which comes at its own additional performance cost. In this paper, we investigate how privacy-focused extensions relying on request and response modification, the powerful capability in question, actually affect browser performance, system performance, and user experience for the two most popular desktop browser, Google Chrome and Mozilla Firefox. We find no evidence that privacy-focused extensions fundamentally degrade performance in any way, but our results show that they improve performance across various metrics. Therefore, we believe it is ill-advised to deprecate the powerful capabilities that privacy-focused extensions rely on. Instead, considering that the functionality can be misused, it appears more wise to leverage the already-existing walled garden ecosystem surrounding extensions (i.e., extensions can only be installed if they are signed and published on the extension store) to limit access to privacy-sensitive functionality to carefully vetted extensions only.

We make the following contributions:

- *We provide the first extensive study investigating how privacy-focused browser extensions affect performance.* We performed our measurements for different privacy-focused extensions, across browsers, and from multiple vantage points, providing additional insight on how performance of the same extensions differs across these variables.
- *We show that privacy-focused extensions can improve browser performance, system performance, and user experience compared to a browser without extensions.* Correspondingly, we urge browser vendors to retain functionality necessary for privacy-focused extensions, but which is slated for deprecation, as the presented performance argument does not appear to be valid.
- *We open source our measurement tools and techniques, so that other researchers can build upon it and reproduce it.*

Following, we first provide the background of how privacy-focused browser extensions work (Section 2). We then detail methodology of our study, like the selection of the extensions, browsers, and metrics, as well as the experimental setup (Section 3). Subsequently, we evaluate how the extensions affect browser and system performance (Section 4) and discuss our results (Section 4.3). Finally, we compare to related work (Section 5) and conclude (Section 6).

2 Background

Following, we briefly describe current techniques that users can employ to protect their privacy against online tracking, how privacy-focused extensions work internally, that is, how they inspect and modify or prevent requests that the browser would otherwise make, and how they decide which requests to modify and prevent.

2.1 Techniques Against Tracking

Users can utilize various techniques to protect themselves from tracking and ads when browsing the web. Some techniques work occur at the network level, such as blocking domain names from being resolved via PiHole [15] or using an interception proxy like Privoxy [29]. And, while these network-level blocking techniques have the benefit that they are independent of the application, they are not sufficiently fine-grained because they lack visibility into the browser (or other application). For example, they cannot determine if traffic to a third-party is intentional (e.g., the user clicked on a link), or if it is unwanted traffic that should be blocked. Privacy-focused browser extensions, albeit application-specific, do not suffer these short-comings and can reliably block web-based tracking, as they can have full visibility into all requests, and can intercept and modify or prevent them.

2.2 Request Analysis

The functionality that current privacy-focused extensions build upon to modify or block in-flight requests is the `webRequest` API, which is part of the WebExtensions API set [44] and works by declaring event listeners [12, 23]. To use it, an extension must be granted the `webRequest` permission. The API allows an extension to intercept, analyze, and modify or block requests at various stages of a request's flow, at which the browser triggers an event and notifies the extension. In particular, it allows to: *(i)* cancel requests in the stages `onBeforeRequest`, `onBeforeSendHeaders`, or `onAuthRequired`; *(ii)* redirect requests in the event handlers for `onBeforeRequest` and `onHeadersReceived`; *(iii)* modify request headers when processing

the `onBeforeSendHeaders` callback; and, *(iv)* modify response headers in the `onHeadersReceived` event handler.

An extension can also subscribe to various other events, which we do not discuss here for space reasons as they are only informational and not critical to the inner workings of privacy-focused extensions. If the API is used to modify and block requests from occurring, then the events must be declared to be synchronous (blocking), and an extension wanting to do so needs the additional `webRequestBlocking` permission. Figure 1 shows the order in which these events occur, with events shaded gray that extensions rely on to intercept and analyze requests, green depicting the successful completion of a request, and red highlighting the event raised if an error occurred.

Beyond inspecting and altering requests, Mozilla Firefox also supports the modification of response content by monitoring and filtering the response via `filterResponseData` on a per request basis. Google Chrome does not support this functionality.

Quite clearly, if an extension is using the synchronous `webRequest` API and its process to decide if a request should be blocked or allowed is slow, then this will affect negatively performance. On the other hand, if the decision-making process is sufficiently fast compared to the cost of retrieving and parsing or rendering the resources, then the blocking API can also improve performance.

2.3 Decision Making

The process that privacy-focused extensions use to decide which requests should be modified or blocked can be broadly categorized in two classes, blacklists and heuristic algorithms.

Most privacy-focused extensions, for example Adblock Plus [10], Ghbostery [11], Disconnect [4], and uBlock Origin [14] rely on traditional blacklisting because it has been shown to be effective. Indeed, some extensions actually share lists, for example EasyList and EasyPrivacy [39], or Peter Lowe's ad and tracking server list" [20], which are often curated by their user community. Beyond community lists, other extensions differentiate themselves by providing lists that they curate. For example, Disconnect and Ghostery use their own private lists. Adblock Plus distinguishes itself from other blockers by also utilizing a whitelist of "acceptable ads" that it does not block. Unfortunately, blacklists are reactive, that is, ads and trackers must be known before they can be added to the blacklist, and they usually need to be verified to prevent collateral damage.

This delay has created the desire for alternatives to traditional blacklisting. One solution are heuristic algorithms that try to detect ads and tracking instead of matching a domain name or embedded script. The "Privacy Badger" extension by the Electronic Frontier Foundation (EFF) [7] is likely the most prominent example. It uses heuristics like repeatedly observing third-party content that matches specific properties (e.g., supercookies) and then modifies or blocks the corresponding requests [5]. To prevent false positives, which occur if a third-party provides legitimate content, Privacy Badger does not always block third-party requests, but may modify the request to protect the user's privacy (e.g., by removing cookies from the request). Naturally, extensions that leverage heuristics or learn about tracking are thought to be computationally more expensive than blacklists and this may affect performance.

3 Methodology

In this section, we detail our methodology, that is, which browsers we investigate, which extensions we analyze, what metrics we will evaluate, how we set up our experiment, and its limitations.

3.1 Browsers

For our analysis, we investigate the performance of privacy-focused extensions on the two most popular desktop browsers, Google Chrome and Mozilla Firefox, which have a combined market share of almost 80% [37]. It is important to note that while Google is planning to remove `webRequest` from Chrome for performance and privacy reasons [43] and replace it with `declarativeNetRequest`, Mozilla has currently no plans to remove the functionality from Firefox [24]. We investigate both browsers to better understand if any performance cost may be related to a browser's implementation.

We perform our analysis on Linux, specifically Debian Linux 10 (buster), with Google Chrome 77 and Mozilla Firefox 68. We drive the two browsers with Selenium, which is a browser automation framework [32]. Unfortunately, while we can run Firefox in headless mode, Chrome does not support extensions in headless mode [17]. Therefore, we run Chrome in headful mode using a virtual display through the X virtual framebuffer (Xvfb) [47]. To ease deployment and reproducibility, we created Docker containers. Finally, we made our code publicly available: https://github.com/noise-lab/privacy-extensions.

3.2 Extensions

We evaluate eight privacy-focused extensions for our analysis: AdBlock Plus [10], Decentraleyes [30], Disconnect [4], Ghostery [11], HTTPS Everywhere [6], NoScript [21], Privacy Badger [7], and uBlock Origin [14]. We selected these extensions because they are among the most popular privacy-focused extensions and they cover different aspects of advertisement blocking and anti-tracking, as shown in Table 1. For example, AdBlock Plus is used by over 20 million users and does not block all ads by default, but allows some ads through the acceptable ads program. On the other hand, uBlock Origin blocks all ads and is used by over 15 million users. Other extensions we include because they address other privacy problems or block ads and tracking in a non-traditional sense. For example, Decentraleyes blocks a non-traditional kind of tracking: It keeps local copies of popular JavaScript libraries and loads the local version instead of the remote version, for which access could be tracked (possibly with cookies and referrer information). HTTPS Everywhere is unique among the selected extensions as it does not block ads or prevent tracking at all, but it is focused on improving privacy in a different way: It prevents eavesdroppers on a network (e.g., in a coffee shop) to observe traffic to websites that still support plain HTTP and do not upgrade the connection to HTTPS. The NoScript Security Suite distinguishes itself from the other extensions by entirely blocking all execution of included content, for example, JavaScript, web fonts, Java, or Flash, which can prevents ads from being shown and online tracking because they rely heavily on script execution.

Finally, we also investigate if an extension configuration that combines multiple extensions (Decentraleyes, Privacy Badger, and uBlock Origin) may lead to a compound performance effect.

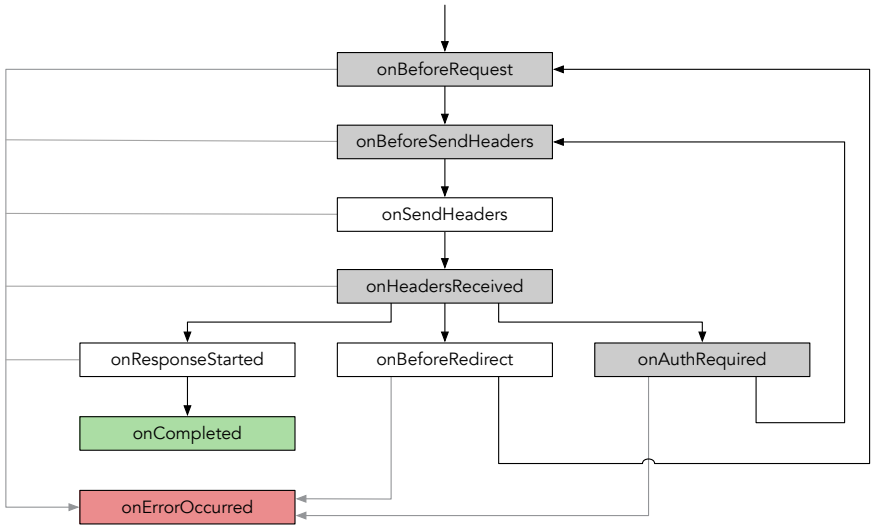

Figure 1: *Common flow of* `webRequest` *events.* Shaded gray are events when an extension can cancel or modify a request. Events shaded in white show informational events. The event in green (onCompleted) indicates that the request was completed. The event in red (onErrorOccurred) means that that an error occurred.

Table 1: Analyzed Privacy-focused Extensions. The indicators have the following meaning: ● indicates that the extension supports blocking this category by default, ◐ indicates that it blocks it partially or a non-traditional set, ◑ indicates that it supports it, but that it is disabled by default, and ○ indicates that it does not support blocking the category.

Extension	Version	Official Description	Ads	Tracking	Users
AdBlock Plus	3.6.3	Blocks annoying video ads on YouTube, Facebook ads, banners and much more. Adblock Plus blocks all annoying ads, and supports websites by not blocking unobtrusive ads by default (configurable).	◐	◑	Over 20 Million
Decentraleyes	2.0.12	Protects you against tracking through "free", centralized, content delivery. It prevents a lot of requests from reaching networks like Google Hosted Libraries, and serves local files to keep sites from breaking. Complements regular content blockers.	○	◐	≈ 280,000
Disconnect	5.19.3	Make the web faster, more private, and more secure.	○	●	≈ 1 Million
Ghostery – Privacy Ad Blocker	8.4.2	Ghostery is a powerful privacy extension. Block ads, stop trackers and speed up websites.	●	●	≈ 4 Million
HTTPS Everywhere	2019.6.27	Encrypt the web! HTTPS Everywhere is a Firefox extension to protect your communications by enabling HTTPS encryption automatically on sites that are known to support it, even when you type URLs or follow links that omit the https: prefix.	○	○	≈ 2.8 Million
NoScript Security Suite	11.0.2	The best security you can get in a web browser! Allow active content to run only from sites you trust, and protect yourself against XSS other web security exploits.	○	◐	≈ 1.6 Million
Privacy Badger	2019.7.1	Automatically learns to block invisible trackers.	○	●	≈ 1.8 Million
uBlock Origin	1.22.2	Finally, an efficient blocker. Easy on CPU and memory.	●	●	Over 15 Million

3.3 Metrics

Intuitively, using a privacy-focused extension will affect performance in some way: The extension inspects and modifies requests, or prevents them from being issued entirely. In turn, it incurs a performance cost for inspection, and modification or prevention. However, in doing so, it may actually lead to a net benefit in performance because resources that have not been requested cannot incur any cost by the browser itself, for example, for parsing an HTML document or parsing and executing JavaScript. We quantify this difference for different metrics that are focused on user experience and reflect a variable that is important to users. These metrics fall into two categories: browser metrics and system metrics.

3.3.1 *Browser Metrics*

Concerning browser metrics, we investigate *(i)* overall page-load time, *(ii)* number of requested resources, *(iii)* overall page-load size, and *(iv)* number of cookies. We measure these metrics by exporting a trace of how the browser interacted with the website and

rendering it, more commonly known as the HTTP Archive format (HAR) [45], which Chrome and Firefox support for performance measurements. It contains all requests that have been made, content responses, HTTP headers, as well as timing information when specific event were triggered by the browser. In particular, it contains timing information for the `onLoad` event, which the browser triggers when it has finished loading and rendering the page, representing page-load time. For each page load, we automatically extract the HAR through a modified version of the open source `HARExportTrigger` extension [26], which we extended to work reliably with Google Chrome in addition to Mozilla Firefox, and which we will open source at the time of publication.

3.3.2 System Metrics

In addition to browser metrics, we also collect system metrics, with the idea behind them being to evaluate the overall impact on system performance. For example, an extension could reduce the overall page-load time, number of requested resources, overall page-load size, as well the number of cookies, that is, all browser metrics. However, the extension may require substantial amount of computation to do so, incurring additional high cost in processor time. Particularly, for each page load, we collect *(i)* processor time, *(ii)* process context switches, and *(iii)* memory page faults.

Processor time measures how much time was actually spent computing across all processor cores, and it does not include time that the processor went to sleep or deep sleep, for example, while waiting on data from the network. Process context switches are the number of times that the operating system switched between processes and when it needed to store a process A's state to pause it and restore a process B's state to resume execution of process B, for example, process context switches can happen when switching between a browser's rendering engine and browser plug-ins, etc. And, finally, memory page faults occur when a process attempts to access memory that is currently not mapped or not loaded in its virtual address space.

We focus on these three system metrics because they are generally associated with high power consumption and, in turn, they can help as a proxy indicator of battery runtime. Considering that mobile devices relying on batteries are ubiquitous, investigating how privacy-focused extensions may affect system-wide user experience through reduced or increased battery runtime, and not just evaluating how they may affect browsing performance, is crucial.

We measure the system metrics through profiling with Linux's `perf_events` [27], which is a performance analysis tool of the Linux kernel that allows instrumentation, tracing, and profiling of the kernel and user space. Albeit `perf_events` is light-weight and supports hardware performance counters, it does incur cost for profiling interesting events. To minimize any potential impact that profiling these metrics may have, we aggregate them at the kernel level during the page load and retrieve results from the kernel only after retrieving all browser metrics.

3.4 Experimental Setup

Following, we detail the experimental setup that we use to measure browser and system performance (Section 4).

3.4.1 Domains

It is crucial to evaluate performance on websites that users actually visit with some regularity. Therefore, we use two distinct parts of the Tranco top 1 million domain list for our measurements [19], which is a ranking of popular websites similar to the Alexa top 1 million, but more stable and less susceptible to manipulation. Specifically, we use *(i)* the top 1,000 domains, which are often visited websites that are highly optimized and often hosted on content-delivery networks, thus, potentially having better performance; and, *(ii)* the domains 99,001 to including 100,000, which are websites that are still relevant, but which are generally less optimized and commonly not hosted on content-delivery networks. Correspondingly, our list of domains contains 2,000 domains.

3.4.2 Hardware

We perform our measurements on the Amazon Web Services (AWS) Elastic Computing (EC2) platform. We use `m5.2xlarge` instances with eight logical cores (four physical cores) of an Intel® Xeon® Platinum 8175M CPU processors with a base speed of 2.5GHz and a turbo boost of 3.5GHz, 32GiB of memory, and up to 10Gbps of network connectivity. Although a server-class Xeon processor could hide some computing inefficiencies, we selected a specific Xeon processor that is comparable in feature set and base clock rate (2.5GHz) to desktop-class processors. We also limited the number of computing cores to a number comparable to desktop-class processors, which is where Xeon processors typically differ substantially from desktop-class processors. Moreover, albeit anecdotal evidence, on a smaller testbed comprised of a desktop-class Intel Core i7 NUC computer and a Comcast 75Mbps Internet connection, we made the same observations as on our large-scale measurements (Section 4).

On each instance, we run only a single browser, so that multiple browser instances cannot interfere with each other and the browser can utilize all eight logical cores on its own. The specification of this instance reflects current medium to high-end desktop class computers in terms of processing power and memory. However, the instances' network connectivity (10Gbps) substantially exceeds what is available to end users, which may affect our results in terms of favoring fetching more or larger resources from network over performing more local computations.

3.4.3 Vantage Points

Naturally, performing our measurements from multiple vantage points is crucial to understanding if regional differences exist. To this end, we utilize the same configuration (including comparable Internet access) at our vantage points and vary the location only. Specifically, we run our experiments from a vantage points in Northern Virginia, United States of America (us-east-1), as well as a second vantage point in Frankfurt, Germany (eu-central-1). In our case, the selection of the vantage points is grounded in the fact that the United States have practically no legislation against the behavior privacy-focused extensions are aiming to prevent, while in Germany (and the European Union in general) the General Data Protection Regulation (GDPR) went into effect in May 2018 [9]. Correspondingly, privacy-focused extensions may perform differently between these two vantage points.

3.5 Limitations

Of course, our measurement has some limitations because of decisions in the design of our experiment. For example, simply collecting metrics incurs additional overhead, which may affect the results. However, we would be unable to make any statement grounded in data on how privacy-focused extensions affect performance or user experience without incurring this additional cost, which we believe is essential to better understand the modern web. The way that we collect these metrics by profiling browser and system is, to the best of our knowledge, the most light-weight technique, incurring the least amount of additional overhead. Therefore, we believe that our methodology provides the most accurate analysis today.

In a different vein, as we touched on before, our experimental setup is substantially better connected to the Internet than how most end users are connected to it, that is, the latency and link throughput to the web are better for our experiment (lower for latency, higher for link throughput) than it would be for most users. Therefore, our analysis effectively assumes a best case scenario in terms of Internet access, and our results thus lean toward a lower bound in page-load time (as more resources can be fetched quicker than for an end user). Correspondingly, the difference between an extension configuration that loads more resources or larger resources and one that loads less or smaller resources will likely be smaller in our experimental setup compared to an end user setup. For example, if an extension configuration with an extension would load a page faster than a configuration without an extension by 1s in our experiment, then we would expect that this gap to increase for the same measurement by an end user, like to 1.2s.

Finally, although our results may not generalize to other hardware, vantage points, operating systems, browsers, or extensions, our comparative evaluation provides new and unique insight in how the two most popular browsers' performance differs when using privacy-focused extensions in an environment that aims to mimic how end users would use it.

4 Evaluation

We performed our measurement based on the previously discussed methodology over a period of 18 days and 8 hours from September 25th, 2019 to October 13th, 2019 from the two AWS EC2 availability zones in Frankfurt, Germany and Northern Virginia, USA.

For our measurement study, we successfully visited each domain in our dataset approximately seven times with Mozilla Firefox, leading to between 13,378 and 13,982 data points per extension configuration for Firefox in the USA and between 13,734 and 13,982 data points per extension configuration for Firefox in Germany. For Google Chrome, we were able to visit each domain only between one and three times successfully per extensions, because we need to allow for a substantially longer extension warmup time, up to an additional 15 seconds per domain and per extension configuration. In total, for each extension configuration for Google Chrome, we successfully retrieved between 2,149 and 5,636 data points for the USA, and between 2,191 and 5,774 data points for our measurements from Germany. Table 2 shows the distribution of data points.

Following, we focus our analysis on the differences between a privacy-focused extension configuration and the browser without extensions. For example, for the metrics page-load time (onLoad) or processor time (CPU clock), our results indicate if an extension is taking more time than the browser without extensions. That is, positive values mean the extension configuration is slower, while negative values mean it is faster. Similarly, for metrics like resources and cookies, a negative value indicates that the extension configuration leads to less resources being requested or cookies being set (e.g., because less tracking resources were requested that want to set a cookie). Figure 2 and Figure 3 show the differences between each extension configuration and the extensionless browser for each metric, with Figure 2 showing Mozilla Firefox in Germany and Figure 3 showing Google Chrome in the USA. Plots are shaded based on the difference of the median in terms of the median absolute deviation (M) from the baseline (the extensionless browser), with blue shades indicating that the extension configuration is faster than the baseline, and red shades indicate that it is slower. For brevity reasons, we do not show figures for Mozilla Firefox in the USA and Google Chrome in Europe, as they perform largely similar to the same browser at the other vantage point, and we discuss the respective differences in Section 4.2.

4.1 Extension Differences

Next, we discuss the differences in how the analyzed privacy-focused extensions affect performance.

4.1.1 Mozilla Firefox

Comparing the performance of a privacy-focused extensions for Mozilla Firefox in Germany (Figure 2), we see that all but AdBlock Plus, HTTPS Everywhere, and Disconnect improve performance across all metrics besides onContentLoad, that is, the time when the initial HTML has loaded, but not the included resources. In fact, even the combination of Decentraleyes, Privacy Badger, and uBlock Origin is faster than the baseline across all metrics besides onContentLoad. For Adblock Plus, the median onContentLoad delay is approximately 1.1s, while it is only 3ms for Decentraleyes, 34ms for NoScript, and 77ms for uBlock Origin. In the 95% percentile, AdBlock Plus introduces a delay of up to 2.2s over an extensionless browser, while uBlock Origin adds a delay of 706ms.

Concerning actual page-load time (onLoad), NoScript is improving substantially over the extensionless browser, by loading websites 491ms faster (median), followed by Disconnect (-244ms faster, median), followed by uBlock Origin (-193ms faster, median). Unfortunately, NoScript bears have a potential usability problem for the normal user: It blocks JavaScript, Flash, Java, etc. by default, which can severely impact a user's experience, as a significant amount of the modern web relies on client-side JavaScript to provide functionality. Interestingly, considering the 95% percentile, Disconnect and uBlock Origin switch positions, indicating that uBlock Origin's effect on websites in the tail is lower (732ms delay over baseline vs. 888ms delay over baseline for Disconnect) and providing a more even experience across websites.

In terms of the data that needs to be downloaded for a website to load entirely, all privacy-focused extensions do better or equal to the extensionless browser. By not executing any scripts, NoScript was able to save up to 50MB in one case (min), and commonly 500KB (median). Considering a more user-friendly experience, Disconnect could save up to 49MB (min), and commonly saves around 200KB.

uBlock Origin fares similarly in the median case (200KB), but worse in the best case (20MB less). The combination of Decentraleyes, Privacy Badger, and uBlock Origin outperforms each extension on their own in he median case (235KB less than baseline; Privacy Badger alone saves 92KB; Decentraleyes does not save

Table 2: Successfully Visited Domains per Extensions Configuration. For our measurement study, we only count domains for which the browser without extensions and the configuration with extensions successfully retrieved the website and the HAR file is complete.

	Mozilla Firefox		Google Chrome	
Extensions Configurations	Germany	USA	Germany	USA
AdBlock Plus	13,953	13,761	3,524	3,321
Decentraleyes	13,785	13,378	4,452	4,227
Disconnect	13,982	13,733	4,496	2,325
Ghostery – Privacy Ad Blocker	13,982	13,711	3,150	3,050
HTTPS Everywhere	13,752	13,479	4,602	4,276
NoScript Security Suite	13,899	13,659	5,563	5,393
Privacy Badger	13,943	13,710	5,774	5,636
uBlock Origin	13,734	13,575	4,998	4,761
Decentralyes + Privacy Badger + uBlock Origin	13,801	13,536	2,191	2,149

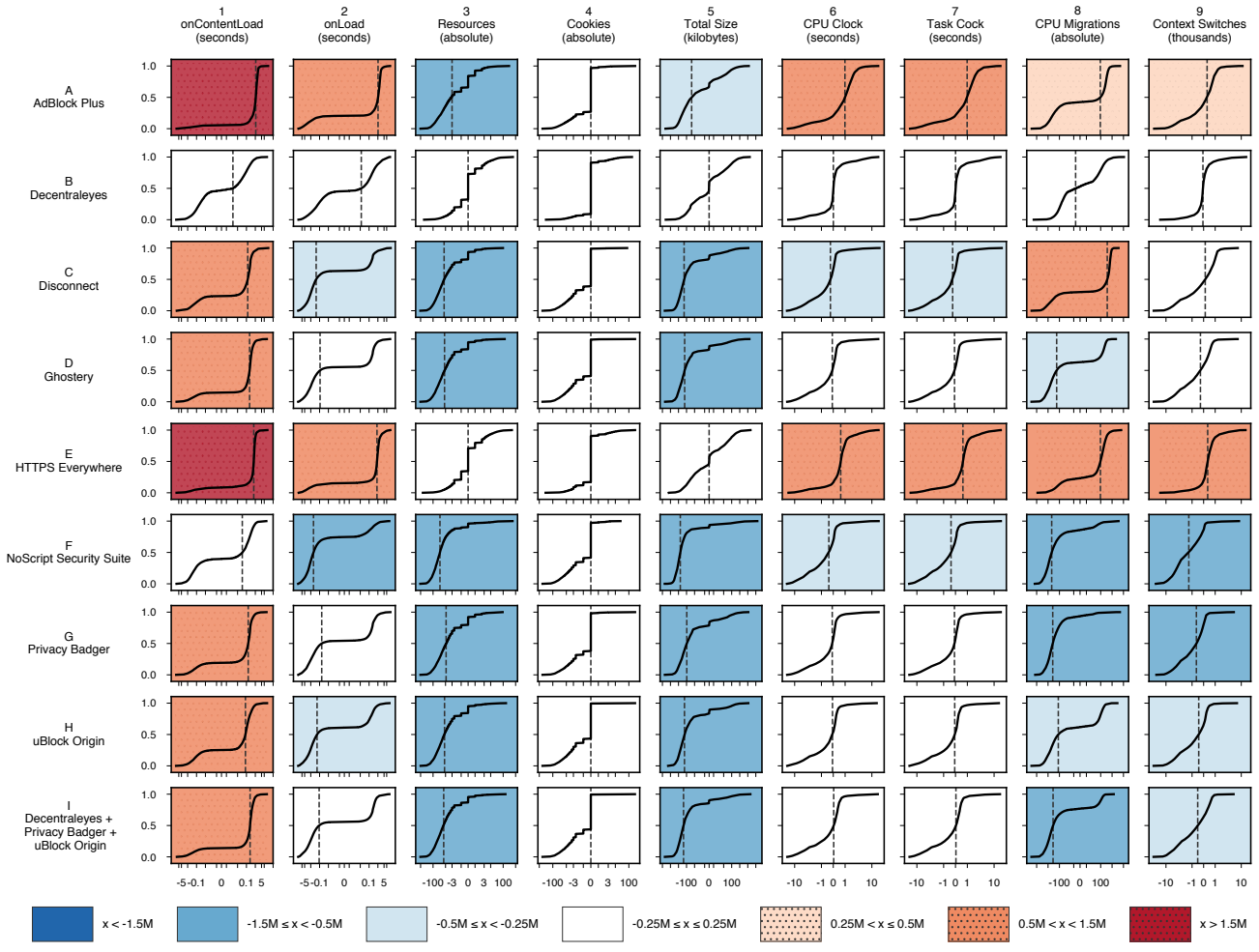

Figure 2: Difference in Performance Metrics for Various Privacy-focused Extensions for Mozilla Firefox from Germany. Negative values (shaded blue) indicate that the extension configuration (row) is performing better for the metric (column) than the browser baseline with no extensions installed. Correspondingly, positive values (shaded red with dots) indicates that the extension configuration is performing worse than the browser baseline.

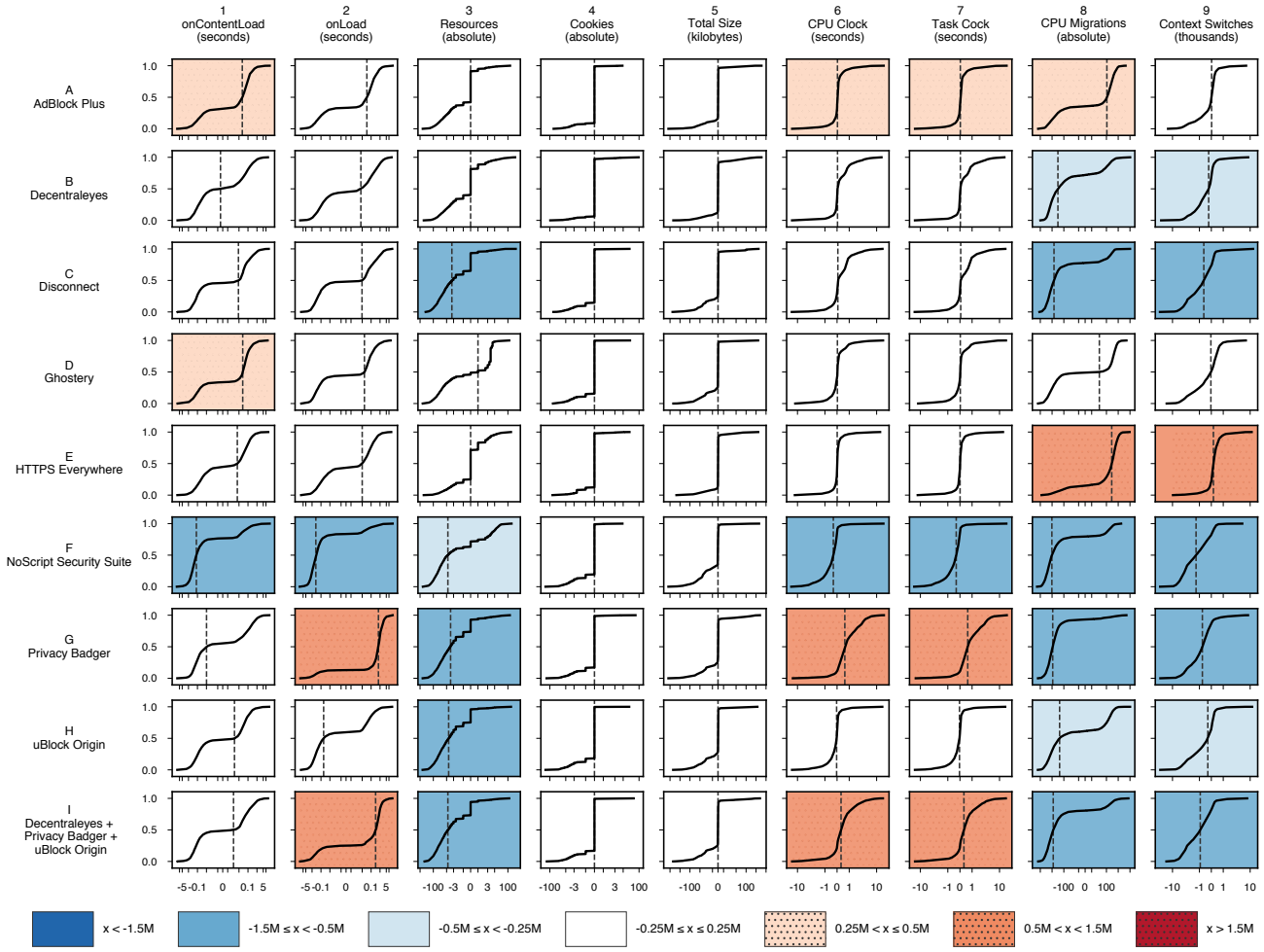

Figure 3: Difference in Performance Metrics for Various Privacy-focused Extensions for Google Chrome from the USA. Negative values (shaded blue) indicate that the extension configuration (row) is performing better for the metric (column) than the browser baseline with no extensions installed. Correspondingly, positive values (shaded red with dots) indicates that the extension configuration is performing worse than the browser baseline.

anything in the median case; and uBlock Origin saves 180KB), indicating that they are complimentary. Nevertheless, the savings that privacy-focused extensions in terms of downloaded data can provide are substantial. Interestingly, in some cases, privacy-focused extensions lead to more data being downloaded. One reason this can happen are pop-ups to request disabling the extension, and another reason is that, instead of showing advertisements, the part of the website shows now more content, which is loaded dynamically. For example, YouTube behaves in this way, loading more video thumbnails if advertisements are blocked.

The privacy-focused extensions perform largely similar in respect to cookies for Mozilla Firefox in Germany, in that all extensions that are focused on blocking tracking block approximately the same amount of cookies (577 cookies less than baseline in the best case, not blocking any cookies in the median case with a median absolute deviation of zero).

They also result in less requests in the median case: 28 requests less than the baseline for NoScript, 14 requests less for the combination of Decentraleyes, Privacy Badger, and uBlock Origin, 13 requests less for Disconnect, 12 requests less for uBlock Origin on its own, 12 requests less for Ghostery, 9 requests less for Privacy Badger, to 3 requests less to AdBlock Plus, and to the same amount of requests for Decentraleyes and HTTPS Everywhere. Notably this means that the combination of Privacy Badger and uBlock Origin block more than each extension on their own, highlighting again that they are indeed complimentary.

Finally, considering system performance, we observe that AdBlock Plus requires substantial more processor time (CPU Clock), an additional 1s of compute time in median case. Decentraleyes and HTTPS Everywhere also incur additional processor time compared to the baseline, 30ms and 630ms respectively. Contrary to these

three extensions, all other extensions on their own result in less processor time being utilized in the median case, between 374ms less CPU time for NoScript, to 240ms less for Disconnect, to 86ms less for Privacy Badger, to 81ms less for Ghostery, to 64ms for uBlock Origin. Although combination of the three extensions leads to a decrease in the other metrics and may improve privacy by blocking more unwanted behavior, it incurs additional processor time (37ms) over the extensionless browser, that is, more privacy-focused extensions do come at an additional performance cost.

Based on our experiments, the best privacy-focused extensions in terms of retaining or improving Mozilla Firefox's existing user experience appear to be Privacy Badger, Disconnect, or uBlock Origin. Very privacy-conscious users may also consider a combination of these extensions because they can be complimentary.

4.1.2 *Google Chrome*

For Google Chrome in the USA, the differences between the extensions and the baseline are generally less pronounced (see Figure 3). Some differences to Firefox are noticable though: Privacy Badger results in a page-load time that is 662ms slower (median), while AdBlock Plus now performs substantially better compared to Firefox (33ms slower than baseline). Interestingly, Disconnect is slower than the baseline for Chrome (10ms, median), but uBlock Origin remains faster (50ms faster, median). AdBlock Plus and Ghostery also block less resources compared to Firefox. An unexplained oddity that demands more attention is that privacy-focused extension do not appear to lead to a reduction in data downloaded for Chrome.

Overall, taking into account that the extensions themselves do not differ substantially between their Firefox and Chrome versions, our results highlight that implementation differences in how Firefox and Chrome implement various parts of their extension capabilities lead to substantial performance differences, identifying a potential point for improving browser performance. Nevertheless, the only metric that uBlock Origin performs worse than baseline Chrome is for onContentLoad, where it is only 3ms slower. This result directly contradicts Google's recent statement that the blocking `webRequest` API is a performance issue.

4.2 Regional Differences

In our measurements, we observe an interesting pattern in regional differences between the USA and Germany. First, all privacy-focused extensions block one to two more resources for the USA than they do for Germany, and they block up to 142 more cookies for the USA than they do for Germany, indicating that privacy-invasive behavior is more prevalent in the USA, likely because of a lack of privacy regulation. Second, extensions that needed less processor time than the baseline in Germany needed even less in the USA, and extensions that required more processor time in Germany needed even more in the USA. And, related to processor time, extensions that loaded faster than the baseline in Germany loaded even faster (comparatively) in the USA, and extensions that resulted in a slowdown over the baseline in Germany lead to an increased slowdown in the USA. That is, privacy-focused extensions appear to have more work in the USA than in Germany, and if they can do their work efficiently, such as Disconnect, Privacy Badger, or uBlock Origin, this will lead to an improved effect over the baseline.

4.3 Summary

We have measured and analyzed the performance cost and benefits of eight privacy-focused extensions. We found that although their introspection and blocking process comes at some cost, this cost is offset by performance improvements that blocking tracking and other unwanted behavior recover. Indeed, leveraging user-sensible metrics like page-load time, downloaded data, and processor time, the best performing extensions results in increased performance for Mozilla Firefox as well as Google Chrome. While we found some evidence that blocking `webRequest` can lead to poor performance, we found no evidence that it is bound to lead to poor performance. In fact, uBlock Origin is performing better or comparable to both browsers on all but one metric (onContentLoad, which is not primarily reflecting user experience and which we included for completeness only). Most notably, this contradicts Google's recent argument that blocking `webRequest` negatively affects performance.

Correspondingly, we urge browser vendors to retain blocking `webRequest` functionality, so that users have choice. To address the API's potential misuse (see Section 1), we recommend to classify it as "privileged" and requiring additional verification for extensions to use it, such as automated software analysis. An implementation of this recommendation could leverage the already extensive infrastructure surrounding the extension stores, similar to the Android's store protection mechanisms (e.g., Bouncer).

5 Related Work

Following, we relate how our research compares to prior work, specifically in the areas of measuring browser performance, and browser extension security and privacy.

5.1 Browser Performance

A multitude of studies have been conducted that analyze how to measure browser performance and how specific variables affect it. These studies have largely focused on network access, the underlying protocols like SPDY and DNS, content distribution, and server-side features.

Newman et al. [25] performed a measurement study similar to ours in 2019, investigating the impact of blocking advertisements on users' browsing quality of experience for the Alexa top 5,000 websites. They perform their study using the software-as-a-service provider WebPageTest (WPT), which uses an unknown setup of hardware, operating system, and network connectivity, raising the question if end-users would experience similar behavior. Unfortunately, they only analyze a single ad blocking extension, AdBlock Plus. Their analysis investigates page-load time and time to first paint, the latter they inaccurately describe as the "initial responsiveness" (time to first paint represents when the browser is starting to render elements, the website becomes responsive at the time of interactive; first paint is followed by first contentful paint, which is followed by first meaningful paint, which is followed by time to interactive). Their results show that time to first paint is faster without AdBlock Plus, but page-load time is lower with AdBlock Plus. To better understand if users experience the website to load faster with or without AdBlock Plus, they perform a user study using Amazon Mechanical Turk, showing page-load videos to users side-by-side. Surprisingly, they find that 71.5% of users experience the website to load faster without AdBlock Plus. However, it is unclear if controls were put in place so that both videos load and start

simultaneously, particularly if users have poor network connectivity, which may delay one video. Furthermore, users were not able to interact with the website (as they were shown a video), which may also affect user perception if a website has loaded. Considering that we have shown that AdBlock Plus is the worst performing extension among all eight extensions we analyzed for Mozilla Firefox across all metrics, and it is performing worse than Google Chrome without extensions, it is not surprising that users may experience pages to load slower with AdBlock Plus. Overall, we believe that their work is complementary to ours: We provide the first in-depth analysis of multiple privacy-focused extensions across two modern browsers with more detailed quality of experience metrics, like response sizes or processor time, while Newman et al. focus on browsing quality of experience for a single extension on Google Chrome 57 (end of life in April 2017) only.

Sundaresan et al. [38] evaluate how broadband network access in over 5,000 homes in the United States can affect page-load times for nine websites, and they found that latency improvements, such as improving slow DNS response times, can lead to pages loading substantially. They also identified that more aggressive DNS, TCP, and content caching–even if a browser already performs similar optimizations–can yield additional improvements. However, Sundaresan et al. do not utilize a full browser for their study and they do not investigate how browser extensions affect performance. As such, their work is complementary to ours. Notably, some of the extensions that we include in our study are performing some of the optimizations they proposed, like Decentraleyes.

Wang et al. [46] introduce WProf based on WebKit to profile how a browser interacts with a website and generate a dependency graph of resources that the browser requested, parsed, rendered, etc., effectively being a predecessor to the HAR format that we use (Section 3.3.1). Using WProf, they perform an analysis of the critical path to understand how the page-load time of 350 websites differs under various settings, namely end-user caching (similar to Decentraleyes), using the SPDY protocol (now deprecated, but which influenced the development of HTTP/2) instead of HTTP, and using the mod_pagespeed server extension. While they identify that end-user caching reduces the page-load time, most objects are not in the critical path and can be fetched in parallel, meaning that the reduction is not proportional. They also discover that script executions account for up to 35% of the critical path, the majority of which is synchronous JavaScript. However, it is unclear if the script executions are related to tracking or advertisements, that is, if they would be blocked by privacy-focused extensions and, thus, if these extensions would improve performance.

Related to Sundaresan et al. and Wang et al., Hounsel et al. [16] analyze how DNS transport protocols (UDP, Do53; TLS over TCP, DNS over TLS, DoT; and, DNS over HTTP/2 over TLS over TCP, DoH) affect the page-load time of a website for Mozilla Firefox under different network conditions, which they find can have a significant impact on page-load time (up to 0.5 seconds if network conditions are poor). Different from Hounsel et al., we do not investigate how DNS affects page-load time, but we analyze how extensions aiming to improve user privacy affect various performance metrics, including page-load time, but also including system performance for both Google Chrome and Mozilla Firefox. Their results indicate that better network connectivity leads to a smaller difference in page-load time between their measurement variables, supporting our interpretation of how network connectivity will generalize in how extensions affect performance.

5.2 Browser Extension Security & Privacy

A substantial amount of work has been carried out to better understand the security and privacy properties of browser extensions.

In this area, most closely related to our work is prior research by Merzdovnik et al. [22], who investigate the effectiveness of anti-tracking extensions across 100,000 websites in terms of being able to successfully prevent tracking. They also tangentially analyze how five anti-tracking extensions affect metrics similar to our system performance metrics, namely processor time and memory consumption [22, Section 5.5], but exact details about how memory consumption and processor time are measured are unclear [22, Section 4.1]. Overall, they find that anti-tracking extensions did not cause substantial processor overhead, but require additional memory. Contrary to their work, we do not analyze if trackers are successful in blocking tracking. Instead, we analyze in-depth how eight popular privacy-focused extensions affect browser performance as well as system performance, and we provide a holistic view on how these extensions affect user experience in terms of the differential in page-load time, requested resources, etc. compared to a plain browser. Moreover, we account for multiple potential issues that may affect generalizability of the results to end users, which Merzdvonik et al. do not account for [22, Section 5.1], such as not running multiple browser instances on the same machine and measuring from multiple vantage points.

Beyond work on understanding how privacy-focused extensions affect security and privacy, a multitude of prior work investigate how browser extensions themselves can subvert users' security and privacy. For example, Chen et al. [2] develop and use a taint analysis framework to study the privacy practices of browser extensions at scale. They discovered over 3,600 extensions potentially leaking privacy-sensitive information, with the ten most used ones having more than 60 million users. Similarly, Kapravelos et al. [18] use a dynamic analysis system to identify browser extensions behaving maliciously, putting more than 5.5 million affected users at risk. And, Starov et al. [34–36] and Trickel et al. [41] investigate how extensions allow users to be uniquely fingerprinted and possibly tracked, and how one can defend against it.

6 Conclusion

In this paper, we analyzed in-depth the performance cost and benefits of various privacy-focused browser extensions for the two most popular desktop browsers, Mozilla Firefox and Google Chrome, by looking at how they affect system-level performance metrics as well as browser performance measures compared to a browser without any extensions installed. We have found no evidence that privacy-focused browser extensions substantially negatively affect performance, neither for Google Chrome nor Mozilla Firefox, contradicting Google's claim that the functionality these extensions rely on is a performance concern that justifies severely restricting privacy-focused extensions, and limiting users' choice. To the contrary, albeit some extensions perform worse, we identified multiple extensions for which almost all metrics indicate that they would improve user experience, such as reducing overall page-load time (time to interactive), amount of data transfered, or processor time.

Future work should investigate if it is possible to quantify the wealth transfer that privacy-invasive techniques enable, like tracking; not only in terms of exposing sensitive data, but also in terms of computational cost, power usage, and users' inattention.

References

[1] R. Amadeo. 2014. Adware vendors buy Chrome Extensions to send ad- and malware-filled updates. (January 18, 2014). Retrieved 10/11/2019 from https://arstechnica.com/information-technology/2014/01/malware-vendors-buy-chrome-extensions-to-send-adware-filled-updates/.

[2] Q. Chen and A. Kapravelos. 2018. Mystique: Uncovering Information Leakage from Browser Extensions. In *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery (ACM), Toronto, ON, Canada, (October 2018). ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734.3243823.

[3] J. Davies. 2019. After GDPR, The New York Times cut off ad exchanges in Europe — and kept growing ad revenue. (January 16, 2019). Retrieved 10/11/2019 from https://digiday.com/media/gumgumtest-new-york-times-gdpr-cut-off-ad-exchanges-europe-ad-revenue/.

[4] Disconnect. [n. d.] Disconnect. Retrieved 10/11/2019 from https://disconnect.me/.

[5] Electronic Frontier Foundation (EFF). [n. d.] How does Privacy Badger work? Retrieved 10/11/2019 from https://www.eff.org/privacybadger/faq#How-does-Privacy-Badger-work.

[6] Electronic Frontier Foundation (EFF). [n. d.] HTTPS Everywhere. Retrieved 10/11/2019 from https://www.eff.org/https-everywhere.

[7] Electronic Frontier Foundation (EFF). [n. d.] Privacy Badger. Retrieved 10/11/2019 from https://www.eff.org/privacybadger/.

[8] J. Enberg. 2019. Digital Ad Spending 2019. (March 28, 2019). Retrieved 10/11/2019 from https://www.emarketer.com/content/global-digital-ad-spending-2019.

[9] European Parliament and the Council of the European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). (April 27, 2016). Retrieved 02/15/2019 from http://data.europa.eu/eli/reg/2016/679/oj.

[10] Eyeo GmbH. [n. d.] Adblock Plus – The world's # 1 free ad blocker. Retrieved 10/11/2019 from https://adblockplus.org/.

[11] Ghostery. [n. d.] Ghostery Makes the Web Cleaner, Faster, and Safer. Retrieved 10/11/2019 from https://www.ghostery.com/.

[12] Google. [n. d.] chrome.webRequest – Google Chrome. Retrieved 10/11/2019 from https://developer.chrome.com/extensions/webRequest.

[13] K. Hill. 2012. How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did. (February 16, 2012). Retrieved 10/11/2019 from https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/#34e325236668.

[14] R. Hill and Contributors. [n. d.] uBlock Origin - An efficient blocker for Chromium and Firefox. Fast and lean. Retrieved 10/11/2019 from https://github.com/gorhill/uBlock/.

[15] Pi-hole LLC. [n. d.] Pi-hole: A black hole for Internet advertisements. Retrieved 10/11/2019 from https://pi-hole.net/.

[16] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster. 2019. Analyzing the Costs (and Benefits) of DNS, DoT, and DoH for the Modern Web, (July 18, 2019). arXiv: 1907.08089 [cs.NI].

[17] [n. d.] Issue 706008: Extensions support in headless Chrome. Retrieved 10/11/2019 from https://bugs.chromium.org/p/chromium/issues/detail?id=706008.

[18] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson. 2014. Hulk: Eliciting Malicious Behavior in Browser Extensions. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*. USENIX Association, San Diego, CA, USA, (August 2014). ISBN: 978-1-931971-15-7. Retrieved 10/10/2019 from https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/kapravelos.

[19] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen. 2019. TRANCO: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS)*. Internet Society (ISOC), San Diego, CA, USA, (February 2019). ISBN: 1-891562-55-X. DOI: 10.14722/ndss.2019.23386.

[20] P. Lowe. [n. d.] Blocking with ad server and tracking server hostnames. Retrieved 10/11/2019 from https://pgl.yoyo.org/adservers/.

[21] G. Maone. [n. d.] NoScript – JavaScript/Java/Flash blocker for a safer Firefox experience! – what is it? – InformAction. Retrieved 10/11/2019 from https://noscript.net/.

[22] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl. 2017. Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools. In *Proceedings of the 2nd IEEE European Symposium on Security & Privacy (EuroS&P)*. Institute of Electrical and Electronics Engineers (IEEE), Paris, France, (July 2017). ISBN: 978-1-5090-5762-7. DOI: 10.1109/EuroSP.2017.26.

[23] Mozilla. [n. d.] webRequest - Mozilla – MDN. Retrieved 10/11/2019 from https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest.

[24] C. Neiman. 2019. Mozilla's Manifest v3 FAQ. Official Mozilla Add-ons Blog. (September 3, 2019). Retrieved 10/11/2019 from https://blog.mozilla.org/addons/2019/09/03/mozillas-manifest-v3-faq/.

[25] J. Newman and F. E. Bustamante. 2019. The Value of First Impressions: The Impact of Ad-Blocking on Web QoE. In *Proceedings of the 20th Passive and Active Measurement (PAM)* (Lecture Notes in Computer Science (LNCS)). Volume 11419. Springer International Publishing, Puerto Varas, Chile, (March 2019). ISBN: 978-3-030-15986-3. DOI: 10.1007/978-3-030-15986-3_18.

[26] J. Odvarko, P. Hedenskog, and M. Bertsch. [n. d.] firefox-devtools/har-export-trigger: Trigger HAR export any time directly from within a page. Retrieved 10/11/2019 from https://github.com/firefox-devtools/har-export-trigger.

[27] [n. d.] perf: Linux profiling with performance counters. Retrieved 10/11/2019 from https://perf.wiki.kernel.org/index.php/Main_Page.

[28] T. Poss. 2016. How Does Load Speed Affect Conversion Rate? (January 14, 2016). Retrieved 10/11/2019 from https://blogs.oracle.com/marketingcloud/how-does-load-speed-affect-conversion-rate.

[29] Privoxy Developers. [n. d.] Privoxy – Home Page. Retrieved 10/11/2019 from https://www.privoxy.org/.

[30] T. Rientjes. [n. d.] Decentraleyes – Local CDN Emulation. Retrieved 10/11/2019 from https://decentraleyes.org/.

[31] G. Ruan. 2014. I am one of the developers of a popular Chrome extension and we've been approached by malware companies that have tried to buy us. AMA! (self.IAmA). Reddit IAmA. (January 18, 2014). Retrieved 10/11/2019 from https://www.reddit.com/r/IAmA/comments/1vjj51/i_am_one_of_the_developers_of_a_popular_chrome/.

[32] [n. d.] Selenium – Web Browser Automation. Retrieved 10/11/2019 from https://docs.seleniumhq.org/.

[33] R. Shields. 2019. U.S. Digital Ad Spend Will Surpass Offline in 2019. (February 20, 2019). Retrieved 10/11/2019 from https://www.adweek.com/programmatic/u-s-digital-ad-spend-will-surpass-offline-in-2019/.

[34] O. Starov and N. Nikiforakis. 2017. XHOUND: Quantifying the Fingerprintability of Browser Extensions. In *Proceedings of the 38th IEEE Symposium on Security & Privacy (S&P)*. Institute of Electrical and Electronics Engineers (IEEE), San Jose, CA, USA, (May 2017). ISBN: 978-1-5090-5533-3. DOI: 10.1109/SP.2017.18.

[35] O. Starov, P. Laperdrix, A. Kapravelos, and N. Nikiforakis. 2019. Unnecessarily Identifiable: Quantifying the fingerprintability of browser extensions due to bloat. In *Proceedings of the The Web Conference (WWW) 2019*. International World Wide Web Conference Committee (IW3C2), San Francisco, CA, USA, (May 2019). ISBN: 978-1-4503-6674-8. DOI: 10.1145/3308558.3313458.

[36] O. Starov and N. Nikiforakis. 2017. Extended Tracking Powers: Measuring the Privacy Diffusion Enabled by Browser Extensions. In *Proceedings of the 26th World Wide Web Conference (WWW)*. International World Wide Web Conference Committee (IW3C2), Perth, Australia, (April 2017). ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052596.

[37] statcounter GlobalStats. [n. d.] Desktop Browser Market Share Worldwide – Sept 2018 - Sept 2019. Retrieved 10/11/2019 from https://gs.statcounter.com/browser-market-share/desktop/worldwide.

[38] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei. 2013. Measuring and Mitigating Web Performance Bottlenecks in Broadband Access Networks. In *Proceedings of the 2013 Internet Measurement Conference (IMC)*. Association for Computing Machinery (ACM), Barcelona, Spain, (October 2013). ISBN: 978-1-4503-1953-9. DOI: 10.1145/2504730.2504741.

[39] [n. d.] The Easy Subscriptions for Adblock, Adblock Plus and uBlock Origin. Retrieved 10/11/2019 from https://forums.lanik.us/.

[40] The People of the State of California. 2018. The California Consumer Privacy Act of 2018, Assembly Bill No. 375. (June 29, 2018). Retrieved 10/11/2019 from https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.

[41] E. Trickel, O. Starov, A. Kapravelos, N. Nikiforakis, and A. Doupé. 2019. Everyone is Different: Client-side Diversification for Defending Against Extension Fingerprinting. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*. USENIX Association, Santa Clara, CA, USA, (August 2019). Retrieved 10/10/2019 from https://www.usenix.org/conference/usenixsecurity19/presentation/trickel.

[42] B. Ur, P. G. Leon, L. F. Cranor, R. Shay, and Y. Wang. 2012. Smart, Useful, Scary, Creepy: Perceptions of Online Behavioral Advertising. Technical report CMU-CyLab-12-007. Carnegie Mellon University, (July 13, 2012). Retrieved 10/11/2019 from https://www.cylab.cmu.edu/_files/pdfs/tech_reports/CMUCyLab12007.pdf.

[43] S. Vincent. 2019. Web Request and Declarative Net Request: Explaining the impact on Extensions in Manifest V3. Developer Advocate for Chrome Extensions; Official Chromium Blog. (June 12, 2019). Retrieved 10/11/2019 from https://blog.chromium.org/2019/06/web-request-and-declarative-net-request.html.

[44] W3C Community Group. [n. d.] Browser Extensions. Retrieved 10/11/2019 from https://browserext.github.io/browserext/.

[45] W3C Web Performance Working Group. [n. d.] HTTP Archive (HAR) format. Retrieved 10/11/2019 from https://w3c.github.io/web-performance/specs/HAR/Overview.html.

[46] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. 2013. Demystifying Page Load Performance with WProf. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Lombard, IL, USA, (April 2013). ISBN: 978-1-931971-00-3. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/wang_xiao.

[47] D. P. Wiggins. [n. d.] Xvfb – virtual framebuffer X server for X Version 11. Retrieved 10/11/2019 from https://www.x.org/releases/X11R7.7/doc/man/man1/Xvfb.1.xhtml.

[48] S. Work. [n. d.] How Loading Time Affects Your Bottom Line. Retrieved 10/11/2019 from https://neilpatel.com/blog/loading-time.